
xivo-cc-doc Documentation

Release

Avencall

October 13, 2016

1	Table of Contents	3
1.1	Introduction	3
1.2	Installation and system configuration	3
1.2.1	CTI components	3
1.2.2	Reporting and statistics	15
1.2.3	Phone integration	25
1.2.4	Third Party Integration	27
1.3	Features	29
1.3.1	Contact center management	29
1.3.2	Agent environment	34
1.3.3	Configuration Management	36
1.4	Administration	37
1.4.1	Log	37
1.4.2	Backup	38
1.4.3	Restore	38
1.5	Xuc Xivo Unified Communication Framework	38
1.5.1	Developer	39
1.5.2	Javascript API	41
1.5.3	Rest API	58
1.5.4	Statistics	60
1.5.5	Technical structure of XiVO-CC	62
1.6	Troubleshooting	62
1.6.1	Xuc et Xuc_mgt - applications web ccmanager, agent et assistant	62
1.6.2	Application Configuration (xuc_rigths)	62
1.6.3	Recording	62
1.6.4	SpagoBI	62
1.6.5	Kibana	62
1.6.6	NGINX - proxy web	62
1.7	XiVO Centralized Interface	63
1.7.1	Installation	63
1.7.2	Web interface	64
1.7.3	REST API	71
2	Indices and tables	83



XiVO-CC is an application suite developed by [Avenacall](#) Group, and provides enhancements of the [XiVO](#) PBX contact center functionalities.

Table of Contents

1.1 Introduction

XiVO-CC provides enhancements of the XiVO PBX contact center functionalities. It gives especially access to outsourced statistics, real-time supervision screens, third-party CTI integration and recording facilities.

1.2 Installation and system configuration

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.

1.2.1 CTI components

In order for these components to be fully functional, some customizations need to be done on the XiVO: they will all be covered in a first part.

Overview

The following components will be installed :

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

Install from repository

There is a package *xivocc-installer* available in the repository which will configure XiVO PBX and install XiVO CC.

Warning: The Xivo is reconfigured during the installation and must be restarted, you may accept the automatic restart during the installation or you need to restart it manually later before starting the docker containers.

Install process

The install process from the repository consists of four parts:

- The first part is to manually add the Avencall repository. You may contact Avencall to get access.
- The second part is to manually run the prerequisites script to install docker and docker compose.
- The third part is the installation process itself.
- The fourth part is to install the package for the recording.

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

- When asked to generate a pair of authentication keys, leave the password field empty.
- Before copying the authentication keys, you will be prompted for the XiVO PBX root password.
- XiVO PBX must restart, the question will prompt you to restart during the process or to restart later.

Install Docker and Docker Compose

Download script which will install docker and docker compose.

```
wget https://gitlab.com/xivoxc/packaging/raw/2016.02/install/install-docker.sh -O install-docker.sh
chmod +x install-docker.sh
./install-docker.sh
```

Package Installation

Install the *xivocc-installer* package via *apt*. It is required to restart XiVO PBX during or after the setup process. The installer will ask whether you wish to restart XiVO PBX later.

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

Package for the recording

To use the recording feature, you must install on the XiVO PBX the debian package available in the repository.

```
apt-get install xivo-recording
```

After-install steps

After the successful installation, start docker containers by an alias which was added to `~/.bashrc`

```
source ~/.bashrc
dcomp -d up
```

If you selected to restart XiVO PBX later, please do so when possible to apply the modifications made by the installer. The XUC server will not be able to connect correctly to the database on XiVO PBX.

To restart XiVO services, on XiVO PBX server run


```
xivo-service restart all
```

Known Issues

To avoid problems when uninstalling, you should:

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

Prerequisites

We will assume your outsourced server meets the following requirements:

- OS : Debian 8 (jessie), 64 bit
- Docker installed
- Docker-compose 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 released version of docker and docker compose

We will make the following assumptions :

- the XiVO has the IP 192.168.0.1
- some data (incoming calls, internal calls etc.) might be available on XiVO (otherwise, you will not see *anything* in the *check-list* below).
- the server has the IP 192.168.0.2
- the latest version of Docker is installed
- the latest version of Docker-compose is installed
- the package xivo-recording is available on a custom Debian mirror. If this is not the case, you will need to skip the `apt-get install` commands and build the packages yourself.

Install ntp server

```
apt-get install ntp
```

XUC the server and the XiVO 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.

XiVO configuration

PostgreSQL configuration

Firstly, allow access to PostgreSQL from the outside. Edit `/etc/postgresql/9.1/main/postgresql.conf`:

```
listen_addresses = '*'
```

Add this line to `/etc/postgresql/9.1/main/pg_hba.conf`:

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

Create a user *stats* with read permissions :

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

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

AMI configuration

- Xivo < 15.18 Add a new user in `/etc/asterisk/manager.conf` with :
- Xivo >= 15.18 Add a file `xuc.conf` in `/etc/asterisk/manager.d` directory with :

```
[xuc]
secret = xucpass
deny=0.0.0.0/0.0.0.0
permit=X.X.X.0/255.255.255.0
read = system,call,log,verbose,command,agent,user,dtmf,originate,dialplan
write = system,call,log,verbose,command,agent,user,dtmf,originate,dialplan
```

Replace `X.X.X.0` by your xivocc network

And reload the AMI :

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

CEL Configuration

Add some events in the CEL. Edit `/etc/asterisk/cel.conf`:

- For Asterisk 11:

```
[general]
enable=yes
apps=dial,park,queue
events=APP_START,CHAN_START,CHAN_END,ANSWER,HANGUP,BRIDGE_START,BRIDGE_END,BRIDGE_UPDATE,USER_DEF

[manager]
enabled=yes
```

- For Asterisk 13:

```
[general]
enable = yes
apps = dial,park,queue
events = APP_START,CHAN_START,CHAN_END,ANSWER,HANGUP,BRIDGE_ENTER,BRIDGE_EXIT,USER_DEFINED,LINKED

[manager]
enabled = yes
```

and reload the cel module in Asterisk :

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

Customizations in the web interface

Create a user Xuc in *Services -> IPBX -> Users* with the following parameters:

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

Create a Web Services user in *Configuration -> Web Services Access* with the following parameters :

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

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

Phone integration

Do not forget to follow configuration steps detailed in [Required configuration for phone integration](#).

Packages for the recording

Still on the xivo, install the package which will handle the recording :

```
apt-get update
apt-get install xivo-recording
```

During the installation, you will be asked for :

- the recording server IP (i.e. 192.168.0.2)
- and the XiVO name (it **must** not contain any space or “-” character).

If you have several XiVO, you **must** give a different name to each of them.

This package has installed two dialplan sub-routines :

- xivo-incall-recording : used to record incoming calls
- xivo-outcall-recording : used to record outgoing calls

You have to manually place them where you want.

If you want to record on a gateway used with Xivo, you must not use the xivo-recording package but gateway-recording.

If you want to use call recording filtering, please install also:

```
apt-get install call-recording-filtering
```

During the installation, you will be asked for :

- the recording server address with protocol and port (i.e. <http://192.168.0.2:9400>)

XiVO CC

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

Retrieve the configuration script and launch it:

```
wget https://gitlab.com/xivoxc/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 address used afterwards for http URLs)

Create the following alias in your `.bashrc` file:

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

Xivo release <= 15.12 (asterisk 11)

Edit the `/etc/docker/compose/docker-xivocc.yml` and replace the image tag for **xuc** and **xivo_stats** with *latestast11*:

```
...
xivo_stats:
  image: xivoxc/xivo-full-stats:latestast11 <----- TO BE REPLACED -----
...

xuc:
  image: xivoxc/xuc:latestast11 <----- TO BE REPLACED -----
...

```

Xivo release > 15.12 (asterisk 13)

In `/etc/docker/compose/docker-xivocc.yml`, check that the image tag for **xuc** and **xivo_stats** is *latestast13*:

```
...
xivo_stats:
  image: xivoxc/xivo-full-stats:latestast13 <----- TO BE CHECKED -----
...

xuc:
  image: xivoxc/xuc:latestast13 <----- TO BE CHECKED -----
...

```

Xivo release = 16.03

In `/etc/docker/compose/docker-xivocc.yml`, check that the image tag for **xuc** is *latestxivo16*:

```
...
xuc:
  image: xivoxc/xuc:latestxivo16 <----- TO BE CHECKED -----
...

```

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_config_mgt_1	bin/config-mgt-docker	Up	0.0.0.0:9100->9000/tcp
xivocc_elasticsearch_1	/docker-entrypoint.sh elas ...	Up	0.0.0.0:9200->9200/tcp, 0.0.0.0:9300->9300/tcp
xivocc_fingerboard_1	/bin/sh -c /usr/bin/tail - ...	Up	
xivocc_kibana_volumes_1	/bin/sh -c /usr/bin/tail - ...	Up	
xivocc_nginx_1	nginx -g daemon off;	Up	443/tcp, 0.0.0.0:80->80/tcp
xivocc_pack_reporting_1	/bin/sh -c echo ...	Up	
xivocc_pgxivocc_1	/docker-entrypoint.sh postgres	Up	0.0.0.0:5443->5432/tcp
xivocc_postgresvols_1	/bin/bash	Exit 0	
xivocc_recording_server_1	bin/recording-server-docker	Up	0.0.0.0:9400->9000/tcp
xivocc_reporting_rsync_1	/usr/local/sbin/run-rsync.sh	Up	0.0.0.0:873->873/tcp
xivocc_spagobi_1	/bin/sh -c /root/start.sh	Up	0.0.0.0:9500->8080/tcp
xivocc_timezone_1	/bin/bash	Exit 0	
xivocc_xivo_replic_1	/usr/local/bin/start.sh /o ...	Up	
xivocc_xivo_stats_1	/usr/local/bin/start.sh /o ...	Up	
xivocc_xivocclogs_1	/bin/bash	Exit 0	
xivocc_xuc_1	bin/xuc_docker	Up	0.0.0.0:8090->9000/tcp
xivocc_xucmgt_1	bin/xucmgt_docker	Up	0.0.0.0:8070->9000/tcp

Checking Installed Version

Component version can be found in the log files, on the web pages for web components. You may also get the version from the docker container itself by typing :

```
docker exec -ti xivocc_xucmgt_1 cat /opt/docker/conf/appli.version
```

Change xivocc_xucmgt_1 by the component version you want to check

Using XivoCC

The various applications are available on the following addresses:

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



Post Installation

User Configuration

- Using the configuration manager : <http://192.168.0.2:9100/> (default user avencall/superpass) add a user to be able to use the recording interface with proper rights.

Note: Xuc server default user is xuc, add xuc as administrator to be able to get call history in web assistant.

SpagoBI

- Go to <http://192.168.0.2:9500/SpagoBI> (by default login: biadmin, password: biadmin)
- Update default language : go to “ Resources” > “Configuration management” > in the “Select Category” field, chose “LANGUAGE_SUPPORTED” and change value of the label “SPAGOB.LANGUAGE_SUPPORTED.LANGUAGE.default” in your language : fr,FR , en,US , ...
- Download the standard reports from https://gitlab.com/xivocc/sample_reports/raw/master/spagobi/standardreports.zip
- Import zip file in SpagoBI, all default options, with Jasper Report Engine as Engine associations.

XivoCC Default Report Sample Use the database status report to check if replication and reporting generation is working :

ACD outgoing calls

XivoCC 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:

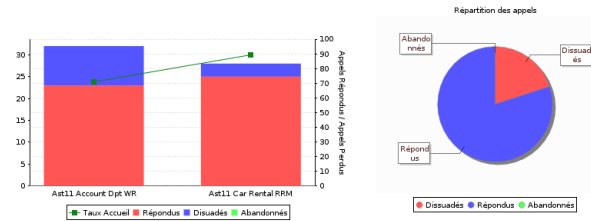
- You need to create an outgoing queue with a name starting with ‘out’, e.g. outgoing_queue.

Files d'attentes mensuelle 12

sauf samedi et dimanche

T1	15
T2	20

	Présentes	Répondus	TMC	Dissuadés	Raccrochés en file d'attente	Abandonnés en moins de T1	TMA	Transférés	Taux d'Accueil
Ast11 Account Dpt WR	32	23	00:02:49	9	0	0	00:00:05	0	71%
Ast11 Car Rental RRM	28	25	00:01:44	3	0	0	00:00:07	0	89%



Etat de la base des statistiques

Dernier CEL

Dernier Queue log

339050470 12/16/15 3:40 PM

id	time
47057935	2015-12-16 15:40:39.027702

Queue specific

Queue periodic

time	queue ref	nb offered
12/16/15 3:15 PM	sales	10

time	queue	total
12/16/15 3:15 PM	sales	10

Agent periodic

Agent specific

Agent queue specific

time	agent	login time
12/16/15 3:15 PM	1645	0 years 0 mons 0 days 0 hours 14

time	agent num	nb offered
12/16/15 3:15 PM	1564	2

time	agent num	queue ref
12/16/15 3:15 PM	1603	travels

Call data

Call on queue

start time	src num	dst num	status	uniqueid
12/16/15 3:40 PM	loadtester	84552	null	1450276822.4231
12/16/15 3:40 PM	loadtester	84557	answer	1450276801.4227
12/16/15 3:39 PM	loadtester	84554	answer	1450276795.4223

queue time	queue re	agent nu	status
12/16/15 3:40 PM	travels	null	null
12/16/15 3:40 PM	carrental	1563	answered

Objets dans la configuration

Agents	Queues	Agent groups	Extensions
173	19	8	420

- This queue must be configured with preprocess subroutine `xuc_outcall_acd`, without On-Hold Music (tab General), Ringing Time must be 0 and Ring instead of On-Hold Music must be activated (both tab Application).
- The subroutine must be deployed on the Xivo server (to `/etc/asterisk/extension_extra.d/` or through the web interface), the file is available from https://gitlab.com/xivoxc/xucserver/raw/master/xivo/outbound/xuc_outcall_acd.conf, with owner `asterisk:www-data` and rights `660`.
- You must also deploy the file https://gitlab.com/xivoxc/xucserver/raw/master/xivo/outbound/generate_outcall_skills.py to `/usr/local/sbin/`, with owner `root:root` and rights `755`.
- Furthermore, you must replace the file `/etc/asterisk/queueskills.conf` by the following one <https://gitlab.com/xivoxc/xucserver/raw/master/xivo/outbound/queueskills.conf> (be sure to backup the original one), without changing the owner or rights
- And finally you need to add a new skill rule on the Xivo server: Services -> Call center -> Skill rules -> Add, with name `'select_agent'` and rules `'$agent > 0'`.

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.

Totem Panels

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

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



Post Installation Check List

- All components are running : `dcomp ps`
- Xuc internal database is synchronized with xivo check status page with <http://xivoccserver:8090/>
- CCManager is running, log a user and check if you can see and manage queues : <http://xivoccserver:8070/ccmanager>
- Web agent is running, log an agent and check if you can change the status : <http://xivoccserver:8070/agent>
- Web assistant is running, and you get call history : <http://xivoccserver:8070/>
- Check database replication status using spagobi system report
- Check elasticsearch database status (totem panels) http://xivoccserver:9200/queuelogs/_status
- Check that you can listen to recordings <http://xivoccserver:9400/>

- Check totem panels <http://192.168.85.102/kibana>

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

Ldap Authentication

Xuc

Configure LDAP authent for CCmanager, Web Assistant and Web Agent

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 ldap configuration (empty by default)

```
include "application.conf"

authentication {
  ldap {
    managerDN = "uid=company,ou=people,dc=company,dc=com"      # user with read rights on the who
    managerPassword = "xxxxxxxxxx"                             # password for this user
    url = "ldap://ldap.company.com:389"                        # ldap URI
    searchBase = "ou=people,dc=company,dc=com"                 # ldap entry to use as search base
    userSearchFilter = "uid=%s"                                # filter to use to search users by
  }
}
```

Recreate the container : `dcomp up -d xuc`

webRTC

Xuc

For the moment available only on the sample page, pre-configured to be used on LAN without ICE for NAT traversal. 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. Direct and attended transfer can be performed using phone section methods. Hold and DTMF features are available via the webRTC API. Current implementation support just one simultaneous call.

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

Xivo

Awaiting integration of the additional SIP options to the Xivo's web interface you need to add manually webRTC peers to the `sip.conf` in the `/etc/asterisk/sip.conf` file, after the current content. You must also update `http` and `rtp` modules configuration.

- `http.conf` - asterisk's webserver must accept connection from outside, the listen address must be updated, for the sake of simplicity let's use 0.0.0.0, you can also pick an address of one of the network interfaces:

```
[general]
enabled=yes
bindaddr=0.0.0.0
bindport=5039
prefix=
tlsenable=yes
tlsbindaddr=127.0.0.1:5040
tlscertfile=/usr/share/xivo-certs/server.crt
tlsprivatekey=/usr/share/xivo-certs/server.key
servername=XiVO PBX
```

Do not forget to reload the configuration by the `module reload http` command on the Asterisk CLI.

- `rtp.conf` - the ICE support must be activated:

```
;
; RTP Configuration
;
[general]
;
; RTP start and RTP end configure start and end addresses
;
; Defaults are rtpstart=5000 and rtpend=31000
;
rtpstart=10000
rtpend=20000
;
; Whether to enable or disable UDP checksums on RTP traffic
;
;rtpchecksums=no
;
; The amount of time a DTMF digit with no 'end' marker should be
; allowed to continue (in 'samples', 1/8000 of a second)
;
;dtmftimeout=3000
icesupport=yes
stunaddr=stun.l.google.com:19302
```

The configuration is reloaded by `module reload res_rtp_asterisk.so`.

- `sip.conf` - You must 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 DTLS certificates, other steps are not necessary.
- *Configure the line of the webrtc user* - You must configure your user line as below, so that it is usable with the softphone WebRTC

General : Set Encryption to Yes

Signaling : Set codec to ulaw

Advanced : Set Transport to ws

Other parameter :

Set the following lines

The screenshot shows the 'Lines > Edit' configuration window. The 'General' tab is active. The 'Encryption' dropdown is highlighted with a red box and set to 'Yes'. Other fields include Username, Password, Context (Appels internes (default)), Language, NAT, and Verify new messages presence (RFC non-conformant). A 'Save' button is at the bottom.

```
avpf = yes
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
force_avp = yes
icesupport = yes
```

1.2.2 Reporting and statistics

Introduction

Pack reporting is a part of the XivoCC, but can also be installed separately. It aims at computing historical statistics, which are stored in the **xivo_stats** database. Sample reports based on them are accessible in **SpagoBI**.

Standalone installation

Warning: Full installation of the pack reporting requires restarting XiVO services, so telephone communications will be cut.

1. Install docker by following installation instructions: <http://docs.docker.com/installation/>
2. Execute the following commands:

```
wget https://gitlab.com/xivoxc/packaging/raw/master/install/install-docker-reporting.sh
bash install-docker-reporting.sh
docker-compose -f /etc/docker/compose/docker-reporting.yml up -d
```

During installation you will be asked for: * the XiVO IP address * the number of weeks to keep in history

At the end of the installation some configuration must be done on the XiVO:

1. edit `/var/lib/postgresql/9.1/main/postgresql.conf` and set `listen_addresses` to *
2. edit `/var/lib/postgresql/9.1/main/pg_hba.conf` and add the following line: **host asterisk stats PACK_REPORTING_IP/32 md5**
3. add the following events to `/etc/asterisk/cel.conf`: **HOLD,UNHOLD,BLINDTRANSFER,ATTENDEDTRANSFER**
4. execute the following command:

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

Lines > Edit

GeneralSignallingT38AdvancedOptional ParametersIPBX Infos

Events in the band RING:

DTMF:

Compensating for RFC 2833 DTMF from another IP PBX:

Monitoring:

RTP timeout:

RTP hold timeout:

RTP keepalive:

Allow transfer:

Auto framing:

Text support: ?

Video support:

Max call bitrate video:

Activate G726 no standard:

Minimum time of the round trip (RTT) messages: ?

Call setup timer: ?

Send «100 Trying» when register: ?

Ignore SDP packets version: ?

Session-timers mode: ?

Maximum session refresh interval: ?

Minimum session refresh interval: ?

Session refresher: ?

Use «Reason» header: ?

Send AOC-D/AOC-E to Snom devices: ?

Disallow SIP methods: ?

Max forward: ?

Codecs

Customize codecs: ☒

Codecs disallow:

1 items selectedRemove allAdd all

G.711 u-law (Audio) —

G.723.1 (Audio) +

GSM (Audio) +

G.711 A-law (Audio) +

ADPCM (Audio) +

16 bit Signed Linear PCM (Audio) +

LPC10 (Audio) +

G.729A (Audio) +

Save

16

Chapter 1. Table of Contents

Lines > Edit

General Signalling T38 **Advanced** Optional Parameters IPBX Infos

Caller ID: "webrtc test" <199>

Insecure: ☐

IP Addressing type: **Dynamic**

IP address or subnet mask allowed:

IP address or subnet mask denied:

Trusting the Remote-Party-ID: ☐

Send the Remote-Party-ID: ☐

Enable support subscriptions: ☐

Enable overlap: ☐

Support for 302 redirects: ☐

Add "user=phone" in the URI: ☐

Redirect media streams: ?

Rewriting the From field-User:

Rewriting the From field-Domain:

Client code:

Transport: **ws** ?

Enable calls counter: ☐ ?

Send BUSY up to num. calls: ☐ ?

Authorised contact addresses: ?

Rejected contact addresses: ?

Unsolicited mailbox: ?

Save

Lines > Edit

General Signalling T38 Advanced **Optional Parameters** IPBX Infos

Option	Value	
avpf	yes	?
dtlsenable	yes	?
dtlsverify	no	?
dtlscertfile	/etc/asterisk/keys/asterisk.p	?
dtlsprivatekey	/etc/asterisk/keys/asterisk.p	?
dtlssetup	actpass	?
force_avp	yes	?
icesupport	yes	?

Save

5. Finish the installation by a full restart of XiVO:

```
xivo-service restart all
```

Checking the installation

Historical statistics

- Docker containers *compose_xivo_replic_1*, *compose_xivo_stats_1* and *compose_pack_reporting_1* should be started
- There should be no errors in */var/log/xivocc/xivo-db-replication/xivo-db-replication.log* and */var/log/xivocc/xivo-full-stats/xivo-full-stats.log*

Kibana / TOTEM

- Data replication can take a long time, so you may need to be patient before finding data in the reports
- Some panels are preconfigured :
 - <http://@IP/kibana/#/dashboard/file/queues.json>
 - <http://@IP/kibana/#/dashboard/file/agents.json>
- To save these panels in Elasticsearch and make them accessible through Kibana menu, you will have to authenticate with **admin/Kibana**

Known limitations

- Queue members should only be agents. If users are members of a queue, their statistics will be incomplete.
- Configuration modifications on the XiVO (such as an agent deletion) are replicated on the statistics server, and their previous value is not kept. However, statistics history is preserved.
- POPC statistics are wrong.
- If two agents are associated to the same call, they will have the same hold time for this call.
- Transfer statistics limitation : given two queues Q1 and Q2, two agents A1 and A2, and an external caller C.
 - C calls Q1 and A1 answers
 - A1 transfers to Q2 and A2 answers
 - A2 transfers to the outside

Then the second transfer is seen as a transfer to the outside.

Attached Data

The pack 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

Upgrade notes

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
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
wget https://gitlab.com/xivoxc/packaging/raw/master/install/install-docker-reporting.sh
bash install-docker-reporting.sh
docker-compose -f /etc/docker/compose/docker-reporting.yml up -d pgxivocc
# Database migration. CHECK THE FREE DISK SPACE
sudo -u postgres pg_dump --format c xivo_stats | docker exec -i xivocc_pgxivocc_1 pg_restore -U p
docker-compose -f /etc/docker/compose/docker-reporting.yml 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-osp
docker exec -ti compose_pgxivocc_1 psql -U postgres -c 'CREATE EXTENSION IF NOT EXISTS "uuid-osp
```

Database schema

call_data

Calls list

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
status	status_type	Call status. Beware: only <i>answered</i> is properly filled.
ring_duration	INTEGER	Ring time of the endpoint answering the call, in seconds
transferred	BOOLEAN	True if the call has been transferred
call_direction	call_direction_type	Call direction (“incoming” : call from the outside, received by XiVO ; “outgoing” : call to the outside, originated by an endpoint associated to XiVO ; “internal” : call taking place entirely inside the XiVO)
src_num	VARCHAR	Calling number
transfer_direction	call_direction_type	Indicates the transfer direction, if relevant
src_agent	VARCHAR	Agent originating the call
dst_agent	VARCHAR	Agent receiving the call, if it is a direct call on an agent. Not filled when the call is destined to a queue
src_interface	VARCHAR	Interface originating the call (in the Asterisk sense, ex : SCCP/01234)

attached_data

Data attached to the call (cf. [Attached Data](#))

Column	Type	Description
id	INTEGER	
id_call_data	INTEGER	Id of the associated tuple in <i>call_data</i>
key	VARCHAR	Name of the attached data
value	VARCHAR	Value of the attached data

call_element

Part of a call matching the reaching of an endpoint

Column	Type	Description
id	INTEGER	
call_data_id	INTEGER	Id of the associated tuple in <i>call_data</i>
start_time	TIMESTAMP	Time at which the endpoint was called
answer_time	TIMESTAMP	Answer time for the endpoint
end_time	TIMESTAMP	End time of this call part
interface	VARCHAR	Endpoint interface

call_on_queue

Calls on a queue

Column	Type	Description
id	INTEGER	
callid	VARCHAR	Call unique reference, generated by Asterisk
queue_time	TIMESTAMP	Time of entrance in the queue
total_ring_seconds	INTEGER	Total ring time, in seconds (includes ringing of non-answered calls)
answer_time	TIMESTAMP	Answer time
hangup_time	TIMESTAMP	Hangup time
status	call_exit_type	Call status (<i>full</i> : full queue; <i>closed</i> : closed queue; <i>joinempty</i> : call arrived on empty queue; <i>leaveempty</i> : exit when queue becomes empty; <i>divert_ca_ratio</i> : call redirected because the ratio waiting calls/agents was exceeded ; <i>divert_waittime</i> : call redirected because estimated waiting time was exceeded; <i>answered</i> : call answered ; <i>abandoned</i> : call abandoned; <i>timeout</i> : maximum waiting time exceeded)
queue_ref	VARCHAR	Technical queue name
agent_num	VARCHAR	Number of the agent taking the call, if relevant

hold_periods

Hold periods

Column	Type	Description
id	INTEGER	
linkedid	VARCHAR	Call unique reference, generated by Asterisk
start	TIMESTAMP	Hold start time
end	TIMESTAMP	Hold end time

stat_queue_periodic

Statistics aggregated by queue and time interval (15 minutes)

Column	Type	Description
id	INTEGER	
time	TIMESTAMP	Start time of the considered interval
queue	VARCHAR	Queue technical name
answered	INTEGER	Number of answered calls
abandoned	INTEGER	Number of abandoned calls
total	INTEGER	Total number of calls received on the queue (which excludes the calls dissuaded before entering the queue)
full	INTEGER	Number of calls arrived on a full queue (diversion before entering the queue)
closed	INTEGER	Number of calls arrived on a closed queue, outsidied 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
di-vert_ca_ratio	INTEGER	Number of calls arrived when the calls / available agents ratio is exceeded (diversion before entering the queue)
di-vert_waittime	INTEGER	Number of calls arrived when the estimated waiting time is exceeded (diversion before entering the queue)
timeout	INTEGER	Nombre of calls redirecting because maximum waiting time is exceeded

stat_agent_periodic

Statistics aggregated by agent and time interval (15 minutes)

Column	Type	Description
id	INTEGER	
time	TIMESTAMP	Start time of the considered interval
agent	VARCHAR	Agent number
login_time	INTERVAL	Login time
pause_time	INTERVAL	Pause time
wrapup_time	INTERVAL	Wrapup 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	Technicxal 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 wrapup 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)

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, in seconds
ringing_time	INTEGER	Ring time on incoming calls from a queue, 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 calls from a queue, 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_transferred_intern	INTEGER	Number of calls coming from a queue and transferred to an internal destination
nb_transferred_extern	INTEGER	Number of calls coming from a queue and transferred 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	Technicxal 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	Wrapup time, in seconds

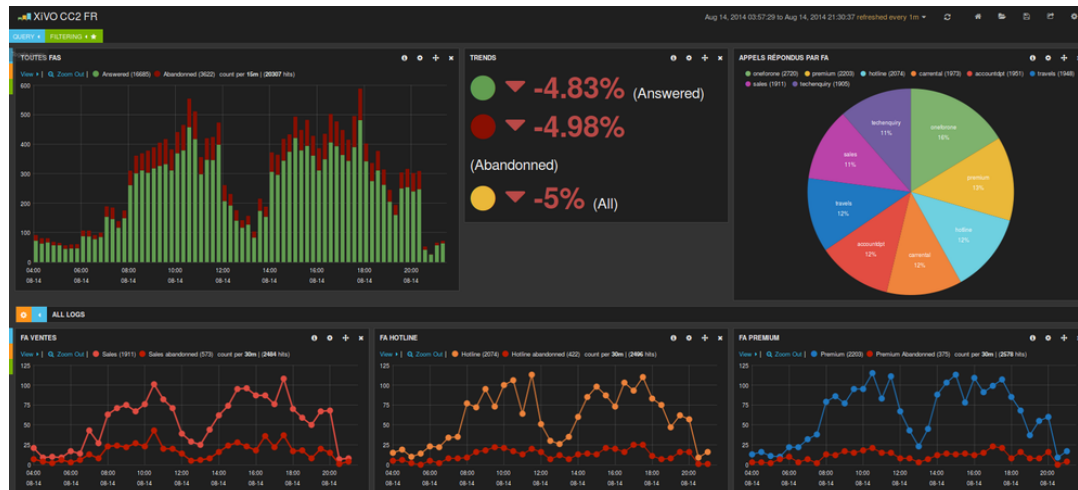
Tables **call_data**, **call_on_queue** et **hold_periods** can be linked together by doing a join on a column holding the call reference. The columns are the following:

Table	Reference column
call_data	uniqueid
call_on_queue	callid
hold_periods	linkedid

D'autre part, les tables **attached_data** et **call_element** contiennent une clef étrangère référençant la colonne **id** de **call_data**.

Using Kibana

Kibana is a web tool used to compute statistics based on Elasticsearch content. The reports packaged with the Pack reporting give you an outline of your recent call center activity. Here is a Kibana sample panel:



Graphs are based on the `queue_log` table, enriched with agent names and agent groups, and inserted into an Elasticsearch index. It contains events about calls placed on queues, and events about agent presences.

For each entry in the `queue_log` index, the following attributes are available:

- `queuedisplayname` : Queue display name
- `data1`: basic `queue_log` data, with a different meaning according to the event
- `callid` : Call unique identifier, generated by Asterisk
- `event` : Call or agent status event - please see below
- `agentnumber`: Agent number

- queueName : Technical queue name
- groupName : Agent group name
- queueTime: Time of the event
- agentName : Name of the agent, if available

The event can be one of the following (for a detailed explanation, please refer to <https://wiki.asterisk.org/wiki/display/AST/Queue+Logs>):

- Call events:
 - FULL
 - CONNECT
 - EXITEMPTY
 - CLOSED
 - EXITWITHTIMEOUT
 - JOINEMPTY
 - ABANDON
 - ENTERQUEUE
 - TRANSFER
 - COMPLETEAGENT
 - COMPLETECALLER
 - RINGNOANSWER
- Agent or queue event:
 - ADDMEMBER
 - PAUSEALL
 - PAUSE
 - WRAPUPSTART
 - UNPAUSE
 - UNPAUSEALL
 - PENALTY
 - CONFIGRELOAD
 - AGENTCALLBACKLOGIN
 - AGENTCALLBACKLOGOFF
 - REMOVEMEMBER
 - PRESENCE
 - QUEUESTART

1.2.3 Phone integration

XUC based web applications like agent interface or xivo client web 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

Supported phones

Manufacturer	Function					
	Answer	Hangup	Hold	Conference	Attended Transfer	Direct Transfer
Snom 7XX	OK	OK	OK	OK	OK	OK
Polycom VVX	OK	OK	OK	NO	OK	OK
Yealink T4X	OK	OK	OK	NO	OK	OK

- NO - Not available

Required configuration

Customize templates for Polycom phones

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 `apps.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"
```

Customize templates for Yealink phones

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**)

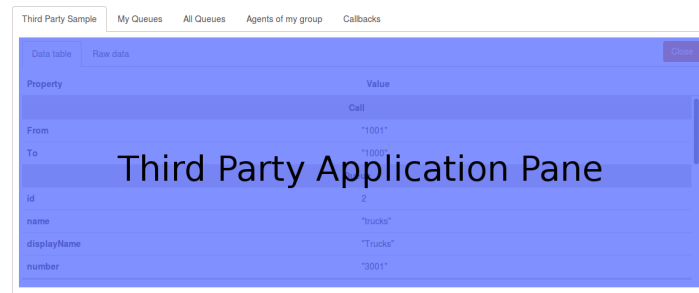
```
{% if XX_options['switchboard'] -%}
push_xml.sip_notify = 1
call_waiting.enable = 0
{% else -%}
push_xml.sip_notify = 1
call_waiting.enable = 1
{% endif %}
```

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()'`

1.2.4 Third Party Integration

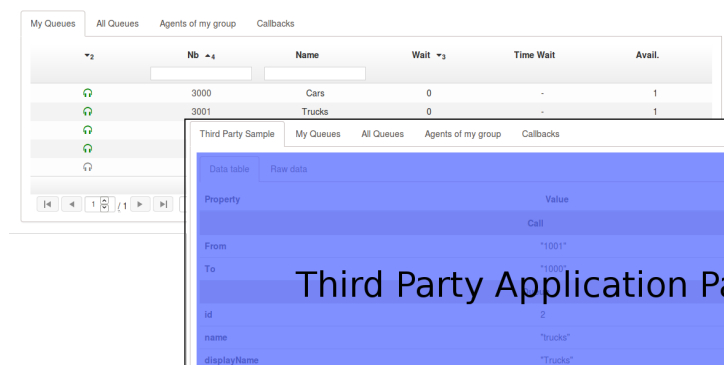
Third party web application integration is possible inside the XucMgt Agent application since XucMgt version 1.49.0. Upon each call, you can display a custom tab inside the agent interface:



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 specified action is to open the application when the call is hung up, a new tab 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 Agent interface. This can be done by specifying a `THIRD_PARTY_URL` environment variables.

For example, inside the dockerfile, in the XucMgt section:

```
environment:
...
- THIRD_PARTY_URL=http://some.url.com/ws/endpoint
```

The specified URL must be accessible from the client browser (i.e. the end user of the Agent application). The call will be made from his browser.

Web Service API

The Web Service url specified in the [Configuration](#) must conform to the following behaviour.

The service will receive a POST request with a payload as `application/json`, for example:

```
{
  "user": {
    "userId": 4,
    "agentId": 1,
    "firstName": "James",
    "lastName": "Bond",
    "fullName": "James Bond"
  },
  "callee": "1000",
  "caller": "1001",
  "queue": {
    "id": 2,
    "name": "trucks",
    "displayName": "Trucks",
    "number": "3001"
  },
  "userData": {
    "XIVO_CONTEXT": "default",
    "XIVO_USERID": "2",
    "XIVO_SRCNUM": "1001",
    "XIVO_DSTNUM": "3001"
  }
}
```

- `user` contains the connected user information
- `callee` contains the number called
- `queue` queue properties
- `userData` call data presented by Xivo

The Web service must answer with an `application/json` content. For example:

```
{
  "action": "open",
  "event": "EventReleased",
  "url": "/thirdparty/open/6bd37819-b4a6-43d3-8fa3-6eb6489bb705",
  "autopause": true,
  "title": "Third Party Sample"
}
```

or:

```
{
  "action": "none"
}
```

- `action` is one of "open" or "none"
- `event` is one of "EventRinging", "EventEstablished", "EventReleased". The third party application will be opened when one the specified event occurs
- `url` should be the url to open inside the application. This url should point to a valid web application that can be specific for each call.
- `autopause` if set to true, the agent will be put on pause when the application pane is opened and back to ready when the application is completed.
- `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. Your 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:

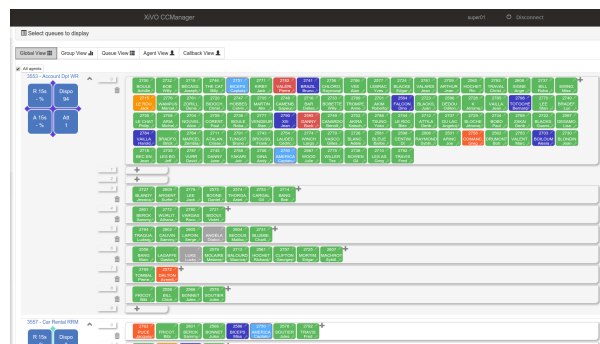
```
(function () {
    function close() {
        parent.window.postMessage("closeThirdParty", "*");
    }

    document.getElementById("close").addEventListener("click", close, false);
})();
```

1.3 Features

1.3.1 Contact center management

Introduction



CCmanager is a web application to manage and supervise a contact center

- Display queues
- Display agents / agents status
- Move or add agents in queue / penalty
- Move of add group of agents in queue / penalty
- **Action on agents**
 - Login / Logout
 - Pause / Unpause
 - Listen ¹

¹ Only supervisors which have their own lines can listen to agents, no supported on mobile supervisors, a line has to be affected to supervisors in xivo

Start the application : <http://<xuc:port>/ccmanager>

Single Agent Edition

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

This interface allows a user to change queue assignment 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”.

Notes

- 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 whitout default configuration remain unchanged.

Global View Group View Queue View Agent View Callback View

Click to toggle selection Click to edit selection

	Nb	First Name
<input type="checkbox"/>		
▼ a_very_long_group_name (1)		
<input checked="" type="checkbox"/>	2500	Brucé
▼ bingba3nguh (2)		
<input checked="" type="checkbox"/>	31000	Francois
<input type="checkbox"/>	123456	Isaac
▼ boats (1)		
<input checked="" type="checkbox"/>	2018	Irène

Update agents

Blanc Sec (1626) Didascaux (1599) Vee (1686)

Add in queues

Number	Name	Penalty
--------	------	---------

Remove from queues

Number	Name
--------	------

Queue

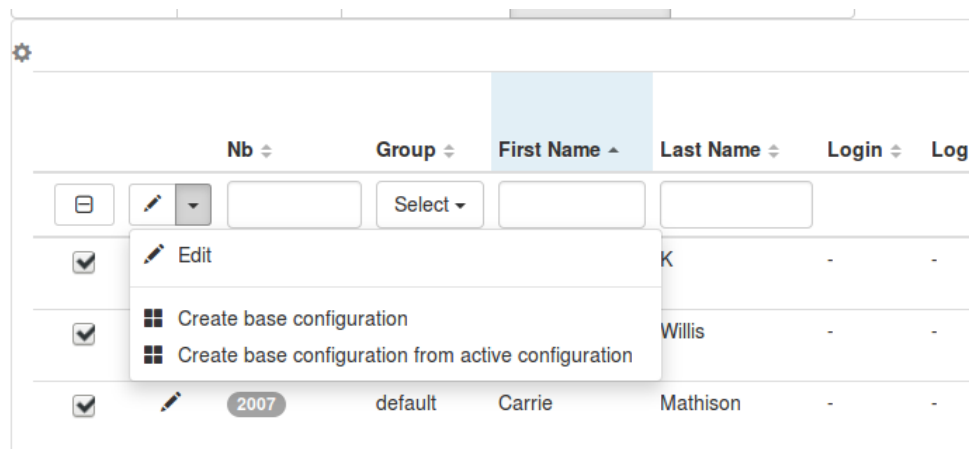
Penalty

Apply default configuration

Save Cancel

Create base configuration for a set of agents

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 screenshot shows a 'Create base configuration' popup window. It displays the names and IDs of the selected agents: K (2006), Mathison (2007), and Willis (2001). Below this is a table with columns: Number, Name, and Penalty. The table contains four rows of queue data. At the bottom, there are fields for Queue and Penalty, and buttons for Save and cancel.

Number	Name	Penalty
3000	Cars	1
3001	Trucks	3
3003	Bikes	1
3006	Boats	4

Queue: Penalty:

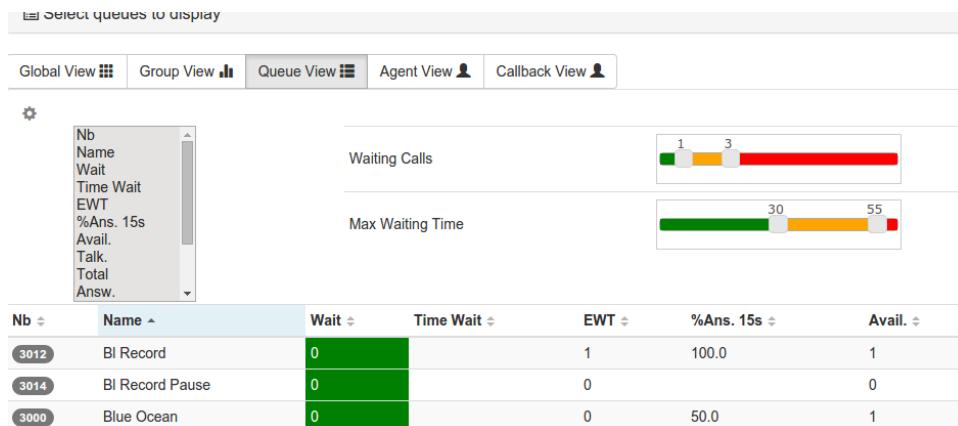
The queue table display the following columns:

- “Number”: The queue number
- “Name”: The queue name
- “Penalty”: The active penalty for the corresponding queue

Click on the plus button to add a queue for selection. Once your configuration is complete, click on save button to apply your configuration change.

Thresholds

Color thresholds can be defines for the waiting calls counter and the maximum waiting time counter



Applies to the queue view and the global view

Callbacks

This view allows to manage callback request : importing a new list of callbacks, monitoring them and downloading the associated tickets.

Parcourir... Aucun fichier sélectionné.				
Send Download tickets				
Sample list - Wisconsin				
Phone numbers	Full name	Company	Description	Taken by
1003 -				
0587963214 - 0789654123	Alice O'Neill	YourSociety		
0230210092 - 0689746321	John Doe	MyCompany	Call back quickly	
1003 -				

Callbacks can be imported from a CSV file into a *callback list*. 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
```

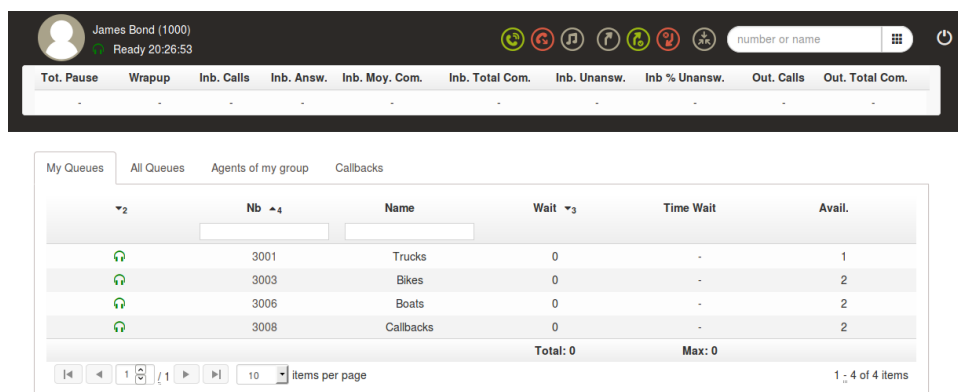
The header line must contain the exact field named described below:

- 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 name (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. The tickets of the processed callbacks can be downloaded by clicking on the `Download tickets` button.

1.3.2 Agent environment



The screenshot shows the agent interface for James Bond (1000). At the top, there's a header with the agent's name, status (Ready 20:26:53), and various icons. Below the header is a table with columns: Tot. Pause, Wrapup, Inb. Calls, Inb. Answ., Inb. Moy. Com., Inb. Total Com., Inb. Unansw., Inb % Unansw., Out. Calls, and Out. Total Com. Below this is a section for 'My Queues' with tabs for All Queues, Agents of my group, and Callbacks. The 'All Queues' tab is active, showing a table with columns: Nb, Name, Wait, Time Wait, and Avail. The table lists four items: 3001 Trucks, 3003 Bikes, 3006 Boats, and 3008 Callbacks. At the bottom, there's a pagination bar showing 10 items per page and 1 of 4 items.

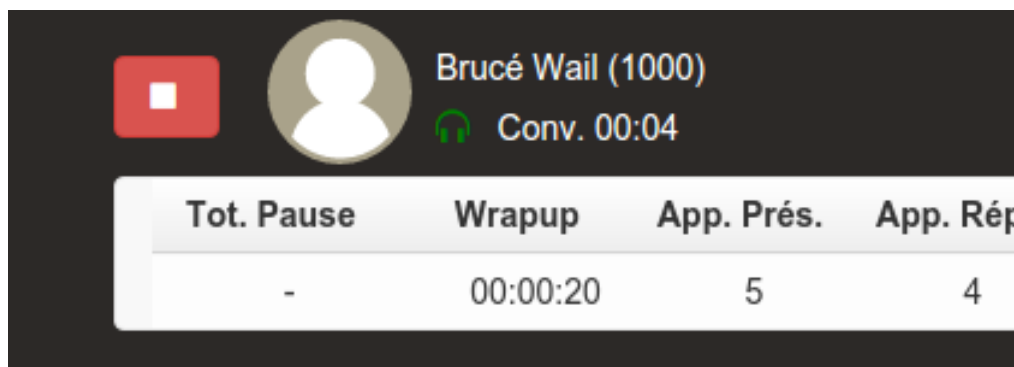
Tot. Pause	Wrapup	Inb. Calls	Inb. Answ.	Inb. Moy. Com.	Inb. Total Com.	Inb. Unansw.	Inb % Unansw.	Out. Calls	Out. Total Com.
-	-	-	-	-	-	-	-	-	-

Nb	Name	Wait	Time Wait	Avail.
3001	Trucks	0	-	1
3003	Bikes	0	-	2
3006	Boats	0	-	2
3008	Callbacks	0	-	2
Total: 0		Max: 0		

Web application for contact center agents

Configuration

Recording can be paused or started by an agent, this feature can be disabled by changing `showRecordingControls` option in `application.conf`, you can also set the environment variable `SHOW_RECORDING_CONTROLS` to false for your `xucmgt` container in docker compose yml file. When disabled the recording status is not displayed any more



The screenshot shows the agent interface for Brucé Wail (1000). At the top, there's a header with the agent's name, status (Conv. 00:04), and a red recording icon. Below the header is a table with columns: Tot. Pause, Wrapup, App. Prés., and App. Rép. The table shows values: Tot. Pause is -, Wrapup is 00:00:20, App. Prés. is 5, and App. Rép. is 4.

Tot. Pause	Wrapup	App. Prés.	App. Rép.
-	00:00:20	5	4

Callbacks panel can be removed using by changing `showCallbacks` option in `application.conf`, you can also use `SHOW_CALLBACKS` environment variable in docker compose yml file.

By using the `showQueueControls` option in `application.conf`, you may allow an agent to enter or leave a queue. You can also use `SHOW_QUEUE_CONTROLS` environment variable in docker compose yml file.

My Queues

All Queues

Agents of my group

Callbacks

▼ ₂	Nb ▲ ₄	Name	Wait ▼ ₃	Time Wait	Avail.
	3001	Trucks	0	-	1
	3003	Bikes	0	-	2
	3006	Boats	0	-	2
	3008	Callbacks	0	-	2
			Total: 0	Max: 0	

◀

1

⌂

1

▶

⏪

10 items per page

1 - 4 of 4 items

Taking Callbacks

The agent can see the *callbacks* related to the queues he is logged on. They are available in the *Callbacks* tab, beside the *Agents of my group* tab.

On this page, the agent only has access to basic information about the callback: the phone number to call, the person's name and its company name. On the left of each callback line, a colored clock indicates the temporal status of this callback:

- yellow if the callback is to be processed later
- green if we are currently inside the callback period
- red if the callback period is over

My Queues	All Queues	Agents of my group	Callbacks
Period ▲ ₃	Phone	Name	Queue
12-08 Après-midi	1002	Jack Bauer	Callbacks
13-08 Après-midi	1002	Jack Bauer	Callbacks
04-09 Matin	1001	James Bond	Callbacks
15-09 Après-midi	1002	Jack Bauer	Callbacks
12-10 Toute la journée	1002	Jason Bourne	Callbacks
10 items per page			
			1 - 5 of 5 items

To process one of these callbacks, the agent must click on one of the callbacks line. This will remove the callback from the other agents' list, and trigger the following screen:

My Queues
All Queues
Agents of my group
Callbacks

Callbacks - Jack Bauer - The Company

Requested on 12-08 Après-midi (14:00:00 17:00:00)

Callback

1002

1003

Description

Some description

Status

Comment

You must callback by clicking on the phone number before saving

Cancel

Save

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 ‘Callback’ as status, the callback can be rescheduled at a later time and another period:

Clicking on the calendar icon next to the “New due date” field, will popup a calendar to select another callback date.

Screen Popup

It is possible to display customer information in an external web application using Xivo [sheet](#) mecanism.

You must define a sheet with two fields

- **folderNumber** have to be defined. Can be calculated or use a default value not equal to “-“
- **popupUrl** The url to open when call arrives : i.e. <http://mycrm.com/customerInfo?folder=> the folder number will be automatically appended to the end of the URL

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

1.3.3 Configuration Management

Callbacks

The callback system in XivoCC aims at performing outgoing calls to specific numbers, to which some information can be associated such as a description ar 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.

Listes de rappel		Périodes de rappel
		
Nom	File d'attente	Nombre de rappels
Liste de test	Wisconsin (wisconsin)	3
		

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.

Listes de rappel		Périodes de rappel
		
Nom	Heure de début	Heure de fin
Après-midi	14:00:00	17:00:00
Matin	08:00:00	12:10:00
Toute la journée	07:00:00	17:00:00
		✓ 

1.4 Administration

1.4.1 Log

The log of each components 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
-- specific-stats.log
-- xivo-db-replication.log
-- xivo-full-stats.log
  -- recording-server
|  -- downloads.log
|  -- downloads.log
|  -- recording-server.log
-- xuc
|  -- xuc.log
-- xucmgt
  -- xucmgt.log
```

1.4.2 Backup

You may backup your statistic database by using a similar command as below

```
docker run --rm --link demo_pgxivocc_1:db -v $(pwd):/backup -e PGPASSWORD=xivocc postgres pg_dump
```

1.4.3 Restore

You may restore a backup using a similar command (to be adapted)

```
docker run --rm -it --link pgxivoccdemo_pgxivocc_1:db -v $(pwd):/backup postgres pg_restore -h db
```

1.5 Xuc Xivo Unified Communication Framework

Xuc is an application suite developed by [Avencall](#) Group, based on several free existing components including [XiVO](#), and our own developments to provide communication services api and application to businesses. Xuc is build on [Play](#) using intensively [Akka](#) and written in [Scala](#)

[XiVO](#) is [free software](#). Most of its distinctive components, and Xuc as a whole, are distributed under the *LGPLv3 license*.

Xuc is providing

- Javascript API
- Web services
- Sample application
- Simple agent application
- Simple unified communication application pratix
- Contact center supervision
- Contact center statistics

The proposed applications are available in English and French. The list of preferred langs sent by the browser is analyzed and the first known lang is used, so if the browser requests it, en and fr the page will be server in en. The fallback language is French. Contributions are welcome, start with opening an issue on gitlab project page.

Xuc is composed of 3 modules

- The server module
- The core module
- The statistic module.

1.5.1 Developer

Building and packaging

Dependencies

- **Xivo Java Cti lib** ; <https://gitorious.org/xivo/xivo-javactilib>
 - mvn install
- **theatrus/akka-quartz** [<https://github.com/theatrus/akka-quartz>]
 - sbt publish-local

(sudo apt-get install devscripts)

Update change log

- dch -i in project root directory, parent of debian/changelog
- edit changelog to add version

Update documentation site

- update src/sphinx/conf.py with new version
- activator make-site
- copy target/sphinx/docs content to public/doc

Update xuc_logger.xml with new version

- Create debian package : activator debian:genChanges

Docker

Building docker image:

```
activator docker:publish
or
activator docker:publishLocal
docker tag xivo/xuc:2.4.32 xivo/xuc:latest
activator clean test docker:publishLocal; docker tag -f xivo/xuc:1.9.0 xivo/xuc:latest; docker push
```

Documentation Guidelines

The Xuc documentation uses [reStructuredText](#) as its markup language and is built using [Sphinx](#).

Sphinx

For more details see [The Sphinx Documentation](#)

reStructuredText

For more details see [The reST Quickref](#)

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

Sections

Section headings are very flexible in reST. We use the following convention in the Xuc documentation:

- # (over and under) for module headings
- = for sections
- – for subsections
- ^ for subsubsections
- ~ for subsubsubsections

Cross-referencing

Sections that may be cross-referenced across the documentation should be marked with a reference. To mark a section use `.. _ref-name:` before the section heading. The section can then be linked with `:ref: `ref-name``. These are unique references across the entire documentation.

For example:

```
.. _xuc-module:

#####
Xuc Module
#####

This is the module documentation.

.. _xuc-section:

Xuc Section
=====

Xuc Subsection
-----

Here is a reference to "xuc section": :ref:`xuc-section` which will have the
name "Xuc Section".
```

Build the documentation First install [Sphinx](#). See below.

Building

For the html and pdf version of the docs:

```
activator make-site

open <project-dir>/target/sphinx/docs/index.html
open <project-dir>/target/sphinx/docs/Xuc-doc.pdf
```

Installing Sphinx and other tools

To be able to generate pdf and documentation you need install Sphinx and other tools:

```
sudo easy_install -U Sphinx
sudo apt-get install texlive-latex-base texlive-latex-recommended texlive-latex-extra texlive-fonts-recommended texlive-fonts-extra
```

1.5.2 Javascript API

Introduction

The Xuc javascript API enables you to integrate enterprise communication functions to your business application. It exposes Cti functions using javascript methods calls.

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](#) ...)

Integration Principles

- Include the Cti and Callback javascript API from the Xuc Server

```
<script src="http://<xucserver>:<xucport>/assets/javascripts/shotgun.js" type="text/javascript"></script>
<script src="http://<xucserver>:<xucport>/assets/javascripts/cti.js" type="text/javascript"></script>
<script src="http://<xucserver>:<xucport>/assets/javascripts/callback.js" type="text/javascript"></script>
<script src="http://<xucserver>:<xucport>/assets/javascripts/membership.js" type="text/javascript"></script>
```

- Include also the xc_webrtc and SIPml5 javascript APIs for the webRTC support:

```
<script src="http://<xucserver>:<xucport>/assets/javascripts/xc_webrtc.js" type="text/javascript"></script>
<script src="http://<xucserver>:<xucport>/assets/javascripts/SIPml-api.js" type="text/javascript"></script>
```

- Connect to the Xuc server using XiVO client username and password

```
var wsurl = "ws://" + server + "/ctichannel?username=" + username + "&agentNumber=" + phoneNumber + "&";
Cti.WebSocket.init(wsurl, username, phoneNumber);
```

- Setup event handlers to be notified on

- Phone state changes
- Agent state changes
- Statistics
- ...

- Eventually also webRTC handlers

- general
- register
- incoming
- outgoing

- Once web socket communication is established you are able to call XuC Cti javascript methods.

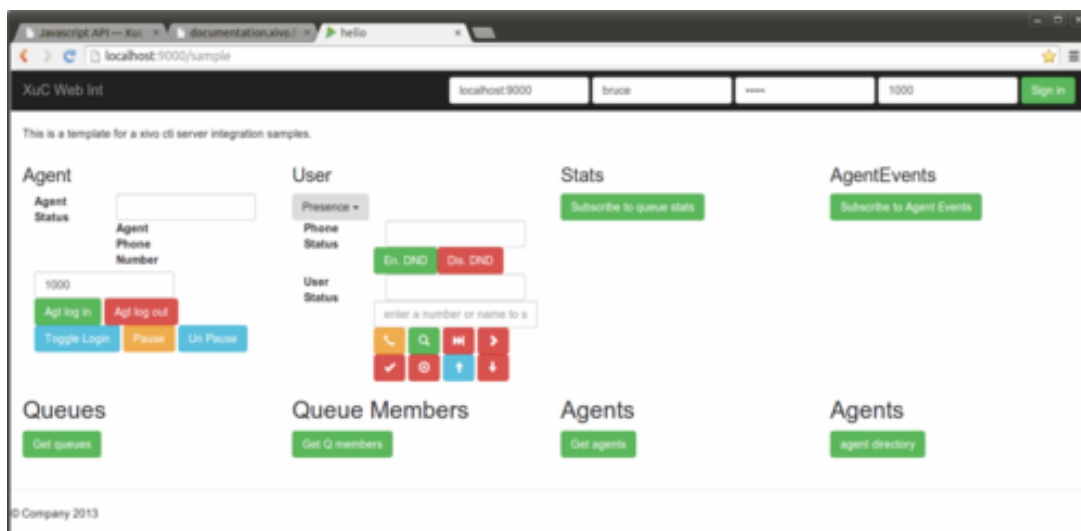
– Place a call, log an agent

```
...
$('#login_btn').click(function(event) {
    Cti.loginAgent($('#agentPhoneNumber').val());
});
$('#logout_btn').click(function(event) {
    Cti.logoutAgent();
});
$('#xuc_dial_btn').click(function(event) {
    Cti.dial($('#xuc_destination').val());
});
...
```

Sample Application

A sample application is provided by the XuC server. This application allows to display events and using different methods exposed by the XuC

`http://<sucserver>:<xucport>/sample`



You may browse and use the `sample.js` javascript file as an example

- Calling Cti methods :

```
$('#xuc_login_btn').click(function(event) {
    Cti.loginAgent($('#xuc_agentPhoneNumber').val());
});

$('#xuc_logout_btn').click(function(event) {
    Cti.logoutAgent();
});
$('#xuc_pause_btn').click(function(event) {
    Cti.pauseAgent();
});
$('#xuc_unpause_btn').click(function(event) {
    Cti.unpauseAgent();
});
$('#xuc_subscribe_to_queue_stats_btn').click(function(event) {
    Cti.subscribeToQueueStats();
});
$('#xuc_answer_btn').click(function(event) {
    Cti.answer();
});
```

```

$('#xuc_hangup_btn').click(function(event) {
    Cti.hangup();
});
$('#xuc_login_btn').click(function(event) {
    Cti.loginAgent($('#xuc_agentPhoneNumber').val());
});
$('#xuc_logout_btn').click(function(event) {
    Cti.logoutAgent();
});
$('#xuc_togglelogin_btn').click(function(event) {
    Cti.toggleAgentLogin();
});
$('#xuc_pause_btn').click(function(event) {
    Cti.pauseAgent();
});
$('#xuc_unpause_btn').click(function(event) {
    Cti.unpauseAgent();
});
$('#xuc_subscribe_to_queue_stats_btn').click(function(event) {
    Cti.subscribeToQueueStats();
});
$('#xuc_answer_btn').click(function(event) {
    Cti.answer();
});
$('#xuc_hangup_btn').click(function(event) {
    Cti.hangup();
});
$('#xuc_get_agent_call_history').click(function() {
    Cti.getAgentCallHistory(7);
});
$('#xuc_get_user_call_history').click(function() {
    Cti.getUserCallHistory(7);
});
});
.....

```

- Declaring events handlers :

```

Cti.setHandler(Cti.MessageType.USERSTATUSES, usersStatusesHandler);
Cti.setHandler(Cti.MessageType.USERSTATUSUPDATE, userStatusHandler);
Cti.setHandler(Cti.MessageType.USERCONFIGUPDATE, userConfigHandler);
Cti.setHandler(Cti.MessageType.LOGGEDON, loggedOnHandler);
Cti.setHandler(Cti.MessageType.PHONESTATUSUPDATE, phoneStatusHandler);
Cti.setHandler(Cti.MessageType.VOICEMAILSTATUSUPDATE, voiceMailStatusHandler);
Cti.setHandler(Cti.MessageType.LINKSTATUSUPDATE, linkStatusHandler);
Cti.setHandler(Cti.MessageType.QUEUESTATISTICS, queueStatisticsHandler);
Cti.setHandler(Cti.MessageType.QUEUECONFIG, queueConfigHandler);
Cti.setHandler(Cti.MessageType.QUEUELIST, queueConfigHandler);
Cti.setHandler(Cti.MessageType.QUEUEMEMBER, queueMemberHandler);
Cti.setHandler(Cti.MessageType.QUEUEMEMBERLIST, queueMemberHandler);
Cti.setHandler(Cti.MessageType.DIRECTORYRESULT, directoryResultHandler);

Cti.setHandler(Cti.MessageType.AGENTCONFIG, agentConfigHandler);
Cti.setHandler(Cti.MessageType.AGENTLIST, agentConfigHandler);
Cti.setHandler(Cti.MessageType.AGENTGROUPLIST, agentGroupConfigHandler);
Cti.setHandler(Cti.MessageType.AGENTSTATEEVENT, agentStateEventHandler);
Cti.setHandler(Cti.MessageType.AGENTERROR, agentErrorHandler);
Cti.setHandler(Cti.MessageType.ERROR, errorHandler);
Cti.setHandler(Cti.MessageType.AGENTDIRECTORY, agentDirectoryHandler);

Cti.setHandler(Cti.MessageType.CONFERENCES, conferencesHandler);
Cti.setHandler(Cti.MessageType.CALLHISTORY, callHistoryHandler);

```

```
xc_webrtc.setHandler(xc_webrtc.MessageType.GENERAL, webRtcGeneralEventHandler);
xc_webrtc.setHandler(xc_webrtc.MessageType.REGISTRATION, webRtcRegistrationEventHandler);
xc_webrtc.setHandler(xc_webrtc.MessageType.INCOMING, webRtcIncomingEventHandler);
xc_webrtc.setHandler(xc_webrtc.MessageType.OUTGOING, webRtcOutgoingEventHandler);
```

Login

User login

Users can connect using login, password and phone number:

```
var wsurl = "ws://" + server + "/ctichannel?username="+username+"&agentNumber="+phoneNumber+"&password="+password;
Cti.WebSocket.init(wsurl, username, phoneNumber);
```

Agent login

An agent can be logged in using *Cti.loginAgent(agentPhoneNumber, agentId)*. For the moment, the phone number used for agent login should be the same as the one used for user login, otherwise you will get many error messages “LoggedInOnAnotherPhone”.

Following cases are handled:

- agent is not logged and requests a login to a known line: the agent is logged in
- agent is not logged and requests a login to an unknown line: an error is raised:

```
{ "Error": "PhoneNumberUnknown" }
```

- agent is already logged on the requested line: the agent stays logged
- agent is already logged on another line: an error is raised and the agent stays logged (on the number where he was logged before the new request). It's up to the implementation to handle this case.

```
{ "Error": "LoggedInOnAnotherPhone", "phoneNb": "1002", "RequestedNb": "1001" }
```

- agent is not logged and requests a login to a line already used by another agent: the agent takes over the line and the agent previously logged on the line is unlogged

Generic CTI Messages

Error

- Cti.MessageType.ERROR

LoggedOn

- Cti.MessageType.LOGGEDON

Sheet

- Cti.MessageType.SHEET

```
{ "msgType": "Sheet", "ctiMessage": { "timenow": 1425055334, "compressed": true, "serial": "xml",
  "payload": { "profile": { "user": { "internal": [ { "content": "xivo", "name": "ipbxid" },
    { "content": "link", "name": "where" }, { "content": "1425055330.23", "name": "uid" },
    { "content": "no", "name": "focus" }, { "content": "1", "name": "zip" } ] },
  "sheetQtui": null, "sheetInfo": [ { "value": "http://www.google.fr/", "name": "popupUrl", "order": 1 } ] } }
```


Phone Events

- Cti.MessageType.PHONEEVENT

Phone events are automatically sent when application is connected

Format

```
{
  "msgType": "PhoneEvent",
  "ctiMessage": {
    "eventType": "EventRinging",
    "DN": "1118",
    "otherDN": "1058",
    "linkedId": "1447670380.34",
    "uniqueId": "1447670382.37",
    "queueName": "blue",
    "userData": {
      "XIVO_CONTEXT": "default", "XIVO_USERID": "9", "XIVO_SRCNUM": "1058", "XIVO_DSTNUM": "3000"
    }
  }
}
```

fields	Description
Event types	<ul style="list-style-type: none"> • EventReleased • EventDialing • EventRinging • EventEstablished
DN	The directory number of the event
otherDN	Can be calling number of called number
queueName	Optional, the queue name for inbound acd calls
UserData	Contains a list of attached data, system data XIVO_ or data attached to the call key beginning by USR_

If you use the following preprocess subroutine

```
[user_data_test]
exten = s,1,Log(DEBUG,**** set user data ****)
same  =      n,SET(USR_DATA1=hello)
same  =      n,SET(USR_DATA2=24)
same  =      n,SET(USR_DATA3=with space)
same  =      n,Return()
```

you will get these data in the events. Data can also be attached using the *Cti.dial* command.

Voice Mail Status Update

- VOICEMAILSTATUSUPDATE : “VoiceMailStatusUpdate”,

```
{ "msgType": "VoiceMailStatusUpdate", "ctiMessage": { "voiceMailId": 58, "newMessages": 2, "waitingMessages": 1 }
```

Link Status Update

- Cti.MessageType.LINKSTATUSUPDATE

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`
- `TotalNumberCallsAbandoned`
- `TotalNumberCallsAbandonedAfter15`
- `PercentageAbandonedAfter15`
- `WaitingCalls`
- `LongestWaitingTime`
- `EWT`
- `AvailableAgents`
- `TalkingAgents`

```
{
  "msgType": "QueueStatistics",
  "ctiMessage": {
    "queueId": 11, "counters": [{ "statName": "AvailableAgents", "value": 0 }, { "statName": "LoggedAgent"
  }
}
```

Some messages contain a `queueRef` with a queue's name instead of the `queueId`. This issue should be eliminated in future versions.

```
{ "queueRef": "travels", "counters": [{ "statName": "TotalNumberCallsAbandoned", "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-
```

Queue Configuration

- `QUEUECONFIG` : “QueueConfig”,

```
{ "id": 8, "context": "default", "name": "blue", "displayName": "blue sky", "number": "3506" }
```

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",
  {"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":
  ]
}
```

Queue Member

- Handler on : Cti.MessageType.QUEUEMEMBER

Received when an agent is associated to a queue or a penalty is updated. Penalty is -1 when agent is removed from a queue

```
{ "agentId":19, "queueId":3, "penalty":12 }
```

Queue Member List

- Handler on : Cti.MessageType.QUEUEMEMBERLIST

```
{
  "msgType": "QueueMemberList",
  "ctiMessage": [
    { "agentId":129, "queueId":8, "penalty":2},
    { "agentId":139, "queueId":168, "penalty":2},
    { "agentId":129, "queueId":10, "penalty":0},
    { "agentId":129, "queueId":11, "penalty":0}
  ]
}
```

Agent State Event

- Cti.MessageType.AGENTSTATEEVENT

- AgentLogin

```
{ "name": "AgentLogin", "agentId":19, "phoneNb": "1000", "since":1423839787, "queues": [8,14,170,
```

- AgentReady

```
{ "name": "AgentReady", "agentId":19, "phoneNb": "1000", "since":0, "queues": [8,14,170,4,1], "cau
```

- AgentOnPause

```
{ "name": "AgentOnPause", "agentId":19, "phoneNb": "1000", "since":0, "queues": [8,14,170,4,1], "c
```

- 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], "c
```

- AgentDialing

```
{ "name": "AgentDialing", "agentId": 19, "phoneNb": "1000", "since": 0, "queues": [8, 14, 170, 4, 1], "c
```

– AgentOnCall

```
{ "msgType": "AgentStateEvent", "ctiMessage": {
  { "name": "AgentOnCall", "agentId": 19, "phoneNb": "1000", "since": 0, "queues": [8, 14, 170, 4, 1]
    "cause": "available", "acd": false, "direction": "Incoming", "callType": "External", "mor
```

– 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", "num
      "agentState": { "agentId": 8, "cause": "", "name": "AgentReady", "phoneNb": "1001", "queue
```

Agent Configuration

- Cti.MessageType.AGENTCONFIG

Triggered when agent configuration changes

```
{ "id": 23, "firstName": "Jack", "lastName": "Flash", "number": "2501", "context": "default" }
```

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"
  { "id": 20, "firstName": "Maricé", "lastName": "Sapritchà", "number": "2602", "context": "default", "grou
  { "id": 147, "firstName": "Etienne", "lastName": "Burgad", "number": "30000", "context": "default", "grou
  { "id": 148, "firstName": "Caroline", "lastName": "HERONDE", "number": "29000", "context": "default", "g
  { "id": 149, "firstName": "Eude", "lastName": "GARTEL", "number": "75000", "context": "default", "groupI
  { "id": 22, "firstName": "Alice", "lastName": "Johnson", "number": "2058", "context": "default", "groupI
]
```

Agent Listen

- AGENTLISTEN: “AgentListen”,

Receives agent listen stop / start event, received automatically if user is an agent, no needs to subscribe.

```
{ "started": false, "phoneNumber": "1058", "agentId": 22 }
```

Agent Group List

- AGENTGROUPLIST : “AgentGroupList”

Agent group list triggered by command : *Cti.getList(“agentgroup”)*

```
[
  { "id":1, "name": "default" },
  { "id":2, "name": "boats" },
  { "id":3, "name": "broum" },
  { "id":4, "name": "bingba3nguh" },
  { "id":5, "name": "salesexpert" },
  { "id":6, "name": "a_very_long_group_name" }
]
```

Agent Statistics

Received on subscribe to agent statistics with method *Cti.subscribeToAgentStats()*, current statistics are received automatically on subscribe.

- AGENTSTATISTICS : “AgentStatistics”

```
{ "id":22,
  "statistics": [
    { "name": "AgentPausedTotalTime", "value":0 },
    { "name": "AgentWrapupTotalTime", "value":0 },
    { "name": "AgentReadyTotalTime", "value":434 },
    { "name": "LoginDateTime", "value": "2015-04-27T08:15:01.081+02:00" },
    { "name": "LogoutDateTime", "value": "2015-04-27T08:14:49.427+02:00" }
  ]
}
```

Call History

Cti.getUserCallHistory(size)

Get the call history of the logged in user, limited to the last *size* calls.

Cti.getAgentCallHistory(size)

Get the call history of the logged in agent, limited to the last *size* calls.

Cti.getQueueCallHistory(queue, size)

Get a call history for a queue or a set of queues. You may pass part of a queue name (not display name).

i.e. pass bl if you want to match queue name blue, black and blow

Associated Handler **CALLHISTORY**

Received when calling the above methods *Cti.getAgentCallHistory(size)* or *Cti.getUserCallHistory(size)* .

- CALLHISTORY : “CallHistory”

```
{
  "start": "2014-01-01 08:00:00",
  "duration": "00:21:35",
  "srcNum": "0115878",
  "dstNum": "2547892",
  "status": "answered"
}
```

For queue calls status can be :

- full - full queue
- closed - closed queue
- joinempty - call arrived on empty queue
- leaveempty - exit when queue becomes empty
- divert_ca_ratio - call redirected because the ratio waiting calls/agents was exceeded
- divert_waittime - call redirected because estimated waiting time was exceeded;
- answered - call answered
- abandoned - call abandoned
- timeout - maximum waiting time exceeded

For other calls

- emitted
- missed
- ongoing

Callback Messages

Callback lists

Received when calling *Callback.getCallbackLists()*.

- CALLBACKLISTS : “CallbackLists”

```
{ "uuid": "b0849ac0-4f4a-4ed0-9386-53ab2afd94b1",
  "name": "Liste de test",
  "queueId": 1,
  "callbacks": [
    { "uuid": "a967da84-bc41-4bf4-a4fc-2bcc54e11606",
      "listUuid": "b0849ac0-4f4a-4ed0-9386-53ab2afd94b1",
      "phoneNumber": "0230210082",
      "mobilePhoneNumber": "0789654123",
      "firstName": "Alice",
      "lastName": "O'Neill",
      "company": "YourSociety",
      "description": null,
      "agentId": null,
      "dueDate": "2016-08-01",
      "preferredPeriod": {
        "default": false,
        "name": "Afternoon",
        "periodStart": "14:00:00",
        "periodEnd": "17:00:00",
        "uuid": "d3270038-e20e-498a-af71-3cf69b5cc792"
      }
    }
  ]
}
```

Callback Taken

Received after taking a callback with *Callback.takeCallback(uuid)*.

- CALLBACKTAKEN : “CallbackTaken”

```
{ "uuid": "a967da84-bc41-4bf4-a4fc-2bcc54e11606",  
  "agentId": 2 }
```

Callback Started

Received after starting a callback with *Callback.startCallback(uuid, phoneNumber)*.

- CALLBACKSTARTED : “CallbackStarted”

```
{ "requestUuid": "a967da84-bc41-4bf4-a4fc-2bcc54e11606",  
  "ticketUuid": "8e82de0f-847a-4606-97bf-bef5a18ea8b0" }
```

Callback Clotured

Received after giving to a callback a status different of Callback.

- CALLBACKCLOTURED : “CallbackClotured”

```
{ "uuid": "a967da84-bc41-4bf4-a4fc-2bcc54e11606" }
```

Callback Released

Received after releasing a callback with *Callback.releaseCallback(uuid)*.

- CALLBACKRELEASED : “CallbackReleased”

```
{ "uuid": "a967da84-bc41-4bf4-a4fc-2bcc54e11606" }
```

Callback Updated

Received when calling *Callback.updateCallbackTicket(uuid, status, description, dueDate, periodUuid)* with a new due date or period.

- CALLBACKREQUESTUPDATED : “CallbackRequestUpdated”

```
{ "request": {  
  "uuid": "a967da84-bc41-4bf4-a4fc-2bcc54e11606",  
  "listUuid": "b0849ac0-4f4a-4ed0-9386-53ab2afd94b1",  
  "phoneNumber": "0230210082",  
  "mobilePhoneNumber": "0789654123",  
  "firstName": "Alice",  
  "lastName": "O'Neill",  
  "company": "YourSociety",  
  "description": null,  
  "agentId": null,  
  "dueDate": "2016-08-01",  
  "preferredPeriod": {  
    "default": false,  
    "name": "Afternoon",  
    "periodStart": "14:00:00",  
    "periodEnd": "17:00:00",  
    "uuid": "d3270038-e20e-498a-af71-3cf69b5cc792"  
  }  
}
```



```
}
}}
```

Membership Messages

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}
  ]
}
```

Methods

Cti.changeUserStatus()

Update user status using a cti server configured status name

Cti.loginAgent(agentPhoneNumber, agentId)

Log an agent

Cti.logoutAgent(agentId)

Un log an agent

Cti.pauseAgent(agentId)

Change agent state to pause

Cti.unpauseAgent(agentId)

Change agent state to ready

Cti.listenAgent(agentId)

Listen to an agent

Cti.dnd(state)

Set or unset do not disturb, state true or false

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.originate(destination)

Originate a call

Cti.hangup()

Hangup a call

Cti.answer()

Answer a call

Cti.hold()

Put current call on hold

Cti.directTransfer(destination)

Tranfert to destination

Cti.attendedTransfer(destination)

Start a transfer to a destination

Cti.completeTransfer()

Complete previously started transfer

Cti.cancelTransfer()

Cancel a transfer

Cti.conference()

Start a conference using phone set capabilities

Cti.monitorPause(agentId)

Pause call recording

Cti.monitorUnpause(agentId)

Un pause call recording

Cti.getList(objectType)

Request a list of configuration objects, objectType can be :

- queue
- agent
- queuemember

Triggers handlers QUEUelist, AGENTlist, QUEUEMEMBERlist. Subscribes to configuration modification changes, handlers QUEUECONFIG, AGENTCONFIG, QUEUEMEMBER can also be called

Cti.setAgentQueue(agentId, queueId, penalty)

- agentId (Integer) : id of agent, returned in message *Agent Configuration*
- queueId (Integer) : id of queue, returned in message *Queue Configuration*
- penalty (Integer) : positive integer

If agent is not associated to the queue, associates it, otherwise changes the penalty

On success triggers a *Queue Member* event, does not send anything in case of failure :

```
{ "agentId": <agentId>, "queueId": <queueId>, "penalty": <penalty> }
```

Cti.removeAgentFromQueue(agentId, queueId)

- agentId (Integer) : id of agent, returned in message *Agent Configuration*
- queueId (Integer) : id of queue, returned in message *Queue Configuration*

On success triggers a queue member event with penalty equals to -1, does not send anything in case of failure :

```
{ "agentId": <agentId>, "queueId": <queueId>, "penalty": -1 }
```

Cti.subscribeToAgentStats()

Subscribe to agent statistics notification. When called all current statistics are receive, and a notification is received for each updates. Both initial values and updates are transmitted by the *Agent Statistics* messages.

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 messages.

Cti.naFwd(destination,state)

Forward on non answer

Cti.uncFwd(destination,state)

Unconditionnal forward

Cti.busyFwd(destination,state)

Forward on busy

Callback Commands

Callback.getCallbackLists()

Retrieve the lists of callbacks with their 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 with the provided description and status. Allowed values for status are:

- NoAnswer
- Answered
- Fax
- Callback

dueDate is an optional parameter specifying the new due date using ISO format (“YYYY-MM-DD”).

periodUuid is an optional parameter specifying the new preferred period for the callback.

Membership Commands

Membership.init(cti)

Initialize the Membership library using the given Cti object.

Membership.getUserDefaultMembership(userId)

Request the default membership for the given user id. Warning, the userId is not the same as the agentId.

Membership.setUserDefaultMembership(userId, membership)

Set the default membership for the given user id. Warning, the userId is not the same as the agentId. 'membership' should be an array of Queue membership like:

```
[
  { "queueId": 8, "penalty": 1 },
  { "queueId": 17, "penalty": 0 },
  { "queueId": 18, "penalty": 0 },
  { "queueId": 23, "penalty": 0 }
]
```

Membership.setUsersDefaultMembership(userIds, membership)

Set the default membership for the given array of user id. Warning, the userId is not the same as the agentId. 'userIds' should be an array of user id like :

```
[1, 2, 3]
```

'membership' should be an array of Queue membership like:

```
[
  { "queueId": 8, "penalty": 1 },
  { "queueId": 17, "penalty": 0 },
  { "queueId": 18, "penalty": 0 },
  { "queueId": 23, "penalty": 0 }
]
```

Membership.applyUsersDefaultMembership(userIds)

Apply the existing default configuration to a set of users. Warning, the userId is not the same as the agentId. 'usersIds' should be an array of userId like:

```
:: [1, 2, 7, 9]
```

webRTC integration**Methods**

Once the cti login done, you can init the webRTC component by calling the `xc_webrtc.init` method.

xc_webrtc.init(name, ssl, websocketPort, remoteAudio, ip)

Init the webRTC connection and register the user's line.

- name - user's login to get the line details,
- ssl - if set to true the wss is used,
- websocketPort, ip - port and address for the webRTC websocket connection, when ip is not passed the xivo ip is used,
- remoteAudio - id of the HTML5 audio element for remote audio player, if not passed 'audio_remote' is used. The element should look like:

```
<audio id="audio_remote" autoplay="autoplay"></audio>
```

xc_webrtc.dial(destination)

Start a webRTC call.

xc_webrtc.answer()

Answer an incoming webRTC call.

xc_webrtc.hold()

Toggle hold on a webrtc call.

xc_webrtc.dtmf(key)

Send a DTMF.

xc_webrtc.setHandler(eventName, handler)

Set a handler for eventName from xc_webrtc.MessageType.

xc_webrtc.disableICE()

Disable ICE server use, only LAN addresses will be used in the SDP.

xc_webrtc.setIceUrls(urls)

Set a list of STUN/TURN servers, for example:

```
[{ url: 'stun:stun.l.google.com:19302'}, { url:'turn:turn.server.org', username: 'user', credenti
```

Events

There are for 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/xivoxc/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.

1.5.3 Rest API

General form

```
http://localhost:$xucport/xuc/api/1.0/$method/$domain/$username/  
withHeaders(("Content-Type", "application/json"))
```

- \$xucport : Xuc port number (default 8090)
- \$method : See available methods below
- \$domain : Represents a connection site, can be anything
- \$username : XiVO client user username

Events

Xuc post JSON formatted events on URL `eventUrl = "http://localhost:8090/xivo/1.0/event/avencal1"` 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"
  }
}
```

Connection

POST `http://localhost:$xucport/xuc/api/1.0/connect/$domain/$username/`

```
{ "password" : "password" }

curl -XPOST -d '{"password": "<password>"}' -H "Content-Type: application/json" http://localhost:
```

DND

POST `http://localhost:$xucport/xuc/api/1.0/dnd/$domain/$username/`

```
{ "state" : [false|true] }

curl -XPOST -d '{"state": false}' -H "Content-Type: application/json" http://localhost:8090/xuc/ap
```

Dial

POST `http://localhost:$xucport/xuc/api/1.0/dial/$domain/$username/`

```
{ "number" : "1101" }

curl -XPOST -d '{"number": "<number>"}' -H "Content-Type: application/json" http://localhost:8090:
```

Phone number sanitization

Dial command automatically applies filters to the phone number provided to make it valid for Xivo. Especially, it removes invalid characters and handles properly different notations of international country code.

Some countries don't follow the international standard and actually keep the leading zero after the country code (e.g. Italy). Because of this, if the zero isn't surrounded by parenthesis, the filter keeps it ¹.

¹ See Redmine ticket #150

Forward

All forward commands use the above payload

```
{ "state" : [true|false],  
  "destination" : "1102" }
```

Unconditionnal

POST [http://localhost:\\$xucport/xuc/api/1.0/uncForward/\\$domain/\\$username/](http://localhost:$xucport/xuc/api/1.0/uncForward/$domain/$username/)

```
curl -XPOST -d '{"state":true,"destination":"<destnb>"}' -H "Content-Type: application/json" http://localhost:$xucport/xuc/api/1.0/uncForward/$domain/$username/
```

On No Answer

POST [http://localhost:\\$xucport/xuc/api/1.0/naForward/\\$domain/\\$username/](http://localhost:$xucport/xuc/api/1.0/naForward/$domain/$username/)

```
curl -XPOST -d '{"state":true,"destination":"<destnb>"}' -H "Content-Type: application/json" http://localhost:$xucport/xuc/api/1.0/naForward/$domain/$username/
```

On Busy

POST [http://localhost:\\$xucport/xuc/api/1.0/busyForward/\\$domain/\\$username/](http://localhost:$xucport/xuc/api/1.0/busyForward/$domain/$username/)

```
curl -XPOST -d '{"state":true,"destination":"<destnb>"}' -H "Content-Type: application/json" http://localhost:$xucport/xuc/api/1.0/busyForward/$domain/$username/
```

Handshake

Will repost all events on the configured URL

POST [http://localhost:\\$xucport/xuc/api/1.0/handshake/\\$domain/](http://localhost:$xucport/xuc/api/1.0/handshake/$domain/)

AgentLogout

Logout un agent

POST [http://\\$xuchost:\\$xucport/xuc/api/1.0/agentLogout/](http://$xuchost:$xucport/xuc/api/1.0/agentLogout/)

```
curl -XPOST -d '{"phoneNumber":"<phoneNumber>"}' -H "Content-Type: application/json" http://localhost:$xucport/xuc/api/1.0/agentLogout/
```

TogglePause

Change state of an agent, pause if ready, ready if on pause

POST [http://\\$xuchost:\\$xucport/xuc/api/1.0/togglePause/](http://$xuchost:$xucport/xuc/api/1.0/togglePause/)

```
curl -XPOST -d '{"phoneNumber":"<phoneNumber>"}' -H "Content-Type: application/json" http://localhost:$xucport/xuc/api/1.0/togglePause/
```

1.5.4 Statistics

Exposed by xuc

Queue statistics

These real time statistics are calculated nearly in real time from the queue_log table. Statistics are reset to 0 at midnight (24h00) and can be changed by configuration.

Real time calculated Queue statistic

name	Description
TotalNumberCallsEntered	Total number of calls entered in a queue
TotalNumberCallsAbandoned	Total number of calls abandoned in a queue (not answered)
TotalNumberCallsAbandonedAfter15	Total number of calls abandoned after 15 seconds
TotalNumberCallsAnswered	Total number of calls answered
TotalNumberCallsAnsweredBefore15	Total number of calls answered before 15 seconds
PercentageAnsweredBefore15	Percentage of calls answered before 15 seconds over total number of calls entered
PercentageAbandonedAfter15	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 are also available for the sliding last hour by adding LastHour to the name .i.e. TotalNumberCallsAbandonedLastHour

For percentage, it is the mean of the sliding last hour value

Other queue statistics

Other queue statistics are calculated by xivo cti server

- AvailableAgents
- TalkingAgents
- LongestWaitTime
- WaitingCalls
- EWT

Definition in xivo documentation [xivo documentation](#)

Calculated Agent statistics

name	Description
PausedTime	Total time agent in pause
WrapupTime	Total time agent in wrapup
ReadyTime	Total time agent ready
InbCalls	Total number of inbound calls received internal and external
InbCallTime	Total time for inbound calls received internal and external
InbAnsCalls	Answered inbound calls received internal and external
InbUnansCalls	Unanswered inbound calls received internal and external
InbPercUnansCalls	Percentage of unanswered inbound calls received internal and external
InbAverCallTime	Average time for inbound calls received internal and external
OutCalls	Total number of outbound calls received internal and external
LoginDateTime	Last login date time
LogoutDateTime	Last logout date time

Inbound calls, are all calls received by an agent, internal, external or acd calls. Outbound calls are all calls dialed by an agent, internal or external calls.

Agent statistics are calculated internally 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.

1.5.5 Technical structure of XiVO-CC

Reporting

The reporting is composed of four packages: pack-reporting, xivo-full-stats, xivo-reporting-db and xivo-db replication.

These packages will feed the tables of the xivo_stats database:

- xivo-db-replication feeds the tables cel and queue_log in real time, and the configuration tables (dialaction, linefeatures, etc...) every 5 minutes
- xivo-full-stats feeds in real time 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

1.6 Troubleshooting

In this section, we give some troubleshooting hints. Continue by choosing the component.

1.6.1 Xuc et Xuc_mgt - applications web ccmanger, agent et assistant

Basic checks

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.

1.6.2 Application Configuration (xuc_rigths)

1.6.3 Recording

1.6.4 SpagoBI

1.6.5 Kibana

1.6.6 NGINX - proxy web

Basic check

On the standard HTTP port of the machine (80) you have the fingerboard page.

Docker says nginx is restarting

- Check logs for missing files or links, nginx refuses to start if one of servers is not accessible, e.g. xuc is down.

1.7 XiVO Centralized Interface

The XiVO Centralized Interface (XCI) 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.

1.7.1 Installation

Requirements

The XiVO Centralized Interface (XCI) requires :

- A Linux server with PostgreSQL, Docker and Docker-Compose installed
- Some XiVOs to manage !

Automated installation

An installation script is provided to execute all the installations tasks. To run it, execute the following command :

```
curl https://gitlab.com/xivo-utils/icdu-packaging/raw/master/install-icdu.sh | sudo bash
```

It will ask you a passphrase for generating an SSH key.

The configuration files are located in `/etc/docker`.

Run the application

Optionally, you can set a bash alias for conveniently run XCI :

```
alias dcomp='docker-compose -p icdu -f /etc/docker/compose/icdu.yml'
```

Then simply :

```
dcomp up -d
```

XCI should now be accessible through <http://my-server-ip:9001>

Manual installation

The configuration files and the Docker-Compose files are available in a specific [Git repository](#).

Database setup

XCI 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 PostgreSQL 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-oss";'
```

I can't connect to PostgreSQL It is possible that PostgreSQL complains when you're trying to connect. The solution is to modify the `pg_hba.conf` (in Debian, located in `/etc/postgresql/X.X/main`) and add the following line at the end :

```
local all all trust
```

Generate SSH key

In order to let XCI 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
```

1.7.2 Web interface

The XiVO Centralized Interface (XCI) is managed through a web interface. In the following sections, we will highlight the main features of the system.

Definitions

XCI uses a few concepts that are important to understand in order to use the interface correctly.

XiVO The XiVOs servers that are managed by XCI. XCI will automatically retrieve the entities and the users from them and apply the configuration to them.

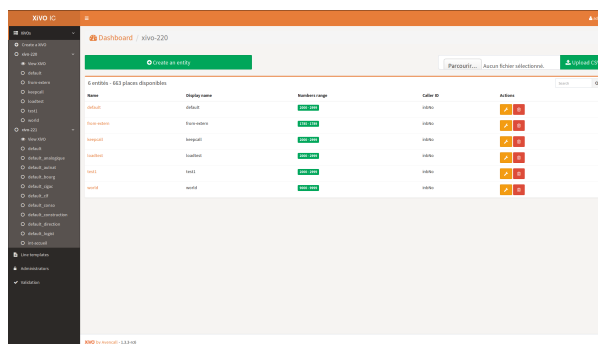
Entity Entities, also called Contexts, are the parts of the dialplan. Users are attached to them.

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 XCI and manage the XiVOs.

View XiVO

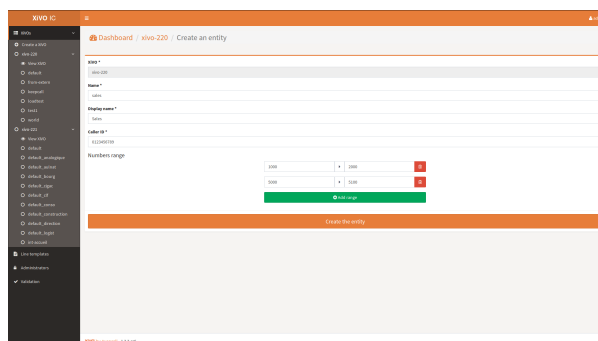


On the sidebar, each XiVO has its own **View XiVO** link. This page allows you to :

- **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

Create entity

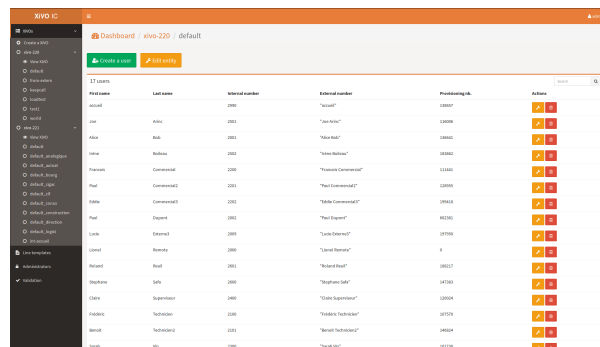


This page allows you to add a new entity to a XiVO. You have to provide the following informations :

- **Name** : name that will be used by the XiVO server
- **Display name** : name that will be displayed on XCI
- **Caller ID** : phone number that will be displayed on outgoing call from this entity
- **Intervals** : ranges of phone numbers that will be available to this entity. For each one, provide :
 - **Start**
 - **End**

The system will return an error if the intervals overlap with other entities

View entity

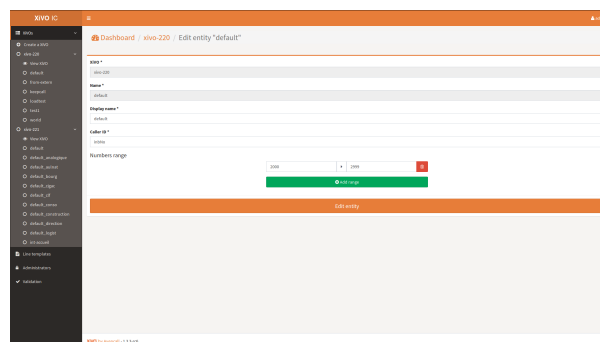


First name	Last name	Internal number	External number	Priority	Actions
John	Doe	2201	2201	1	[wrench] [trash]
John	Doe	2202	2202	1	[wrench] [trash]
John	Doe	2203	2203	1	[wrench] [trash]
John	Doe	2204	2204	1	[wrench] [trash]
John	Doe	2205	2205	1	[wrench] [trash]
John	Doe	2206	2206	1	[wrench] [trash]
John	Doe	2207	2207	1	[wrench] [trash]
John	Doe	2208	2208	1	[wrench] [trash]
John	Doe	2209	2209	1	[wrench] [trash]
John	Doe	2210	2210	1	[wrench] [trash]
John	Doe	2211	2211	1	[wrench] [trash]
John	Doe	2212	2212	1	[wrench] [trash]
John	Doe	2213	2213	1	[wrench] [trash]
John	Doe	2214	2214	1	[wrench] [trash]
John	Doe	2215	2215	1	[wrench] [trash]
John	Doe	2216	2216	1	[wrench] [trash]
John	Doe	2217	2217	1	[wrench] [trash]

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.*

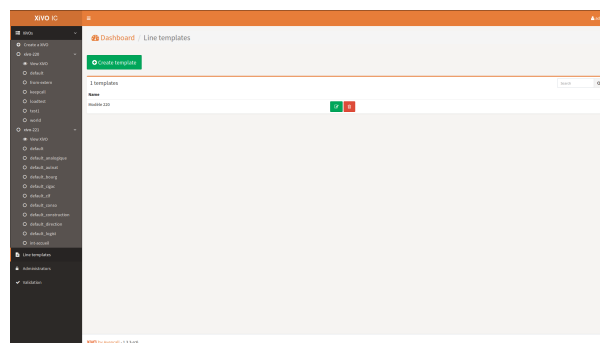
Edit entity



This page allow you to modify an entity. Please refer to the [Create entity](#) section for fields details.

Line templates

List templates

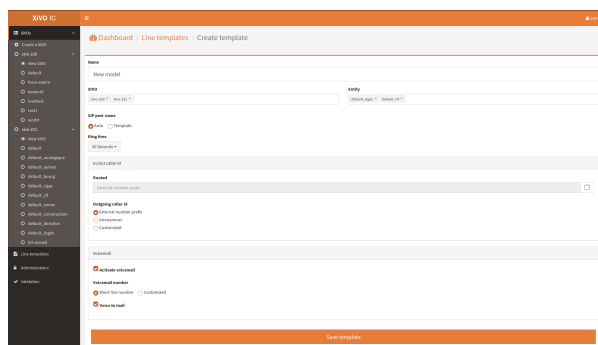


Name	Status
Line 220	[wrench] [trash]

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 displayed on XCI
- **XiVO** : select the XiVOs for which this template will be available
- **Entity** : select the entities for which this template will be available. *Only entities of the selected XiVOs are displayed*
- **SIP peer name** : *Auto* or *Model*
- **Ringing time** : number of seconds before incoming call is rejected
- **Routed** :
 - The text field allows you to provide the SDA prefix to call the phone
 - Uncheck the checkbox if you don't want the phone to be called from the outside
- **Outgoing caller id** : specify what number is displayed on outgoing call. Possible values are :
 - External number prefix
 - Anonymous
 - Customized : a text field appears to provide the custom number
- **Voicemail** :
 - **Activate voicemail** : enable or not the voicemail
 - **Voicemail number** : specify what number is used to call the voice mail. Possible values are :
 - * **Short line number** : use the default short number
 - * **Customized** : a text field appear to provide the custom number
 - **Voice to mail** : whether or not to send an email when a new message is left

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 informations :

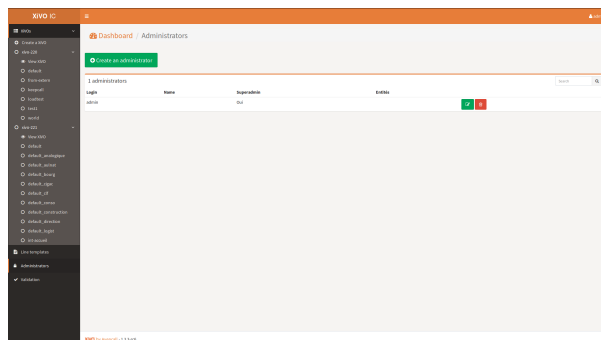
- **Template** : line template to use as a template to create the user. *The main options of the template are displayed below*
- **First name**
- **Last name**
- **Internal number** : number that will be used to internally call the user. *Only the available numbers are displayed*
- **CTI credentials** : provide a login and a password to allow the user to connect through CTI interfaces

Edit user

This page allows you to modify a user. Please refer to the [Create user](#) section for fields details.

Administrators

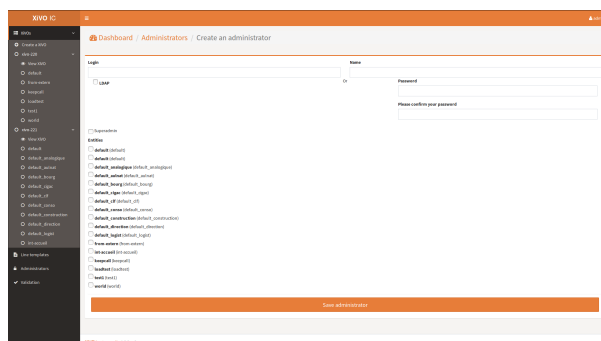
List administrators



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

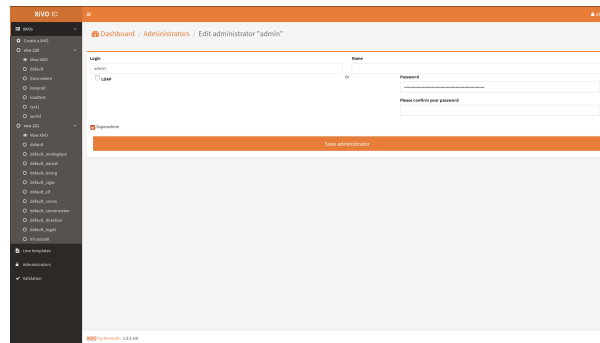
Create administrator



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 XCI
- **Name** : name that will be displayed on XCI
- **LDAP** : if checked, the LDAP authentication configured in `application.conf` will be used
- **Password** : password used by the administrator to connect to XCI. *Shown only if LDAP disabled*
- **Superadmin** : whether or not this administrator is a super-administrator. Super-administrators can manage everything in XCI
- **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.

1.7.3 REST API

The XiVO Centralized Interface (XCI) exposes some REST API that you can use to integrate with your tools.

General form

`http://$my-server-ip:$xcipoint/api/1.0/$method`

withHeaders(("Content-Type", "application/json"))

- \$xcipoint : XCI 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 'xci-cookie' --data '{"login": "admin", "password": "admin"}'
curl 'http://localhost:9000/api/1.0/xivo' -H 'Content-Type: application/json' -b 'xci-cookie'
```

XiVO

The following methods allow you to operate on the XiVOs managed by XCI.

List

List all the XiVOs configured on XCI.

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,
        "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",
          "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,
    "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\\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")
        return self.handle_json_response(response)
```

```

def create_line_template(self, data):
    self.make_post_request("/templates", data)

def get_line_templates(self):
    response = self.make_get_request("/templates")
    return self.handle_json_response(response)

def create_user(self, data):
    self.make_post_request("/users", data)

def make_get_request(self, method):
    request = Request(self.base_url + method, headers = {"Cookie": self.cookie})
    response = urlopen(request)
    return response

def make_post_request(self, method, data):
    header = {"Content-Type": "application/json", "Cookie": self.cookie}
    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")

```

Indices and tables

- `genindex`
- `search`