# Setting up a QueueMetrics WebRTC Softphone

## 1. Introduction

The evolution of modern communications involves the creation of new standards that can, at first, pose some significant implementation challenges.

The ***WebRTC*** project, which laid the cornerstone for browser to browser communications, has undergone a heavy development path. This is also because of the privacy issues that had to be addressed, concerning the exposing of local capabilities and streams.

Now that these issues have been taken care of, ***WebRTC*** offers a stable and secure platform, supporting state of the art encryption standards and effortless communication with users, customers and between employers through voice, video and messaging. All this without the need for any type of additional hardware.

In this tutorial we will guide you, step by step, in creating and setting up a secure ***Asterisk***/***QueueMetrics*** environment supporting ***WebRTC*** technology.

## 2. Asterisk Setup

### 2.0 **Operating System and Software Versions**

The steps detailed in this tutorial refer to a ***CentOS7*** (64 bit) system, using ***Asterisk-13.8.2*** and ***QueueMetrics 15.10***.

### 2.1 **Prerequisites**

Before we begin our process we need to make sure our system has some essential software correctly installed by executing the following instructions.

```
yum update

yum install gcc-c++ make gnutls-devel kernel-devel libxml2-devel ncurses-devel
subversion doxygen texinfo curl-devel net-snmp-devel neon-devel

yum install uuid-devel libuuid-devel sqlite-devel sqlite git speex-devel gsm-
devel python-devel patch autoconf
```

## 2.2 **Install libSRTP**

***libSRTP*** is an open-source library that allows the implementation of Secure Real-time Transport Protocol ( ***SRTP***). It was originally authored by Cisco Systems, Inc. It is available under a BSD-style license. Follow these instruction to correctly install ***libSRTP*** on your system:

```
cd /usr/src/
wget http://srtp.sourceforge.net/srtp-1.4.2.tgz
tar zxvf srtp-1.4.2.tgz
cd srtp
autoconf
./configure CFLAGS=-fPIC --prefix=/usr/local
make
make install
cp /usr/local/lib/libsrtp.a /lib
ldconfig
cd ..
```

## 2.3 **Install Jansson**

Jansson is a C library for encoding, decoding and manipulating JSON data. Since it's vital for WebRTC 's correct functioning install it following these instructions:

```
cd /usr/src
wget http://www.digip.org/jansson/releases/jansson-2.5.tar.gz
tar zxvf jansson-2.5.tar.gz
cd jansson-2.5
./configure -prefix=/usr/
make
make install
ldconfig
cd ..
```

## 2.4 **Install Openssl**

OpenSSL is an open source project that provides a robust, commercial-grade, and full-featured toolkit for the Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols. It is also a general-purpose cryptography library.

```
cd /usr/src
wget http://www.openssl.org/source/openssl-1.0.1g.tar.gz
tar -zxf openssl-1.0.1g.tar.gz
cd openssl-1.0.1g
./config
make
make test
make install_sw
cd /usr/src
rm -rf openssl-1.0.1g.tar.gz
rm -rf openssl-1.0.1g
cd usr/bin/
ln -s /usr/local/ssl/bin/openssl openssl
```

### 2.5 **Install Asterisk**

Now we are ready to install our Asterisk platform. In this tutorial we are using Asterisk-13.9.1 .

```
cd /usr/src
wget  http://downloads.asterisk.org/pub/telephony/asterisk/asterisk-13-
current.tar.gz
tar -xzvf asterisk-13-current.tar.gz

cd asterisk-XX.XX.XX/ && make clean            CHANGE  XX.XX.XX  TO  THE  ACTUAL
CURRENT VERSION NUMBER

sudo ./install_prereq install
sudo ./install_prereq install-unpackaged


./configure  --with-crypto  --with-ssl  --with-srtp=/usr/local/lib
--libdir=/usr/lib64

./configure && make menuselect
```

After verifying things are as needed in *menuselect*, then build and install Asterisk

```
make && make install
```

Execute the next instruction only if you **don't** have a working asterisk configuration, since "make samples" will create new configuration files, thus erasing existing ones!

```
make samples

sudo make config
chkconfig asterisk on
```

You can start your Asterisk by typing:

```
service asterisk start
```

Then you can log in to it with:

```
asterisk -r
```

## 3. Secure Calling (TLS)

### 3.1 **TLS**

The Transport Layer Security (TLS) protocol is the successor of the Secure Sockets Layer (SSL) protocol. Both of them are cryptographic protocols that ensure data integrity, identification and privacy in communications over computer networks.
To ensure the correct functioning of the WebRTC softphone we must implement TLS communication. This precaution is necessary since modern browsers do not allow WebRTC communication unless it's using TLS to guarantee data security.

## 3.2 **Self-Signed certificates**

Normally, in order for TLS to work, an application server needs a digital certificate stating various information about the machine and the owner. These certificates essentially confirm to clients, trying to access your applications, that the server is a trusted peer and that the communication is secure.
TLS Certificates though, require third party confirmation and are released by "Certificate Authorities" (CA) through a process that may vary between different CA's.
An easy solution to this (especially for testing purposes), is to create a Self-Signed certificate, that is the equivalent of a normal certificate, with no third party confirmation. Even if this type of certificate will allow our application to work, it will display a warning in the client's browser the first time it tries to establish a connection with our server.

## 3.3 **Certificates creation (openssl)**

First off, we will need a self made CA certificate. Let's create it by using the openssl tool to make a private key and a CA certificate. We will place them in a folder named Keys.

```
mkdir /etc/asterisk/keys

cd /etc/asterisk/keys
openssl genrsa -des3 -out ca.key 4096
```

At this point openssl will ask us for a password to make sure others don't use our private key. After choosing the password, type:

```
openssl req -new -x509 -days 365 -key ca.key -out ca.crt
```

Let's take a look at the folder's content with:

```
ls
```

We should have two files by now, ***ca.key*** and ***ca.crt*** . Now we need to install ***ca.crt*** on the client's browser (this means we have to copy it on the client machine). Let's take a look on how to install it on Google Chrome for Ubuntu (this procedure may vary depending on the client's browser):

```
certutil -d sql:$HOME/.pki/nssdb -A -t TC -n "certificate_nickname" -i
/path_to_certificate/ca.crt
```

Now, back on the server side, we generate the server self-signed certificate, a certificate request and a private key for the server.

```
openssl genrsa -out key.pem 1024

openssl req -new -key key.pem -out certificate_request.csr

openssl x509 -req -days 365 -in certificate_request.csr -CA ca.crt -CAkey ca.key
-set_serial 01 -out self_signed_certificate.crt
```

Now we need to create a ***.pem*** file containing both our self-signed certificate and our private key.

To do this we can concatenate them in a single file, named ***asterisk.pem***

```
cat key.pem > asterisk.pem

cat self_signed_certificate.crt >> asterisk.pem
```

The content of the file should look like this:

```
-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQCqRw0jpQFn+f+lnDZiZzCRca9ojgu2brO+Q56jnqorvCIlYFC0
[...]
FT65O46u6Vmp1gPbNklOEg7TtZUtfacPY2PyeP4KoHaG
-----END RSA PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
MIIDvDCCAyWgAwIBAgIJAPMabsMiJJQPMA0GCSqGSIb3DQEBBQUAMIGbMQswCQYD
[...]
CfITDxcJBZfeXIPZP52+8FSMlm5985uMvao+emlIUGk11rY61Amxr387grDvgOaI
-----END CERTIFICATE-----
```

All the key and certificate files must be located in the ***/etc/asterisk/keys/*** folder.

Warning!

Until you follow through all of chapter 4 (Asterisk Configuration) steps, you won't be able to connect to Asterisk through TLS connection.

## 4. Asterisk Configuration

### 4.0 **Asterisk Configuration Files**

Asterisk configuration files can be found in the **/etc/asterisk** folder. More precisely, we are interested in **sip.conf**, **http.conf**, **rtp.conf** and **extensions.conf** files.

### 4.1 **sip.conf**

Our goal is to create two sip extensions that can be activated and used through a virtual softphone running inside a modern Internet browser. These extensions will be called 6000 and 6001, and for the sake of this example their password will match their names (not a smart move in real life).
Let's add the following to the end of the file:

```
udpbindaddr=0.0.0.0:5060
realm=xxx.xxx.xxx.xxx (your asterisk server ip address goes here)
transport=udp,ws,wss

;extension to use on web client
[6001]
host=dynamic
secret=6001
context=outgoing
type=peer
encryption=yes
```

```
avpf=yes
icesupport=yes
transport=ws,wss,udp
directmedia=no
disallow=all
allow=all
dtlsenable=yes
dtlsverify=fingerprint
dtlscertfile=/etc/asterisk/keys/asterisk.pem
dtlscafile=/etc/asterisk/keys/ca.crt
dtlssetup=actpass

;extension to use on web client
[6000]
host=dynamic
secret=6000
context=outgoing
type=peer
encryption=yes
avpf=yes
icesupport=yes
transport=ws,wss,udp
directmedia=no
disallow=all
allow=all
dtlsenable=yes
dtlsverify=fingerprint
dtlscertfile=/etc/asterisk/keys/asterisk.pem
dtlscafile=/etc/asterisk/keys/ca.crt
dtlssetup=actpass

;this is a normal sip phone
[200]
deny=0.0.0.0/0.0.0.0
secret=200
dtmfmode=rfc2833
canreinvite=no
context=outgoing
host=dynamic
trustrpid=yes
sendrpid=no
type=friend
nat=no
port=5060
qualify=yes
qualifyfreq=60
transport=udp
avpf=no
icesupport=no
encryption=no
callgroup=
pickupgroup=
dial=SIP/200
mailbox=200@device
permit=0.0.0.0/0.0.0.0
callerid=200 <200>
callcounter=yes
faxdetect=no
```

```
[global]
tlsenable=yes
tlsbindaddr=0.0.0.0
tlscertfile=/etc/asterisk/cert/asterisk.pem
```

## 4.2 **http.conf**

Let's add the following lines under the [general] tag in the **http.conf** file.

```
[general]
enabled=yes
bindaddr=0.0.0.0
bindport=8088
tlscertfile=/etc/asterisk/keys/asterisk.pem
tlsprivatekey=/etc/asterisk/keys/asterisk.pem
tlsenable=yes          ; enable tls - default no.
tlsbindaddr=0.0.0.0:8089
```

## 4.3 **rtp.conf**

Let's add the following lines under the [general] tag in the **rtp.conf** file.

```
[general]
rtpstart=10000
rtpend=20000
icesupport=yes
stunaddr=stun.l.google.com:19302
```

## 4.4 **extensions.conf**

Add the following lines to the end of the **extensions.conf** file. This is in fact a stanza in Asterisk dialplan that will be activated when a softphone extension makes a call through Asterisk.

```
[outgoing]

exten => _X.,1,Dial(SIP/${EXTEN})
exten => _X.,n,Answer()
exten => _X.,n,Hangup()
```

## 4.5 **Firewall Settings**

Now to finish the setup we need to add a firewall exception for our TLS port (8089 in our case).

```
ldconfig
service asterisk restart
sudo iptables -I INPUT -p tcp -m tcp --dport 8089 -j ACCEPT
```

To test your https connection open your browser and navigate to https://asterisk.ip.address:8089 .
If everything is working you should see a notice urging you to accept the untrusted certificate. After you accept the certificate the https connection will work perfectly.

## 5. QueueMetrics Configuration

### 5.1 **Install QueueMetrics**

Since QueueMetrics installation is a subject beyond the purpose of this tutorial we will not detail the procedure. If you need guidance on how to install QueueMetrics please read this other guide:

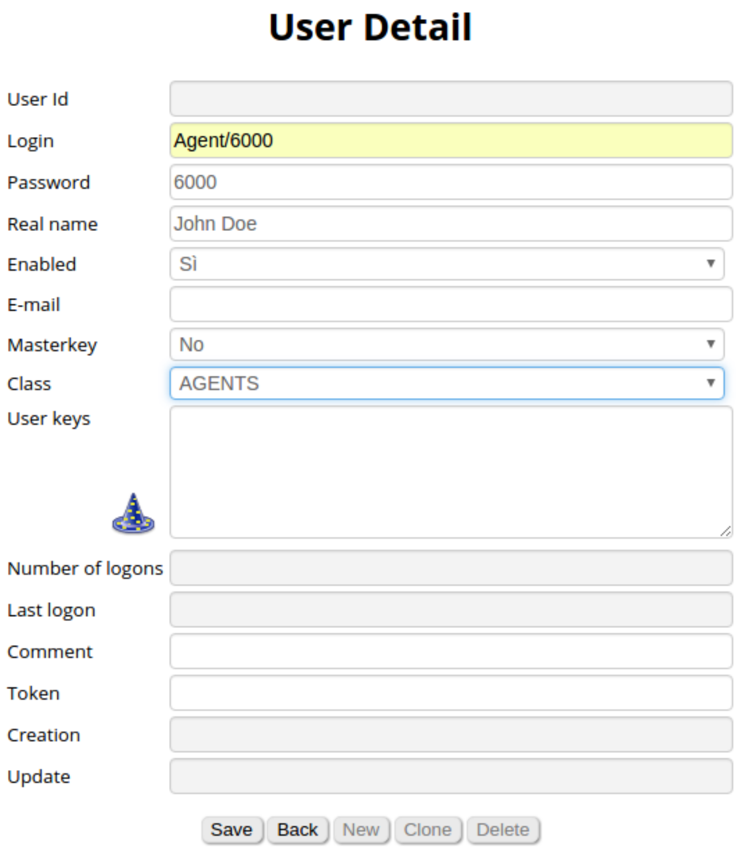
http://manuals.loway.ch/QM_UserManual-chunked/ar01s02.html

### 5.2 **Agents and users creation**

As we saw earlier we created two virtual extensions to be used through ***WebRTC*** technology. For ***QueueMetrics*** to be able to track these extensions (that currently exist only in Asterisk), we must create, for each of them, a corresponding **agent** and a corresponding **user**.

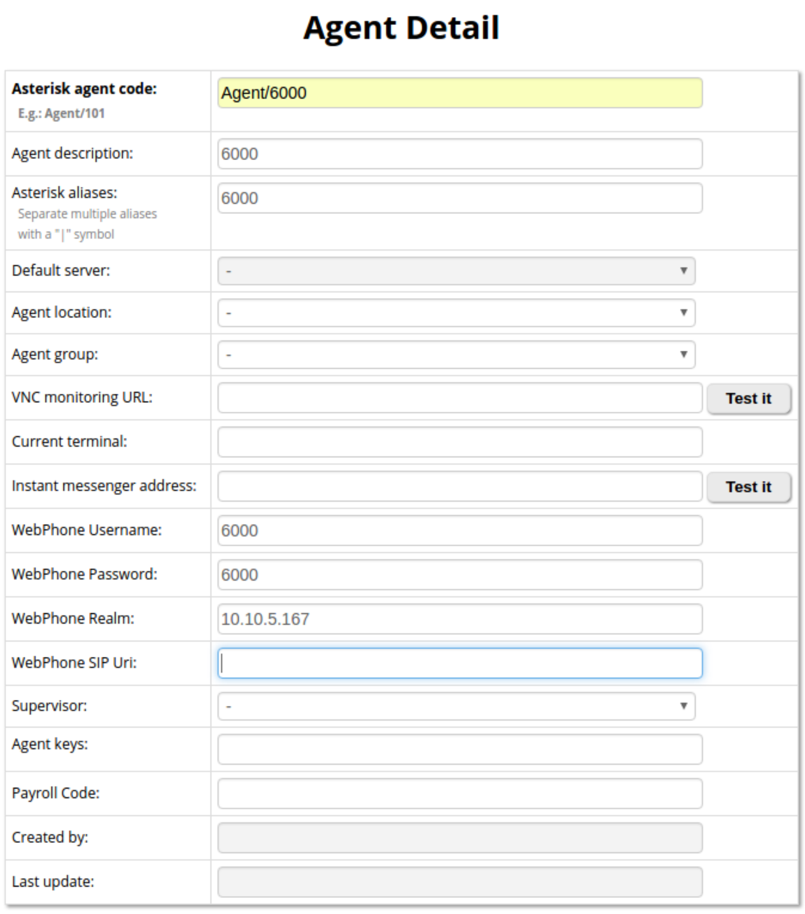
From the QueueMetrics **HomePage** select **Edit Users** → **Create New**

And fill in all the information as showed in the following picture.


User Detail

| | |
|---|---|
| User Id | |
| Login | Agent/6000 |
| Password | 6000 |
| Real name | John Doe |
| Enabled | Sì |
| E-mail | |
| Masterkey | No |
| Class | AGENTS |
| User keys | |
| Number of logons | |
| Last logon | |
| Comment | |
| Token | |
| Creation | |
| Update | |

Then, go back to the **HomePage** and select **Edit Agents** → **Create New** and fill in all the details as shown in the following picture:


Agent Detail

| | |
|---|---|
| **Asterisk agent code:** E.g.: Agent/101 | Agent/6000 |
| Agent description: | 6000 |
| Asterisk aliases: Separate multiple aliases with a "\|" symbol | 6000 |
| Default server: | - |
| Agent location: | - |
| Agent group: | - |
| VNC monitoring URL: | Test it |
| Current terminal: | |
| Instant messenger address: | Test it |
| WebPhone Username: | 6000 |
| WebPhone Password: | 6000 |
| WebPhone Realm: | 10.10.5.167 |
| WebPhone SIP Uri: | |
| Supervisor: | - |
| Agent keys: | |
| Payroll Code: | |
| Created by: | |
| Last update: | |

Where WebPhone Realm has to be filled with the IP address of your Asterisk server.

Now go back to the **HomePage** → **Edit system parameters** and edit the following parameters to match your configuration.

```
# Softphone parameters
default.sipaddress=xxx.xxx.xxx.xxx
default.websocketurl=wss://xxx.xxx.xxx.xxx:8089/ws
default.rtcWebBreaker=true
```

where xxx.xxx.xxx.xxx is the ip address of your **Asterisk** server.

## 6. Enabling QueueMetrics HTTPS Connections

In order for QueueMetrics' Agent Page to allow WebRTC functioning, we need to enable https connections to QueueMetrics.
To do this we must navigate (on our Asterisk server) to **/usr/local/queuemetrics/jdk1.6.0_22/bin/** where we will generate our private key and certificate request.

```
cd /usr/local/queuemetrics/jdk1.6.0_22/bin/
./keytool -genkey -alias queuemetrics.domain.net -keyalg RSA -keystore
keystore.jks -keysize 2048
./keytool -certreq -alias queuemetrics.domain.net -keystore keystore.jks -file
queuemetrics.domain.net.csr
```

Now we need to declare a connector in our **server.xml** file, to allow connections on **port 8443**.

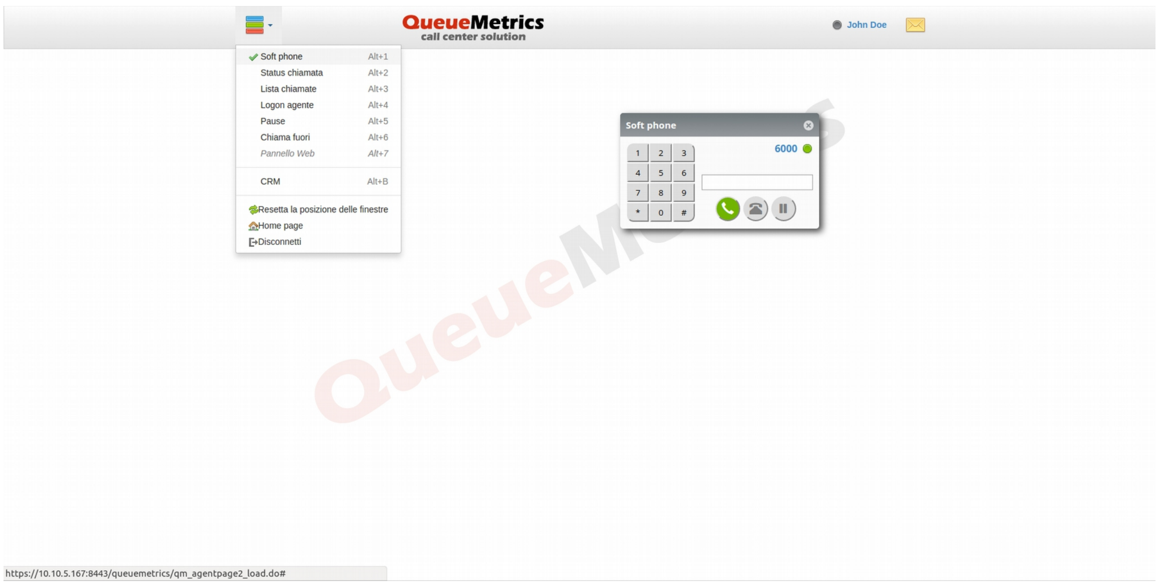
```
nano /usr/local/queuemetrics/tomcat/conf/server.xml

We must add:
<Connector port="8443" maxHttpHeaderSize="8192" maxThreads="150"
minSpareThreads="25" maxSpareThreads="75" enableLookups="false"
disableUploadTimeout="true" acceptCount="100"
secure="true" SSLEnabled="true" clientAuth="false"
sslProtocol="TLS" keyAlias="queuemetrics.domain.net"
keystoreFile="/usr/local/queuemetrics/jdk1.6.0_22/bin/keystore.jks"
scheme="https" keypass="yourpassword" />
```

To apply our changes we need to restart Tomcat.

```
service qm-tomcat6 restart
```

Now we should be ready to go! Login as Agent/6000 and from the upper left icon select the SoftPhone option.



## 7. TroubleShooting

- If you experience faults in asterisk calls through WEBRTC, regarding the following error (in the asterisk cli):

  ```
  “DTLS failure occurred on RTP instance '0x7f16a800a1a8' due to reason
  'missing tmp ecdh key', terminating”
  ```

  it can be solved by applying the following patch:

  ```
  cd /usr/src/asterisk
  wget
  ```

  https://issues.asterisk.org/jira/secure/attachment/52886/52886_res_rtp_asterisk.patch

  ```
  patch -p1 < 52886_res_rtp_asterisk.patch
  ```

  once the patch is applied, you must recompile Asterisk following the procedure listed in **chapter 2.5**.

- If in the Asterisk cli you notice the following warning:

  ```
  Unable to send packet: Address Family mismatch between source/destination
  ```

  you can get rid of it by setting:

  ```
  enabled = no
  ```

  in the **hep.conf** file under the **/etc/asterisk** folder.