Loway

QueueMetrics

call center suite

QueueMetrics for FusionPBX

Loway SA

Version 25.09.3, 2025/11/18: covers QueueMetrics Hosted 24.11

Table of Contents

Introduction

FusionPBX is a very powerful open-source PBX that is able to run a large number of tenants on a single physical cluster. It is typically used to offer hosted telephony to small-to-medium businesses. A lot of these businesses are running internal helpdesk and support lines, or contacting their own customers in a systematic fashion, and are needing a way to keep track and rationalize what is going on. This is where the integration with QueueMetrics Live comes into play.

The goal of this integration is to support complete inbound, outbound, or blended contact centers, offering them all the tools they need to handle their day-to-day activities, including:

- A customizable **real time view** of what is going on, with the ability to interact with calls and log agents on and off;
- Very extensive **reports** that can be customized and sent automatically via email;
- An **agent page** that let agents do their work, including managing their status, handling inbound and outbound calls, connecting to an external CRMs and even handling automated dialing
- Access to **audio recordings** for inspection, quality tracking, and even **live monitoring/coaching**;
- An extensive set of **APIs** for the most demanding scenarios.

Feel free to start from the Quick start guide and come back later to the theory.

How it works – the big picture

The integration has multiple paths to produce all data that is needed, and that is:

- tracking inbound calls on queues
- tracking outbound calls
- executing commands on the PBX
- finding audio recordings
- splitting the data stream to each tenant.

For **inbound calls** that are handled on queues, FusionPBX relies on physical queues handled by FreeSwitch's `mod_callcenter`. Uniloader connects to the switching engine over ESL and listen to call events, that are then formatted and sent to QueueMetrics in real time.

Agents on FusionPBX queues have a **presence** – that is, a set of queues that they are supposed to be active on, and a general presence switch that decides whether they are available, on a temporary break or totally disconnected.

While in FusionPBX the set of queues that an agent may be working on is fixed by the administrator, with the QM we are able to decide different policy and then change the set of queues an agent is working on in real time – even allowing them to login and log off autonomously.

For **outbound calls**, we pretend that those calls happen on queues, even if they are not:

- All calls on the system are tracked as 'outbound'. You can set up a filter so that you capture only calls going to some class of numbers - e.g. only calls going to the PSTN.
- If you use the QM agent page, you can do outbound by selecting a number to dial and a campaign it belongs to. This way the activity an agent makes is correctly accounted to the reason why they were making it – for example, if they are selling tickets for different events, you can report separately on each specific event or on all events at once.

When an agent handles a call, be it inbound or outbound, they can set an **outcome** code for the call and possibly multiple **feature codes** so that it's possible to know how the call went from the reports.

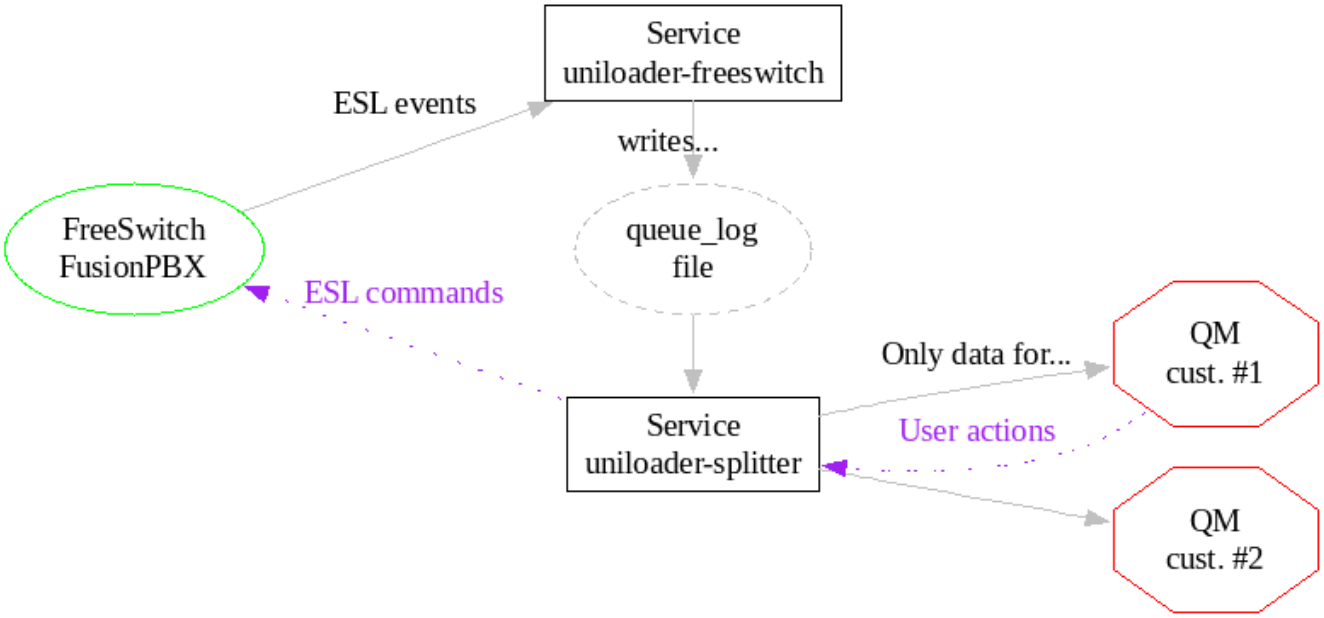
The supervisor can always see in real time what is going on. They can re-route or hang up calls as they best see fit, and they can listen to them in real-time or even join them. They can decide agent presence, moving them between different queues as is needed by the present workload.

FusionPBX may keep **audio recordings** of calls that were handled on the system; from the QM GUI you can listen to them and use them to score how the call was handled, in order to keep track of quality and compliance.

FusionPBX supports multiple tenants, and integration needs vary by tenant. Our approach is to create activity records for all significant system events, but we'll only send live data to tenants who have specifically requested these integrations. All events are captured and stored as activity records, but live data transmission occurs only for tenants with active integration requirements.

The integration in practice

Running QueueMetrics Live on FusionPBX requires you to install one single executable called Uniloader, set up services and define which FusionPBX tenants you want to activate QueueMetrics Live for.



The idea is quite simple:

- a service called `uniloader-freeswitch` gathers data from the PBX, rewrites it so that it is easier to

understand and dumps it to an events file called **queue_log**. It also takes care of applying any request from the QM GUI - for example an agent joining a queue or setting an outcome for specific call.

- a second service called `uniloader-splitter` forwards **queue_log** data to the correct QMLive instances, and takes care of executing on the PBX any action that users require though the QueueMetrics GUI - either directly or through `uniloader-freeswitch`
- a third (optional) service called `audiovault` allows secure access to call recordings that will be available through the QM interface.

All of these services rely on the same Uniloader executable.

Terminology

- **QueueMetrics Live** is the SaaS offering for QueueMetrics. It runs multiple separate **instances**, one for each tenant.
- **Uniloader** is a kind of Swiss army knife that lets you upload data, verify accesses and configuration and in general work with a QueueMetrics instance.
- **AudioVault** is a feature of Uniloader that allows QM to access remote audio files
- Access to the **Operator Portal**, or
- An **API KEY** to generate and manage QueueMetrics Live instances without going through Loway.

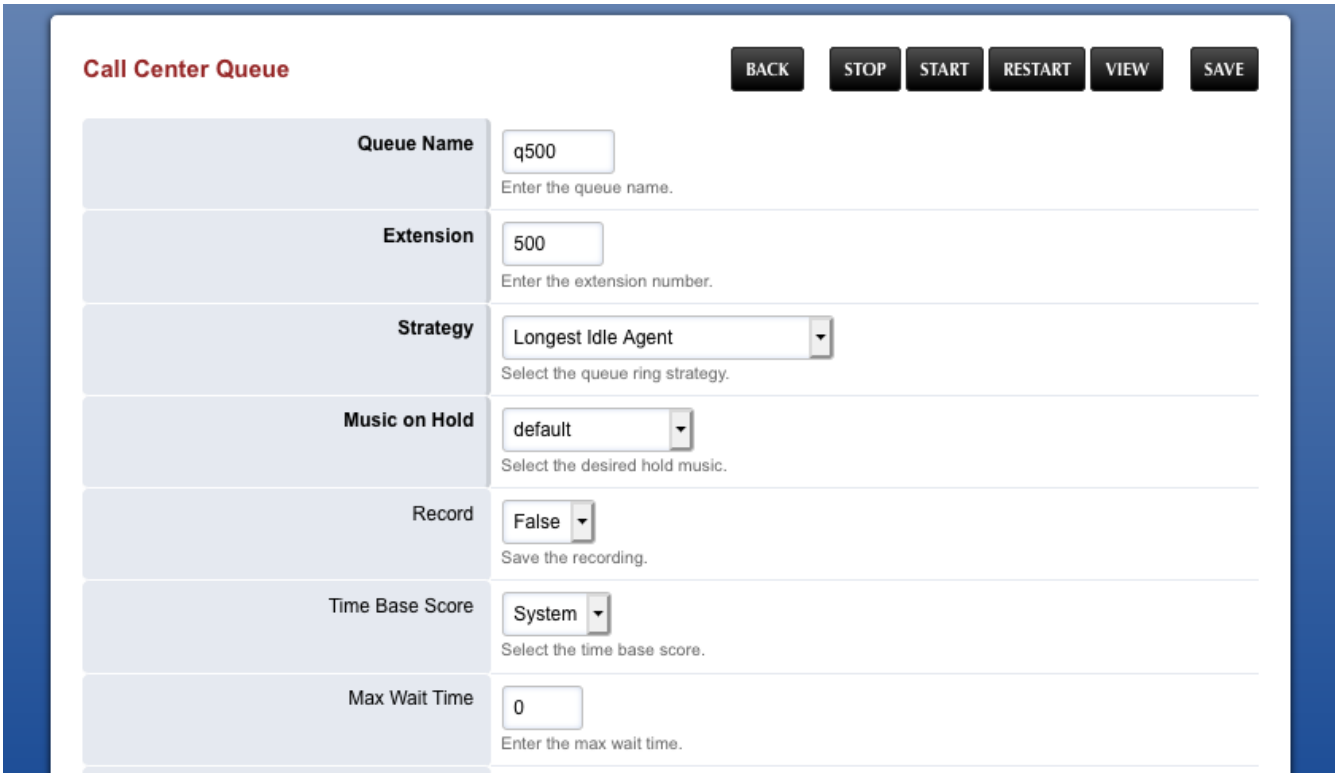
How FusionPBX is configured

For running QueueMetrics Live, you need to have a multi-tenant FusionPBX system, where each tenant runs their own sub-domain. This guide presumes that you already have a working FusionPBX system.

QueueMetrics works on data generated by the `mod_callcenter` module in FreeSwitch. It can be configured in the FusionPBX GUI under "Apps" → "Call centers".

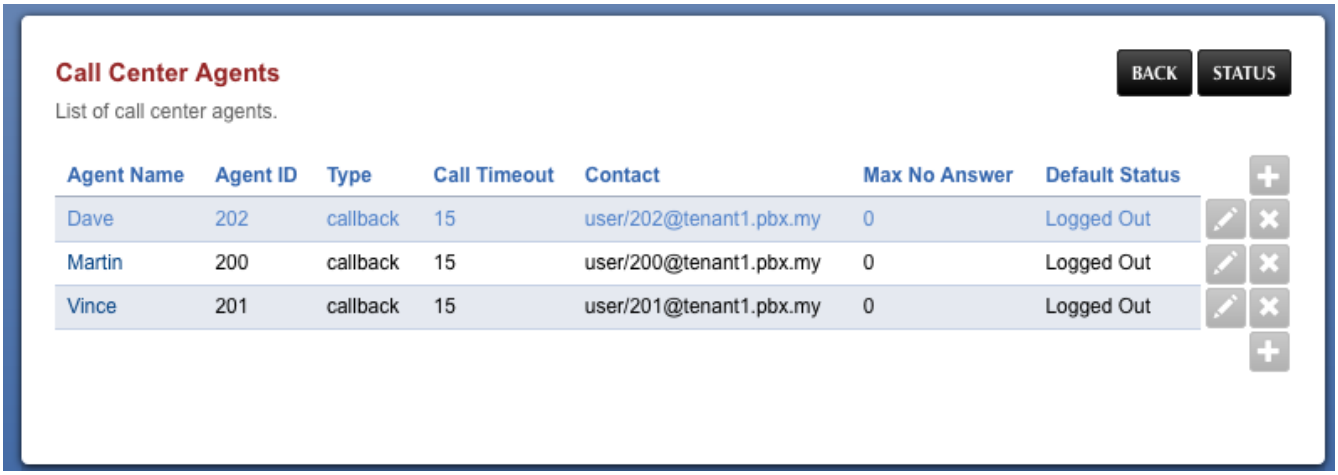
First, you need to have one or more queues defined:

To create them, you need to set a name, an extension to reach it on, and optionally a set of agents that are "known members" of that queue, in the **Tiers** section:

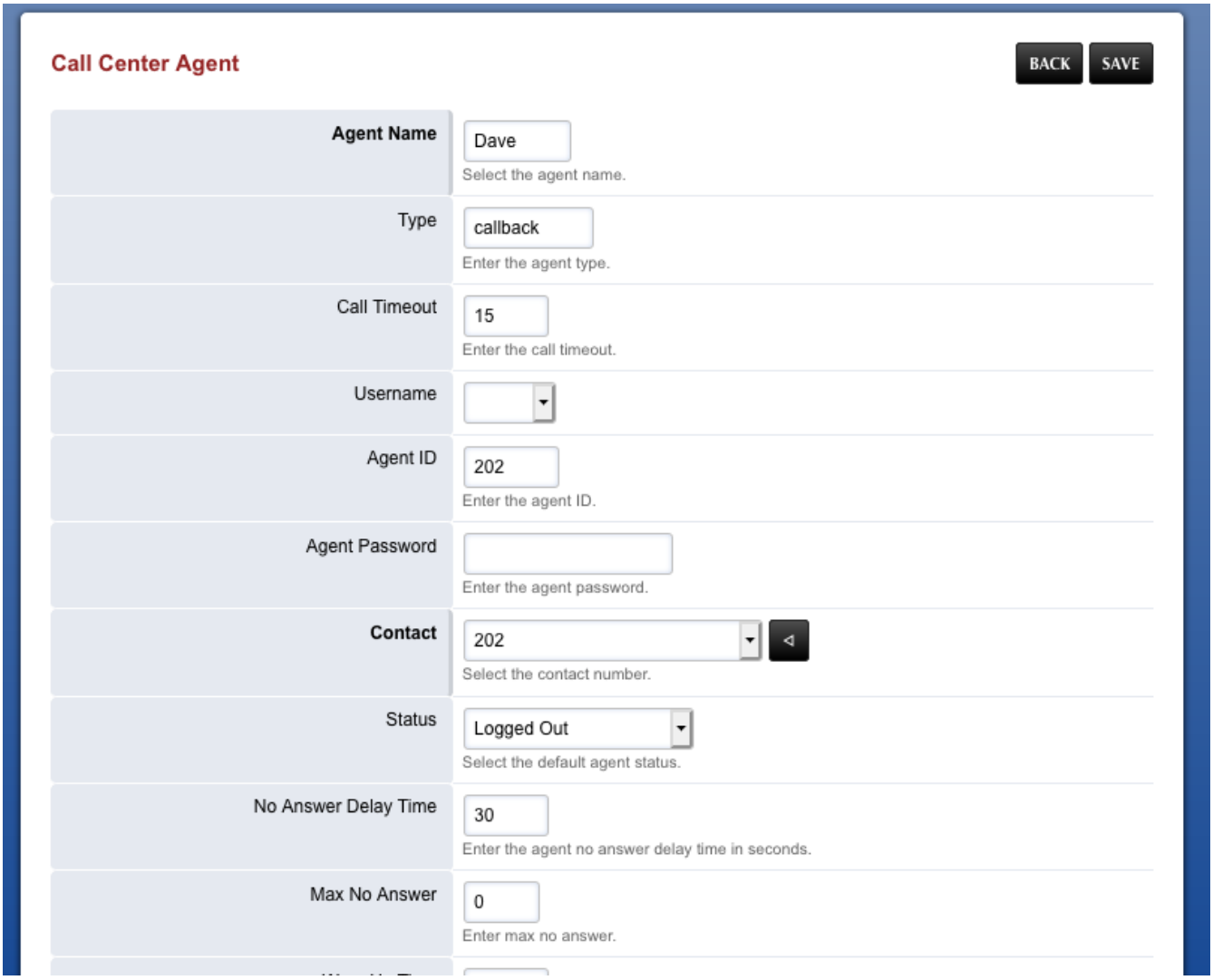


When done, make sure you click on Refresh so that the new configuration is started.

By clicking on the top-right "Agents" button, on the "Queues" page, you can create agents. Agents are defined as:



And each of them is set as:



You may want to add your agents to some queues, in their Tier sections, so that they are immediately available.

After you create new agents or queues, make sure QM knows about them - see Pre-configuring your QueueMetrics Live instance.

Prerequisites

- A working FusionPBX system version 5.3+, set up with multiple tenants, each of which has their own sub-domain of the format `tenant1.pbx.my`. When exporting data, we will not care about the domain name, but only the subdomain, that is `tenant1`.
- Direct access to the Postgres database that FusionPBX runs on
- Root access to the FusionPBX server
- A custom domain name to host your QueueMetrics Live, here `stats.pbx.my`
- A QueueMetrics Live API key (optional)

Quick start: a ready-made script

To use QueueMetrics Live with FusionPBX, it is necessary to install and configure some system services. These are described in detail in Appendix I, should you wish to perform a manual installation. However, the recommended method for creating and managing the integration is to use a self-contained script provided by Loway.

You can find it at https://github.com/Loway/OpenQueueMetricsAddOns/tree/master/ansible-fusionpbx

This script takes care of:

- Installing and optionally updating the required components, including the qm-fsw-ctl helper script.
- Configuring them correctly and starting or restarting them if the configuration changes
- Checking that the parameters are correct, for example, the instance or database credentials

The script works with both Debian and Rocky/CentOS/Red Hat, and in the simplest case, installs everything on the same machine where the PBX is located.

The script can be run through Ansible 2+

First Installation

For the system to function, it is necessary to have at least one instance to which the data will be loaded and one tenant from which to retrieve it.

General variables

At the beginning of the script `fsw_vars.yml`, there is a series of variables that need to be configured with the system credentials and any filters you want to apply to the tracking of outgoing calls.

First we set credentials and logins:

```
uniloader_version: "25.01.1"

# The ESL port on Freeswitch
fsw_host: "127.0.0.1"
fsw_port: "8021"
fsw_auth: "ClueCon"

# Credentials to connect to Postgres DB used by Fusion
fusion_db: "127.0.0.1/fusionpbx?sslmode=disable"
fusion_login: "fusionpbx"
fusion_pwd: "SomePassword"
```

Then we customize the general environment:

```
# Filters for outbound tracking
outbound_include_caller: ""
outbound_exclude_caller: ""
outbound_include_callee: "^9.+"
outbound_exclude_callee: ""

# AudioVault
# You have to escape all "%" characters to "%%"
audiovault: True
av_port: "localhost:4040"
av_public_url: "https://tenant1.server.com/audiovault"
av_path: "file:/var/lib/freeswitch/recordings/%%TE/archive/%%YY/%%ME/%%DD"
av_token: "CHANGEME"

# Autoconfiguration
# You need to have user "robot" enabled in QmLive
# To always force autoconfiguration, we just remove the contents of autoconfigdir
autoconfiguration: True
autoconfigure_always: False
autoconfigure_agent_pwd: "v3rys3cret"
default_domain: "mydomain.com"
```

In the example above:

- We install Uniloader 25.01.1
- We use the default ESL credentials
- We only include outbound tracking for calls starting with the digit 9 - the filter is a regular expression that you can tailor to match your dialplan.
- We install AudioVault
- When agents are configured in QueueMetrics, they can log to their agent page with login `agent/123` (where `123` is their agent code) and the password `v3rys3cret`
- All FusionPBX tenants share the domain `mydomain.com` (see below)

Describing all tenants

In the file `fsw_clients.yml`, for each tenant that you want to activate, you should configure its credentials. For each tenant in FusionPBX, enter their name and their attributes.

The name of the tenant is their domain name minus the `default_domain` part as defined above. For example, if your tenant is called *tenant1.mydomain.com* and the default domain is set to *mydomain.com*, then the tenant name is *tenant1*.

```
tenant1:
    url:   "http://my.queuemetrics-live.com/tenant1/"
    login: "webqloader"
    pass:  "APassword"
```

```
    actions: True
    disabled: False
    av_secret: "ASecretForAudiovault"
    refresh: ""
```

That's all you have to do. The script checks whether the instance is correctly configured, is working and able to receive data.

You don't have to define any tenants you don't want to activate. If you do, e.g. because someone wanted a trial and then changed their mind and you don't want to remove them in case they change their mind again, you can set them to `disabled: True` and they will be ignored.

Applying changes

It is also necessary to modify the `ansible-hosts` file to tell the script which server to connect to (or configure the local server). The script uses Ansible, which must be installed on the machine from which it is launched.

Once configured, simply run the command `./run.sh`.

If changes are made to the script, for example, by adding a new tenant or modifying a configuration, the script itself will take care of applying them.

Where should the integration be installed?

In simpler cases, if the system is not very large, it is possible to install the integration directly on the same PBX. In this case, access to the PBX and its database is very easy as they are already set up for local access from the same machine, and audio recordings are already in the correct folder.

For example, on a Debian 12 server you would do this:

- Install Git and Ansible: `apt-get update && apt-get install git ansible`
- Find a suitable working folder, e.g. /root
- Clone the repo: `git clone https://github.com/Loway/OpenQueueMetricsAddOns.git`
- Go to our Ansible script: `cd OpenQueueMetricsAddOns/ansible-fusionpbx`
- Edit *ansible-hosts* to enable the local connection: make sure the only uncommented entry is `localhost ansible_connection=local`
- Edit variables and instances
- Run everything with `./run.sh`

Next time you have to add/remove/change something, you just edit variables and/or instances and launch the script again.

If the PBX system is very busy, it is possible to install the services on a separate machine in the same data center. The advantage is that any operations will not affect the PBX; the disadvantage is that it is necessary to ensure that the ESL, the PostgreSQL database, and any folders containing audio

recordings are made accessible from this new server.

Planning your moves

The procedure below expects you to be able to create instances in QueueMetrics Live directly, as an Operator. If you are not an Operator, just ask Loway for instances to be created or deleted.

First run

- Decide on one or more tenants on the PBX for which you want to activate the QueueMetrics services.
- Enter the configuration page in FusionPBX and configure/create all the queues and all the agents that will be used. Note down how many agents you create - this will be the size of your instance. See details in Configuring FusionPBX.
- Create them in the QueueMetrics Operator Panel interface and note their addresses and passwords. Wait a few minutes so you can see them working.
- Enter the general connection and configuration information for the PBX in the first part of the script.
- Enter the configurations for each tenant (addresses and passwords).
- Run the script and wait for it to complete. Agents and queues as set in FusionPBX will be pre-configured in your QM instances.
- If you want to run AudioVault, set up an HTTPS proxy for it - see e.g. AudioVault under HTTPS - and set its secret in QueueMetrics.

Importing Configuration Changes

If you need new agents or new queues, you must first create them on the tenant. If they are created only in QueueMetrics, they cannot function correctly.

- If you create new agents, you will have to increase the number of licensed agents for their instance in the Operator Portal
- Edit the script to reflect the new configuration (if needed) and in any case set a new value in the tenant's "refresh" field
- Run the script again.

In a few seconds, all the settings will be recreated, and the information relating to the new agents and queues created will be associated with the corresponding tenant.

Activating a new tenant

- Create the new instance in the Operator's portal and wait for it to be active
- Add your new tenant to the configuration script
- Run the script again.

Deleting a tenant

- Remove the tenant from the script.
- Turn off the tenant's QueueMetrics instance from your Operator Panel.
- Run the script again.

Default users

When you create a new instance, you will have to enter a new password. We call that the default password and is set for all users created automatically.

We suggest that you keep it for users as their "first login" but tell them to change it immediately afterwards.

Pre-defined QueueMetrics user of interest:

- `demoadmin` (password *default*): an all-powerful administrator
- `webqloader` (password *default*): the user that receives new queue-log data. Not meant for interactive login.
- `robot` (password *default*): the user that creates and edits system configuration.
- `agent/xxx` where xxx is the Agent-ID in FusionPBX: an agent - the password will be the one specified in Ansible as `autoconfigure_agent_pwd`.

qm-fsw-ctl

The `qm-fsw-ctl` utility provides administrative commands for managing Uniloader services, tenants, and diagnostics.

When installing via Ansible, the script `qm-fsw-ctl` is installed as well; it helps with a number of manual administrative tasks. It is immediately available after Ansible runs; and will print extensive help if invoked directly.

Usage

```
qm-fsw-ctl [tenants|sync|dumpstate|svcstate|trace]
qm-fsw-ctl [logs|qlog]
```

qm-fsw-ctl tenants

Prints all configured domains and the current size of their recordings.

Example

```
# qm-fsw-ctl tenants

Tenants configured on this server:

1.2G    ./abc.xy
2.8M    ./def.xy
```

qm-fsw-ctl sync

Outputs the current presence state (all agents on all queues) to the queue_log.

Example

```
# qm-fsw-ctl sync

Syncing queue information
2025/11/18 14:05:43 Asking for a queue refresh on 127.0.0.1:8021 with pass '*******'
2025/11/18 14:05:43 Event was sent in 0 ms - all went well
```

You will see all presence state rewritten to the queue_log file.

qm-fsw-ctl dumpstate

Prints the current inner state of service `uniloader-freeswitch`. As that is written on the service journal, it shows the journal. Press Ctrl-C to stop.

Example

```
# qm-fsw-ctl dumpstate

Dumping inner state of daemon uniloader-fusion
2025/11/18 14:06:35 Asking Uniloader Fsw on 127.0.0.1:8021 with pass '*******' to log
its status
2025/11/18 14:06:35 Event was sent in 0 ms - all went well

Nov 18 14:06:35 svr uniloader[569941]: 2025/11/18 14:06:35 ========== Γν θι σεαυτ ν -
NOSCE TE IPSVM ==========
Nov 18 14:06:35 svr uniloader[569941]: 2025/11/18 14:06:35 Open Calls: 14 - Open
agents: 145 (waiting for tiers: 0)
Nov 18 14:06:35 svr uniloader[569941]: 2025/11/18 14:06:35 Ag:04c76072-01a0-4b5e-91a6-
cbedc05c4e82 C:user/2002@abc.xy Logged Out,Waiting Qs:NONE
....
Nov 18 14:06:35 svr uniloader[569941]: 2025/11/18 14:06:35 C:fe84a(mc)-Q=q-
outbound@abc.xy-A=-B=83843-n:
Nov 18 14:06:35 svr uniloader[569941]: 2025/11/18 14:06:35 ----------------  That's
all folks ----------------
```

qm-fsw-ctl svcstate

Prints the current state for all services.

Example

```
 uniloader-freeswitch.service - Loway Uniloader 25.09.3 - FreeSwitch
     Loaded: loaded (/etc/systemd/system/uniloader-freeswitch.service; enabled;
preset: enabled)
     Active: active (running) since Tue 2025-11-18 10:40:10 UTC; 5h 43min ago
   Main PID: 2354555 (uniloader)
      Tasks: 4 (limit: 4531)
     Memory: 5.0M
        CPU: 166ms
     CGroup: /system.slice/uniloader-freeswitch.service
             └──2354555 /usr/bin/uniloader fsw ....

Nov 18 16:05:16 fusion2 uniloader[2354555]: 2025/11/18 16:05:16 Uniloader 25.09.3 (0-
20251118.1037) up - GR: 3 - Mem: Alloc 666k (Free 0k) Sys 6743k
....

 uniloader-splitter.service - Loway Uniloader 25.09.3 - Splitter
     Loaded: loaded (/etc/systemd/system/uniloader-splitter.service; enabled; preset:
enabled)
     Active: active (running) since Tue 2025-11-18 10:40:11 UTC; 5h 43min ago
   Main PID: 2354619 (uniloader)
      Tasks: 5 (limit: 4531)
     Memory: 8.3M
        CPU: 7min 54.911s
```

```
     CGroup: /system.slice/uniloader-splitter.service
             └─2354619 /usr/bin/uniloader .....

Nov 18 16:12:01 fusion2 uniloader[2354619]: 2025/11/18 16:12:01
[http://116.203.46.244:8080/queuemetrics/] Driver error: retrying in 300000 ms
....

 uniloader-audiovault.service - Loway Uniloader 25.09.3 - AudioVault
     Loaded: loaded (/etc/systemd/system/uniloader-audiovault.service; enabled;
preset: enabled)
     Active: active (running) since Tue 2025-11-18 10:40:13 UTC; 5h 43min ago
   Main PID: 2354680 (uniloader)
      Tasks: 4 (limit: 4531)
     Memory: 2.9M
        CPU: 167ms
     CGroup: /system.slice/uniloader-audiovault.service
             └─2354680 /usr/bin/uniloader av serve ....

Nov 18 15:35:19 fusion2 uniloader[2354680]: 2025/11/18 15:35:19 Uniloader 25.09.3 (0-
20251118.1037) up - GR: 2 - Mem: Alloc 539k (Free 0k) Sys 6487k
....
```

qm-fsw-ctl trace

Creates trace files to be sent to Loway (events and logs). This can be run in parallel to an existing `uniloader-freeswitch` server, so you can run it on a full production server without interrupting operations.

The command displays where the trace files will be saved.

Once the trace is over, press Ctrl-C to interrupt.

Though you do not need to stop and start the `uniloader-freeswitch` server, make sure that there are no external calls running apart from the test you want to make, as logs are extremely verbose and hard to understand.

Example

```
# qm-fsw-ctl trace

Now running event trace files:
 - /opt/fusion-splitter-data/traces/evt_251118-1408_trace.txt
 - /opt/fusion-splitter-data/traces/evt_251118-1408_qlog.txt

Press Ctrl+C to terminate.

2025/11/18 14:08:54 Uniloader 25.09.3 (0-20251118.1037) up - GR: 2 - Mem: Alloc 508k
(Free 0k) Sys 6743k
2025/11/18 14:08:54 Starting : FS: 127.0.0.1:8021 (auth '*******') - Qlog:
```

```
'/opt/fusion-splitter-data/traces/evt_251118-1408_qlog.txt' Events: '/opt/fusion-
splitter-data/traces/evt_251118-1408_trace.txt' - DB uri:
'127.0.0.1/fusionpbx?sslmode=disable' user: 'fusionpbx' pass:
'************************' - Shorten? true Addmembers? false - Flt Outbound: Caller
All, Callee All
2025/11/18 14:08:54 Reading all events from ESL - Do not use in production
2025/11/18 14:08:54 ========== Now syncing all agents ===========
2025/11/18 14:08:54 [Cache] Searching for agent: 04c76072-01a0-4b5e-91a6-cbedc05c4e82
2025/11/18 14:08:54 [Cache] Searching for agent: 0803f46d-b51f-4b9a-810f-588493ae2fa1
2025/11/18 14:08:54 [Cache] Searching for agent: 09213a07-aeac-4f13-927b-1d52d6b47bcf

^C
```

When done, gather the files in `/opt/fusion-splitter-data/traces` and send them over, complete with a detailed description of what was tested.

For more details, see this FAQ.

Log Files

qm-fsw-ctl logs

Tails the aggregate logs for services `uniloader-freeswitch` and `uniloader-splitter`. Press Ctrl-C to stop.

Example

```
# qm-fsw-ctl logs

Tailing current state of services

Nov 18 14:06:35 svr uniloader[569941]: 2025/11/18 14:06:35 C:83843(ma)-Q=q-
outbound@abc.xy-A=Agent/27875638048@abc.xy-B=fe84a-n:
```

qm-fsw-ctl qlog

Tails the aggregate queue_log file for all tenants. Press Ctrl-C to stop.

Example

```
# qm-fsw-ctl qlog

Tailing aggregate queue_log file at /opt/fusion-splitter-data/queuelog-synth-fsw.txt

1763474993|f246360e-1682-4f1d-af7d-d5ccc4cb675a|abc-q-outbound|Agent/abc-
27875638075|CONNECT|0||||
1763474998|ul.3912e|abc-q-outbound|Agent/abc-27875638080|ADDMEMBER|||||
```

The QueueMetrics' Agent Page

If you want, you can have your agents log into QueueMetrics and use their own Agents Page to:

- **Log-in and log-off** from their queues
- **View in real-time** the calls they are handling
- Have a **CRM automatically open** when they answer a call
- Start **outbound calls**
- **Pause and unpause** themselves, setting their pause code so the supervisor can see it (e.g. **Lunch** versus **Back office**)
- Set **call outcomes**
- Receive **messages** from their supervisors
- Run a limited set of **self-service reports** to see their own statistics in real-time.

Setting up the Agent's page

The agent's page is already set up in QueueMetrics Live. If you use the autoconfiguration option together with the Ansible script to manage your system, it will work immediately.

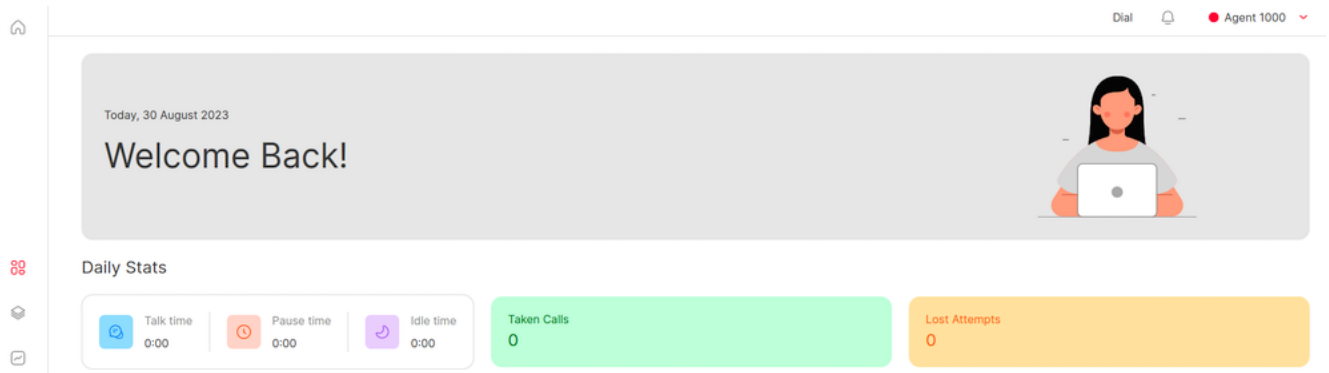
Logging on as an administrator, you can open up the configuration bar and select "Users" to see all agents configured in your system - their login starts with "Agent/" and they belong to class `AGENTS`. Their password is the default one you set up in the Ansible script, but you can change it.

If you add an e-mail address to the User, they can log in using the e-mail address and their password without having to remember a separate set of credentials. QueueMetrics also supports SSO using Microsoft or Google as identity providers.

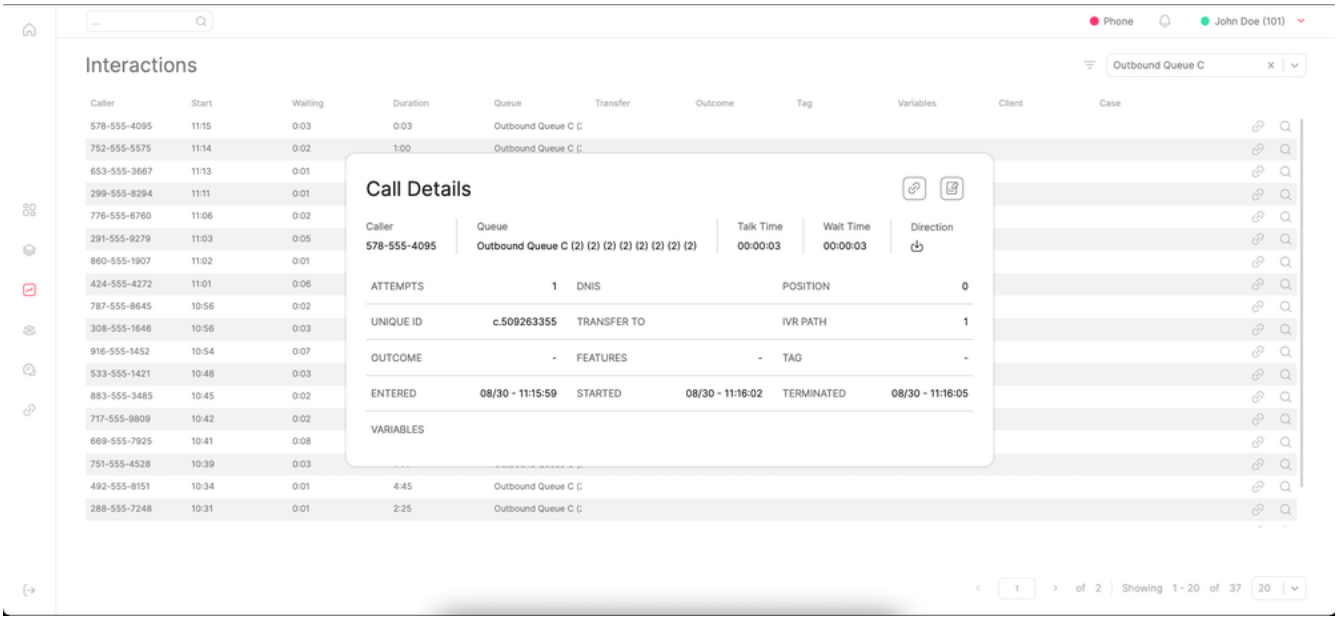
If you create agents and queues manually, and want agent actions, remember to set the "External reference ID" field with the unique-id used internally by FusionPBX to identify each resource. This happens automatically with Uniloader.

Using the Agent's page

An agent logs in from the main page of QueueMetrics Live with their own credentials, usually in the form "Agent/XXX" or (if you set it up on their User) their e-mail address.



From there they can perform all of their tasks, for example, pictured below you can see the list of calls just processed and the details of the current call:



The QueueMetrics User Manual contains more information on how to use the Agent page for your users.

Tracking Outbound Calls

QueueMetrics typically tracks only calls occurring within call queues. Since queues are used to manage inbound traffic, outbound calls (for example, callbacks for missed calls or external notification calls) are not usually monitored. To address this, the outbound call tracking feature has been introduced.

QueueMetrics offers two outbound call tracking methods: automatic (passive) and manual (active). Both are implemented by the `audiovault fsw` daemon.

	Automatic	**Manual**
Tracked calls	All calls	Calls made by QM
Exclusion filters	On caller and called number	No
Logged on queue	`q-outbound`	As selected by the agent
Agent member of queue?	No	Yes (advisable)
Extra features	-	Outcomes, Features, AMO
Configuration	In svc `uniloader fsw`	In QueueMetrics

Automatic outbound tracking

The passive outbound call tracking mode (also known as automatic), as the name says, requires no user intervention. All calls of any type occurring between extensions or to external numbers are automatically included. As QueueMetrics monitors calls within queues, these passive outbound calls are logged "as if" they occurred on an inbound queue named `OUTBOUND`. Callers are automatically added to this queue before the call starts, and leave them soon after the call is finished.

Generally, it is not always desirable for all calls happening on a PBX to be visible as a single pool. Therefore, QueueMetrics allows you to specify filters based on the calling and called numbers. Those enable you to track only the calls of interest - usually these are calls made to external lines, which, depending on the PBX system, often start with "0", "1" or "9". This allows you to avoid tracking all other internal calls. Furthermore, you can set specific exclusion filters on originating numbers, such as excluding calls made by the executive staff.

Manual outbound tracking

Manual outbound calls, on the other hand, are designed to be initiated from the QueueMetrics agent page. Agents must select an outbound campaign and enter the phone number to call. Their extension will ring, and upon answering, the actual outbound call will begin.

Because these calls require specifying an outbound campaign, it is possible to track the reason for the calls more precisely. For instance, calls made for different company departments or for different purposes can be separated on different campaigns. Additionally, agents can set supplementary call information (call outcomes and call features) through their dedicated page.

These calls can also be automated using a QueueMetrics feature called **Assisted Manual Outbound** (AMO), which allows a group of agents to efficiently go through various lists of numbers to call and recall. This is in effect a manual dialer embedded within QueueMetrics that allows a pool of agents to process outbound campaigns.

For optimal visibility of manual outbound calls, it is advisable to create regular queues in the system (i.e., inbound queues) to be used as a roster of available agents. This way, supervisors can easily see all agents available for making outbound calls, just as they would see agents available for receiving inbound calls.

Configuring automatic outbound

- You need to set up the correct traffic filters on `audiovault fsw`, by setting the appropriate regular expressions. For example, `^0.+` means "all calls starting with 0". You can use patterns to include or exclude just some calls, matching either the caller or the callee number.
- No changes to be made on QueueMetrics

Configuring manual outbound

- No changes to be made on `audiovault fsw`
- In QueueMetrics, you need to specify the `platform.freeswitch.agentChannel` and `platform.freeswitch.destinationNumber` to generate the call. It is mandatory that the agent channel has the variable `qm_queue` set to the destination queue. == AudioVault

AudioVault is a software solution that securely exposes call recordings stored on the server, allowing them to be accessed and played back within the QueueMetrics interface. Consequently, when a call detail is selected in QueueMetrics, the associated call recordings can be listened to. Furthermore, it is possible to link not only recordings but also other relevant materials, such as transcriptions, to the call record. Based on this information, an analyst can complete an evaluation form within QueueMetrics to assess call quality (QA form). This enables the evaluation of work performance and the identification of critical areas affecting the entire staff or specific agents.

The operational flow is as follows - when a supervisor clicks on the details of a call within QueueMetrics...

- A secure HTTPS request is sent from QueueMetrics to AudioVault.
- AudioVault verifies the credentials, performs a search, and if one or more relevant files are found, it generates a unique secure download URL for each file, valid for a short period.
- The supervisor's browser downloads the content from the temporary URL.

Securing AudioVault

AudioVault needs to be secured before its use - see AudioVault and HTTPS.

Proxied audio access.

In the default setup, AudioVault needs to be accessible via a public address, because audio files are downloaded directly by the browsers of QueueMetrics users.

For more secure installations, QueueMetrics can act as a proxy, thus concealing the physical address of the AudioVault system from external access:

```
mediaproxy.enabled=true
mediaproxy.linkvalidity=1000
mediaproxy.header=NONE
```

This way, all requests will go through QueueMetrics and will be reported on the system log.

If you are a QueueMetrics Live operator, we can set up an outbound proxy so you can secure accesses to AudioVault only from a given IP address.

System settings for QueueMetrics instances

To make everything work smoothly, your QueueMetrics Live system must be configured to work with FusionPBX/Freeswitch.

Make sure that all the settings are present.

Some - or all - of these configurations can be pre-set in the default image configuration if you are a QueueMetrics Live Operator, so you don't have to do it manually on each instance.

Users

The user `robot` is created and enabled with the default password so you can synchronize the configuration automatically via Uniloader.

On an on-prem system, you need to manually enable users `webqloader` and `robot`. For the latter, make sure that the user has the security keys `USR_QUEUE USR_AGENT USRADMIN` by adding them to the "User keys" box.

If not already present, add an outbound queue called `q-outbound` if you want to use automatic outbound tracking.

Example complete configuration

```
platform.freeswitch.tenant=tenant1.server.com
callfile.dir=fsw:ClueCon@127.0.0.1:8021

default.hotdesking=0
platform.pbx=FREESWITCH_LIVE

platform.freeswitch.use_external_ref=true
platform.freeswitch.verbose=true
platform.freeswitch.addmember=true
platform.freeswitch.use_external_ref=true
platform.freeswitch.agentChannel={qm_queue=${q},origination_caller_id_number=1973,orig
ination_caller_id_name=QM-${q},enable_early_media=true}user/${num}
platform.freeswitch.destinationNumber=${num}
platform.freeswitch.spychannel={qm_ignore=1,origination_caller_id_name=Spy_q${q}_q${a}
}user/${num}
platform.freeswitch.spycmd=queue_dtmf:w${spymode}@500,eavesdrop:${callid}
```

Please note that the property `callfile.dir` points to the address of the ESL as seen from the machine where `uniloader-splitter` runs, while `platform.freeswitch.tenant` is the complete tenant name as it appears in FusionPBX.

Additional configuration for AudioVault

```
audio.server=it.loway.app.queuemetrics.callListen.listeners.JsonListener
audio.jsonlistener.url=https://fusionpbx.company.my/audiovault/search/?tenant=tenant1.
server.com
audio.jsonlistener.method=POST
audio.jsonlistener.searchtoken=x1x1secret
audio.jsonlistener.verbose=false
audio.html5player=true
```

Please note that the property `audio.jsonlistener.url` should point to the server running AudioVault, with the URI parameter "tenant" set to the complete tenant name as it appears in FusionPBX. The parameter `audio.jsonlistener.searchtoken` must be configured as it is given for that tenant in AudioVault.

Adding your logo

This setting will include a custom logo, width 40px tall x 150px across, with transparent background, hosted on your own servers:

```
layout.logo=https://your.company/logo_qm.png
```

Suggested image formats are PNG or SVG.

Further customizations

You can discuss further customizations, including full white-labelling, with our Support team.

Appendix I: Manual service installation

While the section below explains how data export, data splitting and automatic configuration work, by showing how these services are configured, your life will be easier if you designate a small VM separate from the main PBX to do these jobs, and then use the Ansible scripts we provide to run and manage all these tasks for you. See the Quick-start Guide.

Setting up the services

We will be creating two services on the FusionPBX box; one of them `uniloader-freeswitch` downloads data continuously from FreeSwitch and generates a queue_log file; and the other one `uniloader-splitter` knows which tenants are active (based on a file you specify) and uploads data to all of them.

All commands below need to be run as `root`.

Installing Uniloader

We will install Uniloader as a symlink under `/opt/uniloader`, so that it is always in the same place even when it is upgraded. We will also add it to the system path so that it's available from any shell.

The link for the current version of Uniloader is available from https://www.queuemetrics.com/download.jsp

```
mkdir -p /opt/uniloader
cd /opt/uniloader
wget https://downloads.loway.ch/qm/uniloader-25.09.1.tar.gz
tar zxvf uniloader-25.09.1.tar.gz
ln -s ./uniloader-25.09.1/bin/uniloader_amd64 uniloader
ln -s /opt/uniloader/uniloader /usr/bin/uniloader
```

Now we create a folder to store **queue_log** data into:

```
mkdir -p /opt/uniloader/qlogs
```

Sanity checks

In order to continue, you will need accesses to your FusionPBX / FreeSwitch system. Uniloader includes ways to test and debug connections, so we will use it to assert everything is okay.

Checking ESL access to FreeSwitch

You can make sure that local access to FreeSwitch's ESL port is allowed by typing:

```
# uniloader test fsw-esl --host "127.0.0.1" --port "8021" --auth "ClueCon"
```

```
2020/02/04 06:43:19 Testing Freeswitch connection to '127.0.0.1:8021' with auth token
'ClueCon'

2020/02/04 06:43:19 <ESL: Content-Type: command/reply
2020/02/04 06:43:19 <ESL: Reply-Text: +OK accepted
2020/02/04 06:43:19 <ESL:
2020/02/04 06:43:19  ======= Login OK
....
```

If you see the **Login OK** message, then you are set. If not, you will have to check ESL access credentials and allowed IPs.

Checking access to FusionPBX's database

FusionPBX generates a random password for the database on installation. To find the password for your FusionPBX database, print the contents of your file `/etc/fusionpbx/config.php`:

```
//pgsql: database connection information
    $db_host = 'localhost'; //set the host only if the database is not local
    $db_port = '5432';
    $db_name = 'fusionpbx';
    $db_username = 'fusionpbx';
    $db_password = 'UELfQn8gWg6bp6SRMoOhwqZ34F0';
```

Now test that it's working by typing:

```
# uniloader test postgres --ps-uri "localhost/fusionpbx" --ps-login "fusionpbx" --ps
-pwd UELfQn8gWg6bp6SRMoOhwqZ34F0
2020/02/04 06:49:04 Testing Postgres connection to
'postgres://fusionpbx:UELfQn8gWg6bp6SRMoOhwqZ34F0@localhost/fusionpbx'

2020/02/04 06:49:04  -- Connection took 13.287µs
2020/02/04 06:49:04  -- Query took 11.365098ms
2020/02/04 06:49:04 Local time on database is: 2020-02-04T06:49:04.033431+01:00
```

If it prints out the local time, it's working.

Checking data upload to an instance

We first want to make sure that the QueueMetrics Live instance is available and that our password is correct:

```
# uniloader test upload --uri  https://stats.pbx.my/tenant1 --pass 12345678
Testing upload credentials.

High Water Mark is 1580741450 [2020-02-03 15:50:50 +0100 CET]
```

```
Connection OK
```

On a brand new system, you will see the High Water Mark set to zero.

Checking automatic configuration of FusionPBX's agents and queues

We then want to make sure that our tenant is available in FusionPBX:

```
# uniloader pbxinfo fusionpbx --ps-pwd UELfQn8gWg6bp6SRMoOhwqZ34F0

=  Tenant # 1: tenant1.pbx.my

-- Queues found: 3
   #            Code            Name                         Ext.Ref.
   1           00all          00 All
   2             500            q500 64032aac-f3b2-474b-a2c3-08f1a35ea16a
   3             501            q501 53b66cc6-51b7-43b9-84ef-22d44fc46f69

-- Agents found: 3

   #            Code            Name                         Ext.Ref.
   1       Agent/200          Martin 391770f4-aaad-4834-9050-0770a32782d0
   2       Agent/201           Vince 6d26623c-e5a9-46a1-93b3-937389ca8a02
   3       Agent/202            Dave 64eab4f9-829a-4fb8-8cbc-d2e6a5443fd6
```

If you see any agents/queues appearing, it means that the tenant will generate data and can be uploaded.

The number of agents you see here is a good guess about the required size of your QueueMetrics Live instance.

Service uniloader-freeswitch: log generation

We will create a systemd service to export data out of the PBX and format it in a way that is compatible with QueueMetrics.

If this service is not active, queue data generated on the switch **will be lost**. So you should never stop (or even restart) this service while the PBX is running; and if you do, you should do it when there is no traffic on your PBX.

Create a new file `/etc/systemd/system/uniloader-freeswitch.service`.

```
#
# Systemd Unit for Uniloader FreeSwitch ESL Export
#

[Unit]
```

```
Description=Generates queue_log from ESL events
After=syslog.target network.target freeswitch.service

[Service]
Type=simple
#EnvironmentFile=/etc/uniloader-freeswitch
Nice=15
KillMode=process
PIDFile=/var/run/uniloader-freeswitch.pid
ExecStart=/opt/uniloader/uniloader fsw  --queuelog /opt/uniloader/qlogs/qlog-synth.txt
--ps-pwd UELfQn8gWg6bp6SRMoOhwqZ34F0  --shorten-domain "1"
RestartSec=1
Restart=on-failure


[Install]
WantedBy=multi-user.target
```

We now run:

```
systemctl daemon-reload
systemctl start  uniloader-freeswitch
systemctl enable uniloader-freeswitch
```

This way the service is started immediately and will restart on boot, after Freeswitch itself starts.

To check if the system is working correctly, run:

```
# systemctl status  uniloader-freeswitch
- uniloader-freeswitch.service - Generates queue_log from ESL events
   Loaded: loaded (/etc/systemd/system/uniloader-freeswitch.service; disabled; vendor
preset: enabled)
   Active: active (running) since Tue 2024-02-04 07:01:36 CET; 6s ago
```

Now, if there are any agents defined, their current status will be written on `/opt/uniloader/qlogs/qlog-synth.txt`. When any call of interest starts, you will see activity.

Filtering outbound calls

You can control which calls should be included in the automatic outbound ones. This lets you exclude specific extensions from tracking (e.g. the CEO's) or just include some exact routes (e.g. only calls where the callee starts with the digit 0).

See Configuring automatic outbound.

Log rotation

While not technically necessary, we suggest setting up weekly or monthly log rotation. As restarts on the `uniloader-splitter` service will have to do a full rescan of the current file, it would definitely be advisable to keep its size to a manageable amount. Long rescans will cause no data losses, but real-time operations on instances will be delayed.

There is no need to do anything but rename the current `queue-log-synth.txt` file to a new file. The splitter will finishing uploading the old file before moving ahead to the new file.

The Ansible installer sets up a weekly rotation for you, with 50 weeks of data retention on the server..

Service uniloader-splitter: Setting up data export

We now have to define a service for data export. For now, let's imagine we have no tenant, so that it has nothing to do. Our service will read data from the `queue_log` file generated by `uniloader-freeswitch`, check each entry against a set of rules that specify our known tenants we want to feed, and do nothing because we have none so far.

We first create the rules file; edit a new file under `/opt/uniloader/splitter.json` that for now just contains two square brackets, as below:

```
[]
```

Create a new file `/etc/systemd/system/uniloader-splitter.service`.

```
#
# Systemd Unit for Uniloader Splitter
#

[Unit]
Description=Loway Uniloader Splitter
After=syslog.target network.target


[Service]
Type=simple
#EnvironmentFile=/etc/uniloader-splitter
Nice=15
KillMode=process
PIDFile=/var/run/uniloader-splitter.pid
ExecStart=/opt/uniloader/uniloader -s /opt/uniloader/qlogs/qlog-synth.txt  upload
--splitter /opt/uniloader/splitter.json  --pid /var/run/uniloader-splitter.pid
RestartSec=1
Restart=on-failure
```

```
[Install]
WantedBy=multi-user.target
```

We now install the service as:

```
systemctl daemon-reload
systemctl start  uniloader-splitter
systemctl enable uniloader-splitter
```

To make sure it is working correctly, and see its logs when needed:

```
# systemctl status uniloader-splitter
● uniloader-splitter.service - Loway Uniloader Splitter
   Loaded: loaded (/etc/systemd/system/uniloader-splitter.service; disabled; vendor
preset: enabled)
   Active: active (running) since Tue 2020-02-04 07:19:27 CET; 2s ago
 Main PID: 3590 (uniloader)
    Tasks: 6 (limit: 4915)
   CGroup: /system.slice/uniloader-splitter.service
           └─3590 /opt/uniloader/uniloader -s /opt/uniloader/qlogs/qlog-synth.txt
upload --splitter /opt/uniloader/splitter.json --pid /var/run/uni

Feb 04 07:19:27 debian systemd[1]: Started Loway Uniloader Splitter.
Feb 04 07:19:27 debian uniloader[3590]: 2020/02/04 07:19:27 Uniloader 0.6.7 (149-
20191004.1255) starting upload [PID 3590].
Feb 04 07:19:27 debian uniloader[3590]: 2020/02/04 07:19:27 Uniloader 0.6.7 (149-
20191004.1255) is alive - GR: 3 - Mem: Alloc 321k (Free 0k) Sys 68
```

Editing the tenant rules

Our splitter file contains multiple JSON object that tell Uniloader how to match log entries to known QueueMetrics Live instances and where to send them. While the **queue_log** file contains information for all queue activity on the system, you will only be sending out data for tenants you have configured.

In our example:

```
[
    {
        "uri": "https://stats.pbx.my/tenant1",
        "login": "webqloader",
        "pass": "12345678",
        "token": "",
        "matcher": ["tenant1-"],
        "match": "any",
        "removematch": true,
        "disabled": false,
```

```
        "noactions": false,
        "clientname": "tenant1"
    }
]
```

Make sure that:

- `uri` and `pass` are the ones you checked with QueueMetrics Live
- `matcher` is the subdomain you use for your tenant, plus a dash.
- `clientname` is a free comment
- you can set `disabled` to `true` if you want to "switch off" data upload for a tenant without deleting it from this file.

When done, just restart the service so that changes are picked up.

```
systemctl restart uniloader-splitter
```

While the service is restarted (and even if it stays down for a few seconds), no data is being lost, as data is still being written by the `uniloader-freeswitch` service and stored safely into the **queue_log** file. The worst thing that can happen are a few seconds of delay on the wallboards - but data is always computed correctly and will sync automatically.

The splitter file should be backed up after each change.

Checking that everything is working

Try sending a call to a queue on your tenant. Answer the call, then hang it up.

Now log in to your QueueMetrics Live instance; from the Home Page click on "Administrative Tools" → "System Diagnostic Tools" → "Live DB Inspector": you should see a number of records appear on the page.

Pre-configuring a QueueMetrics Live instance

You should now run an automated configuration of the QueueMetrics Live instance:

```
uniloader pbxinfo --mode syncqm -u https://stats.pbx.my/tenant1 -p 12345678
     fusionpbx -ps-pwd UELfQn8gWg6bp6SRMoOhwqZ34F0 --single-tenant tenant1.pbx.my
```

This will:

- Create queues
- Create agents
- Set the correct FusionPBX IDs for agents, so they can be controlled through QueueMetrics

You should run this command whenever the configuration changes on FusionPBX - you create new queues or new agents, so everything stays in sync.

Service uniloader-audiovault

Create a new unit file called `/etc/systemd/system/uniloader-audiovault.service`:

```
[Unit]
Description=Loway Uniloader  AudioVault
After=syslog.target network.target


[Service]
Type=simple
Nice=15
KillMode=process
ExecStart=/usr/bin/uniloader av serve \
              --bind-to "localhost:4040" --public-url "https://av.server.com" \
              --path '{{ av_path }}' \
              --token "n0b0d4kn0ws" --tenants "/opt/uniloader/audiovault-tenants.json"
RestartSec=1
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

Then create a config file called `audiovault-tenants.json` that defines all tenants and their security token:

```
[
    {
      "tenant": "tenant1.server.com",
      "secret": "x1x1secret",
      "note":   "Customer A"
    },
    {
      "tenant": "tenant2.server.com",
      "secret": "x2x2secret",
      "note":   "Customer B"
    }
]
```

AudioVault Configuration

To configure the system, it is necessary to specify an address to which AudioVault will be bound. Typically, a proxy server handling secure connections is placed in front of this address. It is possible to modify the FusionPBX configuration to act as a proxy for AudioVault itself.

For each tenant, a unique secret key is defined for authentication. This key must be entered into the QueueMetrics instance configuration for that tenant, and on QueueMetrics itself.

FusionPBX systems typically store queue recordings organized by tenant, year, month, and day. This configuration needs to be communicated to AudioVault so that it can efficiently locate the files, especially when an external storage is used for recordings and files are periodically moved to this storage. AudioVault can be instructed to search in multiple locations if the recording is not found in the primary specified location.

An example configuration follows:

```
audiovault: True

av_port: "localhost:4040"
av_public_url: "https://server.com/audiovault"
av_path: "file:/var/lib/freeswitch/recordings/%%TE/archive/%%YY/%%ME/%%DD"
av_token: "CHANGEME"
```

In this configuration:

- AudioVault is enabled and installed
- AudioVault binds in HTTP to port 4040, available only on the same server. A proxy needs to be set up to allow external access, to the address "server.com/audiovault"
- AudioVault needs to know the public URL of the proxy - in this case `https://server.com/audiovault`
- A search path is defined - placeholders will be expanded so that the actual path for a recording made on May 8, 2025 for tenant1 might be `/var/lib/freeswitch/recordings/tenant1.server.com/archive/2025/May/08/`
- `av_token` should be a secret string, but it is not used, as each tenant will have their own secret in parameter `av_secret`.

HTTPS Proxy configuration for AudioVault

You should make sure that ALL connections to AudioVault happen over an encrypted channel.

Using nginx as an HTTPS proxy

In FusionPBX, find the file `/etc/nginx/sites-available/fusionpbx` and edit it to add a proxy to local port 4040 when any domain is called with the /audiovault url.

Find the part that says:

```
server {
        listen [::]:443 ssl;
        listen 443 ssl;"
```

and before the first *location* row, add this stanza:

```
location /audiovault/ {
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Host $host;
    proxy_set_header X-Forwarded-Port $server_port;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_set_header X-Forwarded-Server $host;
    proxy_set_header X-Forwarded-Ssl on;
    proxy_set_header X-Forwarded-Webapp /audiovault;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Url-Scheme $scheme;
    rewrite ^/audiovault/(.*)$ /$1 break;
    proxy_pass http://127.0.0.1:4040;
    proxy_buffering off;
    proxy_request_buffering off;
    proxy_read_timeout 240s;
}
```

Then restart nginx. If you connect to any tenant with your browser, e.g. https://tenant1.server.com/audiovault you should see a welcome page.

Make a backup of the `fusionpbx` file before editing it.

Appendix II: FAQs

Service uniloader-freeswitch is generating no data

To test that the system is working, you can either send a call to a `mod_callcenter` queue, or you can change the status of an agent from the FusionPBX GUI. When either of these things happen, Freeswitch will generate events that will trigger the writing on your aggregate `queue_log` file.

If this does not happen, it might mean that:

- Uniloader cannot connect to Freeswitch - you can easily exclude this by checking access - see Checking ESL access
- Freeswitch is generating no events
- Freeswitch is generating events with a format that is not supported

To verify which is which of the last two options, we can connect with and do the same interaction Uniloader does (note: we assume the default address, port and auth token - change as needed):

```
nc -v 127.0.0.1 8021
```

When you have a reply, you type:

```
auth ClueCon
```

And then:

```
events json custom callcenter::info
```

At this points, events are printed. If you log an agent on or off and you receive nothing, then events are not being generated.

To close the connection, press Ctrl + D.

If the command `nc` is not available, the package *netcat* that contains it is available with `apt-get install netcat` on Debian or `yum install netcat` on CentOS.

My tenant is receiving no data

- **Verify queue log generation**: Confirm that a queue_log file is being created at `/opt/uniloader/qlogs/qlog-synth.txt`
- **Validate tenant configuration**: Open the generated queue_log file and verify that your tenant name/ID appears in the logs.
- **Confirm receiving instance status**: Ensure the receiving instance is active and functioning

properly.

- **Test server connectivity**: Run the following command to verify data connectivity from your server:

```
uniloader testupload  --uri "https://qmlive/instance"  --login webqloader --pass
ThEpAsSw0rD
```

Agents cannot log on or off

First, log on to the QueueMetrics Live instance, then click on Home Page → Settings Bar → Diagnostics → AMI Commands.

You should see a list of actions performed recently (or pending), and their own state.

- A state of `wait` means that the action was not yet picked up by Uniloader; a few seconds are OK, but if it stays there for longer, it may mean that service `uniloader-splitter` is not running, or there is a network issue, or maybe you did not set `actions: True` in the Ansible script.
- A state of `ok` means that the action was performed successfully, i.e. the PBX handled it without raising errors.
- A state of `ko` means that the action was not performed: that is, the PBX was unreachable, access credentials were wrong or agent/queue ids were incorrect. Check PBX access using Uniloader.

The time shown in "Sent" is how long the command waited/has been waiting in the queue, while "Proc" is how long it took for the PBX to process it.

How do I take an event trace?

If you need to debug or clarify an unknown issue, Loway support may request an activity trace. This process is straightforward and does not interfere with current operations of service `uniloader-freeswitch`, so it can run in parallel with your current configuration. However, **traces should be captured when there is minimal or no traffic on the system** to ensure the resulting data is easy to analyze, as they are extremely verbose.

Capturing an event trace

- **Access your FusionPBX server** via the command line
- **Start the trace collection**
  - Run the appropriate command from the command line
  - The system will begin printing events as they occur in real-time

An example session could be like:

```
# uniloader-ctl trace

Now running event trace files:
```

```
 - /opt/fusion-splitter-data/traces/evt_251118-1408_trace.txt
 - /opt/fusion-splitter-data/traces/evt_251118-1408_qlog.txt

Press Ctrl+C to terminate.

2025/11/18 14:08:54 Uniloader 25.09.3 (0-20251118.1037) up - GR: 2 - Mem: Alloc 508k
(Free 0k) Sys 6743k
....
^C
```

- **Execute your test call** while the trace is running
- **Stop the trace** by pressing `Ctrl+C` when your test is complete

Submitting your trace files

After stopping the collection, two files will be generated in `/opt/fusion-splitter-data/traces`:

- `evt_(date)_qlog.txt`
- `evt_(date)_trace.txt`

Send both files to Loway support along with detailed information about the test call, including:

- Exact time of the call
- Calling number
- Queue selected
- Agent who answered
- Any other relevant characteristics that will help identify your specific call in the trace

This information is essential for support to quickly locate and analyze your call within the trace data.

You may run multiple traces, and each one will be named differently, in case you need to run multiple separate tests.

For more detail, see qm-fsw-ctl trace.

Can I originate a call via the Freeswitch API and have it tracked?

If you use the Freeswitch API to originate calls, you must "earmark" the call in a specific way so that `uniloader-freeswitch` can pick it up correctly. If you don't, the agent code might be tracked improperly - this happens because when dialing through the API, the first call that appears rings the agent, but what you want to track is the second "call leg" that goes from the agent to the callee.

The secret is adding a new variable to the call, namely `qm_queue=xxx` where *xxx* is a unique code for your outbound campaign. You can have as many codes as you want: this way you can log calls as

appearing to specific sets of calls instead of being simple "outbound".

So for example to originate a call from user *999* in tenant *tenant1.my.srv* to number *0012345678* and have it logged as *camp1*, you could use the following code:

```
api originate
  {qm_queue=camp1,
   origination_caller_id_number=1973,
   origination_caller_id_name=QM-camp1,
   enable_early_media=true}user/999@tenant1.my.srv 0012345678 XML tenant1.my.srv
```

A more complex example where you set the DID and different IDs for the agent leg vs the callee leg, could be:

```
api originate
  {qm_queue=camp1,
   domain_uuid=c42639a1-7536-4d60-8df4-7a6735420c2c,
   domain_name=tenant1.my.srv,
   origination_caller_id_name='Campaign1',
   origination_uuid=c53a59a0-35e9-4d2e-827c-3af089a8e24a,
   outbound_caller_id_name=User_1025,
   outbound_caller_id_number=41771234123,
   origination_caller_id_number=41771234123,
   instant_ringback=true,
   presence_id=1025@tenant1.my.srv,
   call_direction=outbound,
   sip_auto_answer=}user/1025@tenant1.my.srv &transfer('0012345678 XML
tenant1.my.srv')"
```

To view this call, you first have to create a new queue in QueueMetrics. Go to Homepage → Configuration → Queues → "Create new", then set the following fields:

- Queue alias: "Campaign 1"
- Queues: `camp1` (the same value you set in)
- Call flow: Outbound

And then run a report for queue "Campaign 1", or select it on a wallboard, in order to start seeing calls.

See also

- The QueueMetrics User manual: https://docs.loway.ch/QueueMetrics
- The Uniloader user manual: https://docs.loway.ch/Uniloader
- FreeSwitch's mod_callcenter: https://freeswitch.org/confluence/display/FREESWITCH/mod_callcenter
- QueueMetrics Live API manual: available by request to Loway.