



QueueMetrics

call center monitor

USER MANUAL

Version 1.4.1

INDEX

What is QueueMetrics?	7
Installing QueueMetrics	8
Prerequisites: Server	8
Prerequisites: Client	8
Where to install	9
Installing in practice.....	9
Installing using yum	10
Using the JDBC tester page	10
Updating from a previous version of QueueMetrics	12
Automatic update using yum.....	12
The database update utility.....	13
Installing a licence key	14
Setting session timeout	15
Understanding basic security mechanisms	15
Understanding QueueMetrics memory requirements	16
Understanding QueueMetrics disk I/O requirements	17
How much load can QueueMetrics handle?	17
Logging on to QueueMetrics	18
License information	20
Running a report	21
Quick activity reports	21
Agent report	22
Custom reports	22
Understanding results: Common header.....	24
Exporting data from reports.....	24
Understanding results: Answered calls.....	25
Agents on queue	26
Service level agreement	27
Disconnection causes.....	27
Transfers	27

Answered calls by queue	28
Answered calls by direction	28
Answered calls, by stints	28
Detail of answered calls	28
Understanding results: Unanswered calls	29
Disconnection causes.....	30
Unanswered calls, by queue	30
Unanswered calls - distribution by length.....	30
Inclusive Service Level Agreement	30
Unanswered calls by key press.....	31
Unanswered calls, by stints.....	31
All calls, by stints	31
Details of unanswered calls	31
Understanding results: Area code report	32
Understanding results: Inbound ACD call attempts	33
ACD attempts by terminal	35
ACD attempts by queue	35
Understanding results: Call distribution	35
Call distribution per day.....	36
Call distribution per hour	37
Call distribution per day of week	38
Understanding results: Agent activity	39
Agent availability.....	41
Session and pause duration	41
Answered calls for selected queues	42
Answered calls by service groups	42
Session details	42
Pause activity details	43
Agent history popup.....	44
Understanding results: Call outcomes.....	44
Showing call details	47
Detail of answered calls	47

Listening to answered calls	48
Detail of unanswered calls	50
The real-time status panel.....	52
Top status panel.....	53
Calls being processed	54
Agents currently logged in.....	54
Unattended call and VNC monitoring	56
The real-time live page.....	58
The top panel.....	59
Calls being processed	59
Agents currently logged in.....	59
Server status	60
Enabling the real-time live page	60
Help! My Real-time and Live pages display different results!.....	60
The real-time agent page	61
Using the agent's page to control advanced features.....	64
Multi-stint calls	66
Limitations and side-effects	66
Multi-stint calls in QueueMetrics	67
The visitor's page.....	70
Setting up VISITORS in a real life scenario	72
Using Supervisors	73
Automating statistics download: the ROBOT profile	74
<i>Setting up a self-service wallboard</i>	<i>75</i>
Storing queue data on MySQL	77
Who should use MySQL storage?.....	77
Understanding MySQL storage	78
Uploading data to MySQL.....	79
Loading data in QueueMetrics.....	81
Checking MySQL database status.....	81
Optimizing the queue_log table	82
Monitoring clusters with QueueMetrics.....	84

Setting up a cluster.....	84
Setting up the members of the cluster.....	86
Setting up QueueMetrics to access the cluster.....	86
Using the Agent's page with a clustered environment.....	87
Editing QueueMetrics settings	89
Configuring users	90
Editing user classes	93
Configuring queues.....	94
Setting attention levels (Red and yellow alarms).....	96
Configuring agents	97
Configuring locations.....	98
Configuring call outcomes	99
Configuring pause codes.....	101
Automatic QueueMetrics configuration.....	104
Configuring system preferences.....	108
Configuring Asterisk for QueueMetrics.....	113
Configuring queues to report exit status	114
Configuring URLs to be launched by the agent real-time page.....	115
Listening to calls using QM.....	115
Using AddQueueMember for dynamic agents	116
Defining outbound queues	117
Enabling ACD call attempts recording on Asterisk 1.0 and 1.2	118
Enabling ACD call attempts recording on Asterisk 1.4	119
Enabling Unattended Call Monitoring	119
Enabling VNC Monitoring.....	119
Enabling Agent's page actions.....	120
Enabling XML-RPC call listening and streaming	120
Enabling call outcomes	122
Enabling pause codes	123
For more information... ..	124
Appendix I: Default users	125
Appendix II: Security keys.....	126

Appendix III: The [queuemetrics] context.....	127
Appendix IV: Glossary	129

What is QueueMetrics?

QueueMetrics is a versatile call center monitoring system dedicated to call centres based on the Asterisk PBX.

QueueMetrics lets you...

- Run reports on call center activity, divided by queue and filtered by agent and time period, that show what happened (e.g. taken calls, lost calls, agents logging on and off...) during the specified period. Such reports can be run while Asterisk is running, so that you have no delay in seeing what's going on.
- See the details of call center activity, like each single call that was handled or lost, and listen to it through your web browser.
- Have a single real-time panel showing call center activity; you'll see calls being processed by queues and agent activity in the very moment it's happening. You will be able to listen to your agents' calls as they are being made, and optionally see their screen through a VNC application.
- Give your agents a web-based interface panel that lets them see their own calls while they're being handled and optionally launch an external web-app (like a third party CRM module) as the calls come in; they also can use it to log-on to Asterisk, log off and pause/unpause themselves.
- Allow external users, like your clients if you are an outsourcer or the QA dept if you run an in-house call center, monitor your call center in real-time and see a stripped-down version of the current statistics.
- Allows tracking of call completion statuses and pause codes, so you can run statistics on the result of your CC activity and on the time used by your agents, keeping track of their ACD and non-ACD time.

To meet these goals, QueueMetrics processes a file called `queue_log`, i.e. the log file where Asterisk writes signalling events on call queues. QueueMetrics is preconfigured with the standard Asterisk installation paths so it will work out-of-the-box for most installations.

QueueMetrics is meant to be highly customizable; you can alter much of its behaviour to fine-tune it to your own needs (and display your company's – or your client's - logo....).

QueueMetrics is an intranet application as is designed to be used through a web browser. There is no software to install on the client machines. You can access it from anywhere, as long as you have the correct credentials.

QueueMetrics is meant to be free for smaller installations, that is up to two agents, covering most SOHO's and passionate Asterisk hackers. Larger installation can buy a licence based on the call centre size; our clients testify that the extra insight and control on your operation that QueueMetrics makes possible is well worth its price tag!

Installing QueueMetrics

QueueMetrics is written in Java, so it should run on any environment where a Java virtual machine is available. This means that the same version of QueueMetrics runs fine on both Linux and Windows, with no need for a specific version.

Prerequisites: Server

The following software is needed to run QueueMetrics:

- Java SDK, version 1.4 or later
- A modern JSP and servlet container, like Apache Tomcat 5
- MySQL version 4
- Asterisk PBX, version 0.7 or later (versions 1.2 and 1.4 are fully supported)

All said software should be already installed and working on your machine before attempting to install QM.

QM was tested on various distributions of Linux, on Windows 2000/XP and many flavours of Unix.

Prerequisites: Client

QueueMetrics is a web based application, so it does not require any software to be installed on the client machine but a fairly modern web browser.

QM is also a multi-user application, meaning that many users can use share the system at the same time; each user is identified by its credentials and not by its physical location.

The following web browsers have been successfully tested with QM:

- MS Internet Explorer 6
- Mozilla Firefox
- Opera 7, 8 and 9

A number of users have encountered minor annoyances using versions of Firefox before 1.0, mostly on Unix environments.

All versions of Mozilla seem to share a common problem when trying to access multiple user sessions from the same browser instance. You should not therefore use Mozilla to access multiple times to QM; results might be unpredictable.

Where to install

The most common case is to install QM on the same server running your Asterisk instance. This will be fine in most cases, but in very heavily loaded servers running huge analyses it might be possible that QM will end up competing for RAM, CPU and disk I/O with the Asterisk system. In this case, QM should be installed on a separate server and log files should be replicated (or MySQL storage used, see page 77) to minimize impact on the Asterisk server.

In most cases – like mid-sized call centres up to 20 agents on line – it will usually be okay to have everything on the same production server.

It will be fine to have MySQL run on a separate server from the main QM installation.

Installing in practice

Installing QM is easy and only takes a few minutes.

1. Make sure your servlet container is working
2. Make sure your MySQL is working and you have the “create” grants for a new database.
3. Download the latest version of QueueMetrics from <http://queuemetrics.com>
4. Unpack QM in the *webapp/* folder of your servlet container. The folder created will usually be named something like *queuemetrics-1.4.0* – rename it as needed.
5. Download the MySQL connector and place it in WEB-INF/lib with the other Jar archives. It is important that you use the file named *mysql-connector-java-3.0.10-stable-bin.jar*, that can be downloaded from <http://www.mysql.com/products/connector-j/index.html>
Other versions of the MySQL connector will likely work but might require some minor tweaking of parameters¹.
6. Create a database called *queuemetrics* in your MySQL installation and fill it with data taken from the file WEB-INF/README/queuemetrics.sql.

The process will probably be something like:

- Enter your MySQL shell as root typing:

```
mysql mysql
```

- Create the new database

¹ The most common case is when a version of the Connector/J greater or equal than 3.1 is in use. To solve this problem, see the page <http://queuemetrics.com/faq.jsp> . The current versions of QueueMetrics will handle such parameter tweaking automatically.

```
CREATE DATABASE queuemetrics;  
GRANT ALL PRIVILEGES ON queuemetrics.* TO 'queuemetrics'@'localhost'  
IDENTIFIED BY 'javadude';
```

```
- Exit the MySQL shell  
- Load the database sample with something like  
mysql --user=queuemetrics --password=javadude queuemetrics <  
queuemetrics_sample.sql
```

7. Edit WEB-INF/web.xml, change the parameters of JDBC_URL to reflect your installation. The included version uses a database called *queuemetrics* that is on the same server, using a user called “queuemetrics” which password is “javadude”.
8. Restart your servlet container
9. Point your browser to <http://127.0.0.1:8080/queuemetrics>
10. Log in and change the default QM installation passwords.

If you encounter any problems with this setup, you should point your browser to <http://127.0.0.1:8080/queuemetrics/dbtest> for a JDBC tester page.

Installing using yum

On Linux distributions that are derived from Red Hat, it is possible to install QueueMetrics using an automated procedure using the yum utility.

Just type the following commands:

```
wget -P /etc/yum.repos.d http://yum.loway.ch/loway.repo  
yum install queuemetrics
```

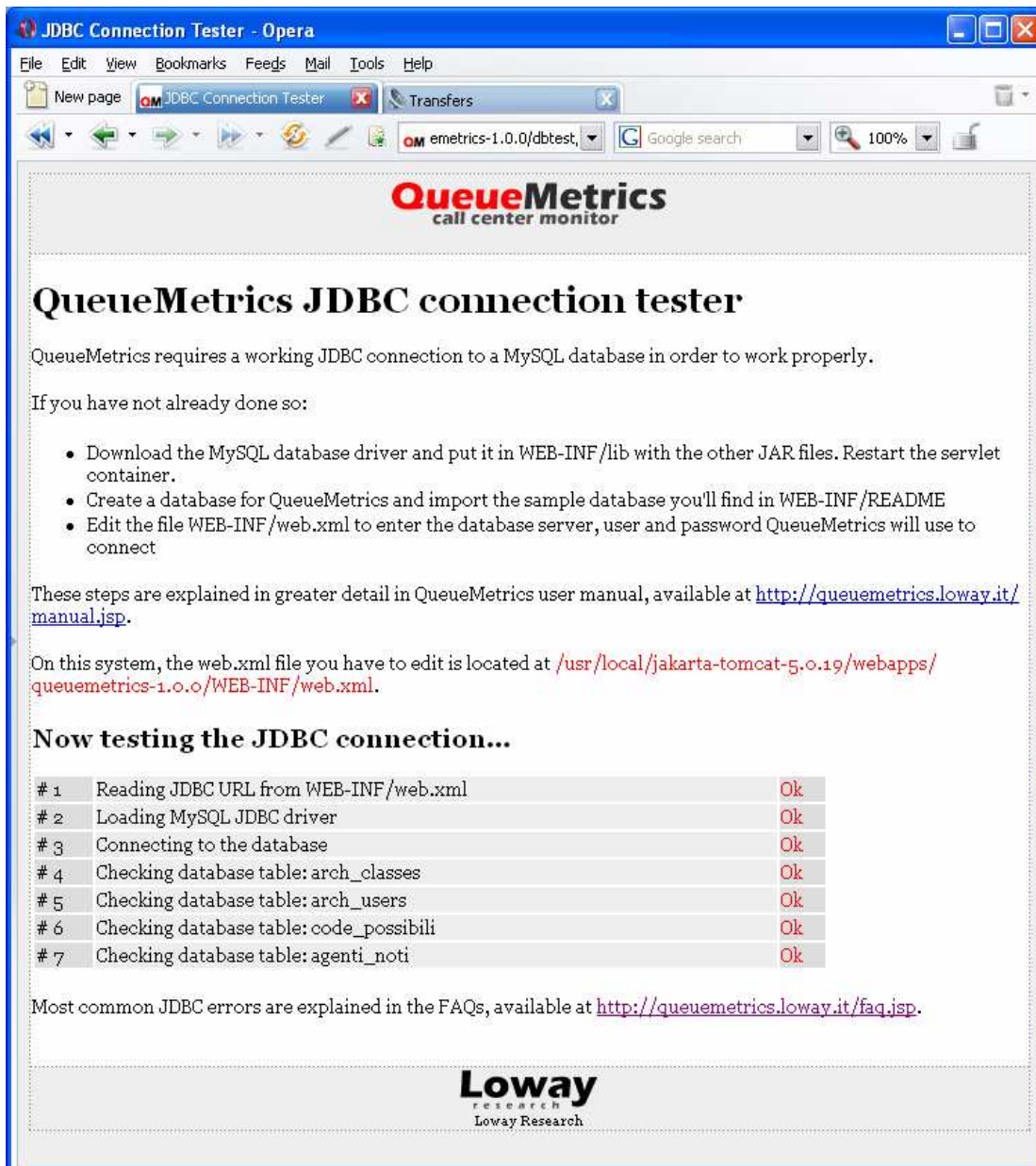
The installation will start automatically and all dependencies will be handled automatically. When it finishes, there is a screen telling you to type a command to create the database; follow the on screen instructions to create it.

When finished, point your browser to <http://127.0.0.1:8080/queuemetrics> and log in using the default credentials.

Using the JDBC tester page

The main source of problems when installing QueueMetrics is to correctly set-up the JDBC connection to the MySQL database. In order to ease the installation process, there is a test page available at the URL <http://127.0.0.1:8080/queuemetrics/dbtest>

The test page will look like the following figure:



If all tests show the OK status, then you are ready to start QueueMetrics. If any test should fail, the web app will tell you the reason of the failure and possible workarounds.

If all tests are Okay, it's a good idea to click on the link that checks that you have the latest version of the database and updates it in case it's necessary.

In this case, for example, one of the tests fails:

Now testing the JDBC connection...

# 1	Reading JDBC URL from WEB-INF/web.xml	Ok
# 2	Loading MySQL JDBC driver	Error

Problem encountered:
The MySQL JDBC driver is not present. Please download it, put it in WEB-INF/lib and restart the servlet container

Java Error:
`java.lang.ClassNotFoundException: com.mysql.jdbc.Driver`

Most common JDBC errors are explained in the FAQs, available at <http://queuemetrics.loway.it/faq.jsp>.

It is very important that you restart the servlet container after tweaking with the JDBC configuration; otherwise your changes may work in the DBTest page but might not be seen by QueueMetrics.

Updating from a previous version of QueueMetrics

If you choose to update from a previous working version of QueueMetrics:

- Make a backup of the files *web.xml* and *configuration.properties* that are found in WEB-INF/. To be extra-safe, make a backup of the whole working webapp and of the database being used.
- Unpack the new version of QueueMetrics
- Copy the old files *web.xml* and *configuration.properties* so your licence and preferences are preserved
- Copy the additional Jar files not distributed with QueueMetrics, e.g. the MySQL connector
- Restart the servlet container
- Run the DB tester
- From the DB tester, run the database update utility
- Once the database update utility reports a success, you're ready to log-in to QueueMetrics

Automatic update using yum

If you originally installed QueueMetrics using yum, you can upgrade your system using yum as well.

- Make a backup copy of the files *web.xml* and *configuration.properties* that are found in WEB-INF/. To be extra-safe, make a backup of the whole working webapp and of the database being used.
- Type the following command:
`yum update queuemetrics`
and follow the update process. Yum will check if a newer version is available and will install it.
- Copy the old *web.xml* and *configuration.properties* over the default ones that were installed using yum.
- Restart QueueMetrics by entering:
`/etc/init.d/queuemetrics restart`
- Point your browser to <http://127.0.0.1:8080/queuemetrics/dbtest> and check if the database is consistent. If there are changes that need to be made to the old database schema, the database update utility (see below) will handle them automatically.


The database update utility

QueueMetrics ships with an utility that makes it very easy to upgrade an existing database to the latest version used by newer versions QueueMetrics. *Before running the update utility, make sure you have a backup of the QM database!*

The database updater works on the same connection that is being tested by the DB connection tester, so it's always a good idea to access the DB connection tester first.

There is a link to the database updater right from the connection tester, or you can access it directly by pointing your browser to

<http://127.0.0.1:8080/queuemetrics/dbtest/dbupdate.jsp>



QueueMetrics database update utility

This utility will check if the database is up-to-date for QueueMetrics version **1.3.0b3** and will update it if necessary.
Before attempting to check/update your QueueMetrics database, **make sure you have a backup of your current database** in case anything goes wrong during the update process!

Your current JDBC URI is:
jdbc:mysql://


- [Yes, I have made a backup and I am ready to check/update the database](#)

Updating database

The database is currently up-to-date.
You are now ready to run QueueMetrics. [Click here to start.](#)

Optimizing database

Description	Result
Optimizing table agenti_noti	OK
Optimizing table arch_classes	OK
Optimizing table arch_users	OK
Optimizing table code_possibili	OK
Optimizing table dbversion	OK
Optimizing table locations	OK
Optimizing table queue_log	OK



Once you access the DB updater, it will update the database and then optimize it for maximum access performance. This may take a while if you have a lot of queue_log data loaded into it.

Installing a licence key

QueueMetrics ships with a limited evaluation key that lets you use the system freely with up to two agents. If you need to evaluate with a larger call center, you will be sent a temporary key that will process as many agents as needed. The same happens when you decide to buy the product.

The key is a single long hexadecimal sequence with minuses in the middle and looks like the following string:

012345678-0987564D-3C082EF8-012345678-0987564D-3C082EF8

The length of the key may vary according to the features needed.

Once Loway Research sends you the temporary or official key, follow these steps to install it:

- Locate the file WEB-INF/web.xml within the QM webapp

- Edit the file with a text editor
- Locate the section with the licence, looking like

```
<init-param>
<param-name>LICENZA_ARCHITETTURA</param-name>
<param-value>.....</param-value>
</init-param>
```

Insert your licence key within the param-value tag, all on one line, exactly as it was sent to you

- Save the modified file
- Restart your servlet container
- Login to QM as usual using your browser
- Click on the "Licence" label to see your current licence.

Setting session timeout

The default session timeout value for QueueMetrics is 30 minutes. This means that if the application is left idle for more than 30 minutes by a user, the resources associated with the user session are reclaimed and the user session expires. If the user tries to continue, he will have to log on again.

It is possible to change the inactivity period that will result in a session timeout by changing the *session-timeout* parameter in *web.xml*, expressed in number of minutes:

```
<session-config>
    <session-timeout>30</session-timeout>
</session-config>
```

If changing this parameter, it is important to keep in mind that real-world users will only seldom use the "Log off" button and will usually rely on closing the browser window when they terminate using QueueMetrics. As the amount of data stored in memory by QueueMetrics can be quite large (runs of tens or hundreds of thousand calls are quite common) they will be using up RAM until the session times out.

Understanding basic security mechanisms

Each user accessing QM should have his own **user** and **password**. The administrator can easily setup multiple accounts from the administrative interface. All user activity is tagged to the user performing it, so it's a good idea to give an account to each person accessing the system. Accounts can be created, blocked and revoked in a matter of minutes.

Each feature that QM offers is enabled by a special **key**, as if there was a padlock protecting it from unauthorized access. The administrator gives each user a key ring that specifies which locks the user can open, and therefore what the user can do. A list of keys used in QM is available in Appendix II on page 126.

To ease the burden of administering multiple users, keys can be grouped into **classes**. Each class offers the additional advantage of giving the key ring a label, so that it's easier to see whether an user is an Administrator, a User or an Agent by looking at the label and not at the very keys s/he holds.

Individual keys can be granted or revoked individually to handle special cases, in addition to the ones anyway present in the user's class. For more information, see page 90.

Understanding QueueMetrics memory requirements

To understand QueueMetrics' memory needs, you must consider that the memory requirements are roughly proportional to the width of the analysis and to the number of required events to track. You may think of it as the number of calls plus the number of agent events, i.e. agents logging on and off and setting pauses on and off.

Calls can be restricted by the queue filter, but all agent events in the required time window are tracked. This gives you an idea of the memory usage.

Though the actual memory requirements depend considerably on the actual content of your analysis and the exact brand and version of Java virtual machine that you are running, you should expect to be possible to track circa 80,000 calls and 40,000 agent events with a standard 64 megabyte Sun Java VM and Tomcat running.

You can of course start your servlet container with more memory in order to allow more room for larger analyses. The standard way in Tomcat is to pass additional Java parameters is to store them in the environment variable `JAVA_OPTS` before starting Tomcat.

Typing:

```
JAVA_OPTS="-Xms256M -Xmx512M"
export JAVA_OPTS
```

And then starting Tomcat will start up a Java virtual machine that has 256 megabytes of available memory and can use up to 512 megabytes. Consider that this memory is shared between all QueueMetrics users and all Java webapps, so the more the better.

Consider also that Java will never return this memory to the system free memory pool, even when it stops using it. The only way to have this memory returned to the system memory pool is to stop the Java VM and restart it.

As a last note, the memory footprint of a Java VM may be quite larger than the memory you give it as Java heap space, as it will need RAM space for the VM itself and all its required libraries. Overheads of 50-100 megabytes are not unheard of.

Understanding QueueMetrics disk I/O requirements

Disk I/O required by QueueMetrics is directly proportional to the queue_log size as it is read from scratch every time you ask for a full analysis. Even if you only care about what happened yesterday between 3 and 4 PM, your 50-megabyte queue_log will be read entirely. As the queue_log usually don't get too large even in the largest installations, this is usually a feasible strategy.

The big advantage of using MySQL as a storage medium is that the queue_log rows are indexed when importing, so only relevant rows are extracted and transferred to QueueMetrics. This should speed things up a bit for the largest installations. Also with MySQL you can put the database on an entirely different server in order to avoid disk I/O problems with the local system running Asterisk – see *Storing queue data on MySQL* on page 77 for complete details.

How much load can QueueMetrics handle?

In order to test if our product behaves correctly under load, we routinely do a stress test of QM simulating 20 users who keep on running reports and real-time monitoring.

We consider the test passed and the product worth releasing if QueueMetrics can handle over one million continuous transactions with no memory problems – they are usually far more than any user will likely do, and with a very constrained VM size.

The stress test that QM 1.4 passed had the following parameters:

- Sun Java 1.4.2_04 running in server mode with 256Mb fixed heap
- SQL storage using connector version 3.10
- 20 concurrent users
- Simulated CC with nearly 1,500 calls per day
- No errors on over 2,000,000 transactions run

QM will easily scale upwards giving it more Java heap space to accommodate larger datasets. Call centres with over 200 agents online and 30,000 calls per day are not an uncommon target for QueueMetrics.

Logging on to QueueMetrics

To log on to QueueMetrics, you have to point your browser to the address of the server where you installed QM. As servlet containers are often installed on ports different than the standard HTTP one, it might be necessary to specify the port address.

For example, if you install Tomcat 5 on the same server you're accessing QM from, you may end up pointing your browser to: <http://localhost:8080/queuemetrics> .

Ask your system administrator for the correct web address of your instance of QueueMetrics.

If all goes well, you will see a page like the following one:

The screenshot shows the QueueMetrics login interface. At the top left is a placeholder for 'Your Logo'. At the top right is the 'QueueMetrics call center monitor' logo and a 'Print' button. The central section is titled 'User Logon' and contains three input fields: 'Login:', 'Password:', and 'Language:' (which is a dropdown menu currently showing 'US English'). Below these fields is a 'Log In »' button. A message below the login fields reads: 'Please ask your system administrator for the correct credentials to access this instance of QueueMetrics.' The footer of the page features the 'Loway research' logo.

This and the following screenshots are taken using Opera 8 on Windows; other environments may present minor discrepancies from what is shown here.

If your system administrator has already configured QM, you might see your firm's logo on the top left part of the screen and a different welcome message.

To enter the system as a user, enter the standard credentials **demouser** with password **demo** and click on the "Log in" button, or use the credentials your administrator has provided.

If you prefer to use a different language from the default English, you can choose one of the other supported languages from the drop-down box. After choosing the language, the main page will be reloaded.



The user is presented with the Home Page, that is the starting point of QM. The name of the user and the current class for the user are shown on the top-right corner of the window.

To end the current session, you have to press the “Log off” icon or close the browser window.

To print the current page in a printer-friendly format, you just press the “Print” icon.

To see more details on the current user and change its access password, click on the “Info” icon.

License information

Pressing the “Licence information” label, a page like the one below is shown.


Your Logo

Demo Admin | Administrator






Home

Licence information

Software release:	Loway QueueMetrics - 1.4.0-beta2
Licensed to:	SvilInternoQM13
Maximum licenced agents:	9999
Licence expires on:	2007-08-25
Operating System:	O.S.: Windows XP - Ver: 5.1 - x86
Java Runtime:	Version: 1.4.2 Vendor: Sun Microsystems Inc. Class Version: 48.0 Java Home: C:\Programmi\j2sdk_nb\j2sdk1.4.2\jre
Loway TPF:	Version: \$Revision: 1.38 \$ #Build: 1169#D
Language pack:	\$Id: queuemetrics_en_US.properties,v 1.11 2007/06/11 20:52:29 lenz Exp \$

The official QueueMetrics website is located at <http://queuemetrics.com>.

Contributors

QueueMetrics would not be possible without the joint efforts of the following people:

- Italian localization: Lorenzo Emilietri, Loway Research
- German localization: Thomas Dreiling, TelzWeb GmbH
- French localization: Vincent Nonnemacher, AgentSpace
- Latin America Spanish localization: Carlos Julio Moya, Colombia

The following high quality Free Software libraries and components are used by this work:

- [Redstone XML-RPC library](#)
Distributed under the LGPL, as of paragraph 6.0 of said licence.
- [The Silk icon set](#) by Mark James
Licensed under a Creative Commons Attribution 2.5 License.

We offer all interested parties the opportunity to download said components sources at no cost from our web site at <http://www.loway.it/third-party>.

This page shows the current release of the software and the current license information.

If you are running a free demo version, you will see that the maximum number of licensed agents is 2 and an additional text will remind you on how to register.

You can also see some information being shown on the Operating System and Java version being used. Such information is very useful to in the case of errors and should be sent to Loway in the case you think you have found a bug.

Running a report

To successfully run a report, your system administrator must have configured the correct queues in use on your system. You will find them in the drop-down menu on top of the page. See page 94 for details on how to do it.


Quick activity reports

The quickest way to obtain an analysis is by selecting the queue you want to analyze and then click on the appropriate time frame below the “Quick activity reports” title on the home page.


The defined time frames are the following:

- *Today, Yesterday, The day before yesterday*
The day in question, starting from midnight to midnight
- *Last day, Last 7 days, Last 30 days, Last 90 days*
The exact time period, starting from the current hour backwards.

The system will then show the “Answered calls” page, like here below.


Your Logo

Demo Admin | Administrator

QueueMetrics
call center monitor

Home
Answered
Unans.
Area
Att.
Distrib.
Agents
Outcomes
All

Report Details:

Atomic queue(s) considered:	00 All
Period start date:	June 22 2007, 0:00
Period end date:	June 22 2007, 23:59
Total calls processed:	1,702
	75.0% ans / 25.0% unans

Answered calls

All calls:

N. calls answered by operators:	1,277
Average call length:	60.0 s.
Min call length:	0:20
Max call length:	2:30
Total call length:	21.3 H
Average call waiting time:	13.3 s.
Min waiting time:	0:10
Max waiting time:	0:20
Total waiting time:	4.7 H

Calls fully within the given time interval:

N. calls answered by operators:	1,276
Average call length:	60.0 s.
Min call length:	0:20
Max call length:	2:30
Total call length:	21.3 H
Average call waiting time:	13.3 s.
Min waiting time:	0:10
Max waiting time:	0:20
Total waiting time:	4.7 H

Agents on queue

Agent	N. Calls		...
John Doe (101)	213	16.7%	<div></div>
Mike Boo (102)	425	33.3%	<div></div>
agent/103	213	16.7%	<div></div>
agent/104	426	33.4%	<div></div>

Export as...   

On top of the page, you can see a multi-tab menu; by clicking on it you can select which part of the report you are going to see. To go back to the home page, click on the “Home” tab. You can also see all the analyses at once by clicking on the “All” label (this is mostly useful when printing the results to paper).

Agent report

If the user has the appropriate grants, s/he can restrict the analysis to a single agent. This way one can see exactly what one agent did.

To use this feature, select the agent you want to filter by and click on the desired time period in the “Agent report” section of the Home Page.

Custom reports

Custom reports are available by clicking on “Run custom report” from the Home Page.

A new menu will appear, asking for custom report parameters:

The screenshot shows a web interface with a 'Home' tab selected. The main content area is titled 'Custom report analysis'. It contains the following fields and controls:

- Queue:** A dropdown menu with 'All' selected.
- Agent:** A dropdown menu with '-' selected.
- Location:** A dropdown menu with '-' selected.
- Start Date:** A date picker showing '22 June 2007'.
- End Date:** A date picker showing '22 June 2007'.
- File:** A text input field containing 'sql:rt'.
- Time zone offset:** A dropdown menu with 'No offset' selected.
- Join multi-stint calls:** A dropdown menu with 'No' selected.
- Buttons:** Two buttons at the bottom: 'Start realtime monitoring' and 'Run custom report'.

The footer of the interface displays the 'Loway research' logo and the text 'Loway Research'.

The meaning is as follows:

- *Queue* is the queue or composite queue you want to analyze;
- *Agent* is an agent to be used as filter, or “-” as no filter;
- *Location* is a location to be used as a filter, or “-” for none;
- *Start* and *end date* let you select the period you want to analyze, with five-minute resolution;
- *File* is the queue_log file you want to analyze. You may want to change it to run reports on a different Asterisk server or on an older archived version of your queue_log. If you run QM on the same machine as Asterisk, the file name should be already correct. Make sure the file is readable to your servlet container. If you use MySQL storage or clusters, the file will look something like “sql:P001” or “cluster:”
- *Time zone offset* is to be set if the Asterisk server that created the queue_log file was in a different time zone from the one you are using.
- *Join multi-stint calls* lets you join together the pieces of the same call if it has been processed by more than one Asterisk queue (see page 66)

By clicking on the “Run custom report” button, you can run the analysis, which output is the same as the “Quick activity report” and will be explained below.

Understanding results: Common header

On the top of each report, a box will be shown showing:

- Which queue or queues were considered for the analysis
- The time period the analysis refers to
- Whether the report is about the whole of the queue or a specific agent
- The total number of calls processed for this analysis, divided into answered and unanswered ones.
- If running in multi-stint mode, the total number of calls that were joined together

There is also a box showing a number of analyses you can export in CSV format.

When running in report mode, QM distinguishes between calls or agent sessions that are complete and calls or agent sessions that are “ongoing” at the moment the report was taken. This behaviour is different from version of QueueMetrics up to 1.1, where incomplete entities were simply discarded.

Ongoing calls or sessions are usually marked in red and counted separately, as data for them is not definitive and will appear differently if you run further reports.

You should also note that a call that has not been answered yet will be counted as “Ongoing unanswered”, though it may well be answered in the nearest future by one of your agents.




In any case, if you need to see calls in progress or whether an agent is logged in, you should rely on the Real-time panels and not on the reports.

Exporting data from reports

It is possible to export data in Microsoft Excel, Comma-Separated Values (CSV) or XML right from most QueueMetrics panels.

Agents on queue

Agent	N. Calls		...
John Doe (101)	215	16.7%	<div></div>
Mike Boo (102)	428	33.3%	<div></div>
agent/103	215	16.7%	<div></div>
agent/104	429	33.3%	<div></div>

Export as...   

By clicking on the Excel, CSV or XML icons below each report, it is possible to save exactly the same report as seen on screen and then edit it using your favourite number-crunching software.

You must be logged in to download the reports, as you see them on screen.

If you are looking for a way to export a full analysis to one file, you should probably have a look at *Automating statistics download: the ROBOT profile* on page 74.

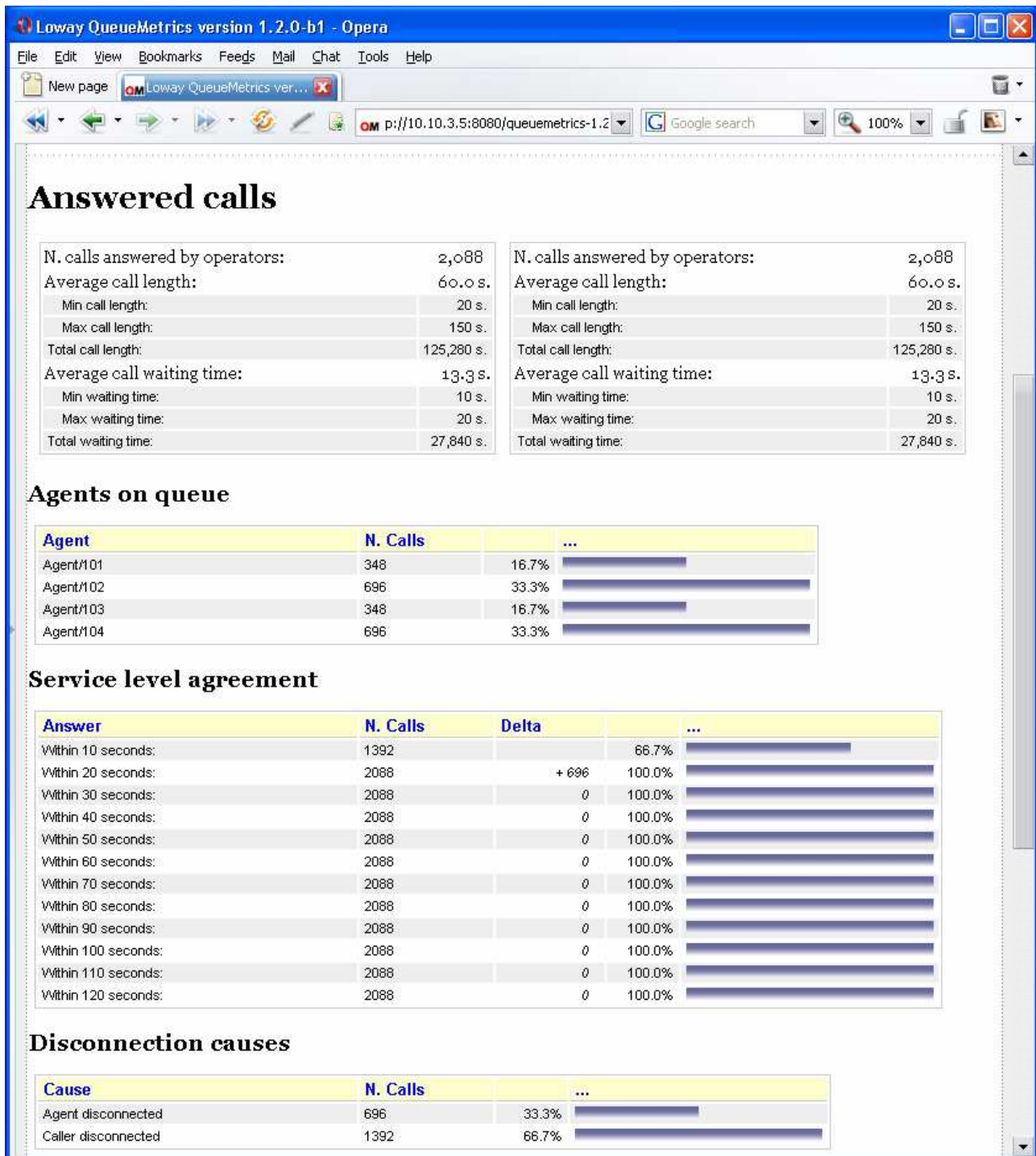
Understanding results: Answered calls

The answered calls section deals with calls that were correctly handled by agents.

The top panel shows:

- How many calls were handled;
- The average call length (i.e. time the caller spends talking to an operator);
- The maximum and minimum call lengths recorded for the given time period;
- The total call length (for all calls on all operators);
- The average call waiting time (i.e. the time a caller was waiting on a queue before being connected to an operator).
- The minimum and maximum call waiting times on record
- The total waiting time for all handled calls.

You can see that the information above is reported twice: on the left for all calls, including incomplete ones, and on the right for complete calls only, i.e. excluding calls that were started before or terminated after the given time frame.



Agents on queue

This report shows which agents have been available for the given queue, how many calls each one handled and the percentage of all calls that each one handled.

If calls are connected directly to a phone terminal, QM tries its best to show the corresponding terminal, usually in the format used by Asterisk, like "SIP/303" to signify a SIP phone whose number is 303.

If you connect to H.323 telephones via the OH323 module, the recorded channel names have no meaning and do not refer to a specific terminal; that's why all OH323 calls are grouped together under the label "OH323/-".

Service level agreement

This report shows the distribution of call waiting times. It shows how many calls were answered within a given time frame, usually 120 seconds in 10 second increments (the time frame and increment can be modified by the administrator, if needed – see page 108).

You get a percentage of how many calls were answered within X seconds; the percentage includes calls answered in a shorter time frame and therefore grows with time.

The "delta" value you see is the absolute increment, expressed in number of calls, between each time frame.

This metric is computed only on answered calls, i.e. ignoring lost calls . If your SLA is defined in terms of taken and lost calls, see the corresponding metrics "Inclusive SLA" on page 30.

Disconnection causes

This report shows the reason why calls were terminated; this means that:

- The agent hung up, or
- The caller hung up, or
- The call was transferred outside the queue and the agent was freed again, or
- The call was ongoing at the time the report was run.

Transfers

This graph shows how many calls were transferred to each extension in the given time frame. This lets you know who is handling exception calls.

Note: when a call is transferred outside the queue system, its length is no more recorded by the queue subsystem; therefore you only get to see the length of the call while the agent was on line.

Answered calls, by queue

Queue	N. Calls		...
q1	643	50.0%	
q2	644	50.0%	

Export as...

Answered calls, by direction

Direction	N. Calls		...
Inbound calls	0	0.0%	
Outbound calls	0	0.0%	
Undefined	1287	100.0%	

Export as...

Answered calls, by stints

Number of stints	N. Calls		...
1	1287	100.0%	

Export as...

Report details answered calls

Total rows processed: 1287

Detail

Answered calls by queue

If more than one queue is in use for the report, this graph shows the relative magnitude of each queue.

Inbound queues are marked with the symbol , while outbound queues use the symbol .

Answered calls by direction

If more than one queue is in use for the report, this graph shows the relative magnitude of inbound versus outbound calls made.

Answered calls, by stints

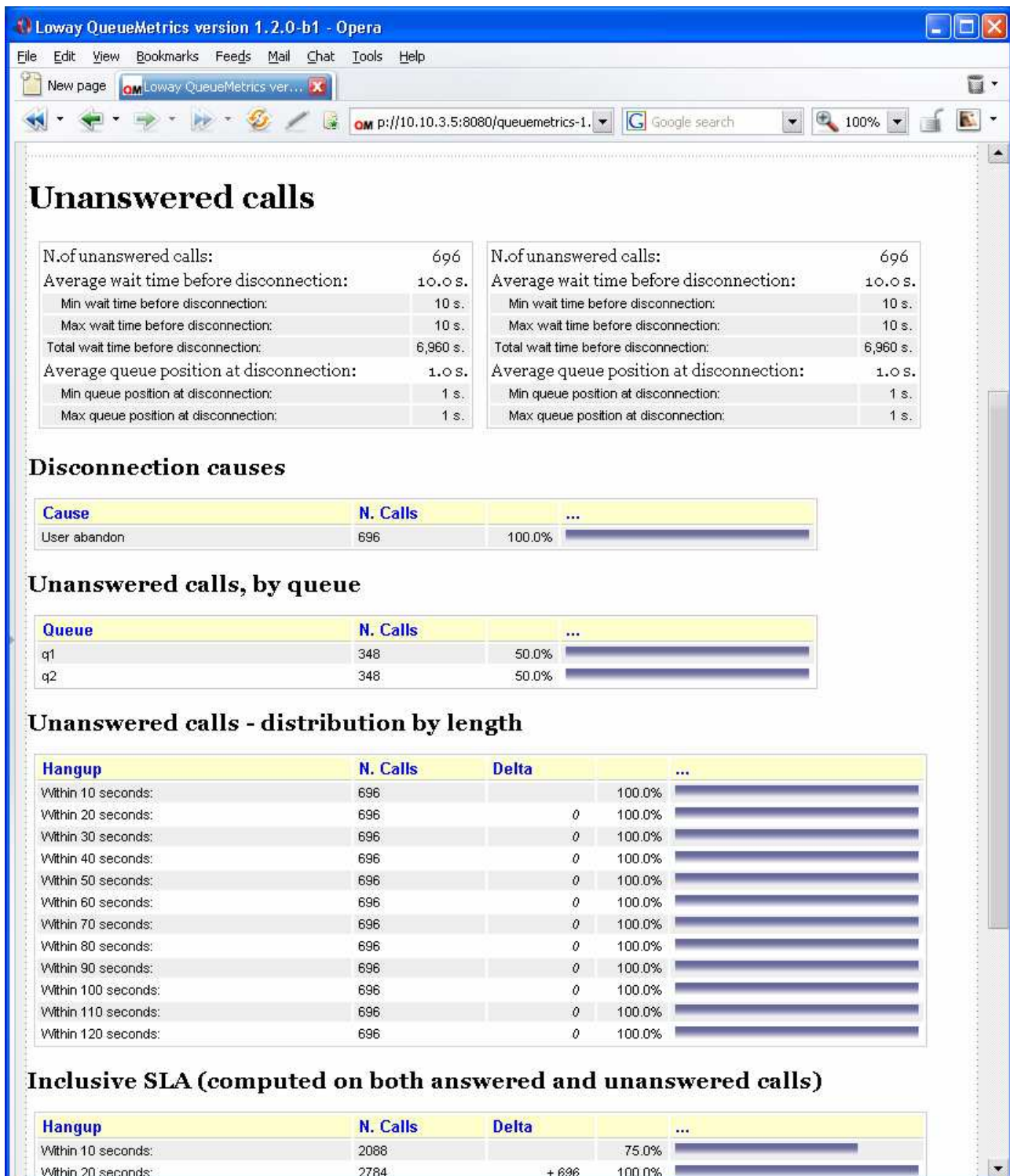
This graph counts the distribution of multi-stint calls on selected queues. If multi-stint mode is not enable, all calls will have only one stint.

Detail of answered calls

This page shows the detail of answered calls (see page 47).

Understanding results: Unanswered calls

Unanswered calls are calls that were lost, i.e. the caller could not connect to an agent. This usually means that either the caller hung up, fed up with waiting, or the queue system decided to discharge the caller, maybe sending him to voicemail or another queue.



The report shows:

- How many calls were lost;
- The average waiting time before disconnection;
- The average queue position at disconnection (i.e. how many calls the queue had to dispatch before connecting the caller to an operator).
- The minimum and maximum wait times
- The minimum and maximum queue position at disconnect.

As with answered calls, this report is computed twice; the version on the left is for all calls monitored, while the version on the right only holds data for calls that were complete at the moment the analysis was run.

Disconnection causes

This report shows the relative magnitude of disconnection causes, that are:

- The caller hung up, or
- The queue timed out and discharged the caller (if this feature is enabled by the queue configuration – see page 114), or
- The caller exited the queue by pressing a key (if this feature is enabled by the queue configuration).

Unanswered calls, by queue

If more than one queue is in use for the report, this graph shows the relative magnitude of each queue.

Unanswered calls - distribution by length

This report is functionally equivalent to “Service level agreement” in the Answered calls section (see page 27), but is computed on lost calls. It shows how many calls were hung up within a given time frame, usually 120 seconds in 10 second increments (the time frame and increment can be modified by the administrator, if needed – see page 108).

You get a percentage of how many calls were lost within X seconds; the percentage includes calls lost in a shorter time frame and therefore grows with time.

The “delta” value you see is the absolute increment, expressed in number of calls, between each time frame.

Inclusive Service Level Agreement

The inclusive SLA corresponds to the Service Level Agreement metrics shown on page 27, with the difference that it is computed taking into consideration both answered and unanswered calls.

Unanswered calls by key press

If there are any calls that are were set unanswered because the caller pressed a key to exit the queue, this graph shows which keys were pressed and how many calls were terminated for that reason.

Unanswered calls, by stints

This graph tells the stint distribution of unanswered calls. It corresponds to the graph called “Answered calls, by stints”.

All calls, by stints

This graph tells the stint distribution of all processed calls. It corresponds to the sum of the graphs called “Answered calls, by stints” and “Unanswered calls, by stints”

Details of unanswered calls

This page shows full details of unanswered calls (see page 50).

Understanding results: Area code report

If the Caller*ID is present, it is possible to break down both answered and unanswered calls to specific area codes by clicking on the “Area code analysis” button.

Area code report

Number of CLID digits to search: 3
Starting from position: 2
Search

Detail for answered calls

Area code/Caller id	Taken calls	Total calling time	Average time per call	Average wait time per call
555	1286	21:26:16	1:00	0:13
HIDDEN	1	0:20	0:20	0:20

Export as...

Detail for unanswered calls

Area code/Caller id	Lost calls	Average wait time per call	Average position
555	430	0:10	1.0

Export as...

By selecting a number of caller id digits to search upon and a starting digit position, you get a number of statistics grouped by area codes.

This report gives an immediate check of the geographical origin of calls handled by your call center.

It is possible to export all the reports as needed.

Understanding results: Inbound ACD call attempts

When running an inbound call center, it is very important to determine the reason why a call is delayed: are your clients refusing to answer? Did they forget to log off before leaving their workplace? The inbound ACD call attempts metrics try to answer to these questions.

As these metrics are not usually recorded by Asterisk, you'll have to patch and recompile your Asterisk system in order to gather them – see the section Enabling ACD call attempts recording on page 118. If you do not do so, the metrics presented here will always appear zeroed out. With Asterisk 1.4, this feature should be automatically enabled with no need to patch the source code.

Report Details:

Atomic queue(s) considered:	00A,ALL q1, q2
Period start date:	2006-03-16, 22:21
Period end date:	2006-06-14, 22:21
Total calls processed:	2784 75.0% ans / 25.0% unans

Export analysis in CSV format:
 Answered calls: Agents on queue

Inbound ACD call attempts

Total ACD attempts sent to operators:	3,480
Taken calls	
Average attempts:	1.3
Min attempts:	1
Max attempts:	2
Total attempts:	2,784
Lost calls	
Average attempts:	1.0
Min attempts:	1
Max attempts:	1
Total attempts:	696

ACD attempts by terminal

Agent	N. lost	Avg ring	Ring (s)	N. Taken	Avg ring	Ring (s)
Agent/101	0	-	0 s.	348	0 s	0 s.
Agent/102	696	10 s	6,960 s.	696	15 s	10,440 s.
Agent/103	696	10 s	6,960 s.	348	10 s	3,480 s.
Agent/104	0	-	0 s.	696	10 s	6,960 s.

ACD attempts by queue

Agent	N. lost	Avg ring	Ring (s)	N. Taken	Avg ring	Ring (s)
q1	696	10 s	6,960 s.	1044	10 s	10,440 s.
q2	696	10 s	6,960 s.	1044	10 s	10,440 s.

This page shows the following pieces of information:

- How many agent attempts were made, i.e. how many times the agent's telephones were rung in total

- The average number of attempts that were necessary for a taken call; the minimum, maximum and total attempts made that resulted in a taken call
- The average number of attempts that were necessary for a lost call; the minimum, maximum and total attempts made that resulted in a lost call

ACD attempts by terminal

This graph breaks down agent attempts by the agent that was called. The following pieces of information are extracted for each agent:

- N. of lost agent attempts (i.e. the agent was called but not responding)
- The average ring time for lost attempts
- The total ringing time for lost calls
- The number of taken agent attempts (i.e. calls answered)
- The average ring duration for taken calls
- The total ring time for taken calls

ACD attempts by queue

The following metrics are extracted and broken down by queue:

- N. of lost agent attempts (i.e. the agent was called but not responding)
- The average ring time for lost attempts
- The total ringing time for lost calls
- The number of taken agent attempts (i.e. calls answered)
- The average ring duration for taken calls
- The total ring time for taken calls

Understanding results: Call distribution

The call distribution report shows when calls were handled, when calls were lost and the average wait times broken down by period.

All percentages are calculated on the call class they belong to, i.e. a 50% of “Unanswered calls” on one day means that 50% of all unanswered calls for the period happened during that day, not that 50% of calls were lost.

For each metrics, the total number of calls is shown, together with average, minimum and maximum times. Graphs are plotted on the total number of calls broken down and on the averages.

It is possible to change the interval in the Hourly graphs, so that you can have reports break down calls e.g. by half-hours or hour quarters, by changing a value in the QueueMetrics master configuration file.

Home
Answered
Unans.
Area
Att.
Distrib.
Agents
Outcomes
All

Report Details:
Atomic queue(s) considered: 00 All
Period start date: June 23 2007, 0:00
Period end date: June 23 2007, 23:59
Total calls processed: 1,002
75.0% ans / 25.0% unans

Answered call distribution per day

Day	Num	Answered calls	Avg	Min	Max	Avg duration
2007-06-22	1	0.1%	0:20	0:20	0:20	
2007-06-23	750	99.9%	1:00	0:20	2:30	

Export as...

Answered call wait time per day

Day	Num	Answered calls	Avg	Min	Max	Avg wait
2007-06-22	1	0.1%	0:20	0:20	0:20	
2007-06-23	750	99.9%	0:13	0:10	0:20	

Export as...

Unanswered call wait time per day

Day	Num	Unanswered calls	Avg	Min	Max	Avg wait
2007-06-23	251	100.0%	0:10	0:10	0:10	

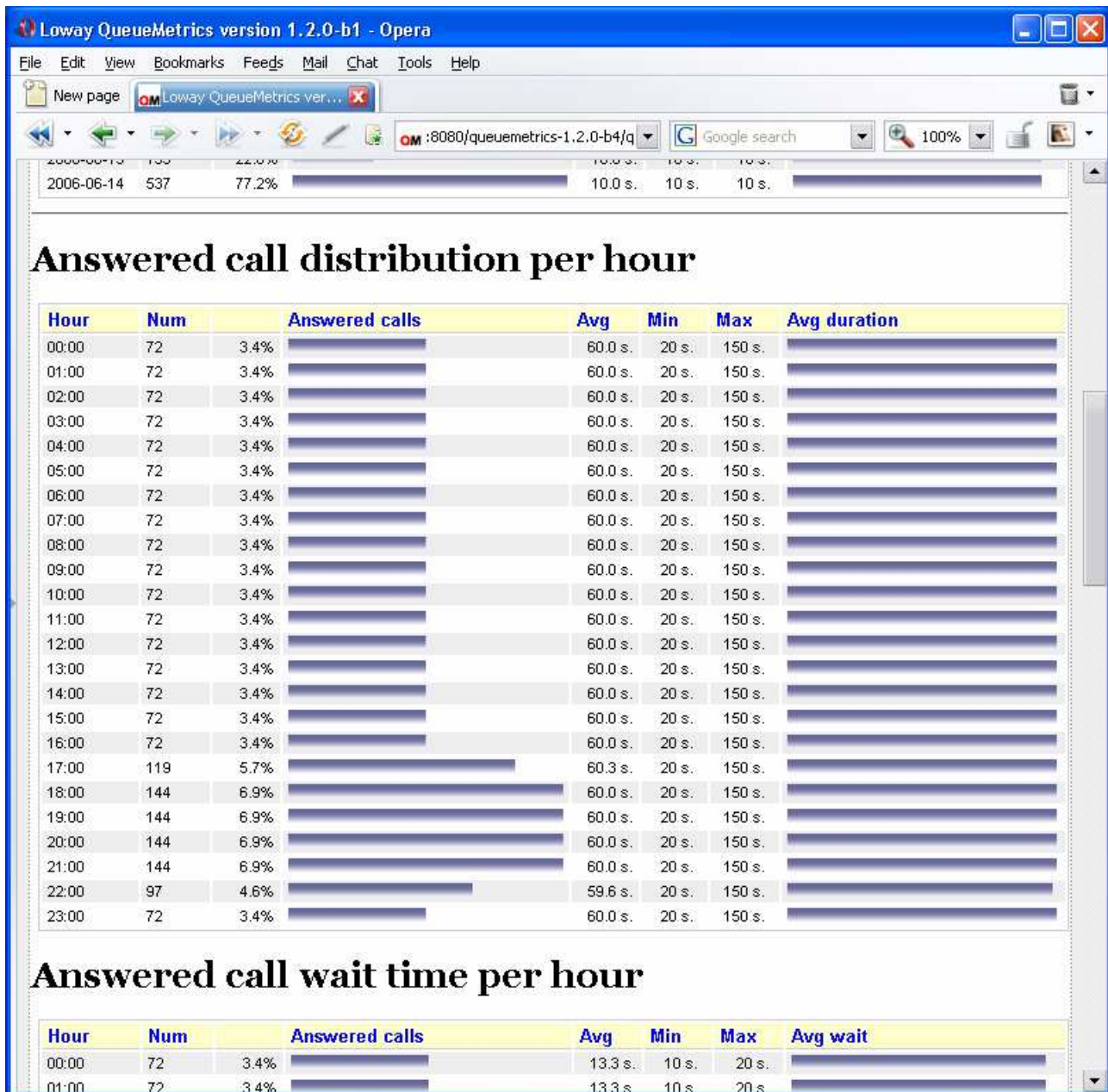
Export as...

Sales per day

Day	Conv.	Sales	Contacts
2007-06-23	40.0%	250	375

Call distribution per day

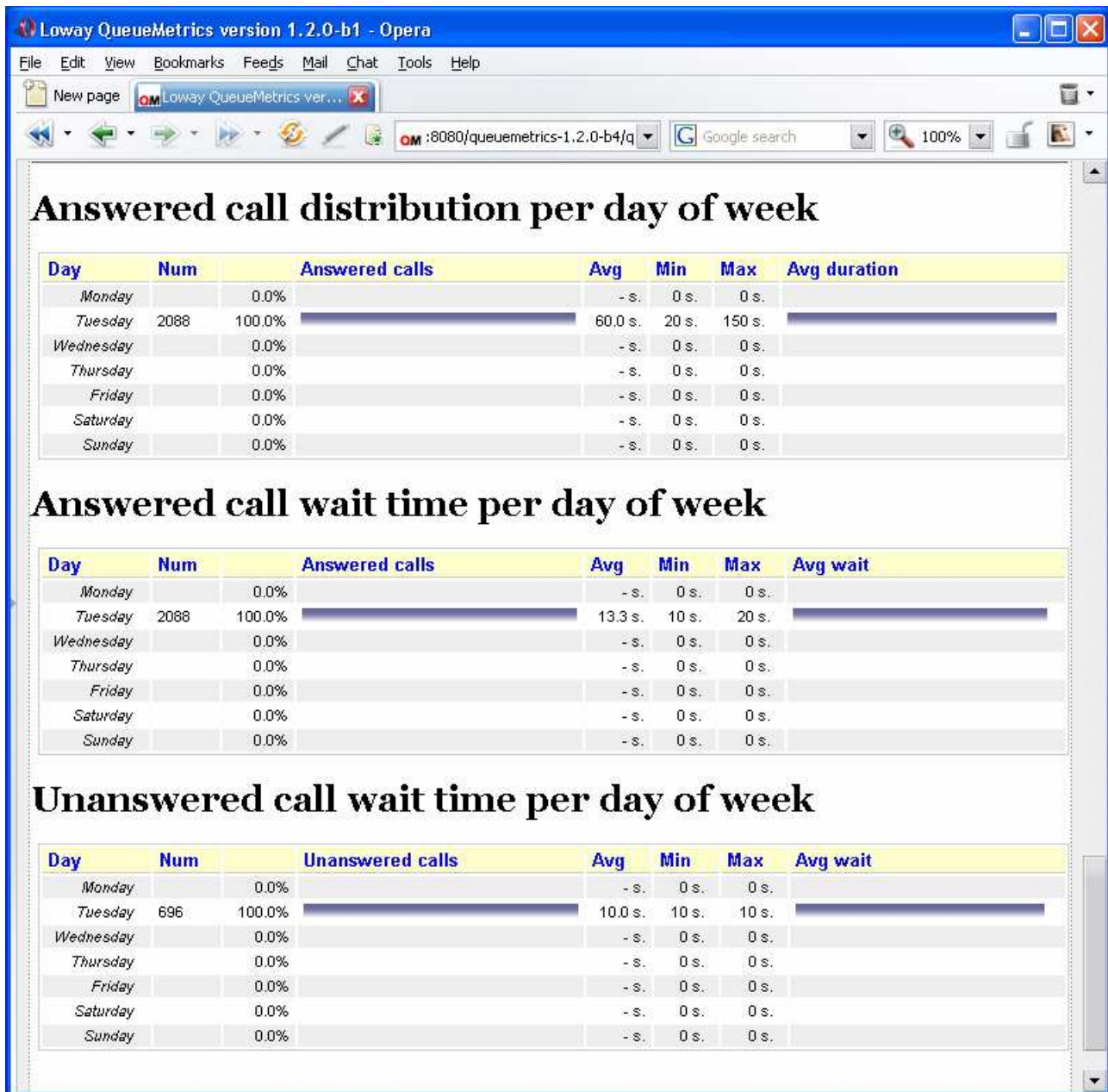
Calls, both taken and lost, are shown per specific day. Days with no events are not shown. The total numbers of call lengths, wait time for answered calls and wait time for unanswered calls are plotted for each day. Sales and contacts are also shown on a daily basis.



Call distribution per hour

Events are shown on a 24-hour distribution. If this graph appears to be incorrect, you have to run a “Custom report” setting the time zone accordingly (see page 22).

The total numbers of call lengths, wait time for answered calls and wait time for unanswered calls, together with sales and contacts, are plotted for each hourly interval. The size of hourly intervals can be controlled by the *default.hourly_slot* configuration property, making it possible to run this reports based on 30-minute, 20-minute or 15-minute intervals.



Call distribution per day of week

This report shows the weekly behaviour of your queues. The longer the analysis period, the more significant its results will be.

The total numbers of call lengths, wait time for answered calls and wait time for unanswered calls are plotted for each day of the week.

Understanding results: Agent activity

Agent activity refers to the behaviour of Asterisk defined agents. If you connect you queues straight to telephone terminals, this section will always be empty.

Each agent may be flagged as being a member of four priority groups:

- *Main*: the agents usually answering the queue
- *Spill*: the agents answering the queue if all “Main” agents are busy or unavailable
- *Wrap*: the agents answering the queue if all “Main” and “Spill” agents are busy or unavailable
- *Undefined*: this agent is not a member of any priority group for this queue

This feature is useful if priority groups are used in the queue configuration. If they are not used, just assign all agents to “Main” for each queue.

If an unknown agent appears on a queue, it will be marked as “Undefined”, written in red.

Agent names are written in blue and are clickable, if you click on them in any of the graphs, you will be lead to a popup that detail the logon and pause history for that agent.

As a default, QueueMetrics will show and count an agent session if and only if the agent handled at least one call during this session. This may not be what you want when you use pause codes – an agent may log on and immediately go on pause to do some back-end activities. If this is the case, you should set the configuration option *default.useRawAgentSessions* to *true* to see all agent sessions.

Home
Answered
Unans.
Area
Att.
Distrib.
Agents
Outcomes
All

Report Details:

Atomic queue(s) considered:	00 All
Period start date:	June 23 2007, 0:00
Period end date:	June 23 2007, 23:59
Total calls processed:	1,002
	75.0% ans / 25.0% unans

Agent session detail

Agent sessions

N. of agents available:	4
Average agent time:	8:40:57
Min agent time:	7:18:00
Max agent time:	10:11:06
Total agent time:	34.7 H

Agent availability (for all the queues they are member of)

Level	Agent	Time	On pause	Billable	Non bill.	...
Undefined	John Doe (101)	8:04:00	2:05:00	0:00	2:05:00	23.2%
Undefined	Mike Boo (102)	10:11:06	0:00	0:00	0:00	29.3%
Undefined	agent/103	9:10:42	1:02:30	1:02:30	0:00	26.4%
Undefined	agent/104	7:18:00	41:40	0:00	41:40	21.0%

Export as...

The report shows:

- How many agents were available for the queue. To be considered available, an agent must have logged in and taken at least one call.
- How much time all agents have been available
- The average agent available time
- The minimum and maximum agent session durations
- The total billable and not billable pause times

Agent availability

This graph shows which agents were available during the specified time frame and the percentage of agents' available time each one cumulated.

This time is calculated per all queues any agent is a member of, as the act of logging on is in general for the whole system and not specific to one single queue.

For each agent, the total time on pause – if any – is computed and broken down as “Billable” or “Not billable” – see the section on Pause Codes.

Session and pause duration

For each agent, the total number of sessions and pauses is computed (session time is already deducted of pause time). For both sessions and pauses, an average length is computed.




The “Pause percentage” is how much time an agent was on pause compared to available time.

The “Pauses per session” computes how many pauses – on average – each agent makes for each log-in session.

These metrics should be considered according to your call center rules on pauses and time-out.




Answered calls (for selected queues)

Level	Agent	N. calls	Total call time	Avg call time	Avg wait time	% of queue
Undefined	John Doe (101)	125	5:12:30	2:30	0:10	16.6%
Undefined	Mike Boo (102)	250	2:25:50	0:35	0:15	33.3%
Undefined	agent/103	125	2:05:00	1:00	0:10	16.6%
Undefined	agent/104	251	2:47:00	0:39	0:15	33.4%

Export as...   




Answered calls by service group

Level	N. calls	Total call time	Avg call time	% of queue
Main	0	0:00	-	0.0%
Wrap	0	0:00	-	0.0%
Spill	0	0:00	-	0.0%
Undefined	751	12:30:20	0:59	100.0%

Export as...   

Answered calls by location

Location	N. calls	Total call time	Avg call time	Avg wait time	% of queue
Main Location	125	5:12:30	2:30	0:10	16.6%
Secondary location	250	2:25:50	0:35	0:15	33.3%
-	376	4:52:00	0:46	0:13	50.1%

Export as...   

Session details

Total sessions: 504	Detail
Total pause activity detail	Detail

Answered calls for selected queues

This graph shows who of your agents answered calls for the queues you selected. The number of calls, together with total and average call durations are computed accordingly.

Answered calls by service groups

This graph show which priority levels handled calls for your queue. This shows whether your main line is staffed enough to handle the load of incoming calls.

Session details

By clicking on the “Detail” button, a new page is shown, detailing each agent session that was recorded.




Detail of agent sessions

Report Details:

Atomic queue(s) considered:	00 All
Period start date:	June 23 2007, 0:00
Period end date:	June 23 2007, 23:59
Total calls processed:	1,002
	75.0% ans / 25.0% unans

Return

Session agent detail

Export as...   

Agent	Start hour	End hour	Duration	Ext.	Termination	Pause	P.Time	Srv
agent/104	06/23 - 0:00:01	06/23 - 0:00:31	0:30			0	0:00	
John Doe (101)	06/22 - 23:56:01	06/23 - 0:00:51	4:50	123		0	0:00	
Mike Boo (102)	06/22 - 23:56:01	06/23 - 0:00:52	4:51	345		0	0:00	
agent/103	06/22 - 23:56:01	06/23 - 0:00:53	4:52	456		0	0:00	
agent/104	06/23 - 0:01:41	06/23 - 0:05:31	3:50	789		1	0:20	
John Doe (101)	06/23 - 0:01:01	06/23 - 0:05:51	4:50	123		1	1:00	
Mike Boo (102)	06/23 - 0:01:01	06/23 - 0:05:52	4:51	345		0	0:00	
agent/103	06/23 - 0:01:01	06/23 - 0:05:53	4:52	456		1	0:30	
agent/104	06/23 - 0:06:41	06/23 - 0:10:31	3:50	789		1	0:20	
John Doe (101)	06/23 - 0:06:01	06/23 - 0:10:51	4:50	123		1	1:00	
Mike Boo (102)	06/23 - 0:06:01	06/23 - 0:10:52	4:51	345		0	0:00	

For each agent session, the start and end times are recorded, together with the total duration in seconds.

If the agent logs on via the call back function, the designated call back extension is shown.

The number of pauses and the total pause time in seconds is shown.

The "Srv" column tells you on which server an agent was working in case you set up a cluster of Asterisk servers.

It is possible to sort the table for each title, in either descending and ascending order. To do this, click once on the desired title for descending sort, and twice for ascending sort. Once the table is sorted, an arrow symbol (↓ or ↑) will appear close to the title, so you know on which column it was sorted last. As the sorting is done on the client machine, it may take a while with very large tables.

Pause activity details

This table shows the specific pauses that each agent took and the pause code that was entered for each pause. It also shows whether the pause taken was considered to be billable or non-billable.

Pause activities detail

Export as...

Agent	Ext.	Code	Activity	Billable?	Start hour	End hour	Duration
agent/104	789	30	Lunch	No	06/23 - 0:02:01	06/23 - 0:02:21	0:20
John Doe (101)	123	30	Lunch	No	06/23 - 0:04:11	06/23 - 0:05:11	1:00
agent/103	456	31	Back Office	Yes	06/23 - 0:04:31	06/23 - 0:05:01	0:30
agent/104	789	30	Lunch	No	06/23 - 0:07:01	06/23 - 0:07:21	0:20
John Doe (101)	123	30	Lunch	No	06/23 - 0:09:11	06/23 - 0:10:11	1:00
agent/103	456	31	Back Office	Yes	06/23 - 0:09:31	06/23 - 0:10:01	0:30
agent/104	789	30	Lunch	No	06/23 - 0:12:01	06/23 - 0:12:21	0:20
John Doe (101)	123	30	Lunch	No	06/23 - 0:14:11	06/23 - 0:15:11	1:00
agent/103	456	31	Back Office	Yes	06/23 - 0:14:31	06/23 - 0:15:01	0:30
agent/104	789	30	Lunch	No	06/23 - 0:17:01	06/23 - 0:17:21	0:20
John Doe (101)	123	30	Lunch	No	06/23 - 0:19:11	06/23 - 0:20:11	1:00
agent/103	456	31	Back Office	Yes	06/23 - 0:19:31	06/23 - 0:20:01	0:30
agent/104	789	30	Lunch	No	06/23 - 0:22:01	06/23 - 0:22:21	0:20

Agent history popup

If you click on an agent's name, a new popup will appear with full history for that agent. You can scroll in it as needed by using arrow keys or the wheel of your mouse.

Detail: agent/103

127.0.0.1

Close

Agent	Ext.	Duration	On pause	Activity	Start hour	End hour
agent/103	456	4:52	-	-	06/22 - 23:56:01	06/23 - 0:00:53
agent/103	456	4:52	-	-	06/23 - 0:01:01	06/23 - 0:05:53
			0:30	Back Office	06/23 - 0:04:31	06/23 - 0:05:01
agent/103	456	4:52	-	-	06/23 - 0:06:01	06/23 - 0:10:53
			0:30	Back Office	06/23 - 0:09:31	06/23 - 0:10:01
agent/103	456	4:52	-	-	06/23 - 0:11:01	06/23 - 0:15:53
			0:30	Back Office	06/23 - 0:14:31	06/23 - 0:15:01
agent/103	456	4:52	-	-	06/23 - 0:16:01	06/23 - 0:20:53
			0:30	Back Office	06/23 - 0:19:31	06/23 - 0:20:01
agent/103	456	4:52	-	-	06/23 - 0:21:01	06/23 - 0:25:53
			0:30	Back Office	06/23 - 0:24:31	06/23 - 0:25:01
agent/103	456	4:52	-	-	06/23 - 0:26:01	06/23 - 0:30:53
			0:30	Back Office	06/23 - 0:29:31	06/23 - 0:30:01
agent/103	456	4:52	-	-	06/23 - 0:31:01	06/23 - 0:35:53
			0:30	Back Office	06/23 - 0:34:31	06/23 - 0:35:01
agent/103	456	4:52	-	-	06/23 - 0:36:01	06/23 - 0:40:53
			0:30	Back Office	06/23 - 0:39:31	06/23 - 0:40:01

Undefined agent/104 126 3:28 125 0:20 8.7% 1.0




Agent sessions are written in black, while pauses are detailed in grey under each session they were taken under. For each pause its activity code is shown (if any).

Understanding results: Call outcomes

If your agents are entering Pause codes or Call outcomes, the "Outcomes" tab will let you report on the information they just entered.




Outcomes

General outcomes	
Total billable time:	163:40:48
Total agent available time:	158:52:48
Total agent billable activities:	4:48:00
Total nonbillable time:	12:48:00
Number of Sales:	1,152
Number of Contacts:	1,728
Sales per Hour (SPH):	7.0
Contacts per Hour (CPH):	10.6
Conversion index:	40.0%

Export as...   




Call results, by outcomes

Call outcome	Type	Total	Taken	Lost	%	
Sold	Sale	1152	1152	0	25.0%	<div></div>
Not interested	Contact	1728	1728	0	37.5%	<div></div>
-	None	1728	576	1152	37.5%	<div></div>

Export as...   




Billable activities

Activity	N. Times	Tot. Time	Avg.	Min.	Max.	%	
Back Office	576	4:48:00	0:30	0:30	0:30	100.0%	<div></div>

Export as...   




Non billable activities

Activity	N. Times	Tot. Time	Avg.	Min.	Max.	%	
Lunch	1152	12:48:00	0:40	0:20	1:00	100.0%	<div></div>

Export as...   

Detailed agent report

Agent name	Avail.	Bill.		NonB.		Sales		Cont.	SPH	CPH	CONV
Mike Boo (102)	46:33:36	100.0%	0:00	0.0%	0:00	0.0%	0	1152	0.0	24.7	0.0%
agent/103	41:55:12	89.7%	4:48:00	10.3%	0:00	0.0%	576	0	12.3	0.0	100.0%
John Doe (101)	36:48:00	100.0%	0:00	0.0%	9:36:00	26.1%	576	0	15.7	0.0	100.0%
agent/104	33:36:00	100.0%	0:00	0.0%	3:12:00	9.5%	0	576	0.0	17.1	0.0%

Export as...   

The top panel will display an overview of the situation, showing:

- How much billable time there has been on this system, broken down by ACD/call time (“agent available time”) and billable activities (agent on pause)
- The total non billable time (e.g. lunch, breaks)
- The total number of Contacts and Sales, as defined by call outcome codes
- The *Sales per Hour* (SPH) and *Contacts per Hour* (CPH) ratios
- The *Conversion index*, that is the percentage of sales over the total number of sales and contacts.

There will be details explaining Billable and Non-billable activities, with average, minimum and maximum session durations, and a percentage on all activities of the same kind.

The Detailed Agent Report will show, for each agent:

- The Available (ACD) time, as an absolute value and a percentage of its total time logged on
- The Billable time, as an absolute value and a percentage of its total time logged on
- The Non-Billable time, as an absolute value and a percentage of its total time logged on
- The number of Sales And Contacts the agent had (if a sale is counted as both a Sale and a Contact, it's counted only once as a Sale)
- The *Sales per Hour* (SPH) and *Contacts per Hour* (CPH) ratios for this agent
- The *Conversion ratio*, that is the percentage of sales over the total number of sales and contacts.

Showing call details

As shown above, QM lets you see the very detail of calls handled by Asterisk.

Detail of answered calls

The screenshot displays the QueueMetrics call center monitor interface. At the top, there's a navigation bar with tabs: Home, Answered (selected), Unans., Area, Att., and Distrib. The main heading is "Detail of answered calls". Below this, there's a "Report Details" section with fields for Atomic queue(s) considered, Period start date, Period end date, and Total calls processed. A "Call Detail" window is open, showing the following information:

- Asterisk Call ID: 1182549661=10.21
- Date and time: 23/06/2007 - 00:03:51
- Caller ID: (555)555-1111
- Handled by: agent/101
- Duration: 150 sec.
- Waiting time: 10 sec.
- Disconnection cause: Agent disconnected
- Transferred to:
- URL: http://www.loway.it
- Status code: 20: Sold
- Srv

Below the "Call Detail" window, there's a "Queue details" section with an "Export as..." button. A table lists the details of answered calls:

Date	Caller	Queue	Wait	Duration	Disconnection	Handled by	Attempts	Code	Stints	Srv
06/22 - 23:59:41	*	q2	0:20	0:20	Caller	agent/104	0		1	
06/23 - 0:01:51	(555)555-4444	q2	0:10	1:00	Caller	agent/103	1	20	1	
06/23 - 0:02:31	(555)555-2222	q1	0:20	0:40	Agent	Mike Boo (102)	1	21	1	
06/23 - 0:01:11	(555)555-1111	q1	0:10	2:30	Agent	John Doe (101)	2	20	1	
06/23 - 0:02:51	(555)555-5555	q2	0:10	1:00	Caller	agent/104	1	21	1	
06/23 - 0:03:51	(555)555-3333	q1	0:10	0:30	Caller	Mike Boo (102)	1	21	1	
06/23 - 0:04:41	(555)555-6666	q2	0:20	0:20	Caller	agent/104	2		1	

For each answered call, the following information is shown:

- Date and time for the call;
- The Caller*ID, if available (the Caller*ID format may differ according to your local Telco – in some countries it include the full name of the caller, in other it might be a number and in others it may be unavailable at all);
- The queue that handled the call;

- The total waiting time before the agent was connected;
- The duration of the call, talking to an agent;
- The cause of disconnection;
- Which agent or terminal handled the call.
- How many agent attempts were made before this call was answered
- The call completion code your agents entered
- How many stints make up this call
- The server that handled this call (in the case of clusters)

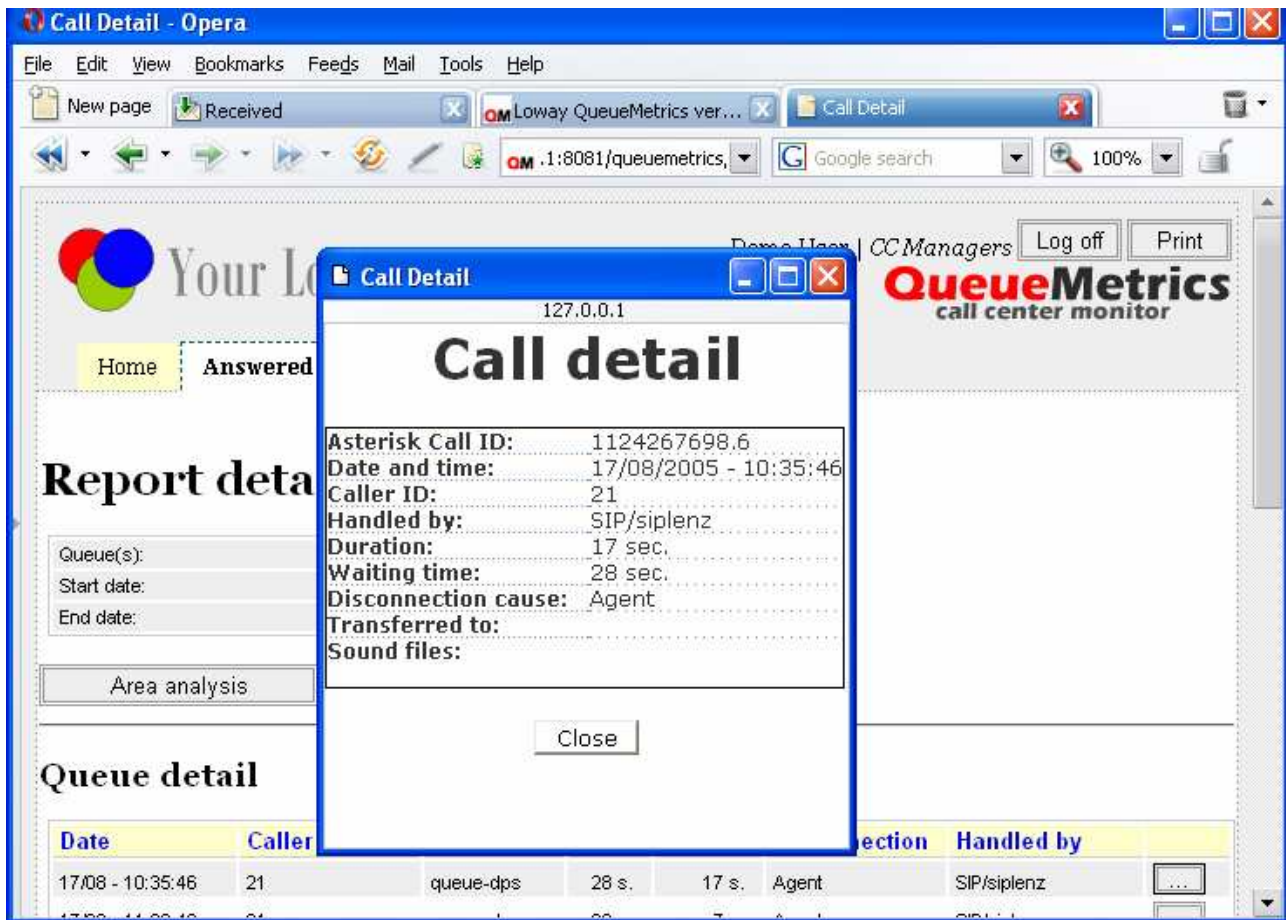
It is possible to sort the table for each title, in either descending and ascending order. To do this, click once on the desired title for descending sort, and twice for ascending sort. Once the table is sorted, an arrow symbol (↓ or ↑) will appear close to the title, so you know on which column it was sorted last. As the sorting is done on the client machine, it may take a while with very large tables.

If you click on the small icon on the right, it will be possible to see the details of the call, including:

- Asterisk's internal Call-ID code
- The call date and time
- The caller-id (if any)
- The agent handling the call
- The call duration
- The wait time
- The disconnection cause
- The extension the call was transferred to
- The URL that was linked to this call through the Queue() command, if any
- The call status code
- The server that handled this call
- The sound files (one or more) that were recorded for this call (see below).

Listening to answered calls

Clicking on the button with three dots near to a call opens a detail popup, like the one below:



For each call, the recorded pieces of information are shown.

If the call was monitored, i.e. recorded to disk, a number of sound files may be shown. By clicking on a sound file you can listen to it straight from your browser.

Please note that:

- The recorded file name must contain the Asterisk Call ID for QM to relate it to the call – see page 115 for tips on how to configure Asterisk correctly to implement this feature;
- The audio storage on the Asterisk server must be readable by the servlet container;
- You must have the correct sound codecs to listen to the sound file on your PC. WAV files usually work out of the box but are comparatively quite big, while GSM files require an additional codec pack on most Windows machines but consume disk storage much more efficiently. The best compromise is usually to use the WAV49 format on Asterisk, that is played natively by Windows machines but has a compression and sound quality comparable to the GSM format
- Asterisk will usually record two different sound files – one for the caller and the other for the agent – and will then mix them together at the end of the call. If

this does not happen automatically, you might find two different sound files, named “-in” and “-out”, each of which contains the voice of one of the parties. If your call is a multi-stint call, you may find a number of different sound files for it.

Detail of unanswered calls

The unanswered calls detail is quite similar to that of answered calls.

[Home](#)
[Answered](#)
[Unans.](#)
[Area](#)
[Att.](#)
[Distrib.](#)
[Agents](#)
[Outcomes](#)
[All](#)




Detail of unanswered calls

Report Details:

Atomic queue(s) considered:	00 All
Period start date:	June 23 2007, 0:00
Period end date:	June 23 2007, 23:59
Total calls processed:	1,002
	75.0% ans / 25.0% unans

[Area analysis](#)
[Return](#)

Queue details

Export as...   

Date	Caller	Queue	Disconnection	Position	Wait	Attempts	Code	Key	Stints	Srv
06/23 - 0:00:11	(555)555-0001	q1	Abandon	1	0:10	1			1	
06/23 - 0:03:41	(555)555-0002	q2	Abandon	1	0:10	1			1	
06/23 - 0:05:11	(555)555-0001	q1	Abandon	1	0:10	1			1	
06/23 - 0:08:41	(555)555-0002	q2	Abandon	1	0:10	1			1	
06/23 - 0:10:11	(555)555-0001	q1	Abandon	1	0:10	1			1	
06/23 - 0:13:41	(555)555-0002	q2	Abandon	1	0:10	1			1	
06/23 - 0:15:11	(555)555-0001	q1	Abandon	1	0:10	1			1	

The following data are shown:

- Date and time of the lost call;
- Caller-ID;
- Queue that handled the call;
- Disconnection cause;
- Position at disconnection, if available;


- Waiting time before disconnection, if available.
- Agent attempts made before disconnection
- The call code, if entered (this might be added automatically by some outbound diallers that will mark unsuccessful attempts as “lost calls”)
- The key pressed on disconnection (if any)
- The number of stints this call has
- The server that handled the call

Please note that on a queue timeout, Asterisk will not report the waiting time, as it is fixed and same as the queue timeout.

It is possible to sort the table for each title, in either descending and ascending order. To do this, click once on the desired title for descending sort, and twice for ascending sort. Once the table is sorted, an arrow symbol (↓ or ↑) will appear close to the title, so you know on which column it was sorted last. As the sorting is done on the client machine, it may take a while with very large tables.

The real-time status panel

The real time status panel can be accessed by clicking the “Start real-time monitoring” label from the home page. It will show a page similar to the one below:


Your Logo


[Home](#)
[Realtime](#)

Realtime call center monitoring - 22:52:27

Queue(s): Q1: Sales, Q2: Support

[Reload now](#)
[Hide calls](#)
[Hide agents](#)
[Show all queues](#)
[Show members only](#)
Location

	Queue	N. agents	Ready agents	On pause	Unk	Bsy	N. Calls waiting	On phone inbound	On phone outbound
	All selected	4	1	0	0	0	0	3	0
	Q1: Sales	1	0	0	2	1	0	2	0
	Q2: Support	0	0	0	1	0	0	1	0

Export as...

Calls being processed:

	Queue	Caller	Entered	Waiting	Duration	Agent	Srv
	Q1: Sales	(555)555-1111	22:50:12	0:10	2:06	Martin (101)	
	Q1: Sales	(555)555-2222	22:51:32	0:20	0:36	Dave (102)	
	Q2: Support	(555)555-5555	22:51:52	0:10	0:26	Alan (104)	

Export as...

Agents currently logged in:

	Agent	Last logon	Extension	On pause	Srv	Last call	On queue
	Martin (101)	06/23 - 22:50:02	123	-			Q1: Sales
	Dave (102)	06/23 - 22:50:02	345	-			Q1: Sales
	Fletch (103)	06/23 - 22:50:02	456	-		22:52:02	Q2: Support
	Alan (104)	06/23 - 22:50:42	789	-			Q2: Support

Export as...

In order to maintain session information, this page will reload automatically

On the top of the page there is a table showing which calls are currently handled by the queue system, and the agents logged in at the moment. This page is invaluable as it refreshes automatically and can tell you in a glimpse what's happening in the call center; it is meant to stay open in a window on the CC manager's workstation to have the exact feeling of what is going on at the moment.

On the sample page above, you can see three calls and four connected agents. Just like in the main analysis, you can choose which queues you want to monitor to avoid being overwhelmed by data.

You can also see that the current call environment has triggered a number of yellow and red alarms, as specified in the queue definition. You can configure red and yellow alarms for most numeric values that appear on screen – see the chapter *Setting attention levels (Red and yellow alarms)* on page 96. You can also set sounds linked to yellow or red alarms, that will be played if a red or yellow alarm is present.



On the bottom of the screen, a moving bar shows the time left before the next reload. The page reloads automatically in order to show what's happening almost in real time. You can anyway force a faster reload by clicking the "Reload" button.

Top status panel

The top status panel shows a quick status report for the current situation.

The first line shows information for all selected queues as a sum, while if there is relevant information for a specified queue it is displayed in a separate line. If an alarm is triggered for one of the numeric values displayed, the relevant cell turns either yellow or red.

The displayed fields have the following meanings:

- **Queue:** The name of the queue. Inbound queues are marked with the symbol , while outbound queues use the symbol .
- **N. agents:** how many agents are logged on to the system, in total
- **Ready agents:** how many agents are ready to take calls, i.e. are logged on and are not in conversation or on pause
- **On pause:** how many agents are currently on pause
- **Unk:** how many agents are currently in conversation, but are not currently known as member of this queue
- **Bsy:** how many agents who are both members of the given queue and some other queue are currently busy because they are on call on the other queue.
- **N. Calls Waiting:** how many inbound calls are currently waiting in the selected queue. Outbound queues never have any call waiting.
- **On phone inbound:** how many agent are talking on the selected inbound queue

- *On phone outbound*: how many agents are talking on an outbound queue

Please note that – as agents are not linked to a specific queue save for the moment they are actually talking to a caller on the queue – the agent information is computed for all agents on the Asterisk server and not for specific queues, unless the “Show members only” button is pressed.

Calls being processed

A list of calls flowing through the selected queues is presented on the middle table. If no call is present the table is displayed empty.

When a call is processed, the following fields are shown:

- Queue: the queue that is handling the call;
- Caller: The Caller*ID, if available;
- Entered: The date and time the call entered the queue system.

If the call is not answered yet, the “Waiting” field is displayed in red and is calculated according to the current date and time of the server. If your QM is on a different server, make sure the clocks are exactly aligned or you may see strange values in this field. The NTP protocol offers excellent clock synchronization precision and is available on most operating systems.

When a call is answered, the “Waiting” field tells the time that the caller had to answer; the “Agent” field shows the agent (or terminal) the caller is talking to and the “Duration”, in red, is the current call duration.

If the call is ongoing and connected to an agent, the Call Monitor and VNC Monitor icons may be present. By clicking on one of these icons you activate the specified monitoring (see below).

As soon as a call is completed or hung up, it exits the Calls panel.

This panel can be turned on or off again by pressing the “Hide calls” button on the top of the page.

The “Srv” column is used only in cluster-based environments to tell you on which server the call is being handled.

Agents currently logged in


A list of available agents for all queues is displayed in this field. For each agent, the name, last log on and extension – if logged in via call-back – is provided.

A graphical indication of the status of each agent is shown using a coloured dot, where the following cases are possible:

- *Green dot*: the agent is ready to take calls
- *Yellow dot*: the agent is currently on a call
- *Red dot*: the agent is currently on pause

If enabled, a VNC Monitoring icon can be shown close to the agent's name; by clicking on it you'll be able to open the VNC monitoring page.

This panel can be turned on or off again by pressing the "Hide agents" button on the top of the page.

The *Queue(s)* field shows the queues an agent is logged on to. This is meaningful only for agents who log-in on a queue-by-queue basis using the AddMember command in Asterisk. If an agent logs on to all queues he's enabled to work on, a small database  logo may be shown, telling the viewer that the agent is linked to queues through the Asterisk's configuration.

As the queue_log file usually contains no information on what queue(s) an agent is a member of, usually all agents are shown when they log on, no matter to what queue they will work on. This might be a problem for larger call centres, so it is possible to see only calls and log-ons of agents that are a member of the current queue.

Membership is set by clicking on the "Agents" button of the queue settings page. Make sure your queue membership data is up-to-date before clicking this button!

If the "Show members only" button was not pressed, all agents logged in on Asterisk will be shown, no matter to which queue they belong. If you press the "Show members only" button, only agents defined for the selected queue will be displayed.

The *On Pause* field will contain the time the agent went on pause; if the agent is using a pause code to mark the reason for going on pause, the decoded pause code is shown as well.

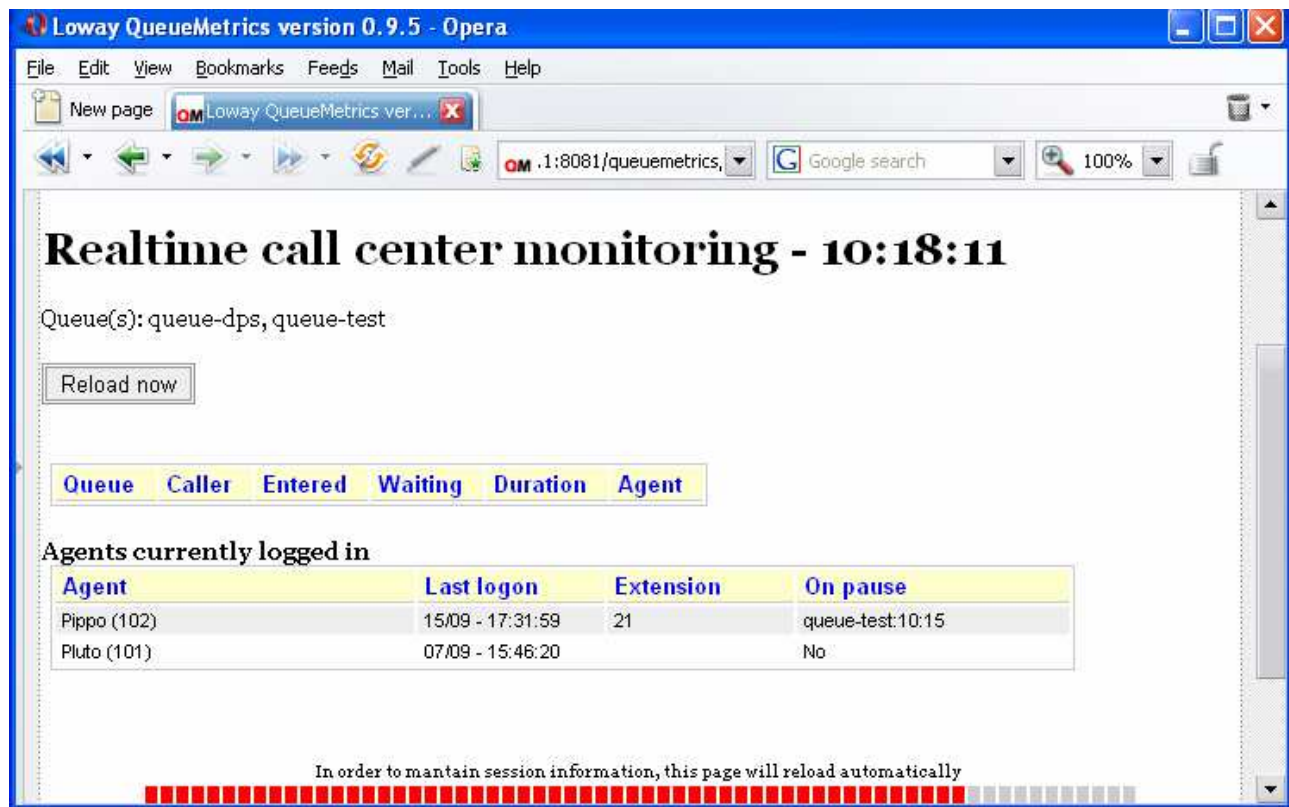
The *Last call* and *On queue* fields show the start or disconnect time of the last call the agent handled (which is latest) and on which queue the last call was. This can be useful to diagnose queue strategy problems that lead to unfair call distribution, or agents having problems with their telephones and therefore not taking calls correctly.

You can also assign each agent to a Location, i.e. a group of agents working together that you want to monitor as a unit. They might be a supervisor's team, or people working in the same building, or maybe in the same location for big multinational call-centres. This way you can avoid being cluttered with information about all agents working on the selected queue(s) and only see those you are actually interested in.

Note that when you filter by agents or locations you may see calls being queued and then disappear. This happens because all calls that wait to be answered on a queue are shown, but answered calls are shown only if the answering agent is a member of your defined filter conditions.

Imagine you have two groups of agents, one working in NY and the other one in LA. You are the supervisor of the NY group, so you are filtering by location. You see a call

entering your queue and then disappear. The reason why might be that it was answered by someone in LA, so QueueMetrics filters it out for you.



If an agent goes on pause, like Agent 102 in the screenshot above, the queue(s) he has paused from are shown and so is the time he left for pause².

Unattended call and VNC monitoring

It is possible for you to set up both Unattended Call Monitoring and VNC Monitoring.

Unattended Call Monitoring makes it possible to listen to an ongoing call from an agent; by clicking on the small telephone-shaped icon a popup will appear where you may enter your current extension or PSTN telephone number.

² Agent pauses are only available on Asterisk after version 1.2.

Remote Call Monitoring - Opera

File Edit View Bookmarks Feeds Mail Chat Tools Help

Your Logo Demo Admin | Administrators Log off Print

QueueMetrics
call center monitor

Home

Realtime call 0:47:53

Queue(s): q1, q2

Reload now Hide calls

Queue	N. agents
All selected	3
q1	0

Calls being processed:

Queue	Caller
q1	(555)555

Agents currently logged

Agent	Last login	Extension	On pause
Pippo (101)	14/06 - 23:46:53	123	No
Pluto (102)	14/06 - 23:46:53	345	No
Tizio (103)	14/06 - 23:46:53	456	No

In order to maintain session information, this page will reload automatically

Loway
research
Loway Research

As soon as you confirm the entered data, your telephone will ring and you will start listening to the ongoing call between the selected agent and the caller.

In order to set up this behaviour, please see the section *Enabling Unattended Call Monitoring* on page 119.

It is also possible to set up the system in order to allow the real-time monitoring of the agent's screen using VNC. If this feature is enabled, a small screen will appear close to the agent's name; by clicking on it, your selected VNC application will be launched and you will be monitoring the agent's screen.

In order to set up this behaviour, please see the section *Enabling VNC Monitoring* on page 119 of this manual.

The real-time live page

The real-time live page can be accessed by clicking on the “Live” tab next to the “Realtime” tab.

Live call center monitoring - 22:59:05

Queue(s): 00 All

Reload now

Server	Queue	Tot.	Free	Pause	Talking	Other q.	Logoff	Length	Max wait
aleph	Q DPS	1	0	0	1	0	5	0	0:00
aleph	Q Test	1	0	0	0	1	3	0	0:00

Export as...

Calls being processed:

Server	Queue	Called ID	Wait	Talk	Q.Pos	Agent	Entered	Status
aleph	Q DPS	896	0:00	0:34		John Doe (101)	06/23 - 22:58:31	AG

Export as...

Agents currently logged in:

Server	Agent	Status	Logon	Queues
aleph	John Doe (101)	Call	8:52:56	queue-test queue-dps

Export as...

Server status

Server	Status	Time (ms)
aleph	OK	234
trix	OK	187

Export as...

In order to maintain session information, this page will reload automatically

This page is not built from the queue_log data as all other information reported by QueueMetrics, but it's read right from each Asterisk box's Manager interface. So what you see in this page is the status of each Asterisk box, as reported by itself.

This feature is still less developed than the Realtime page, but still can be pretty useful.

The top panel

On the top panel, for each queue on each server, you will see the following pieces of information:

- *Tot*: the total number of agents available for this queue
- *Free*: the number of free agents
- *Pause*: the total number of agents on pause
- *Talking*: the total numbers of agents who are in conversation at the moment
- *Other q.*: the number of agents that are logged in to this queue and some other queue, and are at the moment busy on another queue.
- *Logoff*: the number of possible agents that are defined for this queue but are not logged on at the moment.
- *Length*: the current queue length, i.e. how many calls are waiting in line before being connected to an agent
- *Max wait*: the current maximum wait time for this queue.

Calls being processed

In the calls panel you see the following pieces of information, sorted from oldest to newest by call start-time:

- *Server*: the server that is handling this call
- *Queue*: the name of the queue
- *Caller-id*: the caller-id of this call, if any
- *Wait*: the wait time (if the call is not connected)
- *Talk*: the total duration (if the call is connected)
- *Q.Pos*: the queue position (if the call is waiting)
- *Agent*: the agent handling this call, if connected
- *Entered*: the time this call was queue
- *Status*: the call status

The Wait and Talk times cannot be distinguished at the moment.

Agents currently logged in

In the agents panel you can see the following pieces of information:

- *Server*: the server your agent is logged on to
- *Agent*: the agent

- *Status*: if the agent is free, paused or on call
- *Logon*: the time this agent logged on
- *Queues*: to which queues is this agent connected at the moment

Server status

The last panel details the status of each server making up the cluster. If a server is not correctly set up, it will appear as KO.

- *Server*: the server that QM is polling
- *Status*: OK – the server answered correctly; KO – it was impossible to retrieve information from this server
- *Time*: how much querying this server took. If this value goes up all of a sudden, your server is likely experiencing overload.

Enabling the real-time live page

To enable the real-time live page you must do the following:

- Make sure that your users have the RTLIVE security key
- Make sure you have a clustering set up and the manger interfaces are set correctly. You can even not use clustering for reporting, though the manager interfaces will be read through the *cluster.***.manager* properties.
- Make sure that Asterisk has the manager API enabled, and that your IP address, login and password are correct. E.g. *tcp:dial:bingo@10.10.3.100* will tell QueueMetrics to connect to the manager port on server *10.10.3.100* and use the user *dial* with password *bingo* to log on.

Help! My Real-time and Live pages display different results!

In this case most likely the Live page is correct. This is due to the fact that sometimes Asterisk will not log some events correctly, and so the status of the call-center inferred from the queue_log file may end up not being correct.

If this happens to you:

- Log agents off and on again
- Check that Asterisk is correctly installed
- Check that error queue exist (e.g. timeouts) log their status correctly

The real-time agent page

QM lets each agent have his/her own page, where they can see the current flow of calls they have just answered and launch external CRM web apps.

This is quite useful, because:

- Each agent can see their own status, i.e. whether they're logged on or they've been disconnected;
- Each agent can see their last calls, including information like Caller ID, duration and waiting times;
- Each agent can see from which queue the call is coming, even if they lost the announcement message;
- Each agent can launch external web apps – like CRM software – that might be automatically linked to the Caller*ID or other information input by the caller.

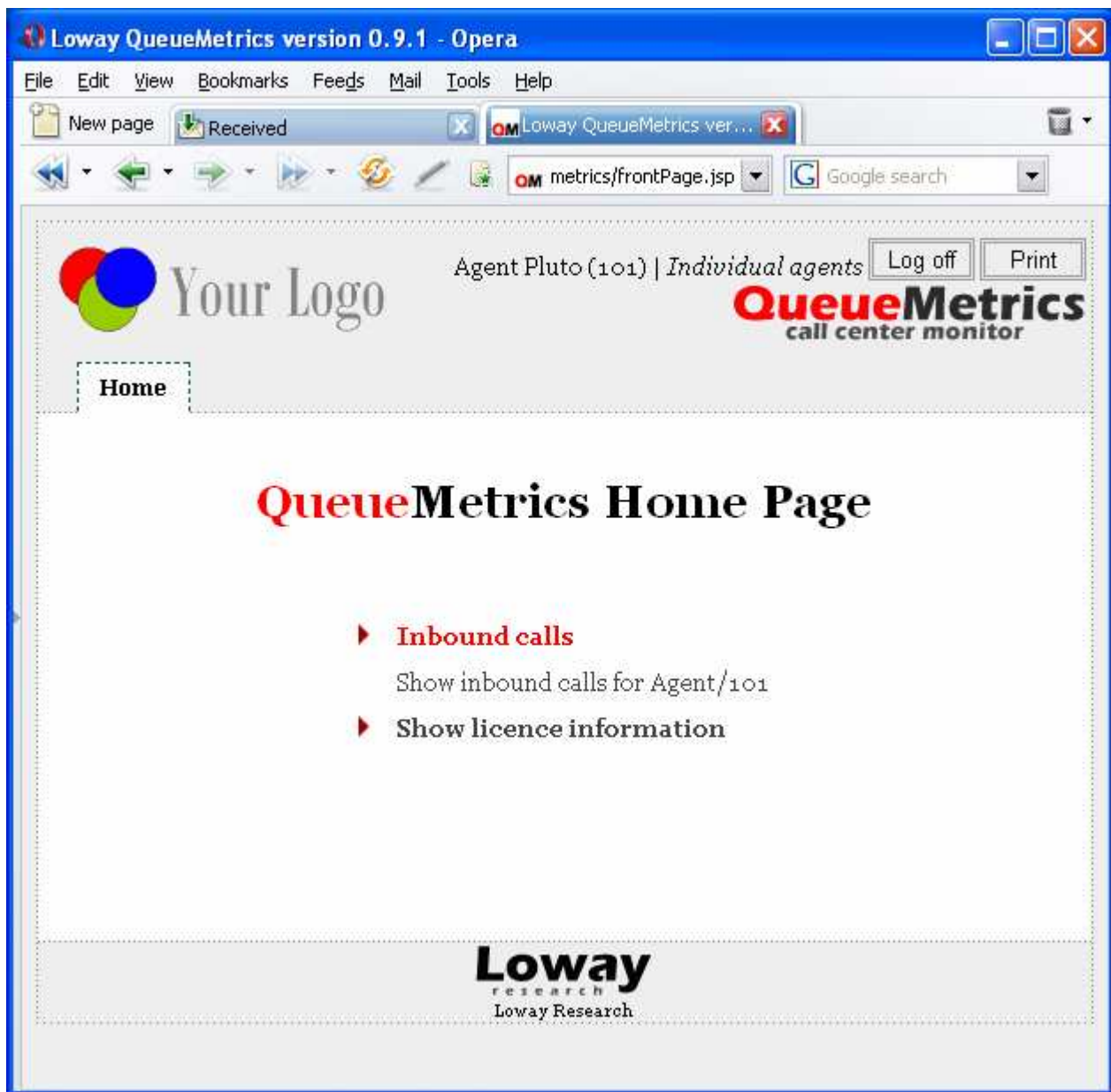
Also, it is possible to use this page in order to:

- Log the agent on and off to Asterisk
- Pause and unpause the agent, entering the pause activity code if needed
- Set the outcome code for each call

To avoid excessive consumption of system resources on big centres, only the most recent part of the log file is processed and so only a few calls are shown.

In order to use this feature, a user must be configured as having the same login as the Asterisk agent string (e.g., "Agent/101") and it must hold the key AGENT. Our suggestion is to use the same set of credentials the agent uses to login to the Asterisk system.

When the agent logs on – you can try this by using the demo account **Agent/101**, password **999** - s/he sees a reduced home page like the following one:



By clicking on “Show inbound calls”, the agent is led to the inbound calls page:

Active calls for agent Martin

agent/101: Agent is currently logged on

Reload now Log on Log off Pause Unpause

Entering at	Waiting	Talking	Caller ID	Queue	URL	Status	Transfer to	Outcome
06/23 - 23:25:12	0:10	0:52	(555)555-1111	Q1: Sales	Open			20: Sold
06/23 - 23:20:12	0:10	2:30	*	Q1: Sales	-	Terminated		

In order to maintain session information, this page will reload automatically

Loway
research
Loway Research

This page tells the agent that he's logged on and shows the last calls he has handled.. In this case we see that there is an ongoing conversation and many previous calls.

The fields are as follows:

- *Entering at* is the date and time the call entered the queue system;
- *Waiting* is the waiting time before being connected to the agent;
- *Talking* is the talking time for that call. If the call is ongoing, the time will be estimated and written in red.;
- *Caller ID* is the call's identification, if available;
- *Queue* is the queue handling the call;
- *URL* (if present) is a clickable link that opens a third party CRM app. It is defined in the Queue command on Asterisk. The agent opens the third party CRM app by clicking on it. If the URL contains the sequence "[A]", it is substituted with the actual agent handling the call.
- *Status* is whether the call is ongoing or terminated;
- *Transfer* is the extension the call was transferred to (if any).
- *Outcome* is the call outcome that the agent can set for this call

On top of the page a field tells the agent whether s/he is:

- *Logged on*: ready and able to take calls;
- *Logged off*: the agent has voluntarily left or has forcibly been disconnected by the queue system;
- *On pause*: the agent has asked for a pause from the queue system;
- *Undetermined*: whether there is no relevant information to tell the agent status in the last part of the log file.

To avoid hammering the QM server with excessive work, only the last 60k or so of the log are analyzed. This parameter can be fine tuned by the system administrator in order to maximise usefulness without creating an excessive server load (see page 108).

Whenever an agent receives a call, s/he should press the “Reload” button on the page in order to see the incoming call.

The page reloads automatically every two minutes in order to maintain the user logged on in QM.

Using the agent's page to control advanced features

The buttons on the agent's page can be used to log agents on and off, pause and unpause them and set the call status. When pressed, each button will open up a pop-up window asking for details:

The screenshot shows the QueueMetrics 'Agent activities' page for agent Martin. A pop-up window titled 'Agent pause' is open, showing 'Agent's code: 101' and 'Pause reason: Lunch break'. The background page shows 's for agent Martin' and 's currently paused (30: Lunch break)'. Below this are buttons for 'Log on', 'Log off', 'Pause', and 'Unpause'. A table at the bottom shows call history with columns: Queue, URL, Status, Transfer to, Outcome. The first row shows 'Q1 Sales', 'Open', 'Terminated', and '20: Sold'. At the bottom, a message states: 'In order to maintain session information, this page will reload automatically'.

Queue	URL	Status	Transfer to	Outcome
Q1 Sales	Open	Terminated		20: Sold

Once the user clicks on Run, the command will be sent to the Asterisk server and the page will be reloaded. It is possible that on very busy machines the commands may be delayed a few seconds, so that it is necessary to reload the page manually in order to check that the command has run successfully.

Please note that for this to work it is necessary that Asterisk has been configured to manage Agent actions – it will NOT work on an Asterisk server that's not been specifically configured to work with it!

See section *Enabling Agent's page actions* on page 120 in order to set up this feature.

If you want to have agents logging on, it is mandatory that the underlying Asterisk agents are defined without a password.

Multi-stint calls

In QueueMetrics, we define a multi-stint call as a call that was processed on more than one queue, with one or more queue terminating it for timeout, transfers or key exits.

In the standard QueueMetrics reporting mode, this call would be seen as a series of “lost calls” on one or more queues, possibly followed by a taken call if the call was answered at all; the system does not notice that those events happened on the same call.

Running QueueMetrics in multi-stint mode, calls will be grouped together based on the call’s Unique-ID, and a single call will be rebuilt as a multi-stint call so that:

- The call looks like it was handled on the first queue it was presented on; the queue enter time and queue position are those of the first queue.
- The call will be considered “answered” if the last stint is an answered call, “lost” in all other cases
- The wait time will be the sum of the wait times on different queues (if there are intermediate wait times, like those for IVR menus, they will not be counted)
- The talking time and agent taken the call will be taken from the last taken call
- All stats (number of call, call distribution, etc) will be counted on the new multi-stint calls.

Limitations and side-effects

Multi-stint calls aren’t for everyone. There are a number of limitations and side effects you should be aware of before attempting to run QueueMetrics in this mode:

- Calls are grouped by the Asterisk Unique-ID code; this means that if different call stints happen on different servers in a cluster, they will not be grouped together
- All queues the call passed on must be included in the report. If you include only the master queue, stints on other queues will not be seen.
- Because of the previous bullet, it is usually better to configure separate “wrap up” or “timeout” queues, that is, instead of having both a Sales and a Support queue that will send people to the General queue on timeout, it would be better to have “Sales” → “General-sales” and “Support” → “General-support”, even if “General-sales” and “General-support” are actually aliases of the same queue.
- All stints of all calls must be included between the starting and ending report times. Stints that start before the start data or end after it will be ignored.

- Run time and memory will be comparatively more than a standard analysis, as the grouping and additional data stored take their toll on the system
- Stint-grouping does not work for real-time analysis.

Multi-stint calls in QueueMetrics

If you run calls with multi-stint mode enabled, the string “Multi-stint calls joined together” will appear on the top panel, and the number of joined together calls will be shown.

Period start date:	May 01 2007, 8:00
Period end date:	May 30 2007, 18:00
Total calls processed:	15,316
	88.5% ans / 11.5% unans
Multi-stint calls joined together	173

The distribution of taken calls by stints will be shown in the “Answered calls” tab:

Answered calls, by stints

Number of stints	N. Calls	...
1	13427	99.1%
2	124	0.9%
3	3	0.0%

Export as...

The distribution of lost calls by stints will be shown in the “Lost calls” tab; aggregate calls by stints will also be shown in the “Lost Calls” page:

Unanswered calls, by stints

Number of stints	N. Calls		...
1	1716	97.4%	
2	43	2.4%	
3	3	0.2%	

Export as...

All calls, by stints

Number of stints	N. Calls		...
1	15143	98.9%	
2	167	1.1%	
3	6	0.0%	

Export as...

On the “Taken calls” list, multi-stint calls will have a blue icon next to the magnifying glass icon; by clicking on that icon it will be possible to access stint details for that call.

Stint Detail

127.0.0.1

Detail of call stints

Entered	Wait	Queue	Disconnection
05/02 - 10:33:24	61	qh...	Timeout
05/02 - 10:34:25	63	qh...	Caller

Close

The same thing will be true for lost calls:

05/02 - 17:20:42		Abandon	2	1:12	0	1	aleph
05/02 - 17:23:05		Abandon	1	3:33	0	1	aleph
05/02 - 17:24:38						1	aleph
05/02 - 17:27:13						1	aleph
05/02 - 17:27:23						1	aleph
05/02 - 17:29:55						1	aleph
05/02 - 17:31:09						1	aleph
05/02 - 17:35:27						1	aleph
05/02 - 17:41:49						1	aleph
05/02 - 17:57:37						1	aleph
05/02 - 18:00:07						1	aleph
05/02 - 18:09:47						1	aleph
05/02 - 18:12:23						1	aleph
05/02 - 18:12:38		Abandon	1	1:03	0	2	aleph
05/02 - 18:14:44		Abandon	3	0:47	0	1	aleph
05/02 - 18:14:44		Abandon	1	2:14	0	1	aleph

Stint Detail

127.0.0.1

Detail of call stints

Entered	Wait	Queue	Disconnection
05/02 - 18:12:38	61		Timeout
05/02 - 18:13:39	2		Abandon

Close

The visitor's page

If you run a call center, it is sometimes desirable to offer some strangers access to the system in order to demonstrate the quality of the work you are doing. Of course you do not want them to access directly the full reports pages, but still you want them to be able to get a hold of the current activity and, optionally, to monitor your agents directly, listening to the conversation or having a look at the agent's screen.

This feature may be handy, for example, if your call center handles inbound queues for third party clients. If you let your users access the QueueMetrics server, they will be able to see your work in real-time. Or maybe you want the marketing department to check the quality of your work, without giving them a fully-featured report that may be hard to understand for a complete stranger.

To solve this problem QueueMetrics implements the VISITOR profile; when a visitor logs on, they see a web page like the following one:



They can then select one or more queues that they have the privilege to see, and click on “Show current system activity”.

Loway QueueMetrics version 1.2.0-b1 - Opera

File Edit View Bookmarks Feeds Mail Chat Tools Help

Sample Visitor | Visitors Log off Print

QueueMetrics
call center monitor

Home

Visitor Realtime call center monitoring - 17:32:50

Queue(s): Coda 1, Coda 2 OUT

Reload now

Calls being processed:

Queue	Caller Id	Waiting	Duration	Agent	Ext.
Coda 1	(555)555-1111	0:10	0:28	Pippo (101)	101

Today at a glance:

Queue	N. calls	Avg wait	Avg talk
Total calls inbound	631	13.3 s.	73.3 s.
Coda 1	631	13.3 s.	73.3 s.
Total calls outbound	632	13.3 s.	46.6 s.
Coda 2 OUT	632	13.3 s.	46.6 s.

Queue	N. calls	Avg wait
Total lost calls inbound	211	10.0 s.
Coda 1	211	10.0 s.
Total lost calls outbound	210	10.0 s.
Coda 2 OUT	210	10.0 s.

In order to maintain session information, this page will reload automatically

This page looks very similar to the real-time page, where the calls for the given queues are shown in real time. If the user is given the MON_AUDIO or MON_VNC keys, he will have the opportunity to click on the VNC or the unattended audio monitoring icons and start the procedure exactly the way it happens for the real-time page.

In addition to the data about the real-time activity, the user will be able to see a report of the number, average duration and average wait time for answered and unanswered calls on the selected queue(s).

The page reloads automatically just like the real-time page, or can be reloaded by clicking on the “Reload now” button.

Setting up VISITORS in a real life scenario

- You may be running a number of queues for different clients, and you do not want one client to see the others’ queues. This is obtained very simply by protecting each queue with a different key and then assigning each visitor the correct key
- You want some visitors to use unattended audio or VNC monitoring. Distribute or revoke the keys MON_VNC and MON_AUDIO accordingly.
- A sample visitor user has been created for the demo database that ships with QueueMetrics; it is called *demovisitor* with password *demo*.

Using Supervisors

A supervisor, for what QueueMetrics is concerned, is a user holding the key SUPERVISOR. One such user has the ability to be assigned to the known agents as their supervisor and to run a report with the additional criterion of filtering the results for all agents he is the supervisor of. This will work in much the same way as the current Location reporting.

On the main page and on the Custom report analysis, if a user is a Supervisor, he will have an additional option: “Run the analysis for my competency”. This option will at the moment be mutually incompatible with Location filtering (if both are chosen, an error will be shown). The analysis will proceed as usual.

In the real-time page there will be a new toggle button “Competency” to filter agents by the competency. Even in this case the filter may not be used together with Location filtering.

Automating statistics download: the ROBOT profile

It is sometimes desirable to obtain a snapshot of the reports QueueMetrics produces at a given moment in time for future access or for uniformity of comparison. You may, for example, want to store on disk a snapshot of the current daily activity every day at 19.00, for future reference.

The ROBOT profile was thought for this purpose: automating access to the wealth of statistics that QueueMetrics is able to provide.

To set this up, first make sure that you have at least one user holding the key ROBOT that is used for remote access. A sample user called *robot* password *robot* is provided in the sample database that ships with QueueMetrics.

Point your browser to the QueueMetrics server with a URL like the following:

```
http://server/queuemetrics/qm_rep.do?user=robot&pass=robot&queues=q1&period=t0
```

will download today's report – the full version - for queue “q1”, while the following one

```
http://server/queuemetrics/qm_rt.do?user=robot&pass=robot&queues=q1|q2
```

will download the realtime page for queues “q1” and “q2”, and

```
http://server/queuemetrics/qm_wab.do?user=robot&pass=robot&queues=q1|q2
```

will download the realtime wallboard for queues “q1” and “q2”.

It is then easy to automate this behaviour using an automated downloader, like for example the *wget* command in the Unix environment.

The following web parameters are accepted by the *qm_rt* (realtime page) and *qm_rep* (report page) generators:

Parameter	Notes	WAB	RT	REP
user	The user name. Mandatory.	X	X	X
pass	The user password. Mandatory.	X	X	X
logfile	The log file to use (with full path). If not defined,	X	X	X

	the default one will be used.			
queues	One or more queues to be analyzed. Use the pipe symbol to separate multiple entries. Use the Asterisk name for each queue.	X	X	X
period	The time period to use. Composed by a single letter plus the number of days to report about. t0: today – t1: yesterday d1: last 24hrs – d2: last 48 hrs			X
filter	A single agent's name, like Agent/123, that will be used as a filter for the analysis.			X
t_from	Initial time, expressed in the format <i>yyyy-MM-dd.HH:mm:ss</i> , e.g. 2006-01-03.12:00:00.			X
t_to	Ending time, expressed in the format <i>yyyy-MM-dd.HH:mm:ss</i> , e.g. 2006-02-04.03:00:00.			X
reloads	Always set to 1 if the session is to generate a reloadable page. Do not use for general report extraction.	X	X	

If you run a report, a time interval must be specified, i.e. you have to supply either a “period” or a “t_from”/”t_to” couple.

In addition to the key ROBOT, your user will need the key QUEUE_AN for reporting and REALTIME for realtime monitoring.

Setting up a self-service wallboard

By using the ROBOT profile in conjunction with reloads=1, it is quite simple to set-up an unattended wallboard for QueueMetrics.

First of all set up a low-cost Linux box to boot in its graphical environment, automatically launch a web browser and go to the following URL:

`http://server/queuemetrics/qm_wab.do?user=robot&pass=xxx&reloads=1&queues=q1|q2`

This command will show an auto-reloading wallboard showing the real-time status of queues Q1 and Q2.

If you connect the new Linux box to a large screen or a video beamer and set it in your call-center where it will be visible by your agents, you have just set-up a wallboard at a very low cost using commodity hardware and requiring no human intervention but turning it on in the morning and turning it off in the afternoon.

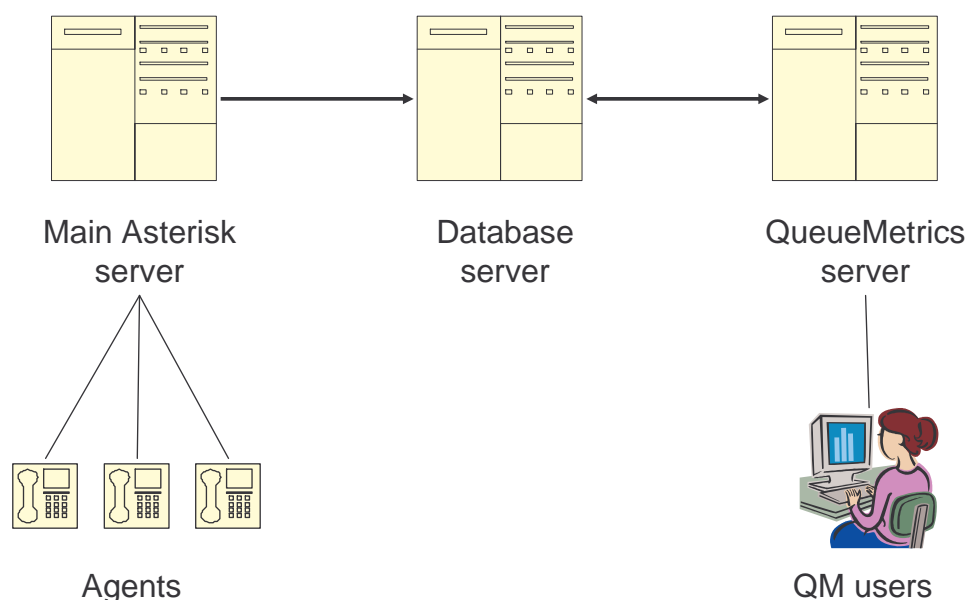
You can do the same with the real-time screen by using the *qm_rt.do* command to create a very simple real-time monitor running all day long for your supervisors.

Storing queue data on MySQL

QueueMetrics lets you store the queue_log data on a MySQL table and is able to produce the very same analyses – including real-time analyses – from data stored on a database.

This scenario is mostly useful for large call centres, where the queue_log data starts to be quite large and the main Asterisk server is quite busy handling its own traffic. In this case, it would be a better solution to have QM run on a separate server, so that even if it has to run a huge analysis the main Asterisk server will not be slowed down.

QM lets you have a deployment scenario like the following one:



In this case, we see that we are using two separate servers; one for the database and one for the QueueMetrics server itself. It is possible to use the same server for both the database and QM, or to consolidate the database on an existing database server and QM on an existing servlet container.

It is very important that all the servers share the same system time; this way real-time events will be shown in an exact way³.

Who should use MySQL storage?

MySQL storage is useful in the following scenarios:

- Large call centres with a very busy or mission-critical Asterisk server

³ The *ntpdate* command can be used on Linux to synchronize the system clock to an external timing source with a high degree of precision. Usage in a daily cron script is highly recommended.

- Large QM reports run very often
- A large number of agents reloading QM agents pages
- Clustered call-centres monitored by a single QueueMetrics instance

In smaller environments (up to 10 agents), it is probably overkill to use MySQL storage, because the extra complexity will not be matched by an extra performance advantage.

Understanding MySQL storage

The QM database storage engine was built with a need to adapt to existing MySQL schemas; therefore the database storage option is very flexible.

It lets you:

- Define the names of each SQL field
- Define the name of the SQL table used (it must reside in the same database as the other QM tables)
- Define one or more table partitions

The storage system makes no assumptions on the underlying field layout of the table used, therefore you are free to define each field as you best see fit for your scenario.

To obtain these results, the SQL settings are divided into presets and partitions.

A **preset** is a schema definition to be used, i.e. the names of each field involved in database storage. You can have a number of different presets, e.g. to connect to different tables in the same database. Presets are defined in the *WEB-INF/configuration.properties* file.

A sample preset can be seen here:

```
# Preset 1: standard DB access. Edit to suit your DB needs.
sqlPreset.1.table=queue_log
sqlPreset.1.f_time_id=time_id
sqlPreset.1.f_call_id=call_id
sqlPreset.1.f_queue=queue
sqlPreset.1.f_agent=agent
sqlPreset.1.f_verb=verb
sqlPreset.1.f_partition=partition
sqlPreset.1.f_data1=data1
sqlPreset.1.f_data2=data2
sqlPreset.1.f_data3=data3
sqlPreset.1.f_data4=data4
```

```
sqlPreset.1.f_incr=unique_row_count
```

You can have more than one preset, by entering the same data multiple times under sqlPreset.1....., sqlPreset.2....., sqlPreset.3..... and so on.

The values for each field are:

- *Table* is the table name
- *Time_id* is the first field in the queue_log. This is used for most extractions and should be an access key.
- *Call_id* is the second field of the queue_log
- *Queue* is the third field of the queue_log
- *Agent* is the fourth field of the queue_log
- *Verb* is the fifth field of the queue_log
- *Data1...Data4* are the remaining fields of the queue_log. Currently Data4 is not defined in the queue_log; in case just leave it blank.
- *Partition* is a logical partition of the table.
- *Incr*: as the minimum time detail for Asterisk activity is by the second, it is possible that events that happen on the very same second seem to happen in the wrong, meaningless order when the data is read back from the database. It is possible to define an auto-increment field on the table that is used to make sure that rows are fetched from the database in the same exact order they were inserted into. This table definition is the default for QM since version 1.1

A **partition** is a key under which separate entries are present in the same queue_log table. You could have separate servers – like test and production - uploading each one to a different partition, and each of them would be completely independent. This is also used for clustering scenarios, where a number of different Asterisk server upload data to the same database.

If you use a partition, your *partition / time_id* combo should be an access key for the table, as QM will access the table every time under this plan.

If you do not use a partition, just leave this field blank and make sure that time_id is an access key for the table.

Uploading data to MySQL

There are a number of ways for data to be uploaded into MySQL. If we plan to use the real-time monitoring features, we must upload data to MySQL as events happen, in order to have them seen immediately by QM.

We have developed a very safe script suitable for small to very high volume for high-volume production systems called **qloaderd**. It can be easily started and stopped from the *init.d* commands and comes complete with start-stop scripts. Its main advantages are the following:

- Extra safe: will check for duplicate lines in the database
- Extra safe: will retry loading data on MySQL connection errors
- Creates a full import log
- Can be started/stopped as a standard system service

You can find it under the *WEB-INF/mysql-utils/qloader*; do not forget to read the installation docs that are in *qloader-README* file and to use the correct init-script for your system.

A number of other techniques are available to upload a *queue_log* file as it is being written to a MySQL table, you may also want to consider the Unix named pipe method reported here: <http://lists.digium.com/pipermail/asterisk-users/2005-July/116881.html>

It is also possible to use the old **queueLoader.pl** script also distributed under *WEB-INF/mysql-utils*. To use the script, edit it with a text editor in order to set the MySQL server, database, user and password, like in the following example:

```
my $mysql_host = "10.10.3.5";  
my $mysql_db   = "log_code";  
my $mysql_user = "queuemetrics";  
my $mysql_pass = "javadude";
```

After the database is set, you can upload a whole file (in this case, */my/queue_log*) to a partition named "P01" by running:

```
perl queueLoader.pl /my/queue_log P01
```

You can also upload your *queue_log* file to partition "P02" as it is being written by issuing:

```
tail -n 0 -f /var/log/asterisk/queue_log | queueLoader.pl - P02
```

(do not forget the minus sign, separated by spaces, before the partition name!).

In the future, we expect Asterisk to be able to write *queue_log* data straight to a database via ODBC, so these tricks will not be necessary anymore.

Loading data in QueueMetrics

After you configured the table in *configuration.properties*, using the table is only a matter of inputting “*sql:[partition] | [preset]*” as the *queue_log* file name to analyze. The partition defaults to “” (blank) if absent, while the default preset is 1.

You can do it directly from the “Run custom report” form, or preset the file name in *configuration.properties* as you best see fit by setting the *default.queue_log_file* property.

Examples:

`sql:P03`

Means accessing the partition named “P03” for preset #1.

`sql:X23 | 3`

Means accessing the partition named “X23” for preset #3.

`sql: | 2`

Means accessing the present #2 with no partition, and

`sql:`

Accesses preset #1 with no partition.

If you use agents pages, keep in mind that the value in *realtime.max_bytes_agent* will not be the portion of the *queue_log* to be read, but the time interval (in seconds) that will be read for the current agent (i.e. if set to 10000, it will search agent data for the last three hours or so).

When you enter a “sql:” file name, the error “The file sql: does not exist” means that there is a misconfiguration of the table access fields in *configuration.properties*.

Checking MySQL database status

As it is not very immediate to “see” if a partition is being loaded and how much information is available on the database, we provide a “Mysql Storage Information” page (accessible from the main “Edit settings” menu if the user holds the key `USR_MYSQL`) that provides general database information.

By clicking on the link, a new page will be loaded showing the available partitions; by clicking on the “Details” button, information for the chosen partition is extracted.

[Home](#)

Current storage info

Total number of rows in table: 31.104

Total table space: 4,1 M (Data: 2,7 M - Indexes: 1,4 M)

Partition	Entries	N. calls	From:	To:	Days of data:	Last heartbeat:	
rt	31.104	4.608	2007-06-21 06:26	2007-06-23 06:25	2,0 days	2007-06-23 06:25	...

Note: the "Number of Calls" shown here is meant as a rough estimate and may differ from the one actually shown by the reports.

Details for partition: rt

N. rows in partition: 31.104

Queues:

Queue	Entries	From:	To:	Days:
q1	9.216	2007-06-21 06:26	2007-06-23 06:25	2,0 days
q2	9.216	2007-06-21 06:26	2007-06-23 06:25	2,0 days

Agents:

Agent	Entries	From:	To:	Days:
Agent/101	5.184	2007-06-21 06:26	2007-06-23 06:25	2,0 days
Agent/102	6.912	2007-06-21 06:26	2007-06-23 06:25	2,0 days
Agent/103	6.336	2007-06-21 06:26	2007-06-23 06:25	2,0 days
Agent/104	6.912	2007-06-21 06:26	2007-06-23 06:25	2,0 days

The total number of rows in table and the total table space is shown; for each partition, its minimum and maximum data entries and its "heartbeat", that is fake entries that the qloaderd process will add to notify the server that the connection is still alive even if Asterisk is producing no data.

The "number of calls" is a **very rough** estimate with no logic in it – it may differ a lot from the actual data calculated by reading the log. Only its order of magnitude should match the other reports.

For each partition, all distinct agents and queues are reported, and their first and last appearance on the database. The "Days" is the time difference in days between the first and last reference.

Please note that accessing this page causes a number of table-scan queries to be performed on the MySQL table – the page might become irresponsive or MySQL can be slowed down if your queue_log table is very large.

Optimizing the queue_log table

If you do a number of deletes followed by inserts on the queue_log table, for example because you manually delete a partition and upload data in another one, the table

access plan may become sub-optimal and performance may suffer. The same happens if you upload multiple queue_log instances at once to different partitions, for example if you run a cluster.

In this case, you can manually run the following MySQL query to optimize the table:

```
ALTER TABLE `queue_log` ORDER BY partition, time_id, unique_row_count
```

This might take a while to run and may lock your table until it's complete. It is not necessary to run this query if you only upload data without ever deleting it for one single partition.

If you run a busy cluster, running it daily at a scheduled, off-peak time might produce the best results.

Monitoring clusters with QueueMetrics

QueueMetrics is able to monitor clusters of Asterisk servers, in order to monitor large call centres that are spread over a number of physical machines. This setting is often used for large deployments, as it leads to a number of advantages:

- The overall call center is safer, as the failure of one single Asterisk box leads to a down of only part of the call center and not its entirety
- The call center can easily grow to hundreds of seats simply by adding more Asterisk servers, without special optimizations or weird configurations
- There is less risk of a deadlock on one single Asterisk instance, as the load on each box is kept low enough not to be a problem

In order to implement this, QueueMetrics has been extended to support the notion of *cluster*, that is a set of Asterisk servers working together as if they were one single box. The cluster can be set up as is better fit, for example:

- Different queues for each Asterisk box, or
- The same queues on more than one Asterisk box
- Some boxes are used for inbound and some for outbound

When QueueMetrics runs in cluster mode, the whole call center is monitored as if it were a big single Asterisk box, and the basic unit for reporting remains the set of selected queues. QueueMetrics will internally query the different servers or *queue_log* files as needed, and will automatically dispatch events to the correct Asterisk box.

Setting up a cluster

To set up a cluster, you should define the following configuration variables in `configuration.properties`:

```
cluster.servers=aleph|trix
```

This statement tells QM that the current cluster has two members, that are called “aleph” and “trix”. We suggest using a short name for each server, as it will appear in many different screenshots. One option would be using the capital letters, like “A”, “B”, “C” etc for different members of the cluster.

For each server (in our case “aleph”, but we’ll have to repeat it for all members of the cluster), we will define the following properties:

```
cluster.aleph.manager=tcp:user:pass@10.10.3.5
```

This tells QueueMetrics that the manager interface for aleph can be found at 10.10.3.5, logging in as “user” with password “pass”. The manager interface is needed to run Live monitoring and can be used to run commands to Asterisk (like logging agents on and off, starting chanspy sessions, etc).

```
cluster.aleph.queuelog=sql:P001
```

This tells QM that the queue_log file (or its contents) can be downloaded from partition P001 of the QM database. You must use MySQL storage in order for clustering to work at all.

```
cluster.aleph.monitored_calls=/share /aleph/calls/
```

This tells QM where to look for recorded calls on each Asterisk server. This is used by QueueMetrics in order to click-and-listen to recorded calls. A NFS or SMB share is usually a good starting point. As an alternative, you can enter the URL of an XML-RPC server that will return information about the recorded call (for more information on this topic, see *Enabling XML-RPC call listening* and streaming on page 120).

```
cluster.aleph.callfilesdir=/share/aleph/callfiles/
```

If you do not want to connect to your Asterisk servers using the manager interface, you still need a way to send it commands (e.g. to start a chanspy session). In order to do this, you should give QM a directory to write callfiles to. If you use the manager interface, leave this entry blank. (We strongly suggest doing so and using the manager interface instead).

```
cluster.aleph.audioRpcServer=http://myserver/xmlRpcServer
```

If you use an XML-RPC “broker” in order to used live calls listening using a third-party software like Orecx, you should enter its URL here. This must be activated at once for all servers by not leaving blank in the property default.audioRpcServer. In all other cases, just leave this property blank. (for more information on this topic, see *Enabling XML-RPC call listening* and streaming on page 120).

```
cluster.aleph.agentSecurityKey=AAA
```

When using the agent’s page in cluster mode, you must make sure that each agent “points” to the correct server, as this server will be used for both pulling agent’s data and sending logon/logoff commands. This is obtained on the agent’s page through a pull-down menu where the agent must select the correct server he’s logged on to. In order to avoid mistakes, it is possible to protect a server by adding a security key, so

that only agents having that security key will see that server. If an agent has only one possible server, that server will be automatically selected. In practice, this means that you could create two agent classes, we call them AGENT_A and AGENT_B. They have the same keys, but in class AGENT_A there is the key SERVER_A, and in the other SERVER_B. We protect each server entry with SERVER_A for the first and SERVER_B for the other. Then we assign users to classes AGENT_A (for agents working on the first server) and AGENT_B (for agents working on the second server). If you want agents to manually switch servers, or your cluster is made up of only one machine, leave this blank.

Setting up the members of the cluster

On each box that is a member of the cluster, you should set up the following items:

- *Call recording*: if calls are recorded to be played back through QueueMetrics, you should store them all in a directory that is accessible through the QueueMetrics server, or set up an external XML-RPC call broker.
- *Commands*: if commands are to be sent to each Asterisk box, you should set up the *[queuemetrics]* context in the dial plan, and make sure the manager interface is set up or the */vars/spool/asterisk/callfiles* directory is shared and accessible to the QueueMetrics server. A sample *[queuemetrics]* context can be found under *WEB-INF/mysql-utils* in the directory *extensions-examples*.
- *Logs*: you should use *qloaderd* to upload data to a partition on the main QueueMetrics database. Make sure that each server uploads data to a different partition in the same database.
- *Clock*: make sure the clocks on all members of the cluster is synchronized, and the same goes for the clock used on the QueueMetrics box and on the MySQL database. An utility that sync your machine's clock to an external timing source like *ntpdate* will take care of this problem if run periodically through *cron*.

Setting up QueueMetrics to access the cluster

First thing, you should make sure that you have a clustered license for QueueMetrics and that your license is big enough in terms of agents to support all agents that are present in the call-center. Older licenses are valid for one Asterisk server only and QueueMetrics will complain they are not correct. The first releases of QueueMetrics 1.4 will anyway allow accessing a cluster up to a specified future date (likely October 2007).

To report on all members of a cluster, you should set the property:

```
default.queue_log_file=cluster:*
```

This means that all boxes defined as members of this cluster will be used as a data source.

To report on a subset of the members of the cluster, you will use a syntax like:

```
default.queue_log_file=cluster:A|B|C
```

This way you will be reporting on boxes A, B and C.

If you want to report only on a single box, the syntax:

```
default.queue_log_file=cluster:C
```

Will be appropriate.

You can then change this property on-the-fly by going to the “Custom reports” page and editing as needed under the “File” text box.

If you have agents using QueueMetrics’s Agent’s pages, you should set them up so that each agent points to its correct server.

Using the Agent’s page with a clustered environment

The agent’s page on QueueMetrics acts as a kind of portal for an agent; she can use it to log on, log off, go to pause, enter pause codes, launch external apps linked to a call (e.g. CRM apps) and enter call codes (see *The real-time agent page* on page 61).

As the number of agents can be very high if compared to the number of supervisors who run reports or monitor the call center, QM uses a “minimal impact” policy: the page must be refreshed manually by each agent in order to avoid hammering the server with repeated page hits and the analysis run is a stripped-down, low-fat version of the full analysis QueueMetrics is able to perform. When coming to clusters, this means that to avoid useless load, calls for an agent will be searched only on the server the agent is working on and not on the entire cluster.

Also, we have the problem of defining where an agent is supposed to work: as QM can issue commands to Asterisk on behalf of an agent, it needs to know to which Asterisk server those commands must go. This is obtained by using the Server selection that will appear on the agent’s page if QueueMetrics is running in clustered mode. If more than one server is selectable, the combo box will let the agent switch server as she best

sees fit (if only one server is selectable, QueueMetrics will use that server immediately and will make the combo locked).

As a QM installer, you can control which servers are selectable to which agents by setting the properties *cluster.---.agentSecurityKey* correctly for each Asterisk server in the cluster.

Editing QueueMetrics settings

System configuration must be done by the system administrator. Most configuration may be done straight from QM itself, while system wide preferences must be set editing a text file on the installation server.

To log on as an administrator, you can use the supplied account **demoadmin**, password **demo**, that will bring you to a home page like the following one:

The screenshot shows the QueueMetrics Home Page. At the top left is a logo with three colored circles (red, blue, green) and the text "Your Logo". At the top right, it says "Demo Admin | Administrator" with icons for help, chat, and print. Below this is the "QueueMetrics call center monitor" logo. A "Home" button is visible in the top left corner. The main heading is "QueueMetrics Home Page". Below this is a "Queue:" dropdown menu set to "All". A list of menu items follows, each with a red arrow icon: "Quick activity reports" (with sub-links: Today | Yesterday | The day before yesterday | Last day | Last 7 days | Last 30 days | Last 90 days), "Real-time report" (with sub-links: Start realtime monitoring | Start wallboard), "Custom report" (with sub-link: Run custom report), "Agent report" (with a "Filtered for agent:" dropdown set to "-" and sub-links: Today | Last 7 days | Last 30 days | Last 90 days), "Edit QueueMetrics settings" (with sub-links: Administer users | Edit queues | Edit agents | Edit locations | Edit call outcomes | Edit pause codes | Setup wizard - Load data from Asterisk | Mysql storage information), and "Show licence information". At the bottom center is the "Loway research Loway Research" logo.

Configuring users

Users and classes can be added, modified and deleted right from QM.

A list of users is presented and you can filter it by class or user name.

Loway QueueMetrics version 0.9.1 - Opera

File Edit View Bookmarks Feeds Mail Tools Help

New page Received Loway QueueMetrics ver...

OM .1:8081/queuemetrics/usradmin/usradm Google search 100%

Your Logo Demo Admin | Administrator Log off Print

QueueMetrics
call center monitor

Home Cfg Users Cfg Queues Cfg Agents Outbound

Users

All

Id	Login	Real name	Enabled	Class	User keys	Detail
26	Agent/101	Agent Pluto (101)	Yes	AGENTS		...
24	Agent/102	Agent Pippo (102)	Yes	AGENTS		...
1	demoadmin	Demo Admin	Yes	ADMIN		...
25	demouser	Demo User	Yes	MANAGERS		...

Loway
research
Loway Research

For each user, the login and full name are shown, together with the current class and any additional user keys. A user must be enabled in order to log on, so if you want to prevent somebody from logging on without deleting its user information, you can simply disable it.

The “Add new user” button lets you add new users while the “User classes” button leads to the class editor.

Loway QueueMetrics version 0.9.1 - Opera

File Edit View Bookmarks Feeds Mail Tools Help

New page Received Loway QueueMetrics ver...

1.0.1:8081/queuemetrics/usradmin/usradr Google search 100%

Your Logo Demo Admin | Administrator Log off Print

QueueMetrics call center monitor

Home Cfg Users Cfg Queues Cfg Agents Outbound

User

User id	26
Login	Agent/101
Password	999
Real name	Agent Pluto (101)
Enabled	Yes
E-mail	
Masterkey	No
Class	AGENTS
User keys	
Number of logons	97
Last logon	2005-09-07 15:43:40
Comment	
Token	
Creation	demoadmin [1], 23/03/2005, 1
Update	demoadmin [1], 06/09/2005, 1

Update user Delete user Back to users

Loway

When you add or edit a user, you are presented with a list of fields to enter:

- *Userid* is a technical reference used internally. Read only.
- *Login* is the login string.
- *Password* is the password, shown in clear text.
- *Real name* is the name shown in the top part of the screen
- *Enabled* lets you temporarily disable somebody from using QM.
- *E-mail* is the user's e-mail address (Optional).
- *Masterkey*: if set to Yes, all security key checks are bypassed. **DO NOT SET UNLESS YOU KNOW WHAT YOU ARE DOING!**
- *Class* is the current user class

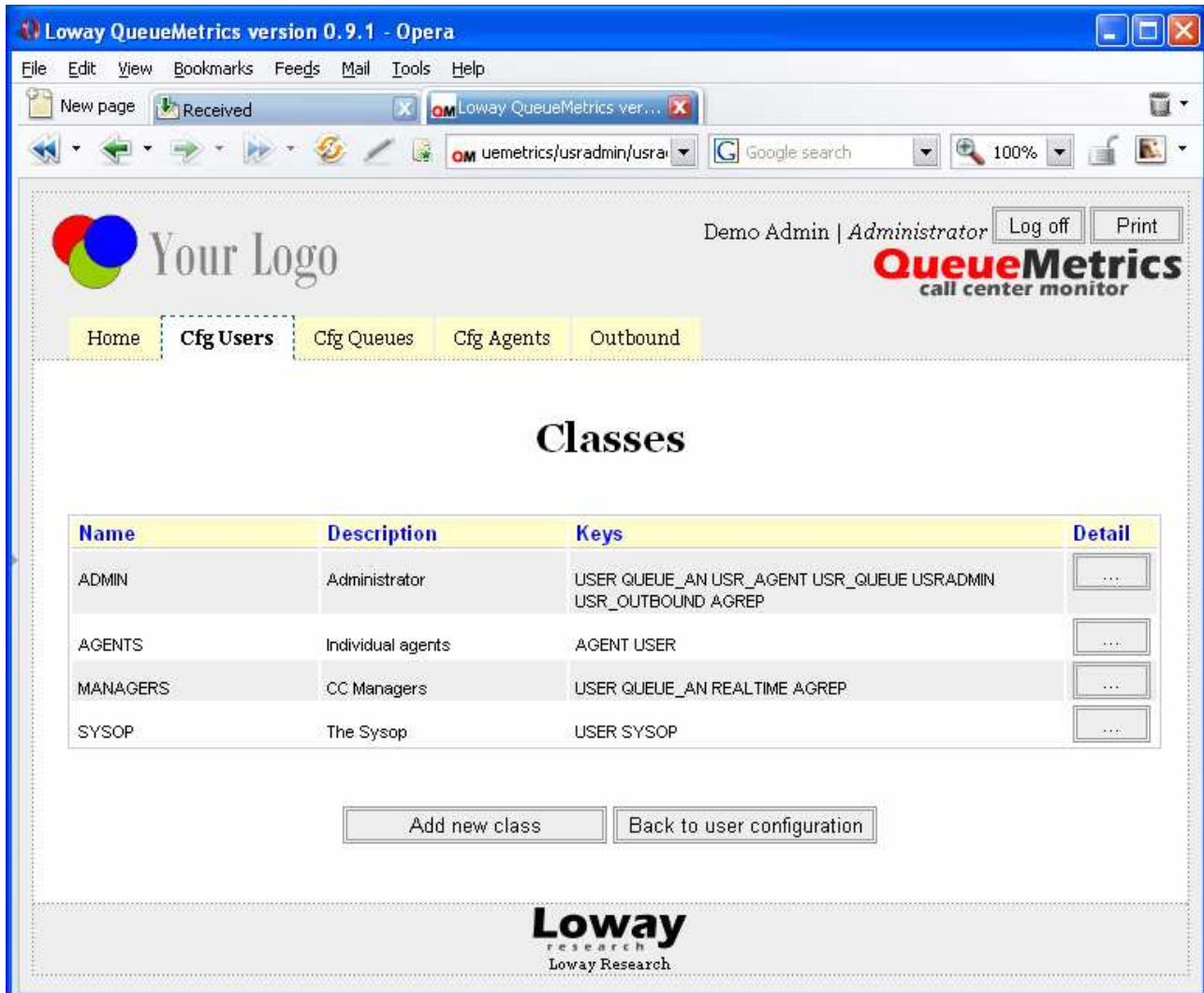
- *User keys* are additional keys the user holds. Separate each key with a space. If a key is preceded by the minus sign, it means it's revoked even if the class grants it.
- *Number of logons* tells how many times the user logged on in QM. Read only.
- *Comment* is an optional free comment.
- *Token* has no current use. Read only.
- *Creation* and *Update*: the user and date/time when the record was first created and then last updated. Read only.

IMPORTANT:

When you first log on to QueueMetrics, you must change the passwords to all default users. Failure to do so represents an important security breach!

Editing user classes

User classes can be configured freely; you can create individual key rings with special privileges to best suit your needs.

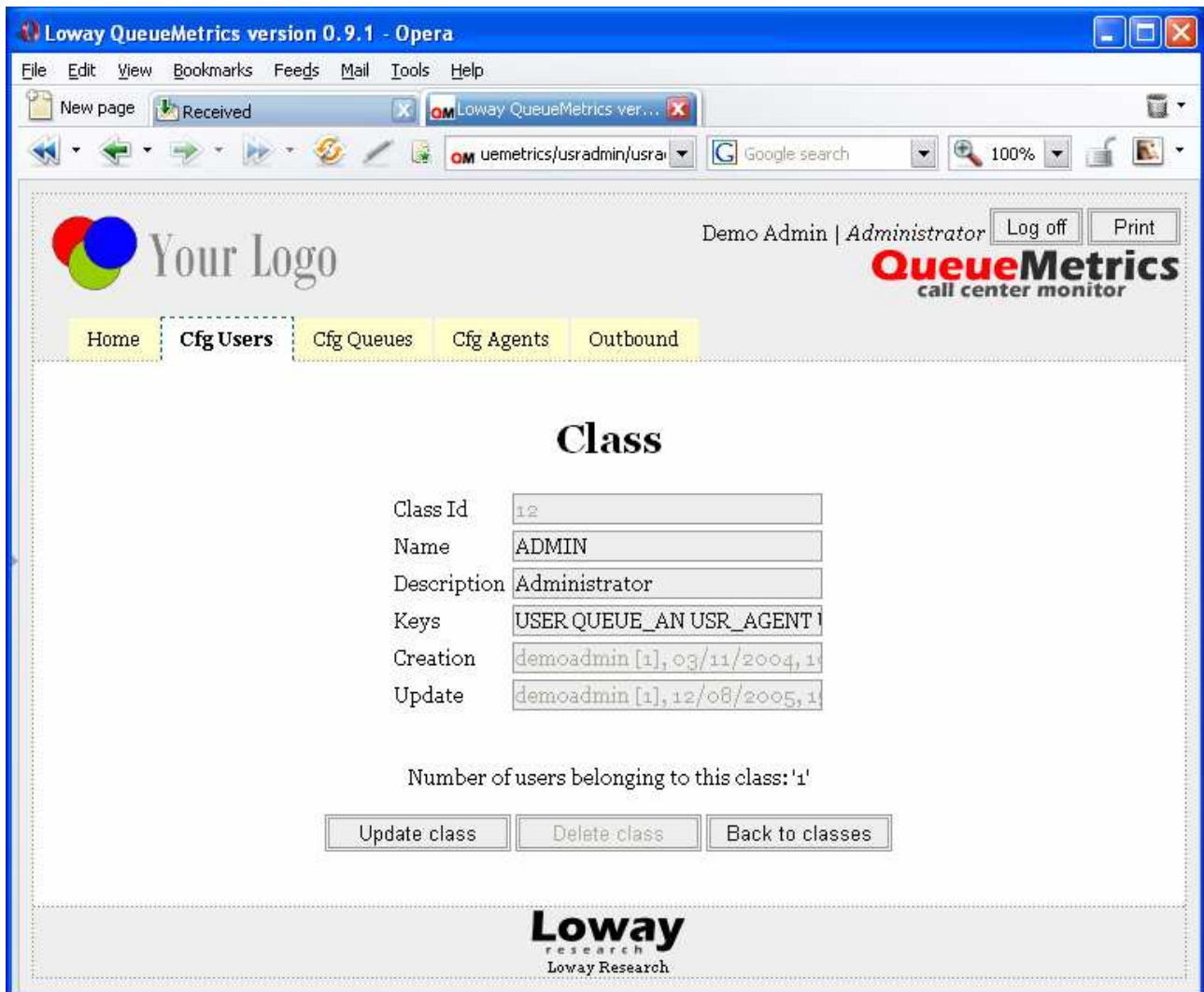


The screenshot shows the 'Classes' configuration page in the QueueMetrics web interface. The page has a navigation bar with 'Home', 'Cfg Users', 'Cfg Queues', 'Cfg Agents', and 'Outbound'. The 'Cfg Users' tab is active. The main content area is titled 'Classes' and contains a table with the following data:

Name	Description	Keys	Detail
ADMIN	Administrator	USER QUEUE_AN USR_AGENT USR_QUEUE USRADMIN USR_OUTBOUND AGREP	...
AGENTS	Individual agents	AGENT USER	...
MANAGERS	CC Managers	USER QUEUE_AN REALTIME AGREP	...
SYSOP	The Sysop	USER SYSOP	...

Below the table are two buttons: 'Add new class' and 'Back to user configuration'. The footer of the page displays the 'Loway research' logo and 'Loway Research' text.

Each class has a set of keys that can be freely edited in much the same way as users.



No class can be deleted as long as there is at least one user that is member of it.

The default classes should be enough to get most systems started:

- ADMIN is for the system administrator only, and lets you do nearly everything, including system configuration;
- MANAGERS is for most QM users, the ones that have to run the reports and monitor real-time activity;
- AGENTS is for individual agents logging on to their web page.
- VISITORS is for visitors accessing the simplified real-time page
- ROBOTS is for automated data download

Configuring queues

A list of queues must be set before accessing QM. Each queue can be made visible to only a specific set of users by adding a key – this can be useful if, for example, each

queue has a manager viewing data for it, while only a CC manager sees data for all queues in the center.

[Home](#)
[Cfg Users](#)
[Cfg Queues](#)
[Cfg Agents](#)
[Cfg Locations](#)
[Cfg Outcomes](#)
[Cfg Pauses](#)

Queues Configuration

Alias	Queue(s)	Wrap-up	Ann.	Key			
00 All	q1, q2, queue-dps, queue-test	0 s.	0 s.				
Q DPS	queue-dps	0 s.	0 s.				
Q Test	queue-test	0 s.	0 s.				

Click on a column name to sort data.
 : inbound queue - : outbound queue

Edit

Alias:
Queue(s):

Wrap-up time (sec.):

Announcement (sec.):

Visibility key:

Call flow:

Attention levels

Yellow alarm

Red alarm

Number of calls in queue:

Number of agents on call:

Number of agents waiting:

Number of agents paused:

Call wait duration:

Call talking duration:

Update now

For each queue you have to define:

- An *Alias*, that is the name users will see in the queues combo box on the Home Page;
- A set of *Queues*, that can be the name of an Asterisk queue as seen from the Queue() command or a set of names separated by the pipe symbol, as in *queue1|queue2|queue3*. This lets you aggregate queues freely.
- An optional *Wrap-up time*, i.e. how many seconds an agent stays idle after hanging up;
- An optional *Announcement* duration, that lets you deduce the duration of the queue announcement that is played to the agent from the actual metrics;

- An optional *Visibility key*, that makes the queue visible only to users holding that key.
- The *Call flow direction*, i.e. whether the queue is an inbound (classical) queue or an outbound queue (made to track individuals agents calling out or the activity of a full-fledged predictive dialler).
- A number of *Attention levels*, see below.

By clicking on the Agents icon, you can define the position of each agent as a member of the service groups for that key. An agent cannot be a member of more than one group per each queue s/he is a member of.

It is of course perfectly legal for an agent defined not to be used in a specific queue.

Loway QueueMetrics version 1.2.0-b1 - Opera

File Edit View Bookmarks Feeds Mail Chat Tools Help

Your Logo Demo Admin | Administrators Log off Print

QueueMetrics call center monitor

Home Cfg Users **Cfg Queues** Cfg Agents

Agents available for queue -- ooAALL

MAIN Front line	WRAP Backup of Main	SPILL Last resort
<input type="checkbox"/> Anna (304)	<input type="checkbox"/> Anna (304)	<input type="checkbox"/> Anna (304)
<input checked="" type="checkbox"/> Caio (104)	<input type="checkbox"/> Caio (104)	<input type="checkbox"/> Caio (104)
<input type="checkbox"/> Giusy (305)	<input type="checkbox"/> Giusy (305)	<input type="checkbox"/> Giusy (305)
<input type="checkbox"/> Laura (303)	<input type="checkbox"/> Laura (303)	<input type="checkbox"/> Laura (303)
<input type="checkbox"/> Mirko (306)	<input type="checkbox"/> Mirko (306)	<input type="checkbox"/> Mirko (306)
<input type="checkbox"/> Paola (302)	<input type="checkbox"/> Paola (302)	<input type="checkbox"/> Paola (302)
<input type="checkbox"/> Pippo (101)	<input checked="" type="checkbox"/> Pippo (101)	<input type="checkbox"/> Pippo (101)
<input type="checkbox"/> Pluto (102)	<input type="checkbox"/> Pluto (102)	<input type="checkbox"/> Pluto (102)
<input type="checkbox"/> Sara (301)	<input type="checkbox"/> Sara (301)	<input type="checkbox"/> Sara (301)
<input type="checkbox"/> SIP Phone (899)	<input type="checkbox"/> SIP Phone (899)	<input type="checkbox"/> SIP Phone (899)
<input type="checkbox"/> Tizio (103)	<input type="checkbox"/> Tizio (103)	<input checked="" type="checkbox"/> Tizio (103)

Update now Back to queues

Setting attention levels (Red and yellow alarms)

It is possible – but not mandatory – to define all or some attention levels for the given queue. To do so, you have to fill in each queue attention levels parameter with an

expression that will be matched to the current property's value in order to trigger a defined alarm.

QueueMetrics does currently allow to set two possible alarm thresholds; that is a "yellow" and a "red" alarm. You can define one or both of these properties, according to your preferences. Those values are used currently only to trigger alarms in the real-time panel.

For example, imagine we want to set a yellow alarm on the queue wait for each call; we want cells to turn yellow if the wait time exceeds 30 seconds, and to turn red if it is over one minute. To do so, we enter "> 30" in the yellow alarm box near to "Call wait duration", and "> 60" in the red alarm box on the same line.

In the case where both yellow and red conditions match, the red alarm prevails.

Currently, the following functions can be used to match a value "=", ">", "<", "!=" (different).

The possible alarms are the following:

- *Number of calls in queue*: how many calls are present in the queue.
- *Number of agents on call*: how many agents are on call
- *Number of agents waiting*: how many agents are idle
- *Number of agents paused*: how many agents are on pause
- *Call wait duration*: how much a call is waiting before being answered
- *Call talking duration*: the duration of the agent's conversation

Configuring agents

Agents can be configured so that:

- They're decoded to their own name when they are found in reports
- They can be set as members of service levels for queues.
- They can be assigned an optional Location, that can also be used as a filter condition.

Home
Cfg Users
Cfg Queues
Cfg Agents
Cfg Locations
Cfg Outcomes
Cfg Pauses

Agents Configuration

Agent code	Agent description	Location	Term.	Mon.	Superv.		
Agent/101	John Doe (101)	Main			demosupervisor		
Agent/102	Mike Boo (102)	Other	12		-		

Click on a column name to sort data.

Add new agent

Agent code:

Agent description:

Agent location:

VNC monitoring URL:

Current terminal:

Supervisor:

Test it

For each agent in use, enter:

- *Agent code* as the Asterisk agent code, e.g. Agent/101;
- *Agent description* as the agent's own name.
- *Agent location* can be selected from a drop-down list of defined locations. Leave blank if not needed.
- *VNC Monitoring*: the URL that will launch the VNC monitoring app for the given agent
- *Current terminal*: the current terminal for the given agent. If this field is left blank, unattended audio monitoring will not work. If you are using regular Asterisk agents, just enter “-“ as the current terminal to make audio monitoring work.
- *Supervisor*: the supervisor for this agent. This can be selected between all users holding the key SUPERVISOR.

If you want an agent to log on to their own page, you also have to create a user with the same name.

Configuring locations

The following configuration transaction lets you define locations for your agents. To access this page, a user must be holding the USR_LOCATION key.

Home
Cfg Users
Cfg Queues
Cfg Agents
Cfg Locations
Cfg Outcomes
Cfg Pauses

Locations Configuration

Name	Description	Key	N. of agents		
Main	Main Location		1		
Other	Secondary location		1		

Click on a column name to sort data.

Edit location

Name:
Description:
Security key:

Add now

Each location has a short name, a longer description, and a visibility key, so that only users holding that key may select that location as a source for reports.

A location cannot be deleted if at least one agent is defined for that location.

Configuring call outcomes

We define a *call outcome* as a flag to be added to a call, either when the call is ongoing or when the call has just finished, that will signal the result of the call from a business point of view. Such a flag is optional for QueueMetrics and can be added to both incoming and outgoing traffic.

The call outcome will be defined by a numeric sequence that the agent will either key in on their telephone terminal or report through QueueMetrics itself through the Agent's page. QueueMetrics will not consider how the sequence is entered, as long as it's present in the *queue_log* data it runs on. Such records can be generated, for example, by an outbound dialler that is able to pre-screen answered traffic.

To minimize internal searching costs, the call activity must be entered either while the call is in progress or within one hour of its completion. If more than one call activity code is entered, the latest takes precedence over the previous ones.

Home
Cfg Users
Cfg Queues
Cfg Agents
Cfg Locations
Cfg Outcomes
Cfg Pauses

Call Outcomes Configuration

Add new entry

Code	Description	Contact?	Sale?	Key		
20	Sold	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
21	Not interested	<input checked="" type="checkbox"/>	<input type="checkbox"/>			
a	Answering Machine	<input type="checkbox"/>	<input type="checkbox"/>			
b	Busy	<input type="checkbox"/>	<input type="checkbox"/>			
dc	Disconnected	<input type="checkbox"/>	<input type="checkbox"/>			
dnc	Do not call anymore	<input checked="" type="checkbox"/>	<input type="checkbox"/>			
n	Not Available	<input type="checkbox"/>	<input type="checkbox"/>			
ni	Not Interested	<input checked="" type="checkbox"/>	<input type="checkbox"/>			
nq	Not Qualified	<input checked="" type="checkbox"/>	<input type="checkbox"/>			
sale	Sale	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
xfer	Transfer to IVR	<input type="checkbox"/>	<input type="checkbox"/>			

Click on a column name to sort data.

Add new entry

As you can see, each outcome can set two flags: a “This call qualifies as contact?” flag and a “This call qualifies as sale?” flag. This will be used in order to produce statistics on traffic. If a call code is found but not defined through the configuration screen, QM will report on it and treat it as a “No contact” and “No sale” call.

Home
Cfg Users
Cfg Queues
Cfg Agents
Cfg Locations
Cfg Outcomes
Cfg Pauses

Edit call outcome

Status code:
Description:
Security key:
Counts as Contact?
Counts as Sale?
Created by: demoadmin, 22/06/2007, 10:27
Last updated: demoadmin, 22/06/2007, 10:27

Save
Back
New

The editor page lets you set:

- A numeric⁴ code for that outcome. The system will check that it will not be duplicated on the list
- A text label for the outcome (e.g. “Contact”)
- A flag telling the system whether that outcome counts as a “Contact”
- A flag telling the system whether that outcome counts as a “Sale”
- An optional security key for that outcome. This will be used only when displaying outcome choices for a given call in the Agent’s page. The reporting engine will report on all outcomes present in its analysis.

Configuring pause codes

The agent’s time is defined in QueueMetrics as made up of different activities. The main activity for an agent will be “Available time”, i.e. the time when an agent is ready taking or placing calls. When an agent pauses out of “Available time”, they may want to flag the reason for the pausing, e.g. doing backend activities, lunch, etc. This way you can track agent activities punctually. If they don’t flag a pause, it will be computed as simply “Pause” time.

Each pause code is written on the queue log while the pause is in progress, i.e. after the agent goes on pause and before the agent stops that pause. The pause code will usually be defined by a numeric sequence that the agent will either key in on their telephone terminal or report through QueueMetrics itself through the Agent’s page. QueueMetrics will not consider how the sequence is entered, as long as it’s present in the *queue_log* data it runs on.

⁴ The code must be numeric so it may optionally be keyed-in using the rep’s terminal.

[Home](#)
[Cfg Users](#)
[Cfg Queues](#)
[Cfg Agents](#)
[Cfg Locations](#)
[Cfg Outcomes](#)
[Cfg Pauses](#)

Pause Codes Configuration

Add new entry

Code	Description	Billable?	Key		
10	Lunch				
11	Hourly break				
12	Email				
13	Backoffice				
30	Lunch				
31	Back Office				

Click on a column name to sort data.

Add new entry

For each pause code, it is possible to tell QueueMetrics whether that time is billable (from a business perspective, e.g. time an agent is disconnected from the ACD but doing back-office activities) versus non-billable activities (e.g. pausing for lunch, taking mandatory breaks, and so on).

All activities are optional and may be added or deleted at will. The following fields apply:

- A numeric⁵ code for that activity. The system will check that it will not be duplicated on the list
- A text label for the activity (e.g. "Lunch")
- A flag telling the system whether that activity is: Billable, Not Billable or Other activity. Other activities are considered non-telephone related activities and are by definition non billable. With Other activity you may add activities like training, etc.
- An optional security key for that activity. This will be used only when displaying activity choices in the Agent's page. The reporting engine will report on all activities present in its analysis.

⁵ The code must be numeric so it may optionally be keyed-in using the rep's terminal.

Home	Cfg Users	Cfg Queues	Cfg Agents	Cfg Locations	Cfg Outcomes	Cfg Pauses
------	-----------	------------	------------	---------------	--------------	-------------------

Edit pause codes

Status code:	<input type="text" value="12"/>
Description:	<input type="text" value="Email"/>
Security key:	<input type="text"/>
Counts as Billable activity?	<input type="text" value="Yes"/>
Created by:	demoadmin, 18/06/2007, 22:25
Last updated:	

Automatic QueueMetrics configuration

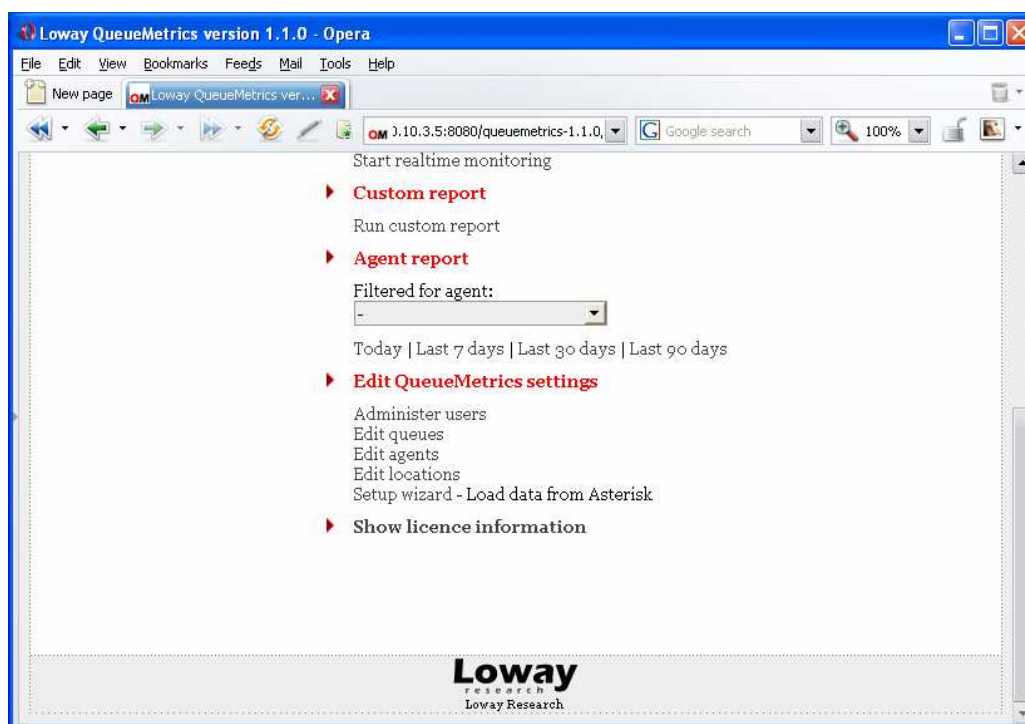
In order to save time and make sure that QM is always up-to-date with the underlying Asterisk configuration, it is possible to run a wizard that will load the following data straight from Asterisk configuration files:

- Which queues are in use, and their configuration
- Which agents are being referenced, their name and how they belong to the various queues

It is also possible to automatically create users out of the defined agents, so that they can log-on to QueueMetrics with the very same password they use to log-on to Asterisk.

In order for the wizard to be run, the user must hold the grants to administer users, edit queues and edit agents too.

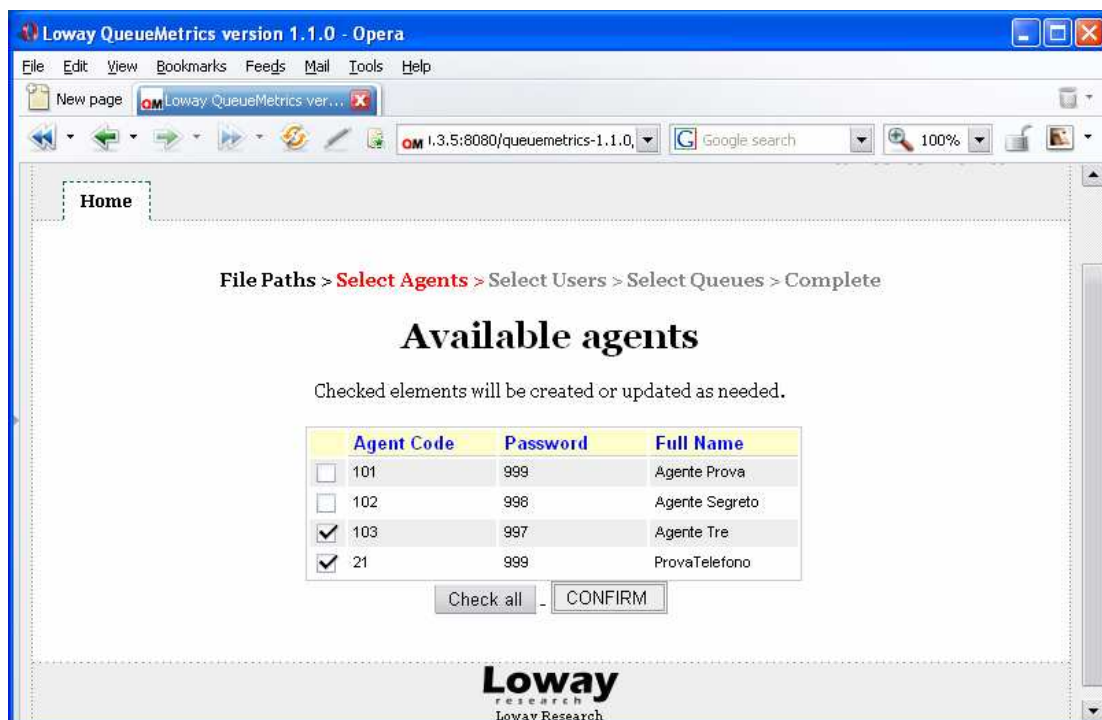
If the user holds the required keys, the label “Setup wizard” will be shown on the front page:



By clicking on it, the administrator will be lead to a page where he can select the local paths for the agents.conf and queues.conf files. If those files are not present or not readable, a red “Error” statement will appear next to the file names.



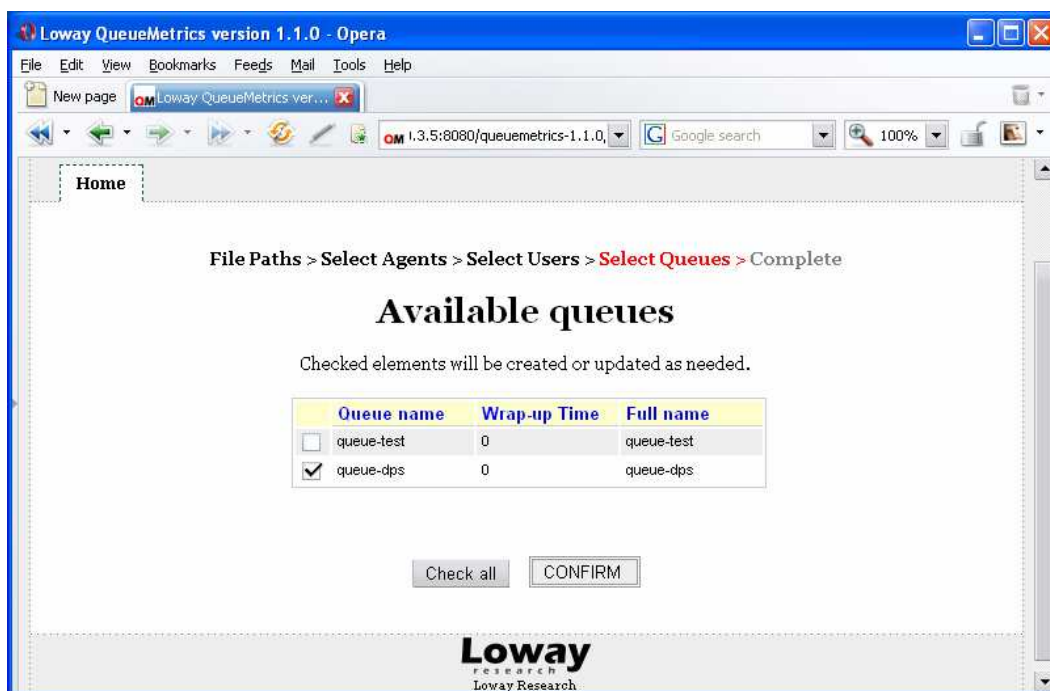
If all seems to be correct, press on the “Confirm” button. Press “Retry” after modifying a path to check if it is readable by QueueMetrics.



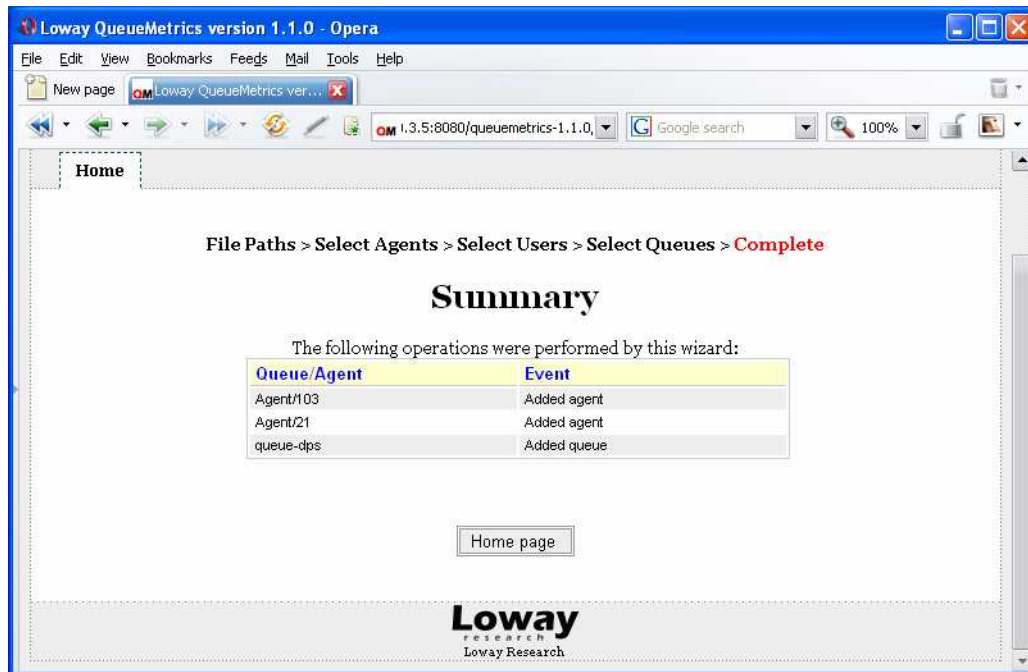
Asterisk will scan the available agents and present you a list of agents to be created or updated. By default, this wizard will try not to modify an agent or a queue that is already present in QM, that is the found data will be shown but unchecked. Check on the items to include/exclude them as needed.



If the corresponding QM users for selected agents only are not present, they are created automatically by this mask.



The queues will be created or updated as needed; existing queues will not usually be overwritten without explicit user permission.



This last mask will display a summary of the updates that have been performed by this wizard.

Configuring system preferences

System preferences can be edited by editing a text file called *configuration.properties* located in the WEB-INF directory of the QM webapp.

This lets you set system-wide parameters and edit the logo and splash screen text for QM.

Property name	Description
default.queue	Internal ID (ex. 7, 49....) of the default queue, leave blank for no default queue.
Default.queue_log_file	Default queue log file.
Realtime.max_bytes_agent	When the real-time page for an agent is computed, the queue_log is NOT read in its entirety but only the last 'n' bytes. In database storage mode, the number of seconds, starting from now and counting backwards, that will be queried for agent events.
Default.monitored_calls	The top level directory where monitored calls are held. All its subdirectories are explored recursively. Do NOT forget to add an ending slash.
Default.areacode_digits	How many digits to consider as a default area code
default.start_hour	Preset start and end hours and number of days for the custom report.
Default.end_hour	
default.days	
liveclock.enable	If live clock is enabled, the system clock is synchronized with Asterisk server system clock.
Default.max_realtime_age	How old a call can be included in real-time report
default.permanentCallbackAgents	If call-back agents should be considered still logged on after a system reload; the current version of Asterisk will do this automatically.

	Default: true
default.considerIncompleteEntities	If incomplete entities (calls or agent sessions that are in progress at the moment that are in progress at the moment the analysis is being run) should be counted in the reports or not. Default: true
default.rewriteLocalChannels	Rewrites queue_log entries in the form <i>Local / xxx@context</i> to <i>Agent / xxx</i> to make deployment simpler for AAH users. Default: false.
Default.language	The default language. Must be one of the installed language packs. Default: en
Default.country	The default country for the Locale. Must be one of the installed language packs. Default: US
sla.max_monitored_delay	The max delay and interval that will be shown in the TOS graph
sla.interval	
layout.logo	Your company logo (full or relative path) – shall be resized to be an image 200 x 72. The variable \$WEBAPP refers to the local webapp, as an alternative use the full http://... URL.
Layout.splash	HTML string displayed on the login page.
sqlPreset.i.table	Sets the table name for preset ‘i’
sqlPreset.i.f_time_id	Defines the field used by the table in MySQL storage. See page 75 and following for complete information.
sqlPreset.i.f_call_id	
sqlPreset.i.f_queue	
sqlPreset.i.f_agent	
sqlPreset.i.f_verb	
sqlPreset.i.f_partition	
sqlPreset.i.f_data1	
sqlPreset.i.f_data2	

sqlPreset.i.f_data3	
sqlPreset.i.f_data4	
sqlPreset.i.f_incr	
Realtime.calls_invisible	Is the calls panel in the realtime page invisible by default? 0 false, 1 true
realtime.agents_invisible	Is the agents panel in the realtime page invisible by default? 0 false, 1 true
realtime.members_only	Are not the only agents to be shown on the realtime page those who are “known” for the queue? 0 false, 1 true
realtime.refresh_time	In how many seconds is the realtime page to refresh?
realtime.use_sql_now	0: analyze all available data; 1: analyze all data which timestamp is lower than the current NOW() function. Do not change.
callfile.dir	<p>The call-file directory Asterisk uses to generate calls based on .call files. Must be writable by the Java process. Default <i>/var/spool/asterisk/outgoing</i></p> <p>As an alternative, you may enter a Manager interface URI here, in the format <i>tcp:user:password@server</i></p> <p>If you do, QM will not generate call-files but will use the manger interface to generate calls.</p>
callfile.monitoring.enabled	If unattended audio monitoring is enabled on this system. Default true.
callfile.monitoring.channel	<p>The channel, and extension@context that will be called to implement the unattended audio monitoring functionality. Do not forget the trailing /n in the channel.</p> <p>A number of variables act as placeholders to be substituted by the actual data Asterisk is using:</p> <ul style="list-style-type: none"> • \$AG : the current agent
callfile.monitoring.extension	
callfile.monitoring.context	

- \$AE : the agent's extension
- \$EM : the monitoring extension

See *Enabling Unattended Call Monitoring* on page 119 for further information.

callfile.agentpause.enabled

callfile.agentpause.channel

callfile.agentpause.extension

callfile.agentpause.context

This function is used to start a pause from the Agent's page and to set its Pause Code - see the sample dial plan provided.

callfile.agentunpause.enabled

callfile.agentunpause.channel

callfile.agentunpause.extension

callfile.agentunpause.context

This function is used to end a pause from the Agent's page - see the sample dial plan provided.

callfile.agentlogin.enabled

callfile.agentlogin.channel

callfile.agentlogin.extension

callfile.agentlogin.context

This function is used to log in an agent from the Agent's page - see the sample dial plan provided.

callfile.agentlogoff.enabled

callfile.agentlogoff.channel

callfile.agentlogoff.extension

callfile.agentlogoff.context

This function is used to log off an agent from the Agent's page - see the sample dial plan provided.

callfile.calloutcome.enabled

callfile.calloutcome.channel

callfile.calloutcome.extension

callfile.calloutcome.context

This function is used to set the call outcome code from the Agent's page - see the sample dial plan provided.

callfile.agentdial.enabled

This function is not implemented yet.

callfile.agentdial.channel

This function is not implemented yet.

callfile.agentdial.extension

callfile.agentdial.context

realtime.startHour

The starting hour of the day, in order to compute realtime report. It can be either a fixed hour (e.g. 3: from 3:00 AM) or a sliding window if prefixes with S (e.g. s3: the last three hours). Default value is 0 (from midnight). A useful value is also -24 (yesterday's midnight).

realtime.all_subqueues

Enable default showing of all subqueues if set to 1

default.joinMultiStintCalls

If true, multi-stint calls are joined by default

default.useEndingChannelName

If true, the last reference to an agent is used as its name (in case they are different)

default.showQueueComposition

If true, show the details of the queues composing the aggregate queue; if false, show only the aggregate queue's name

default.useXmlExcel

True: Generate the Excel file as an XML file (mandatory for UTF charsets); false: generate as an ISO-8859 CSV file

default.hourly_slot

How long in minutes is an hourly slot for hourly breakdown. Default 60 minutes (1hr). If set to e.g. 15, calls will be broken down by 15 minute intervals.

default.audioRpcServer

The URL of an external XML-RPC server for both listening of recorded calls and live call monitoring.

default.useRawAgentSessions

If true, show all agent sessions. If false, show only agent sessions with at least one call handled. Defaults to false.

default.lockedAgentPopupCode

If true, the agent cannot change their code in the login/logoff/pause popups. Defaults to false.

<code>realtime.waitAlarmOnLiveCalls</code>	Decide whether to check for alarms on the wait time of ongoing conversations.
<code>realtime.hideExportButtons</code>	If true, hide export buttons on the Real-time page. Defaults to false.
<code>sound.yellowAlarm</code>	Sound to be played if a yellow alarm is triggered. Can be both an absolute URL or a relative path
<code>sound.redAlarm</code>	Sound to be played if a red alarm is triggered. Can be both an absolute URL or a relative path
<code>cluster.servers</code>	A set of servers, which names must be used for subsequent properties
<code>cluster.servername.manager</code>	The manager API for this server, in the format <code>tcp:user:pass@server</code>
<code>cluster.servername.queuelog</code>	The queue log partition to use, in the format <code>sql:P001</code>
<code>cluster.servername.monitored_calls</code>	The directory where monitored calls for this server can be found. If it starts with “http”, an XML-RPC server to query this information
<code>cluster.servername.callfilesdir</code>	The directory in which callfiles must be generated for this sever. Usually leave blank.
<code>cluster.servername.audioRpcServer</code>	The URL of an XML-RPC server to be used for audio monitoring
<code>cluster.servername.agentSecurityKey</code>	The key with which this cluster entry must be protected on the Agent’s page

Configuring Asterisk for QueueMetrics

Though QueueMetrics is designed to analyze `queue_log` data provided by any Asterisk installation, the following guidelines help make the most out of it.

Configuring queues to report exit status

In the following example:

- all calls are monitored, i.e. saved to disk;
- if after 60 seconds on the queue the call is unanswered, the call is routed to voicemail and this event is reported correctly by QM;
- there are two levels of agents: agents 302 and 303 will answer the queue (level 1); only if none of them is available the call is routed to agent 301 (level 2). If nobody is available, the queue keeps trying until timeout is reached.
- Agents can transfer the call to other extensions by pressing the “#” key;
- Agents terminate the current call by pressing the “*” key.

Extensions.conf

```
[q-my-sample]
; ...queue description.....

exten => s,1,SetVar(MONITOR_FILENAME=/var/spool/asterisk/QSAMPLE-
${UNIQUEID})
exten => s,2,Queue(q-sample|nt||60)
exten => s,3,Playback(voicemail-invitation)
exten => s,4,VoiceMail,s2001
```

Queues.conf

```
[q-sample]
music = default
announce = q-sample-announce
strategy = roundrobin
timeout = 60
retry = 5
maxlen = 0
announce-frequency = 0
announce-holdtime = no
monitor-format = wav
monitor-join = yes
queue-youarenext = silence
queue-thankyou = q-sample-thankyou
member=>Agent/302,0
member=>Agent/303,0
member=>Agent/301,1
```

Make sure that you do not forget the explicit timeout when calling the Queue() command from *extensions.conf*, or queue timeouts will not be logged by Asterisk and

therefore not reported by QM. A patch that corrects this Asterisk behaviour can be found at <http://bugs.digium.com/view.php?id=5422>.

Configuring URLs to be launched by the agent real-time page

The URL should be embedded in the Queue() command as prescribed by Asterisk

```
Exten => s,7,  
Queue(myqueue|nt|http://mysite/app?uid=${UNIQUEID}&clid=${CALLERID}||60)
```

This command launches the queue „myqueue“ and launches the webapp located at <http://site/app> passing the following parameters:

1. *uid* is the Asterisk internal unique call id
2. *clid* is the Caller*ID for the current call

Listening to calls using QM

- *Make sure it is legal*

This is not strictly a QM issue, but before attempting to record all calls on a queue, you should consult a lawyer to make sure it is legal in your country. It would be probably fair enough to tell your operators their calls are being recorded and to add a voice message telling the customers their call will be recorded.

- *Tell Asterisk to record all calls*

To record all calls add something like this to *extensions.conf*:

```
exten => s,1,SetVar(MONITOR_FILENAME=/var/spool/asterisk/q/ QSAMPLE-  
${UNIQUEID})  
exten => s,2,Queue(q-sample|nt||60)
```

This way all sound files are stored under */var/spool/asterisk/q/* with the name of the queue (QSAMPLE) followed by the call id.

- *Tell QueueMetrics where to look for the calls*

You should set up the *WEB-INF/configuration.property* file in QM like this:

```
default.monitored_calls=/var/spool/asterisk/q/
```

When looking for the recording of a call, QM will explore all files contained in */var/spool/asterisk/q/* and any directories below for a file name containing the right call ID. It might find more than one file name and will display all of them. It is possible that sometimes Asterisk fails at mixing together the two files (Asterisk records separate files for the caller and the agent, and then tries to mix them together at the end of the call) so you will find two files named *-in* and *-out* instead.

- *Tell QueueMetrics you have the right to listen to the calls*

Any user willing to listen to calls must hold the key CALLMONITOR. This is to

make sure that only authorized personnel can listen to recorded calls. If you do not have this key, no sound files will be shown.

- *Make sure QueueMetrics has the right to read saved calls*
You should make sure that the process running QM (i.e. the servlet container, might be Tomcat, Jetty, or something else depending on your setup) has the rights to access the files where recorded calls are stored.
If using a separate web server, it should not be able to access those files directly, as QM will pipe out files only after enforcing security checks.
- *Debug tip: see which files QM sees*
There is a hidden transaction in QM made to debug call listening. To launch it, logon as an administrator and type the transaction "*qm_show_files.do*" in the URL bar instead of the page name.
You will be lead to a page showing the filenames QM can read from the hard disk, whether the current user has the CALLMONITOR key and the search path as defined in *default.monitored_calls*.

Using AddQueueMember for dynamic agents

AddQueueMember is a command that lets you add dynamic agents to a queue. Its main advantage is that you can add channels, i.e. terminals, so you'll have most of the advantages of agents without the performance and stability problems that the agents module may cost in very large systems.

Its disadvantage is that it does not log the agent login/logoff to the queue_log, and so programs that analyze the queue log data like QueueMetrics will not see agents logging on and off. This is a major organizational problem in a real-world call center, where tracking agent logons and logoffs is vital to the smooth running of the operations.

The answer is to add a fake queue_log data for each logon and logoff. For QM, it is important to avoid multiple logoff lines and to compute online permanence with logoffs.

To do the adding, you dial 422XX, where XX is your local extension; the same happens with 423XX to be logged off.

```
; Add Member - 422
exten => _422XX,1,Answer
exten => _422XX,2,AddQueueMember(my-queue,SIP/${EXTEN:3})
exten => _422XX,3,System( echo
"${EPOCH}|${UNIQUEID}|NONE|SIP/${EXTEN:3}|AGENTLOGIN|- " >>
/var/log/asterisk/queue_log )
exten => _422XX,4,DBput(dynlogin/log_Agent-${EXTEN:3}=${EPOCH})
exten => _422XX,5,Hangup
```

```
; Remove Member - 423
exten => _423XX,1,Answer
exten => _423XX,2,RemoveQueueMember(my-queue,SIP/${EXTEN:3})
exten => _423XX,3,DBget(ORGEPOCH=dynlogin/log_Agent-${EXTEN:3})
exten => _423XX,4,Set(RV=${${EPOCH} - ${ORGEPOCH}})
exten => _423XX,5,GotoIf("${RV}" = "0"?8:6)
exten => _423XX,6,System( echo
"${EPOCH}|${UNIQUEID}|NONE|SIP/${EXTEN:3}|AGENTLOGOFF|-|${RV}" >>
/var/log/asterisk/queue_log )
exten => _423XX,7,DBdel(dynlogin/log_Agent-${EXTEN:3})
exten => _423XX,8,Hangup
```

With this setup, we verified that the `queue_log` can be analyzed by QueueMetrics and the dynamic agent shows up fine (albeit with the name of a terminal, like `SIP/23`, instead of the usual `Agent/23` string, but you can modify it in QM itself).

This setup might even be used in a call center where agents are not actually used but queues connect straight to terminals to "fake" agent login/logoff, in order to have such data available for reporting.

Defining outbound queues

Standard Asterisk queues are, by definition, inbound queues; they accept a number of incoming calls, let them wait in line and distribute them to various agents based on the queue logic.

To make it possible to analyze outbound calls with QM, we create the concept of an "outbound queue", that is all calls made by different agents that belong to the same campaign. Of course there is no such thing as an outbound queue in Asterisk, so we have to run an AGI script to produce the same information on `queue_log` for outbound calls as it is automatically produced for inbound queues.

As this only regards the actual `Dial(...)` statement that Asterisk runs, it is possible to have different sources of numbers to be dialled by agents on outbound queues; they might enter the number on their keypad, or use the telephone, or maybe use a predictive dialler for the task. QueueMetrics does not care, as long as the correct events are logged.

The AGI script to be used instead of the `Dial(...)` command is available in the standard QM distribution and can be used in the following way:

```
exten => xxx,1,DeadAGI(queueDial.agi|Number|DialString|QueueName|Agent)
```

The following parameter have to be passed by dialplan logic:

- *Number*: the number you are trying to dial. Needed for correct logging only.
- *DialString*: the actual Asterisk dial string, like SIP/34, or maybe IAX2/usr:pass@iax.server/8885551234. If you need additional parameters in the Dial() command, modify the AGI script manually.
- *QueueName*: the outbound queue to be used for accounting. Must be defined in QueueMetrics and must not exist in Asterisk!
- *Agent*: the agent placing the call, e.g. Agent/123

A working example might be the following:

```
exten => 426,1,DeadAGI(queueDial.agi|34|SIP/34|queue-out-1|Agent/101)
```

The terminal SIP/34 is dialled and the resulting events are logged as if generated by Agent/101 working on queue-out-1.

Please note:

- The outbound queue should not be defined in Asterisk, but must be in QueueMetrics.
- When running a QueueMetrics analysis, some values are their own mirrors: like, the Caller*ID of an incoming call is the number dialled of an outbound queue.
- When monitoring calls in real-time, it is impossible to distinguish calls waiting to be answered from calls in conversation. This is an Asterisk limitation, as the generated events are not provided in real-time. Those values are anyway correct in the reports.
- Extensive debugging output is available at /var/log/asterisk/agi-log.txt

Enabling ACD call attempts recording on Asterisk 1.0 and 1.2

To get the AGENTATTEMPT code to work, it is necessary to patch the Asterisk module called app_queue.c in order to track down the required information. In order to perform this task, you must be confident with general Unix project patching and recompiling. It is advisable that Asterisk be shut down before applying the patch.

In order to apply the patch, just copy the file *app_queue_agentattempt.patch* found under *WEB-INF/README/* to the *apps/* directory of your Asterisk project, and then issue the following statement:

```
patch -p0 < app_queue_agentattempt.patch
```

As long as you see no errors, the patching process worked successfully. It's now time to rebuild the app by issuing a general make statement from the main Asterisk directory.

Restart Asterisk and check that the queue system is still working fine.

To see if the patch was correct, try dialling a queue and see that Asterisk writes AGENTATTEMPT records to the queue_log file.

Enabling ACD call attempts recording on Asterisk 1.4

Asterisk 1.4 is natively able to produce the RINGNOANSWER log entry that serves the same purpose of AGENTATTEMPT, so no patching is necessary.

Enabling Unattended Call Monitoring

To enable unattended audio monitoring you'll have to edit the Asterisk dialplan in order to start audio monitoring.

- Make sure that the *queuemetrics* context exists and that the extensions 10 and 11 are defined for it. See *Appendix III: The [queuemetrics] context* on page 127
- Make sure that the channel is set to *Local/\$EM@sip/n* if your current telephones call on context *sip*.
- Make sure that the extension/context are set to *11/queuemetrics* (the unattended audio monitoring endpoint).
- Make sure that the *callfile.dir* property points to a valid callfile directory, and that will be writable by QueueMetrics. As an alternative you may enter a Monitor URI in the format *tcp:user:pass@server*; in this case QM will not attempt to generate a call-file but will use the Manager command to create an equivalent call instead.
- Make sure the *callfile.monitoring.enabled* configuration property is set to *true*
- Make sure your users hold the MON_AUDIO key
- Make sure that each agent will have their local extension set in QueueMetrics; usually entering “-“ will be enough.
- Now, when you click on the icon, a callfile will be generated and call snooping will start.

Enabling VNC Monitoring

To enable VNC monitoring you will first need a VNC server that is running on each client's machine and that will serve the current layout.

You will also have to create a web page with a VNC client that may accept a VNC URL and show a VNC client (there are a number of Java-based VNC clients that can be displayed as an applet).

Configure the VNC URL as something like:

<http://myserver/vncpage.php?ip=192.168.3.17>

Where the PHP page will connect the VNC applet to the server located on address 192.168.3.17

Make sure that your users hold the MON_VNC key in order to be able to access this feature.

As an alternative, we have some clients that use a simpler setup with each machine having their own copy of UltraVNC - <http://ultravnc.sourceforge.net/> - and each machine running a web server with the locally-configured Java viewer. The VNC url is then the address of the local machine; when a person connects to it, s/he is asked for a password and then the screen is displayed through a Java applet. They report this setup to be very simple and working very well.

Enabling Agent's page actions

In order to enable actions on the Agent's page:

- Check that all actions are enabled in the properties, this means that `callfile.actionname.enabled=true`
- Check that a Manager API is configured correctly for the server
- Check that the dialplan on the server contains the appropriate commands for this action. A sample [queuemetrics] context you can include easily within a standard dialplan using call-back agents is provided as a reference.

Enabling XML-RPC call listening and streaming

It is possible to run remote audio monitoring of both completed and ongoing calls using third party monitoring tools, for example Orecx. As QueueMetrics has no way of knowing the internal details of such applications, we made it possible to call an external XML-RPC server (we offer a stub written in PHP, but it can be written in any language and reside on any server, as long as it uses an XML-RPC library) that will basically pass back to QM the URLs required to perform the required task.

The XML-RPC server will be set by setting its URL in a configuration property, like for example:

`default.audioRpcServer=http://127.0.0.1/xmlrpc/xmlrpc_findaudio.php`

The server must implements two XML-RPC calls called:

- QMAudio.findStoredFile

This function is used to find and play back a stored audio file, by returning the URL of a player that will play it or the audio file itself.

This function has in input the following parameters:

- \$ServerID: ignore for now
- \$AsteriskID: The Asterisk call-id, as written in the second field of queue_log
- \$QMUserID: the ID of the current QM user
- \$QMUserName: the name of the current QM user

and it must return the following values:

- \$FILE_FOUND : If the file was found or not (maybe it was not recorded)
- \$FILE_LISTEN_URL . an URL to open up a player for this call
- \$FILE_LENGTH : size of the audio file (shown as passed)
- \$FILE_ENCODING : encoding of the audio file (eg mp3)
- \$FILE_DURATION : duration of the audio file

- QMAudio.listenOngoingCall

This function is used to query for an ongoing call. If found, QM will launch a new popup to open the player which URL is returned.

This function has in input the following parameters:

- \$ServerID: ignore for now
- \$AsteriskID: The asterisk call-id, as written in the second field of queue_log
- \$Agent: the name of the agent being monitored e.g. "agent/101"
- \$QMUserID: the ID of the current QM user
- \$QMUserName: the name of the current QM user

and it must return the following values:

- \$CALL_FOUND: If the call was found or not
- \$CALL_LISTEN_URL : the URL of the player
- \$CALL_POPUP_WIDTH, \$CALL_POPUP_HEIGHT: width and height of the popup being opened. Currently a double popup is opened.

To make implementer's life easier, we provide a simple XML-RPC stub server under *WEB-INF/mysql-utils/xml-rpc* that can be used as a starting point: no need to handle the XML-RPC stuff, just change the results of the two supplied functions and data goes back to QueueMetrics.

Enabling call outcomes

A call tracking code is a code to be input by a user telling the status of a call, be it inbound or outbound. This status code is a string (though we suggest to use numeric status codes, in order to make it easy to input them using a telephone keypad) and may be input either when the call is ongoing or after a short while from its end.

The queue_log entry looks like the following one:

```
1234 | 1231.1 | NONE | Agent/1234 | CALLSTATUS | 21
```

This will set the CALLSTATUS to "21" for the call which Call-ID is "1231.1"; it may be an open call or it may be terminated by no longer than 30 minutes.

If it is not possible to force the Call-ID, a second version of the verb is available:

```
1234 | 2222.3 | NONE | Agent/1234 | CALLSTATUS | 21 | 1231.1
```

This has exactly the same meaning; the second Call-ID passed as a parameter will override the original one.

If you prefer, you may log the queue name instead of "NONE" field shown above; in any case QM will ignore this piece of information.

The following rules apply:

- A CALLSTATUS row must be set after the call is started or it's terminated; in any other case it's simply discarded
- There may be multiple CALLSTATUS rows for the same Call-ID; in this case, the last one overrides previous codes.
- The CALLSTATUS must be passed within 30 minutes from the end of a call.
- CALLSTATUS for a non-existent Call-ID will be discarded
- Even if a queue reset is detected, CALLSTATUS for existing Call-ID are applied

The agent may either be a fill "Agent/xxx" string or the valid name of an Asterisk channel. It is acceptable to use a generic channel name instead of the specific one, i.e. "SIP/123" and "SIP/123-abcd" are equivalent.

The sample [queuemetrics] context that comes with QueueMetrics can be used as a starting point to output such data.

Enabling pause codes

A pause reason code is a code to be input by a user telling the reason why a pause was started. It should be ideally input together with the decision to go on pause, though QueueMetrics will accept the code and will attach to the correct pause even if the pause is resumed, as long as no other pause is started. The reason code is a string - though we suggest to use numeric status codes, in order to make it easy to input it using a standard telephone keypad.

The format is the following one:

```
1234 | 1231.1 | NONE | Agent/1234 | PAUSEREASON | 21
```

This will set the pause reason to “21” for the pause that is either going on or has just finished. If the code is input after over 30 minutes from the end of the last pause, it is discarded.

The following rules apply:

- A PAUSEREASON row must be set after the agent’s pause is started or it’s terminated; in any other case it’s simply discarded
- There may be multiple PAUSEREASON rows for the same pause; in this case, the last one overrides pervious codes.
- The PAUSEREASON must be passed within 30 minutes from the end of a pause; otherwise it will be silently discarded.
- PAUSEREASON for a non-existent agent pause will be discarded.
- If a pause extends over multiple call sessions, the PAUSEREASON will be correctly set only for sessions terminating after the PAUSEREASON has been set.
- Even if a queue reset is detected, PAUSEREASON for existing pause are applied
- The agent may either be a fill “Agent/xxx” string or the valid name of an Asterisk channel. It is acceptable to use a generic channel name instead of the specific one, i.e. “SIP/123” and “SIP/123-abcd” are equivalent.

The sample [queuemetrics] context that comes with QueueMetrics can be used as a starting point to output such data.

For more information...

To know more about QueueMetrics in your specific setting or inquire about commercial licences, please feel free to contact Loway.

The latest version of QM can be found on the home page located at the address <http://queuemetrics.com>

A number of how-to's and recipes about QM are available on AstRecipes, see <http://www.astrecipes.net>

There is a QueueMetrics users forum for mutual support, troubleshooting and ideas at <http://forum.queuemetrics.com>

Appendix I: Default users

The following users are come preconfigured in the default database.

Login	Password	Explanation
demoadmin	demo	The sample admin user
demouser	demo	The sample CC manager
demovisitor	demo	A sample visitor
demosupervisor	demo	A sample supervisor
robot	robot	A sample robot
Agent/101	999	A sample agent
Agent/102	998	Another sample agent

Make sure you change their default passwords before letting users access QM!

Appendix II: Security keys

The following security keys are defined:

KEY	MEANING
USER	Must be held by any valid user
USRADMIN	User can edit other users and classes
USR_AGENT	User can edit agents
USR_QUEUE	User can edit queues
USR_LOCATION	User can edit locations
USR_OUTCOME	User can edit call outcomes
USR_PCODE	User can edit pause codes
USR_MYSQL	User can see the MySQL database page
REALTIME	User can see real-time stats
QUEUE_AN	User can run reports
AGREP	User can filter reports by agent
AGENT	User is an agent and sees agent page
CALLMONITOR	The user can listen to a recorded call
MON_AUDIO	The user can monitor a real-time call
MON_VNC	The user can monitor an agent's screen via VNC
ROBOT	User may launch ROBOT transactions.
CHPASSWD	User can change his own access password
SUPERVISOR	User is a supervisor and can run the supervisor's report

Appendix III: The (queuemetrics) context

QueueMetrics is able to trigger a number of advanced functionalities, like audio monitoring, clients logging in, going on pause, etc. right from the Asterisk dialplan.

In order to make this portable and easy to understand, we suggest to create a special context named *queuemetrics* in your dialplan where QueueMetrics will trigger functions through a callfile. An example file that is ready-to-use for most call centres can be found under *WEB-INF/mysql-utils/extensions-examples* – see the included README file for more details.

We therefore define a number of functions in the terms of extension relative to the context *queuemetrics*, as follows:

Ext. Notes

10 *Dummy extension.*

Used only because a call-file requires two end-points in any case. Define as:

```
exten => 10,1,Answer
exten => 10,2,Wait(10)
exten => 10,3,Hangup
```

11 *Remote monitoring.*

This extension makes unattended monitoring possible through the command ChanSpy(). The variables QM_AGENT_CODE, QM_EXT_MONITOR and QM_AGENT_EXT are set. The following example explains how the feature works:

```
exten => 11,1,Answer
exten => 11,2,NoOp( "QM_AGENT_CODE: ${QM_AGENT_CODE}" )
exten => 11,3,NoOp( "QM_EXT_MONITOR: ${QM_EXT_MONITOR}" )
exten => 11,4,NoOp( "QM_AGENT_EXT: ${QM_AGENT_EXT}" )
exten => 11,5,ChanSpy( ${QM_AGENT_CODE} | q )
exten => 11,6,Hangup
```

12 *Call status code*

This extension logs a calls status code. The variables CALLSTATUS, CALLID and AGENTCODE are defined. The following example explains how the feature works:

```
exten => 12,1,Answer
exten => 12,2,NoOp( "QM: Setting call status '${CALLSTATUS}' on call
'${CALLID}' for agent '${AGENTCODE}'" )
exten => 12,3,System( echo
```



```
"${EPOCH}|${CALLID}|NONE|Agent/${AGENTCODE}|CALLSTATUS|${CALLSTATUS}" >>
/var/log/asterisk/queue_log )

exten => 12,4,Hangup
```

20 *Agent login*

This extension logs in a call-back agent. The variables AGENTCODE and AGENT_EXT are defined. Please note that for this to work properly, there must be no password set on the Asterisk agent.

The following example explains how the feature works:

```
exten => 20,1,Answer

exten => 20,2,NoOp( "QM: Logging on Agent/${AGENTCODE} to extension
${AGENT_EXT}@sip" )

exten => 20,3,AgentCallBackLogin(${AGENTCODE}||${AGENT_EXT}@sip)

exten => 20,4,Hangup
```

21 *Agent logoff*

This extension logs off an agent. The variable AGENTCODE is defined. The following example explains how the feature works:

```
exten => 21,1,Answer

exten => 21,2,NoOp( "QM: Logging off Agent/${AGENTCODE}" )

exten => 21,3,System(asterisk -rx "agent logoff Agent/${AGENTCODE}")

exten => 21,4,Hangup
```

22 *Agent pause (with pause code)*

This extension pauses an agent and sets the pause code. The variables AGENTCODE and PAUSEREASON are defined. The following example explains how the feature works:

```
exten => 22,1,Answer

exten => 22,2,NoOp( "QM: Pausing Agent/${AGENTCODE} with pause reason
'${PAUSEREASON}' " )

exten => 22,3,PauseQueueMember(|Agent/${AGENTCODE})

exten => 22,4,System( echo
"${EPOCH}|${UNIQUEID}|NONE|Agent/${AGENTCODE}|PAUSEREASON|${PAUSEREASON}"
>> /var/log/asterisk/queue_log )

exten => 22,5,Hangup
```

23 *Agent unpause*

This extension unpauses an agent. The variable AGENTCODE is defined. The

following example explains how the feature works:

```
exten => 23,1,Answer
exten => 23,2,NoOp( "QM: Unpausing Agent/${AGENTCODE} " )
exten => 23,3,UnpauseQueueMember( |Agent/${AGENTCODE} )
exten => 23,4,Hangup
```

In order to trigger these functions, QueueMetrics need to be able to access the Asterisk callfile spool, as defined by the callfile.dir property.. If your Asterisk system is remote, you'll have to arrange a periodic file transfer or use a disk share in order to make the above features work.

As an alternative, QueueMetrics may connect to a working Asterisk server over the Manager interface. See the description of the callfile.dir property for more information.

Appendix IV: Glossary

Agent: a person working at the monitored call center and answering to calls. Asterisk offers a way for agents not to be bound by physical telephone terminals but to log on to tell the system they are available.

Call-back agent: an agent that will not stay on-line, but which telephone will be rung by Asterisk when a call comes in for him.

Caller: a person calling the Asterisk system

Callfile: a function in Asterisk, where by writing a specially-crafted file, it is possible to interact with the dialplan.

Composite queue: A virtual queue made of more than one physical queue. Useful for reporting all center activity at once.

Manager interface: a TCP/IP Asterisk interface, where a process with the right credentials can connect to a remote Asterisk server over the network and control or query its behaviour. Must be enabled manually by the Asterisk administrator.

Monitoring: in Asterisk terminology, the act of recording to disk.

Queue: the call distribution object that let Asterisk keep callers waiting and distributes them in the correct order to available agents. Each caller is processed on a first-come-first-server basis.

VNC: a technology that can display the screen of another computer on your own screen through a TCP/IP connection. A number of free and commercial VNC implementations exist.

