Loway

QueueMetrics

call center suite

QUEUEMETRICS USER MANUAL

Loway SA

Version 26.10, 2026/10/12

Table of Contents

What is QueueMetrics? . . . 1
- Reporting . . . 1
- Supervision . . . 1
- Agent page . . . 2
- Quality Assessment . . . 2

Installing QueueMetrics . . . 3
- The big picture . . . 3
- Installing QueueMetrics . . . 3
  - Installing dependencies . . . 4
  - Installation . . . 6
    - Installing an exact version . . . 7
  - Adding a license . . . 7
  - Further configuration . . . 7
    - Internet access . . . 8
    - Setting up HTTPS . . . 8
    - Logging . . . 8
    - JMX monitoring . . . 9
    - Installing in Secure Mode . . . 9
  - Upgrading . . . 9
    - The database update utility . . . 9
  - Backing up . . . 10
  - Using queuemetrics-cli . . . 11
  - Updating from a RPM-based distribution . . . 11
    - Collecting information from the old installation . . . 12
    - Decommissioning the old system . . . 12
    - Reinstalling on the same server . . . 13
    - Installing on a different server . . . 14
- Installing QueueMetrics using Docker . . . 14
  - Installing the official QueueMetrics Docker image . . . 14
    - Create a data container . . . 15
    - Start the QueueMetrics image . . . 15
    - Accessing an instance using the shell . . . 16
    - Terminating an instance . . . 16
    - Using a data-only volume . . . 16
  - Applying a license . . . 16
  - Upgrading QueueMetrics . . . 17
  - Tips and tricks . . . 17
    - Defining a fixed QM port . . . 17

Configuring license, memory, time zones and more . . . 17
Configuring QueueMetrics properties . . . 19
Installing from RPM . . . 19
Configuring QueueMetrics . . . 19
Uniloader configuration . . . 19
QueueMetrics configuration . . . 20
Include QueueMetrics dialplan in Asterisk . . . 21
Granting access to local recordings . . . 21
Does it work? . . . 22
Troubleshooting . . . 22
Understanding QueueMetrics memory requirements . . . 22
Understanding basic security mechanisms . . . 23
QueueMetrics is not available . . . 23
Editing configuration settings . . . 24
The System Diagnostic Tools . . . 24
Using an HTTP/S proxy . . . 24
Version numbering scheme . . . 26
Backing up QueueMetrics . . . 26
Fixing damaged indexes . . . 27
License keys . . . 27
What about licenses? . . . 28
Installing a new key through the GUI . . . 28
License expiration notification . . . 29
Monitoring clusters with QueueMetrics . . . 30
Setting up a cluster . . . 30
Setting up your PBXs as members of the cluster . . . 31
Setting up QueueMetrics to access the cluster . . . 32
Using the Agent's page with a clustered environment . . . 33
Logging into a server from the Icon Agent's Page . . . 33
QueueMetrics Home Page . . . 35
InfoBoxes . . . 36
InfoBox Selector . . . 36
Administration Menu . . . 37
Search Bar . . . 37
Header . . . 38
System Administration . . . 38
Diagnostics . . . 39
License information . . . 40
Change Password and Theme . . . 40
Automatic URL login . . . 41
Change Your Logo image . . . 42

Report Page . . . 43
What is a Report? . . . 43
Unsaved vs Saved Reports . . . 43
DataBlocks . . . 45
Pages . . . 46
Icons . . . 47
Create Report . . . 47
Settings . . . 48
Add DataBlocks . . . 48
Share . . . 48
Export in PDF Format . . . 48
Export in XLS Format . . . 48
Export Report Schema . . . 48
Import Report Schema . . . 48
Schedule export Job . . . 48
Delete Report . . . 49
Filter Recap Panel . . . 49
Call Stats . . . 49
Selected Report . . . 49
Selected Queue . . . 50
Time Period . . . 50
Filter Expression . . . 50
Rename Report . . . 50
Delete Report . . . 51
Change Queues . . . 51
Save or Apply . . . 51
Add/Edit Filter Expression . . . 52
Type of filters . . . 53
Combo Filters . . . 53
Combo Filters with Creatable Options . . . 53
Text Filters . . . 53
Regular Expressions . . . 53
Filter Expression . . . 53
Add Filters . . . 53
Change Operator . . . 54
Grouping Filters . . . 54
Deleting a Filter . . . 54
Deleting a Filter Group . . . 54
Editing a Filter . . . 55
Editing a Regexp Filter . . . 55
Negating a Filter . . . 55

Editors . . . 56
Page . . . 56
QueueMetrics Print Report Page Feature . . . 57
Print mode . . . 57
Exiting print mode . . . 59
DataBlock Editor . . . 59
Recap Row . . . 60
Report Settings . . . 61
Schedule Export Job . . . 62
Self Service Agent Reporting (SSAR) . . . 63
Timeline . . . 65
Call Details . . . 66
Timeline . . . 66
Recordings and QA . . . 67
Audio Player . . . 67
Chat . . . 68
Chat and Transcription Tabs . . . 68
Viewing Messages . . . 69
Transcription Message Synchronization . . . 69
Editing Messages . . . 70
Editing message text . . . 70
Changing the speaker . . . 71
View History, Managing Speakers and Delete Chat . . . 71
Managing Speakers . . . 71
Viewing Message History . . . 72
Chat Delete . . . 73
Call Grouping . . . 73
Call filtering criteria . . . 74
Filtering by call variables . . . 75
Short calls . . . 75
Understanding results: Answered calls . . . 76
Agents on queue . . . 76
Service level agreement . . . 77
Disconnection causes . . . 77
Transfers . . . 78
Answered calls by queue . . . 78
Answered calls by direction . . . 78
Answered calls, by stints . . . 79
Queue position . . . 79
IVR selection . . . 79
DNIS used . . . 80

Detail of answered calls . . . 80
Understanding results: Unanswered calls . . . 80
Disconnection causes . . . 81
Unanswered calls, by queue . . . 81
Unanswered calls - distribution by length . . . 81
Inclusive Service Level Agreement . . . 82
Unanswered calls by key press . . . 82
Unanswered calls, by stints . . . 82
All calls, by stints . . . 83
Enter queue positions . . . 83
Enter queue positions for all calls . . . 83
IVR selection . . . 84
IVR selection for all calls . . . 84
DNIS used . . . 84
DNIS used, for all calls . . . 84
Details of unanswered calls . . . 85
Understanding results: Inbound ACD call attempts . . . 85
ACD attempts by terminal . . . 85
ACD attempts by queue . . . 86
Understanding results: Call distribution . . . 86
Call distribution per day . . . 86
Call distribution per hour . . . 87
Call distribution per day of week . . . 87
Understanding results: Agent activity . . . 88
Agent availability . . . 88
Session and pause duration . . . 89
Answered calls for selected queues . . . 89
Answered calls by service groups . . . 89
Session details . . . 90
Pause activity details . . . 90
Agent history popup . . . 91
Understanding results: Call outcomes . . . 91
How are Call Outcomes calculated? . . . 92
Multi-stint calls . . . 93
Limitations and side-effects . . . 93
Multi-stint calls in QueueMetrics . . . 94
Displaying calling/called numbers . . . 94
Exporting calls - an overview (Deprecated function) . . . 95
Exporting calls in practice (Deprecated function) . . . 97
Output format . . . 99
Available implementors . . . 100

MP3 conversions on the fly . . . 101
Showing call details . . . 105
Detail of answered calls . . . 105
Listening to answered calls . . . 107
The HTML5 Audio Player and Markers . . . 108
Auditing access to recordings: the Media Proxy . . . 109
Securing the media storage layer . . . 110
Detail of unanswered calls . . . 110
Detail of IVR calls . . . 112
Report Details . . . 113
Historical reports - Answered calls . . . 114
OK01 - All calls . . . 114
OK02 - Calls fully within the given time interval . . . 115
OK03 - Agents on queue . . . 115
OK04 - Service level agreement . . . 116
OK05 - Disconnection causes . . . 118
OK06 - Transfers . . . 119
OK07 - Answered calls, by queue . . . 120
OK08 - Answered calls, by direction . . . 120
OK09 - Answered calls, by stints . . . 121
OK10 - Queue position . . . 121
OK11 - IVR selection . . . 122
OK12 - DNIS used . . . 122
OK13 - Music On Hold by Agent . . . 123
OK14 - Report Header . . . 123
OK15 - Required Skills for Taken Calls . . . 124
OK16 - Overview by Agent . . . 125
OK17 - Overview by Queue . . . 126
Historical reports - Details of answered calls . . . 127
OD01 - Queue details . . . 127
OD02 - Add to export job . . . 127
OD03 - Queue details (paged) . . . 128
OD04 - IVR details (paged) . . . 128
OD05 - IVR details (full list) . . . 129
OD06 - Executive Summary . . . 129
Historical reports - Unanswered calls . . . 131
UN01 - All calls . . . 131
UN02 - Calls fully within the given time interval . . . 132
UN03 - Disconnection causes . . . 133
UN04 - Unanswered calls, by queue . . . 134
UN05 - Unanswered outbound calls, by agent . . . 134

UN06 - Unanswered calls - distribution by length . . . 135
UN07 - Inclusive SLA (computed on both answered and unanswered calls) . . . 136
UN08 - Unanswered calls by key press . . . 137
UN09 - Unanswered calls, by stints . . . 138
UN10 - All calls, by stints . . . 138
UN11 - Enter queue position . . . 138
UN12 - Enter queue position for all calls . . . 139
UN13 - IVR selection . . . 140
UN14 - IVR selection, for all calls . . . 140
UN15 - DNIS used . . . 141
UN16 - DNIS used, for all calls . . . 141
UN17 - Call Overview . . . 142
UN18 - Inclusive Answered SLA . . . 143
UN19 - Required Skills for Lost Calls . . . 144
UN20 - Required Skills for All Calls . . . 144
UN21 - Numbers to be recalled . . . 145
UN22 - Numbers recalled . . . 146
UN23 - Recap of all recalls . . . 147
Historical reports - Details of unanswered calls . . . 148
UD01 - Detail of unanswered calls . . . 148
UD02 - Lost calls (paged) . . . 148
Historical reports - Area code analysis . . . 149
AC01 - Area code report . . . 149
AC02 - Detail for answered calls . . . 149
AC03 - Detail for unanswered calls . . . 150
AC04 - Frequent Area Codes . . . 150
Historical reports - Distributions . . . 151
AT01 - Inbound ACD call attempts . . . 151
AT02 - ACD attempts by queue . . . 152
AT03 - ACD attempts by terminal . . . 153
AT04 - Last ACD Attempts . . . 154
Historical reports - Call distribution by day . . . 154
DD01 - Answered call distribution per day . . . 155
DD02 - Answered call wait time per day . . . 155
DD03 - Unanswered call wait time per day . . . 156
DD04 - Sales per day . . . 156
DD05 - Schedule Adherence per day . . . 156
DD06 - Queue length per day . . . 157
DD07 - Inclusive SLA per day . . . 157
DD08 - Traffic Analysis by period - per day . . . 158
DD09 - Skill sets per day . . . 159

DD10 - Detail Summary . . . 161
Historical reports - Call distribution by hour . . . 162
DH01 - Answered call distribution per hour . . . 162
DH02 - Answered call wait time per hour . . . 163
DH03 - Unanswered call wait time per hour . . . 164
DH04 - Sales per hour . . . 164
DH05 - Schedule Adherence per hour . . . 165
DH06 - Queue length per hour . . . 166
DH07 - Inclusive SLA per hour . . . 167
DH08 - Traffic Analysis by period - per hour . . . 168
DH09 - Skill sets per hour . . . 169
DH10 - Detail Summary . . . 170
Historical reports - Call distribution by day of week . . . 172
DW01 - Answered call distribution per day of week . . . 172
DW02 - Answered call wait time per day of week . . . 172
DW03 - Unanswered call wait time per day of week . . . 173
DW04 - Sales per day of week . . . 173
DW05 - Schedule Adherence per day of week . . . 174
DW06 - Queue length per day of week . . . 174
DW07 - Inclusive SLA per day of week . . . 175
DW08 - Traffic Analysis by period - per day of week . . . 176
DW09 - Skill sets per day of week . . . 176
DW10 - Detail Summary by Day of the week . . . 177
Historical reports - Agents and Sessions . . . 177
AG01 - Agent session detail . . . 178
AG02 - Session and pause durations . . . 179
AG03 - Agent availability (for all the queues they are member of) . . . 179
AG04 - Answered calls (for selected queues) . . . 180
AG05 - Answered calls by custom group . . . 180
AG06 - Answered calls by location . . . 181
AG07 - Answered calls by service group . . . 182
AG08 - Agent Performance by ACD Group . . . 182
AG09 - Agent Occupancy Report . . . 183
AG10 - Agent Session Time by Hour . . . 185
AG11 - Agent Payable Time by Hour . . . 186
AG12 - Agent Billable Time by Hour . . . 187
AG13 - Agent Session View . . . 187
AG14 - Queue Session View . . . 189
AG15 - Agents by Hour . . . 190
AG16 - Tag Session View . . . 191
AG17 - Cumulated sessions by Queue, Tag . . . 193

AG18 - Programmable Agent Performance . . . 194
AG19 - Detailed Pause Tracking . . . 194
AG20 - Agent Talk Time Per Hour . . . 195
AG21 - Agent Outbound Productivity . . . 196
AG22 - Agent Session Overview . . . 197
AG23 - Details of Pause Types . . . 198
Historical reports - Details of agent sessions and pauses . . . 198
AD01 - Detail of agent sessions . . . 198
AD02 - Detail of agent pauses . . . 199
Popup of agent activity . . . 200
Historical reports - Call outcomes . . . 201
OU01 - Outcomes . . . 201
OU02 - Call results, by outcomes . . . 202
OU03 - Billable activities . . . 203
OU04 - Non billable activities . . . 203
OU05 - Detailed agent report . . . 204
OU06 - Outcomes per agent . . . 204
OU07 - Call results, by tag . . . 205
OU08 - Call results, by feature code . . . 206
Historical reports - IVR tracking . . . 207
IV01 - IVR Traversals . . . 208
IV02 - IVR Timing . . . 208
IV03 - IVR Goals . . . 209
Historical reports - First Call Resolution (FCRs) . . . 210
Repeated callers and clusters . . . 210
FC01/2/3 - Break-down of recalls . . . 211
FC04/5/6 - Recall details . . . 212
FC07/8/9 - Cluster details . . . 213
FC10 - Retry Rate and Lost Caller Behaviour . . . 214
FC11 - Most Frequent Called-Calling Number . . . 216
Historical reports - Cases and Clients . . . 217
CA01 - Case Distribution . . . 217
CA02 - Client Distribution . . . 217
Historical reports - Microsoft Teams . . . 218
TM01 - External Callers . . . 219
TM02 - External Callees . . . 220
TM03 - External Callers on All Calls . . . 220
TM04 - External Callees on All Calls . . . 221
TM05 - Teams Caller Classification . . . 221
TM06 - Teams Callee Classification . . . 222
Externally generated reports . . . 222

XS01 - Plain HTML . . . 222
XS02 - XML-RPC data source . . . 223
XS03 - JSON data source . . . 223
User-Programmable Reports . . . 223
Variables bound for Agent blocks . . . 225
JSON and XML-RPC access . . . 226
The real-time status panel . . . 227
Top status panel . . . 229
Calls being processed . . . 230
Agents currently logged in . . . 232
Aggregated by queue view . . . 234
Aggregated by tag view . . . 234
Agent and outcome view . . . 235
Using Locations . . . 236
Unattended call and VNC monitoring . . . 237
Real-time performance considerations . . . 238
The real-time live page . . . 239
The top panel . . . 240
Calls being processed . . . 241
Agents currently logged in . . . 241
Server status . . . 241
The Wallboard . . . 243
Understanding wallboards . . . 243
Saved Wallboards . . . 244
Settings . . . 244
Editing Widgets . . . 245
Alarms . . . 245
Running in kiosk mode . . . 245
Wallboard filters . . . 246
Wallboard actions . . . 248
Agent actions . . . 248
Call actions . . . 250
Importing and Exporting Wallboards . . . 251
Exporting a Wallboard . . . 251
Importing a Wallboard . . . 253
Wallboard softphone . . . 255
System Parameters Defaults . . . 257
The Softphone Panel . . . 258
Call Transfer . . . 259
Advanced Audio Device Selection . . . 262
Audio Device Routing Options . . . 262

HID Device Management . . . 263
HID Device Button Behavior . . . 263
Device Disconnection and Fallback . . . 263
Persistence and Memory . . . 263
Known Limitations . . . 264
ICE Gathering Optimization . . . 265
ICE Gathering Mechanism . . . 265
Configuration . . . 265
Behavioral Logic Flow . . . 265
Component Details . . . 266
Additional details . . . 266
Circular Panels . . . 267
PC01 - Waiting Calls . . . 267
PC02 - Lost Calls . . . 268
PC03 - Answered Calls . . . 269
PC04 - Offered Calls . . . 270
PC05 - Longest Wait . . . 271
PC06 - Agents On Call . . . 272
PC07 - Lost Calls % . . . 273
PC08 - Answered Calls % . . . 274
PC09 - Agents Ready . . . 275
PC10 - SLA Level % . . . 276
PC11 - Reservable Calls . . . 277
PC12 - Incomplete Calls . . . 278
PC13 - Ephemeral - Value . . . 279
Tables . . . 280
TC01 - Live Agents . . . 281
TC02 - Live Calls . . . 282
TC03 - Live Queues . . . 282
TC04 - Aggregated By Queue . . . 283
TC05 - Aggregated By Tag . . . 284
TC06 - Agents and Outcomes . . . 284
Items . . . 285
IT01 - N. of Calls Offered . . . 285
IT02 - Longest Wait . . . 286
IT03 - N. of Calls Lost . . . 287
IT04 - N. of Calls Answered . . . 288
IT05 - Ephemeral - Time . . . 288
Composite Items . . . 289
MI01 - Live Agents Counters . . . 289
MI02 - Live Agent Info . . . 290

MI03 - Live Queue Info . . . 292
MI04 - Average Queue Info . . . 293
MI05 - Campaign Stats . . . 294
Charts . . . 295
BC01 - Bar Graph SLA . . . 295
Other . . . 296
CL01 - Clock . . . 296
TX01 - Text . . . 296
TX02 - Ephemeral - Text . . . 297
HTML01 - HTML . . . 298
DB01 - DataBlock Table . . . 299
DB02 - DataBlock Chart . . . 300
Expired widget . . . 301
Agent page . . . 302
Login . . . 302
Overview . . . 304
Start/End Work . . . 305
Navigation Menu . . . 305
Control . . . 307
Extension . . . 307
Joining queues . . . 308
Interactions . . . 310
Interaction Details . . . 310
Wallboard . . . 312
WombatDialer Recalls . . . 312
Clients and Cases . . . 313
Clients . . . 313
Cases . . . 315
CRM Integration . . . 316
Screen-pops to your CRM application . . . 317
SoftPhone . . . 318
Softphone settings . . . 321
Advanced Audio Device Selection . . . 323
Audio Device Routing Options . . . 323
HID Device Management . . . 324
HID Device Button Behavior . . . 324
Device Disconnection and Fallback . . . 324
Persistence and Memory . . . 324
Known Limitations . . . 325
ICE Gathering Optimization . . . 326
ICE Gathering Mechanism . . . 326

Configuration . . . 326
Behavioral Logic Flow . . . 326
AMO Campaigns . . . 327
Disable outbound autonomous calls . . . 329
Agent Status Panel . . . 330
Notifications . . . 331
QueueMetrics Memo System . . . 332
Daily View . . . 333
Weekly View . . . 333
Manager View . . . 334
Memo Editor . . . 335
Memo Creator . . . 336
Notifications . . . 336
Recall Memos . . . 337
WombatDialer Preview Dialing . . . 337
Configuration . . . 338
The Dialing Workflow . . . 338
Managing Input Attributes and External URLs Integration . . . 339
Input Attribute Visibility . . . 339
Individual External Links per Attribute . . . 339
Call Handling and Disposition . . . 340
Setting the Call Outcome (Extended Status) . . . 340
Defining Output Attributes (Custom Data Fields) . . . 341
Reference: WombatDialer Integration Properties . . . 341
Full Configuration Example . . . 342
Keyboard Shortcuts . . . 342
Softphone Keyboard Shortcuts . . . 343
Softphone System Parameters . . . 343
Debug Page Activation . . . 343
Assisted Manual Outbound . . . 345
Assisted campaigns . . . 345
The mixer . . . 346
Rescheduling . . . 346
Attributes . . . 346
Call lists . . . 347
A call's lifecycle . . . 347
Failures and recalls . . . 349
QueueMetrics Users . . . 350
Default users . . . 350
Password reset . . . 350
Restricted user administrators . . . 351

Using Supervisors . . . 351
Using an external auth source for QueueMetrics . . . 351
Overview . . . 351
Driver: JSON . . . 353
Driver: LDAP . . . 354
Driver: XML-RPC . . . 355
Using Single Sign-On with QueueMetrics . . . 355
Overview . . . 356
Why flagging users for access through the token field . . . 357
Creating users for SSO . . . 358
Automatic creation of foreign users . . . 358
Security considerations . . . 359
Logging . . . 359
Limitations . . . 359
The return URL . . . 359
SSO drivers: Google . . . 359
Creating credentials . . . 360
Configuring QueueMetrics . . . 360
SSO drivers: Microsoft . . . 360
Creating credentials . . . 360
Configuring QueueMetrics . . . 361
Generate PDF and XLS reports and send them by e-mail . . . 361
Setting up a self-service wallboard . . . 362
Editing QueueMetrics settings . . . 364
Configuring users . . . 364
Editing user classes . . . 367
Password Encryption . . . 369
Configuring queues . . . 369
The Queue URL . . . 373
Setting attention levels (Red and yellow alarms) . . . 374
Using wildcards in queue names . . . 374
AMO Settings . . . 375
Configuring redial services through WombatDialer . . . 375
Configuring queues to be processed by the AGAW Runner . . . 375
Configuring AMO campaigns . . . 377
Configuring AMO lists . . . 378
Configuring AMO numbers . . . 379
Configuring agents . . . 380
Asterisk aliases . . . 383
Configuring locations . . . 384
Configuring call outcomes . . . 384

Configuring Feature Codes . . . 386
Configuring Tag Codes . . . 388
Configuring pause codes . . . 389
Configuring agent groups . . . 391
Configuring skills . . . 392
Configuring known numbers (VIP callers / blacklists) . . . 393
Configuring QA forms . . . 395
Handling dependent questions . . . 398
Configuring QA items . . . 399
Item value type: Numeric . . . 401
Item value type: Yes/No . . . 401
Item value type: Combo . . . 401
Defining agent performance rules . . . 401
Available targets . . . 404
Setting targets . . . 404
Dataset-based agent performance wizard . . . 405
Configuring Custom DataBlocks . . . 405
DataBlock Source Code . . . 407
Adding Custom DataBlocks to a report . . . 408
Explain DataBlocks . . . 408
Custom DataBlocks Charts . . . 409
Report Export Configuration pages . . . 412
Scheduled Jobs Configuration page . . . 413
Configuring IVR and DID/DNIS names . . . 415
Configuring CBTs (Computer Based Training) . . . 416
Configuring Cases and Clients . . . 416
Configuring Clients . . . 417
Configuring Cases . . . 418
Configuring Transcription Servers . . . 418
Configuring paged call lists . . . 420
Configuring the new Real-time Agents Page . . . 423
QueueMetrics configuration wizard . . . 425
Unattended QueueMetrics configuration and update . . . 431
Configuring system preferences . . . 433
Explore system parameters . . . 433
Edit system parameters . . . 434
Using the DbTest Diagnostic Tools . . . 436
Checking the current system configuration . . . 436
Checking an Asterisk Manager connection . . . 437
Checking the AMI Queue . . . 438
Live inspector of the QueueMetrics database . . . 439

Live ephemeral attribute inspector of the QueueMetrics database . . . 442
RAM cache inspector . . . 442
Testing email delivery . . . 443
System audit log inspector . . . 444
Engine audits . . . 445
QueueMetrics Themes . . . 445
Currently selected theme . . . 446
User Information Page . . . 446
QueueMetrics Configuration Synchronizer . . . 446
JSON File . . . 447
URL . . . 448
Queue Log Database Table . . . 449
Queue Log File . . . 450
Editing the configuration manually . . . 450
Deleting agents and queues . . . 452
Assigning Agents to Queues . . . 452
Applying the configuration . . . 453
Disabling actions . . . 453
Exporting configurations . . . 455
Creating an Agent . . . 455
Creating a Queue . . . 456
General Actions . . . 458
The QueueMetrics Platform model . . . 460
Generic concepts . . . 460
Defining system properties for members of a cluster . . . 460
Ingestors . . . 461
Storage models . . . 461
Outbound dialer . . . 461
None . . . 461
WombatDialer . . . 461
Listening to calls using Pluggable Modules (PM) . . . 462
Supported platforms . . . 463
Classic . . . 463
QueueMetrics-Live . . . 463
Freeswitch . . . 464
Enswitch . . . 464
MirtaPBX . . . 464
JSON API . . . 465
Platform: PBX Actions . . . 466
DirectAMI actions . . . 466
DirectAMI Live actions . . . 469

Classic PBX Actions . . . 469
QMLive PBX Actions . . . 470
Freeswitch actions . . . 470
Freeswitch Live actions . . . 473
JSON API Actions . . . 473
Enswitch actions . . . 476
MirtaPBX actions . . . 479
Platform: PMs to match Recorded Calls . . . 482
Using multiple PMs at once: MultiListener . . . 482
Plain old recordings: LocalFiles . . . 483
Large storage with recordings: LocalFilesByDay . . . 484
Pluggable Listener for the Enswitch platform . . . 485
Pluggable Listener for the MiRTA PBX platform . . . 485
Query an external system: JsonListener . . . 486
External audio recorder: OrekaWeb . . . 488
Propagating the SIP header . . . 489
Configuring event capture in Oreka . . . 490
Which version of Oreka do I need? . . . 490
Video playback . . . 490
Advanced Oreka support: OrekaEncrypted . . . 490
Secure access . . . 491
(Obsolete) Using an external server: ClassicXmlRpcRecordings . . . 492
(Obsolete) Pluggable Listener for Enswitch Platform Integration . . . 492
(Obsolete) Pluggable Listener for CallCabinet Platform Integration . . . 493
Cluster Mode . . . 496
Platform: PMs to match Live Calls . . . 497
Live calls through QueueMetrics: ClassicQMListenerRT . . . 497
(Obsolete) Live calls through an external module: ClassicXmlRpcListenerRT . . . 497
(Obsolete) Live calls through Oreka: OrekaWebRT . . . 498
Using only one Oreka server . . . 499
Using a cluster of Oreka servers . . . 499
Using multiple Oreka servers with UniqueID . . . 499
Expanded properties . . . 500
Quality Assessment in QueueMetrics . . . 501
Enabling QA monitoring . . . 501
Understanding Quality Assessment . . . 501
Grading calls . . . 502
Grading historical calls . . . 502
The input form . . . 502
Grading over HTTP access . . . 505
Removing or Editing QA forms . . . 506

Running QA reports . . . 506
The main QA report . . . 508
The QA Summary report . . . 510
The Training And Coaching Page . . . 512
Advanced tracking of agent and grader performance . . . 515
Tracking agent performance . . . 516
Dynamic drill down . . . 520
Taking remedial actions . . . 520
Move the agent to a different group . . . 521
Send a CBT to the selected agent . . . 521
Send a Coaching (Meeting) task to the selected agent . . . 521
The Grader's Page - Finding calls to be graded . . . 521
Selecting which calls to grade . . . 524
Grader calibration reports . . . 524
Call transcriptions and Automated QA . . . 527
Concepts . . . 527
The life-cycle of a call . . . 527
Setting up the server pool . . . 528
Monitoring . . . 529
Transcribing calls, in practice . . . 529
Extracting a call's Unique-Id from the recording name . . . 529
I forgot to capture the Unique-Id. What now? . . . 530
Setting up QueueMetrics for transcriptions . . . 531
Automating Quality Assessment . . . 534
Automated QA in practice . . . 535
Troubleshooting automated QA . . . 537
Transcription drivers . . . 538
OpenAI . . . 538
OpenAI-compatible API . . . 539
Deepgram . . . 540
KrokoAI . . . 541
LLM drivers . . . 541
OpenAI . . . 541
Anthropic . . . 542
QueueMetrics Tasks . . . 544
The task page . . . 544
Extended task reports . . . 547
Task Statistics . . . 547
Tasks for groups: Class tasks . . . 548
Task validity and expiration . . . 549
Types of tasks handled by QueueMetrics . . . 549

Call QA Graded Task . . . 549
Note Task . . . 549
Training Task . . . 550
Meeting Task . . . 550
RSS data export for tasks . . . 550
Setting computed URLs . . . 550
Payroll data in QueueMetrics . . . 552
How it works . . . 552
Payroll web pages . . . 552
The search page . . . 553
The Sessions page . . . 554
The Agent detail page . . . 555
The Session editor page . . . 556
The pause editor page . . . 556
Editing the system queue_log file . . . 558
Well-formed agent sessions . . . 558
The editing log . . . 558
Configuring Asterisk for QueueMetrics . . . 559
Configuring queues to report exit status . . . 559
Configuring URLs to be launched by the agent real-time page . . . 560
Listening to recorded calls using QM . . . 560
Creating stereo recordings . . . 561
Using AddQueueMember for dynamic agents . . . 562
Defining outbound queues (campaigns) . . . 563
Placing outbound calls . . . 563
Placing outbound calls through the AGI script . . . 564
Enabling ACD call attempts recording on Asterisk 1.0 and 1.2 . . . 565
Enabling ACD call attempts recording on Asterisk 1.4 . . . 566
Listening to live calls: Unattended Call Monitoring . . . 566
Enabling VNC Monitoring . . . 567
Enabling Agent's page actions . . . 567
Enabling XML-RPC call listening and streaming . . . 568
Enabling call outcomes . . . 569
Keeping the UNIQUEID of the call when setting status code . . . 570
Enabling call features . . . 571
Enabling pause codes . . . 572
Tracking required skills . . . 573
Closing ongoing calls . . . 573
Tracking DNIS and IVR information . . . 574
Adding TAG information to calls . . . 575
Enabling Hotdesking in the agent page . . . 575

How it works . . . 575
Example hotdesking configuration . . . 576
Changes to configuration properties . . . 576
Changes to extensions_queuemetrics.conf . . . 577
Running Asterisk 1.8 with QueueMetrics . . . 578
Handling Agents priorities on queues . . . 579
Using known numbers . . . 579
Configuring the AMI connection . . . 582
Listening to encrypted recordings . . . 583
What is a filter . . . 583
Setting up a filter . . . 583
Encrypting calls . . . 584
Tracking IVR data . . . 584
A call's life-cycle . . . 585
IVRs and QueueMetrics . . . 585
Using IVR areas . . . 586
Tagging calls by area in the QueueMetrics security system . . . 586
Using areas . . . 587
Implementing IVR tracking . . . 588
Dial-plan tracking . . . 588
Tracking areas . . . 589
Deprecated IVR tracking verbs . . . 589
The FreePBX module . . . 589
Appendix A: Security keys . . . 594
Appendix B: The [queuemetrics] context . . . 600
Appendix C: System preferences . . . 606
Defaults . . . 606
Short calls . . . 608
Maximum reporting period . . . 609
SMTP settings . . . 609
Call SLA . . . 610
Platforms . . . 610
Parser . . . 611
System administration . . . 613
Layout and behavior . . . 613
Database Access . . . 614
Realtime Page . . . 615
Wallboard . . . 616
Agent's Realtime Page . . . 616
Asterisk Interaction . . . 620
WombatDialer integration . . . 623

Real-Time Sounds . . . 623
Cluster configuration . . . 623
Audio Monitoring . . . 624
LLM . . . 624
QA . . . 625
WebRTC settings . . . 625
Misc . . . 626
Autoconfiguration Wizard . . . 627
Synchronizer . . . 627
External auth sources . . . 629
Single Sign-On (SSO) . . . 629
Assisted manual outbound (AMO) . . . 630
Obsolete parameters . . . 630
An example configuration . . . 631
Additional queue_log logs for QueueMetrics . . . 643
Basic queue_log structure . . . 643
Updating past calls . . . 643
Music-On-Hold Tracking . . . 643
Tracking IVR . . . 644
Adding DID/DNIS information . . . 644
Adding Tags to a call . . . 645
Adding Feature codes to a call . . . 645
Adding custom variables to a call . . . 646
Known variables . . . 646
Tracking Outbound calls . . . 646
Tracking pause reasons . . . 647
Adding the call status (outcome) . . . 647
Tracking Hotdesking events . . . 648
Tracking Skills . . . 648
Appendix E: Audit log records . . . 649
Action class: User lifecycle (10XX) . . . 649
Action: user logon - successful . . . 649
Action: user logoff . . . 649
Action: user logon - unsuccessful . . . 649
Action: password change . . . 650
SSO: OAuth consent granted . . . 650
SSO: OAuth consent failed . . . 650
SSO: OAuth profile failed . . . 650
SSO: User locked . . . 650
SSO: User not enabled . . . 651
SSO: New user created . . . 651

Action class: Key management (11XX) . . . 651
Action: key changed . . . 651
Action: key accessed via XML-RPC . . . 651
Action: AGAW key changed . . . 651
Action: AGAW key accessed via XML-RPC . . . 652
Action: AGAW restarted . . . 652
Action class: QueueLog editing (20XX) . . . 652
Action: QueueLog edited . . . 652
Action class: QA editing (21XX) . . . 652
Action: QA form deleted . . . 652
Action: Deletion of a comment . . . 652
Action: Deletion of all comments . . . 653
Action class: Recordings (22XX) . . . 653
Action: Media tag removed . . . 653
Action: Media tag added . . . 653
Action: MediaProxy - Media played . . . 653
Action: MediaProxy - Media failed . . . 653
Action: MediaProxy - Media denied . . . 653
Action class: Realtime agent management (23XX) . . . 653
Action: Realtime Agent Logon . . . 654
Action: Realtime Agent Logoff . . . 654
Action: Realtime Agent Pause . . . 654
Action: Realtime Agent Unpause . . . 654
Action: Realtime Agent SMS . . . 654
Action class: Realtime call management (24XX) . . . 654
Action: Call soft hangup . . . 654
Action: Call transfer . . . 655
Action: Call closure . . . 655
Action class: Engine audits (25XX) . . . 655
Action: Too many rows scanned . . . 655
Action: Report taking too long . . . 655
Action: License error . . . 655
Action: Out of memory error . . . 656
Action: Exceeded maximum reporting period . . . 656
Action: Report created . . . 656
Action: Report processed . . . 656
Action: Report Saved . . . 656
Action: Report Deleted . . . 656
Action class: Platform actions (26XX) . . . 656
Action: LOGIN . . . 657
Action: LOGOFF . . . 657

Action: ADDMEMBER . . . 657
Action: REMOVEMEMBER . . . 657
Action: PAUSE . . . 657
Action: UNPAUSE . . . 657
Action: OUTCOME . . . 657
Action: ADD FEATURE CODE . . . 658
Action: REMOVE FEATURE CODE . . . 658
Action: CUSTOM DIAL . . . 658
Action: SEND SMS . . . 658
Action: HANGUP . . . 658
Action: TRANSFER . . . 658
Action: MONITOR INBOUND CALL . . . 659
Action: MONITOR OUTBOUND CALL . . . 659
Action: SET VARIABLE . . . 659
Action class: Cron jobs (30XX) . . . 659
Action: Cronjob dispatcher started . . . 659
Action: Cronjob run successfully . . . 659
Action: Cronjob error . . . 659
Action: Export failed . . . 660
Action class: E-mail (31XX) . . . 660
Action: E-mail sent successfully . . . 660
Action: E-mail sending failed . . . 660
Action class: AMO events (32xx) . . . 660
Action: AMO list completed . . . 660
Action: AMO list reopened . . . 660
Action: AMO campaign completed . . . 661
Action: AMO campaign reopened . . . 661
Action: AMO number batch loaded . . . 661
Action: AMO list updated via API . . . 661
Action: AMO numbers recycled for list . . . 662
Action class: Configuration changes (4XXX) . . . 662
Action class: Wallboard events (50XX) . . . 664
Action: Wallboard opened . . . 664
Action: Wallboard saved . . . 664
Action: Wallboard deleted . . . 665
Action class: Chat events (51XX) . . . 665
Action: Chat deleted . . . 665
Action class: Testers (52XX) . . . 665
Action: AMI tester run . . . 665
Action: SMTP tester run . . . 665
Appendix F: The QueueMetrics watchdog page . . . 667

Accessing JMX counters . . . 667
Appendix G: Keyboard shortcuts . . . 669
Agent Page Shortcuts . . . 669
Softphone Keyboard Shortcuts . . . 669
Wallboard Shortcuts . . . 669
Appendix H: Glossary . . . 671
Appendix I: Removed and deprecated features . . . 673
Deprecated features . . . 673
Removed features . . . 673
Appendix J: Softphone Troubleshooting . . . 674
Common Issues . . . 674
The SoftPhone does not work with an extension configured for Zulu . . . 674
The SoftPhone does not register . . . 674
Outgoing Calls Hang up as soon as I click the Dial button . . . 676
Incompatible SDP . . . 676
Commmon issues when not using Nat (WebRTC clients are on same local network) . . . 677
Common issues when using NAT (WebRTC clients are NOT on same local network) . . . 677
SIP Failure code 503 . . . 677
One way audio . . . 677
Signs of a good ICE Connection . . . 677
Signs of a bad ICE Connection . . . 677

What is QueueMetrics?

The balancing of all internal elements that shape the ecosystem of an Asterisk PBX based call center relies on accuracy of measurements and precision of analytical reports. An efficient call center management can improve performances by 100% or more, while also significantly improving customers' satisfaction. Monitoring, reporting and improving performances is the bread and butter of QueueMetrics software.

The functional field of QueueMetrics call center suite divides into four major categories: Reporting, the Supervisor page, the Agent page and the Quality assessment.

Reporting

QueueMetrics collects Asterisk data and generates analytical reports for over 150 metrics, including:

- when did the call come in and who answers
- who initiates the call, process it and on which queue
- waiting time before the caller was connected to an agent and time needed to process the call
- number of attempts that caller has made before being connected
- how and why the call ends
- what number was dialed
- amount of successful and failed calls
- maximum, minimum, average and total waiting time and call processing time, as well as related data on service quality
- call breakdown by agents, by dialed numbers, transferred calls by internal numbers, by queues, by direction, by position number in a queue, by choices taken in IVR
- time on hold stats after connection, by agent, by country, region codes of calling and dialed subscribers
- starting and finishing times of agent sessions, time of entering or leaving the pause and the reason of it.

Supervision

With QueueMetrics a supervisor tracks data on queues, calls and agents' performance in real time, including:

- incoming calls with phone numbers, waiting times, conversation time and engaged agents
- agents online status (available, busy, on pause), time of last connection to a queue, last call and name of it
- add and remove agents from queues
- listen to agents conversations

- send instant messages to agents
- open and view an agent's desktop via remote access.

Agent page

QueueMetrics Icon agent page is a workstation where agents track incoming calls, connect to queues, disconnect for a break, and enter any information about calls using CRM and applications (contractor card, client reference, etc.), including:

- call starting, waiting and conversation time
- caller and internal number where the call was transferred.
- automatically or manually open a connected interface page in the local CRM system
- join or leave a queue, switch to pause mode, select a pause indicator (Lunch, Service Break, etc.), initiate an outgoing call, dial a number using a built-in soft-phone, as well as view his or her current status.

Quality Assessment

QueueMetrics allows for a full-scale automated quality control of agents' workflow and set of metrics definition for call grading. Supervisors create forms containing adjustable control criteria for subjective quality control, like for example:

- how friendly an agent was with a client
- how useful the agent's answers were
- agent follow up of corporate standards while processing a call
- amount and duration of processed calls
- number of missed calls
- call results

And many other criteria.

QueueMetrics is designed for call centers built on clustered servers and employing thousands of agents, easily integrating with all Asterisk and Freeswitch distributions (including FreePBX, FusionPBX, etc.) and is localized into more than 25 languages, including English, Spanish, Portuguese and Brazilian Portuguese, German, Arabic, Chinese, French, Dutch, Italian, Japanese, Hebrew, Polish, Slovak and Korean.

Installing QueueMetrics

QueueMetrics is able to operate in various server configurations in order to meet all your needs.

Depending on how big is your system you might want to have a single-server solution (PBX and QueueMetrics on the same server), a separated-server solution (PBX and QueueMetrics reside on two different servers) and a cluster solution (one QueueMetrics server monitoring many PBX servers).

QueueMetrics is a web application, so it runs on a server and is accessed through a web browser. There is nothing to install on clients; you just need to share the main URL where QueueMetrics is available.

QueueMetrics targets current "evergreen" browsers, so it works with Chrome, Firefox, Edge and Opera; IE 11+ and Safari 9.1+. It works on desktop systems, tablets and mobile devices.

The QueueMetrics server needs outbound HTTP connections to the Internet in order to work. If needed, it can be set up to use a proxy where direct connectivity is not possible.

The big picture

QueueMetrics bases all of its live and historical reporting on data coming from the PBX, namely its *queue_log* file.

It can be installed on the same server as the PBX (a single-server configuration) or on a separate server (a separated-server configuration); we recommend using a separate server, as it keeps QueueMetrics from competing with the PBX for resources and makes both systems easier to maintain and upgrade independently.

To send this data to QueueMetrics, you need to set up the Uniloader as well, on the same server as the PBX. The Uniloader reads the new data from the *queue_log* file and sends it to the QueueMetrics database, wherever it is; it does this without overloading the PBX server(s), and it's essential in a cluster-server configuration (also, it contains a wealth of tools for debugging/testing). We always suggest using Uniloader, whatever configuration you have.

Once QueueMetrics is installed, you will also want to set up connectivity between QueueMetrics and the PBX, so that QueueMetrics can send it commands (e.g. to log agents on and off the queues or place calls); see Configuring QueueMetrics.

Installing QueueMetrics

This section describes how to install QueueMetrics as a standalone application from a compressed archive, using the provided install script.

We provide a simple install script that covers the most common Linux distributions (RHEL/CentOS, Debian/Ubuntu, and more).

QueueMetrics is written in Java and distributed as a standalone executable JAR file, so it can

run almost anywhere - including Windows, macOS and other Linux distributions - as long as the following software is installed and available:

- A Java 25 JRE
- MySQL or MariaDB (any currently supported version)

We cannot cover every possible environment with the install script, but if you have a specific need that can't be solved another way, our Support team will be happy to help you.

Installing dependencies

Depending on your Linux distribution, you need to run different commands in order to install the required minimal dependencies.

Some older distributions do not ship Java 25 in their repositories. On those systems (marked *Adoptium* in the table below), you first need to add the Eclipse Adoptium repository, which provides the Temurin JDK packages, by following the instructions at https://adoptium.net/installation/linux. Then run the install command shown for your distribution.

Distribution	Package install command
Red Hat Enterprise Linux 9/10 and derivatives (AlmaLinux, Rocky Linux, Oracle Linux, CentOS 9/10)	`dnf install -y java-25-openjdk mariadb-server tar gzip nano` `systemctl enable --now mariadb`
CentOS 8/Rocky 8 (*Adoptium*)	Install Temurin as described above. ⚠ CentOS 8 is end-of-life and this setup is not expected to work. Consider moving to a newer operating system.

Debian 11/12 (*Adoptium*)	Add the Adoptium repository (see above), then:
	`apt-get update`
	`apt-get install -y temurin-25-jdk mariadb-server tar gzip nano`

Debian 13	`apt-get update` `apt-get install -y openjdk-25-jre-headless mariadb-server tar gzip nano`

Distribution	Package install command

Ubuntu 20.04 (*Adoptium*)	Same as Debian 11/12: add the Adoptium repository (see above), then:
	`apt-get update`
	`apt-get install -y temurin-25-jdk mariadb-server tar gzip nano`

Ubuntu 22.04/24.04/26.04 LTS	`apt-get update` `apt-get install -y openjdk-25-jdk-headless mariadb-server tar gzip nano`

SUSE Linux Enterprise Server 15 SP7 and 16.x	`zypper --non-interactive refresh`
	`zypper --non-interactive install java-25-openjdk-devel mariadb tar gzip nano`
	`systemctl enable --now mariadb`

Fedora 43/44	`dnf install -y java-25-openjdk-devel mariadb-server tar gzip nano` `systemctl enable --now mariadb`

Arch Linux	`pacman -Syu --noconfirm --needed jre25-openjdk mariadb tar gzip nano`
	`mariadb-install-db --user=mysql --basedir=/usr --datadir=/var/lib/mysql`
	`systemctl enable --now mariadb`

Gentoo	Make sure the system is using systemd:
	`eselect profile show`
	Then:
	`emerge --sync`
	`emerge --ask=n --noreplace dev-java/openjdk-bin:25 dev-db/mariadb app-arch/tar app-arch/gzip app-editors/nano`
	`emerge --config dev-db/mariadb`
	`systemctl enable --now mariadb`
	`eselect java-vm set system openjdk-bin-25`

Remember that the server must have internet access for license validation and API calls.

The default installer sets QM up with one gigabyte of heap memory, enough for all but the largest systems, but it can be reduced (see below) to fit on smaller servers.

Installation

Just run this one command as `root`:

```
curl -sL get.queuemetrics.com | bash
```

And... that's it!

The script performs the following steps:

- Verifies that all required system tools are present (Java, MySQL client, systemctl, etc.)
- Creates a `queuemetrics` system group and a `queuemetrics` service user
- Copies the application files to `/opt/loway/queuemetrics/`
- Creates directories for logs (`/opt/loway/queuemetrics/logs`) and thread dumps (`/opt/loway/queuemetrics/threaddumps`)
- Attempts to create the `queuemetrics` database automatically. If the script can connect to MySQL as root without a password, it creates the database and imports the initial schema.
- Installs the systemd service unit and the default configuration file at `/etc/default/queuemetrics` (existing configuration files are preserved)
- Starts and enables the QueueMetrics service
- Installs the `queuemetrics-cli` command-line tool to `/usr/bin` (see queuemetrics-cli)
- Prints the version that was just installed



If the script cannot connect to the database, it prints a warning and skips this step. In that case, you will need to create the database manually: import the provided SQL files from the `db/` directory, and update the JDBC connection string in `/etc/default/queuemetrics`.

Once the installation completes, open a browser and navigate to:

```
http://<your-server-ip>:8080/queuemetrics
```

Log in with the default credentials: username `demoadmin`, password `demo`.

After verifying that QueueMetrics is running, install Uniloader on the PBX server to begin uploading call data (see section Configuring QueueMetrics).

If the server does not seem to be up after a couple of seconds, type:

```
queuemetrics-cli status
```

To see what is happening.

Installing an exact version

The install script will automatically download and install the latest version for you. If you don't want that, you could download and install the exact version yourself. In any case, make sure you installed any reqred packages for your OS.

Start by going to https://downloads.loway.ch/software/queuemetrics/ and download the version you prefer from there.

Then, as root:

```
tar zxvf QueueMetrics-26.10.3-standalone.tar.gz
cd QueueMetrics-26.10.3
./install.sh
```

That's all you have to do.

Adding a license

QueueMetrics ships with an embedded demo license that lets you activate and try the product immediately. To install a full license key:

- You can use the page Install new license key, or
- You can edit the configuration file:

```
sudo nano /etc/default/queuemetrics
```

Set the `QM_LICENSE_KEY` parameter to the license key you received:

```
QM_LICENSE_KEY="your-license-key-here"
```

Restart the service for the change to take effect:

```
queuemetrics-cli restart
```

Further configuration

All configuration for QueueMetrics standalone is managed through a single file:

```
/etc/default/queuemetrics
```

This file controls network settings, database connection, logging, Java memory allocation, and other

runtime options. The file is well-commented and each parameter is explained inline.

The most commonly adjusted settings are:

- `QM_PORT` — the TCP port the server listens on (default: 8080)
- `QM_JDBC_URI` — the JDBC connection string for the database
- `QM_JAVA_MEM` — JVM heap size (default: 1 GB)
- `QM_LOG_LEVEL` — log verbosity (default: SEVERE)

After changing any parameter, restart the service:

```
queuemetrics-cli restart
```

Internet access

QueueMetrics needs to access the internet to validate its licenses, otherwise it won't be able to start. If the server does not have direct outgoing internet connectivity, you need to set up a proxy - see section Proxy.

Setting up HTTPS

QueueMetrics standalone listens on plain HTTP. To serve QueueMetrics over HTTPS, place a reverse proxy in front of it. We recommend using Nginx or Caddy. For a detailed proxy configuration guide, see Setting up an Nginx reverse proxy.

When running behind a reverse proxy, change the listen address in `/etc/default/queuemetrics` so that QueueMetrics only accepts local connections:

```
QM_ADDRESS="127.0.0.1"
```

For extra protection from Cross-Site Request Forgery (CSRF) attacks, you may also want to "lock" QueueMetrics' CSRF protection to the exact hostname(s) you are using, by explicitly setting:

```
QM_SAFE_HOSTS="https://queuemetrics.my-server.com:443"
```



If you don't do this, the valid host name will be computed out of the one QM is receiving traffic on. This usually works fine, unless your reverse proxy allows arbitrary headers in, or you do not use one at all.

Logging

When logging is enabled (by setting `QM_LOG_LEVEL` to a value other than `OFF`), log files are stored in `/opt/loway/queuemetrics/logs`.

Thread dumps, when generated, are stored in `/opt/loway/queuemetrics/threaddumps`.

JMX monitoring

JMX remote monitoring can be enabled by uncommenting the `QM_JAVA_JMX` line in the configuration file. This opens ports 30000 and 30001 for tools such as VisualVM or JConsole to connect over an SSH tunnel.

For details, see Monitoring QueueMetrics with JMX.

A simpler way to track JMX counters can be found in JMX Counters.

Installing in Secure Mode

QueueMetrics can be installed in "secure mode" by setting a configuration property `layout.secureMode=true`. This mode is meant for systems that are to be exposed to the internet or accessible to a wider audience, and implement a number of security mitigations. See Configuring system preferences.

Under this mode:

- Security enumeration of users is disabled. All failed logins expose the same error message, as not to give away whether a user exists and the password is wrong, or the user is locked, or non existent.
- HTTP session-ids appear on the system log viewer in an abridged format, e.g. session cookie"80EFB4F8317A80CBF59F17A2D1888407" will appear as "80EFB4...888407", as not to make it possible to retrieve usable session IDs.
- Verbose error messages are disabled. The only message displayed is a `Reference-id: XA6T23T` that can be used to find the actual error on system logs.
- The login page is de-branded. This forces the login page just to display the log-on form and the welcome message you set. No logos, product versions or external links will be shown. This will happen on all pages if the user is not logged on. (as per ISO 27001 section 11.5.1)



When running in Secure mode, you may want to make sure that the properties `url.qm` and `url.rss` are set, so that host header attacks are not possible. If they are not, those are inferred using a series of heuristics to detect front proxies that may get a crafty attacker to override the default settings. If you run a proxy in front of QM, make sure extra headers are stripped.

Upgrading

To upgrade QueueMetrics, download the new version archive and repeat the installation procedure described above. Make a backup before upgrading (see Backing up).

The install script preserves your existing configuration file at `/etc/default/queuemetrics`. The database schema is updated automatically on the first run of the new version.

The database update utility

QueueMetrics ships with a utility that makes it very easy to check and 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, as explained in Backing up!

You can access it directly pointing your browser to `http://127.0.0.1:8080/queuemetrics/dbtest/`



Once you access the DB updater, it will check and 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.

From this very page, you can also check a number of system properties through the so-called DbTest Diagnostic Tools:

- The current QueueMetrics configuration properties settings
- The Java environment used
- The Java memory and CPU settings
- Whether the AMI connection to the Asterisk is working
- The current PBX configuration
- The current MySQL storage data (with search ability)

For further details, see the complete description of the DBTEST module.



As this page lets you access the inner configuration of QueueMetrics, it should be turned off on publicly accessed systems. This can be obtained by setting a configuration parameter as detailed on the DBTEST page itself.

Backing up

To create a complete backup of your QueueMetrics installation:

1. Dump the `queuemetrics` database:

```
mysqldump queuemetrics > queuemetrics-backup.sql
```

2. Copy the configuration file:

```
cp /etc/default/queuemetrics queuemetrics-defaults-backup
```

Store both files in a safe location. These two items are sufficient to restore your QueueMetrics installation on a new server.

A more detailed explanation is available at Backing up QueueMetrics.

Using queuemetrics-cli

The `queuemetrics-cli` tool is a convenience command-line utility installed to `/usr/bin` during setup. It provides quick access to common service management tasks, so you don't have to remember them and will ask for `sudo` privileges when needed.

Usage:

```
queuemetrics-cli <command>
```

Available commands:

Command	Description
`restart`	Restart the QueueMetrics service
`stop`	Stop the QueueMetrics service
`status`	Show the current status of the QueueMetrics service
`threaddump`	Generate a thread dump and class histogram, saved to `/opt/loway/queuemetrics/threaddumps`
`version`	Display the installed QueueMetrics version
`dbupdater`	Run the database schema updater manually

Examples:

```
# Check if the service is running
queuemetrics-cli status

# Show the installed version
queuemetrics-cli version

# Generate a thread dump for troubleshooting
queuemetrics-cli threaddump
```

Updating from a RPM-based distribution

Suppose you are updating a system that was originally installed from the RPM package (via `yum`), and that this system:

- has a license installed
- has some customizations (e.g. changes to the amount of usable memory, uses a databse on a different server, etc.)

Moving to the standalone version requires following the procedure below. The advantage of the standalone version is that it does not depend on a specific package manager and works on any modern Linux distribution.



If you have WombatDialer running on the same server, we strongly suggest that you move the new QueueMetrics to a different server, as the new standalone settings might conflict with the existing ones (e.g. using the same HTTP port, using incompatible versions of Java, etc.).

Collecting information from the old installation



This procedure describes the general case, and only goes so far: we have no way of knowing what customizations were made to your specific installation over time. It is your responsibility to review your system and identify any changes not covered here before proceeding.

First, retrieve the old license key:

```
grep LICENZA_ARCHITETTURA /usr/local/queuemetrics/qm-current/WEB-INF/tpf.properties
```

Look at the last line of the output: this code will be needed to install your license on the new system. Copy it somewhere; you'll need it to activate

If QM connects to a remote database, you also need to recover its connection credentials:

```
grep JDBC_URL /usr/local/queuemetrics/qm-current/WEB-INF/tpf.properties
```

To check whether any configuration changes were made (e.g. the amount of usable memory, or a specific time zone), run:

```
cat /etc/sysconfig/qm-tomcat6
```

and check for any changes (they are usually added to `JAVA_MEM`). Note down anything you find, as you will need to reapply it after the upgrade.

Decommissioning the old system

Stop the running service, make a safety backup of all data, and uninstall the original package. This is also a good time to ask yourself whether it is actually worth upgrading the existing system in place, or whether it would be better to also replace the operating system with a fresh install on a different server - for example, because the current OS is too old (e.g. CentOS 8).

Either way, the procedure is very similar.

First, take a full dump of the MySQL database. It is important to understand what is actually being decommissioned here: it is the old **server** (or, when reinstalling in place, the old **package**) that goes away - not the database itself, whose contents you want to carry over intact to the new installation. Strictly speaking a dump is not required when reinstalling on the same server (see below), but it costs little and is a safety net in case something goes wrong; it becomes essential if you are moving to a different server, since in that case the old server - database included - will no longer be available. Note that this may require a fair amount of disk space.

```
mysqldump queuemetrics | gzip > queuemetrics.sql.gz
```

Then copy the files collected earlier (license and configuration):

```
cp /usr/local/queuemetrics/qm-current/WEB-INF/tpf.properties .
cp /etc/sysconfig/qm-tomcat6 .
```

At this point, remove the `queuemetrics` package, which will also remove the database. Since the files that hold the database contents keep the same format and are already in the right place, reinstalling should just work. If it does not, drop the database and reload it from the backup you just made.

```
service queuemetrics stop
yum remove queuemetrics
```

Reinstalling on the same server

You can now install the new required packages:

```
dnf install -y java-25-openjdk mariadb-server tar gzip nano

curl -sL get.queuemetrics.com | bash
```

Now let's set the new license key and reapply any Java environment customizations noted earlier:

```
nano /etc/default/queuemetrics
```

and edit the properties you need:

```
QM_LICENSE_KEY="....."
QM_JDBC_URI="...."
QM_JAVA_MEM="-Xms1G -Xmx1G"
```



You should find the license key already installed - just check that it matches the one you have saved.

Once done, restart QueueMetrics and then check that it is up:

```
queuemetrics-cli restart

queuemetrics-cli status
```

As soon as you connect to the QueueMetrics webapp, the database upgrade wizard will appear - follow it to completion, and QM will be ready to go.

If you access local recordings, make sure you grant appropriate file permissions.

Installing on a different server

```
dnf install -y java-25-openjdk mariadb-server tar

curl -sL get.queuemetrics.com | bash
```

The installation will create the default database; overwrite it with your backup:

```
zcat queuemetrics.sql.gz | mysql queuemetrics
```

Apply the environment customizations described above, then restart QueueMetrics.

Installing QueueMetrics using Docker

Docker is an infrastructure-management tool that lets you run software "as-if" it was running on a dedicated VM, without the administrative and technical overhead of running a separate VM.

Docker makes your life very easy if you need to manage multiple copies of QueueMetrics - you just run them. Docker makes it very easy to install QueueMetrics, no matter what the underlying OS is - the Docker image contains QueueMetrics with all its dependencies, a MySQL server and all the needed tooling.

And when you need to upgrade, you just terminate the old instance and start a new one pointing to the same data - it just works.



Even if Docker makes your life easier, you need to familiarize yourself with it before you consider running a production service on it.

Installing the official QueueMetrics Docker image

Create a data container

The first thing you have to do is to create a place to store persistent data - this might be a directory on your disk or what Docker calls a "persistent data-only volume".

The easiest way to get started is to use a local directory. Let's say you create a directory called */opt/qm1data* for QM to store its data in.

Start the QueueMetrics image

You can easily start QueueMetrics with the syntax:

```
docker run --name=QM1 --volume=/opt/qm1data:/data -P -d loway/queuemetrics
```

If you do, Docker will download the current version of QueueMetrics from Docker Hub and will run it. The whole process typically takes only a few seconds.

When done, you can issue a "docker ps" to see what is happening:

```
# docker ps
CONTAINER ID        IMAGE                 COMMAND             CREATED
STATUS              PORTS                      NAMES
7c2920e8e28f        loway/queuemetrics    "/ww/run"           12 seconds ago      Up 10
seconds       0.0.0.0:32770->8080/tcp   QM1
```

This means that if you connect to your server as *http://server:32770/queuemetrics* (as written under the PORTS column) you will access your new QueueMetrics instance. You can see that the MySQL database and QueueMetrics configuration and license are visible under */opt/qm1data*.

If you want, you can run multiple instances of QueueMetrics on the same machine:

```
# docker ps
CONTAINER ID        IMAGE                 COMMAND             CREATED
STATUS              PORTS                      NAMES
7c2920e8e28f        loway/queuemetrics    "/ww/run"           12 seconds ago      Up 10
seconds       0.0.0.0:32770->8080/tcp   QM1
c152da318fe2        loway/queuemetrics    "/ww/run"           17 seconds ago      Up 16
seconds       0.0.0.0:32769->8080/tcp   QM3
6fef35c26c6d        loway/queuemetrics    "/ww/run"           48 seconds ago      Up 46
seconds       0.0.0.0:32768->8080/tcp   QM2
```

as long as each of them points to a separate data volume and has a proper license key.



If you plan on running only one instance of QM, it would be easier to set the same port on the host and on the container - see Using fixed ports.

Accessing an instance using the shell

You can enter the QM1 instance by running:

```
docker exec -it QM1 bash
```

This will run a bash shell, and you can do whatever you want - install RPM packages, inspect the system, check error logs, make backups using `mysqldump` (see Backing up).



Keep in mind that all data that is not stored under */data* is stored in the main image, and the main image will be removed on restart/upgrades.

Terminating an instance

To terminate one instance:

```
docker stop QM1
docker rm QM1
```

Everything will be removed, but the contents of your data folder.

Using a data-only volume

If you prefer, you can run QueueMetrics using a named data-only volume:

```
docker run --name=MYQM loway/data true
docker run --volumes-from MYQM -P -d loway/queuemetrics
```

This is very similar to using the local directory, though it makes your life harder to just check what is on the volume.

If you do not start QueueMetrics with a data container, it will create one within the current disk image - but when you throw away the image, all data created will be lost.

Applying a license

When running QueueMetrics under Docker, the easiest way is to use the Install new license key page.

If you prefer, the license key can be passed in from the outside as an environment variable, using the `-e` flag of `docker run`, when starting the container:

```
docker run -e QM_LICENSE_KEY="your-license-key-here" --name=QM1
--volume=/opt/qm1data:/data -P -d loway/queuemetrics
```

The license key only needs to be specified once: it is written to the instance's persistent data, so it

does not need to be passed again on later restarts.

For the full list of environment variables that can be passed to a QueueMetrics Docker instance, see Configuring license, memory, time zones and more.

Upgrading QueueMetrics

In order to do an upgrade, you must have started the original instance using a data volume or a shared folder.

If you did, you first download the new QueueMetrics image, then you stop (and remove) the current version and then run the image again using the same data folder:

```
docker pull loway/queuemetrics:latest
docker stop QM1
docker rm QM1
docker run --name=QM1 --volume=/opt/qm1data:/data -P -d loway/queuemetrics
```

At this point, it will just work.

Please note that system logs, the compilation cache and everything else that is not in your persistent data folder will be deleted.

Tips and tricks

Defining a fixed QM port

If you want an instance to use a fixed port (e.g. because you have a proxy in front), you can easily do that using the syntax:

```
docker run --name=QM1 -p 12345:8080 -P -d loway/queuemetrics
```

This will use local port 12345 and redirect it to the instance's port 8080.

It is of course very common to use the host's port 8080 as the public access port for QM, as in:

```
docker run --name=QM1 -p 8080:8080 -P -d loway/queuemetrics
```

Note that to use a fixed port on the host, it must be available - you cannot run two instances on the same host port.

Configuring license, memory, time zones and more

The default QueueMetrics instance runs with just 128M of Java heap - not enough to run a large production server. Environment variables can be passed to `docker run` (using the `-e` flag) to configure the instance without editing any file inside the container; they are the same `QM_*` settings used by the standalone installation (see Further configuration). For example, to increase the JVM

heap size:

```
docker run -e QM_JAVA_MEM="-Xms1G -Xmx1G" --name=QM1 --volume=/opt/qm1data:/data -P -d
loway/queuemetrics
```

Multiple variables can be passed by repeating the `-e` flag. As with the license key, these values are only needed on first start: once written to the instance's persistent data, they are reused automatically on every subsequent restart.

The following variables are supported:

Variable	Value	Notes
`QM_LICENSE_KEY`	license key string	The QueueMetrics license key. Leave empty to run in free/demo mode. See Adding a license.
`QM_ADDRESS`	IP address	The address the server listens on. Default: `0.0.0.0`.
`QM_PORT`	TCP port	The port the server listens on. Default: `8080`.
`QM_JDBC_URI`	JDBC connection string	Connection string for the QueueMetrics database (host, database name, user, password and connection parameters).
`QM_LOG_DIR`	directory path	Directory where QueueMetrics writes its log files. Default: `./logs`.
`QM_LOG_LEVEL`	log level	Log verbosity, from least to most verbose: `OFF`, `SEVERE`, `WARNING`, `INFO`, `CONFIG`, `FINE`, `ALL`. Default: `SEVERE`.
`QM_JAVA_MEM`	JVM heap options	JVM heap memory allocation, e.g. `-Xms1G -Xmx1G`. Increase for large installations or if you experience OutOfMemory errors.
`QM_JAVA_OPTS`	JVM options	Additional JVM options for performance and behavior tuning. The defaults are sensible and rarely need to be changed.
`QM_JAVA_TIMEZONE`	JVM system property	Forces the JVM to use a specific IANA time zone (e.g. `-Duser.timezone=Europe/Zurich`), overriding the system default.
`QM_JAVA_JMX`	JVM options	Enables JMX remote monitoring, opening ports 30000/30001 for tools such as VisualVM or JConsole. See JMX Counters.
`QM_JAVA_PROXY`	JVM proxy options	Configures an HTTP/HTTPS or SOCKS proxy for outbound connections (license validation, API calls). See Using an HTTP/S proxy.

Configuring QueueMetrics properties

You can configure a QueueMetrics instance in many ways:

- Accessing it through your browser and configuring it
- Editing the contents of the data directory
- Using the QueueMetrics JSON services to create users and queues

Though the third option is the most practical if you need to deploy a large number of instances, all three options are quite feasible.

Installing from RPM

The historical RPM-based installation of QueueMetrics is no longer supported. This is because many PBX distributions are no longer based on RHEL and its derivatives, and moving to a single standalone install script lets us support a much wider range of operating systems with the same procedure.

- If you are installing QueueMetrics on a new system, see Installing QueueMetrics.
- If you are updating an existing RPM-based installation, see Updating from a RPM-based distribution.

Configuring QueueMetrics

Now that you have got QueueMetrics running you need to configure it in order to retrieve data from the PBX.

Uniloader configuration

On the PBX system, edit the file `/etc/sysconfig/uniloader`; you need to set the following values in order to tell Uniloader to work with the local QueueMetrics database:

```
URI="mysql:tcp(127.0.0.1:3306)/queuemetrics?allowOldPasswords=1"
LOGIN=queuemetrics
PASS=javadude
TOKEN=P001
```

If you're running a PBX server cluster you also have to modify the TOKEN value, giving to each of them a different partition name (usually P001, P002, P003, etc). This is fundamental to avoid any possible concurrent writing issue.



Each PBX server's Uniloader MUST use a different partition.

Now restart Uniloader:

```
service uniloader restart
```

If Uniloader is on a different machine than QueueMetrics (and MySQL) you will need to create a new database user able to connect remotely.

Log in where MySQL is as *root*:

```
# mysql -uroot -p
```

One you're in create a new remote user and give it all grants on the *queuemetrics* database:

```
mysql> CREATE USER 'queuemetrics'@'%' IDENTIFIED BY 'javadude';
mysql> GRANT ALL PRIVILEGES ON queuemetrics . * TO 'queuemetrics'@'%';
mysql> FLUSH PRIVILEGES;
```

and then set the correct database in the `URI` parameter of Uniloader.



Uniloader is very flexible and has multiple data upload modes, plus it can perform many functions. See its manual at https://docs.loway.ch/Uniloader/index.html

QueueMetrics configuration

If you have a single PBX server and have not modified the TOKEN value in the Uniloader configuration add (or, if exists, modify), in the configuration properties, the following values:

```
default.queue_log_file=sql:P001
callfile.dir=tcp:admin:password@ip-address
        # user:pass are the credentials for the Asterisk's AMI interface.
        # You might have to configure a user in each of the
        # Asterisk servers' /etc/manager.conf file.
        # serverhost is the IP address of that server.
```

QueueMetrics is now already able to retrieve information from the PBX.
Instead, if you have a server cluster add or modify these values as follows:

```
default.queue_log_file=cluster:*

# Put here the server names (A, B, C, etc could be a choice) separated
# by a pipe.
cluster.servers=A|B|C

# For each of them put these values, replacing 'servername' with
# the name chosen above
cluster.servername.manager=tcp:user:pass@serverhost
        # user:pass are the credentials for the Asterisk's AMI interface.
        # You might have to configure a user in each of the
        # Asterisk servers' /etc/manager.conf file.
        # serverhost is the IP address of that server.
```

```
cluster.servername.queuelog=sql:partition-name
        # partition-name is the value of PARTITION in the Uniloader's
        # configuration file in that server.
```

Include QueueMetrics dialplan in Asterisk

Now, to enable QueueMetrics to place calls and log the agents on and off the queues you need to include the QueueMetrics dialplan in Asterisk. If you installed QueueMetrics with *yum*, you can find it in **/usr/local/queuemetrics/qm-current/WEB-INF/mysql-utils/extensions-examples/extensions_queuemetrics_18.conf**.

Copy the file under **/etc/asterisk/** then add the following line at the end of your **extensions.conf**, or **extensions_custom.conf** if your PBX has this file:

```
#include extensions_queuemetrics_18.conf
```

Then open the Asterisk CLI and reload the dialplan:

```
Asterisk CLI> dialplan reload
```

Granting access to local recordings

QueueMetrics runs as the `queuemetrics` user (group `queuemetrics`), while call recordings on the PBX are usually owned by the PBX's own user and group; QueueMetrics cannot read them by default, and click-and-listen will fail.

The simplest fix is to add the `queuemetrics` user to the PBX's group, so that it inherits read access to the recordings without changing any file permission.

The example below is for Asterisk, but the same reasoning applies to any PBX, including FreeSWITCH: identify the user/group the PBX process runs as, and the group that owns the recordings directory, then add `queuemetrics` to that group.

To find the right group, run the following diagnostic commands and compare their output:

```
echo "* Directory permissions:"; ls -ld /var/spool/asterisk/monitor
echo "* Asterisk process:"; ps -eo user,group,cmd | grep '[a]sterisk -'
echo "* asterisk.conf config:"; grep -E '^(runuser|rungroup)'
/etc/asterisk/asterisk.conf
```

If all three outputs agree on the same group, that is the group to use. If they disagree, trust the value in `asterisk.conf`, as that is what actually determines the group the process will run as on the next restart.

Once you know the group, add `queuemetrics` to it:

```
usermod -aG <group> queuemetrics
```

Group membership changes only apply to new processes, so restart QueueMetrics for the change to take effect:

```
queuemetrics-cli restart
```



The same issue (and solution) applies if you read the `queue_log` file directly.

Does it work?

QueueMetrics has some test tools to check if all is running the right way and for troubleshooting.
The System Diagnostic Tools page offers a number of tools that check database and AMI connections and more (see Troubleshooting).

To check that everything works you can open your browser and point it to *http://hostname:8080/queuemetrics*, the first time you run it you might get an error, it will disappear in a few seconds and QueueMetrics will guide you through the database configuration; probably you'll just need to confirm without making any change.
After that you will be able to log in with the default username *demoadmin* and password *demo*.
Obviously, at the moment, no data is shown.

Troubleshooting

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 about 80,000 calls and 40,000 agent events with a standard 64 megabyte 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. See Further Configuration.

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. Therefore, it's a good idea to perform a scheduled restart of the servlet container, to avoid possible memory leaks and to reclaim now-unused memory to the main pool.

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 100-200 megabytes are not unheard of, depending on the Java Virtual Machine in use.

A more in-depth discussion of QueueMetrics memory usage and live monitoring can be found on the "Advanced Configuration Manual", chapter "Tuning QueueMetrics memory settings".

A real-life example for a large call-center with about 400 agents logged on and running on a dedicated server may be like:

```
-Xms4G -Xmx4G -server -XX:+UseG1GC -XX:MaxGCPauseMillis=500
-XX:InitiatingHeapOccupancyPercent=45
```

But the actual optimal configuration on a large system should be determined by running the system for a while under the required workload and monitoring the actual resource consumption.

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 A: Security keys.

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 a 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 Editing QueueMetrics settings.

A list of default users provided with the standard QM installation and their default passwords can be found in Default users.

Just to be on the safe side, QueueMetrics keeps an *Audit Log* of all activities that may have security implications; see The Audit log for more details.

QueueMetrics is not available

Often this is a firewall problem. First of all, check if QueueMetrics is running:

```
lsof -i -P | grep 8080
```

If it returns something then QueueMetrics is running; if so, do try temporarily stopping the firewall:

```
service iptables stop
```

Now retry pointing your browser to *http://hostname:8080/queuemetrics*.

If it works now, you should setup your firewall in order to accept incoming connections to port 8080.

Editing configuration settings

See Configuring system preferences.

The System Diagnostic Tools

QueueMetrics has some tools that help you to solve most of the possible issues.
You can find them clicking on System Diagnostic Tools or pointing your browser to *http://hostname:8080/queuemetrics/dbtest/*.

- *View configuration*: this page lets you know what configuration properties are used by QueueMetrics and the JVM.
- *AMI tester*: it's useful to test to have a working AMI connection with the PBX server(s).
- *Live DB Inspector*: use it if you want to check QueueMetrics is receiving the correct data from the PBX server(s).
- *RAM caching*: it allows you to look at the QueueMetrics caching stats and to clean the cache when necessary.

You can have a full description in chapter Using the DbTest Diagnostic Tools.

Using an HTTP/S proxy

QueueMetrics needs to establish direct HTTP outbound connections:

- to validate the current license.
- to access external services (e.g. JSON audio searching, external user auth, OAuth authentication)

If such connections are unavailable, then QM will not even start up (the Java process will start, but the GUI won't become available).

As in some corporate environment, direct outbound connections are not possible, so you need to use an intermediate proxy.

The JVM has a standard way to set up HTTP proxies, by setting the JVM properties:

```
-Dhttp.proxyHost=10.10.1.1 -Dhttp.proxyPort=8888
-Dhttps.proxyHost=10.10.1.1 -Dhttps.proxyPort=8888
```

In the example above, you tell QM to use a proxy located at address `10.10.1.1` on port 8888 that handles both HTTP and HTTPS traffic.

Or, to set up a SOCKS proxy:

```
-DsocksProxyHost=1.2.3.4 -DsocksProxyPort=63080
```

You will need both HTTP and HTTPS proxy access, as license validation happens over HTTPS.



These properties are ready to be edited at Further Configuration.

To check whether connectivity is working correctly, you can use a script called `checkLicenseConnectivity.sh` that can be found under `WEB-INF/README/`.

For example, to check connectivity with no proxy, you would simply run:

```
# ./checkLicenseConnectivity.sh
============================================================
  Attempting connection to the Loway licensing servers
============================================================

No proxy configuration settings
============================================================
   All went well.
============================================================
```

While to attempt using a SOCKS proxy, you could for example try:

```
# ./checkLicenseConnectivity.sh -DsocksProxyHost=127.0.0.1 -DsocksProxyPort=63080
============================================================
  Attempting connection to the Loway licensing servers
============================================================

Proxy configuration:

 |Property      |Value    |
-+--------------+---------|
0|socksProxyHost|127.0.0.1|
1|socksProxyPort|63080    |


Feb 04, 2022 9:21:44 AM it.loway.tpf.web.LowayHttpClient connect
SEVERE: Error accessing license server
java.net.SocketException: Connection refused (Connection refused)
    at java.net.SocksSocketImpl.connect(SocksSocketImpl.java:435)

============================================================
  Connection failed:
```

```
Error accessing HTTP URL
============================================================
```

Version numbering scheme

Since January 2012, QueueMetrics uses a numbering system that is based on when a major release is built. The version is then year plus the month of the release. So 12.01.1 is the first in the family that was released in January 2012. See how easy that is? It is also nice because then you know exactly how old or new the version you are dealing with is.

Older versions of QueueMetrics had a "classical" numbering scheme, as per 1.2.3. There is no change on the licensing keys or anything else from the old to the new numbering scheme.

Backing up QueueMetrics

The current state of a working QueueMetrics system is held in two places:

- its *configuration files*, containing:
  - the current license and database connection (stored in file `/etc/default/queuemetrics`)
- its *database* (usually named `queuemetrics`), containing everything else, including call data and audit logs.

Therefore, in order to back up a running system, you need to:

- back up the original install file (e.g. RPM file, TGZ, Docker image or whatever you used to install). Recent versions of QueueMetrics are always available from Loway, but older versions are removed. You may also simply back up the whole webapp, that will contain all of QueueMetrics plus its configurations.
- back up the configuration files, and
- back up the database

In order to back up the database, you may simply run:

```
mysqldump queuemetrics | gzip  > queuemetrics.sql.gz
```

to create a compressed database dump, and:

```
gunzip --to-stdout queuemetrics.sql.gz | mysql queuemetrics
```

to reinstall it.

As the database might be very large, before you attempt a backup, you must be sure that you have enough disk space to store it in full. The Live Inspector page allows you to see how large the database is on disk, and estimates how large its compressed data might be.



If space is at premium on the QueueMetrics server, you may perform a full backup from a different server by telling `mysqldump` to connect remotely, and using `scp` (or a similar tool) to copy files.

Fixing damaged indexes

Sometimes, the database will refuse to use an existing index. This may happen if the index file is broken (e.g. full disk, unclean shutdown, crashed tables) - though sometimes this happens for no apparent reason.

You will notice that operations take a lot more to complete - a report that took a second or two starts taking a few minutes to run - and that all the load in on MySQL/MariaDB and/or disk access.

The easiest way to solve the issue, if you are not a MySQL/MariaDB expert, is to:

- Stop QueueMetrics and Uniloader
- Make a database backup
- Drop and create the database (existing permissions will be kept, so you don't have to create them again)
- Import the backup - data will be inserted from scratch, and all indexes will be rebuilt
- Restart QueueMetrics and Uniloader

E.g.

```
service uniloader stop
service queuemetrics stop

mysqldump queuemetrics | gzip  > queuemetrics.sql.gz
echo "drop database queuemetrics; create database queuemetrics;" | mysql
gunzip --to-stdout queuemetrics.sql.gz | mysql queuemetrics

service queuemetrics start
service uniloader start
```

Make sure that you have enough disk space for the backup to complete successfully, and run this off-hours, as a very large database might take hours to backup and rebuild.

License keys

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 sends you the temporary or official key, you can install it either through the graphical interface or manually through a shell.

What about licenses?

QueueMetrics is licensed based on:

- the number of agents it needs to track
- the number of PBX servers you want to monitor
- a specific duration

You need an **agent** for each separate agent entity you want to keep track of - so if you have two shifts of 20 persons each, you need 40 agents total, no matter how many of them are logged in at once. If your agents change over time and you want to keep track of both present and past ones each using a different code for each, you will have to factor that in as well. So, for most call centers, you will need something that matches the number of paychecks you write every month for call-center agents.

QueueMetrics natively tracks only calls on queues, so any entity that is not answering a queue (or taking part in an outbound campaigns) is not counted. If you use automated outbound tracking, all phones will be tracked and so every one of them will be counted as one agent.

You will need a **cluster** license if you want to track multiple PBXs at once - whether this happens because you want to view multiple PBXs from the same GUI (e.g. different branches, or different countries) or because you have a high-availability system where calls may be processed in multiple locations. Vanilla QM licenses let you track only one PBX, but it's possible to have more when ordering the license. See Monitoring clusters with QueueMetrics for further information on clusters.

Each license has a specified **expiry date**, that is set at creation and cannot be altered. After a license is expired, the system displays an error message and becomes unavailable. Before this happens, it displays a message to remind you when the license is almost expired.

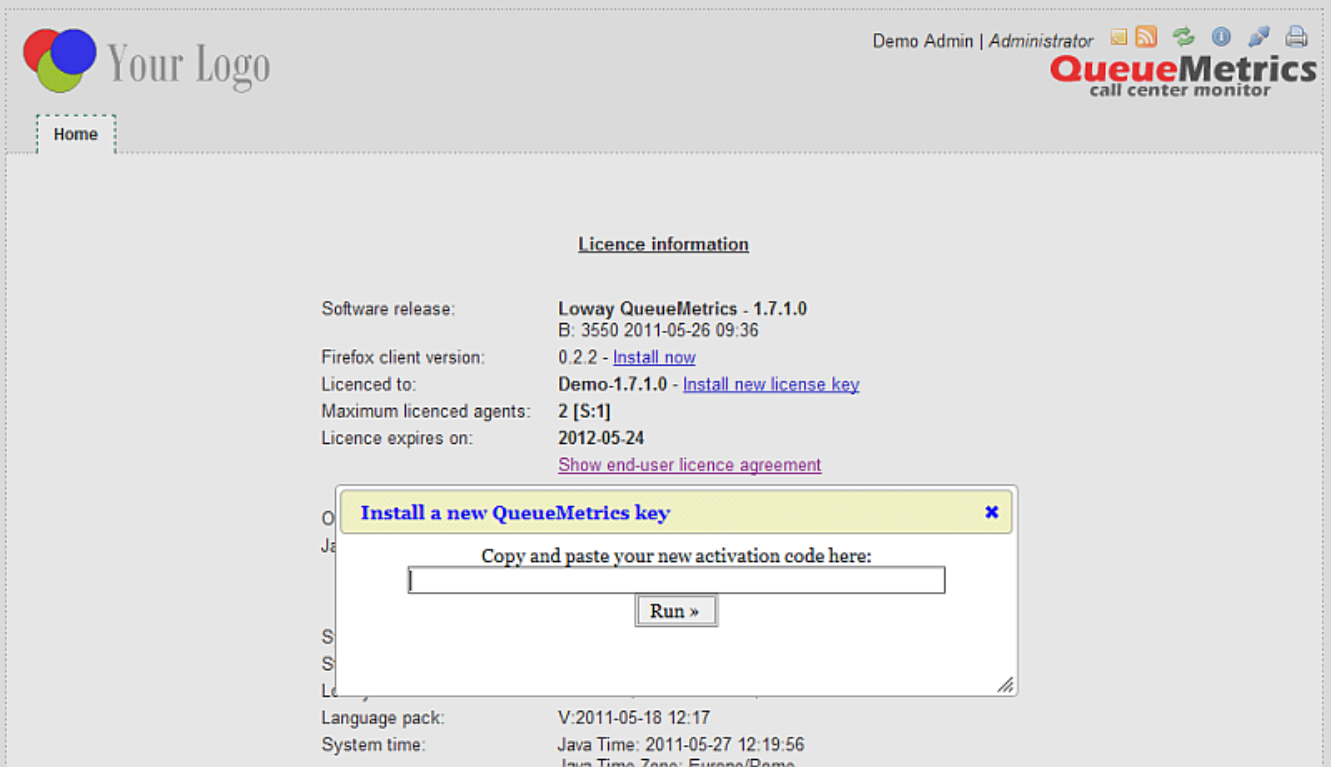
If you need to install multiple QM systems, you will need to order a separate key for each system. For the common case where you have a stand-by system for **high availability**, you may order it together with its master key and it has a special price.



If you use a key on more than one system at once, the key may be blocked, the systems will refuse to start and you may incur in a reactivation fee. So don't do it. QueueMetrics systems need to access the Internet for key validation and will not start if the network is unreachable.

Installing a new key through the GUI

Log on to QueueMetrics as "demoadmin" and click on the License page, if you have the correct grants you should see a label called "Install new license key"; click on it.



Copy the activation code you received by e-mail into the license box and press "Install". The system will restart in a few seconds (you may see a blank page - if you do, just try and reload). Log off an on again. On the License page you should see the new key.

As an alternative, you can set the license in the configuration file. The former method (GUI) is easier if you manage the system yourself; the latter is better if you manage a fleet of QueueMetrics servers. To install a license through the configuration file:

- If you installed a standalone version, see Adding a license
- If you are running a Docker image, see Applying a license

What happens if you install two licenses, one from the GUI and one in the configuration?

QueueMetrics tries both licenses in sequence - first the last one installed through the GUI (if present), then the one set in the configuration file. The first of them that is active is then used.

If you want to use the one set in the configuration file, but already entered one though the GUI, you can look up the property `qm.license` in the preferences editor and set it to an empty string.

License expiration notification

QueueMetrics will notify users on the Home Page when the license is about to expire; this helps preventing downtime in case a license actually expires without renewal.



It is possible to turn off this additional notification by setting a configuration property.

Monitoring clusters with QueueMetrics

QueueMetrics is able to monitor clusters of PBX 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 PBX 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 PBX servers, without special optimizations or weird configurations
- There is less risk of a deadlock on one single PBX 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 PBX 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 PBX box, or
- The same queues on more than one PBX 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 PBX 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 PBX 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.



The name you assign for the each machine is called `token` when uploading data.

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 PBX 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).

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

If you do not want to connect to your PBX 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).



Actually, all "default" parameters that pertain to audio playback (Pluggable Listeners) are automatically transformed from the form "default" to "cluster.SERVER": for example, in cluster mode, the parameter `default.monitored_calls` will be automatically read from the property `cluster.SERVER.monitored_calls`.

```
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 your PBXs as 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 PBX box, you should set up the *[queuemetrics]* context in the dial plan, and make sure the manager interface is set up or the */var/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 *uniloader* 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, that is, uses a different token. We suggest using the HTTP upload mode, as it is more resilient to errors and just requires HTTP access from the PBX to your QueueMetrics instance. While not recommended, it is possible to override the `token` settings for a specific user by setting the `tokenfor._user_` system property.
- *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 PBX server only and QueueMetrics will complain they are not correct.

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 report page and editing as needed under the report's settings.

If you have agents using QueueMetrics's Agent's pages, you should set them up so that each agents 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*).

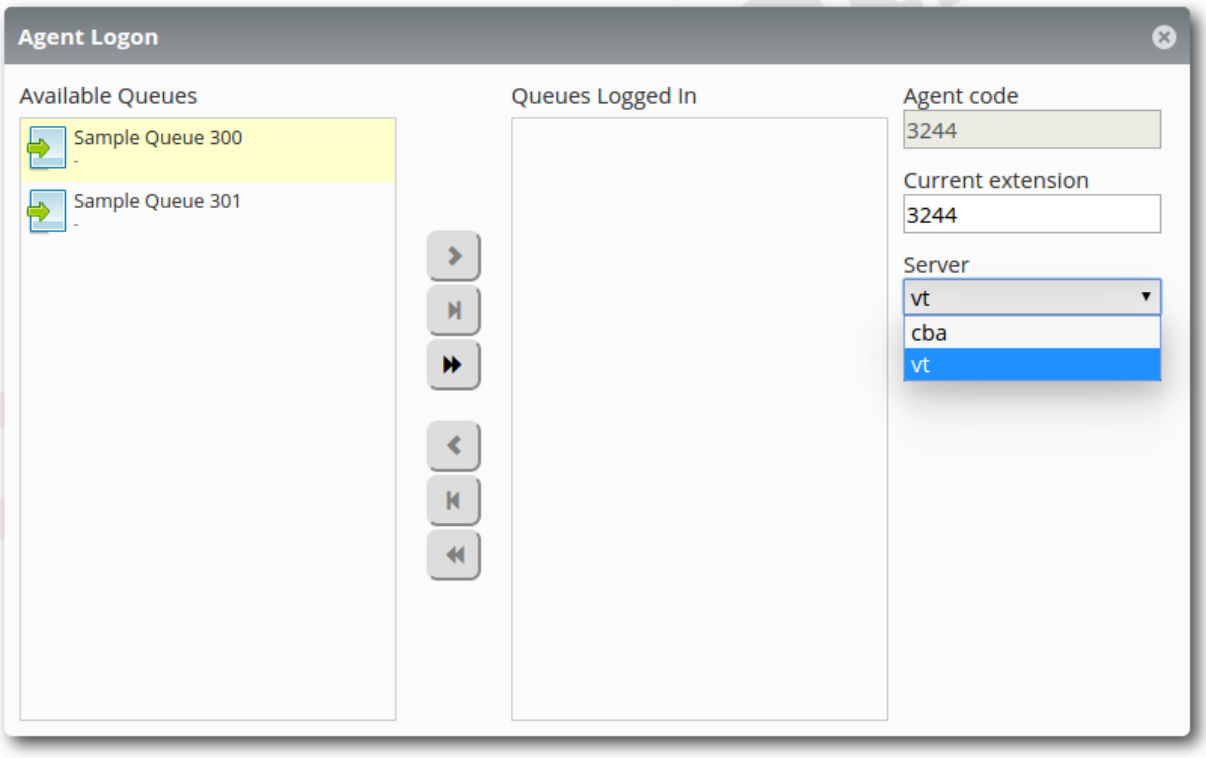
As the number of agents can be very high if compared to the number of supervisors who run reports or monitor the call center, QM uses a "minimal impact" policy: the page must be refreshed manually by each agent in order to avoid hammering the server with repeated page hits and the analysis run is a stripped-down, 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 the PBX on behalf of an agent, it needs to know to which PBX 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).

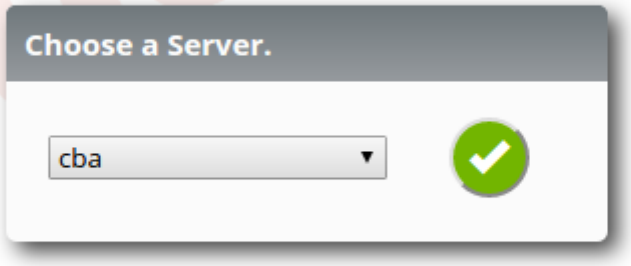
As a QM installer, you can control which servers are selectable to which agents by setting the properties *cluster.---.agentSecurityKey* correctly for each PBX server in the cluster.

Logging into a server from the Icon Agent's Page

When working on a clustered environment, an agent is allowed to access only one server at a time. The server selection is usually done through the Agent Logon Panel, where you choose the queues to log on to, the server to use and the agent extension.



the first time an agent logs into the Icon Agent's Page though, a special panel will appear that will ask the agent for the preferred server. This can be changed later on from the Agent Logon Panel.



There are some rules that QueueMetrics enforces, that must be understood:

- An agent can log onto only one server at a time.
- When the agent is logged onto a queue of a particular server, the server selection will be disabled.
- If a default server is set for the agent, QueueMetrics will disable the server selection for the agent, therefore the agent will not be able to use a server different from the default server.

To sum it up, an agent will be able to choose a server only if three conditions are met:

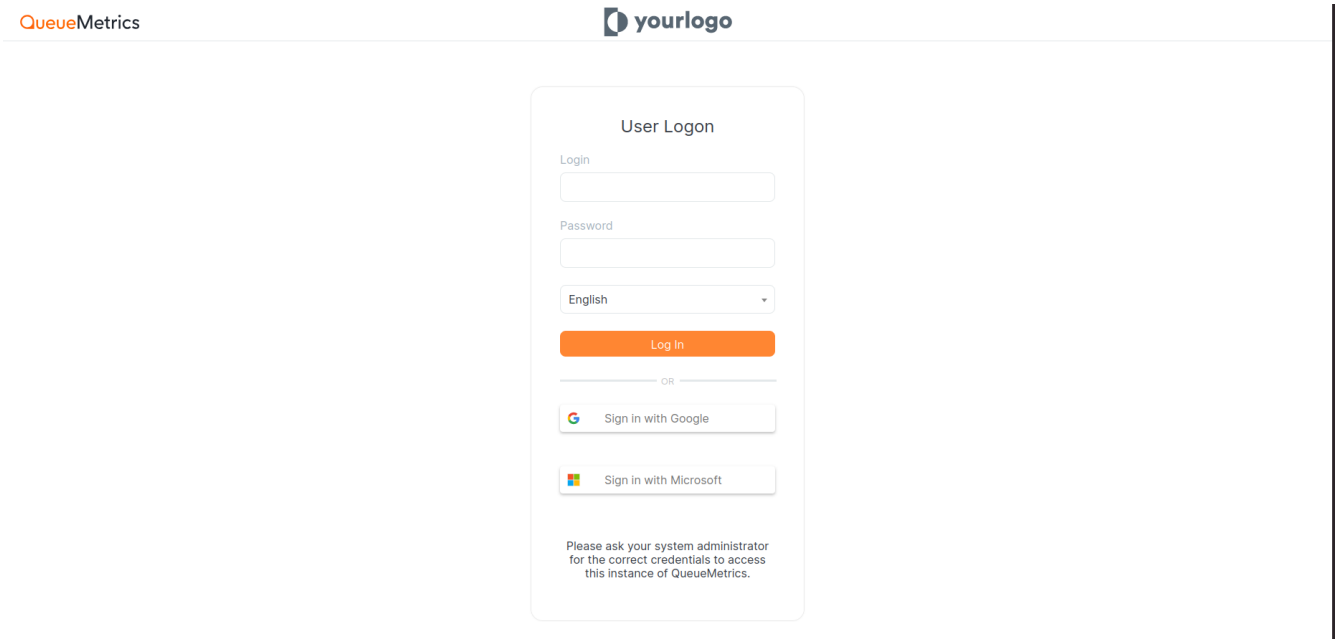
- QueueMetrics is working in Cluster Mode.
- The agent is not logged onto any queue in any server.
- There is no specified default server for the Agent. === Logging on to QueueMetrics

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 6 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:



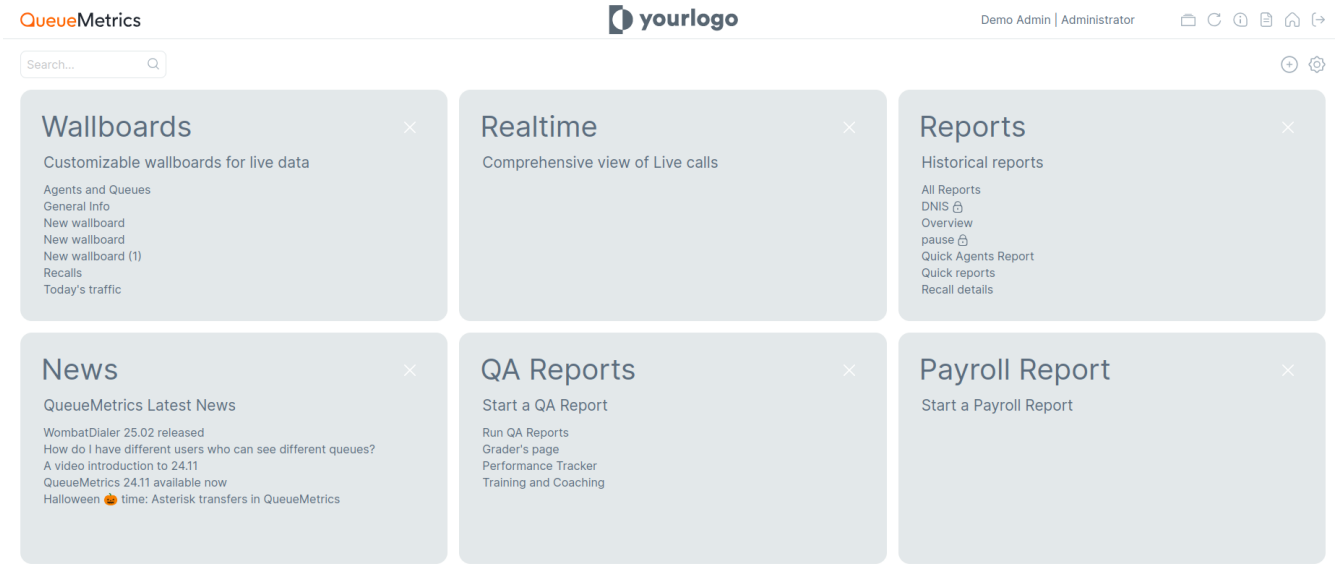
This and the following screenshots are taken using Chrome 85 on Ubuntu; other environments may present minor discrepancies from what is shown here.

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.

QueueMetrics Home Page



The user is presented with the Home Page, that is the starting point of QM.

InfoBoxes

The center part of the Home Page, is composed of multiple square boxes that are called "InfoBoxes".

Each InfoBox represents an integral feature of QueueMetrics, and allows you to navigate to it, by clicking anywhere inside the InfoBox.

Clicking on the cross icon on the top right of an InfoBox, will remove it from the HomePage, while clicking on the Plus icon located at the top left of the HomePage will Open the InfoBox selector, where a user can add or remove the InfoBoxes based on their personal preference.

Some InfoBoxes, such as the "Reports" InfoBox, or the "Wallboards" InfoBox, present built-in quick links, that will direct the user to specific reports, or wallboards that the User may have access to.

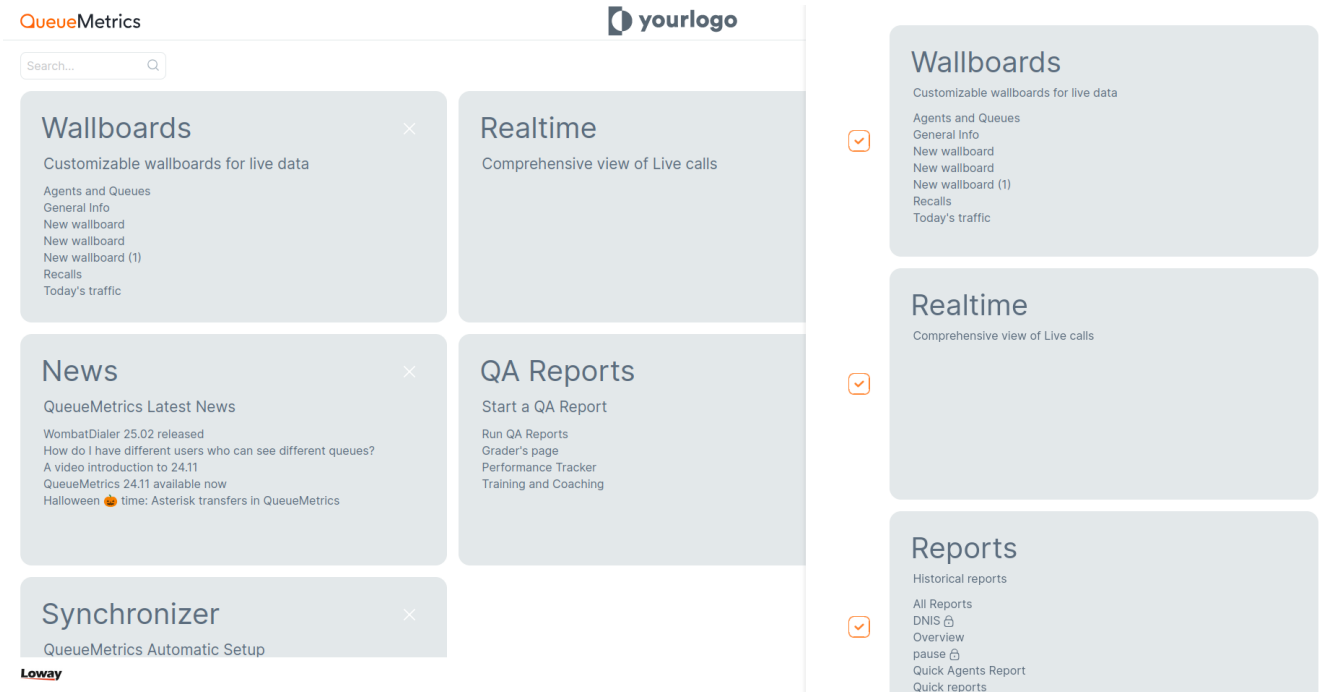


Clicking on the "Reports" InfoBox will open a new Unsaved Report. This report can be edited freely, but needs to be saved to persist on the system, by clicking on the "Save" Icon after making changes. If an Unsaved Report is not saved, it will be lost forever.

InfoBox Selector

Depending on the user's credentials and security keys, different InfoBoxes will be available to be added to the Home Page.

To see the list of all the InfoBoxes available to the user, click on the plus icon on the top left.



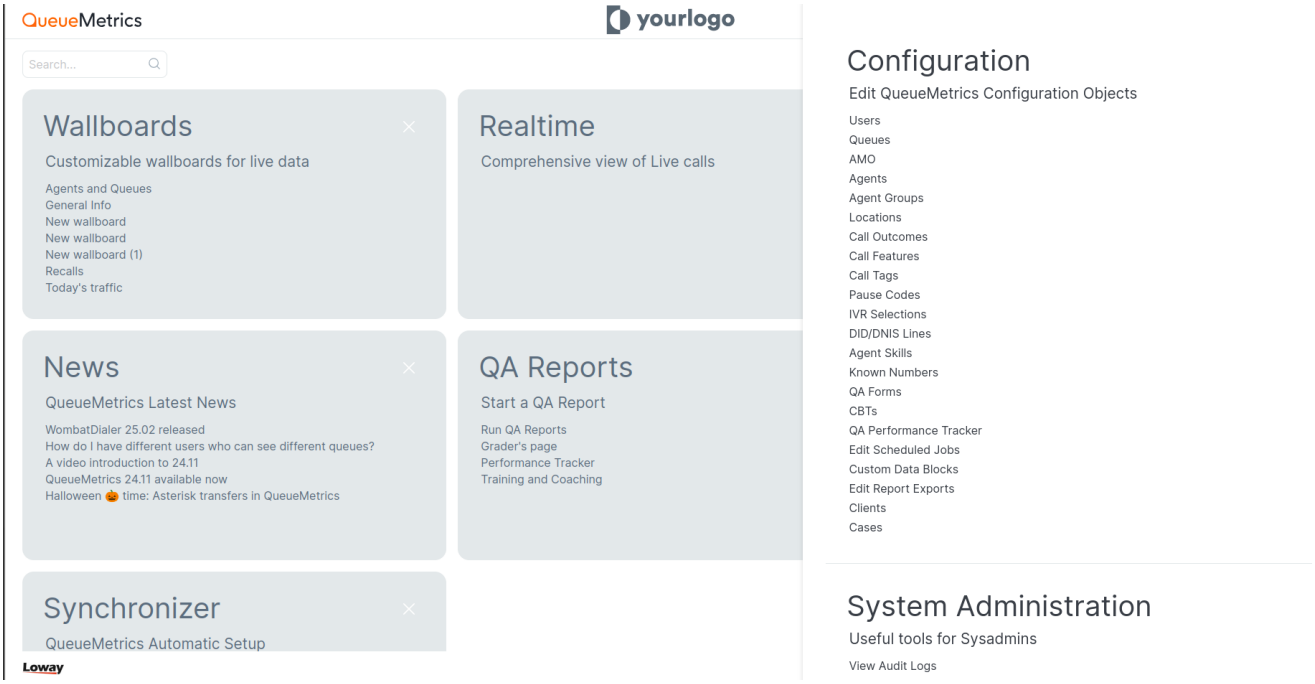
From this side menu, the user can add or remove InfoBoxes to the HomePage, or make use of the InfoBoxes directly without adding them to the Home Page.

InfoBox settings are persistent, so when the user logs in next time they will find the same InfoBoxes that were set last time the user accessed the system.

To close the InfoBox Selector, just click outside of the side bar, or close the side bar by clicking on the cross icon on upper left.

Administration Menu

To access QueueMetrics Configuration Editors, System Administration features and System Diagnostics features, the user can click on the Gear Icon at the top right of the screen.

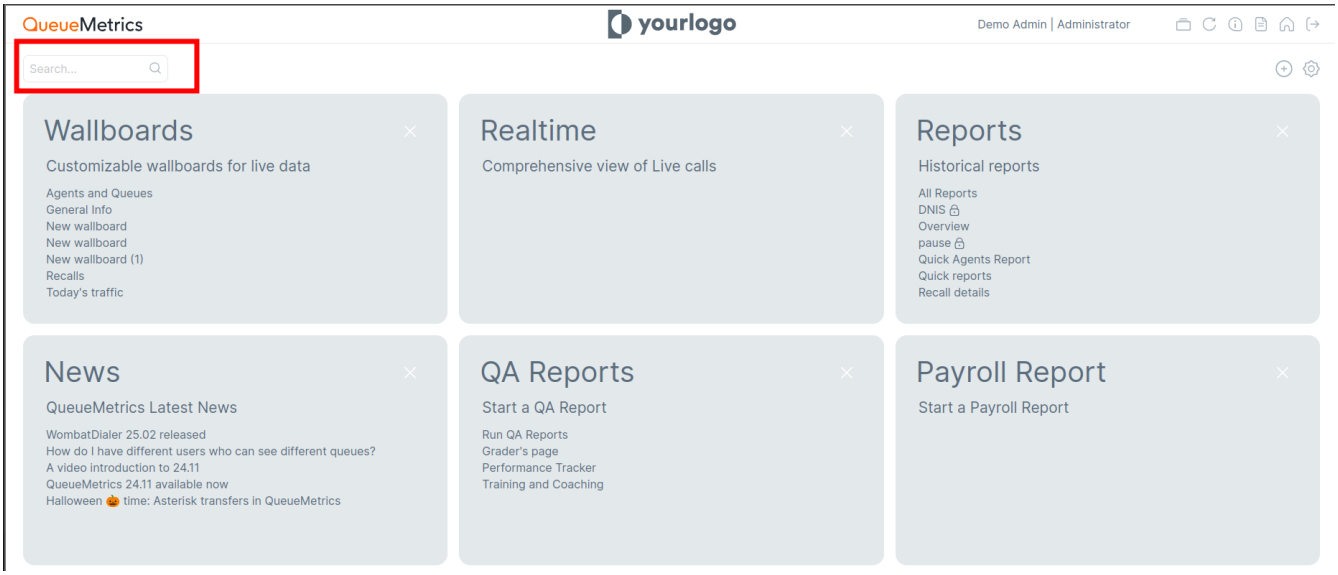


This will open the Administration Menu, which will display links to all the relevant features, that the user has access to.



All the configuration editors fall under this menu, so if the user is looking to create Queues, Agents, Pause Codes or any other kind of QueueMetrics Configuration Object, they will find them under this menu, provided they hold the appropriate security keys.

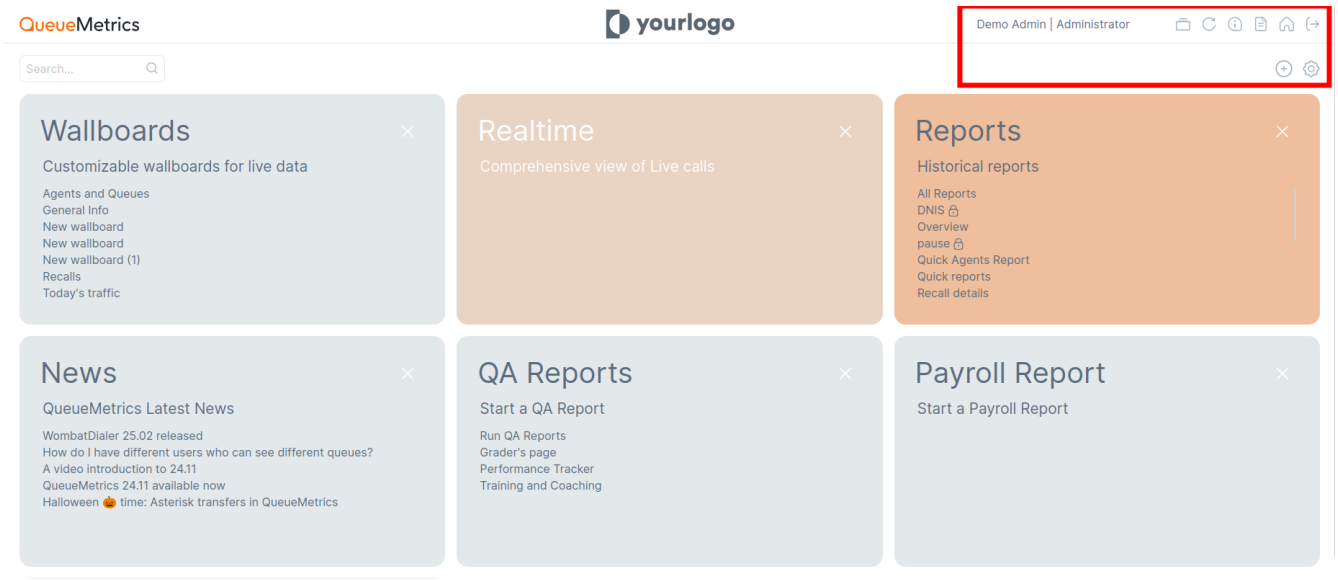
Search Bar



By typing a query in the search bar located at the top of the HomePage, the user can search for specific quick link, or QueueMetrics feature.

The filtering enacted by the search bar is applied to the central grid, the InfoBox Selector and the Administration Menu. The search query is case insensitive.

Header

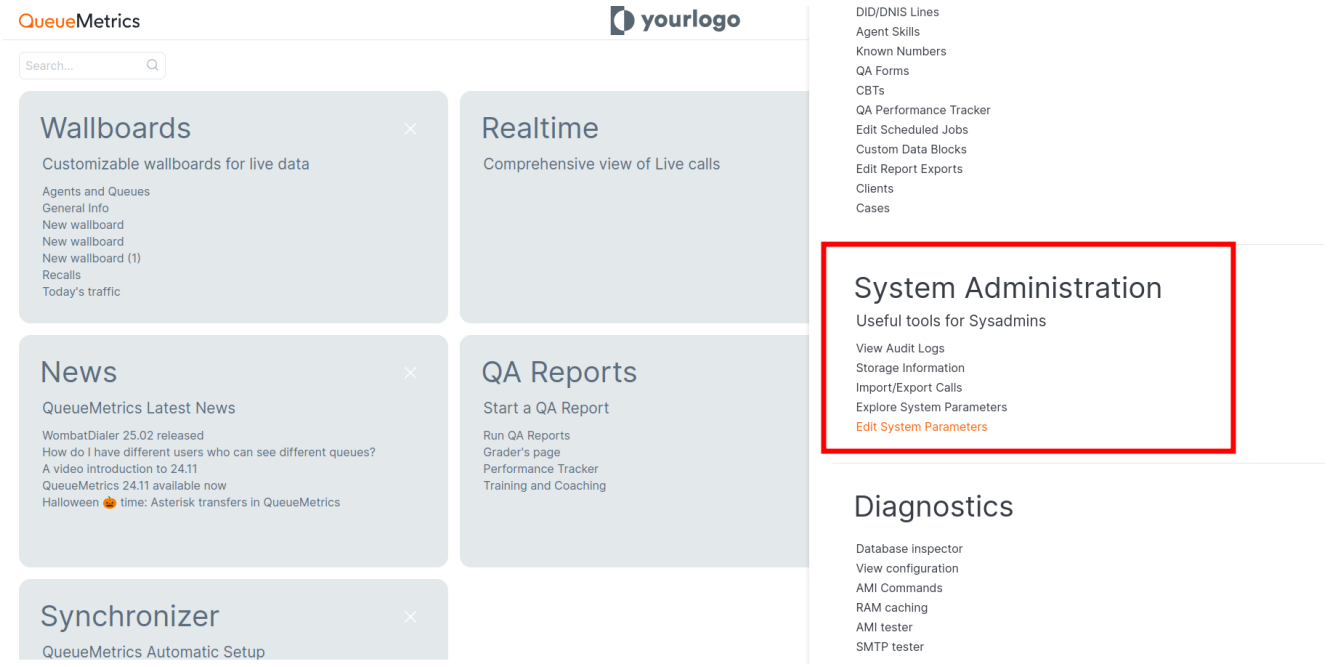


The name of the user and the current class for the user are shown on the top-right corner of the window.

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

To see more details on the current user and change its access password, or the currently selected theme, click on the "Info" icon.

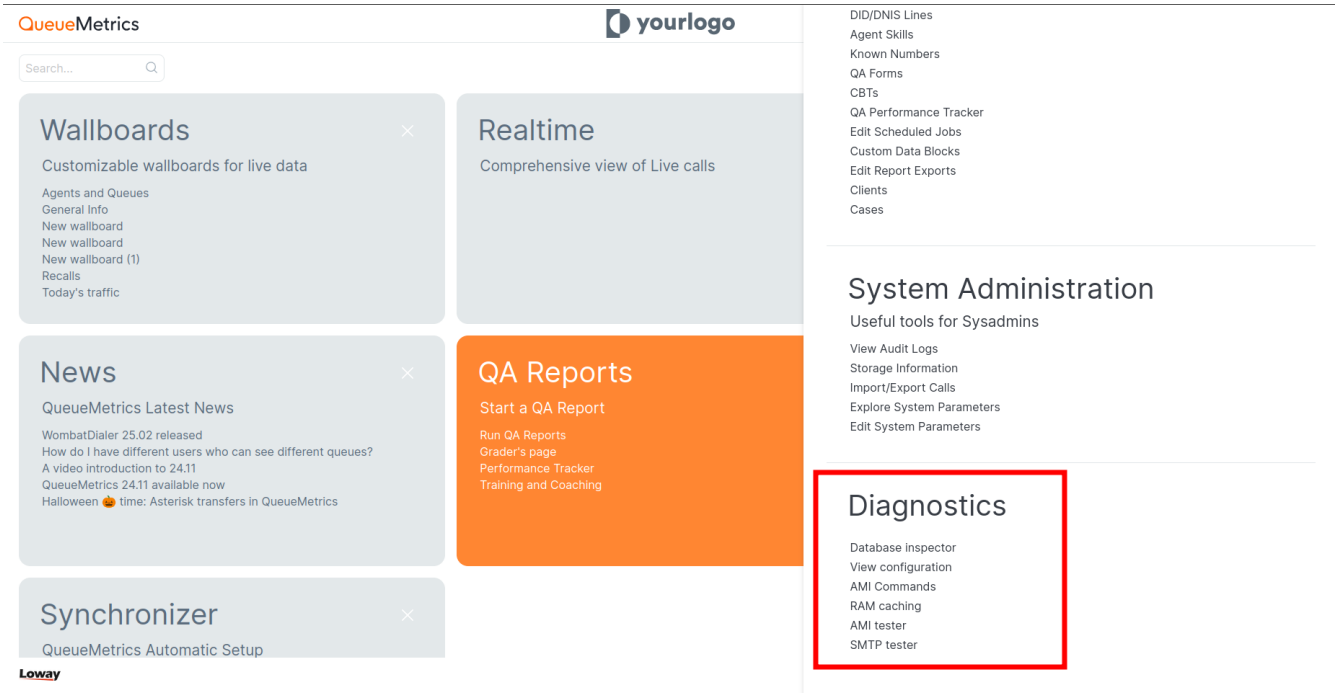
System Administration



To view the configuration properties from the Home page, see Configuring system preferences . This is achieved by adding the key EDIT_CFG to the admin user, which will enable two links on the Home page, within the Administration Menu, that will allow access to the file. It is possible to edit it either in unstructured format (as a text file), by clicking on "Edit System Parameters". If you are looking for the most common options, a GUI interface is available as explained in Configuring system preferences , under the "Explore System Parameters".

From this menu, the user can also look at Audit Logs, Mysql storage information, access the Agent Awareness Manager and use the Import/Export calls function (Deprecated function).

Diagnostics

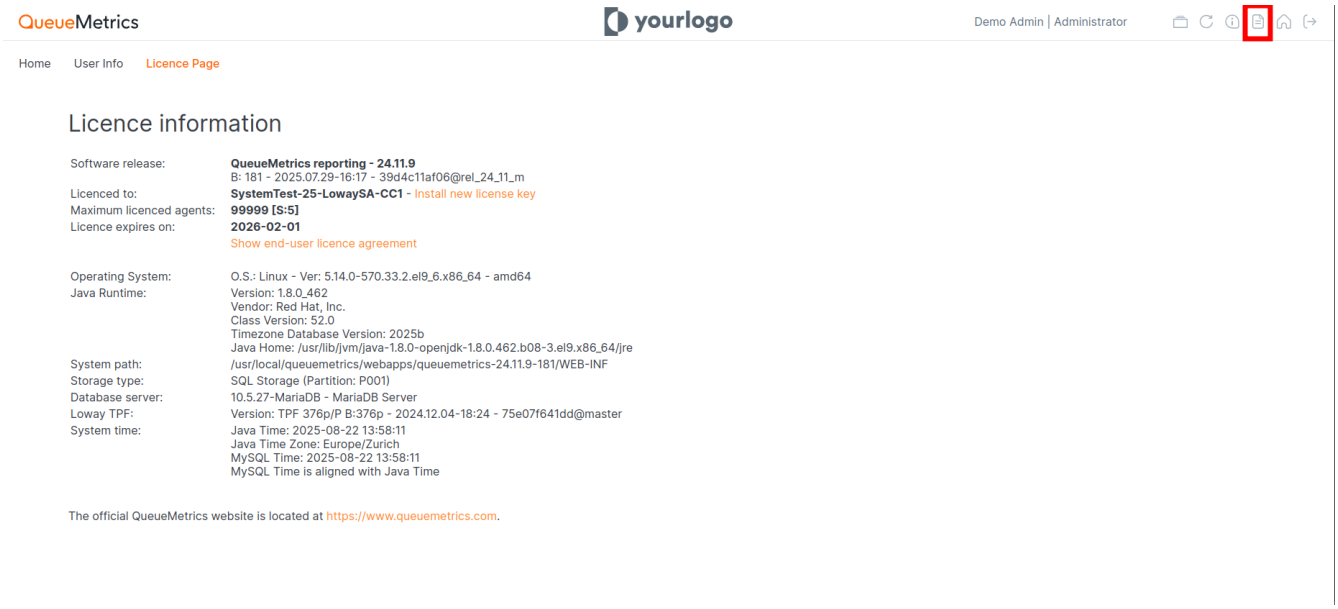


From this menu the user, provided he holds the correct security keys, can access the System Diagnostic Tools, which comprise of a Database Inspector, the View Configuration page, A debug

page for AMI Commands, A page to check and eventually reset RAM Caching, A tester for AMI connections as well as an SMTP connection tester.

License information

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



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

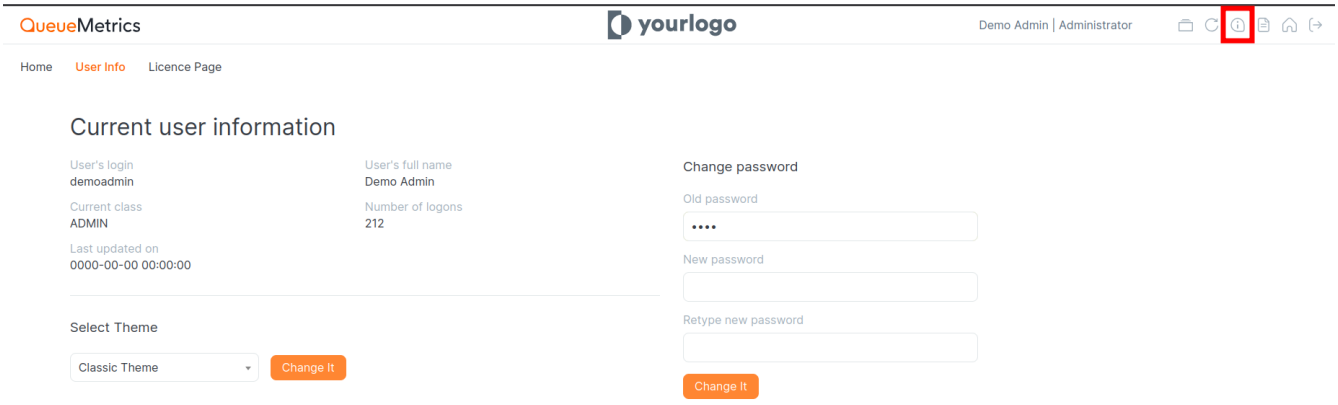
If you are running a free demo version, you will see that the maximum number of licensed agents is 2 and an additional text will remind you on how to register.

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.

If QueueMetrics is to be run on a publicly-accessible box, it is possible to hide all technical information from the user by setting a configuration property.

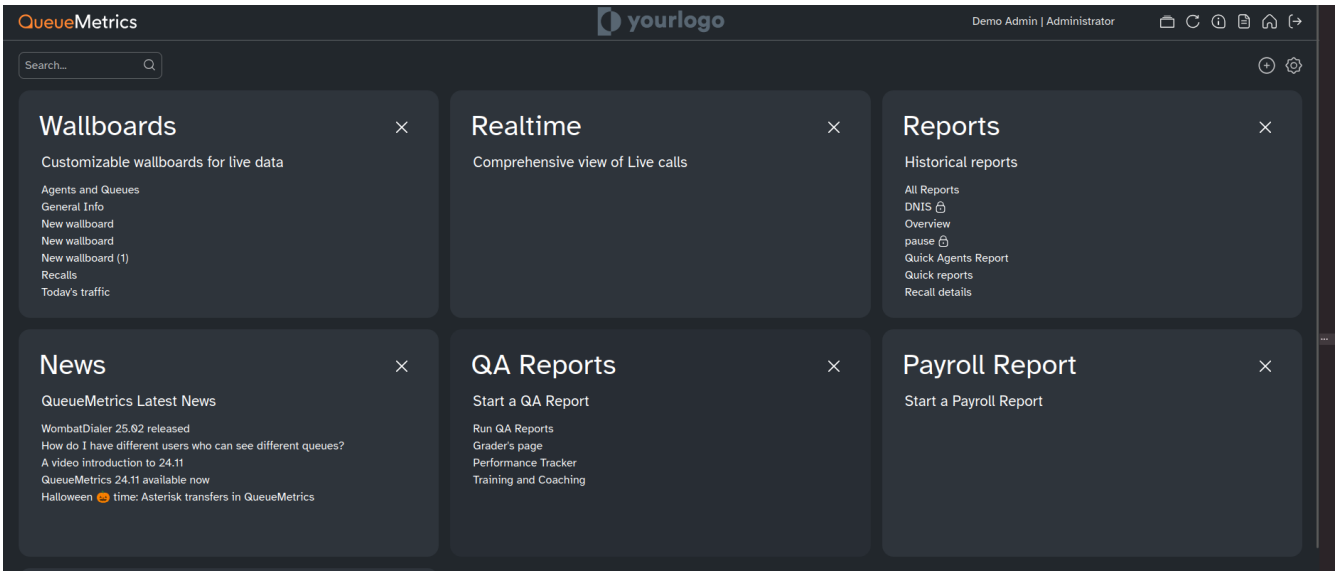
Change Password and Theme

In the User Information page, accessible from the Header icon bar by clicking on the "Info" icon, the user can change their own password, and select the QueueMetrics theme to be displayed.



By selecting the desired theme, and clicking on the Change It! button, the user can apply the selected theme.

Now, and on all subsequent logins, the theme will be displayed as selected.





This information is stored as a Cookie for the user.

Automatic URL login

It is possible to use a direct link to log in into QueueMetrics from an external browser application.

If you click on an URL like the following:

```
http://ipaddress:8080/queuemetrics/qm_autentica.do?AUTH_logon=demoadmin&AUTH_password=demo
```

then the user will be implicitly logged in as "demoadmin" password "demo". Auth parameters can be passed in either HTTP GET or POST, with the advantage of POST of making them invisible to the calling user.

Change Your Logo image

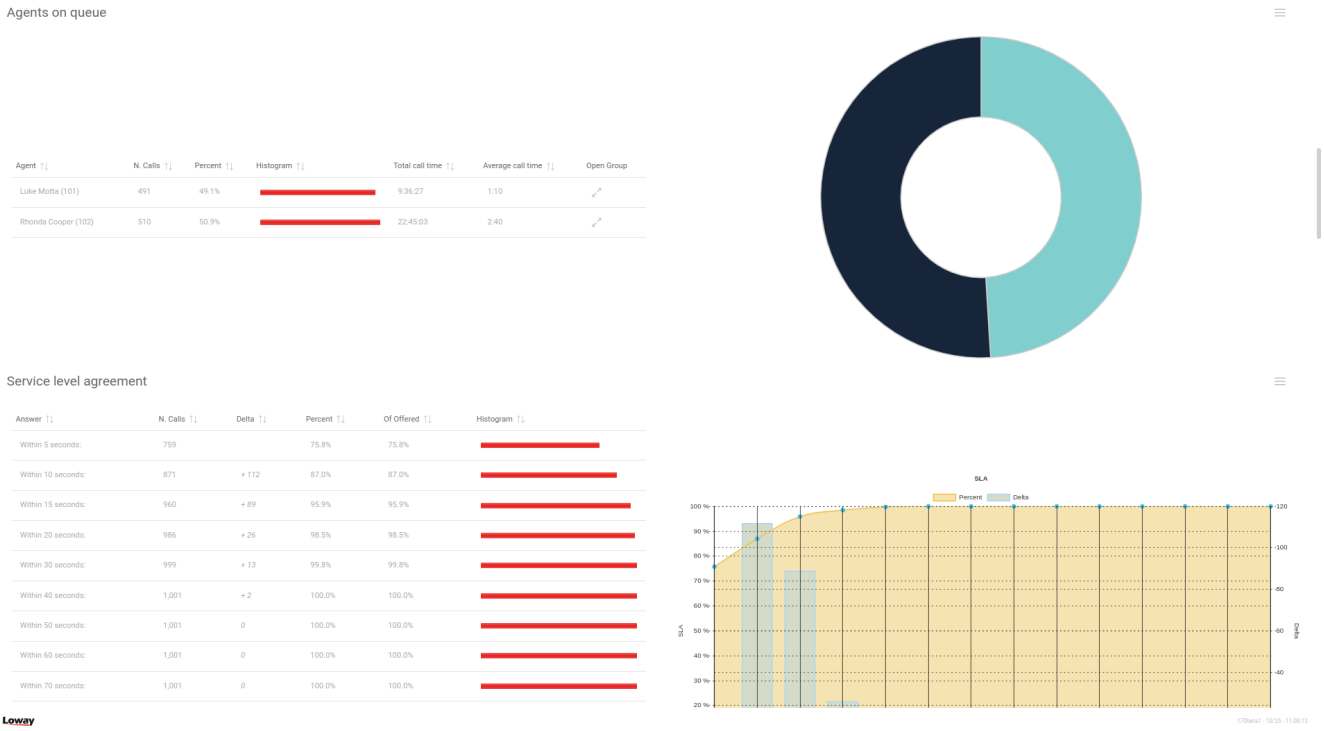
In order to change your logo, you have to edit the System Property "layout.logo", its description and what to insert can be found in layout and behavior preferences. This property can be edited as described in Configuring system preferences.

To ensure a smooth integration of your logo within the QueueMetrics header, we ask that you provide an image that meets the following specifications:

Fixed height: 40px
Recommended maximum width: 200px (if possible, preferably within 150px)
Recommended format: PNG or SVG for best quality on all devices
Background: Transparent or solid color compatible with the interface

If your logo has a particularly vertical or squarish format, we suggest that you provide a more horizontal version for better fit to the header. This will ensure a more balanced visual appearance and greater readability.

Report Page



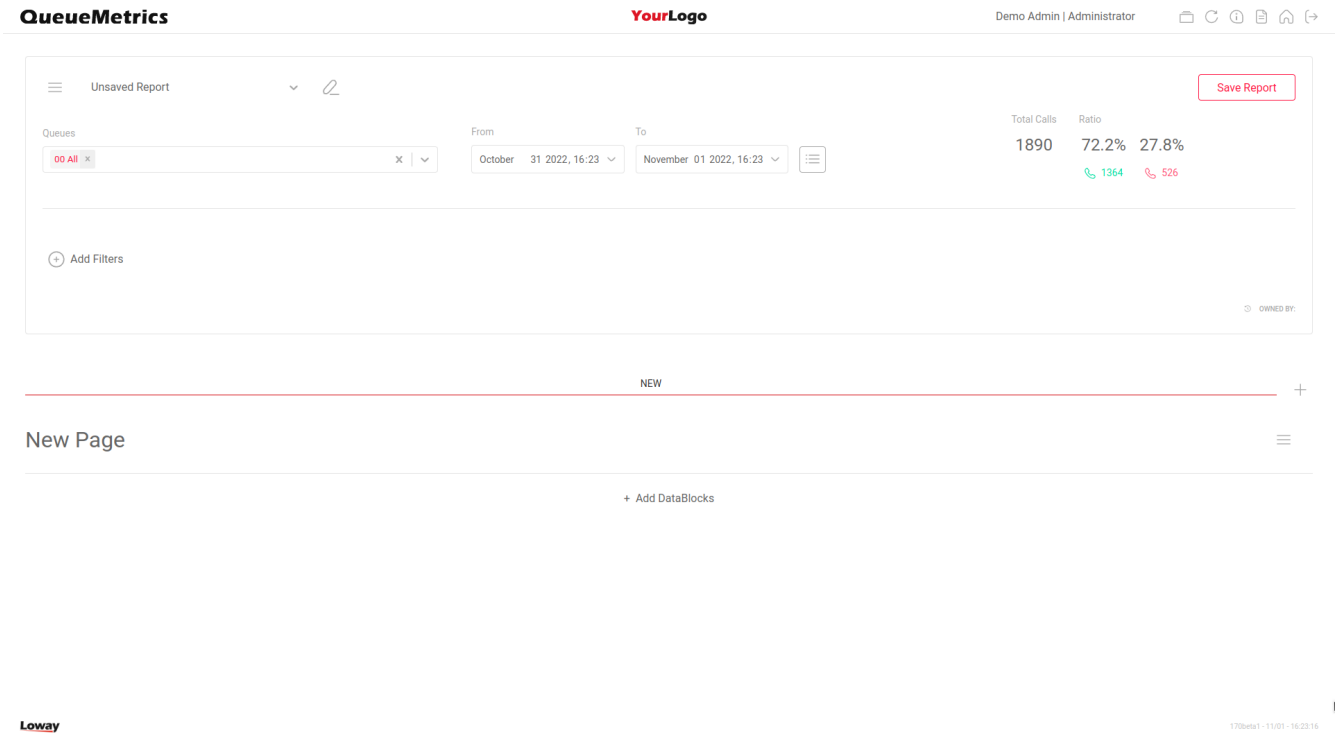
The new Report Page offers a wide variety of functions, that allow the user to create, customize, publish and share their own report.

What is a Report?

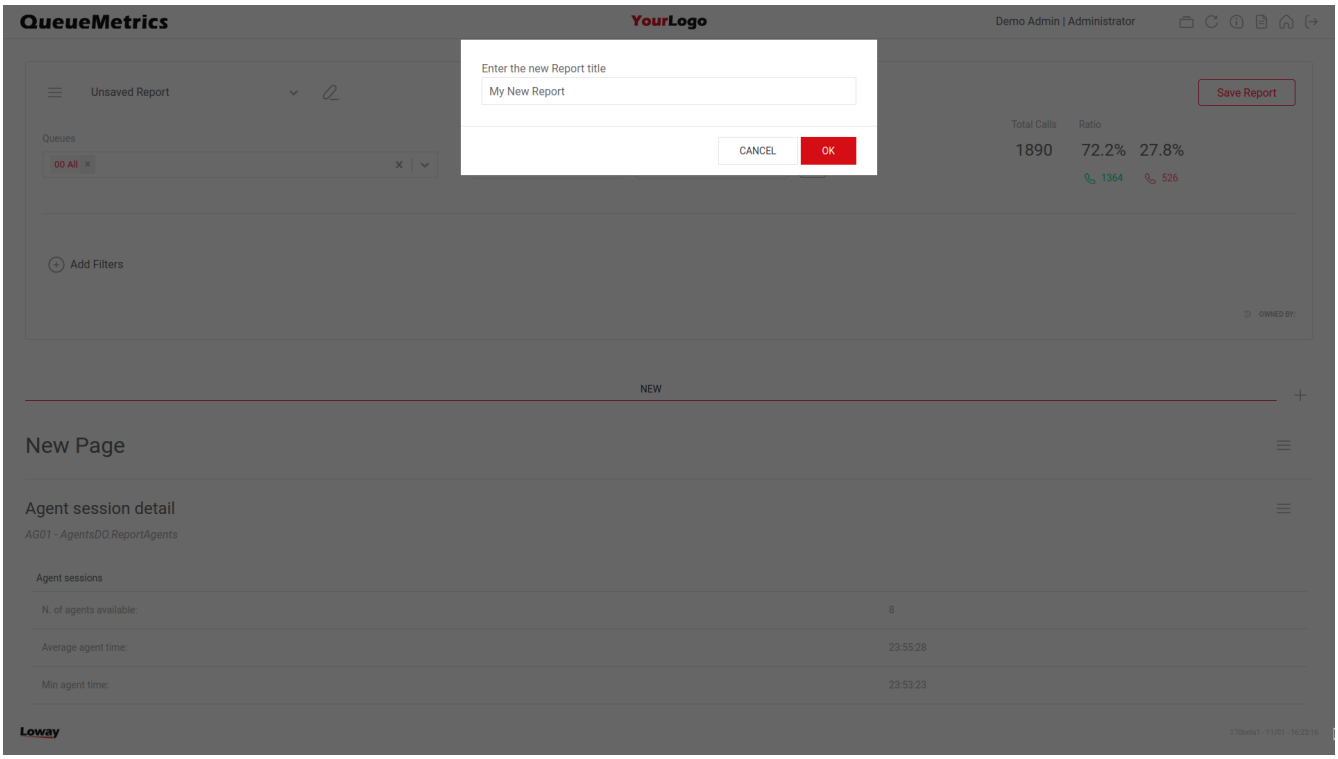
In QueueMetrics a Report is a collection of different pages, each one containing a variable number of DataBlocks.

Unsaved vs Saved Reports

A newly created or cloned report will begin its life in the "Unsaved" state. You can easily spot when a report is Unsaved, because the report title will be "Unsaved Report".



After making changes to the Unsaved Report, a Save Button will Appear in the top right that needs to be clicked in order to save the report. If the button is clicked, the user will be prompted to input a name for the Report.



Once a report is saved to the system, the user can still edit it as per usual, but now the report will be accessible even after exiting the report page. If you leave the report page before saving a report - a system popup will warn you when you leave - the report you were working on will be lost forever.



You cannot export nor share an Unsaved Report.

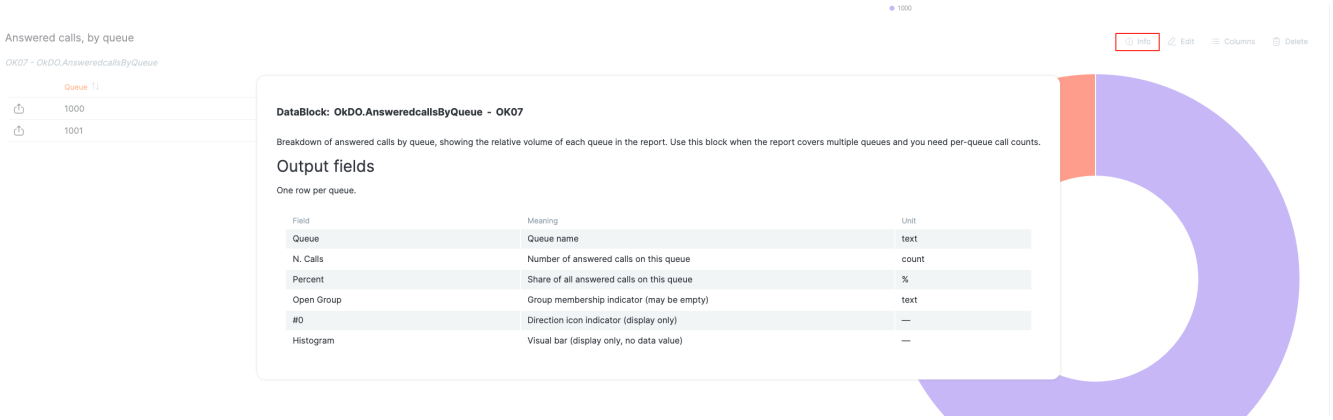
DataBlocks

A DataBlock is a block of information with a Title, a Subtitle, a Table and possibly a Chart.



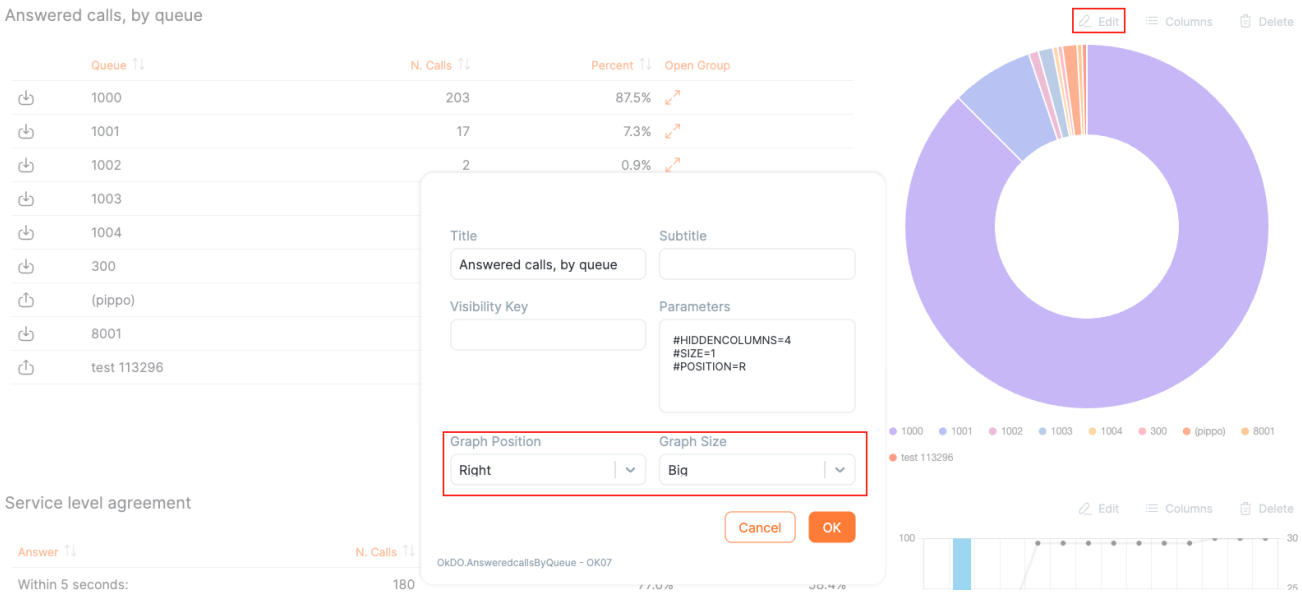
QueueMetrics offers 150+ different DataBlocks, each with unique information and metrics. When building a report, the user will pick and choose the DataBlocks they are interested in, and add them to the current report. In doing so, the user is able to create a Report that perfectly matches their needs.

To view additional details about a DataBlock, clicking the "Info" button next to its name opens an informational modal.



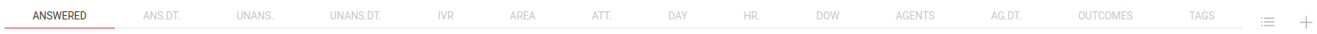
Users can access report that are created by them, or public reports created by other users, provided they hold the necessary security keys.

As of QueueMetrics 26.01, datablock graphics now feature adjustable sizing and positioning. These options are accessible via the Edit icon within the datablock settings popup.



Pages

Each Report can contain any number of pages. The user can navigate through pages by clicking on the various tabs that are shown at the beginning of the page.

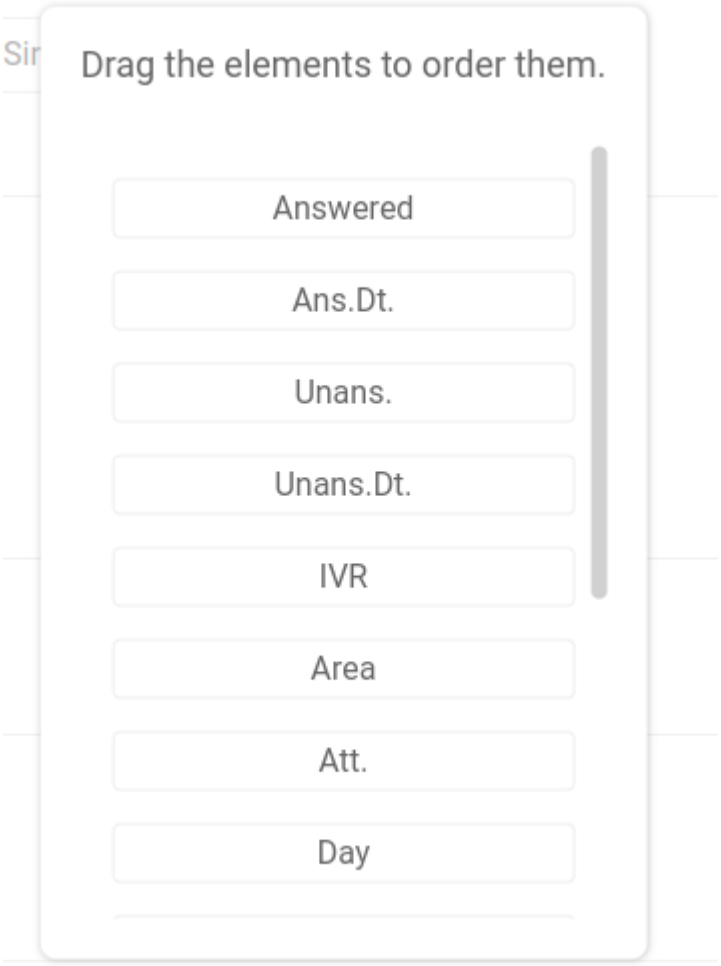


To create a new page, the user can click on the "plus" icon on the right

To edit the page order, the user can click on the Edit Tabs Order Icon next to the plus icon

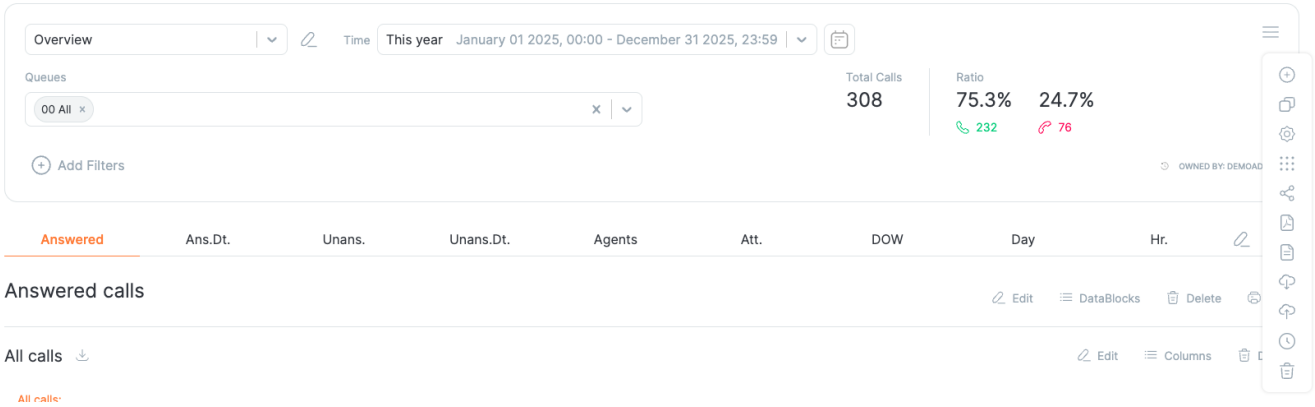


This will open a panel where the user can drag and drop the Page names to reorder them



Icons

In the top right section of the screen, the user has access to a menu containing a series of icons, that allow for different actions to be performed.



Create Report

The "Plus" Icon, if clicked, prompts the user to input a name for a new report to be created. If clicked, a new report will be created with one empty page.

Settings

The "Gear" Icon will open the Report Settings Panel, where the user can edit the current report's settings.



this section contains the autosave feature to enable/disable the autosave function for your reports.

Add DataBlocks

The "DataBlock" icon opens the DataBlock Selector, displaying a live preview of all available QueueMetrics DataBlocks. Users can add selected DataBlocks directly to the current report page. To view additional details about a DataBlock, clicking the "Info" button next to its name opens an informational modal.

Share

The "Share" Icon will copy a shareable link to the current report, in the user's clipboard. This URL, if navigated to, will prompt an unauthenticated user to access QueueMetrics, then redirect him to the current report. If the user is already logged in QueueMetrics when he navigates to the link's destination, then he will open the current report directly.

Export in PDF Format

The "PDF" Icon will export the current report in PDF Format.

Export in XLS Format

The "XLS" Icon will export the current report in XLS Format.

Export Report Schema

The "Download" Icon will export the current report in JSON Format, so it can be imported in a different QueueMetrics instance.

Import Report Schema

The "Upload" Icon will prompt the user to pick a QueueMetrics report schema (Exported from this QueueMetrics or another instance), to import it. The QueueMetrics report schema file should be in JSON Format.

Schedule export Job

The "Clock" Icon will prompt the user a popup to create and schedule an export job. See Schedule Export Job

Delete Report

The "Trash" icon will prompt the user to confirm the deletion of the current report.

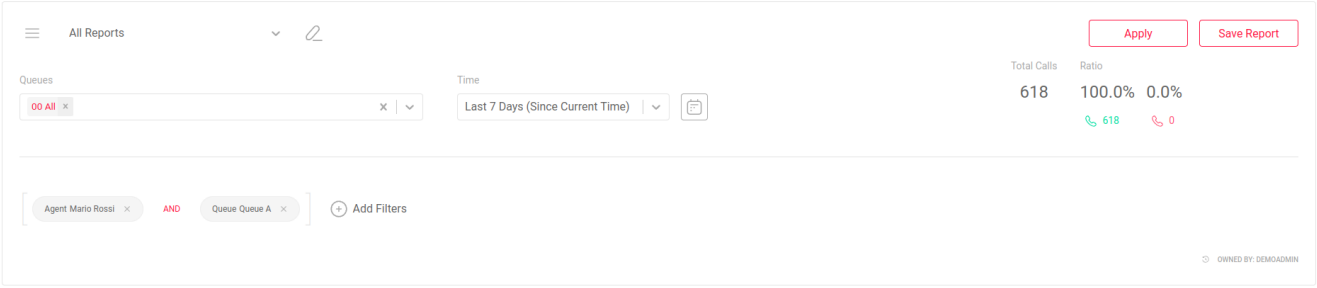


if the report contains some custom DataBlock that are not present on the importing QueueMetrics instance, QueueMetrics is going to follow this process:

A) Look for a Custom DataBlock with the same ID. If it exists, go to step B. If not, go to step C. B) Checks if the existing DataBlock is the same as the one from the other system If so, it keeps it, if not, go to step C. C) Creates a new DataBlock with the same properties as the one on the old system. If the ID is already in use, it creates a new one.

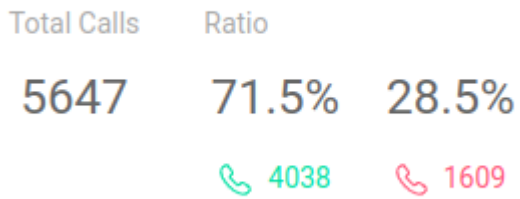
Filter Recap Panel

The area at the top of the Report Page, is the Recap Panel, that offers immediate insights about the current report and the applied filters.



Call Stats

The right section shows the number of taken, lost and offered calls, together with their percentage ratio. They also show the number of processed calls in the report.



Selected Report

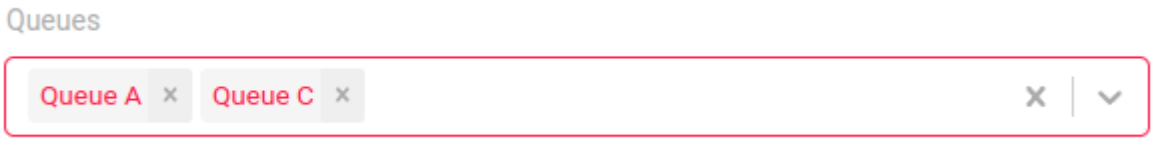
The left section shows the name of the currently selected report.



The user can change report by using this menu and selecting one of the available reports.

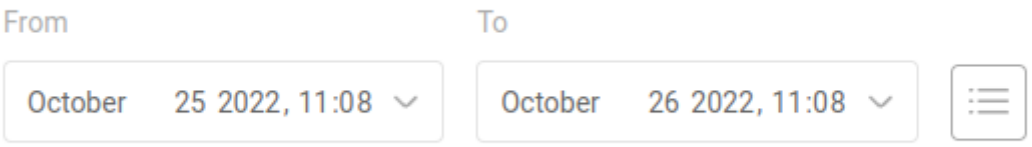
Selected Queue

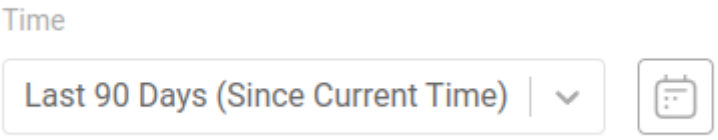
The "Queues" section shows the name of the currently selected queue or queues.



Time Period

Finally, a time selector allows the use to change the time period the report is considering when analyzing data. The user can pick a date range or a preset time period. To switch between time selection modes, the user must click on the calendar icon.

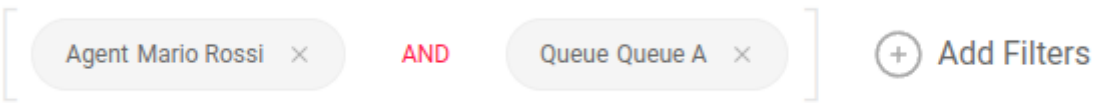




If needed, the maximum allowable time period for reports can be set system-wide and then overridden on a user-by-user basis - the default is that any user can run reports of any length, but this can be changed by the administrator. See Maximum reporting period for further details.

Filter Expression

If a filter expression is set for the report, it will appear at the bottom of the Recap panel.



Rename Report

To rename the current report, the user can click on the edit icon next to the report name



This will make the Report Name editable for the user.



Here the user can confirm the changes by clicking on the confirm icon or delete the report.

Delete Report

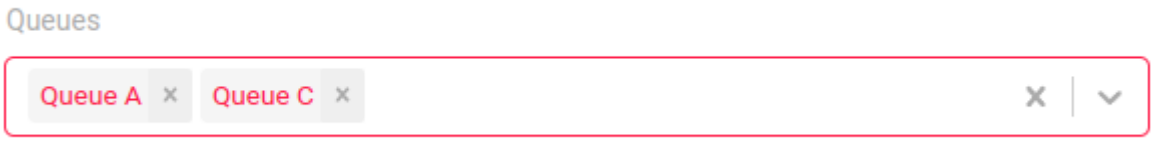
To delete the current report, the user can click on the delete icon instead of the confirm icon, when editing the Report Title



Since queuemetrics 26.01 you can also delete from the top right menu

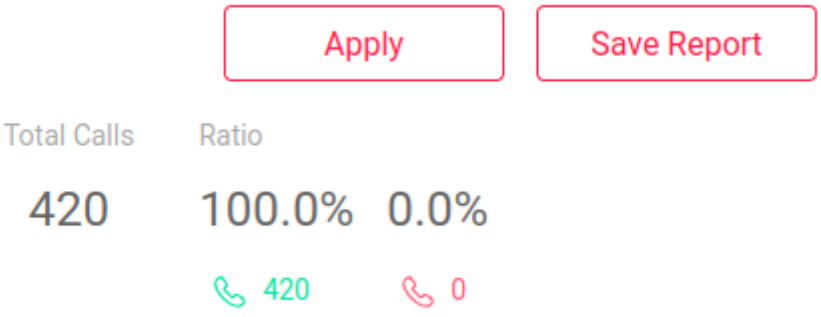
Change Queues

To change the queues to consider for the current report, the user can use the following menu. More than one queue can be selected at once.



Save or Apply

When the user makes some changes to the report, two buttons will appear in the top right of the screen. These are the Apply and Save button.



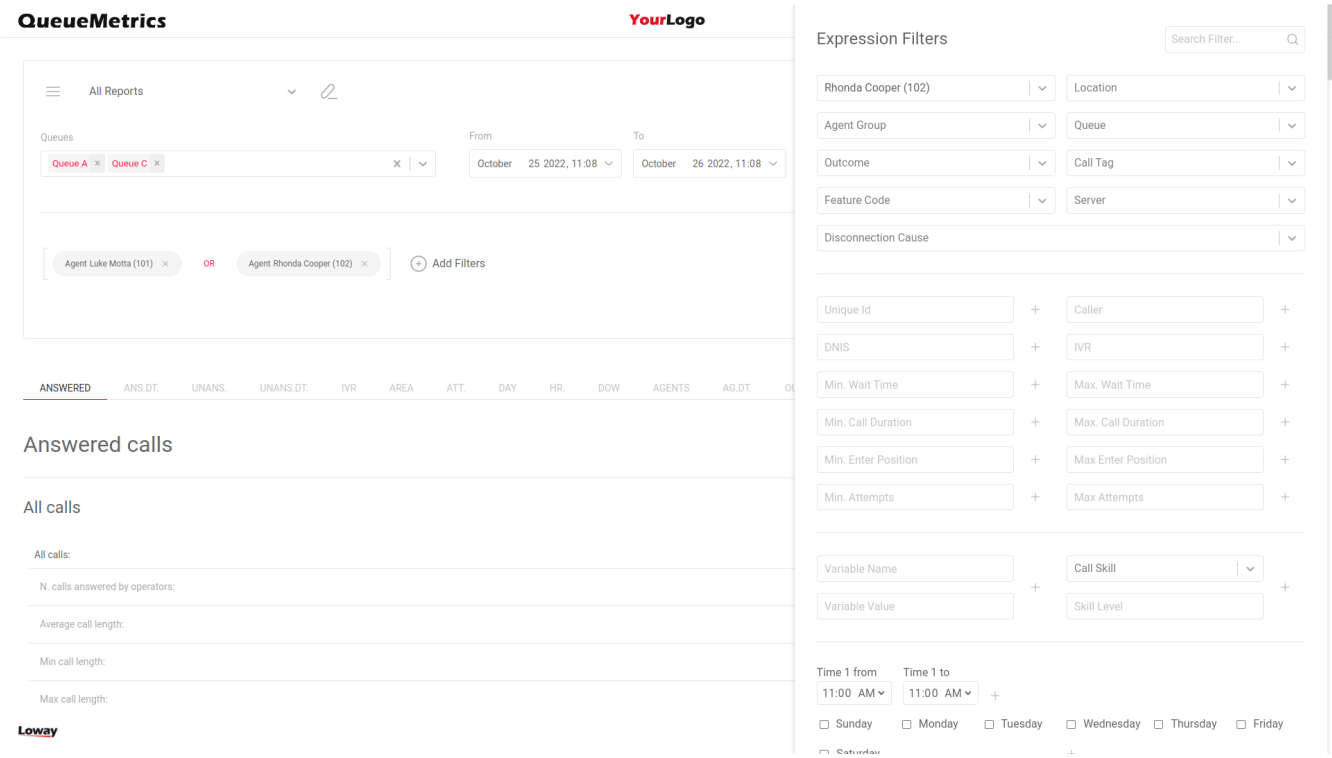
Clicking on the Apply button will apply the new changes to the report, and recalculate the report with the new parameters and filters. This will however not save the report.

Clicking on the Save button will first save the new report parameters and then recalculate the report, exactly as if the user had pressed on the Apply button.

Add/Edit Filter Expression

To further filter the current report, the user has access to an entire list of different filters that can be applied. These filters can be combined in an infinitely complex expression, that offers granular control over the data included in the current report.

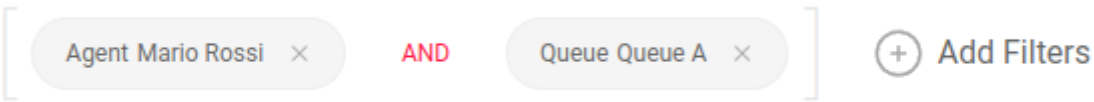
To add a filter to the filter Expression, the user must click on the "Add Filter" button next to the current filters, at the bottom of the Recap Panel.



This will open the Filters Collection panel, where the user can add filters to the current report.

To easily access the desired filter, the user can use the search bar at the top.

Once the user inputs a new filter, it will appear in the Report's filter expression.



Here, the user can add as many reports as they want, and combine them together with operators.

Type of filters

There are three types of available filters in the Filter Collection Panel.

Combo Filters

These filters show the user a series of options, and the user can only select one of the options. The user can search through the available options by typing a string with the menu open. This will filter the options to show only options containing the inputted string. An example of this type of filter is the "Agent Group" filter.

Combo Filters with Creatable Options

These filters are similar to Combo Filters, as they show the user a series of options and are searchable. The only difference is that if while inputting a string to filter the available options, no available option matches the inputted string, the user will be presented with an option to create a new option from scratch, with the inputted string as value. The Agent filter is an example of such filter.

Text Filters

These filters are simple text boxes that accept a value inputted by the user. Depending on the filter some values might not be valid. When the user inputs a value and the value is a valid value, a "plus" icon will appear next to the filters that needs to be clicked to add the filter to the expression.

Regular Expressions

Most of the available filters that accept user input as a value, are also capable of accepting a regular expression, as long as it starts with "^".

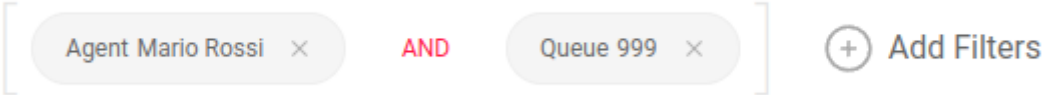
Filter Expression

For more information about the available filters, please consult Call filtering criteria.

Add Filters

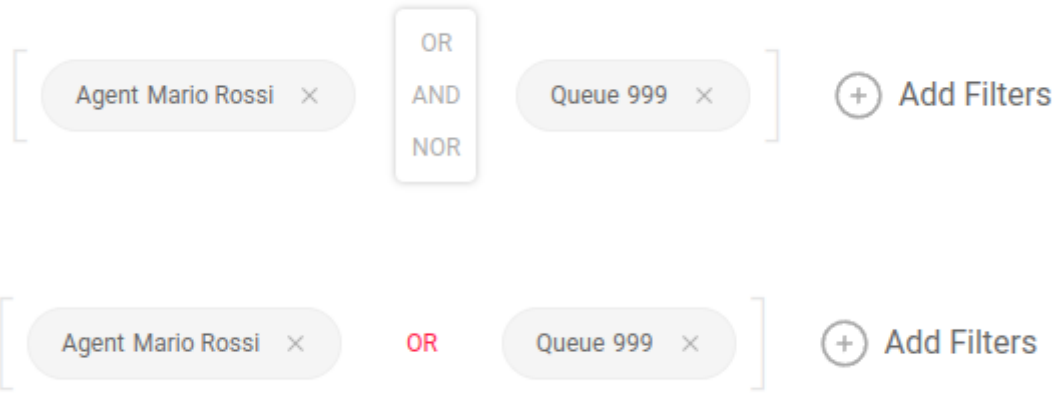
When a filter is selected, it is then added to the filter expression, at the bottom of the Report Editing Panel

More than one filter can be added at once. They will be joined with the "And" operator by default.



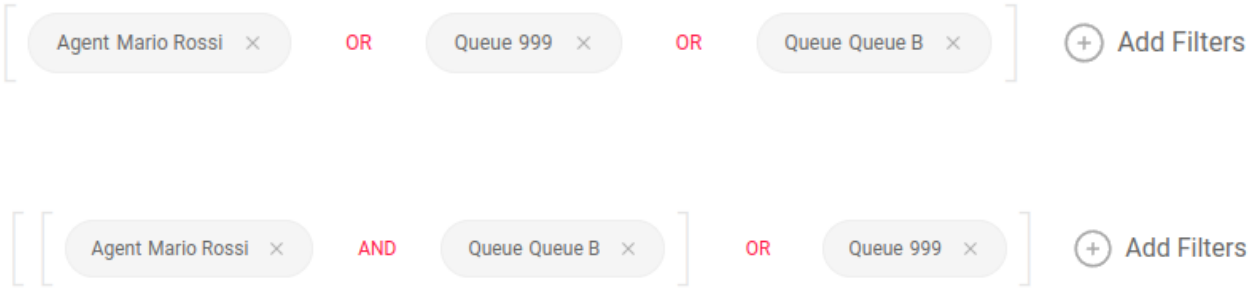
Change Operator

By clicking on the operator, the user can change the operator for the entire group of filters.



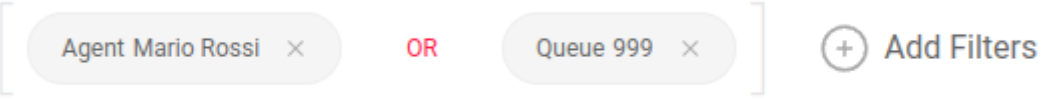
Grouping Filters

Filters can be grouped together by dragging one filter upon another.



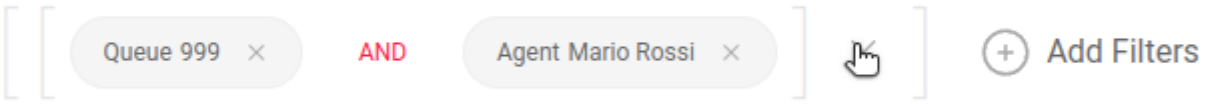
Deleting a Filter

The user can delete a filter by clicking on the cross icon inside of the filter.



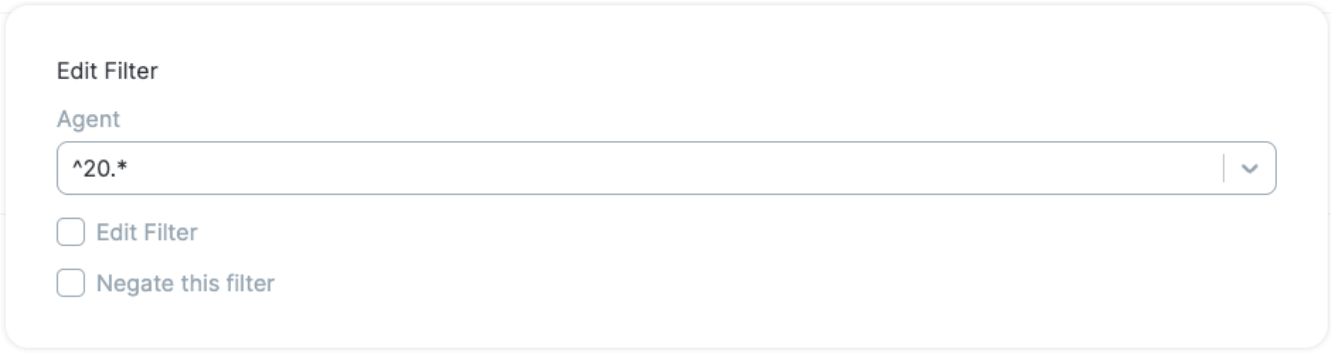
Deleting a Filter Group

The user can delete a Filter Group by hovering over a closing parenthesis with the mouse (or keeping it pressed while on mobile) and then clicking on the "cross" icon that appears.



Editing a Filter

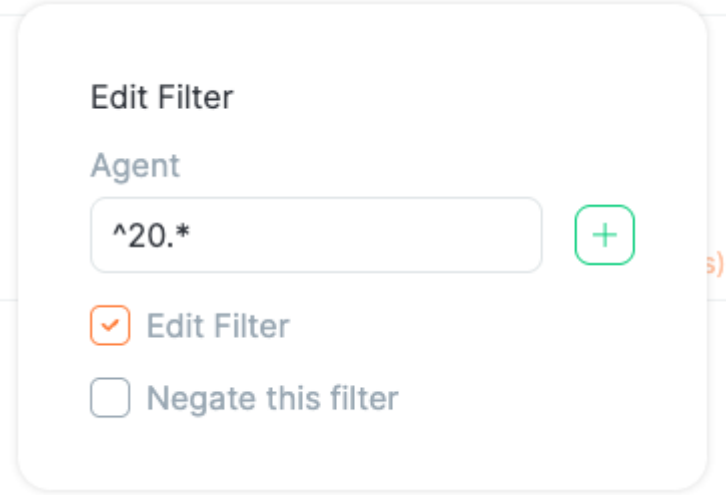
Clicking on a filter's value will open the Filter Editing Panel, where the user can edit the value of the current filter.



Editing a Regexp Filter

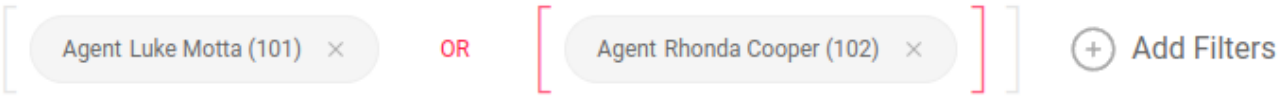
Since QueueMetrics 26.01, in the Filter Editing Panel, if a filter supports regular expression input, clicking "Edit Filter" allows the user to modify the existing filter without recreating it and apply the changes by clicking the "plus" icon.

To be considered a regular expression filter, the value must start with "^".



Negating a Filter

In the Filter Editing Panel, by clicking on the "Negate This Filter" checkbox, the user can negate a filter. If, for example, a report is filtered by Agent 101, by negating the filter, the report will show all agents except Agent 101.



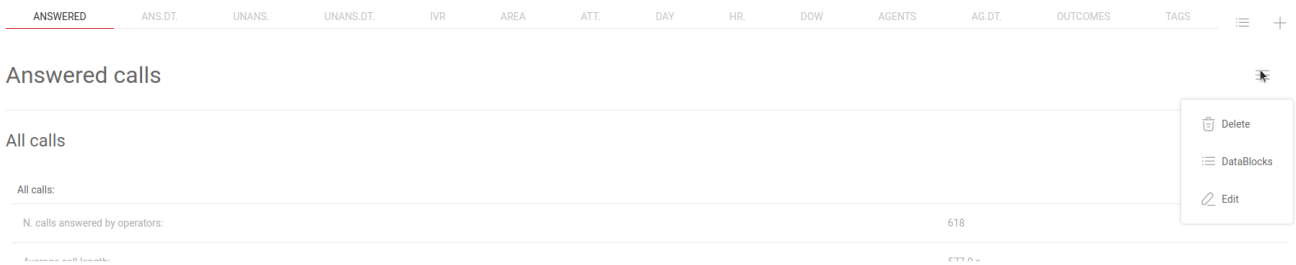
Negating a filter effectively means that the filter is in a NOR group by itself. If another filter is dragged on top of a negated filter, the two filters will be grouped in a group with the NOR operator between them.

Editors

The Report Page offers different editors to allow the user to customize Pages and DataBlocks.

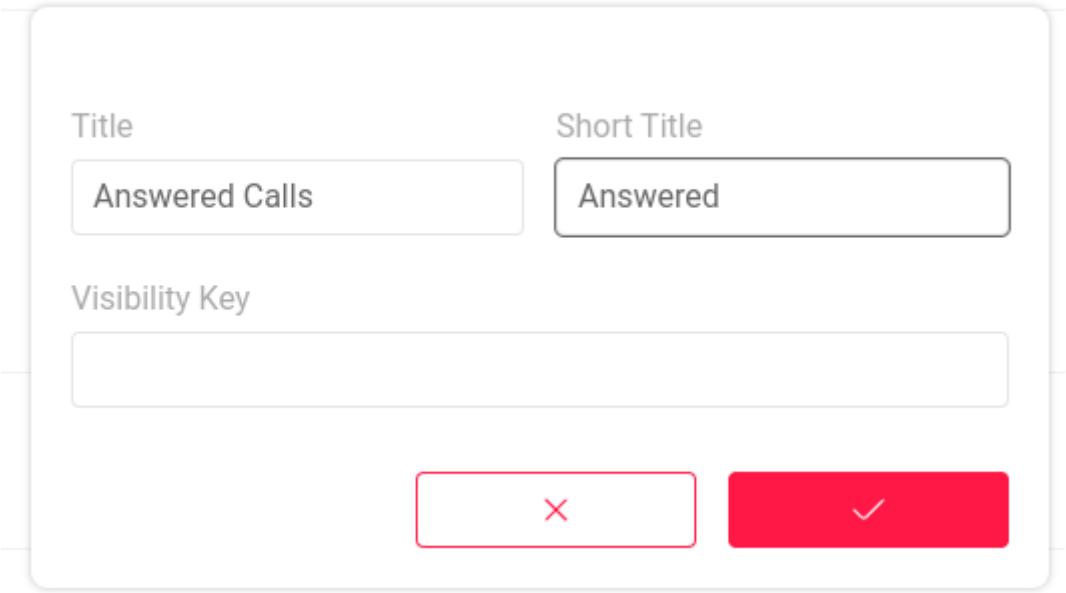
Page

To the right side of a Report's page, you will find a drop-down menu that allows you to Edit, Delete, Print or Reorganize your report page.



By clicking on the Delete icon, you will delete the current page, after confirming your choice.

By clicking on the Edit icon, you will open the Page Editor.



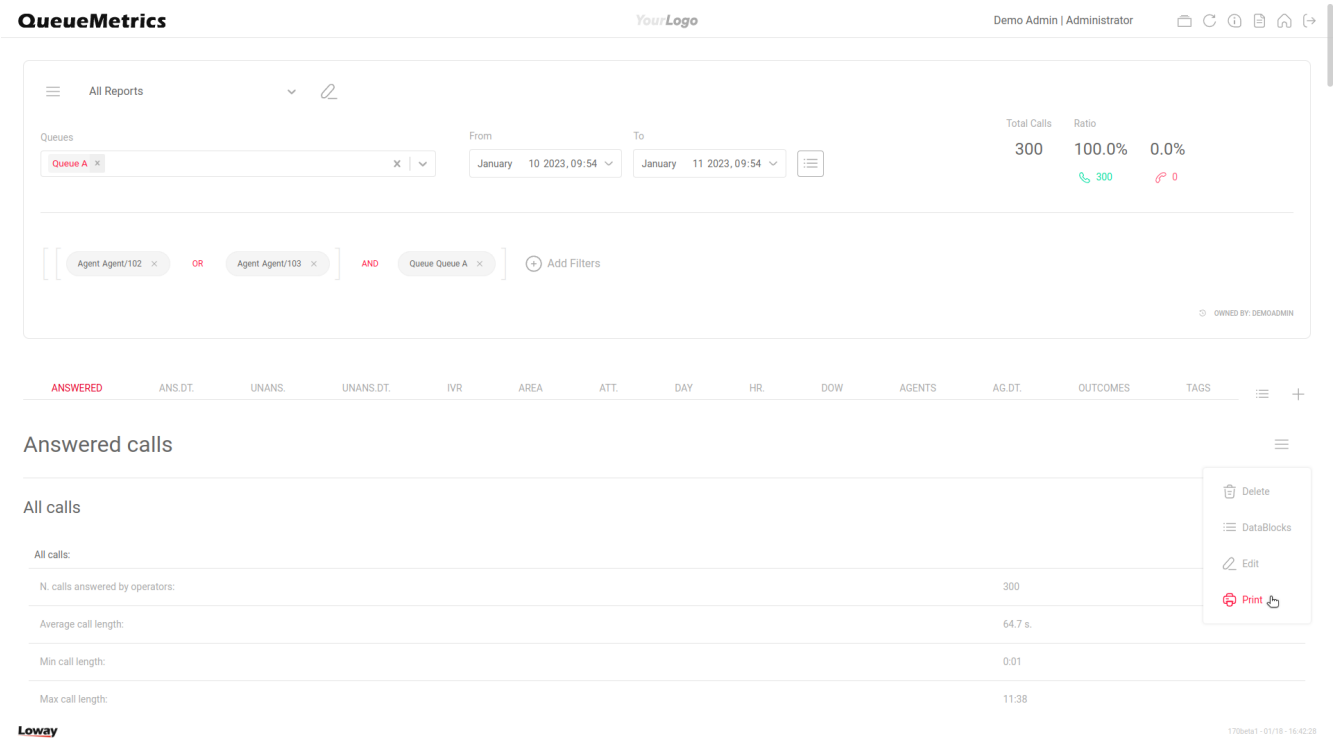
The Page Editor allows the user to edit the page information.

As shown, you can edit the following information:

- Page Title
- Page Short Title (Shown in the tab selectors)
- Visibility Key

QueueMetrics Print Report Page Feature

Now QueueMetrics allows users to print a report page. The printed report page, as opposed to a normal Report Export, contains all of the Report page's charts as well as its DataBlocks.

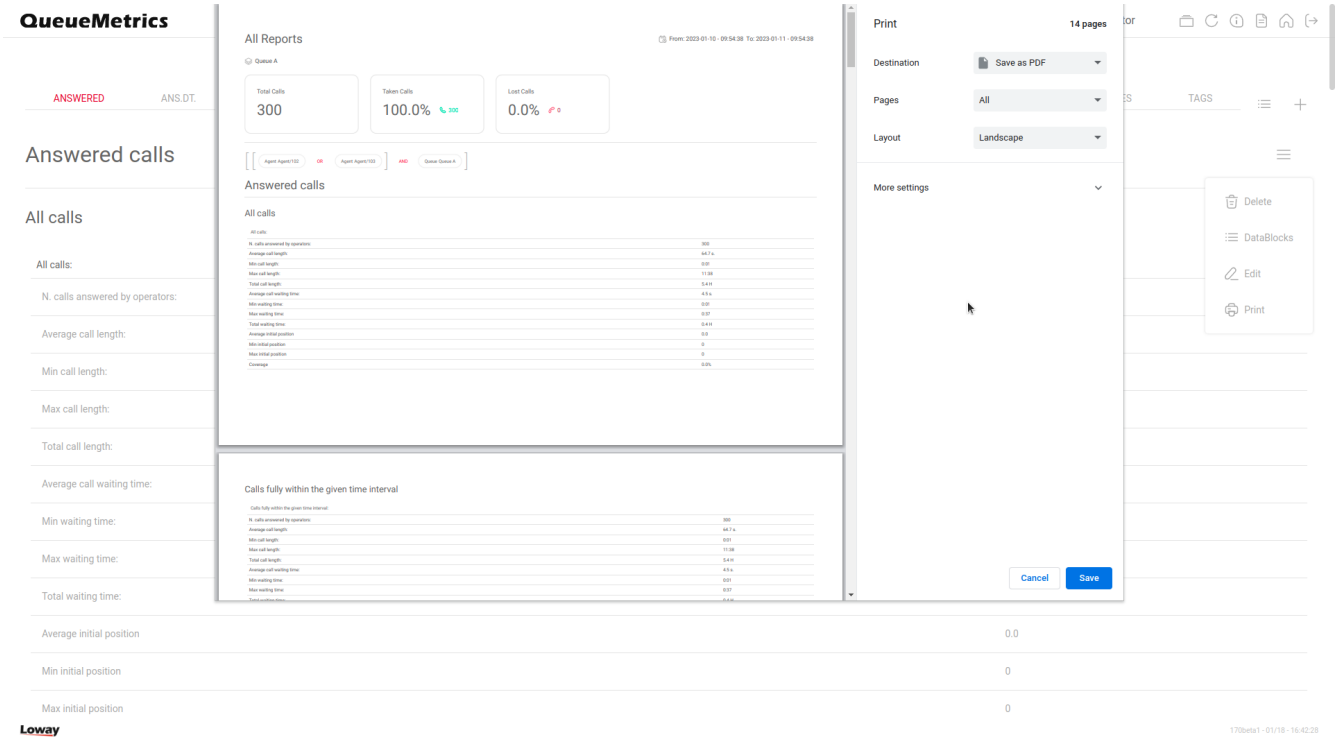


Print mode

To effectively print a report page, QueueMetrics must perform a few operations before opening the print dialogue. Namely, parts of the usual QueueMetrics GUI must be hidden from view, while a print-friendly recap of all the filters and report parameters will be added at the top of the report page.

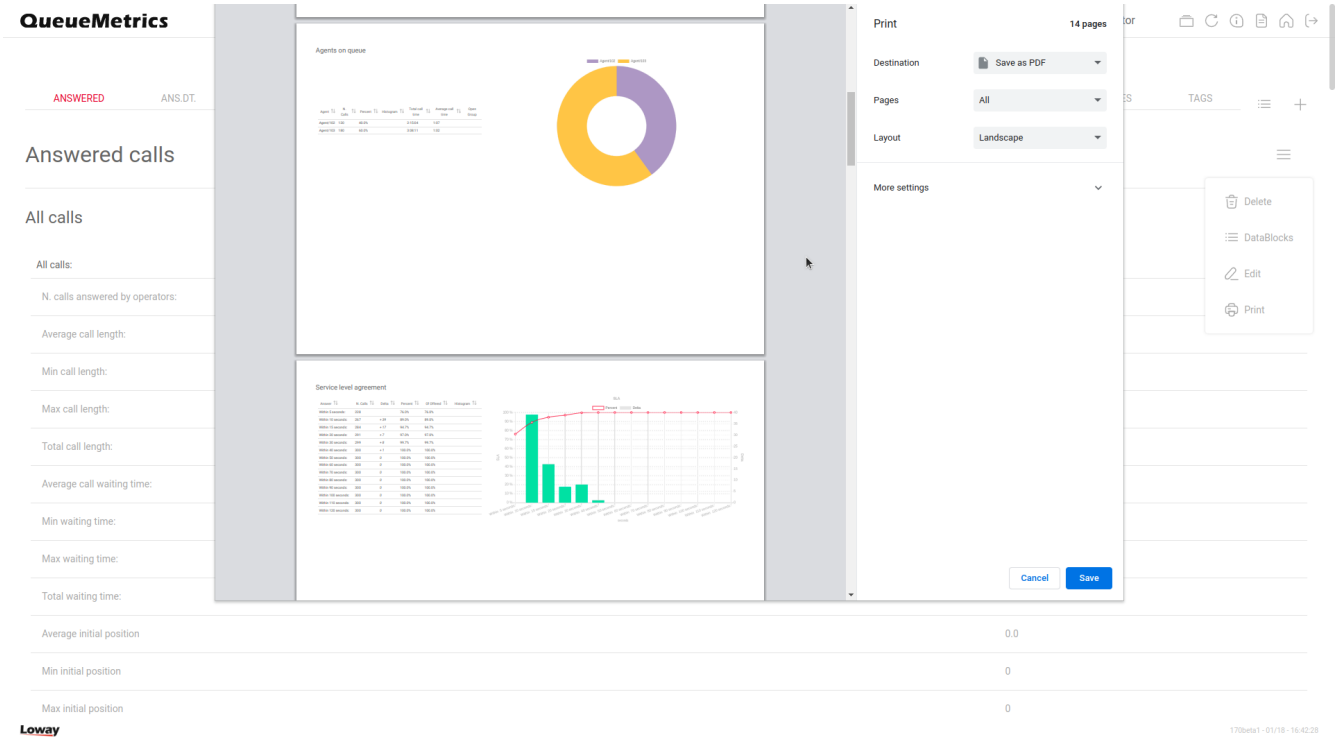
Most importantly, all DataBlocks must be shown without pagination. Please note that this means that if your report contains a lot of calls, it could take a while to load in print mode, and we recommend against printing reports with more than a hundred calls, especially if you're trying to print the details of said calls.

Once QueueMetrics has completed all the necessary preparations, it will open the Print Preview panel of your browser, where you can finalize the settings for the printing of the current report page.



We recommend printing the document in Landscape mode instead of Portrait, for better readability.

Every DataBlock will break the current page and begin a new one, to avoid tables being cut in half as much as possible. The font-size is set at 12pt, following printed documents readability guidelines for Google Chrome and Mozilla Firefox.



Due to Firefox's issues with handling web pages that include Flexboxes during printing, the charts in Firefox will be displayed under the relevant table instead of on the side. Chrome will display them on the side instead.

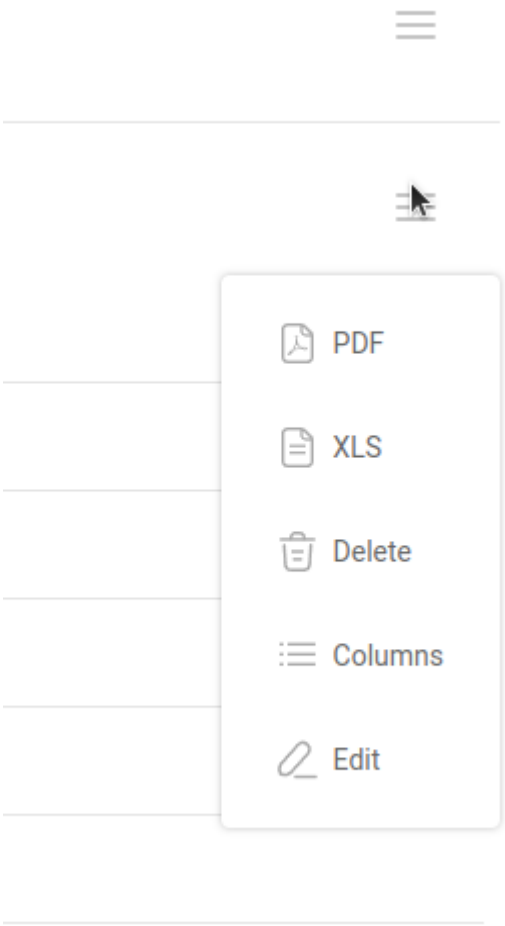
Once the preview is generated, the document can be printed or saved as PDF.

Exiting print mode

After printing the document or cancelling the preview page, QueueMetrics will take a few seconds to restore the report to its original form. This is because every table must be reloaded in paged mode, and the GUI must be restored to its default state.

DataBlock Editor

To the right side of a DataBlock, you will find a drop-down menu that allows you to edit, delete, export and reorganize a DataBlock.

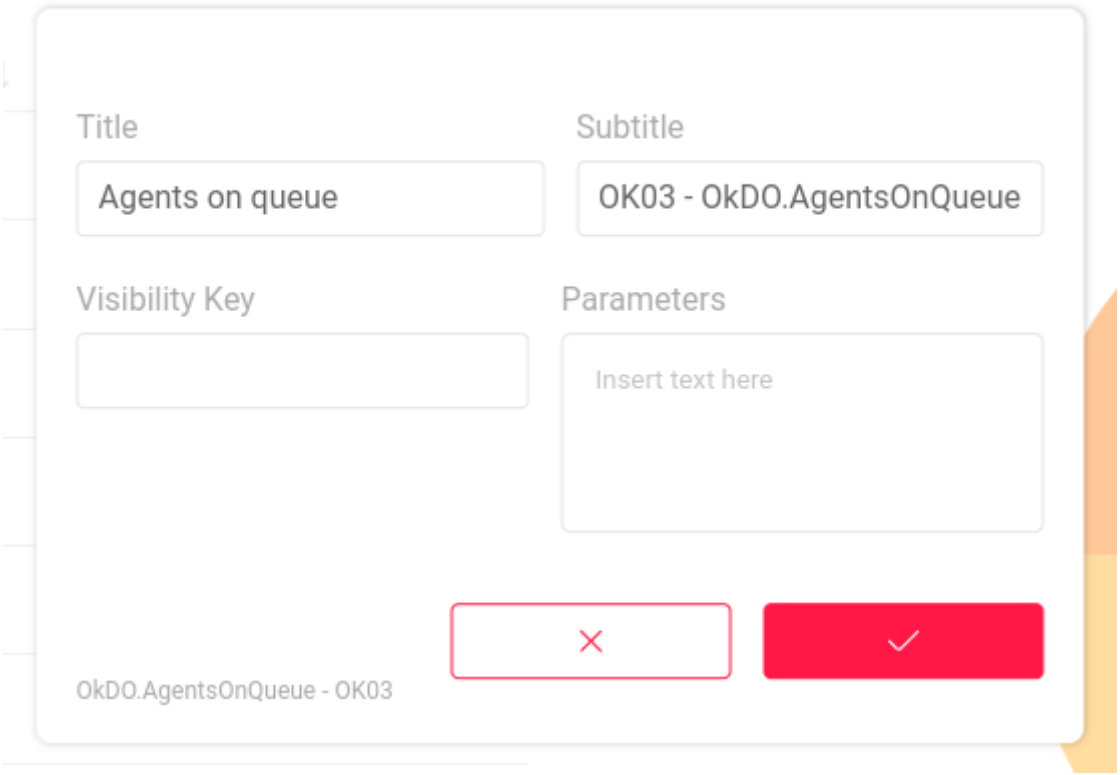


By clicking on the Delete icon, you will delete the DataBlock from the current Report page.

By clicking on the PDF or XLS icons, you will export the DataBlock in the selected format.

By clicking on the Columns icon, you will open the DataBlock columns panel, where you can hide/show and rearrange the DataBlock's columns.

By clicking on the Edit icon, you will open the DataBlock Editor panel, where you can edit the DataBlock's information.



As shown, you can edit the following information:

- Title
- Subtitle
- Visibility Key
- Parameters

Recap Row

Some DataBlocks may present a recap row at the start like the one showed in this picture:



Overview by Agent

Edit Columns Delete

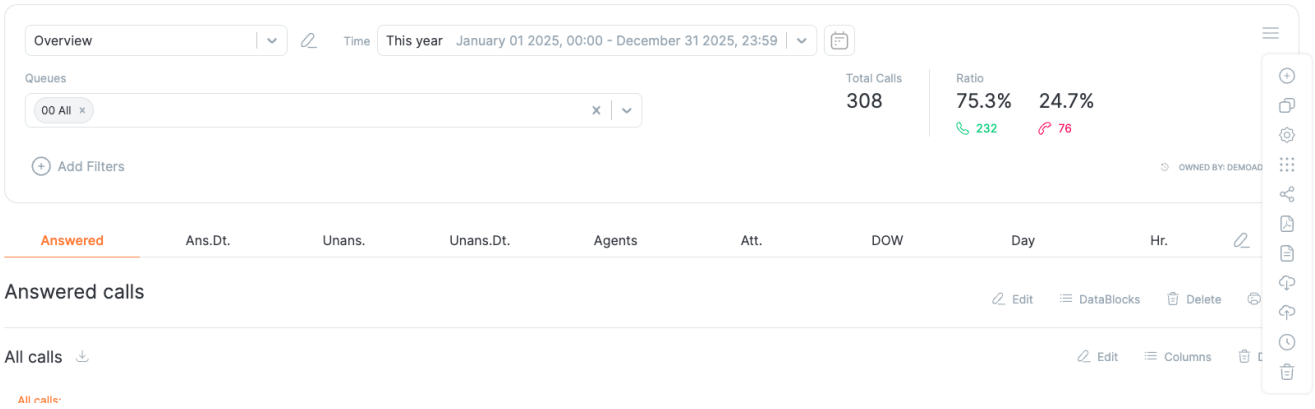
OK16 - OkDO.OverviewByAgent

Agent	Offered	Answered calls	Ans %	Lost att.	Lst att %	Outbound	Outb.%	Total Duration	Average Duration	Max Duration	ASA	Ans in SLA %	Effective %
-	**21,487**	**13,484**	**62.75%**	**8,003**	**37.25%**	**0**	**0.00%**	**11:02:56**	**1:06**	**13:15**	**0:02**	-	-
John Blackson	2,690	1,678	62.38%	1,012	37.62%	0	0.00%	31:38:48	1:07	10:21	0:03	98.20%	12.80%
Mike Bosh	2,715	1,667	61.40%	1,048	38.60%	0	0.00%	31:46:58	1:08	10:08	0:03	98.40%	13.60%
Rose Miller	2,740	1,716	62.63%	1,024	37.37%	0	0.00%	31:18:02	1:05	10:43	0:02	99.00%	13.60%
Mario Rossi	2,624	1,685	64.21%	939	35.79%	0	0.00%	31:50:10	1:08	13:15	0:03	98.50%	14.00%
Michael Fontana	2,741	1,717	62.64%	1,024	37.36%	0	0.00%	30:33:06	1:04	11:53	0:03	98.70%	12.50%
Gene Berry	2,617	1,657	63.32%	960	36.68%	0	0.00%	31:23:54	1:08	9:58	0:02	98.70%	15.20%
Bella Ferdinand	2,711	1,680	61.97%	1,031	38.03%	0	0.00%	31:25:17	1:07	10:50	0:03	98.50%	14.20%
Jim Cipriano	2,649	1,684	63.57%	965	36.43%	0	0.00%	31:06:41	1:06	10:49	0:02	98.60%	13.50%

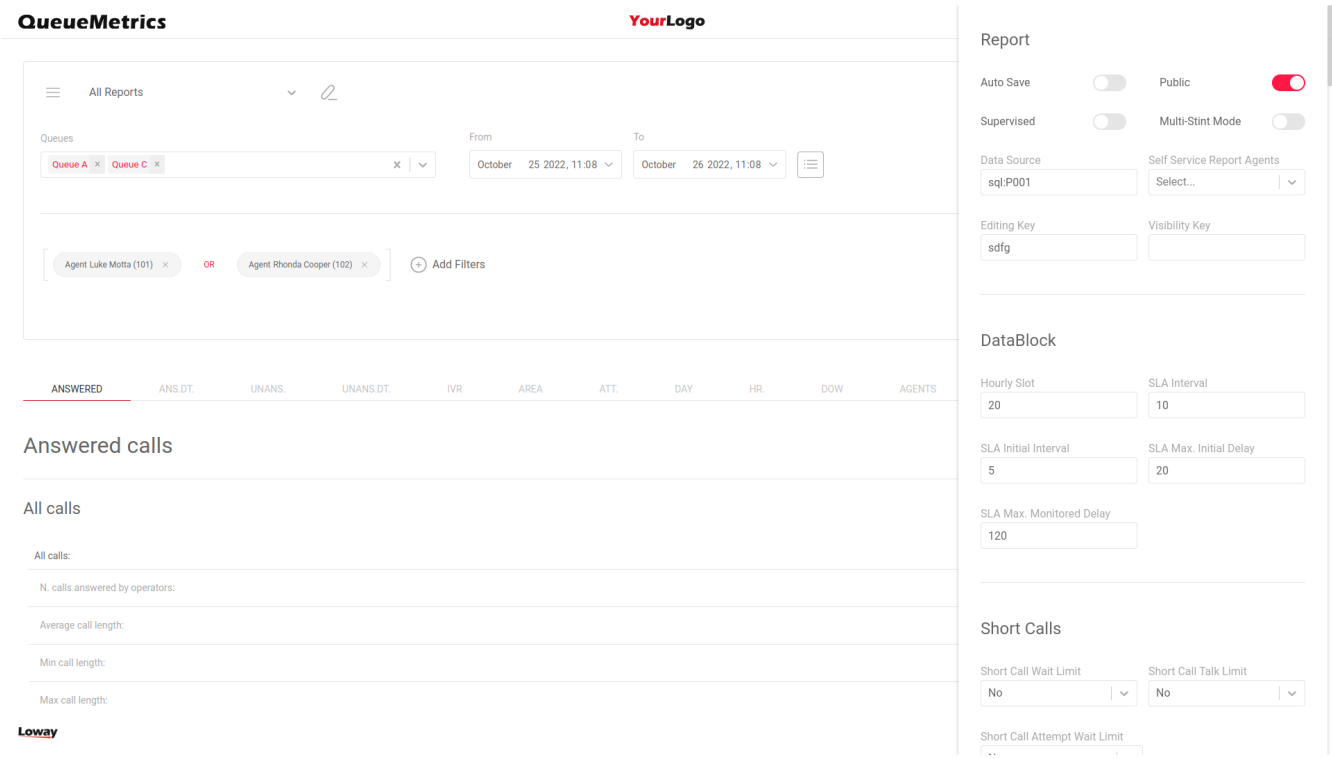
These recap rows often don't speak about any individual agent or queue, but represent a summation of the total values of every other row, in order to give an insight into the aggregation of the data displayed in the DataBlock.

Report Settings

To open the report settings, the user can click on the "Gear Icon" in the drop-down menu in the top left side of the screen.



This will open the settings menu



Here the user can change the following settings:

Table 1. Settings

Setting	Description
Auto Save	If set to yes, the report will be saved automatically after any relevant change.
Public	If set to yes, users other than the owner will be able to access this report.
Supervised	lets the user search only for agents that have the current user as their supervisor.

Setting	Description
Multi-Stint mode	Enables/disables Multi-Stint mode, joining together all calls with the same Unique Id.
Data Source	The data source for the queue_log data.
Visibility Key	Visibility Key for the report.
Self Service Report Agents	Specifies the agent that can access this report in Self Service Agent Report (SSAR) Mode.
Hourly Slot	The time interval for time distributions in Hour distribution tables.
SLA Interval	The time interval for SLA Charts and Tables
SLA Initial Interval	The initial time interval for SLA Charts and Tables
SLA Max. Initial Delay	The time interval before SLA Interval takes effect instead of SLA Initial Interval
SLA Max. Monitored Delay	The maximum time interval for SLA Charts and Tables
Short Call Wait Limit	How long must a call be in wait status before being counted as a valid call
Short Call Talk Limit	How long must a call be in talk status before being counted as a valid call
Short Call Attempt Limit	How long must a call attempt be long to count as a valid call attempt
Number of CLID digits to search	Number of CLID digits to count as prefix in Area Code calculations
Starting from position	Starting position of the Prefix in Area Code Calculations

Schedule Export Job

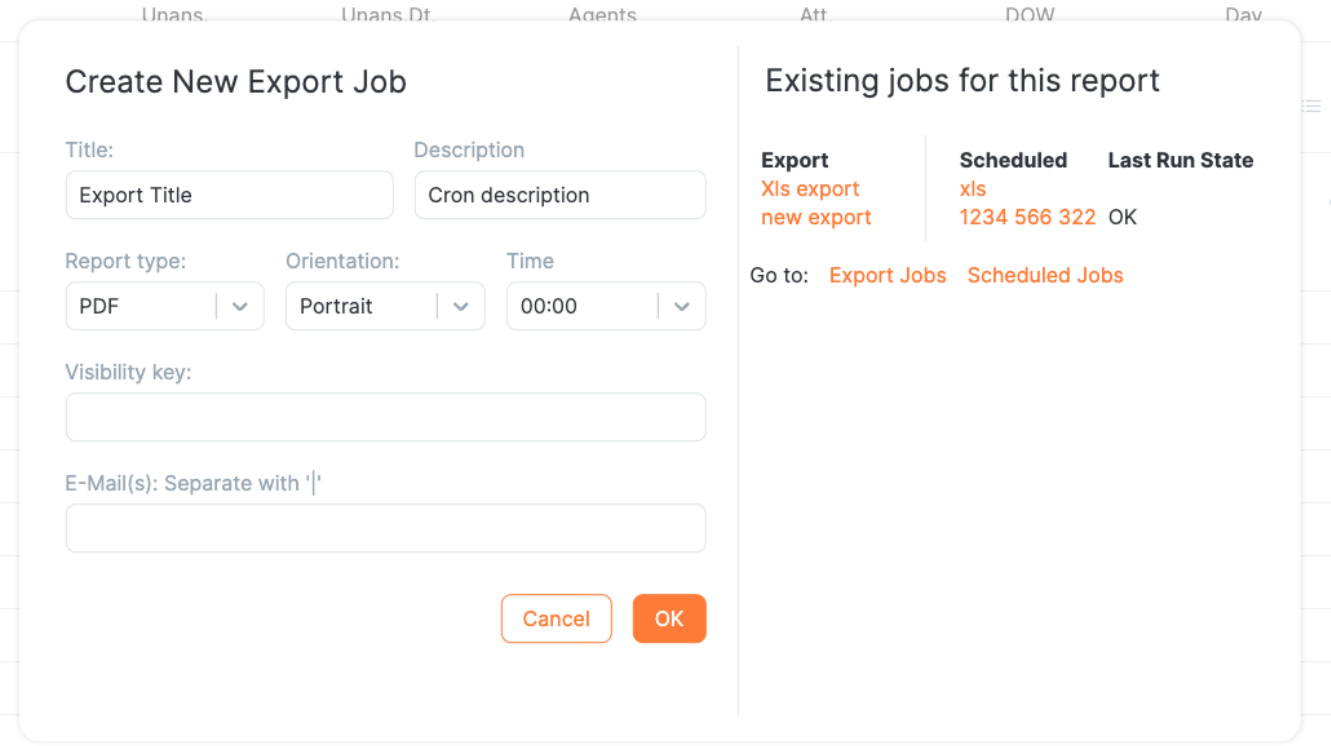
As of QueueMetrics 26.01, it is possible to create and schedule export jobs more simply within the current report by selecting the clock icon from the menu.

The popup allows you to quickly create a simplified version of both an Export Job and a Cronjob. You can specify the export title, the cronjob description, the file format, and the page orientation. Additionally, the cronjob run once per day at a selected time.

This interface also provides an overview of all existing jobs:

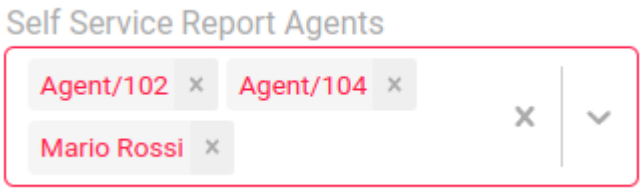
- **Status Monitoring**: View the current status of each job. Hovering over the status reveals the timestamp of the last execution.
- **Quick Access**: Clicking on a title or description redirects you directly to the full Export Job or Cronjob configuration menu for more detailed adjustments.

Only pairs consisting of both an Export Job and a linked Cronjob are displayed here. If an Export Job exists without a corresponding Cronjob, it will not appear in this list.



Self Service Agent Reporting (SSAR)

Reports can also be accessed by agents in Self Service Agent Reporting mode, by assigning specific reports to such agents in the settings panel.



To be able to access a report in SSAR mode, an agent needs to hold the following security key:

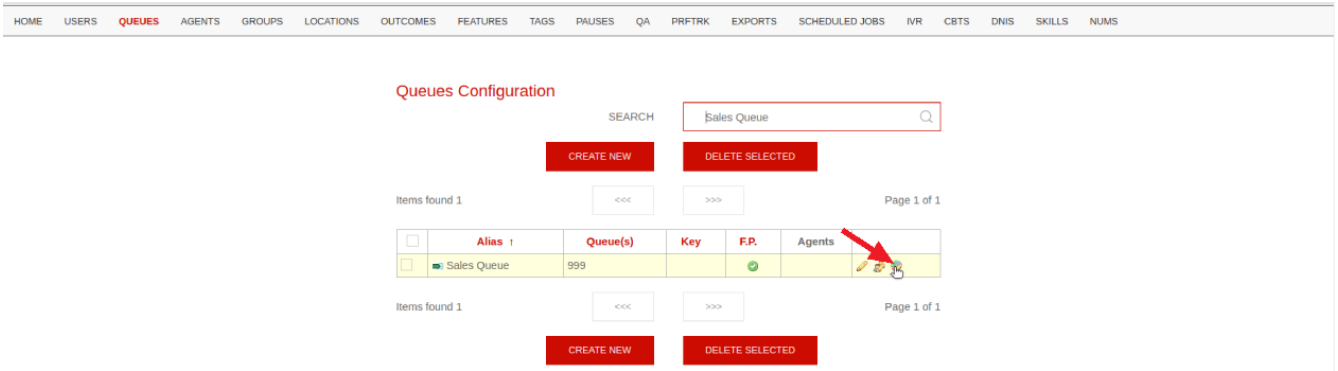
- SSAREPORT

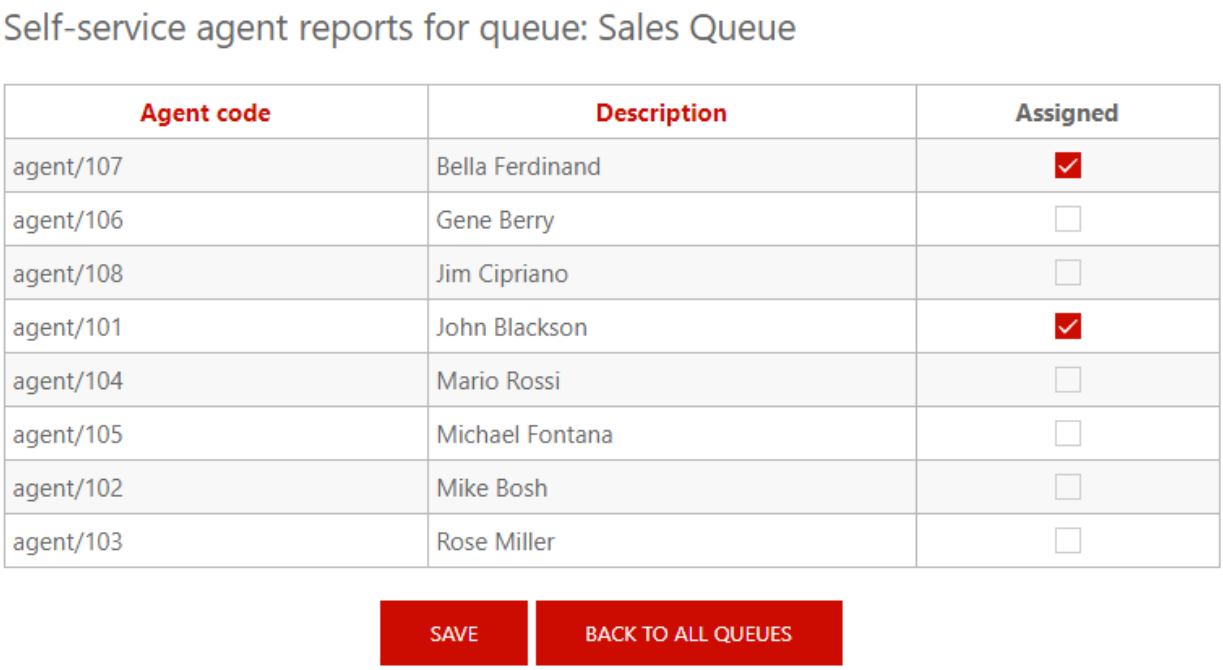
This key is usually assigned to the AGENTS class by default.

The last requirement to be able to start SSAR Reports, is that the agent needs to be assigned a set of queues to monitor.

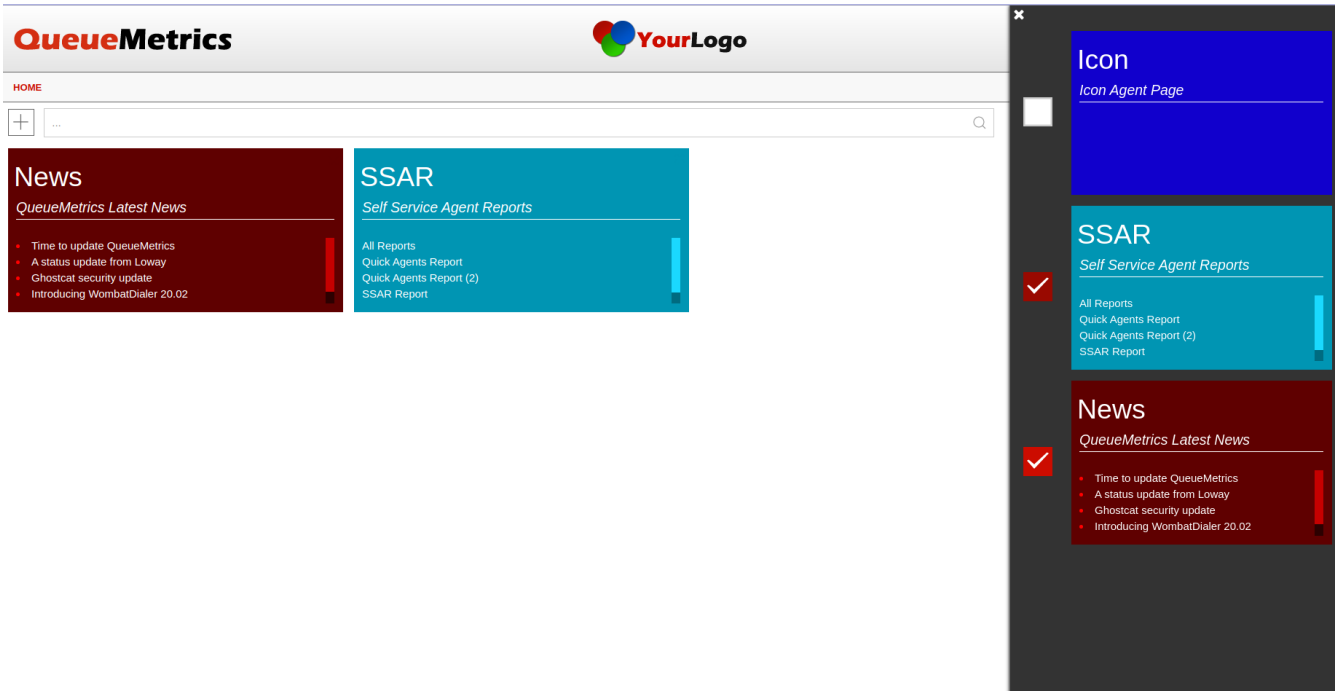
To assign a queue to an agent, a user with sufficient permissions has to navigate to the Queue Editor from the HomePage, and find the queue to assign to the agent.

Once the queue is found, then the user can assign it to the agent by clicking on the "Assign reports to agent" icon on the right side.





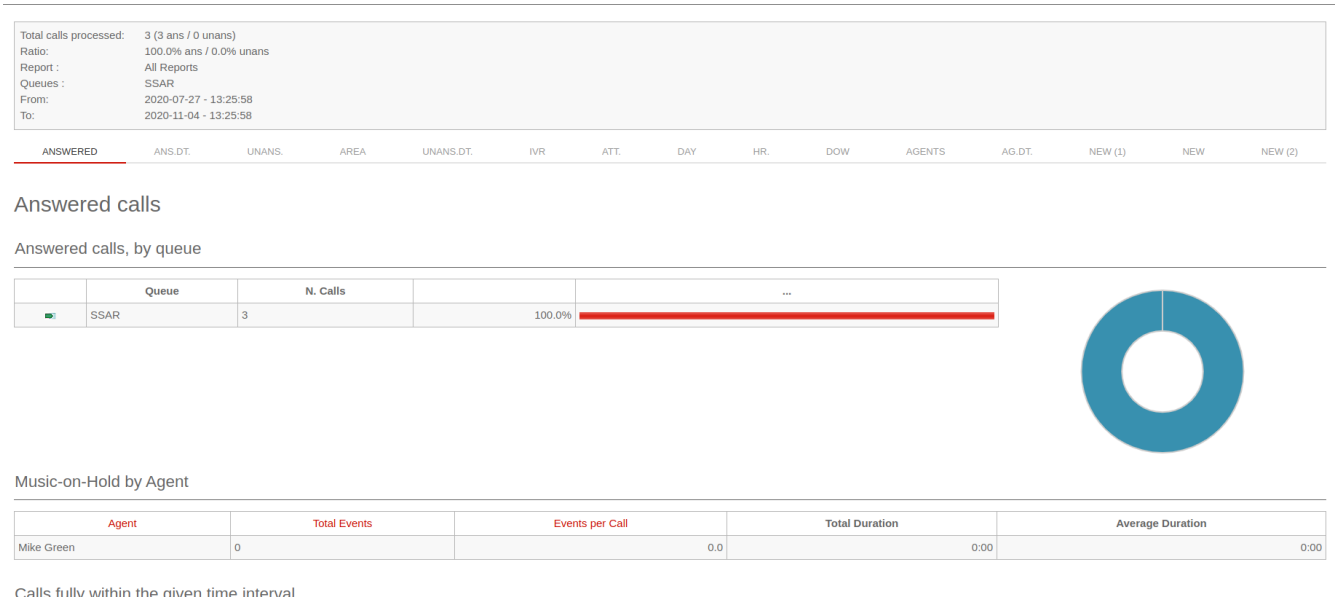
Now, if the agent accesses the Home Page, he should see the SSAR InfoBox



In the InfoBox the agent will see the Reports they have access to in SSAR Mode.

When accessing a report in SSAR mode, an agent will only be able to see the Report, without being able to edit it.

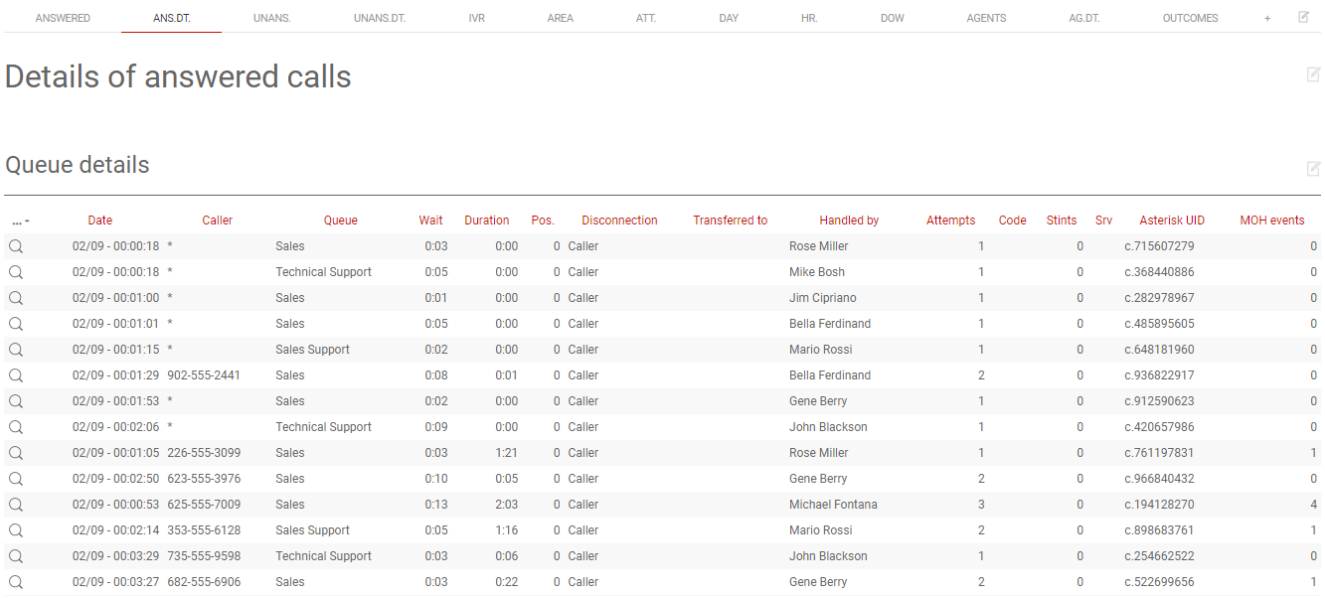
Information will also automatically be filtered to show only data relative to that specific agent.



Timeline

Introduced with QueueMetrics 22.02, the new Timeline panel replaces the old call details panel.

To access a call's Timeline panel, the user must click on the Magnifying Glass icon on the call's row, in the Queue Details (OD01) or in the Detail of unanswered calls (UD01) DataBlock.

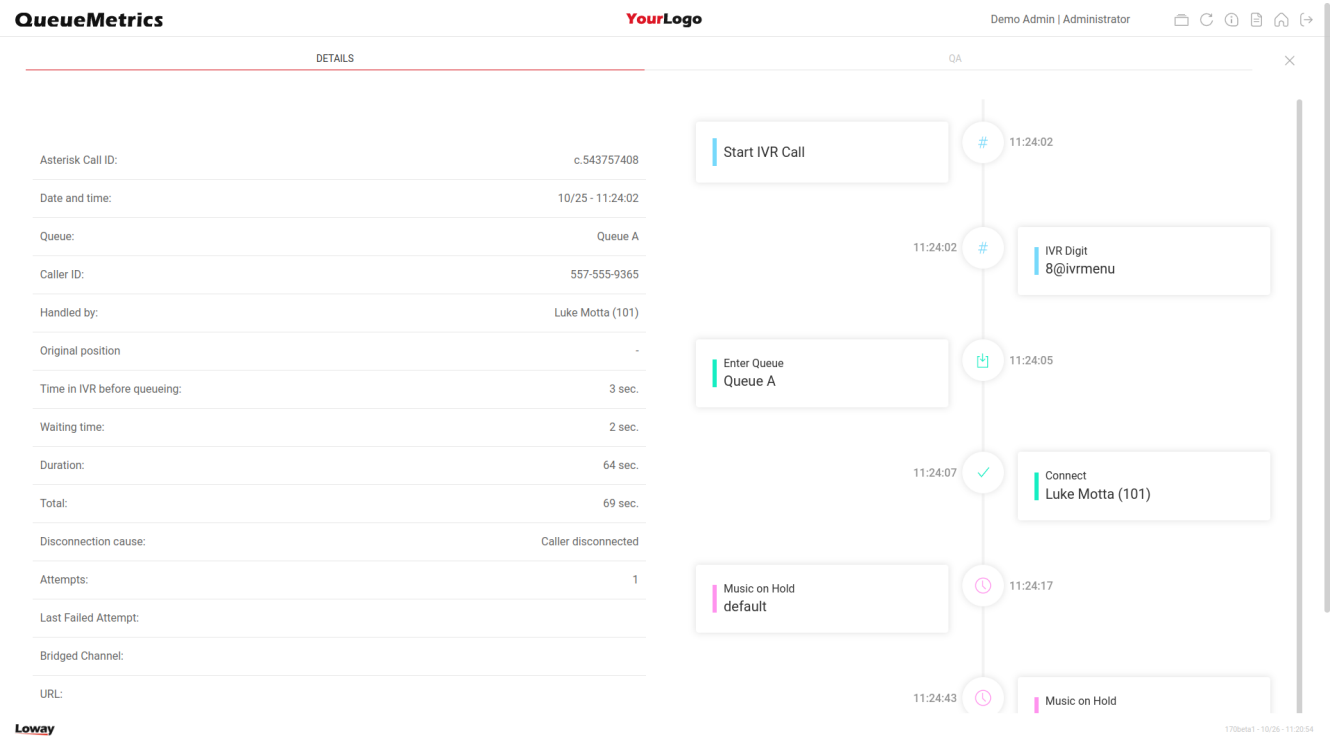


...	Date	Caller	Queue	Wait	Duration	Pos.	Disconnection	Transferred to	Handled by	Attempts	Code	Stints	Srv	Asterisk UID	MOH events
	02/09 - 00:00:18	*	Sales	0:03	0:00	0	Caller		Rose Miller	1		0		c.715607279	0
	02/09 - 00:00:18	*	Technical Support	0:05	0:00	0	Caller		Mike Bosh	1		0		c.368440886	0
	02/09 - 00:01:00	*	Sales	0:01	0:00	0	Caller		Jim Cipriano	1		0		c.282978967	0
	02/09 - 00:01:01	*	Sales	0:05	0:00	0	Caller		Bella Ferdinand	1		0		c.485895605	0
	02/09 - 00:01:15	*	Sales Support	0:02	0:00	0	Caller		Mario Rossi	1		0		c.648181960	0
	02/09 - 00:01:29	902-555-2441	Sales	0:08	0:01	0	Caller		Bella Ferdinand	2		0		c.936822917	0
	02/09 - 00:01:53	*	Sales	0:02	0:00	0	Caller		Gene Berry	1		0		c.912590623	0
	02/09 - 00:02:06	*	Technical Support	0:09	0:00	0	Caller		John Blackson	1		0		c.420657986	0
	02/09 - 00:01:05	226-555-3099	Sales	0:03	1:21	0	Caller		Rose Miller	1		0		c.761197831	1
	02/09 - 00:02:50	623-555-3976	Sales	0:10	0:05	0	Caller		Gene Berry	2		0		c.966840432	0
	02/09 - 00:00:53	625-555-7009	Sales	0:13	2:03	0	Caller		Michael Fontana	3		0		c.194128270	4
	02/09 - 00:02:14	353-555-6128	Sales Support	0:05	1:16	0	Caller		Mario Rossi	2		0		c.898683761	1
	02/09 - 00:03:29	735-555-9598	Technical Support	0:03	0:06	0	Caller		John Blackson	1		0		c.254662522	0
	02/09 - 00:03:27	682-555-6906	Sales	0:03	0:22	0	Caller		Gene Berry	2		0		c.522699656	1

By clicking on the icon, the Timeline panel will appear. This panel presents a few features that are detailed below.

Call Details

On the left side of the screen, when looking at the Timeline panel, the user can see the details of the selected call. These details are exactly the same as the ones shown in the old Call Details popup.

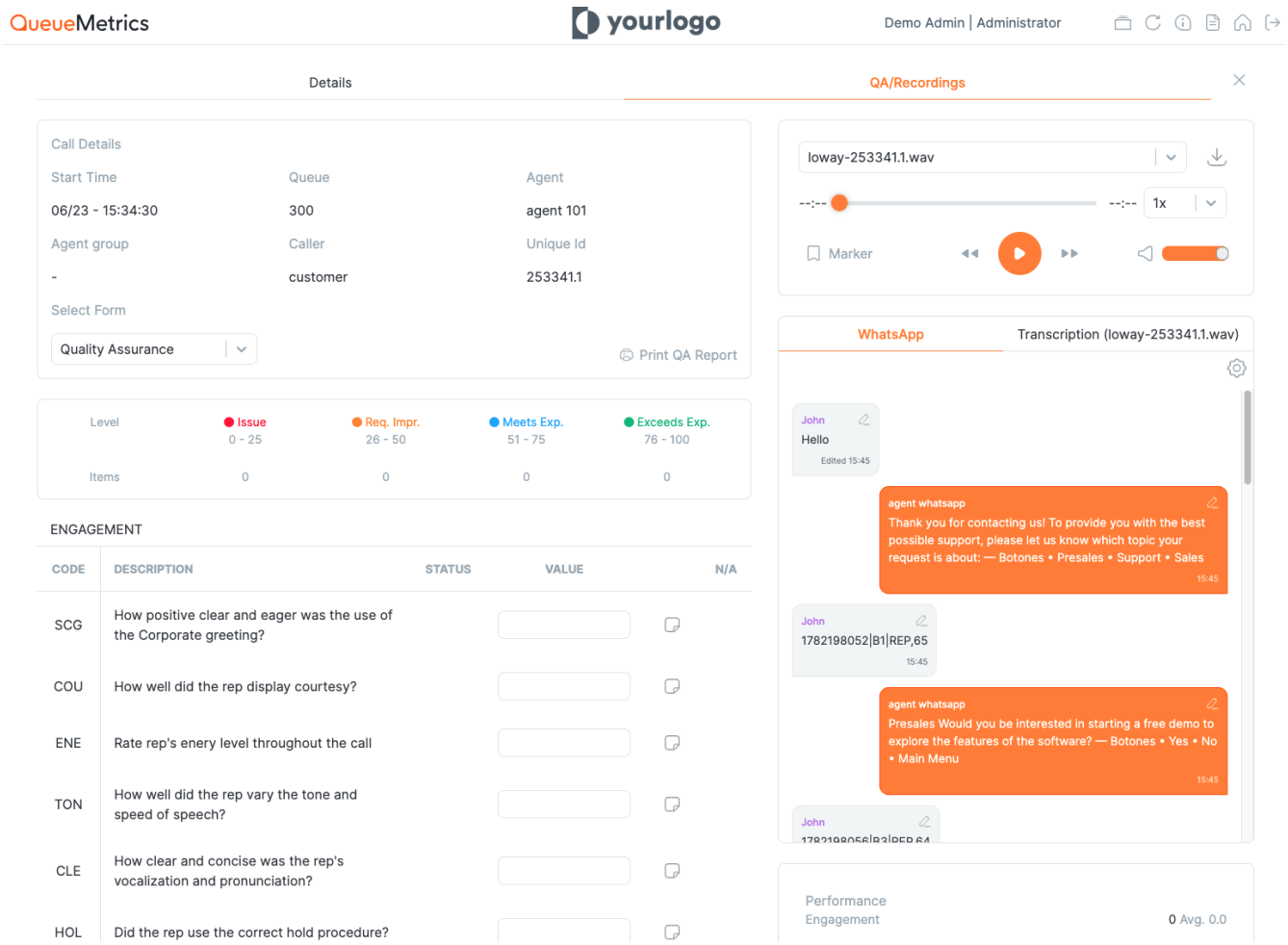


Timeline

On the right side of the screen, the user can see a vertical Timeline listing every call event in chronological order.

Each event shows the time the event happened at, the description and, depending on the case, some additional information about the event itself.

Recordings and QA

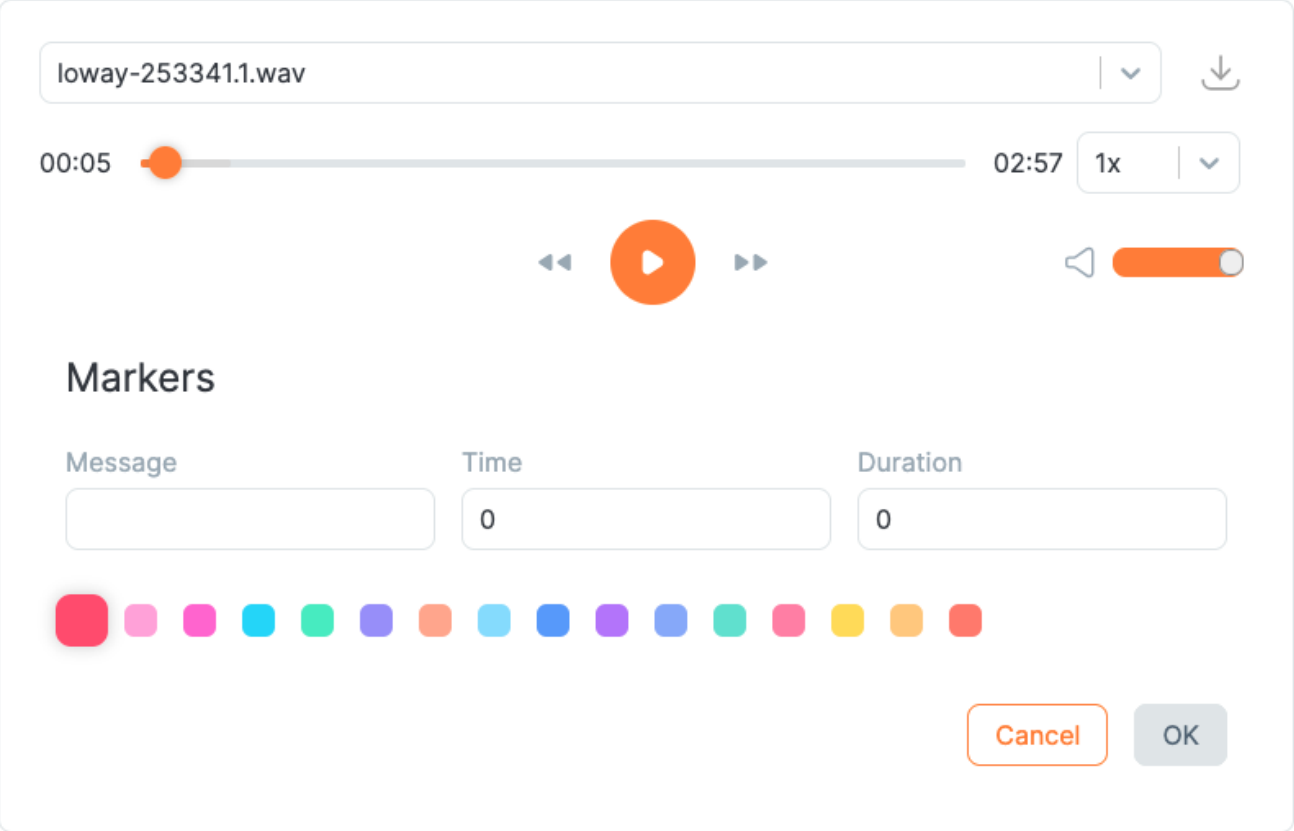


By clicking on the «QA» tab in the top left, the user can access the Quality Assurance and Audio Player tab.

Audio Player

The Audio Player in the top right will allow the user to see which audio recordings are associated with the current call. For each one of them, QueueMetrics allows the user to create markers at specific timestamps and color code them. The user can also increase the playback speed or loop the recordings.

By clicking on a saved marker, the player will skip to the noted timestamp, making navigating recordings easier.



Chat

Introduced with QueueMetrics 26.10, the Chat section provides users with access to chat conversations and call transcriptions directly from the QA section.

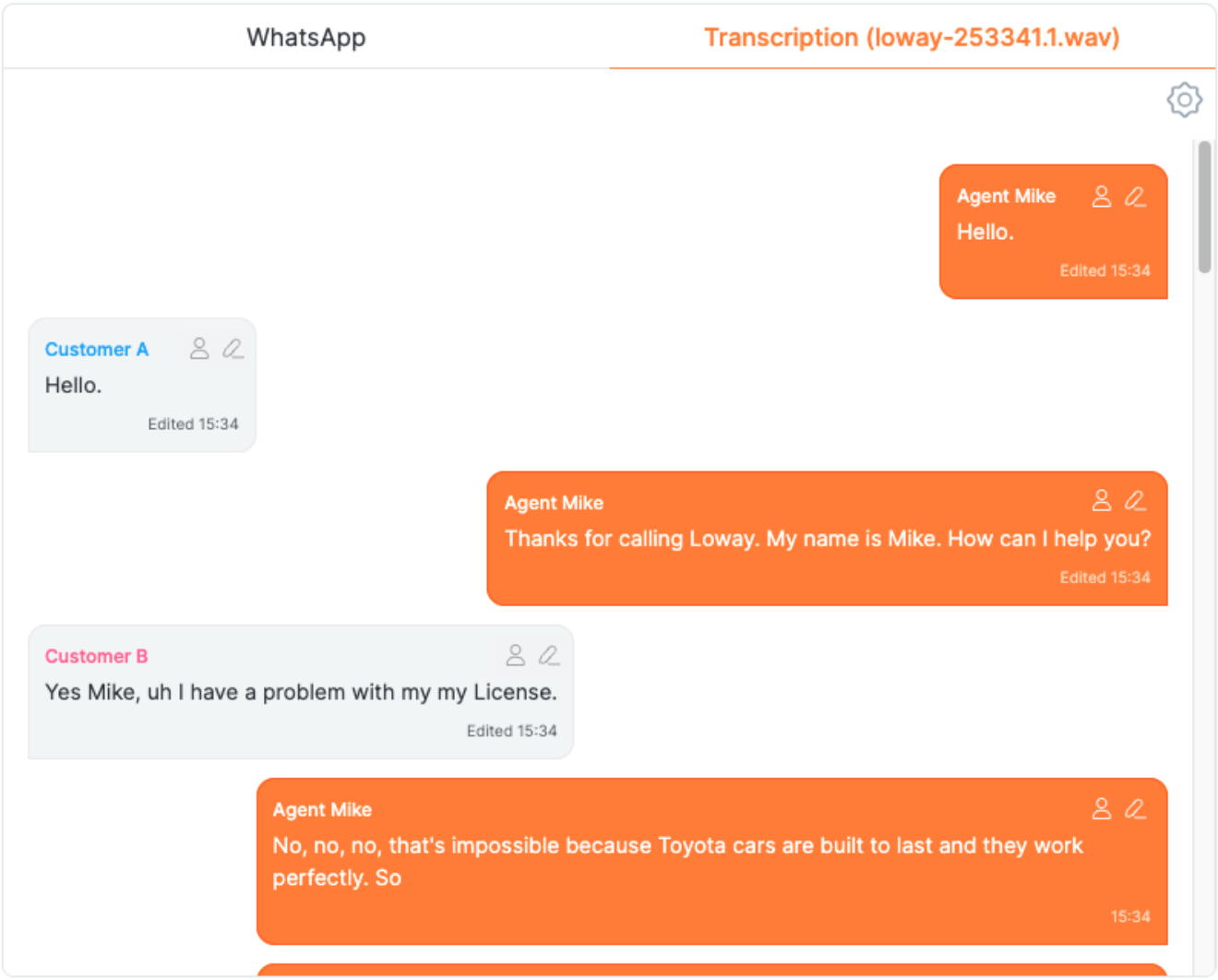
Users must possess the `CHAT_READ` permission key to view this feature.

Chat and Transcription Tabs

When accessing the QA tab, three types of tabs are available:

- **Chat Tab** - Displays chat conversations associated with the current call
- **Whatsapp Chat Tab** - Displays whatsapp chat conversations associated with the current call
- **Transcription Tabs** - One tab is created for each audio recording linked to the call that has been transcribed

Users can switch between tabs to view different chat conversations or transcriptions.



Viewing Messages

Within the Chat section, users can read all messages in the conversation thread. Each message displays:

- The timestamp of when the message was sent
- The sender's name
- The message content
- An `'edited'` label next to the timestamp if the message has been modified

Transcription Message Synchronization

When viewing transcription messages, clicking on any message will automatically synchronize the audio player to the exact timestamp of that message in the corresponding call recording. The audio player will skip back by 0.5 seconds from the message timestamp, allowing users to hear the context leading up to the transcribed text.

This feature enables seamless navigation between the transcription text and its corresponding audio without manually seeking through the recording.

Editing Messages

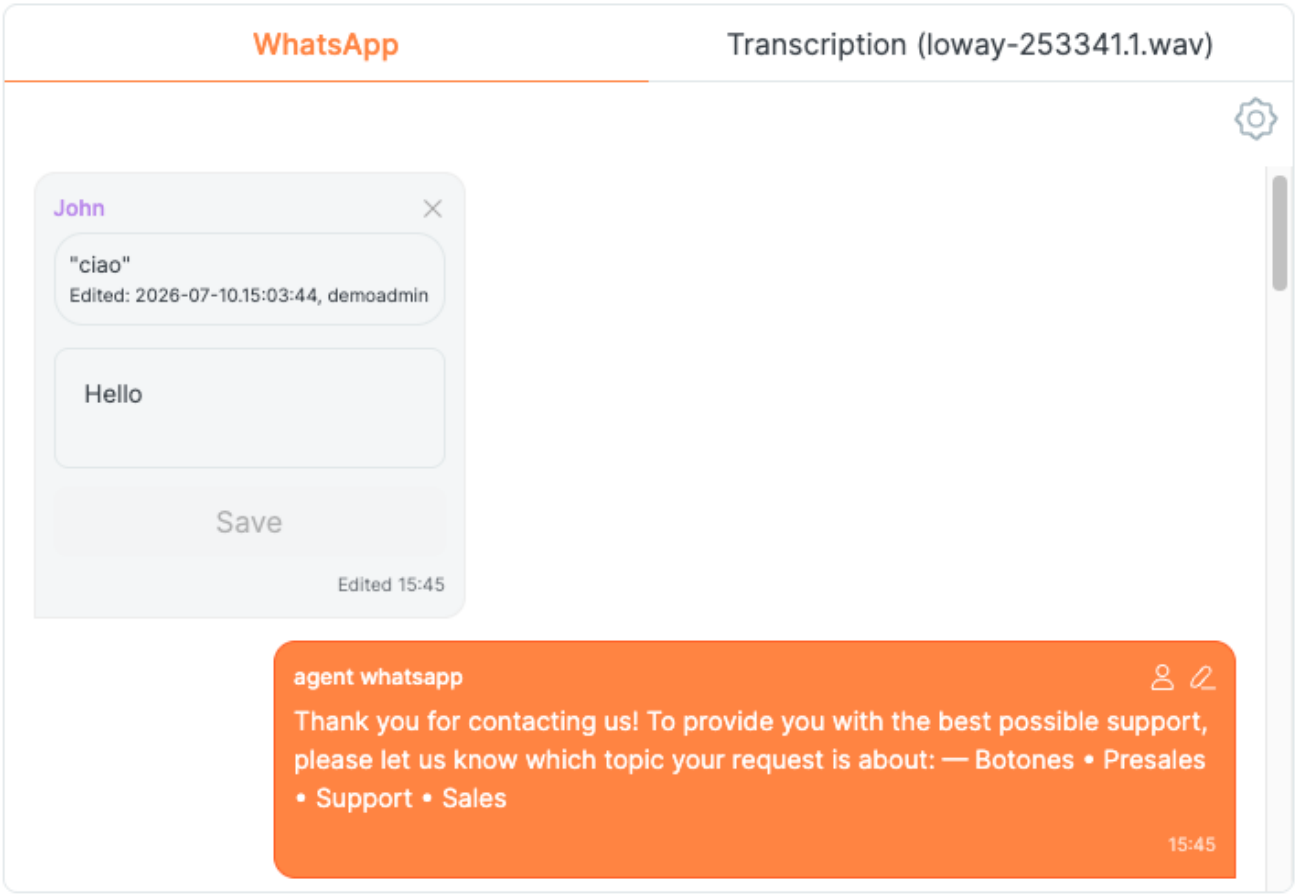
Users with the `CHAT_EDIT` permission can correct the text and the speaker of a message directly from the chat view.

Editing message text

To edit the text of a message click the pencil icon in the top-right corner of the message.

While in edit mode, the message displays:

- The current message text, in an editable field.
- The edit history: for each previous edit, the earlier text, the date and time of the edit, and the user who made it.
- A **Save** button, enabled only when the text has been changed and is not empty.



To leave edit mode without saving, click the close (X) icon in the top-right corner of the message.

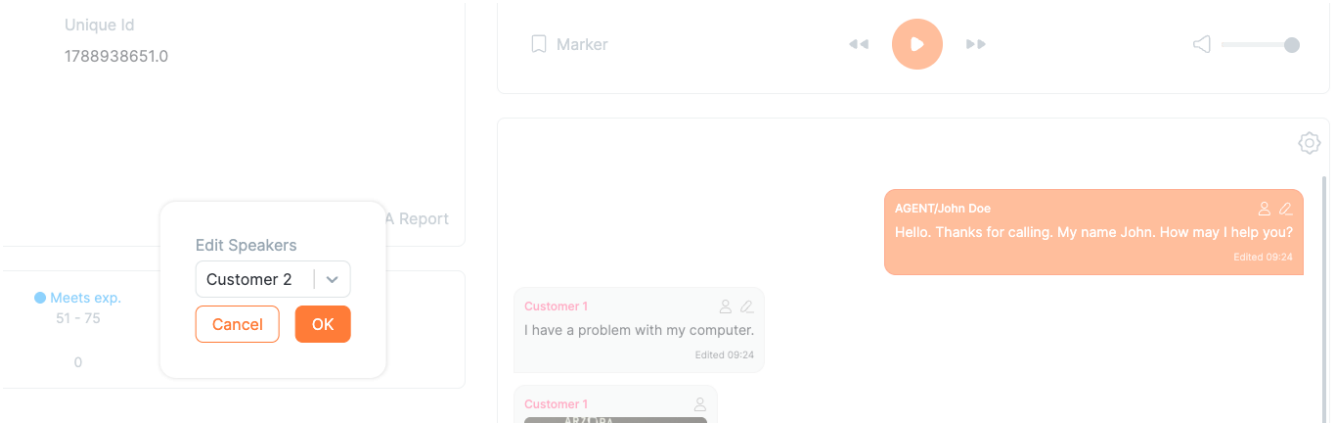
After saving, the message is marked **Edited** with the time of the change.



The pencil icon is displayed only on text messages. Other message types, such as images, cannot be edited.

Changing the speaker

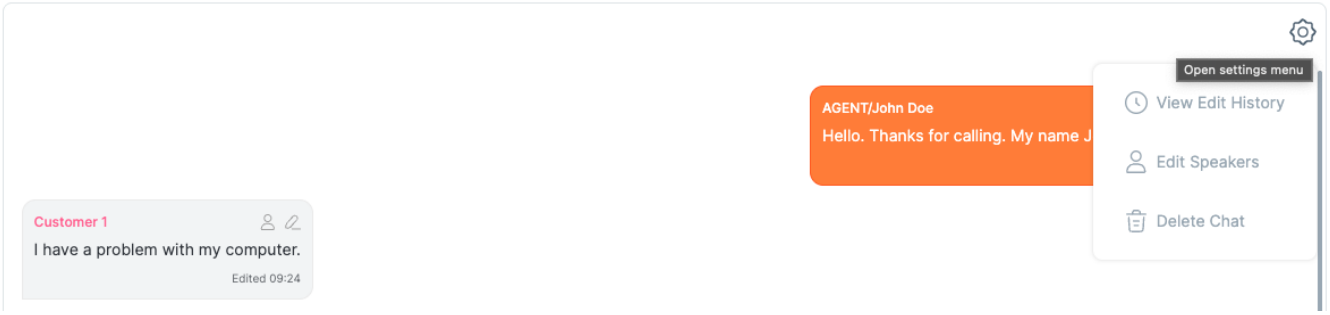
To reassign a message to a different speaker click the person icon in the top-right corner of the message. In the **Edit Speakers** dialog, select the correct speaker from the drop-down list, then Click **OK** to apply the change, or **Cancel** to close the dialog without changes.



The person icon is displayed only in chats with more than one speaker.

View History, Managing Speakers and Delete Chat

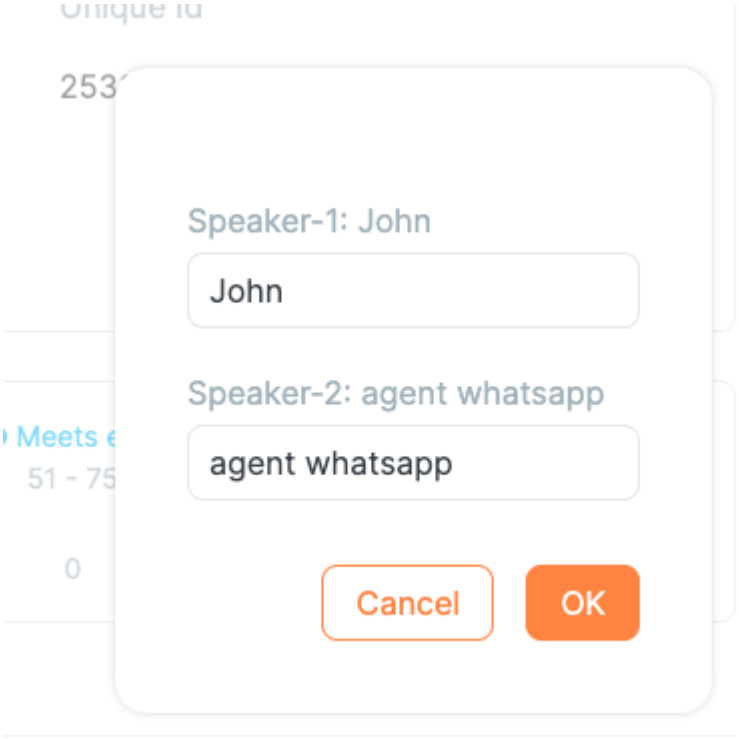
By clicking the cog icon in the top-right corner of the chat, you can choose to edit the speakers' names, view the chat's editing history or delete chat.



Managing Speakers

The gear icon in the top-right corner of the chat interface provides access to speaker management. Users with the `CHAT_EDIT` permission can:

- Click **"Edit Speakers"** to modify the names of conversation participants. Conversations may involve more than two speakers.



By default, the first speaker in the conversation is identified as the agent. If the agent speaker order is incorrect, you can correct it by renaming the agent's speaker name to one of the following:

- `"a"`
- A name starting with `"agent"`
- A name starting with `"rep"`

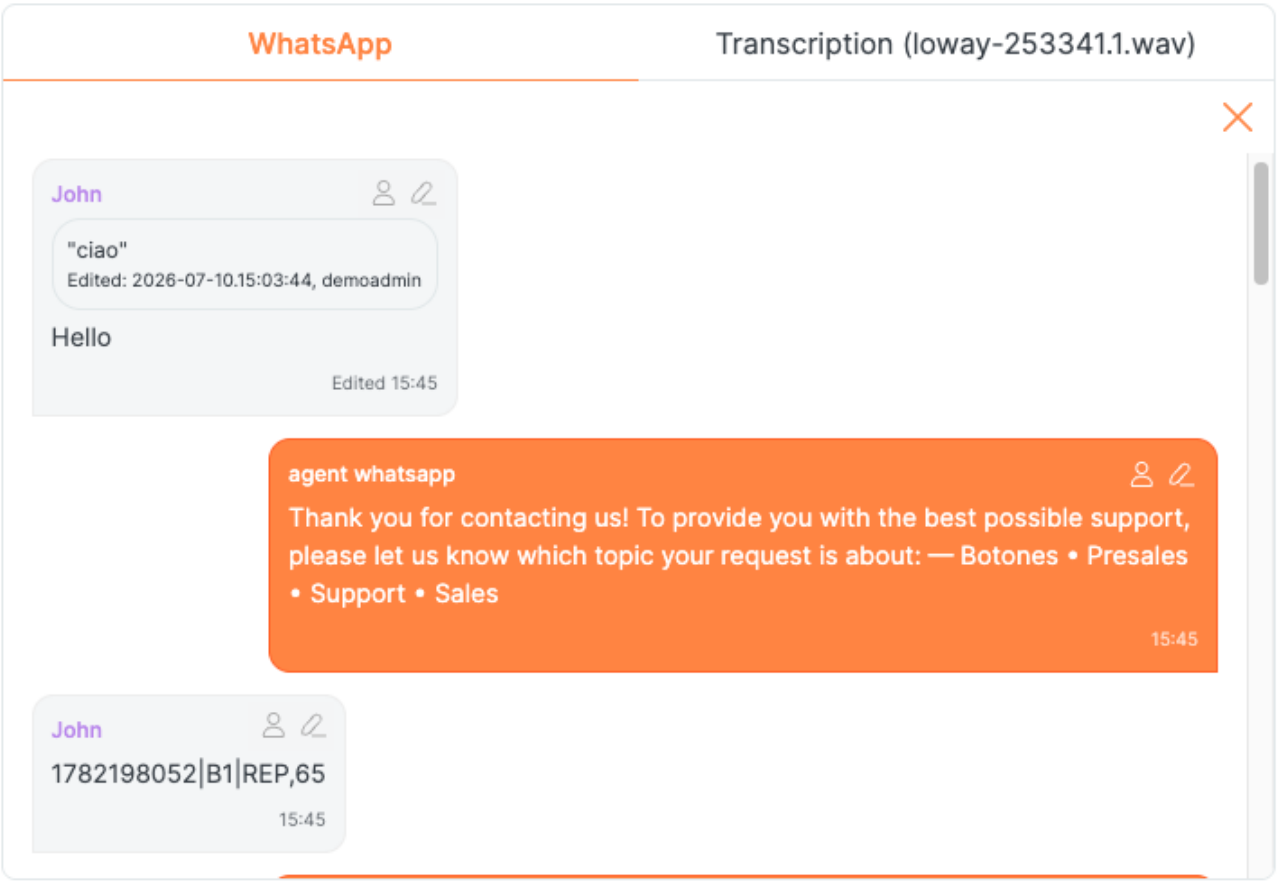
This will ensure the agent is properly identified regardless of their position in the speaker list.



Each speaker's name must be unique.

Viewing Message History

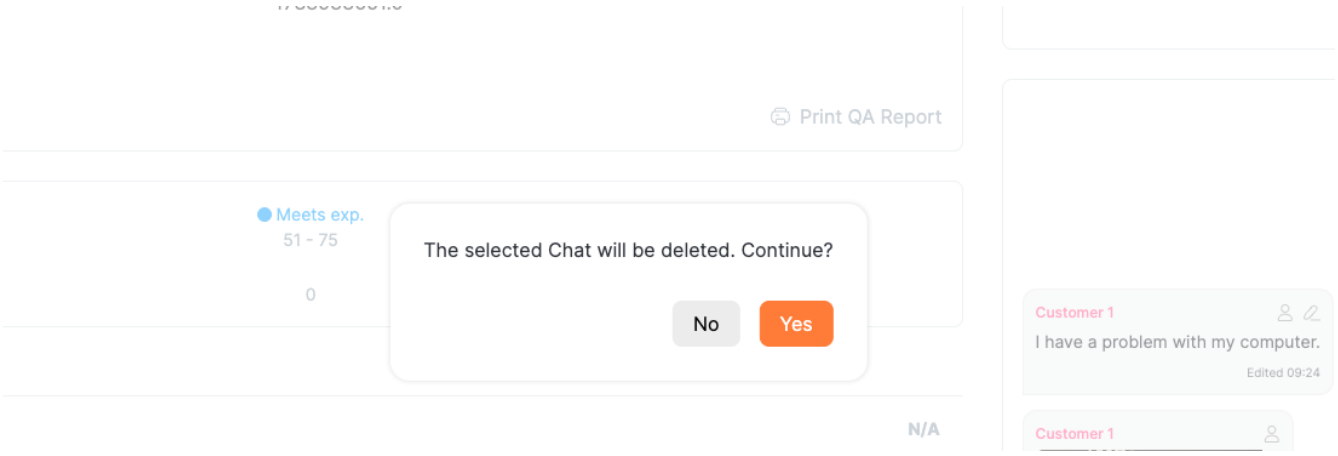
To review all modifications made to messages, users can click the **"View History"** button in the gear menu. This expands all edited messages and displays the complete history of changes applied to each message.



All modifications to chat messages and speaker names require the `CHAT_EDIT` permission key.

Chat Delete

Users with the `CHAT_DELETE` permission can delete a chat directly from the chat view by clicking the gear icon in the top-right corner of the chat panel and than clicking the **"Delete Chat"** button. In the confirmation dialog, click **Yes** to delete the chat, or **No** to cancel.



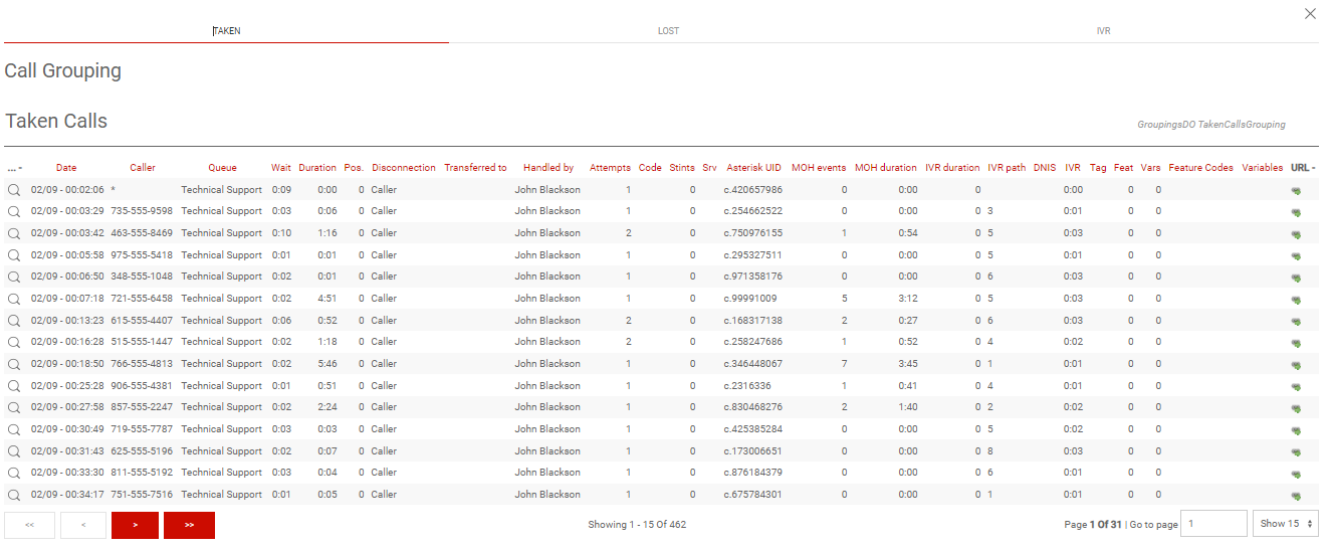
Call Grouping

QueueMetrics 22.02 introduces the concept of Call Grouping. From this version onwards, Datablocks that show metrics that includes a field to indicate the number of calls that fulfill a certain criteria, and that also show this icon:



will offer the user the ability to see tables indicating which calls make up that number.

The calls are divided into Taken and Lost calls, while a third tab is dedicated to all IVR events.



The user is also able, by clicking on the magnifying glass, to reach the Timeline panel for each of the calls listed in these tables.

Call filtering criteria

In the filter expression, a number of criteria can be specified to better zoom in on a given set of calls.

- *Agent* is a specific agent code
- *Location* is a given location
- *Supervision* lets you search only for agents that have the current user as their supervisor
- *Outcome* lets you select a call outcome
- *Call features* lets you select a call feature code
- *Call tags* lets you filter by call tag
- *Variables* lets you filter by a generic variable associated to a call (see below).
- *Asterisk call-id* search by substring on the Asterisk's UniqueID of the call
- *Caller* search by substring on the Caller-ID. By using a regular expression, this filter can also be used to match on specific *Area Codes*: for example, a filter `^..393` will match all numbers for area code 393 skipping the first two digits.
- *Wait duration* lets you specify the call waiting duration

- *Call duration* lets you specify a minimum and maximum for the call duration
- *Disconnection cause* lets you pick a disconnection reason for the call
- *Enter position* lets you enter a minimum and maximum enter position (note: this is not tracked for every call)
- *Number of attempts* lets you select a given number of attempts
- *DNIS* lets you select calls that have a specific DNIS (if tracked)
- *IVR choice* lets you select calls that have a specific IVR choice (if tracked)
- *Server* for clustered systems, lets you select only calls that were processed on a given server
- *Non-contiguous reports* let you choose the day(s) of the week and the time periods that you want to include in the reports.

Note that:

- Criteria involving a full-text search (e.g. Caller) can optionally support full Regular Expressions; if they start by "^" they will be processed as Regular Expressions. E.g. entering "^\d+$" means "find all fields which value is made up only of numeric characters, having at least one character".
- Running criteria with multi-stint calls may or may not lead to the results you are expecting. See Multi-stint calls for more information on this issue.

When a report is run with criteria set, all statistics are computed "as if" those were the only calls available; so e.g. agent sessions may yeld different results from what you would get with no criteria.

Filtering by call variables

Variables allow for very flexible filtering where you can check multiple variables at once, as in:

- "A B C" (or "A: B: C:") means that on every call all three variables must be present
- "A:12 B:" (or "A:12 B") means that on the call there must be a variable "A" set to "12" and "B" must be present.
- "A:\^x." means that the value for A is checked against the regular expression "^x." that matches for example "xy" but not "qqq"

Variables are not logged natively by Asterisk, and can be added as explained in Adding custom variables - but they can be provided automatically if you run a WombatDialer campaign.

Short calls

It is possible to filter out of the report so called "short calls", that is calls where the caller hung up before they could be meaningfully processed. Such calls inflate statistics and provide an inaccurate picture of what actually happened on the queues being processed.

QueueMetrics can filter out three types of short calls:

- *Short Call Wait Limit*: excludes calls with a wait time shorter than X seconds that ended with a caller abandon.

- *Short Call Talk Limit*: excludes taken calls with a conversation time shorter than X seconds that ended with a caller abandon.
- *Short Attempt Wait Limit*: excludes attempts to agents that were shorter than X seconds

While calls are simply excluded and will not appear in reports, if you have short attempts and look at the call details, you can still see the "short attempts" under a different event class, but these events will not be counted for attempt statistics.

Short calls can be enforced on a global basis by setting the properties `default.shortCallWait`, `default.shortCallTalk` and `default.shortAttempt`. == Understanding results

Understanding results: Answered calls

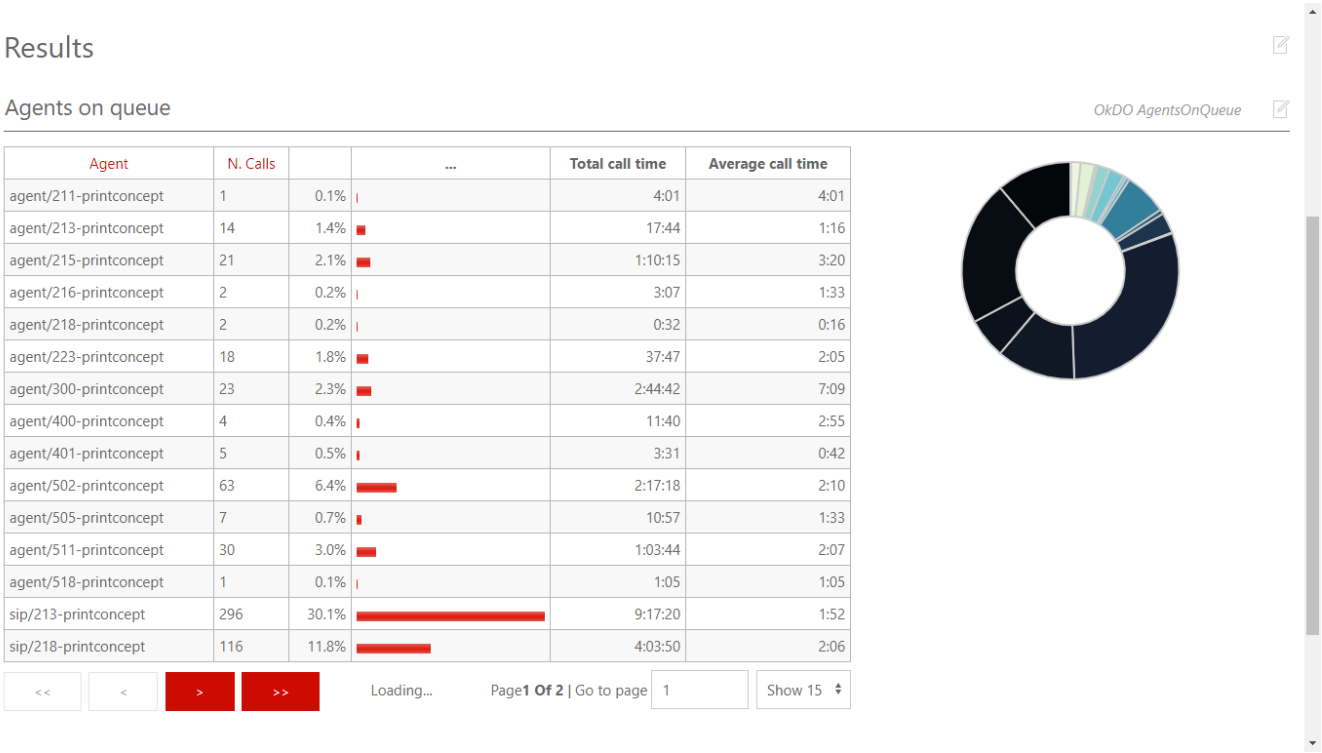
The answered calls section deals with calls that were correctly handled by agents.

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 Configuring system preferences).

It is also possible to have two time frames in order to have a higher granularity for shorter time periods - e.g. having SLA computed in intervals of 5 seconds up to 20 seconds, and in intervals of 10 seconds up to 120 seconds.

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.

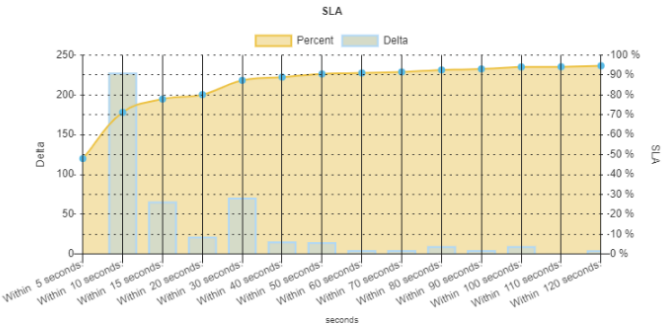
This metric is computed only on answered calls, i.e. ignoring lost calls . If your SLA is defined in terms of taken and lost calls, see the corresponding metrics "Inclusive SLA" on Inclusive Service Level Agreement.

Results

Service level agreement

OkDO ServiceLevelAgreement

Answer	N. Calls	Delta	Percent	Of Offered	...
Within 5 seconds:	473		48.0%	42.1%	
Within 10 seconds:	701	+ 228	71.2%	62.4%	
Within 15 seconds:	767	+ 66	77.9%	68.2%	
Within 20 seconds:	789	+ 22	80.1%	70.2%	
Within 30 seconds:	860	+ 71	87.3%	76.5%	
Within 40 seconds:	876	+ 16	88.9%	77.9%	
Within 50 seconds:	891	+ 15	90.5%	79.3%	
Within 60 seconds:	896	+ 5	91.0%	79.7%	
Within 70 seconds:	901	+ 5	91.5%	80.2%	
Within 80 seconds:	911	+ 10	92.5%	81.0%	
Within 90 seconds:	916	+ 5	93.0%	81.5%	
Within 100 seconds:	926	+ 10	94.0%	82.4%	
Within 110 seconds:	927	+ 1	94.1%	82.5%	
Within 120 seconds:	932	+ 5	94.6%	82.9%	



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.

Disconnection causes

OkDO DisconnectionCauses

Cause	N. Calls		...
Agent Disconnected	622	63.1%	
Caller disconnected	351	35.6%	
Transferred	12	1.2%	



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.



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.

Transfers

OkDO Transfers

Transfer to	N. Calls		...
	973	98.8%	
s	12	1.2%	



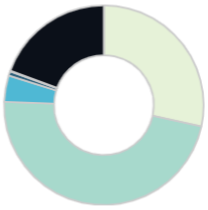
Answered calls by queue

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

Answered calls, by queue

OkDO AnsweredcallsByQueue

	Queue	N. Calls		...
	Technical Support	280	28.4%	
	Backoffice	464	47.1%	
	Finance	42	4.3%	
	Logistics	7	0.7%	
	Marketing Campaign 1	1	0.1%	
	[Q] Outbound	191	19.4%	



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 direction

OkDO AnsweredcallsByDirection

	Direction	N. Calls		...
	Inbound calls	794	80.6%	
	Outbound calls	191	19.4%	
	Any call	0	0.0%	



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.

Answered calls, by stints

OkDO StintsOk

Number of stints	N. Calls		...
1	985	100.0%	



Queue position

This graph shows the initial queue position that the calls had when they joined the queue. For example, a queue position of 1 means that a call was first in line, of 5 means that a call had four other calls in line before being answered. As the logging of queue positions is a bit inconsistent, some calls might be missing it ("Untracked")

Queue position

OkDO QPosOk

Position at enter	N. Calls		...
Untracked	191	19.4%	
1	774	78.6%	
2	17	1.7%	
3	3	0.3%	



IVR selection

This graph shows the distribution of IVR selections available in the calls processed.

This must be tracked manually in Asterisk - See "Configuring Asterisk for QueueMetrics" below.

IVR selection

KoDO IvrKo

IVR	N. Calls		...	Total Duration	Average Duration	Min Duration	Max Duration
Untracked	139	100.0%					



DNIS used

This graph shows the distribution of DNIS lines available in the calls processed.

This must be tracked manually in Asterisk - See "Configuring Asterisk for QueueMetrics" below.

DNIS used

KoDO DnisKo

DNIS	N. Calls		...
Untracked	139	100.0%	



Detail of answered calls

This page shows the detail of answered calls. See Showing call details.

Queue details

DetailsDO CallsOK

...	Date	Caller	Queue	Wait	Duration	Pos.	Disconnection	Transferred to	Handled by	Attempts	Code	Stints	Srv	Asterisk UID	MOH events	MOH duration	IVR duration	IVR path	DNIS	IVR	Tag	Feat	Vars	Feature Codes	Variables	URL
	09/07 - 08:50:56	33663767051	Technical Support	0:06	1:25	1	Caller		Adil	1				1599461434.5249809	0	0:00				0:00		0	0			
	09/07 - 08:56:46	3223615959	Backoffice	0:02	3:28	1	Agent		Anjuli	1				1599461782.5249974	0	0:00				0:00		0	0			
	09/07 - 09:42:01	32476501724	Backoffice	0:08	1:05	1	Agent		Anjuli	1				1599464498.5253131	0	0:00				0:00		0	0			
	09/07 - 09:42:13	32495569087	Backoffice	1:14	1:41	1	Caller		Anjuli	1				1599464512.5253157	0	0:00				0:00		0	0			
	09/07 - 09:50:10	32478306617	Backoffice	0:04	2:27	1	Caller		Anjuli	1				1599464968.5253997	0	0:00				0:00		0	0			
	09/07 - 09:52:19	0033698849385	[Q] Outbound	0:11	2:17	0	Agent		agent/502-printconcept	1				1599465139.5254486	0	0:00				0:00		0	0			
	09/07 - 09:57:03	3222002020	Backoffice	0:07	5:17	1	Caller		Anjuli	1				1599465401.5255096	0	0:00				0:00		0	0			
	09/07 - 10:06:04	003227155050	[Q] Outbound	0:03	0:09	0	Agent		agent/223-printconcept	1				1599465964.5257002	0	0:00				0:00		0	0			
	09/07 - 10:00:54	31356719371	Backoffice	1:40	3:44	1	Agent		Anjuli	1				1599465632.5255871	0	0:00				0:00		0	0			
	09/07 - 10:01:04	3250330270	Backoffice	5:47	1:04	2	Caller		Anjuli	2				1599465642.5255953	0	0:00				0:00		0	0			
	09/07 - 10:14:36	212676436877	Technical Support	0:12	0:20	1	Agent		Adil	1				1599466448.5258321	0	0:00				0:00		0	0			
	09/07 - 10:18:23	3250335219	Backoffice	0:04	1:03	1	Caller		Anjuli	1				1599466681.5259052	0	0:00				0:00		0	0			
	09/07 - 10:11:21	003227155050	[Q] Outbound	0:03	8:54	0	Agent		agent/223-printconcept	1				1599466281.5257927	0	0:00				0:00		0	0			
	09/07 - 10:20:11	3289772469	Backoffice	0:08	0:27	1	Caller		Anjuli	1				1599466790.5259340	0	0:00				0:00		0	0			
	09/07 - 10:27:26	3293679067	Backoffice	0:43	0:01	1	Caller		Anjuli	3				1599467224.5260715	0	0:00				0:00		0	0			

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.

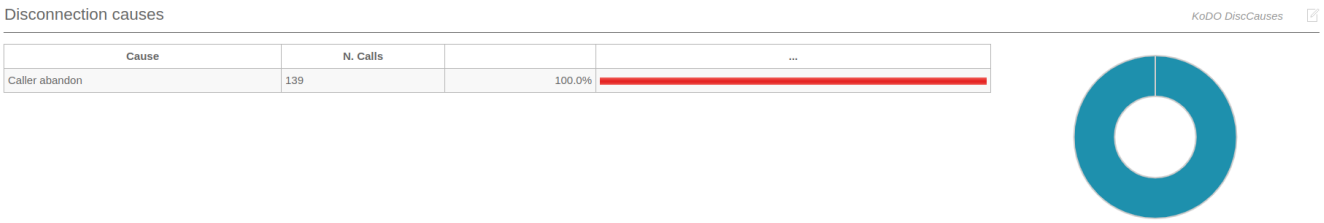


if you run a report with an agent filter, or a supervisor filter, or a location filter, the number of lost calls in the report is usually zero, showing just outgoing calls, if any. This is because the agent is specified only for taken calls and not lost ones, so not deleting them all would show, e.g., the taken call data for one single agent and the lost calls for all of the queue.

Disconnection causes

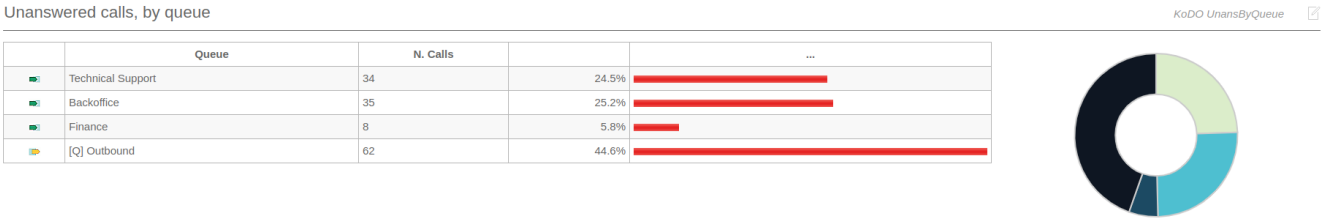
This report shows the relative magnitude of disconnection causes, that are:

- The caller hung up, or
- The queue timed out and discharged the caller (if this feature is enabled by the queue configuration - see Configuring Asterisk for QueueMetrics), 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 Service level agreement), 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 Configuring system preferences).

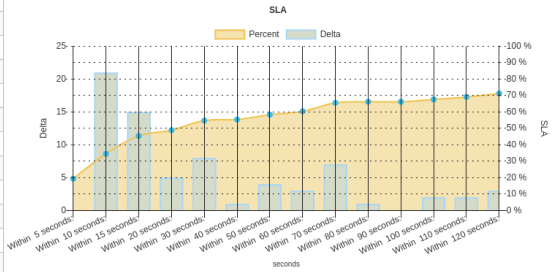
You get a percentage of how many calls were lost within X seconds; the percentage includes calls lost in a shorter time frame and therefore grows with time.

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

Unanswered calls - distribution by length

KoDO UnansByLen

Hangup	N. Calls	Delta	Percent	Of offered	...
Within 5 seconds:	27		19.4%	2.4%	
Within 10 seconds:	48	+ 21	34.5%	4.3%	
Within 15 seconds:	63	+ 15	45.3%	5.6%	
Within 20 seconds:	68	+ 5	48.9%	6.0%	
Within 30 seconds:	76	+ 8	54.7%	6.8%	
Within 40 seconds:	77	+ 1	55.4%	6.9%	
Within 50 seconds:	81	+ 4	58.3%	7.2%	
Within 60 seconds:	84	+ 3	60.4%	7.5%	
Within 70 seconds:	91	+ 7	65.5%	8.1%	
Within 80 seconds:	92	+ 1	66.2%	8.2%	
Within 90 seconds:	92	0	66.2%	8.2%	
Within 100 seconds:	94	+ 2	67.6%	8.4%	
Within 110 seconds:	96	+ 2	69.1%	8.5%	
Within 120 seconds:	99	+ 3	71.2%	8.8%	



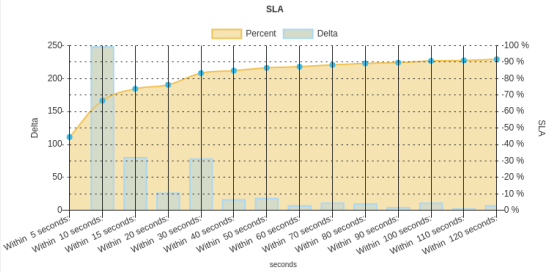
Inclusive Service Level Agreement

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

Inclusive SLA (computed on both answered and unanswered calls)

KoDO InclusiveSLA

Hangup	N. Calls	Delta	Percent	...
Within 5 seconds:	500		44.5%	
Within 10 seconds:	749	+ 249	66.6%	
Within 15 seconds:	830	+ 81	73.8%	
Within 20 seconds:	857	+ 27	76.2%	
Within 30 seconds:	936	+ 79	83.3%	
Within 40 seconds:	953	+ 17	84.8%	
Within 50 seconds:	972	+ 19	86.5%	
Within 60 seconds:	980	+ 8	87.2%	
Within 70 seconds:	992	+ 12	88.3%	
Within 80 seconds:	1003	+ 11	89.2%	
Within 90 seconds:	1008	+ 5	89.7%	
Within 100 seconds:	1020	+ 12	90.7%	
Within 110 seconds:	1023	+ 3	91.0%	
Within 120 seconds:	1031	+ 8	91.7%	



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".

Unanswered calls, by stints

KoDO StintsKo

Number of stints	N. Calls		...
1	139	100.0%	



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"

All calls, by stints

KoDO StintsOkKo

Number of stints	N. Calls		...
1	1124	100.0%	



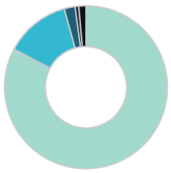
Enter queue positions

This graph shows the initial queue position that the calls had when they joined the queue. For example, a queue position of 1 means that a call was first in line, of 5 means that a call had four other calls in line before being answered. As the logging of queue positions is a bit inconsistent, some calls might be missing it ("Untracked")

Enter queue position

KoDO QPosKo

Position at enter	N. Calls		...
Untracked	0	0.0%	
1	115	82.7%	
2	18	12.9%	
3	3	2.2%	
4	1	0.7%	
5	2	1.4%	



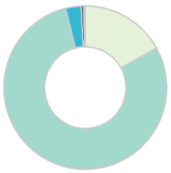
Enter queue positions for all calls

This graph shows the initial queue positions for both answered and unanswered calls.

Enter queue position for all calls

KoDO QPosOkKo

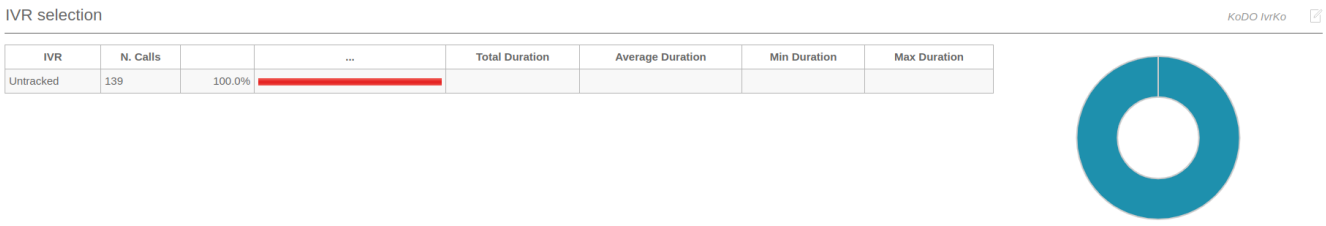
Position at enter	N. Calls		...
Untracked	191	17.0%	
1	889	79.1%	
2	35	3.1%	
3	6	0.5%	
4	1	0.1%	
5	2	0.2%	



IVR selection

This graph shows the distribution of IVR selections available for lost calls.

This must be tracked manually in Asterisk - See "Configuring Asterisk for QueueMetrics" below.

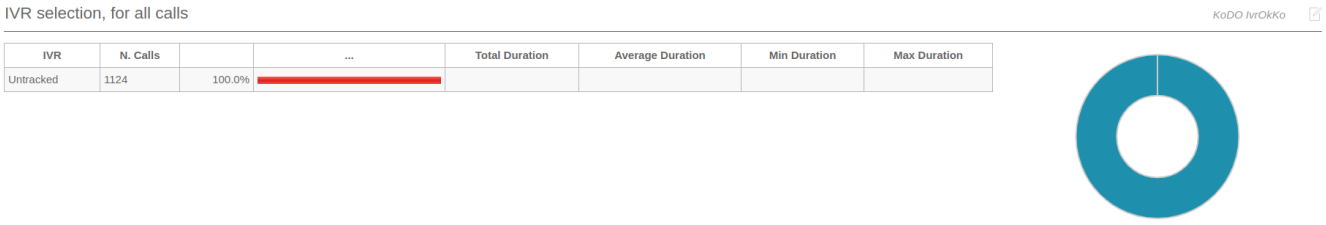

IVR selection

KoDO IvrKo

IVR	N. Calls		...	Total Duration	Average Duration	Min Duration	Max Duration
Untracked	139	100.0%					

IVR selection for all calls

This graph shows the distribution of IVR selections for all calls (taken and lost) available in the report.


IVR selection, for all calls

KoDO IvrOkKo

IVR	N. Calls		...	Total Duration	Average Duration	Min Duration	Max Duration
Untracked	1124	100.0%					

DNIS used

This graph shows the distribution of DNIS lines that lead to lost calls.

This must be tracked manually in Asterisk - See "Configuring Asterisk for QueueMetrics" below.


DNIS used

KoDO DnisKo

DNIS	N. Calls		...
Untracked	139	100.0%	

DNIS used, for all calls

This graph shows the distribution of DNIS lines for all calls (taken and lost) available in the report.

DNIS used, for all calls

KoDO DnisOkKo

DNIS	N. Calls		...
Untracked	1124	100.0%	



Details of unanswered calls

This page shows full details of unanswered calls Detail of unanswered calls.

Detail of unanswered calls

DetailsDO CallsKO

Date	Agent	Caller	Queue	Disconnection	Position	Wait	Pos.	Attempts	Code	Key	Stints	Srv	Asterisk UID	IVR duration	IVR path	DNIS	IVR	Tag	Feat	Vars	Feature Codes	Variables	URL
09/07 - 09:50:53	agent/502-printconcept	0698849385	[Q] Outbound	Agent	1	0:10	1	1					1599465053.5254292				0:00		0	0			
09/07 - 09:51:48	agent/502-printconcept	0032698849385	[Q] Outbound	Agent	1	0:15	1	1					1599465108.5254427				0:00		0	0			
09/07 - 10:35:30		212676436877	Technical Support	Agent	1	0:28	1	0					1599467705.5262372				0:00		0	0			
09/07 - 10:37:48		212676436877	Technical Support	Agent	1	1:08	1	0					1599467850.5262930				0:00		0	0			
09/07 - 10:39:42		212676436877	Technical Support	Agent	1	1:08	1	0					1599467965.5263293				0:00		0	0			
09/07 - 10:41:15		212676436877	Technical Support	Agent	1	1:52	1	0					1599468058.5263614				0:00		0	0			
09/07 - 10:44:08		212676436877	Technical Support	Agent	1	0:29	1	0					1599468233.5264310				0:00		0	0			
09/07 - 10:51:25		212676436877	Technical Support	Agent	1	0:39	1	0					1599468670.5265871				0:00		0	0			
09/07 - 11:00:42	agent/509-printconcept	003226418949	[Q] Outbound	Agent	1	0:03	1	1					1599469242.5267934				0:00		0	0			
09/07 - 11:03:41		212676436877	Technical Support	Agent	1	0:06	1	0					1599469406.5268698				0:00		0	0			
09/07 - 11:34:49		32486201514	Technical Support	Agent	1	7:38	2	0					1599471269.5275858				0:00		0	0			
09/07 - 15:13:08		33782733153	Finance	Agent	1	2:54	1	0					1599484386.35054				0:00		0	0			
09/07 - 15:48:53		32497549808	Technical Support	Agent	1	2:13	1	0					1599486507.40228				0:00		0	0			
09/07 - 17:04:02		32487395097	Technical Support	Agent	1	34:08	1	0					1599491017.52116				0:00		0	0			
09/08 - 10:41:46		32497549808	Technical Support	Agent	1	2:11	1	0					1599554480.88914				0:00		0	0			

<< < > >> Showing 0 - 15 Of 139 Page1 Of 10 | Go to page 1 Show 15

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.

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 terminal

DistrDO AcdByTerminals

Agent	N. lost	Avg ring	Ring (s)	N. Taken	Avg ring	Ring (s)
Anjuli	89	0:14	21:27	296	0:04	22:39
Petronella	7	0:12	1:30	116	0:03	6:13
Ayoub	45	0:11	8:42	0	-	0:00

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

ACD attempts by queue — *DistrDO AcdByQueue*

Queue	N. lost	Avg ring	Ring (s)	N. Taken	Avg ring	Ring (s)
Technical Support	18	0:15	4:30	280	0:03	18:25
Backoffice	250	0:13	58:13	464	0:04	34:19
Finance	0	-	0:00	42	0:03	2:31
Logistics	1	0:15	0:15	7	0:04	0:31
Marketing Campaign 1	0	-	0:00	1	0:04	0:04
[Q] Outbound	0	-	0:00	191	0:11	35:15

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.

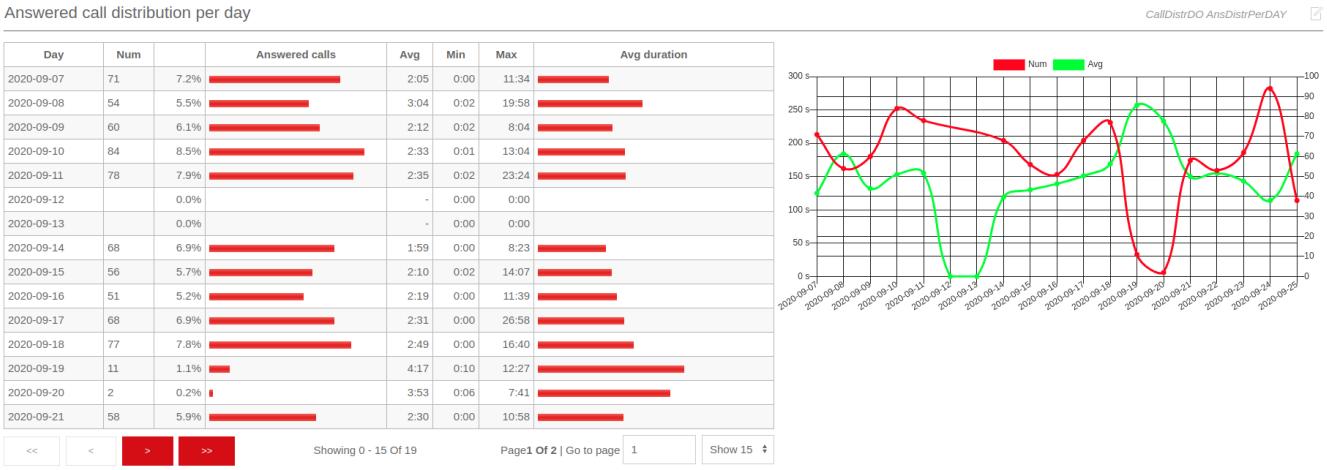
It is possible to change the interval in the Hourly graphs, so that you can have reports break down calls e.g. by half-hours or hour quarters, by changing a value in the QueueMetrics Report Settings (Hourly Slot).

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.

The Schedule Adherence report shows the number of distinct agents that were detected during the given period. This makes it possible to detect the number of different people that had been working on a given moment.

The Queue Length report shows the average length of the queue for each period, giving minimums and maximums. The Steps computation shows how fast each queue progresses during the period, expresses in steps per hour. Note: giving a meaning to the Queue Length reports may be hard in the case of composite queues.



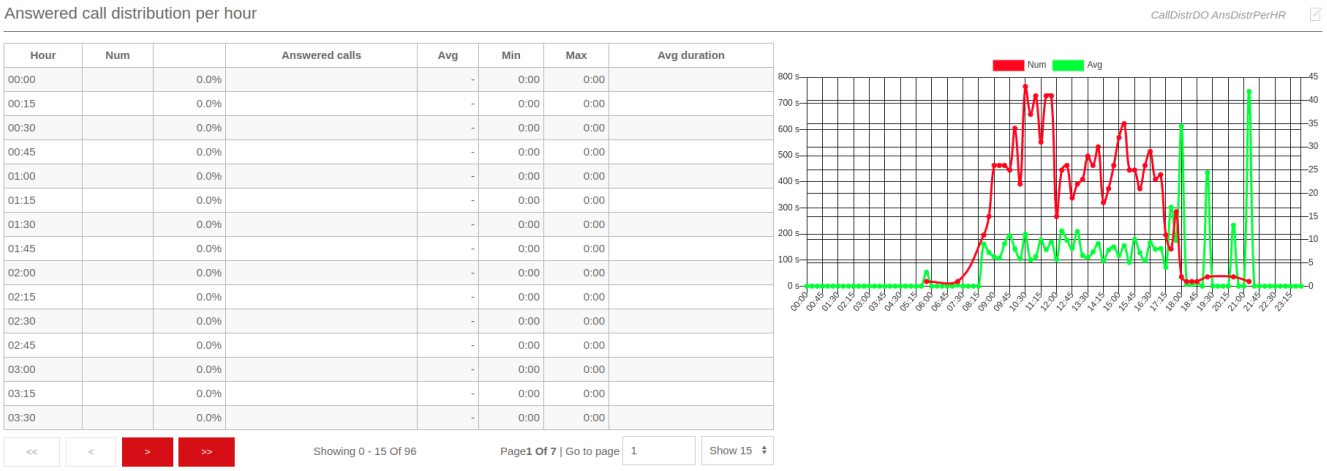
Day	Num		Answered calls	Avg	Min	Max	Avg duration
2020-09-07	71	7.2%		2:05	0:00	11:34	
2020-09-08	54	5.5%		3:04	0:02	19:58	
2020-09-09	60	6.1%		2:12	0:02	8:04	
2020-09-10	84	8.5%		2:33	0:01	13:04	
2020-09-11	78	7.9%		2:35	0:02	23:24	
2020-09-12		0.0%		-	0:00	0:00	
2020-09-13		0.0%		-	0:00	0:00	
2020-09-14	68	6.9%		1:59	0:00	8:23	
2020-09-15	56	5.7%		2:10	0:02	14:07	
2020-09-16	51	5.2%		2:19	0:00	11:39	
2020-09-17	68	6.9%		2:31	0:00	26:58	
2020-09-18	77	7.8%		2:49	0:00	16:40	
2020-09-19	11	1.1%		4:17	0:10	12:27	
2020-09-20	2	0.2%		3:53	0:06	7:41	
2020-09-21	58	5.9%		2:30	0:00	10:58	

<< < > >> Showing 0 - 15 Of 19 Page1 Of 2 | Go to page 1 Show 15

Call distribution per hour

Events are shown on a 24-hour distribution.

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.



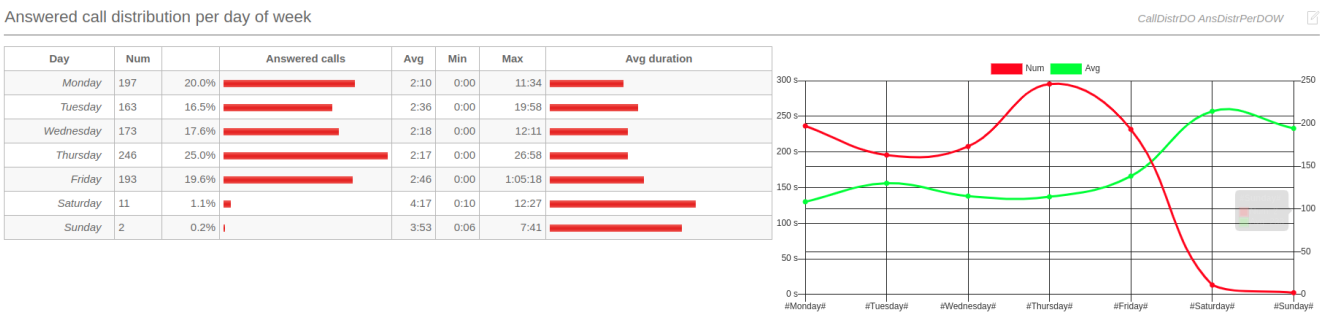
Hour	Num		Answered calls	Avg	Min	Max	Avg duration
00:00		0.0%		-	0:00	0:00	
00:15		0.0%		-	0:00	0:00	
00:30		0.0%		-	0:00	0:00	
00:45		0.0%		-	0:00	0:00	
01:00		0.0%		-	0:00	0:00	
01:15		0.0%		-	0:00	0:00	
01:30		0.0%		-	0:00	0:00	
01:45		0.0%		-	0:00	0:00	
02:00		0.0%		-	0:00	0:00	
02:15		0.0%		-	0:00	0:00	
02:30		0.0%		-	0:00	0:00	
02:45		0.0%		-	0:00	0:00	
03:00		0.0%		-	0:00	0:00	
03:15		0.0%		-	0:00	0:00	
03:30		0.0%		-	0:00	0:00	

<< < > >> Showing 0 - 15 Of 96 Page1 Of 7 | Go to page 1 Show 15

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.



Day	Num		Answered calls	Avg	Min	Max	Avg duration
Monday	197	20.0%		2:10	0:00	11:34	
Tuesday	163	16.5%		2:36	0:00	19:58	
Wednesday	173	17.6%		2:18	0:00	12:11	
Thursday	246	25.0%		2:17	0:00	26:58	
Friday	193	19.6%		2:46	0:00	1:05:18	
Saturday	11	1.1%		4:17	0:10	12:27	
Sunday	2	0.2%		3:53	0:06	7:41	

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.

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.

The "overlapping" is the amount of time that an agent was paused BUT taking calls. As you can pause in the middle of a call (maybe your system doesn't allow this, but it is conceivable) you could

have a period where you are paused BUT on conversation as well.

The % written next to the graph is the percentage of the pause time.

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

AgentsDO AgentAvail

Level	Agent	Time	On pause	Billable	Non bill.	Overlapping		...
Undefined	Anjuli	135:22:40	21:54:39	0:00	21:50:59	3:40	29.2%	
Undefined	Petronella	43:16:43	18:29:04	0:00	18:29:04	0:00	9.3%	
Undefined	Ayoub	35:17:55	79:29:31	65:59:40	13:11:53	17:58	7.6%	
Undefined	Adil	141:21:42	26:59:39	0:00	26:56:26	3:13	30.5%	
Undefined	Mohammed M.	107:36:28	20:08:23	11:04:28	9:03:55	0:00	23.2%	

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.

Session and pause durations

AgentsDO SessionPauseDur

Level	Agent	Sessions	Avg Session	Pauses	Avg Pause	Pause %	Pauses per session
Undefined	Anjuli	15	9:01:30	54	24:20	13.9%	3.6
Undefined	Petronella	10	4:19:40	19	58:22	29.9%	1.9
Undefined	Ayoub	12	2:56:29	34	2:20:16	69.2%	2.8
Undefined	Adil	20	7:04:05	179	9:02	16.0%	8.9
Undefined	Mohammed M.	11	9:46:57	60	20:08	15.8%	5.5

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 (for selected queues)

AgentsDO AnsCallsQueues

Level	Agent	N. calls	Total call time	Avg call time	Avg wait time	% of queue
Undefined	Anjuli	296	9:17:20	1:52	0:41	37.3%
Undefined	Petronella	116	4:03:50	2:06	0:19	14.6%
Undefined	Ayoub	59	4:31:36	4:36	1:23	7.4%
Undefined	Adil	213	8:22:33	2:21	0:30	26.8%
Undefined	Mohammed M.	110	5:07:42	2:47	0:27	13.9%

Answered calls by service groups

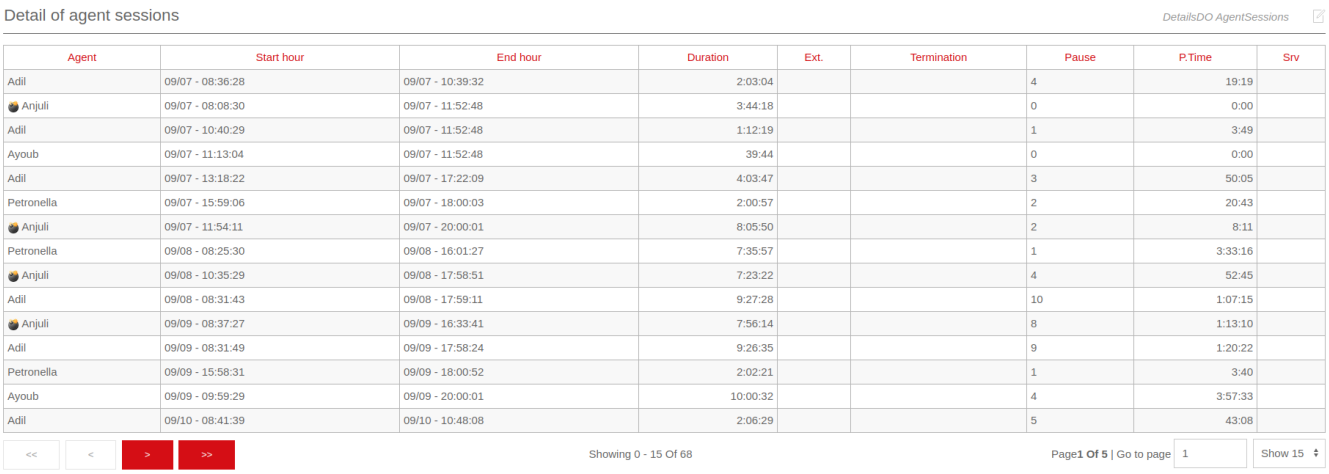
This graph show which priority levels handled calls for your queue. This shows whether your main line is staffed enough to handle the load of incoming calls.

Answered calls by service group

AgentsDO AnsCallsSG

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	794	31:23:01	2:22	100.0%

Session details



Detail of agent sessions

DetailsDO AgentSessions

Agent	Start hour	End hour	Duration	Ext.	Termination	Pause	P.Time	Srv
Adil	09/07 - 08:36:28	09/07 - 10:39:32	2:03:04			4	19:19	
Anjuli	09/07 - 08:08:30	09/07 - 11:52:48	3:44:18			0	0:00	
Adil	09/07 - 10:40:29	09/07 - 11:52:48	1:12:19			1	3:49	
Ayoub	09/07 - 11:13:04	09/07 - 11:52:48	39:44			0	0:00	
Adil	09/07 - 13:18:22	09/07 - 17:22:09	4:03:47			3	50:05	
Petronella	09/07 - 15:59:06	09/07 - 18:00:03	2:00:57			2	20:43	
Anjuli	09/07 - 11:54:11	09/07 - 20:00:01	8:05:50			2	8:11	
Petronella	09/08 - 08:25:30	09/08 - 16:01:27	7:35:57			1	3:33:16	
Anjuli	09/08 - 10:35:29	09/08 - 17:58:51	7:23:22			4	52:45	
Adil	09/08 - 08:31:43	09/08 - 17:59:11	9:27:28			10	1:07:15	
Anjuli	09/09 - 08:37:27	09/09 - 16:33:41	7:56:14			8	1:13:10	
Adil	09/09 - 08:31:49	09/09 - 17:58:24	9:26:35			9	1:20:22	
Petronella	09/09 - 15:58:31	09/09 - 18:00:52	2:02:21			1	3:40	
Ayoub	09/09 - 09:59:29	09/09 - 20:00:01	10:00:32			4	3:57:33	
Adil	09/10 - 08:41:39	09/10 - 10:48:08	2:06:29			5	43:08	

<< < > >> Showing 0 - 15 Of 68 Page1 Of 5 | Go to page 1 Show 15

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.

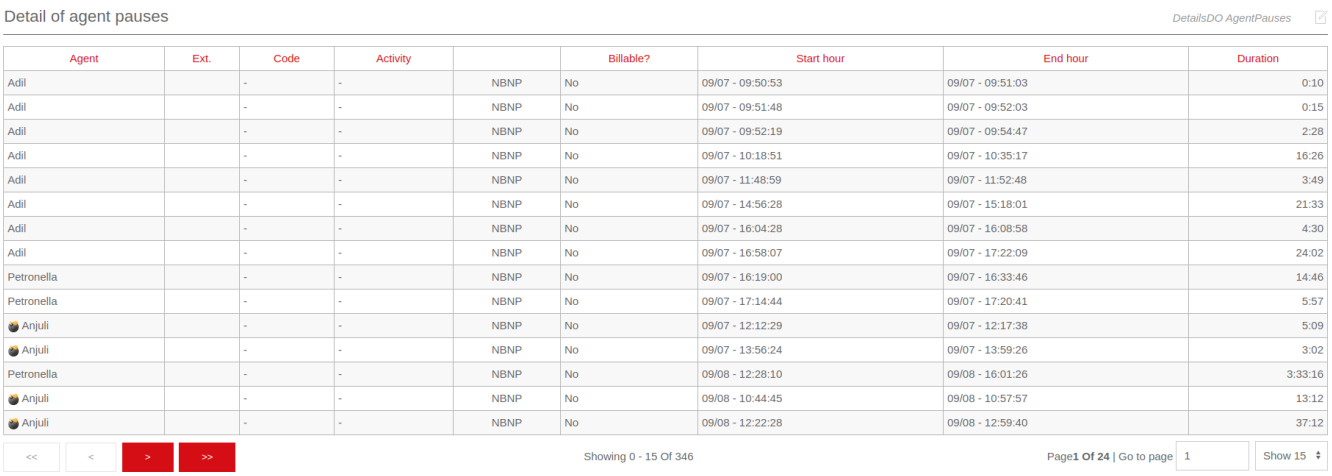
The number of pauses and the total pause time in seconds is shown.

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 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.



Detail of agent pauses

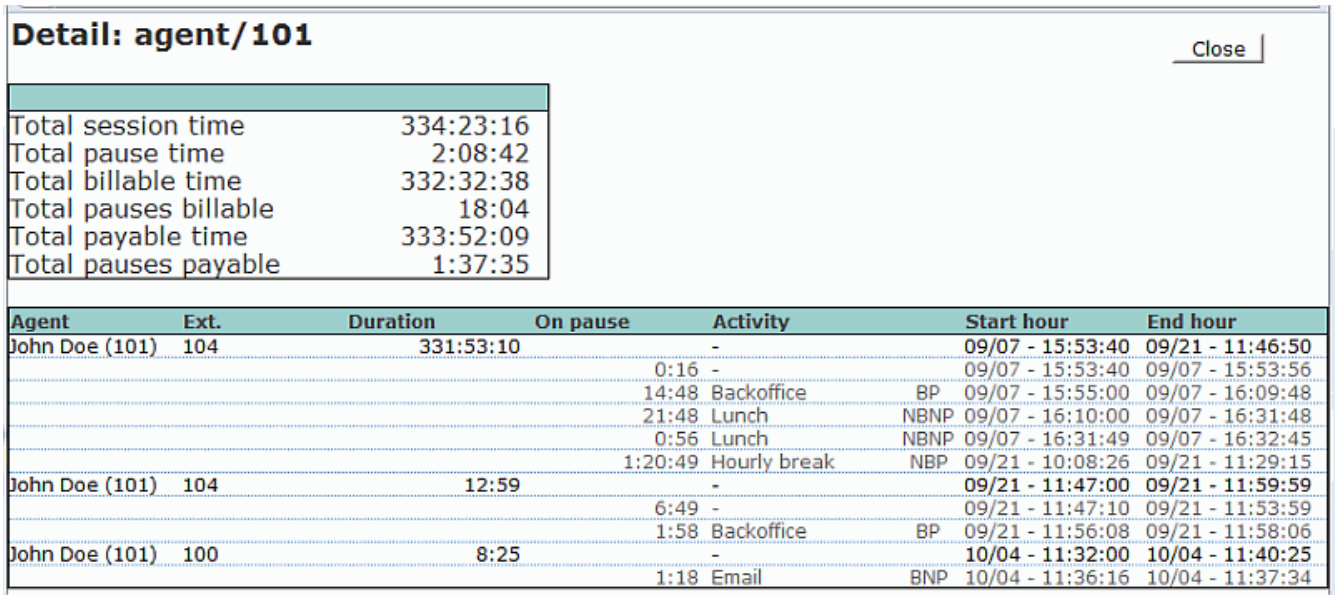
DetailsDO AgentPauses

Agent	Ext.	Code	Activity		Billable?	Start hour	End hour	Duration
Adil		-	-	NBNP	No	09/07 - 09:50:53	09/07 - 09:51:03	0:10
Adil		-	-	NBNP	No	09/07 - 09:51:48	09/07 - 09:52:03	0:15
Adil		-	-	NBNP	No	09/07 - 09:52:19	09/07 - 09:54:47	2:28
Adil		-	-	NBNP	No	09/07 - 10:18:51	09/07 - 10:35:17	16:26
Adil		-	-	NBNP	No	09/07 - 11:48:59	09/07 - 11:52:48	3:49
Adil		-	-	NBNP	No	09/07 - 14:56:28	09/07 - 15:18:01	21:33
Adil		-	-	NBNP	No	09/07 - 16:04:28	09/07 - 16:08:58	4:30
Adil		-	-	NBNP	No	09/07 - 16:58:07	09/07 - 17:22:09	24:02
Petronella		-	-	NBNP	No	09/07 - 16:19:00	09/07 - 16:33:46	14:46
Petronella		-	-	NBNP	No	09/07 - 17:14:44	09/07 - 17:20:41	5:57
Anjuli		-	-	NBNP	No	09/07 - 12:12:29	09/07 - 12:17:38	5:09
Anjuli		-	-	NBNP	No	09/07 - 13:56:24	09/07 - 13:59:26	3:02
Petronella		-	-	NBNP	No	09/08 - 12:28:10	09/08 - 16:01:26	3:33:16
Anjuli		-	-	NBNP	No	09/08 - 10:44:45	09/08 - 10:57:57	13:12
Anjuli		-	-	NBNP	No	09/08 - 12:22:28	09/08 - 12:59:40	37:12

<< < > >> Showing 0 - 15 Of 346 Page1 Of 24 | Go to page 1 Show 15

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/101

Close

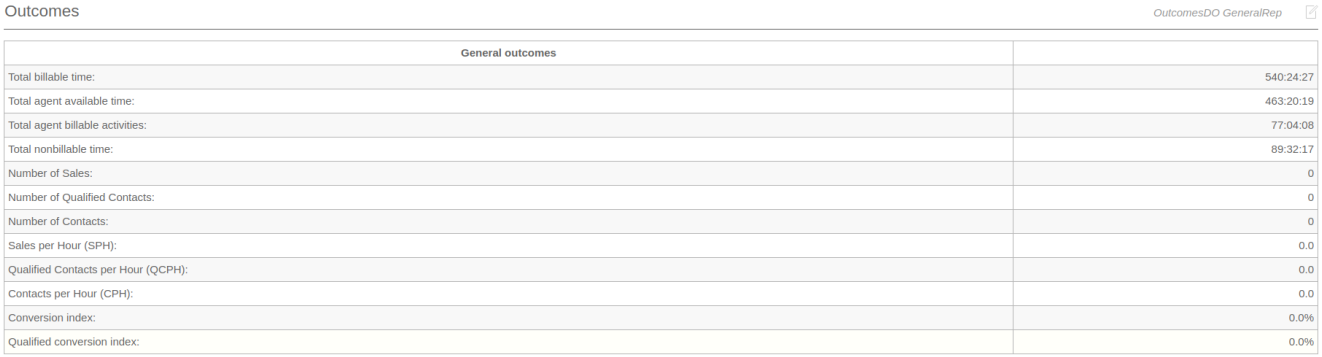
Total session time	334:23:16
Total pause time	2:08:42
Total billable time	332:32:38
Total pauses billable	18:04
Total payable time	333:52:09
Total pauses payable	1:37:35

Agent	Ext.	Duration	On pause	Activity		Start hour	End hour
John Doe (101)	104	331:53:10		-		09/07 - 15:53:40	09/21 - 11:46:50
			0:16	-		09/07 - 15:53:40	09/07 - 15:53:56
			14:48	Backoffice	BP	09/07 - 15:55:00	09/07 - 16:09:48
			21:48	Lunch	NBNP	09/07 - 16:10:00	09/07 - 16:31:48
			0:56	Lunch	NBNP	09/07 - 16:31:49	09/07 - 16:32:45
			1:20:49	Hourly break	NBP	09/21 - 10:08:26	09/21 - 11:29:15
John Doe (101)	104	12:59		-		09/21 - 11:47:00	09/21 - 11:59:59
			6:49	-		09/21 - 11:47:10	09/21 - 11:53:59
			1:58	Backoffice	BP	09/21 - 11:56:08	09/21 - 11:58:06
John Doe (101)	100	8:25		-		10/04 - 11:32:00	10/04 - 11:40:25
			1:18	Email	BNP	10/04 - 11:36:16	10/04 - 11:37:34

A complete description of the popup is available in the Report Details: Popup of agent activity .

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

OutcomesDO GeneralRep

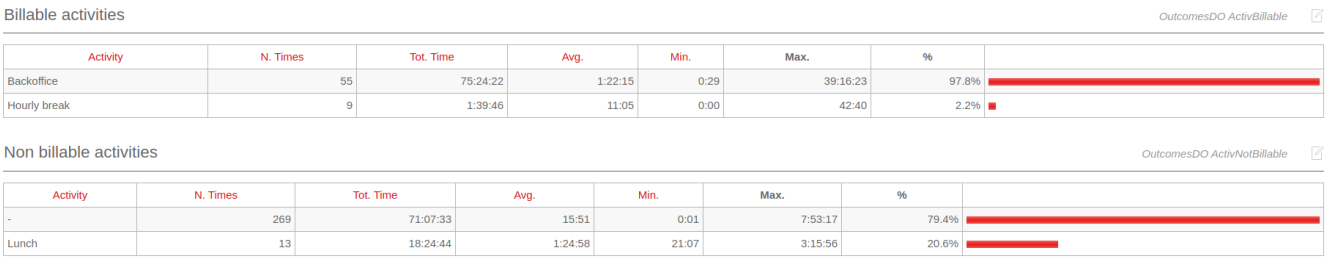
General outcomes	
Total billable time:	540:24:27
Total agent available time:	463:20:19
Total agent billable activities:	77:04:08
Total nonbillable time:	89:32:17
Number of Sales:	0
Number of Qualified Contacts:	0
Number of Contacts:	0
Sales per Hour (SPH):	0.0
Qualified Contacts per Hour (QCPH):	0.0
Contacts per Hour (CPH):	0.0
Conversion index:	0.0%
Qualified conversion index:	0.0%

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, Qualified Contacts and Sales, as defined by call outcome codes
- The *Sales per Hour* (SPH), *Qualified Contacts per hour* (QCPH) and *Contacts per Hour* (CPH) ratios
- The *Conversion index*, that is the percentage of sales over the total number of sales and contacts.

Further down the page, you can find details explaining Billable and Non-billable activities, with

average, minimum and maximum session durations, and a percentage on all activities of the same kind.



Billable activities

OutcomesDO ActivBillable

Activity	N. Times	Tot. Time	Avg.	Min.	Max.	%
Backoffice	55	75:24:22	1:22:15	0:29	39:16:23	97.8%
Hourly break	9	1:39:46	11:05	0:00	42:40	2.2%

Non billable activities

OutcomesDO ActivNotBillable

Activity	N. Times	Tot. Time	Avg.	Min.	Max.	%
-	269	71:07:33	15:51	0:01	7:53:17	79.4%
Lunch	13	18:24:44	1:24:58	21:07	3:15:56	20.6%

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 a s 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.

How are Call Outcomes calculated?

The idea is that a call can be a Contact, or a specialized contact that is a Qualified Contact, or a specialized Qualified Contact that is a Sale

This is needed because all the SPH, CPH and QCPH are calculated not on the totals of each class, but on sums of that class and generic types, like:

```
S = Number of Sales
C = Number of Contacts
Q = Number of Qualified Contacts

CPH = C / (logon time - not billable pause time)
QCPH = Q / (logon time - not billable pause time)
SPH = S / (logon time - not billable pause time)
```

Conversion indexes are calculated as:

```
CO% = S / C
QC% = S / Q
```

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.



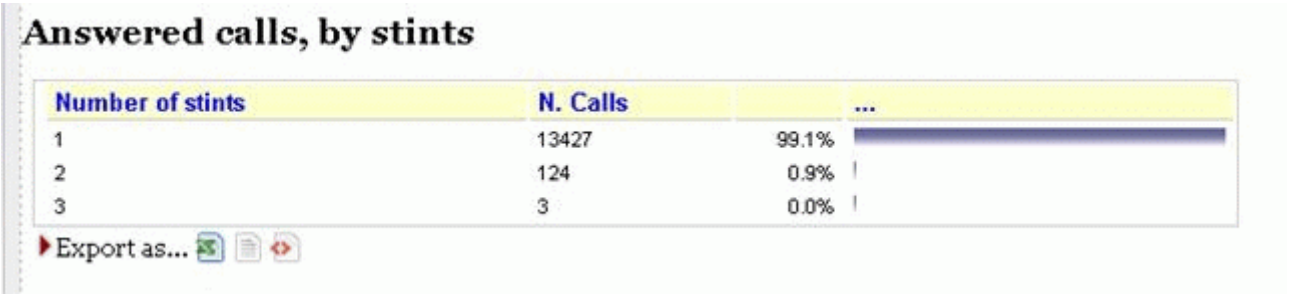
In multi-stint mode, calls are joined together before the search criteria are applied to the aggregated results. This avoids the problem of only some stints of a call matching the filtering criteria while others do not.

Multi-stint calls in QueueMetrics

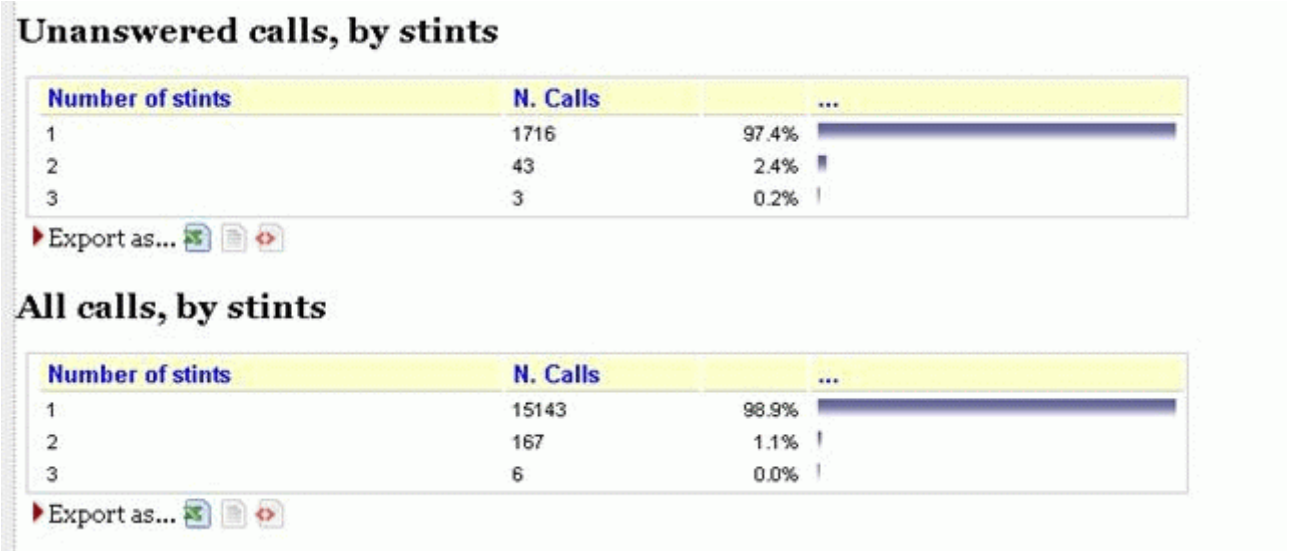
If you run calls with multi-stint mode enabled, the string "Multi-stint calls joined together" will appear on the top panel, and the number of joined together calls will be shown.



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



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:



Details of multi-stints calls are displayed in the call details pop-up, in a separate tab. See Showing call details.

Displaying calling/called numbers

By default, QueueMetrics will display numbers with the format they are in as reported by the PBX. This may or may not include a relevant name for the caller.

If the number is known to QueueMetrics, as defined in Known Numbers, then it is rewritten as to display the name that was set up in Known Numbers, and if the number is earmarked as VIP or BLACKLIST, a special icon is added by the name. If you do not want this behavior, you may switch it

off through a configuration property.

It is also possible to hide the display of numbers from some (or all) users, on an user-by-user basis. This behavior is sometimes needed in secure environments where you do not want agents to see the caller.

To turn this on for some user, you need to add the security key `HIDE_NUMBERS` to their profile. At this point, QM will hide all numbers, according to an hiding policy, as specified in the table below, that is to be set in the property `default.hiddenNumberPolicy`:

Policy	Description
`hash`	Number is replaced with a value, as to make it unreadable. Different numbers will have different values. All computations based on numbers (eg FCR) are preserved.
`stars`	Number is replaced with three stars.
`hide`	Number is replaced with as many stars as the number has characters.
`hide3`	The last three digits are replaced with stars.
`hide5`	The last five digits are replaced with stars.
`none`	Number is unaltered.

Some special numbers (e.g. blank, "none", "unknown", etc.) are not replaced.



When hiding numbers, features that depend on the hidden numbers (e.g. FCR, area codes, AMO dialing, etc.) may not work or produce partially unreliable results for users with hidden numbers enabled.

Exporting calls - an overview (Deprecated function)



This feature is disabled. The suggested way to export audio calls in batches is through the APIs, using webservice `QmFindAudio`, as documented on the JSON API Manual.

The main problems that arise from exporting calls are:

- Selecting the set of calls that have to be exported, and
- Retrieving all audio calls for export

We need to be able to select freely a number of calls for export, given one or more export criteria; we want to be able to review the results before they are final, and we want to exclude specific calls when reviewing.

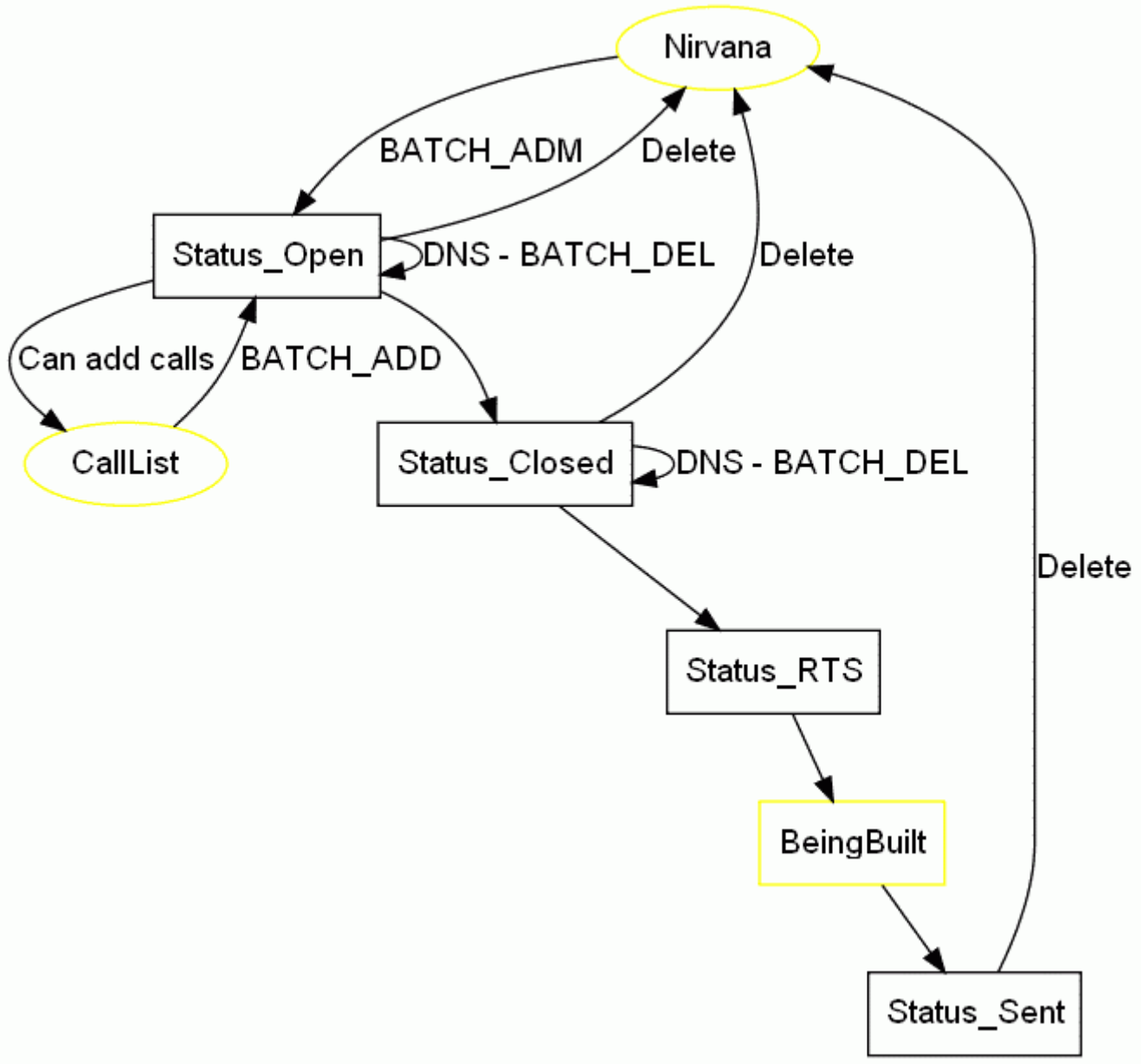
The retrieval of audio causes a similar problem; first, audio retrieval was not really made for batch access, so seek times for individual files may be in the order of one per seconds; secondarily, those files may well require a large disk space when preparing the batch.

The call export feature works in batches, that is, at any given time there are a set of batches that an administrator creates and that are to be exported. Each of them has a name and a status.

This is the life-cycle of a batch:

- An administrator creates a batch for a given data export needed and gives it a meaningful name, e.g. "Client X week 02/10". The batch is now in state Open.
- Users holding the correct key will be able to add calls to the batch, that is, when they run a report in QM, on the "Taken calls" page they see a button that invites them to add the selected set of calls to the open batches. They can repeat this process as many times as it is needed. If the same call is added multiple times to the batch, it only appears once.
- When the batch is ready, the administrator closes it. When the state is Closed, it is not possible anymore to add calls to that batch.
- When the batch is Open or Closed, it is possible for the administrators to see the list of calls in the batch and to listen to their audio/video attachments. Individual calls may be flagged as "Do not send" - those calls will appear in the batch but will not be show in the outgoing records.
- When the batch is finalized, the administrator will flag it as "Ready to send". Batches that are "Ready to send" are actually being built by QM - it may take a while to create them and download the audio files required.
- When the batch is finished processing, it will appear as "Sent".
- A batch can be deleted at any time by the administrator, unless it is in status "Ready To Send".

The graph provides a visual representation of the whole process:



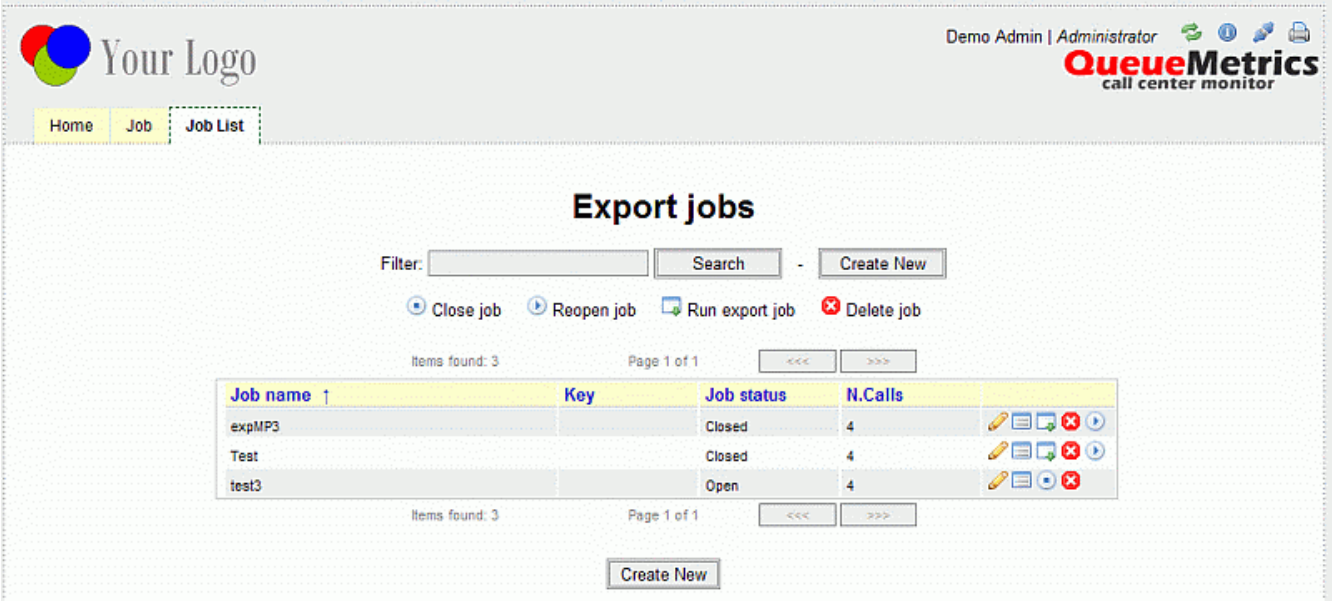
The process of building a batch may take a while - therefore there is a transaction that simply keeps displaying a page in a browser that shows a progress bar while the project is being exported.

Exporting calls in practice (Deprecated function)

In order to export calls, an administrator must first create a batch. To do this, they go to the Home Page → Import/Export calls:



From here you select List Export Jobs.



You can then create a new job.

When configuring the job, you have to specify:

- A **name** for the job. This will be the name of the folder that will be created including the call details and the audio files.
- A **security key** so that only some agents can add calls to this job.
- An **export folder** - it can be the same for all jobs, and be on the server QM is on. It should be writable by the java process that runs QM,.
- An **implementor**, that is, a data format for Manifest file of the job. See below for more details.
- If the implementor requires them, you can specify a set of **parameters**.

These properties can be changed through the lifetime of the job.

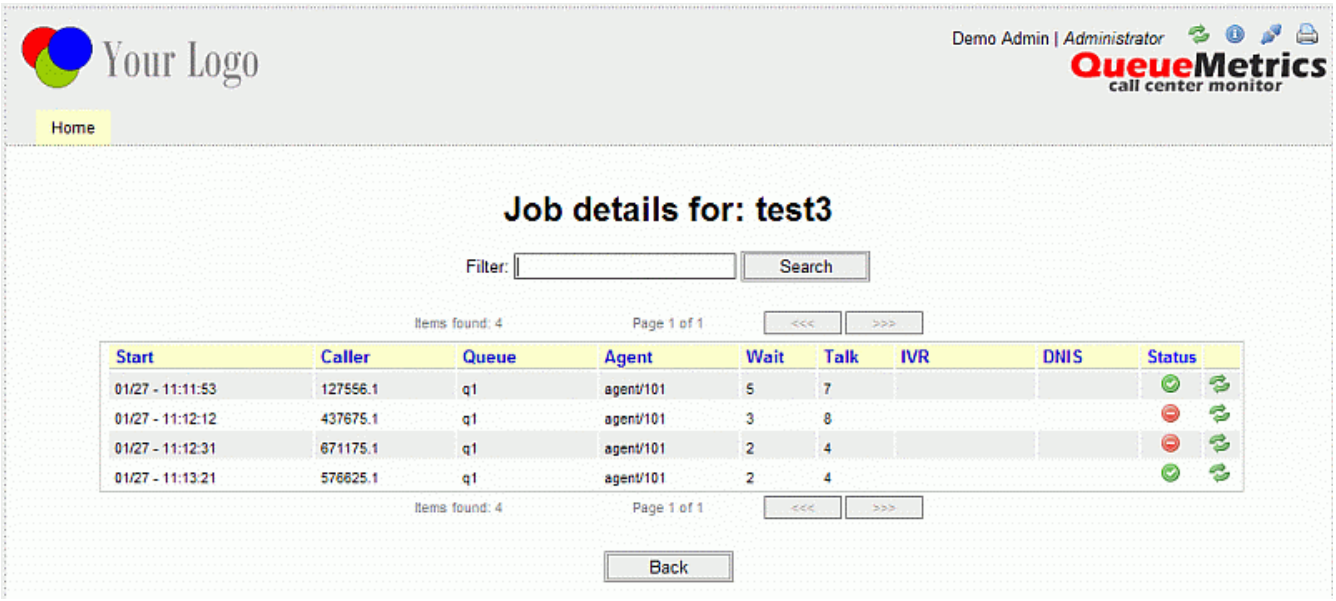
From now on, users accessing the Detail of Taken Calls will see a gadget by the end of the page like the one shown here:



This basically lets you add all the calls above to the export job you select. If the export gadget should not be visible, you have to add it manually to the current report (its code is OD02 - see Block OD02).

After you add sone calls to a job, the import transaction will confirm the add through a popup screen that displays the number of calls imported and the number of calls rejected (because they were already a member of this job). Only jobs in state Open will be availble for adding calls.

You can now see a list of calls from the Export Jobs page:



You can search calls within the current job and toggle their inclusion in the exported list by clicking on the reload icon next to each call.

When the job is ready for shipment, the administrator first has to Close it and then to Export it. This will take a while. Audio files will be retrieved through the curremtly configured Pluggable Module for audio records and will be saved in the job folder.

Output format

Thought the actual attributes used are based on the Implementor module used, the following are common attributes.

Batch attributes

- Name
- Disk path (must be accessible to Java)
- Created by, on date
- Closed by, on date
- Sent by, on date
- Video (yes/no)
- QA (yes/no)

Each call in the batch has the following attributes:

- Type: "T" taken "L" lost (initially we will only have Taken calls)
- Cluster-ID

- Server-ID
- Entered at
- Wait time
- Talk time
- Caller
- Agent
- Queue
- Call status
- Call status type (e.g Sale)
- N. of audio pieces
- Names of the audio pieces, comma-separated (a single call may have multiple recordings)

Batch disk format

A call batch appears on disk as a folder under the system call batch folder. The folder is created if not present (it must be in a Java-writable location). When the batch is in status "Ready to send", the audio files are copied to this folder; at the end of it all, a manifest file that includes the details is written. Each downloaded audio file is renamed in order to be unique and coherent, typical file names may be:

```
0000123-1.wav
0000123-2.mov
```

Both files are about the 123rd call, the first one being an audio recording and the second one a video recording.

When the batch is in status "Sent" QM no longer cares about the disk representation - it can be moved, sent as FTP, compressed and encrypted, whatever.

Manifest file format

The manifest file format should be chosen by the sysadmin - it is implemented as an abstract class for ease of change.

- It will be called Manifest.xml
- It will include a set of <call> entries including all data as per the previous section "Batch attributes"

Available implementors

HTTP file transfer

This is the basic implementor and produces an XML file.

HTTP MP3 file transfer

This implementor does two things:

- produces a basic manifest
- retrieves QA valuation data if present
- if audio files are in MP3, will insert or set ID3v2 tags so that the title of the MP3 contains information about the call.

The following ID3 tags are created:

- Call details
  - caller id
  - queue
  - dnis
  - call length
  - start time
  - length
  - agent
- Call status
- Date of Grading
- Time of Grading
- Grader ID and/or Name
- Overall Grade
- Graders Comment



this implementor DOES NOT transcode files to MP3 - they must already be in MP3 format or you should provide an external batch script to do the conversion. More details are provided in the following paragraph.

MP3 conversions on the fly

QueueMetrics lets you able to perform an external batch script call in order to convert call files to different format like, for example, wav files in mp3s. This feature is applied only for export jobs implemented with the "HTTP MP3 File Transfer" object. No external calls will be made for "HTTP File Transfer" enabled export jobs.

Assuming to have this type of export job parameters: Job name: MyJob Export folder (on server): /var/spool Implementor: HTTP MP3 File Transfer

the work flow followed by QueueMetrics, for HTTP MP3 File Transfer enabled jobs, will be like depicted below.

For each call in the job and for each file associated to a specific call:

1. The file will be stored on server folder /var/spool/MyJob folder
2. A new temporary folder will be created on /var/spool/MyJob folder
3. An external bash script will be called. The script will receive, as parameters:
    - The full file name of the file to be converted
    - The full name of the temporary folder created
    - The name of the job as defined in the job definition page
    - The parameters string as defined in the job definition page
4. The script should convert the file in the preferred format and should place the result in the provided temporary folder. QueueMetrics will wait for the conversion termination
5. QueueMetrics will move (not copy) the full conversion result it will find in the temporary folder to the original destination folder (in this case /var/spool/MyJob). Please note that there should be more than one file resulting in the conversion (like, for example, a preview quality and a hi-res quality .mp3 files) and QueueMetrics will copy all of that.
6. The temporary directory will be removed by QueueMetrics
7. QueueMetrics will publish in the manifest all the files found in the temporary folder. These files will be associated to the specific call and the number of chunks published in the manifest will reflect this.

When all files in the job are properly downloaded and converted, QueueMetrics will add all relevant QA information only to files with name ending with .mp3

Please note that:

1. The external script to be called by QueueMetrics should be specified in the configuration properties through the key "export.conversionCommand". It should be executable by Tomcat. If no key was defined, the HTTP MP3 File Transfer will skip all actions specified in the 2, 3, 4, 5, 6 steps above described (i.e. it will simply download the files and apply QA informations to eventually present .mp3 files)
2. If the external script is not present in the server and/or QueueMetrics is not able to run it, this will be signaled in the <errors> field present in the manifest (one for each call). In this case QueueMetrics will publish in the manifest the original file name.
3. QueueMetrics will not delete the original file from the /var/spool/MyJob folder. If you need to have it deleted, your conversion script should do it.
4. QueueMetrics will not publish the original file name in the manifest, unless if exceptions were raised in the conversion/move process. If you need to have the original file published in the manifest, your script should move (not copy) it to the temporary folder.
5. If some exception is raised when moving files or calling the external bash script, QueueMetrics will publish the original file name in the manifest.
6. If the conversion script generates a (set) of filename with name(s) already present in the /var/spool/MyJob folder, QueueMetrics will rename it (them) prepending the name(s) with a random 5 digit number followed by an underscore sign. The new name(s) will be published in the manifest

To summarize the overall process, a simple example of working environment is provided:

In the configuration properties is the key:

```
export.conversionCommand=/usr/local/apache-tomcat-5.5.25/webapps/ROOT/testbatch
```

and in the /usr/local/apache-tomcat-5.5.25 is an executable testbatch script:

```
[root@qmmachine ROOT]# ls -la testbatch
-rwxr-xr-x 1 root root 87 Mar 10 05:17 testbatch
```

In this simple example, the batch script copies the original file in the temporary folder (with a not unique name in this example)

```
#!/bin/bash
# $1 is the source file
# $2 is the output directory
# $3 is the job name
# $4 is the parameter field defined in the job definition page

cp $1 $2/testresult
```

The resulting working folder will contain something like:

```
....
-rw-r--r--  1 root root  408516 Mar 10 05:18 17619_testresult
-rw-r--r--  1 root root   50110 Mar 10 05:18 18542_testresult
-rw-r--r--  1 root root  884372 Mar 10 05:18 18795_testresult
-rw-r--r--  1 root root    4740 Mar 10 05:19 20110304_084640-99.wav
-rw-r--r--  1 root root  419784 Mar 10 05:19 20110304_084640.wav-99-1.avi
....
```

And the manifest something like:

```
....
<call>
<files>testresult,2071_testresult</files>
<errors></errors>
<chunks>2</chunks>
<uniqueid>3033212900899824</uniqueid>
<videocall>true</videocall>
<callid>4006</callid>
....
```



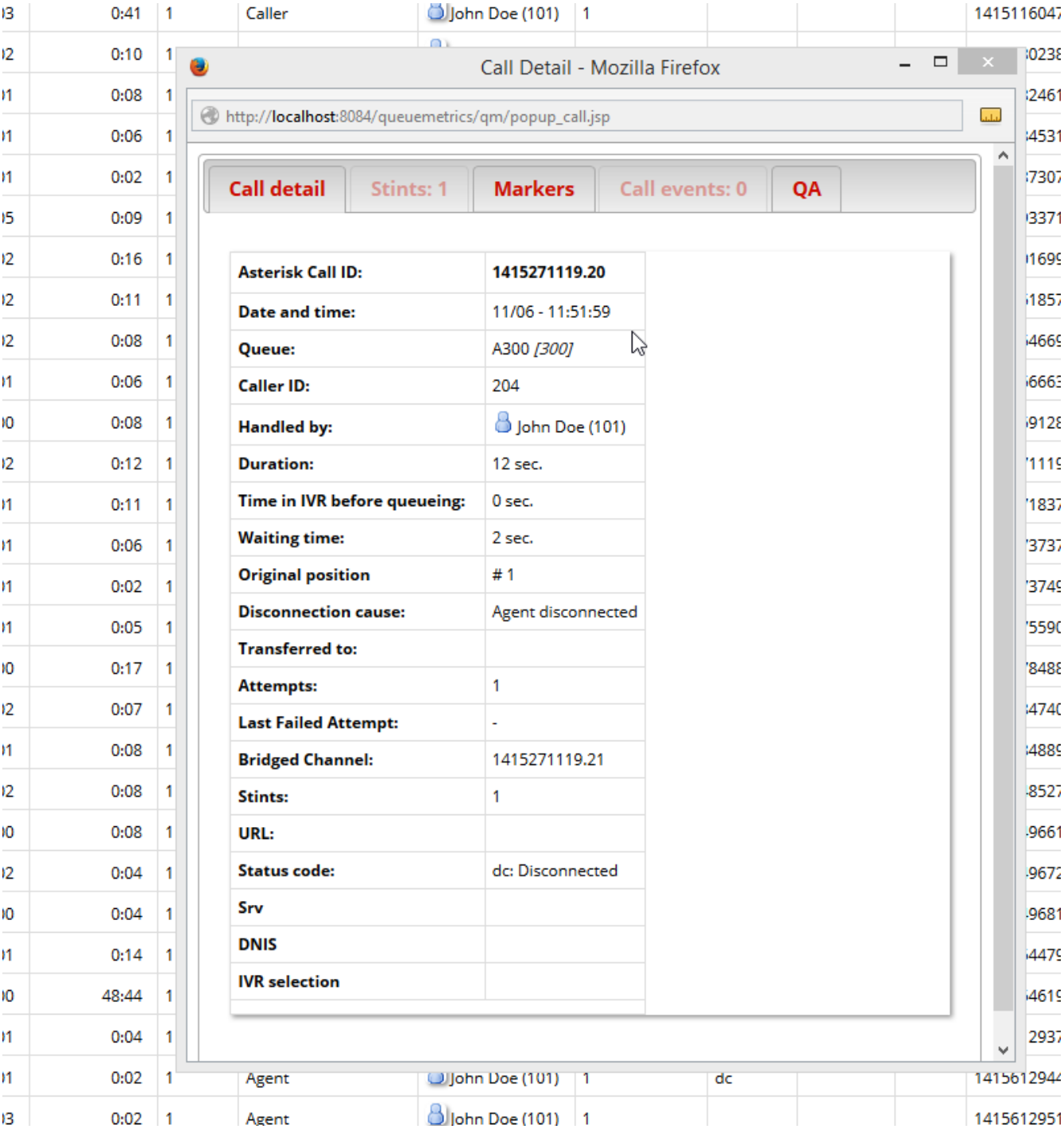
For security reasons, QueueMetrics can only call the external conversion script if

> the JVM property `-Dqm.execCmds=1` is set. Without it, the script specified in `export.conversionCommand` will not be executed. The property can be added to the JVM options (`QM_JAVA_OPTS`) - see Further configuration.

Showing call details

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

Detail of answered calls



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 others 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 initial position of the call
- 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)

Optionally other information could be shown:

- The asterisk unique ID associated to each call
- An icon that opens a new web page with an URL user customizable (useful for proprietary CRM integrations). To enable these two columns, the keys *default.crmapp* and *default.showAstClid* have to be correctly set. Please read Configuring system preferences and Appendix C: System preferences for further details.

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 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).

If the call is ongoing and you have the special grants to do so, a red scissor icon might appear next to the call status to allow for brute-force call closure. See the section Closing ongoing calls for

further details.

The call may also display:

- If the call has more than one stint - that is was rebuilt through multi-stint mode - then a list of stints is visible. If a call has only one stint, then the Stints tab is not needed and appears disabled. See Multi-stint calls
- If a call has events on it - that is specific items happening at points in time that go beyond what is displayed for the main call - they are shown in the Events tab. If not, they are disabled. The ones currently tracked are IVR traversal events, failed call connection attempts and music-on-hold events.
- A set of markers (Note: QueueMetrics till release 14.10.4 shows "tags" instead of markers).
- A link to open the QA form(s) available for that call

Listening to answered calls

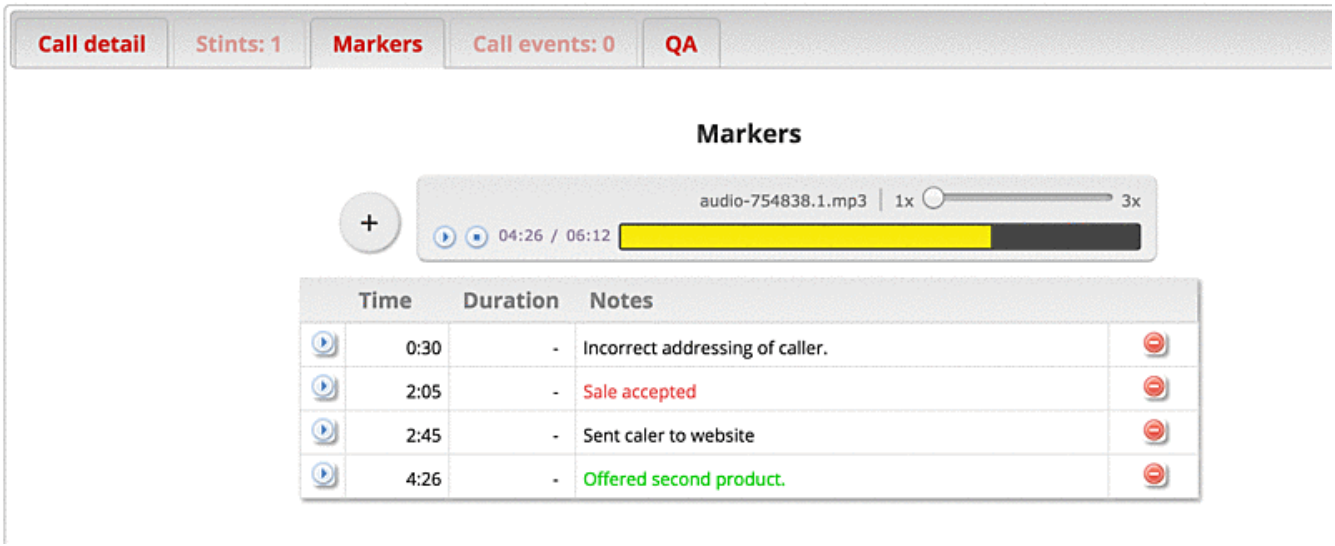
If there are audio recordings, they will be shown by the bottom of the call details page. If the call was monitored, i.e. recorded to disk, a number of sound files may be shown. By clicking on a sound file you can listen to it straight from your browser.

You can optionally have other types of media linked to a specific call, for example chat transcripts, free text or documents. They will be opened in the browser.

Please note that:

- The recorded file name must contain the Asterisk Call ID for QM to relate it to the call - see Listening to recorded calls using QM 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. The HTML5 player requires WAV or MP3 files.
- 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.
- It is possible to use different PMs to handle different audio needs - see Listening to calls using Pluggable Modules (PM).
- If generated by Asterisk, QueueMetrics can display a variety of other file type call attachments, ie. calls that are shown with a file extension that is not necessarily a sound file (image, video, audio, text or application files).

It is also possible to add markers in order to keep notes about "points of interest" for the call at hand, as shown in the example below:



By clicking on the Play icon next to a marker, the correct audio recording is loaded in the embedded player and it is started at the point in time specified in the marker.

The HTML5 Audio Player and Markers

It is possible to play audio straight in the browser using an HTML5-enabled audio player.

This has many advantages:

- You can play the audio straight in the browser, jumping back and forth as needed
- You can speed-up the audio playing, in order to take less time to listen to calls
- It integrates with the Markers editor, so you can add a comment at a specific point in time and see it later. When you click on a comment, the audio editor starts playing at that very point in time.

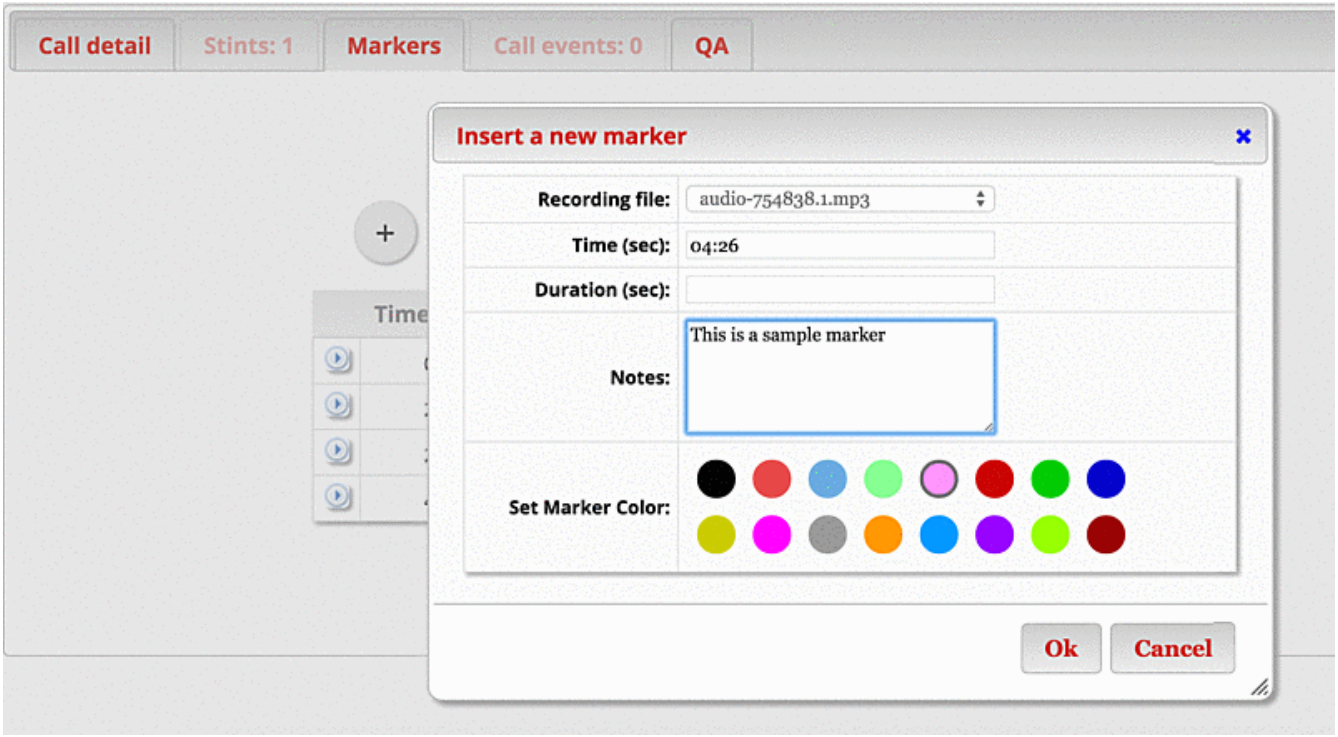
This is extremely powerful, and can be done either on the Call Details panel (under the Markers tab) or on the QA form.

In order for this to work, you need:

- A modern browser supporting HTML5 audio playback
- The system property '*audio.html5player*' must be set to true.
- If audio files are hosted outside of QueueMetrics, your HTTP server must support Http-Range requests. QueueMetrics supports Http-Range file streaming natively.
- Files must be stored in a format that is compatible with the browser. Presenty this means that your audio files should either be saved as WAV or MP3. GSM and WAV49 files will likely not work.
- Streaming is not supported for encoded audio recordings.

Users having the capacity of adding and removing markers should hold the keys

CALLMONITOR_ADDTAGS and CALLMONITOR_DELTAGS respectively.



The following pieces of information are tracked through the Markers page:

- System ID and Asterisk unique-id; marker author and time
- Time offset and duration (in seconds). You can leave the duration field blank. Values can be input as an integer number of seconds or as MM:SS
- A title (max 200 characters). Newline characters are allowed.
- A color for display (this may be used to set, say, all sales in red, all issues in blue, etc).

Auditing access to recordings: the Media Proxy

QueueMetrics can be set up so that each time a user listens to a call, or visits an attached media file, this activity is logged. In order to do this, when the user wants to download a file, QueueMetrics stays "in the middle" - it acts as a proxy and downloads the file on your behalf, forwarding it to to you as it comes.

This way QueueMetrics can track (on the system's Audit log):

- Who listened to which media, when this happened, and how much of the media was downloaded.
- Who tried listening to media, and was rejected for security reasons
- Whether a file was not downloadable or the upstream service failed

Another important advantage, especially if you use a third-party recorder, is that the QM proxy will replace audio URLs with opaque links which IDs change every time a file is accessed. Therefore users have no visibility on what happens within the storage layer. Links remain valid only for a given period of time (usually 1000 seconds, but this can be changed) and will be rejected after this period.

One last advantage is that all accesses to your media storage layer will only come from your QueueMetrics system, and that can be further authenticated by adding a custom security HTTP header. So it is easy to allow storage access only from a small set of "good" IP addresses.

The main disadvantage, of course, is that using the proxy uses up bandwidth, threads and in general resources on your QM server.

In order to enable this feature, you have to set:

```
mediaproxy.enabled=true
mediaproxy.linkvalidity=1000
```

This replaces all links from all audio PMs with custom links that do trigger the proxy.

From the point of view of your users, nothing much will change; but access to media will appear on the Audit log.

Securing the media storage layer

It is possible to have QueueMetrics add some security headers to the HTTP requests that MediaProxy forwards. If you want to, you can have a custom header added that can in turn be checked by your media storage to identify the QM server used.

```
mediaproxy.header=X-Application/CustomToken
mediaproxy.token=VERYSECURE
```

In this case, the following headers will be sent along with the media requests:

```
X-Application/CustomToken: VERYSECURE
X-Application/QM-remoteIp: 1.2.3.4
X-Application/QM-remoteUser: joe_l
```

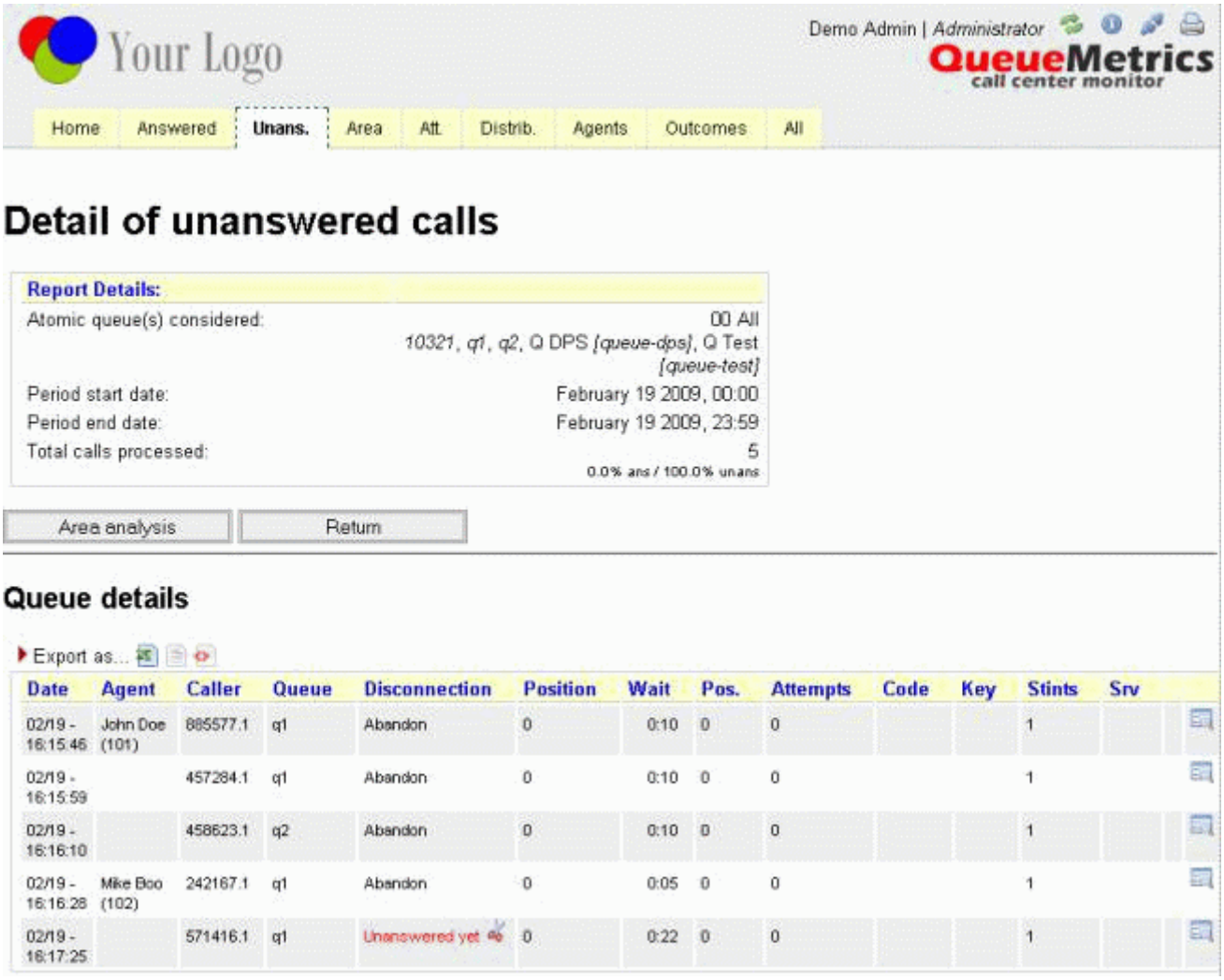
The remote IP of the client and the current user name are always forwarded automatically by the Media Proxy.



Some web servers will return error 400 if they receive unknown headers. In this case, you can set `mediaproxy.header=NONE` to avoid adding any extra `X-Application` headers.

Detail of unanswered calls

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



The following data are shown:

- Date and time of the lost call;
- The Agent that placed the call (if it's outbound) or blank if inbound;
- Caller-ID;
- Queue that handled the call;
- Disconnection cause;
- Position at disconnection, if available;
- Waiting time before disconnection, if available;
- The initial position of the call when it joined the queue, if available;
- The number of Agent attempts made before disconnection;
- The call code, if entered (this might be added automatically, e.g. by outbound diallers marking unsuccessful attempts as "unanswered" versus "fax" or "voicemail")
- The key pressed on disconnection (if any)
- The number of stints this call has
- The server that handled the call

Optionally other information could be shown:

- The asterisk unique ID associated to each call
- An icon that opens a new web page with an URL user customizable (useful for proprietary CRM integrations). To enable these two columns, the keys *default.crmapp* and *default.showAstClid* have to be correctly set. Please read Configuring system preferences and Appendix C: System preferences for further details.

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 column, 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 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 the call is ongoing and you have the special grants to do so, a red scissor icon might appear next to the call status to allow for brute-force call closure. See the section Closing ongoing calls for further details.

Detail of IVR calls

IVR calls, that is calls that were handled without being queued, can be displayed through the pop-up linked to the OD04 - IVR details (paged) data block. The contents are very similar to the information displayed for taken or lost calls, though the only information currently tracked are call IVR events, DID and caller-id.

Report Details

Reports can be fully configured by deciding which of the following blocks shall be included in each.

QueueMetrics ships with some default reports - feel free to copy or change them to create your own.

- **All reports** is a collection of (most) reports that QM can generate. It is meant as a showcase that you will customize.
- **Overview** provides a 360-degree view of all call activities on the call center queues. It covers metrics on all calls, answered and unanswered calls by queue, service level agreements, disconnection causes, and detailed session insights for agents. With comprehensive data on call distribution by day and hour, this report serves as an essential tool for evaluating overall performance and identifying areas for improvement in customer service and operational efficiency.
- **Quick Agent Report** is a simplified view of agent activity during the previous month
- **Quick reports** is a very short daily recap meant to be sent by email by setting up a Scheduled Job for automatic export by e-mail.
- **SLA Overview** provides a comprehensive overview of service level agreement (SLA) compliance across all queues. It highlights key performance metrics such as response times and percentage of calls answered within a customizable threshold.
- **Recap** offers a comprehensive look at the call dynamics within your call center over a selected timeframe. It breaks down the overview of queue calls, provides an hourly summary of answered and lost calls, and details call attempts by each agent, whether taken or lost. This report is ideal for understanding traffic flow and agent activity, aiding in operational adjustments and resource allocation.
- **Recall details** focuses on the specifics of customer recalls, providing essential data for follow-up strategies. It includes a recall summary, lists numbers that need to be recalled, and tracks numbers that have already been recalled within the selected timeframe. Additionally, it analyses first call resolution effectiveness and identifies frequently contacted numbers, making it invaluable for improving customer satisfaction and resolution efficiency.
- **Weekly Agent Performance** assesses the effectiveness of each agent over a week, detailing call attempts, session activities, pause durations, and availability by hour. It also examines agent productivity through metrics such as queue talk time and answered call distribution across the week. Designed to support performance management, this report helps supervisors recognize achievements and target areas for agent development, aligning efforts with overall call center goals.

A set of more pre-defined reports is available at https://github.com/Loway/Templates/ - and you are welcome to contribuite your own if you create interesting reports or wallboards.



The GitHub repo also contains the "pristine" version of the reports QM ships with - so if you delete or change one of the example reports, you can restore it from there.

Historical reports - Answered calls

OK01 - All calls



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

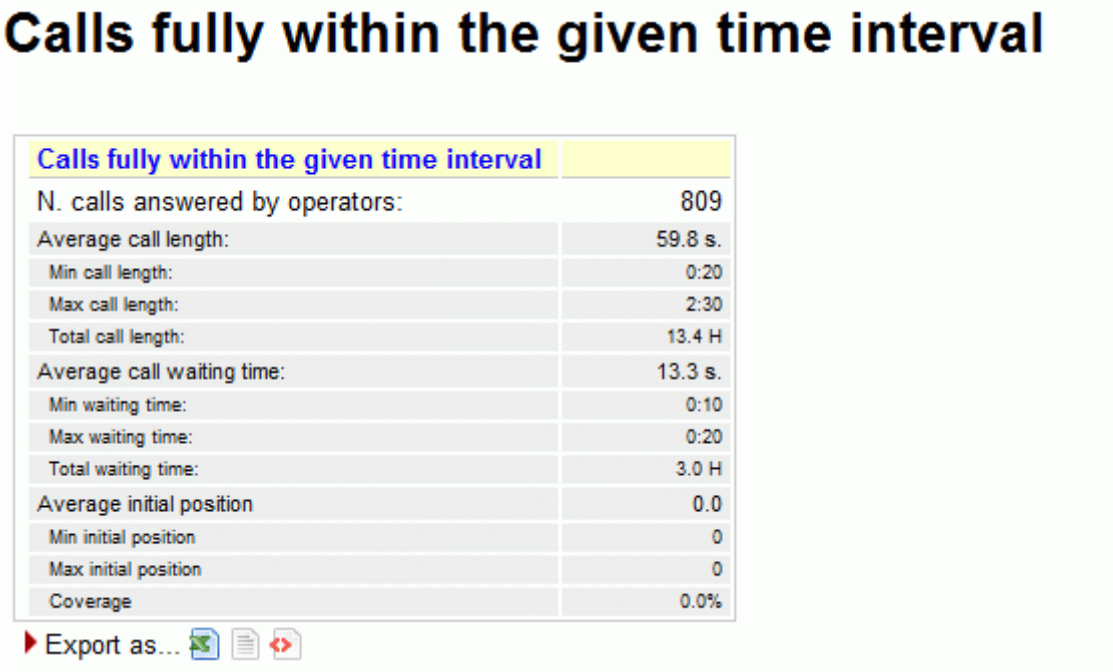
The top panel shows:

- 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.
- The average initial position of the call in the queue
- The minimum and maximum initial queue positions that have been detected
- The queue position coverage: as this information is not tracked for all calls, this ratio shows the average number of calls that had queue position record.

Available since	1.6.0
Default page	Answered calls
Shortcut code	OK01
API block name	OkDO.RiassAllCalls
Parameters	-

See also	

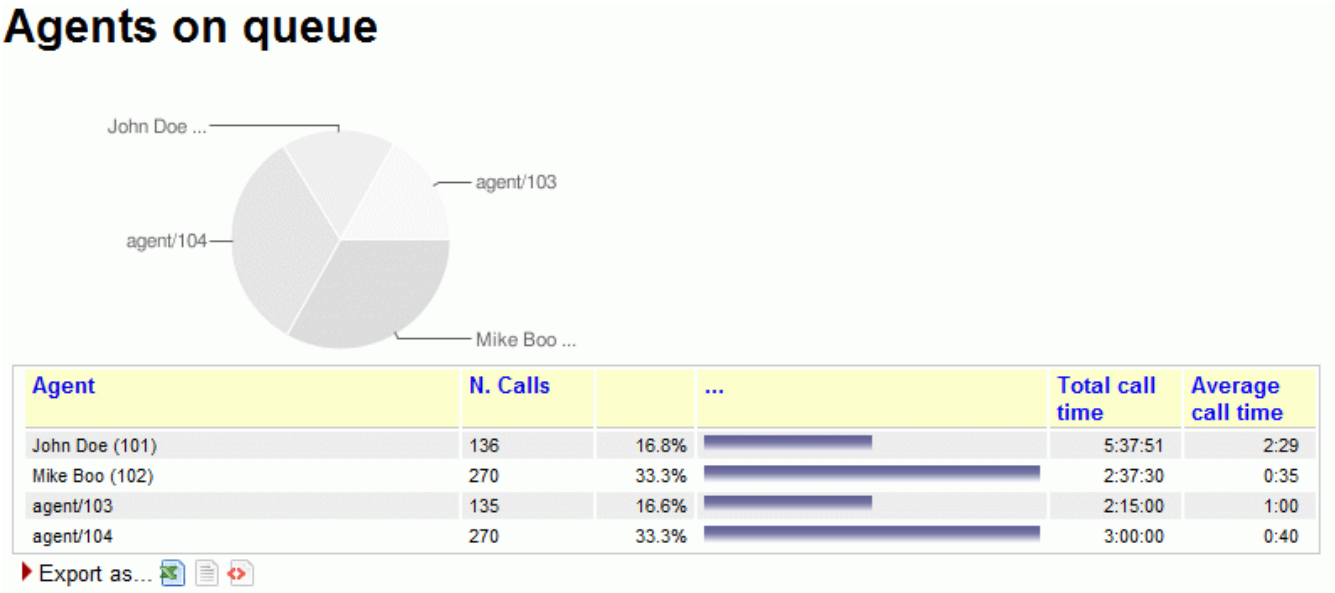
OK02 - Calls fully within the given time interval



The answered completed calls section deals with calls that were correctly handled by agents. This is similar to what's reported on previous panel but may exclude calls that were started before or terminated after the given time frame.

Available since	1.6.0
Default page	Answered calls
Shortcut code	OK02
API block name	OkDO.RiassFullyWithin
Parameters	-
See also	OK01 - All calls

OK03 - 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.

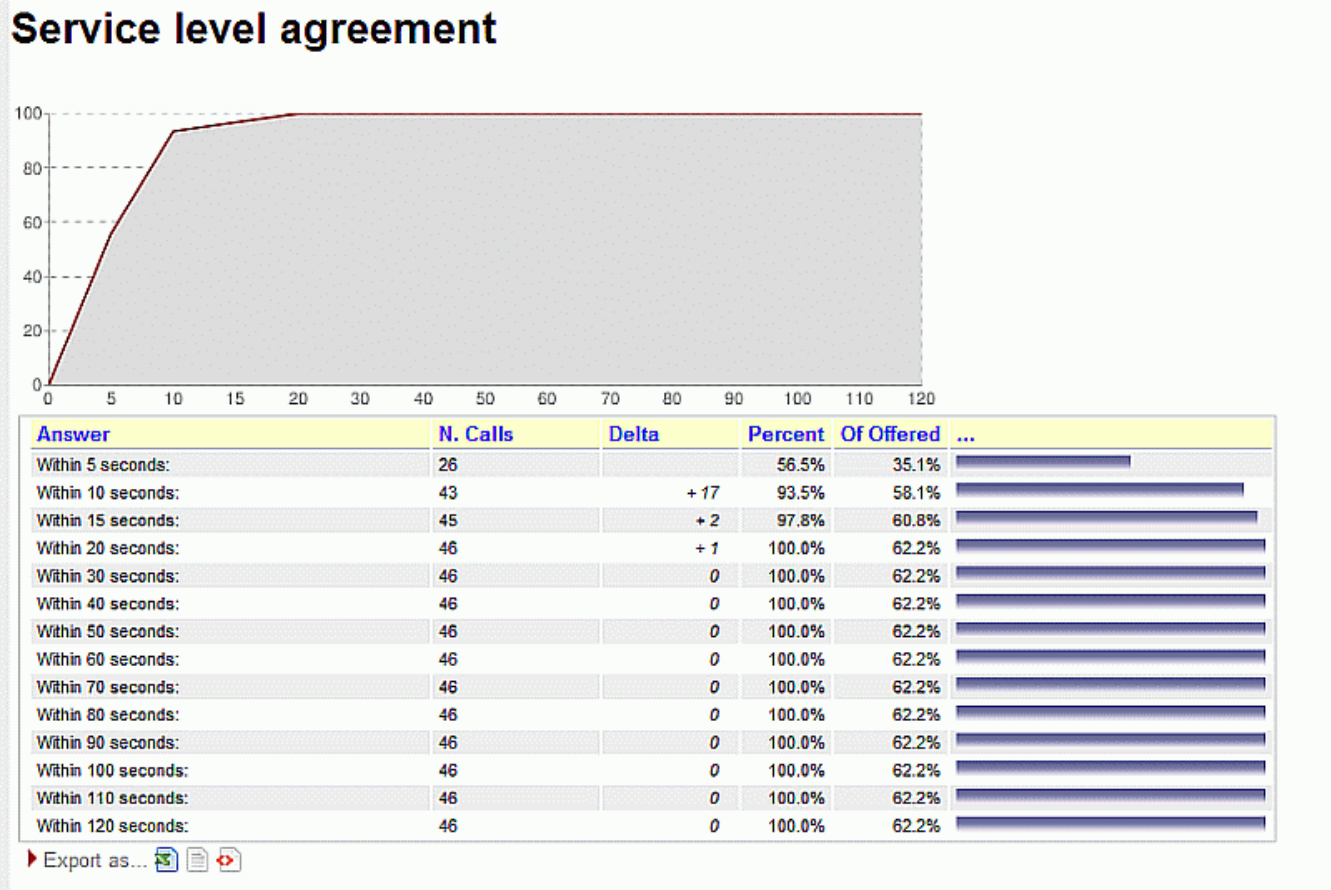
If calls are connected directly to a phone terminal, QM tries its best to show the corresponding terminal, usually in the format used by Asterisk, like "SIP/303" to signify a SIP phone whose number is 303.

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/-".

The pie graph shows which agents have been available for the given queue representing the percentage of all calls that each one handled.

Available since	1.6.0
Default page	Answered calls
Shortcut code	OK03
API block name	OkDO.AgentsOnQueue
Parameters	-
See also	

OK04 - 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 below).

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, while the "Offered" column displays the result of the taken calls divided by the total taken plus the total lost.

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 Inclusive Service Level Agreement.

The graph reports the percentage of how many calls were answered within X seconds, as reported in the table.

It is possible to configure the time frame and increment separately for an initial period and the rest of the interesting period; in this way it is possible to have different breakdowns, e.g. every 5 seconds up to 20 seconds and every 10 seconds up to 120 seconds. See Configuring system preferences.

For example, by setting:

- initial_interval=5 and max_initial_interval=20
- interval=10 and max_monitored_delay=60

You get the following cutoff points: 5, 10, 15, 20, 30, 40, 50, 60 seconds

By setting:

- initial_interval=3 and max_initial_interval=3
- interval=5 and max_monitored_delay=60

You get the following cutoff points: 3, 5, 10, 15, 20, 25,... seconds

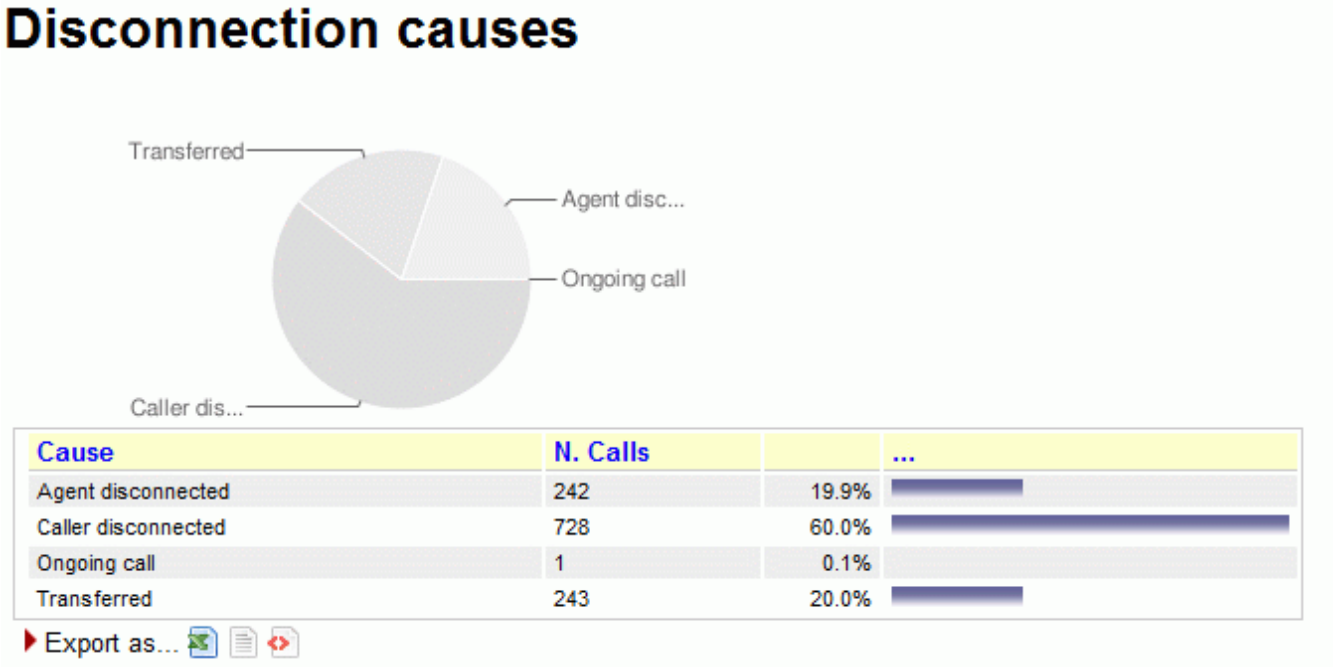
By setting

- initial_interval=0 and max_initial_interval=0
- interval=10 and max_monitored_delay=120

You get the default cutoff points: 10, 20, 30, 40, 50 , 60, 70, 80, 90, 100, 110 and 120 seconds

Available since	1.6.0
Default page	Answered calls
Shortcut code	OK04
API block name	OkDO.ServiceLevelAgreement
Parameters	-
See also	UN18 UN07 UN06

OK05 - Disconnection causes



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

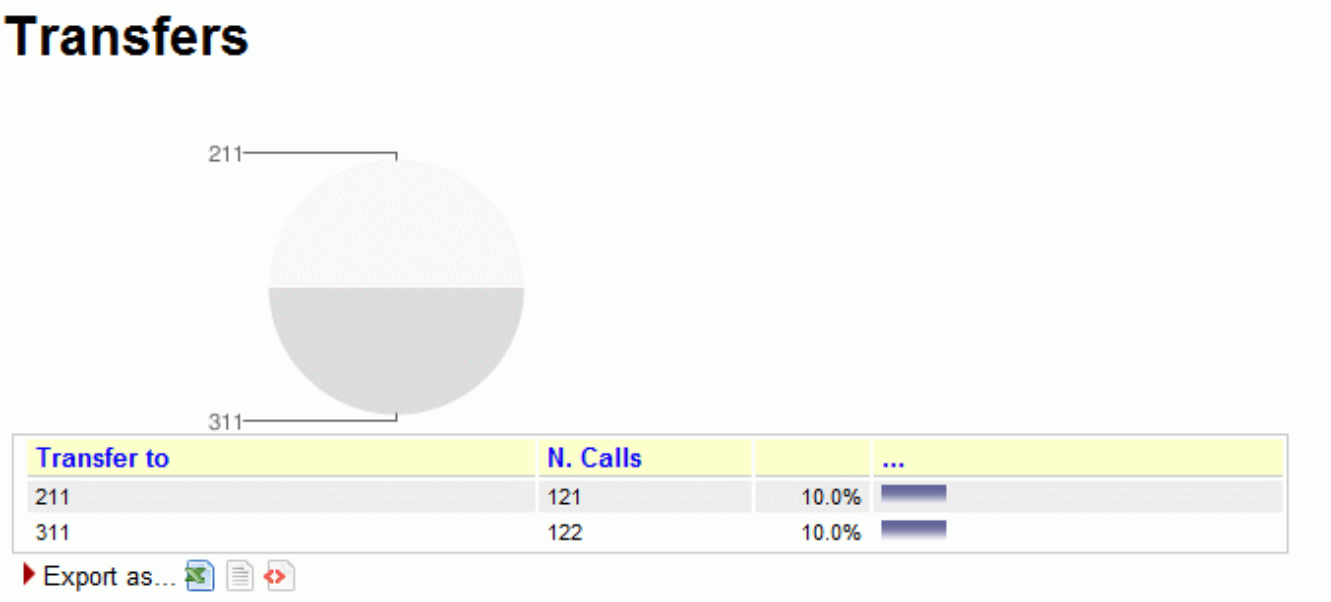
- The agent hung up, or
- The caller hung up, or

- 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.

The graph reports the percentage values associated to the reason of why calls were terminated, as calculated in the table.

Available since	1.6.0
Default page	Answered calls
Shortcut code	OK05
API block name	OkDO.DisconnectionCauses
Parameters	-
See also	

OK06 - Transfers



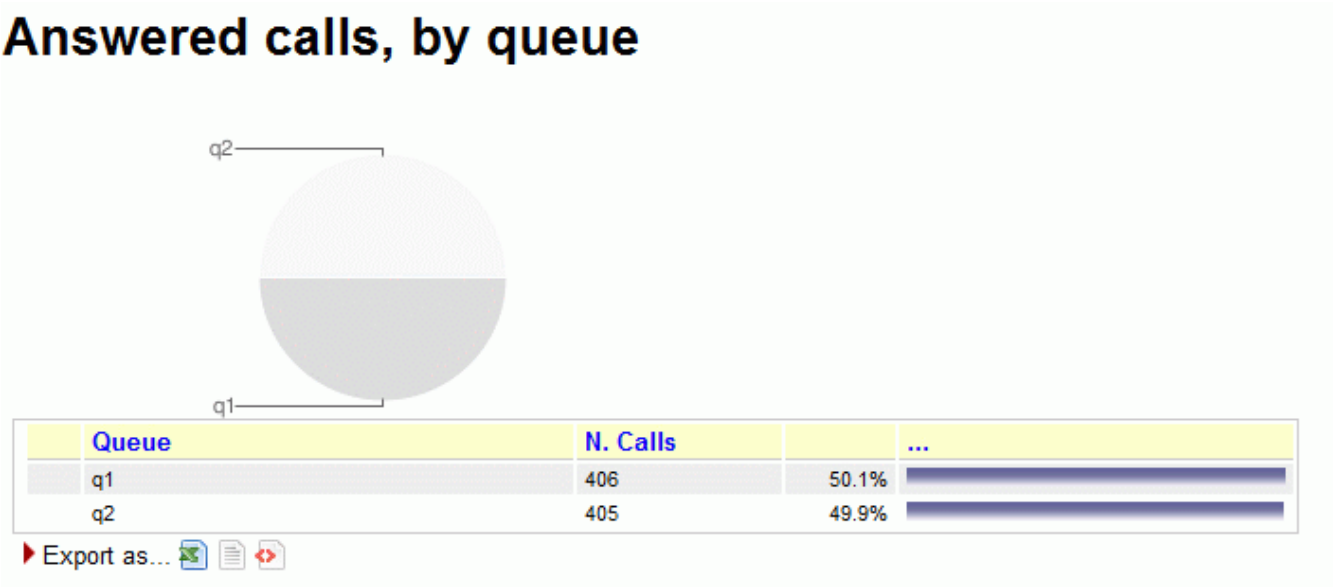
This graph shows how many calls were transferred to each extension in the given time frame. This lets you know who is handling exception calls.



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.

Available since	1.6.0
Default page	Answered calls
Shortcut code	OK06
API block name	OkDO.Transfers
Parameters	-
See also	

OK07 - Answered calls, by queue

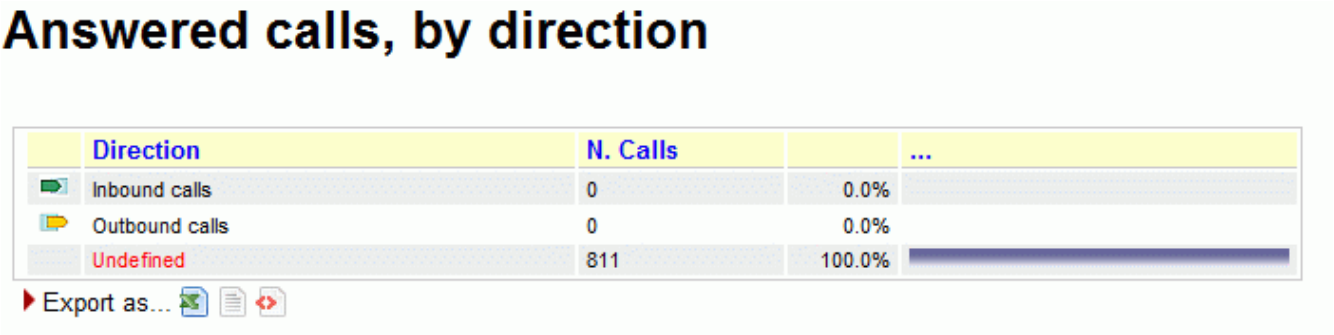


If more than one queue is in use for the report, this table shows the relative magnitude of each queue.

The graph reports the percentage associated to each queue in the table.

Available since	1.6.0
Default page	Answered calls
Shortcut code	OK07
API block name	OkDO.AnsweredcallsByQueue
Parameters	-
See also	

OK08 - Answered calls, by direction



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

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

The graph reports the percentage associated to each queue in the table.

Available since	1.6.0
Default page	Answered calls
Shortcut code	OK08
API block name	OkDO.AnsweredcallsByDirection
Parameters	-
See also	

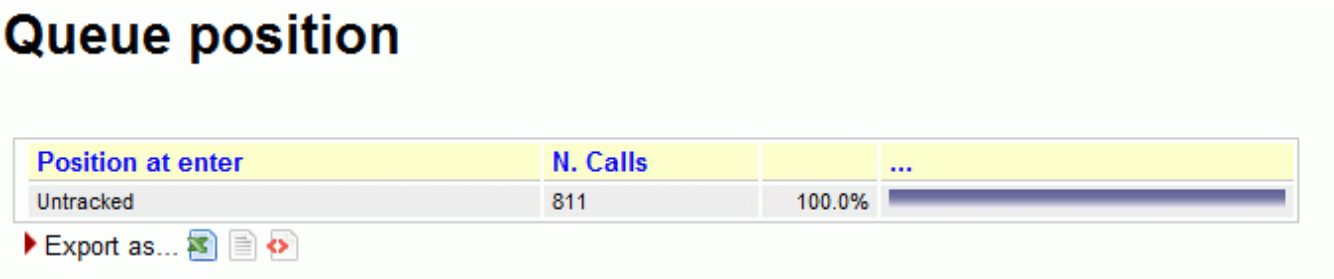
OK09 - Answered calls, by stints



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.

Available since	1.6.0
Default page	Answered calls
Shortcut code	OK09
API block name	OkDO.StintsOk
Parameters	-
See also	

OK10 - Queue position

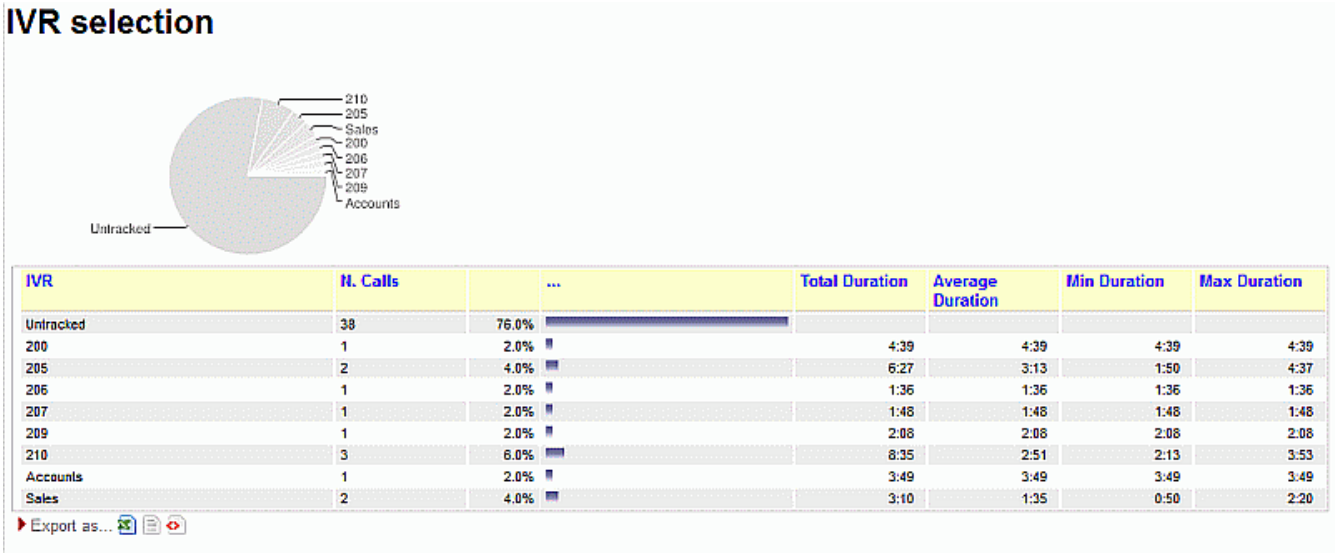


This graph shows the initial queue position that the calls had when they joined the queue. For example, a queue position of 1 means that a call was first in line, of 5 means that a call had four other calls in line before being answered. As the logging of queue positions is a bit inconsistent, some calls might be missing it ("Untracked")

Available since	1.6.0

Default page	Answered calls
Shortcut code	OK10
API block name	OkDO.QPosOk
Parameters	-
See also	

OK11 - IVR selection

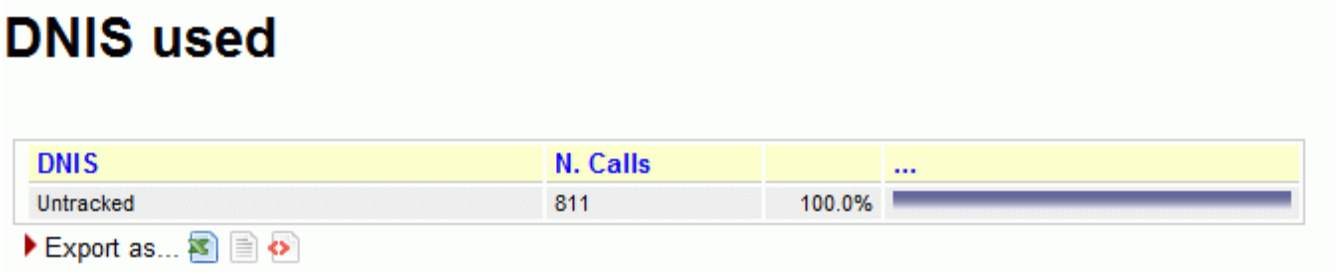


This graph shows the distribution of IVR selections available in the calls processed. We can also see the IVR duration values, which is related to the time that the call was within the IVR before entering the queue.

This must be tracked manually in Asterisk - See "Configuring Asterisk for QueueMetrics" below.

Available since	1.6.0
Default page	Answered calls
Shortcut code	OK11
API block name	OkDO.IvrOk
Parameters	-
See also	

OK12 - DNIS used

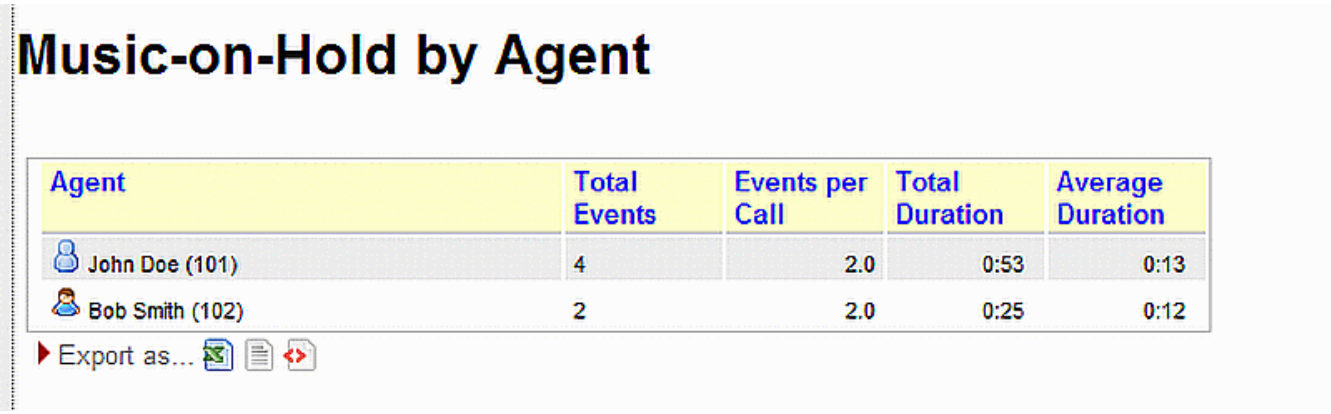


This graph shows the distribution of DNIS lines available in the calls processed.

This must be tracked manually in Asterisk - See "Configuring Asterisk for QueueMetrics" below.

Available since	1.6.0
Default page	Answered calls
Shortcut code	OK12
API block name	OkDO.DnisOk
Parameters	-
See also	

OK13 - Music On Hold by Agent

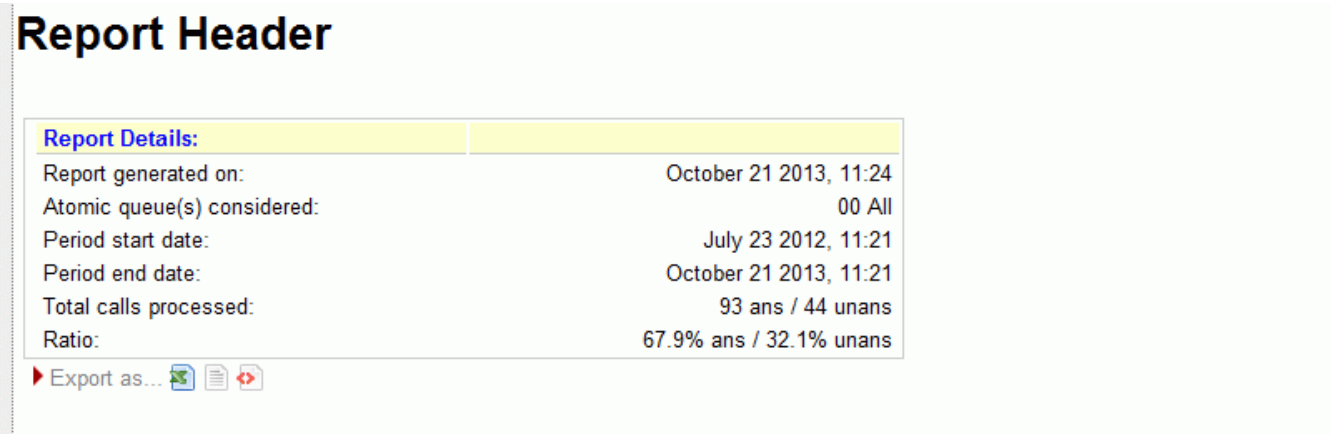


This allows to see the total number of Music on Hold (MOH) events per agent, how many MOH instances took place throughout a call, the average and total duration of MOH events.

This must be tracked manually in Asterisk - See "Configuring Asterisk for QueueMetrics" below.

Available since	12.2.0
Default page	Answered calls
Shortcut code	OK13
API block name	OkDO.MOHOk
Parameters	-
See also	

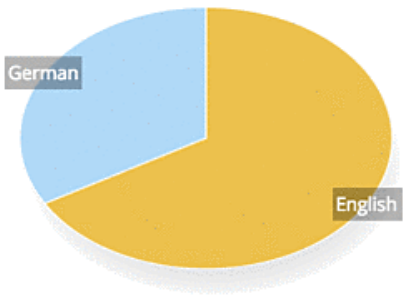
OK14 - Report Header



This is the report header and will be added automatically on each tab on the screen and at the beginning of each session in the PDF and Excel report export.

Available since	13.04.9
Default page	Beginning of each page
Shortcut code	OK14
API block name	OkDO.HDRRpt
Parameters	-
See also	HDRRpt

OK15 - Required Skills for Taken Calls



Skill Required	N. Calls	Percent	Avg Wait	Avg Talk	Min lvl	Avg lvl	Max lvl
English	2	100.0%	0:06	0:20	60	65	70
German	1	50.0%	0:02	0:10	90	90	90

This data block displays the skills that were required for taken calls in the current reports.

As multiple skills may be required for the same call, or some calls may have no required skill, the sum of all skills may be significantly different from the number of calls.

For each skill, the average wait and talk duration of all calls having that specific skill is computed. Plus, the minimum, average and maximum level at which this skill was requested for the specific set of calls is displayed.

Available since	15.10
Default page	Answered calls
Shortcut code	OK15
API block name	OkDO.SkillsOk
Parameters	-
See also	UN19, UN20

OK16 - Overview by Agent



This data block displays all activity that happened towards all agents in the current report. This report and its twin AG17 are meant for a quick but effective way to "grok" what happened in a specific report. The radar chart displays the percentage of calls answered for each agent.

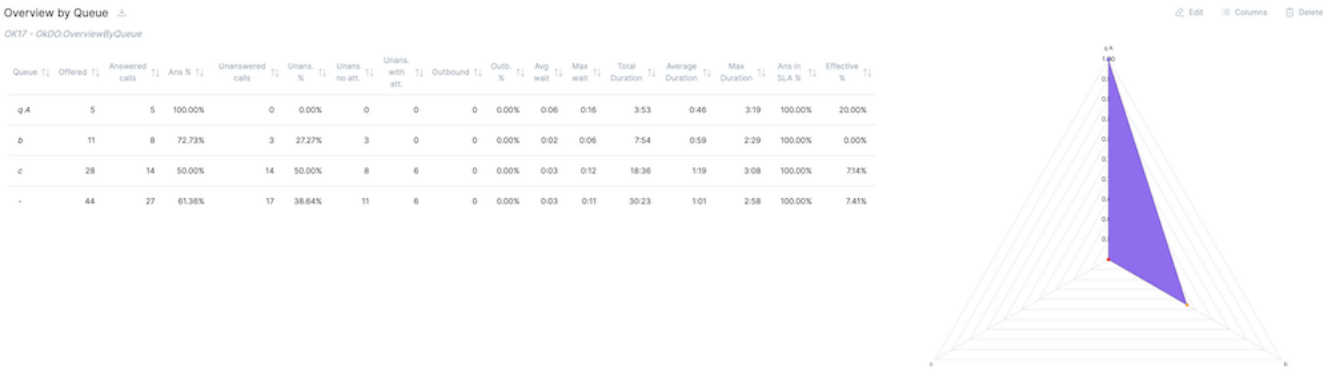
The columns are:

- Agent code/name
- Number of calls offered - in this context, how many times the agent's phone was rung. So for example if the same call tries the agent twice, it is counted as two times offered. So it does NOT match the number of taken + lost calls available elsewhere.
- Number of calls answered, and their precentage (of all offered).
- Number of lost attempts, and their percentage
- Outbound calls, and their percentage
- Total, average and maximum call duration for this agent
- ASA - Average speed of answer - how quickly the agent picked up calls when they answered. This information may not be correct for all types of PBXs.
- For taken calls, how many of them were answered within the SLA, as used in block DD08.
- For taken calls, how many of them are considered "effective", as used in block AG21.

A recap row is added to the table detailing the sums or averages of previous rows. It is always added, even if there is no data.

Available since	24.11
Default page	Answered calls
Shortcut code	OK16
API block name	OkDO.OverviewByAgent
Parameters	-
See also	OK17, AG13, AT03, AG21, DD08

OK17 - Overview by Queue



This data block is similar to AG16 but with a perspective on queues.

Columns include:

- Queue name or id
- Number of calls offered, as all calls taken plus all calls lost
- Number of percentage of answered calls
- Number of percentage of all unanswered calls
- Number of unanswered calls having no attempts. This means that no agent could possibly pick them up, because either no agents were available or they all were busy in the given period.
- Number of unanswered calls having at least one attempt. Such calls might have been answered, had the agents pick up the call when offered.
- Number and percentage of outbound calls (taken and lost)
- Average and maximum wait time for the call, computed on all offered calls
- Average, maximum and total call duration for answered calls.
- For taken calls, how many of them were answered within the SLA, as used in block DD08.
- For taken calls, how many of them are considered "effective", as used in block AG21.

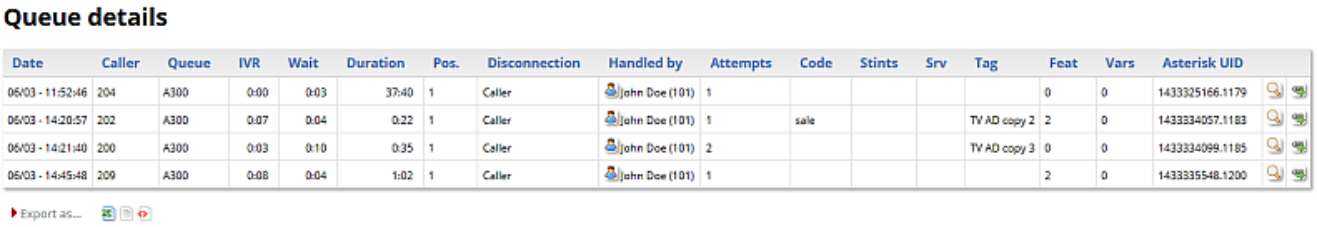
A recap row is added to the table detailing the sums or averages of previous rows. It is always added, even if there is no data.

Available since	24.11

Default page	Answered calls
Shortcut code	OK16
API block name	OkDO.OverviewByQueue
Parameters	-
See also	OK16, AG13, AT03, AG21, DD08

Historical reports - Details of answered calls

OD01 - Queue details



This page shows the detail of answered calls. See Showing call details.

Available since	1.6.0
Default page	Details of answered calls
Shortcut code	OD01
API block name	DetailsDO.CallsOK
Parameters	-
See also	OD03 - Queue details (paged)

OD02 - Add to export job

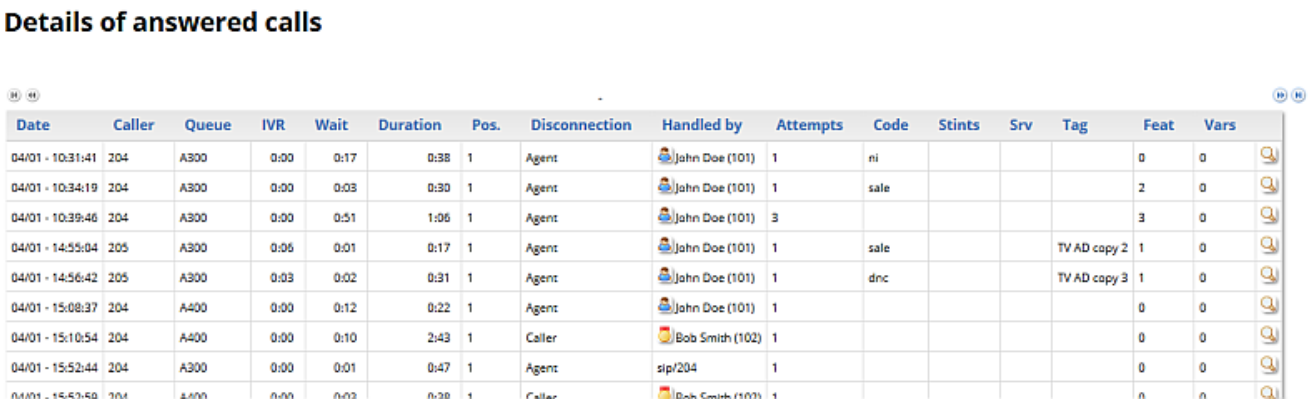


This is a pseudo-block that is used to display a button for call export. The button may not be displayed if the user does not have the correct grants. As it does not actually contain data, it cannot be queried over XML-RPC.

Available since	1.7.0 - **Removed in 20.11**
Default page	Details of answered calls
Shortcut code	OD02
API block name	-
Parameters	-

See also	

OD03 - Queue details (paged)



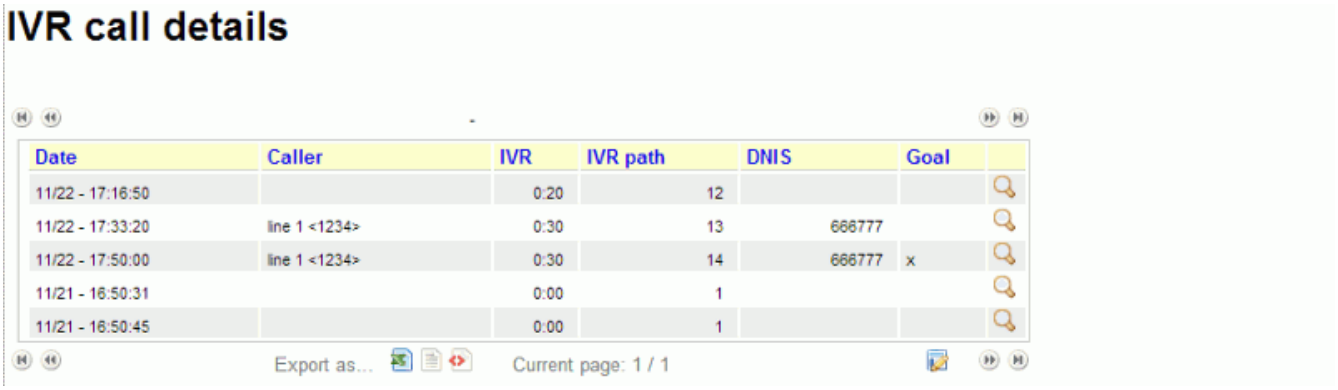
This page shows the paged detail of answered calls. See Showing call details.

The details of calls are paged - so you can move back and forth - and you can control which columns will appear on screen. The paged mode uses way less memory and is quicker to use than the full list as in OD01 - Queue details.

When accessing the data source though XML-RPC or printing a report, it is mandatory to use the OD01 block, as this block is interactive and will not render correctly.

Available since	13.03 - **Removed in 20.11**
Default page	Details of answered calls
Shortcut code	OD03
API block name	-
Parameters	-
See also	OD01 - Queue details and Configuring paged call lists

OD04 - IVR details (paged)



This block shows the paged detail of IVR calls. These are calls that did not hit a queue in the current report and thererfore are reported as neither answered nor unanswered.

The details of calls are paged - so you can move back and forth - and you can control which columns will appear on screen. The paged mode uses way less memory and is quicker to use than the full list as in OD05 - IVR details (full list).

When accessing the data source though XML-RPC or printing a report, it is mandatory to use the OD05 block, as this block is interactive and will not render correctly.

Available since	13.03 - **Removed in 20.11**
Default page	Details of answered calls
Shortcut code	OD04
API block name	-
Parameters	-
See also	OD05 - IVR details (full list)

OD05 - IVR details (full list)

This block shows the paged detail of IVR calls. These are calls that did not hit a queue in the current report and thererfore are reported as neither answered nor unanswered.

This block prints a complete list and should be used only for printing and XML-RPC querying.

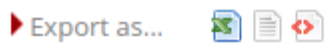
Available since	13.03
Default page	Details of answered calls
Shortcut code	OD05
API block name	DetailsDO.CallsIVR
Parameters	-
See also	OD04 - IVR details (paged)

OD06 - Executive Summary

Executive Summary

Executive Sumary

Executive Summary	
General Statistics	
Taken Calls	46
Avg. elapsed time	34
Lost Calls	4
Total Calls	50
Total elapsed time	1594
Caller Statistics	
Short Calls	7
Unique Callers	19
Unique First Taken	16
Unique First Lost	3
Unique Final Lost	2
Lost Calls Statistics	
A	4



This report provides a wealth of information regarding the situation in general. With a final focus on lost calls and disconnection reasons. The information is the following:

General Statistics

- Taken Calls
- Avg. Elapsed Time
- Lost Calls
- Total Calls
- Total Elapsed Time

Caller Statistics

- Short Calls
- Unique Callers
- Unique First Taken

- Unique First Lost
- Unique Final Lost

Lost Calls Statistics A list of disconnection reasons and the number of lost calls with each disconnection reason

Notes:

- the number of seconds needed for a call to be classified as long is dictated by the following configuration property: default.shortcall_treshold = 30 If the property is not specified, it will deafault at 30.

Available since	17.06.16
Default page	Executive Summary
Shortcut code	OD06
API block name	DetailsDO.ExecutiveSummary
Parameters	-
See also	OD06 - Executive Summary

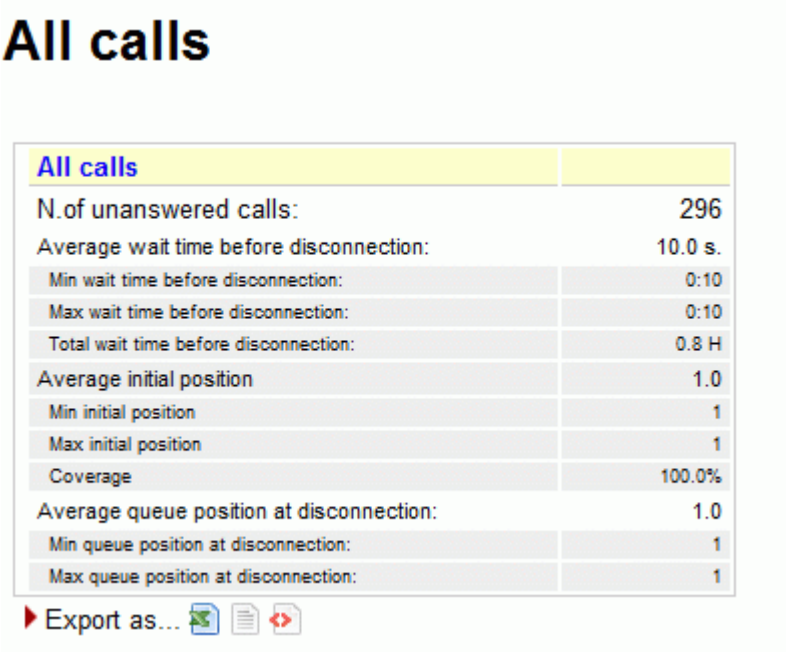
Historical reports - Unanswered calls

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.



if you run a report with an agent filter, or a supervisor filter, or a location filter, the number of lost calls in the report is usually zero, showing just outgoing calls, if any. This is because the agent is specified only for taken calls and not lost ones, so not deleting them all would show, e.g., the taken call data for one single agent and the lost calls for all of the queue. You can override this behaviour through a configuration switch if you feel this is not correct for you.

UN01 - All calls

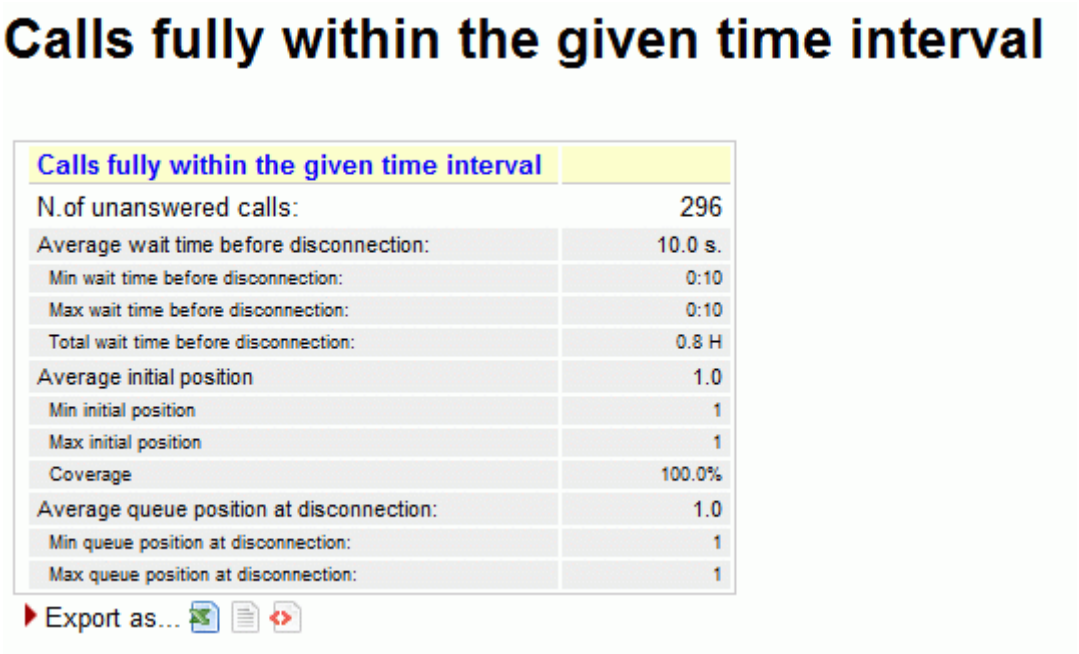


The report shows:

- 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.
- The average, minimum and maximum initial queue position, and the coverage given for this computation

Available since	1.6.0
Default page	Unans.
Shortcut code	UN01
API block name	KoDO.ReportKoAll
Parameters	-
See also	

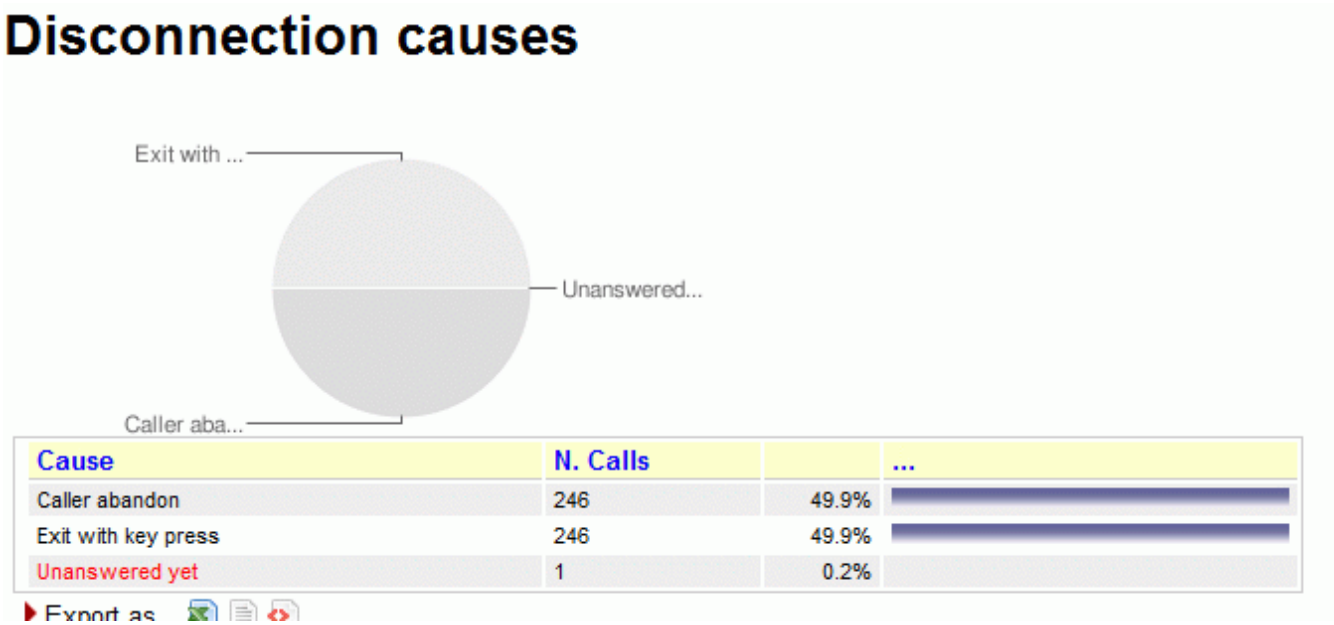
UN02 - Calls fully within the given time interval



The unanswered completed calls section deals with calls that were lost. This is similar to what's reported on previous panel but may exclude calls that were started before or terminated after the given time frame.

Available since	1.6.0
Default page	Unans.
Shortcut code	UN02
API block name	KoDO.ReportKoFully
Parameters	-
See also	

UN03 - Disconnection causes

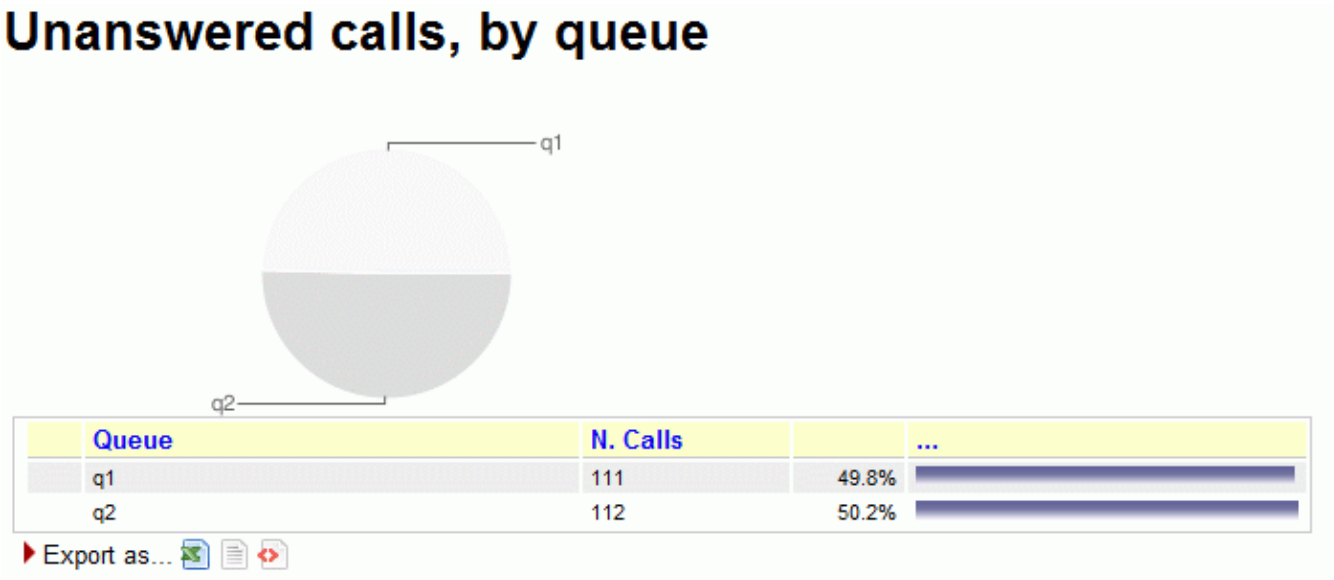


This report shows the relative magnitude of disconnection causes, that are:

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

Available since	1.6.0
Default page	Unans.
Shortcut code	UN03
API block name	KoDO.DiscCauses
Parameters	-
See also	

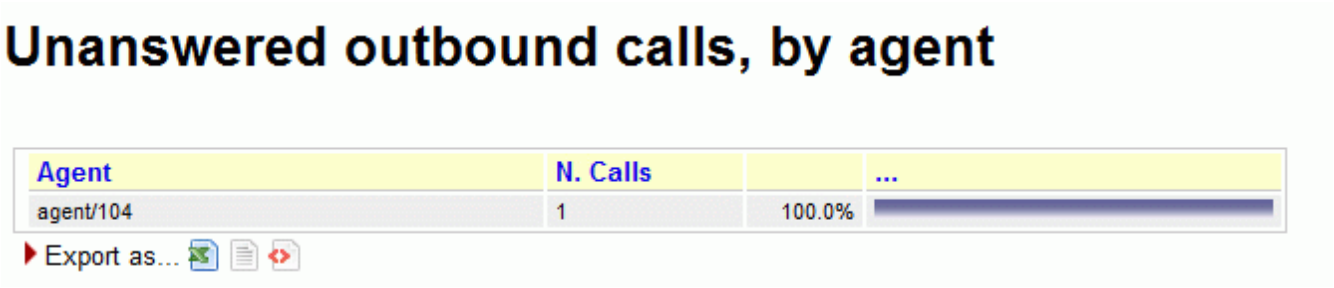
UN04 - Unanswered calls, by queue



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

Available since	1.6.0
Default page	Unans.
Shortcut code	UN04
API block name	KoDO.UnansByQueue
Parameters	-
See also	

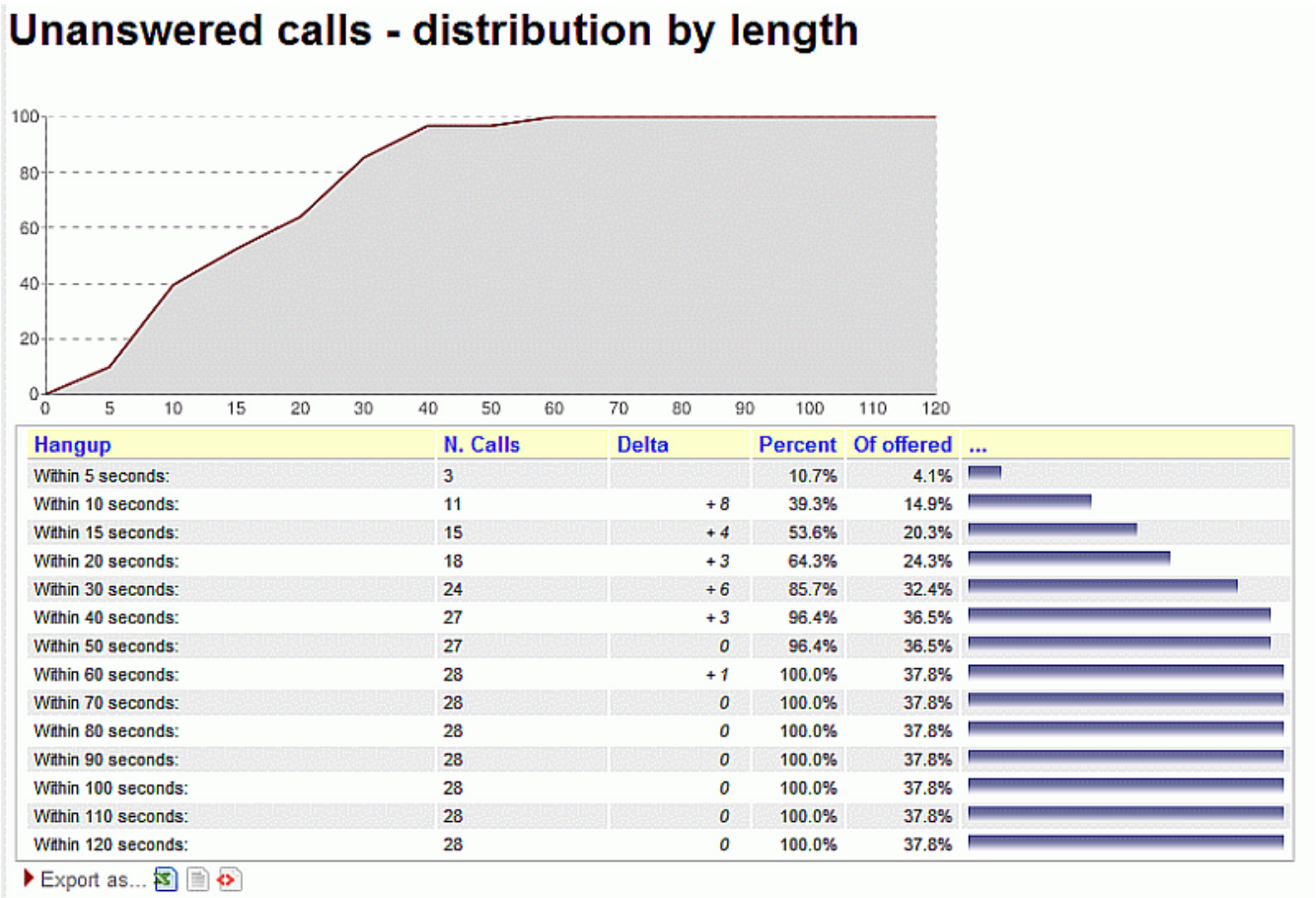
UN05 - Unanswered outbound calls, by agent



This graph shows the relative magnitude of unanswered outbound calls, grouped by agent.

Available since	1.6.0
Default page	Unans.
Shortcut code	UN05
API block name	KoDO.OutboundKo
Parameters	-
See also	

UN06 - Unanswered calls - distribution by length



This report is functionally equivalent to "Service level agreement" in the Answered calls section (see Service level agreement), 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 OK04 - Service level agreement).

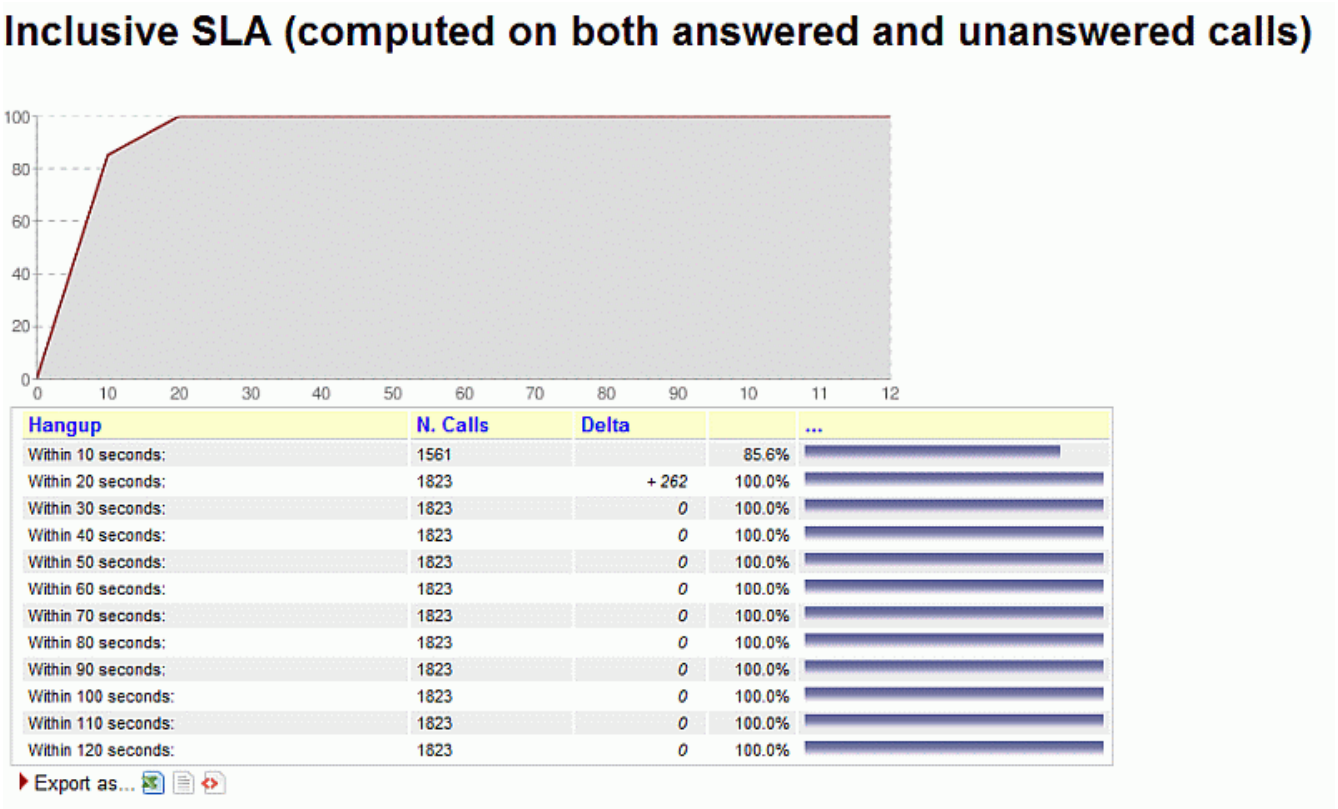
You get a percentage of how many calls were lost within X seconds; the percentage includes calls lost in a shorter time frame and therefore grows with time.

The "delta" value you see is the absolute increment, expressed in number of calls, between each time frame, while the "Offered" column displays the result of the taken calls divided by the total taken plus the total lost.

The graph reports the percentage of how many calls were not answered within X seconds, as reported in the table.

Available since	1.6.0
Default page	Unans.
Shortcut code	UN06
API block name	KoDO.UnansByLen
Parameters	-
See also	UN07 UN18 OK04

UN07 - Inclusive SLA (computed on both answered and unanswered calls)



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

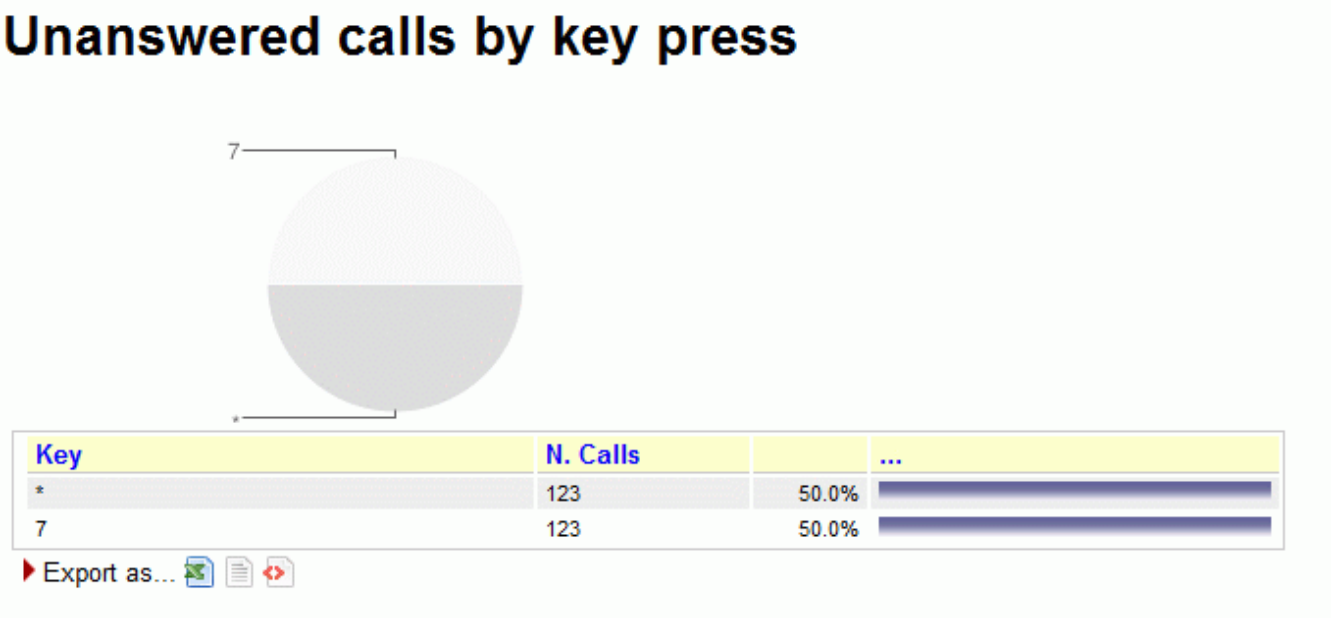
The difference between UN07 and UN18 is that the number of calls in UN18 is only the number of taken calls within the given answer period, while in UN07 it is the total number of taken and lost calls within the time period.

The graph reports the same information found in the table, but in a graphical way.

The time frame and increment can be modified by the administrator, if needed - see OK04 - Service level agreement.

Available since	1.6.0
Default page	Unans.
Shortcut code	UN07
API block name	KoDO.InclusiveSLA
Parameters	-
See also	OK04 UN06 UN18 (Inclusive Answered SLA)

UN08 - 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.

Available since	1.6.0
Default page	Unans.
Shortcut code	UN08
API block name	KoDO.ReportKoKeyPress
Parameters	-
See also	

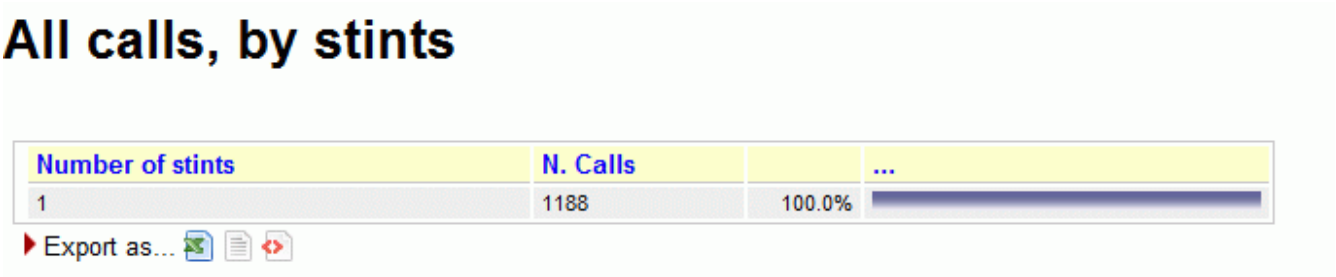
UN09 - Unanswered calls, by stints



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

Available since	1.6.0
Default page	Unans.
Shortcut code	UN09
API block name	KoDO.StintsKo
Parameters	-
See also	

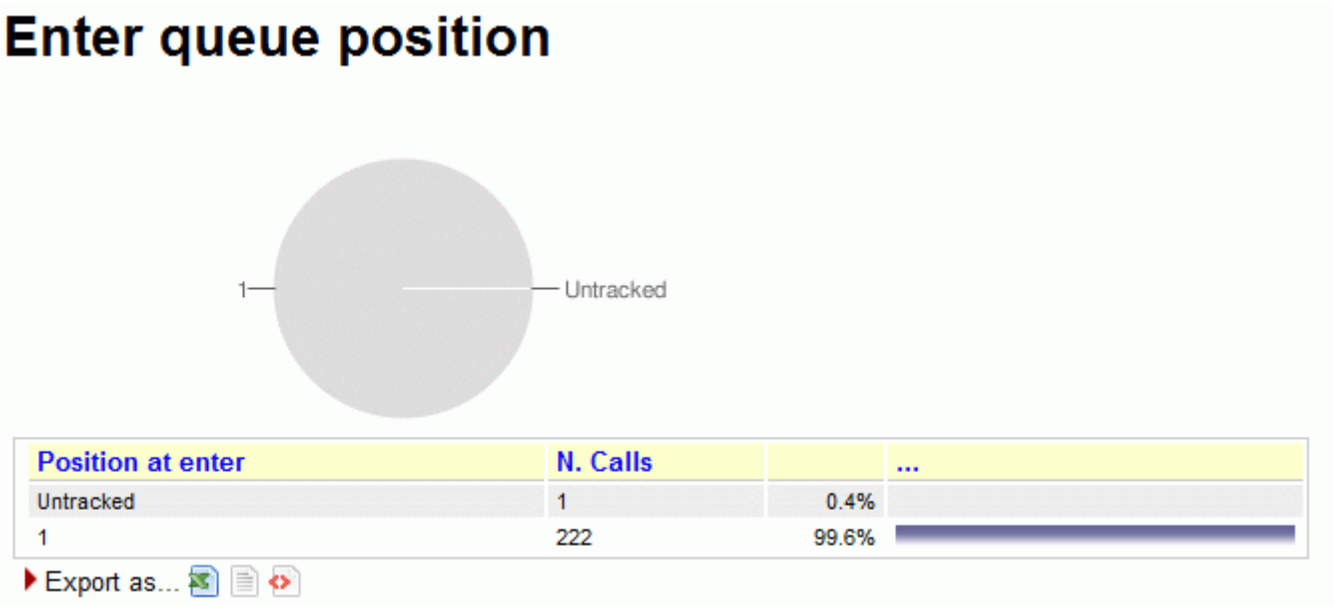
UN10 - All calls, by stints



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"

Available since	1.6.0
Default page	Unans.
Shortcut code	UN10
API block name	KoDO.StintsOkKo
Parameters	-
See also	

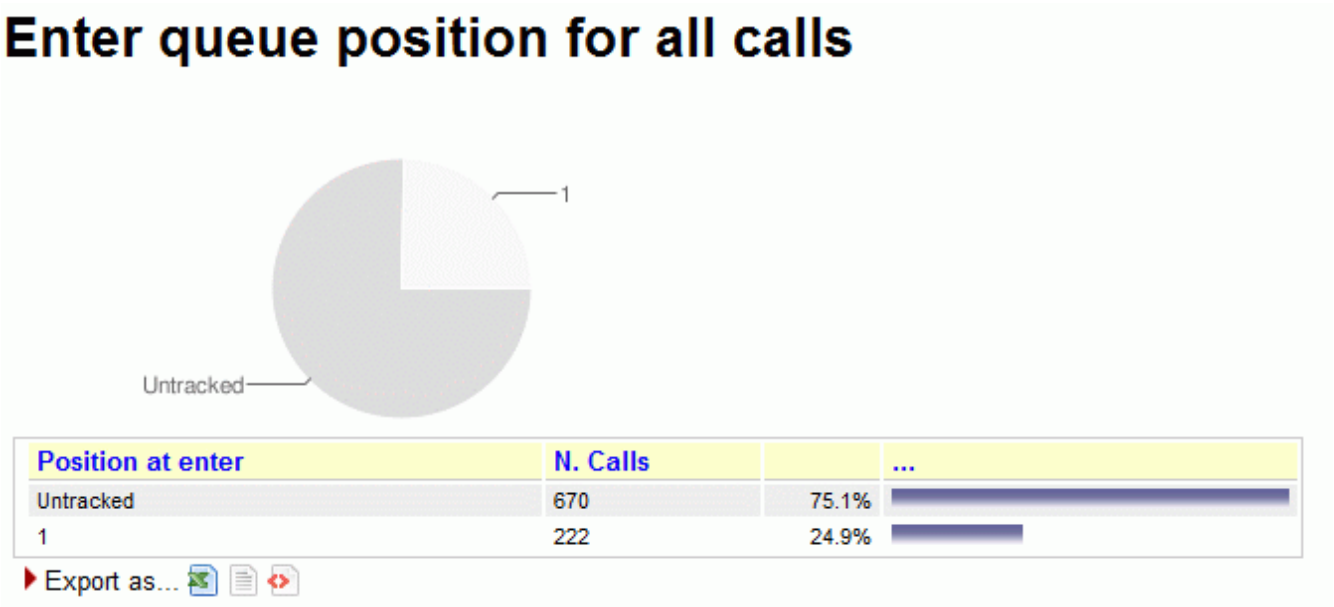
UN11 - Enter queue position



This graph shows the initial queue position that the calls had when they joined the queue. For example, a queue position of 1 means that a call was first in line, of 5 means that a call had four other calls in line before being answered. As the logging of queue positions is a bit inconsistent, some calls might be missing it ("Untracked")

Available since	1.6.0
Default page	Unans.
Shortcut code	UN11
API block name	KoDO.QPosKo
Parameters	-
See also	

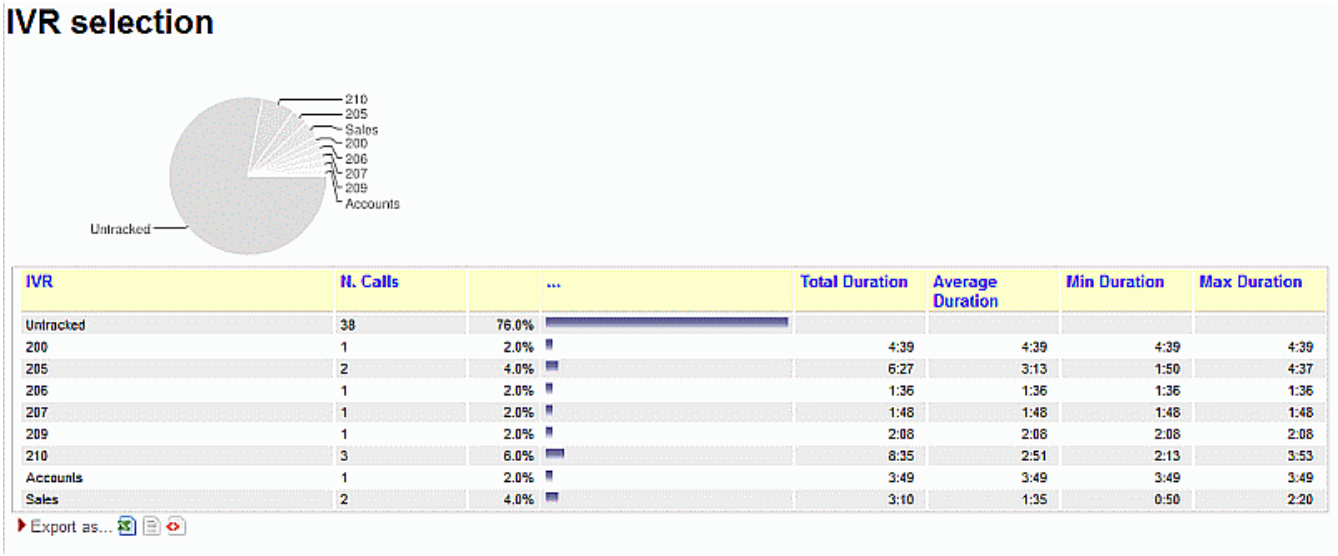
UN12 - Enter queue position for all calls



This graph shows the initial queue positions for both answered and unanswered calls.

Available since	1.6.0
Default page	Unans.
Shortcut code	UN12
API block name	KoDO.QPosOkKo
Parameters	-
See also	

UN13 - IVR selection



This graph shows the distribution of IVR selections available for lost calls.

We can also see the IVR duration values, which is related to the time that the call was within the IVR before entering the queue.

This must be tracked manually in Asterisk - See "Configuring Asterisk for QueueMetrics" below.

Available since	1.6.0
Default page	Unans.
Shortcut code	UN13
API block name	KoDO.IvrKo
Parameters	-
See also	

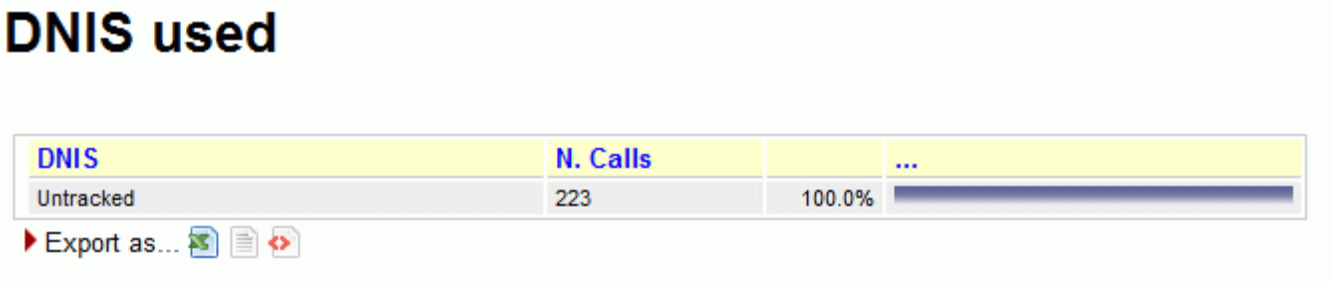
UN14 - IVR selection, for all calls



This graph shows the distribution of IVR selections for all calls (taken and lost) available in the report.

Available since	1.6.0
Default page	Unans.
Shortcut code	UN14
API block name	KoDO.IvrOkKo
Parameters	-
See also	

UN15 - DNIS used



This graph shows the distribution of DNIS lines that lead to lost calls.

This must be tracked manually in Asterisk - See "Configuring Asterisk for QueueMetrics" below.

Available since	1.6.0
Default page	Unans.
Shortcut code	UN15
API block name	KoDO.DnisKo
Parameters	-
See also	

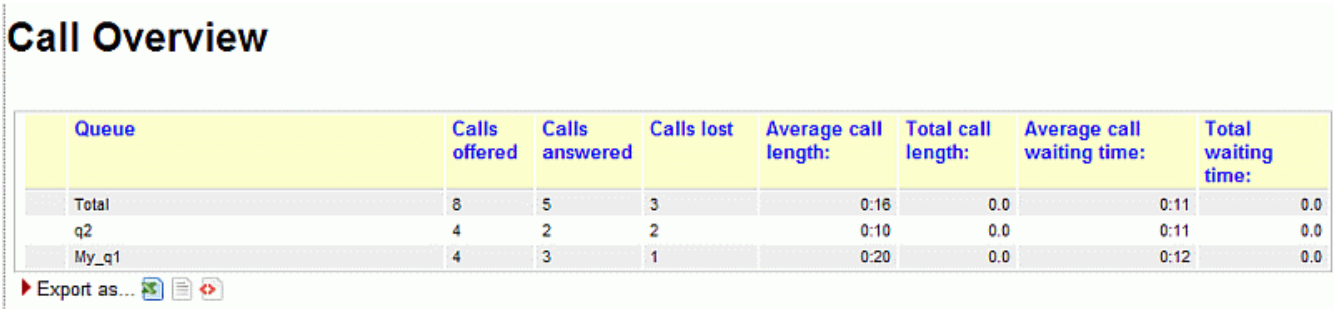
UN16 - DNIS used, for all calls



This graph shows the distribution of DNIS lines for all calls (taken and lost) available in the report.

Available since	1.6.0
Default page	Unans.
Shortcut code	UN16
API block name	KoDO.DnisOkKo
Parameters	-
See also	

UN17 - Call Overview



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

For each queue included in the report, the following data will be shown:

- Calls offered: total number of calls in the period to hit that queue
- Calls answered: total number of answered calls
- Lost calls: total number of calls that have not been answered
- Average call length: it computed only on the talk time of answered calls, expressed as MM:SS
- Total call length: the total cumulate speak time for each queue, expressed as decimalized hours
- Average wait Q: the average wait time for both answered and lost calls
- Total wait Q: total cumulate wait time for answered and unanswered calls, expressed as decimalized hours

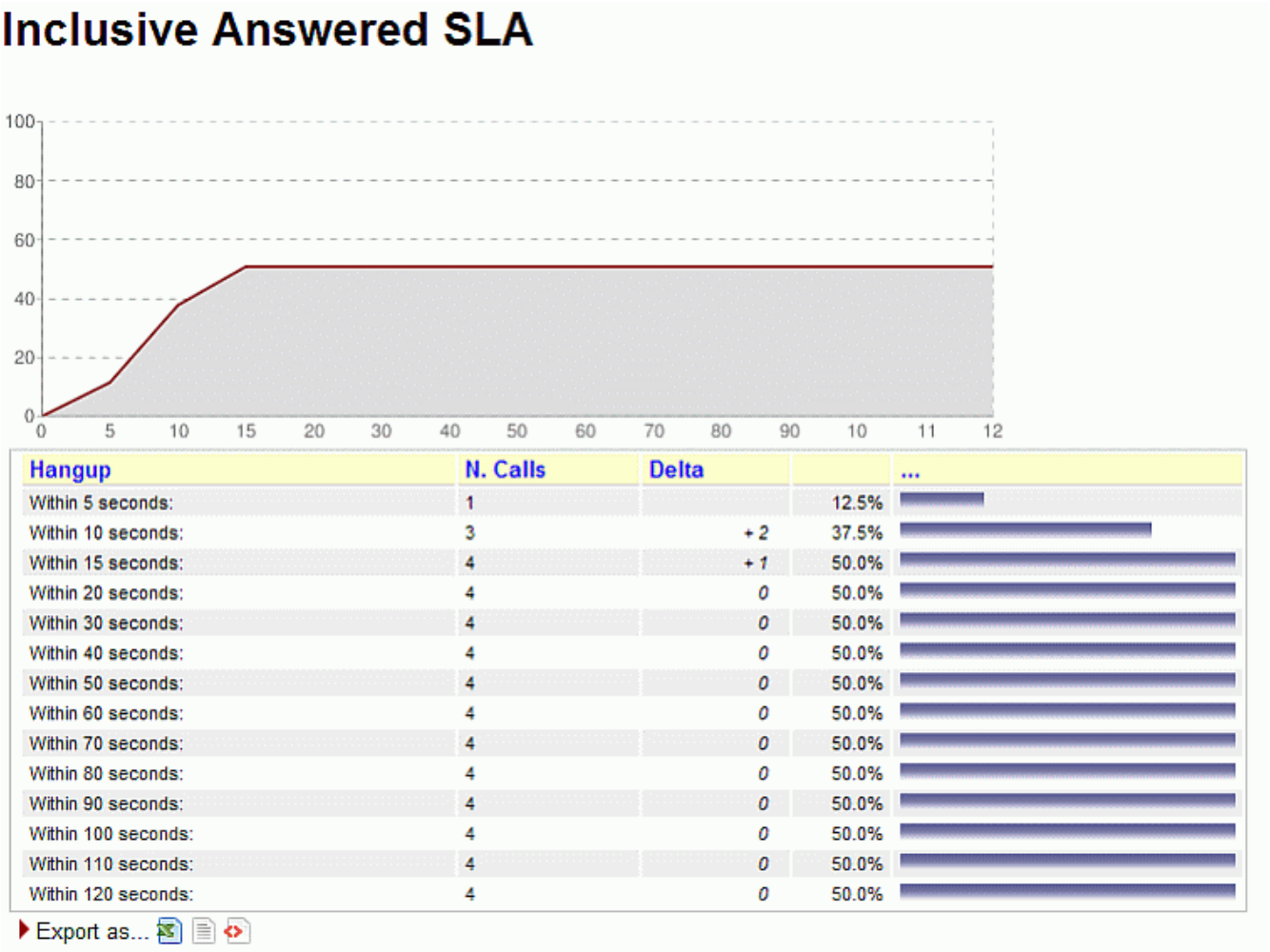
In the first line, there is a "Total" line, that sums up the values shown in previous lines.

The following lines are sorted according to the number of offered calls.

Available since	1.6.1.1

Default page	Unanswered calls
Shortcut code	UN17
API block name	KoDO.OverviewOkKo
Parameters	-
See also	

UN18 - Inclusive Answered SLA



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

The difference between UN07 and UN18 is that the number of calls in UN18 is only the number of taken calls within the given answer period, while in UN07 it is the total number of taken and lost calls within the time period.

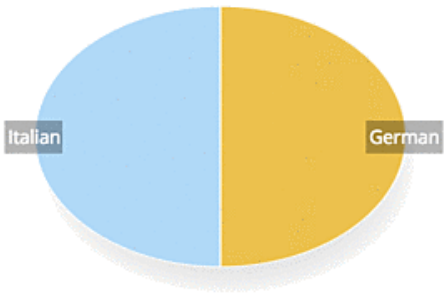
The graph reports the same information found in the table, but in a way that is easier to read.

The time frame and increment can be modified by the administrator, if needed - see OK04 - Service level agreement.

Available since	1.6.1.2

Default page	Unans.
Shortcut code	UN18
API block name	KoDO.InclusiveAnswSLA
Parameters	-
See also	UN07 (Inclusive SLA) UN06 OK04

UN19 - Required Skills for Lost Calls



Skill Required	N. Calls	Percent	Avg Wait	Avg Talk	Min lvl	Avg lvl	Max lvl
German	1	100.0%	0:20	0:00	30	30	30
Italian	1	100.0%	0:20	0:00	70	70	70

This data block displays the skills that were required for lost calls in the current reports.

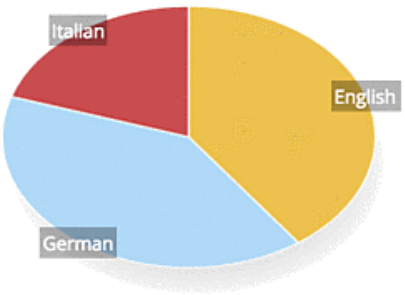
As multiple skills may be required for the same call, or some calls may have no required skill, the sum of all skills may be significantly different from the number of calls.

For each skill, the average wait duration of all calls having that specific skill is computed. As lost calls never have any talk time, it always appears as 0:00 and is displayed for symmetry with blocks OK15 and UN20.

The minimum, average and maximum level at which this skill was requested for the specific set of calls is displayed.

Available since	15.10
Default page	Unans.
Shortcut code	UN19
API block name	KoDO.SkillsKo
Parameters	-
See also	UN20, OK15

UN20 - Required Skills for All Calls



Skill Required	N. Calls	Percent	Avg Wait	Avg Talk	Min lvl	Avg lvl	Max lvl
English	2	66.7%	0:06	0:20	60	65	70
German	2	66.7%	0:11	0:05	30	60	90
Italian	1	33.3%	0:20	0:00	70	70	70

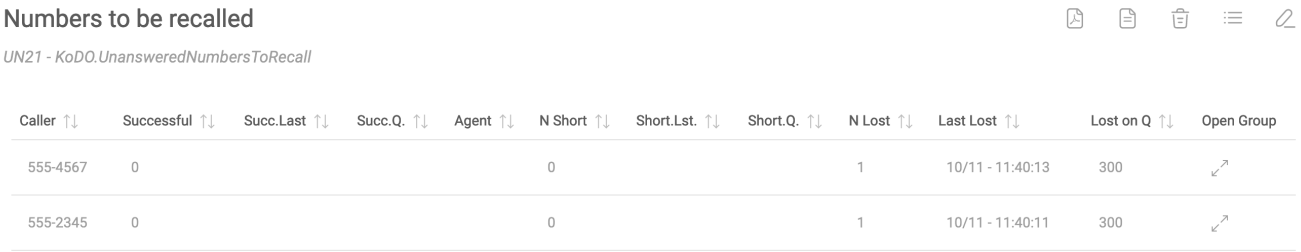
This data block displays the skills that were required for all calls in the current reports.

As multiple skills may be required for the same call, or some calls may have no required skill, the sum of all skills may be significantly different from the number of calls.

For each skill, the average wait and talk duration of all calls having that specific skill is computed. Plus, the minimum, average and maximum level at which this skill was requested for the specific set of calls is displayed.

Available since	15.10
Default page	Unans.
Shortcut code	UN20
API block name	KoDO.SkillsOkKo
Parameters	-
See also	UN19, OK15

UN21 - Numbers to be recalled



This data block displays the calling numbers that are to be recalled. A number is considered to be recalled if it matches the following criteria:

- it has at least one lost call within the period, on any queue, or it has at least one connected short call, and
- there is no connected call for the same number following the most recent lost/short call

(whether it's outbound or inbound)

So we can say that a number is to be recalled if the last call from that number was unsuccessful. In this data block we display a list of calls, by default sorted with the most recent attempt on top, where such calls are in states **Not yet recalled** (when there is no successful call at all) or **Needs further recall** (when there is a lost call after a successful call).

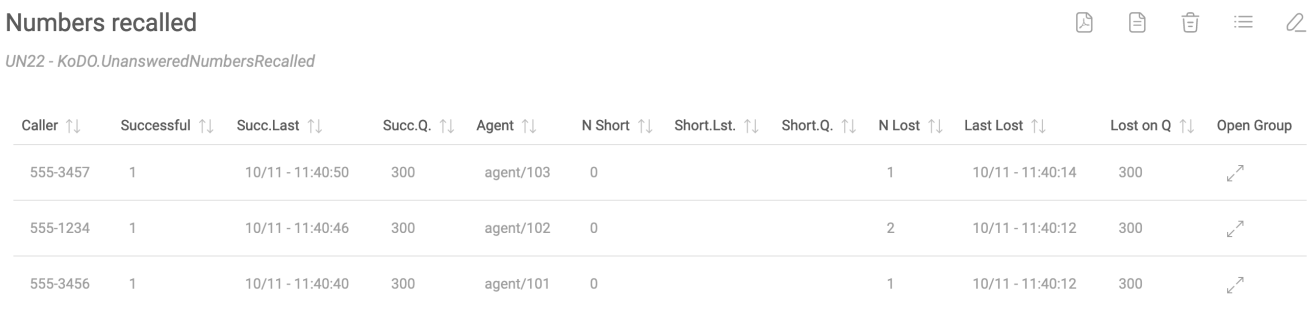
Available columns:

- *Caller*: the caller-id
- *Successful*: the number of successful calls made to the caller
- *Successful Last*: when the last successful call was made
- *Successful Queue*: the inbound queue or outbound campaign that the successful call was on
- *Agent*: The agent who was involved in the successful call
- *N Short*: the number of short calls
- *Short Last*: when the last short call was made
- *Short Queue*: the queue where the last short call happened
- *N Lost*: The number of lost calls by this caller
- *Last Lost*: When the last lost call was
- *Lost on Queue*: The queue where the last lost call was processed
- *Open group*: A group of all the calls involved (check the different tabs for Taken/Lost calls!)

When embedded in a wallboard, this block displays the current recall activity to be implemented and is a simple but effective way of keeping tab of missed contacts.

Available since	22.10
Default page	Unans.
Shortcut code	UN21
API block name	KoDO.UnansweredNumbersToRecall
Parameters	-
See also	UN22, UN23

UN22 - Numbers recalled

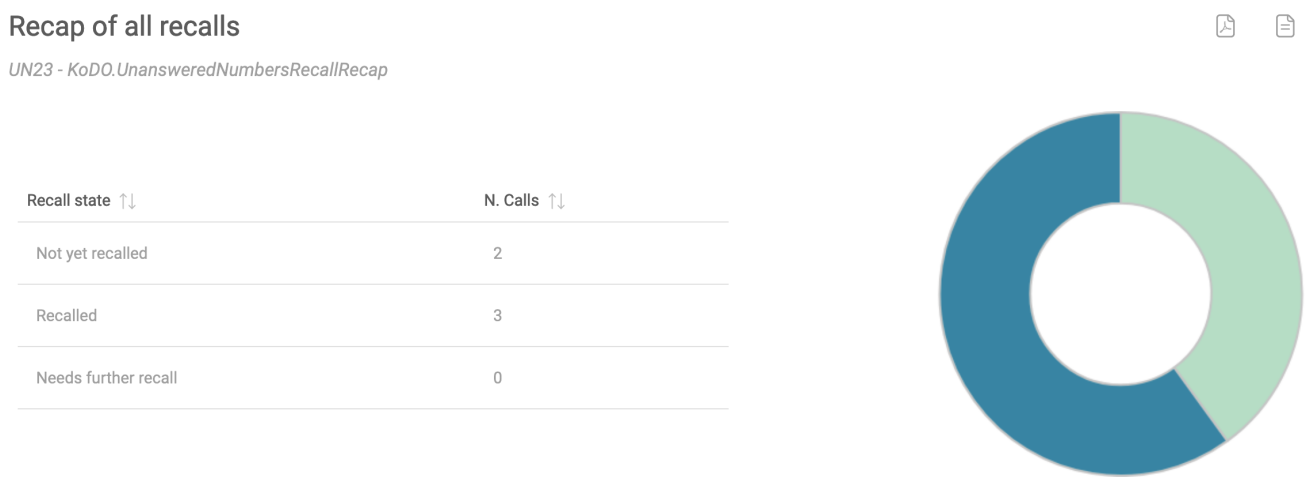


This data block displays callers that were **recalled**, that is, that had short or lost calls, but also had a subsequent complete call. SO the number of *Successful* calls will always be at least one.

Columns and the way they are computed are the same as block UN21.

Available since	22.10
Default page	Unans.
Shortcut code	UN21
API block name	KoDO.UnansweredNumbers Recalled
Parameters	-
See also	UN21, UN23

UN23 - Recap of all recalls



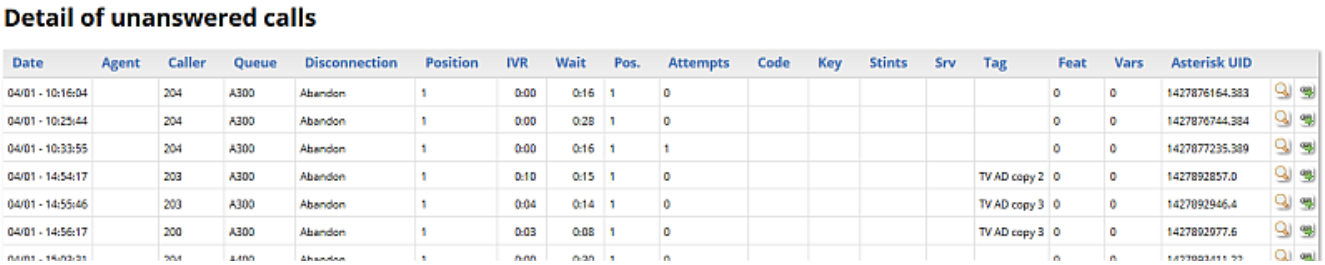
This block displays a quick at-a-glance state of how many calls were:

- **Not yet recalled** (as in UN21)
- **Recalled** (as in UN22)
- **Need further recall** (as in UN21)

Available since	22.10
Default page	Unans.
Shortcut code	UN21
API block name	KoDO.UnansweredNumbers RecallRecap
Parameters	-
See also	UN21, UN22

Historical reports - Details of unanswered calls

UD01 - Detail of unanswered calls



Detail of unanswered calls

Date	Agent	Caller	Queue	Disconnection	Position	IVR	Wait	Pos.	Attempts	Code	Key	Stints	Srv	Tag	Feat	Vars	Asterisk UID
04/01 - 10:16:04		204	A300	Abandon	1	0:00	0:16	1	0						0	0	1427876164.383
04/01 - 10:25:44		204	A300	Abandon	1	0:00	0:28	1	0						0	0	1427876744.384
04/01 - 10:33:55		204	A300	Abandon	1	0:00	0:16	1	1						0	0	1427877235.389
04/01 - 14:54:17		203	A300	Abandon	1	0:10	0:15	1	0					TV AD copy 2	0	0	1427892857.0
04/01 - 14:55:46		203	A300	Abandon	1	0:04	0:14	1	0					TV AD copy 3	0	0	1427892946.4
04/01 - 14:56:17		200	A300	Abandon	1	0:03	0:08	1	0					TV AD copy 3	0	0	1427892977.6

This page shows full details of unanswered calls Detail of unanswered calls.

Available since	1.6.0
Default page	Details of unanswered calls
Shortcut code	UD01
API block name	DetailsDO.CallsKO
Parameters	-
See also	UD02 - Lost calls (paged)

UD02 - Lost calls (paged)



Details of unanswered calls

Date	Agent	Caller	Queue	Disconnection	Position	IVR	Wait	Pos.	Attempts	Code	Key	Stints	Srv	Tag	Feat	Vars
04/01 - 10:16:04		204	A300	Abandon	1	0:00	0:16	1	0						0	0
04/01 - 10:25:44		204	A300	Abandon	1	0:00	0:28	1	0						0	0
04/01 - 10:33:55		204	A300	Abandon	1	0:00	0:16	1	1						0	0
04/01 - 14:54:17		203	A300	Abandon	1	0:10	0:15	1	0					TV AD copy 2	0	0
04/01 - 14:55:46		203	A300	Abandon	1	0:04	0:14	1	0					TV AD copy 3	0	0
04/01 - 14:56:17		200	A300	Abandon	1	0:03	0:08	1	0					TV AD copy 3	0	0
04/01 - 15:03:31		204	A400	Abandon	1	0:00	0:30	1	0						0	0
04/01 - 15:55:23		200	A300	Abandon	1	0:03	3:50	1	0					TV AD copy 3	0	0

This page shows the paged detail of unanswered calls. See Showing call details.

The details of calls are paged - so you can move back and forth - and you can control which columns will appear on screen. The paged mode uses way less memory and is quicker to use than the full list as in UD01 - Detail of unanswered calls.

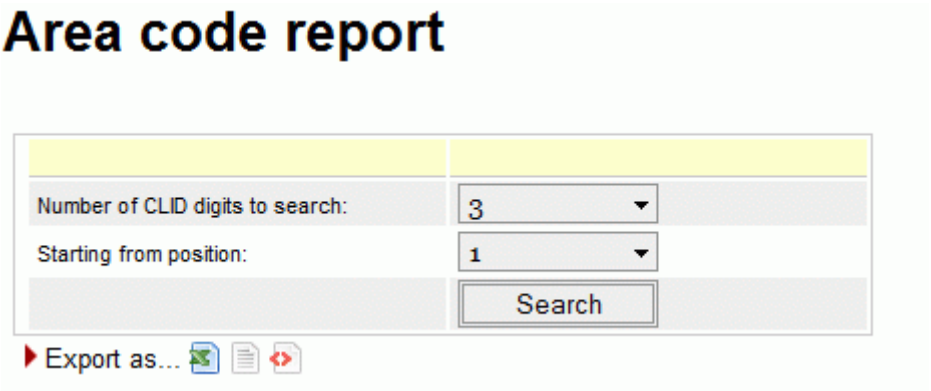
When accessing the data source though XML-RPC or printing a report, it is mandatory to use the UD01 block, as this block is interactive and will not render correctly.

Available since	13.03 - **Removed in 20.11**
Default page	Details of unanswered calls
Shortcut code	UD02

API block name	-
Parameters	-
See also	UD01 - Detail of unanswered calls and Configuring paged call lists

Historical reports - Area code analysis

AC01 - 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.

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.

Available since	1.6.0 - **Removed in 20.11**
Default page	Area code analysis
Shortcut code	AC01
API block name	AreaAnDO.Setup
Parameters	-
See also	

AC02 - Detail for answered calls



This report shows the answered calls grouped following the rules defined in AC01 - Area code report.

Available since	1.6.0
Default page	Area code analysis
Shortcut code	AC02
API block name	AreaAnDO.CallsOK
Parameters	-
See also	AC01 - Area code report

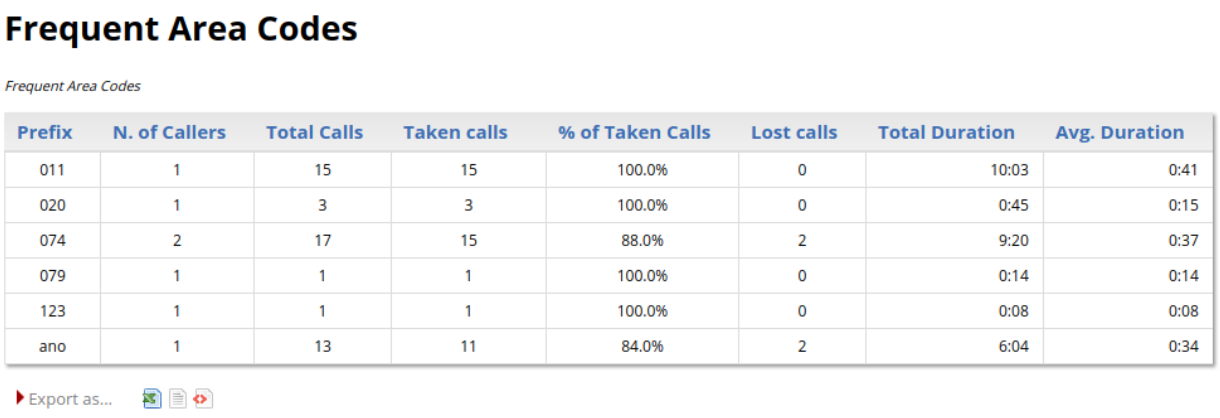
AC03 - Detail for unanswered calls



This report shows the unanswered calls grouped following the rules defined in AC01 - Area code report.

Available since	1.6.0
Default page	Area code analysis
Shortcut code	AC03
API block name	AreaAnDO.CallsKO
Parameters	-
See also	AC01 - Area code report

AC04 - Frequent Area Codes



This report shows detailed information regarding different Area Codes. We take into account the quantity and quality of calls related to different areas, by calculating the following information:

- Prefix
- N. of Callers
- Totale Calls
- Taken Calls
- % of Taken Calls
- Lost Calls
- Total Duration
- Avg. Duration

Available since	17.06.16
Default page	Area code analysis
Shortcut code	AC04
API block name	AreaAnDO.FrequentAreaCodes
Parameters	-
See also	AC01 - Area code report

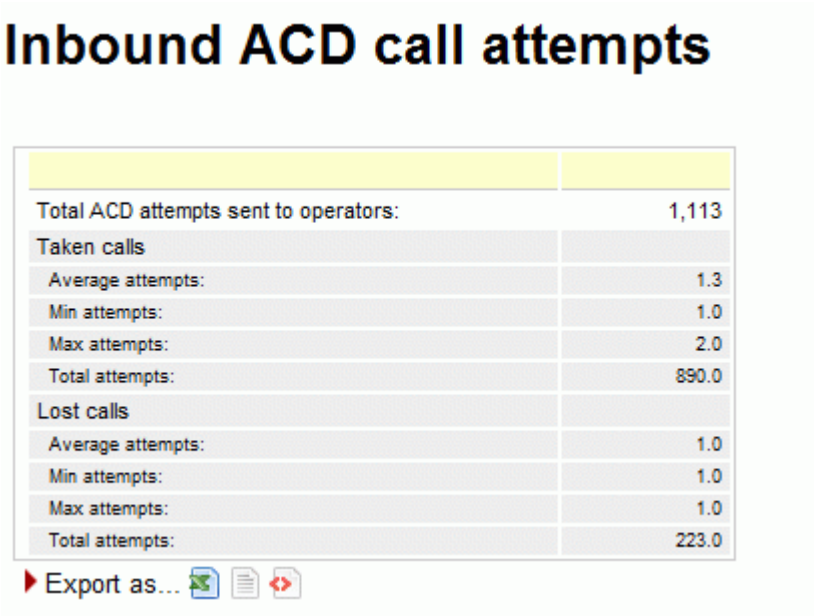
Historical reports - Distributions

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? Is the phone malfunctioning? The inbound ACD call attempts metrics try to answer to these questions.



If your Asterisk system supports `RINGCANCELED` events, those events are displayed and counted on the call details, but will be purposedly ignored for these data blocks. This is because those are not taken attempts, but not lost attempts either, as it is the ACD itself "retiring" the attempts. By adding them to lost or taken attempts, we would end up inflating these numbers providing no actual insights. On the other hand, if you have a call that has one `RINGNOANSWER`, one `RINGCANCELED` and was connected, the call list will say that the call has three attempts, but these distributions will say it just has two.

AT01 - Inbound ACD call attempts

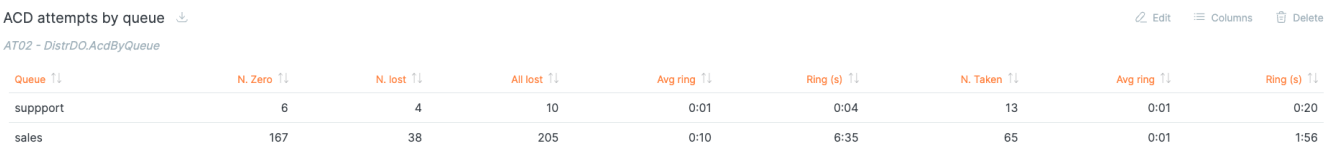


This page shows the following pieces of information:

- 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

Available since	1.6.0
Default page	Distrib.
Shortcut code	AT01
API block name	DistrDO.ReportAcd
Parameters	-
See also	

AT02 - ACD attempts by queue



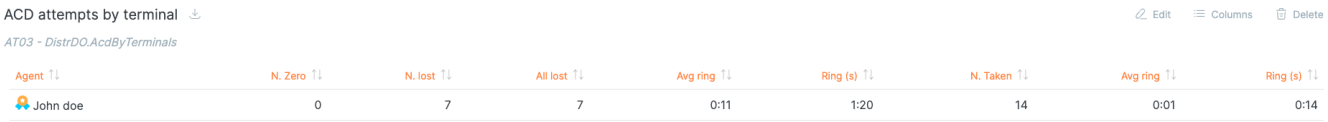
The following metrics are extracted and broken down by queue:

- N. Zero: The number of lost agent attempts with a ring duration of 0 seconds.
- N. Lost: The number of lost agent attempts with a ring duration greater than 0 seconds (excluding "Zero" attempts).
- All Lost: The total number of lost agent attempts (the sum of N. Zero and N. Lost).

- Avg. Ring (Lost): The average ring time for lost attempts, calculated exclusively for attempts with a ring duration greater than 0 seconds.
- Total Ring (Lost): The total cumulative ringing time for all lost calls.
- N. Taken: The number of taken agent attempts (i.e., calls successfully answered).
- Avg. Ring (Taken): The average ring duration for taken calls.
- Total Ring (Taken): The total cumulative ring time for taken calls.

Available since	1.6.0
Default page	Distrib.
Shortcut code	AT02
API block name	DistrDO.AcdByQueue
Parameters	-
See also	

AT03 - ACD attempts by terminal

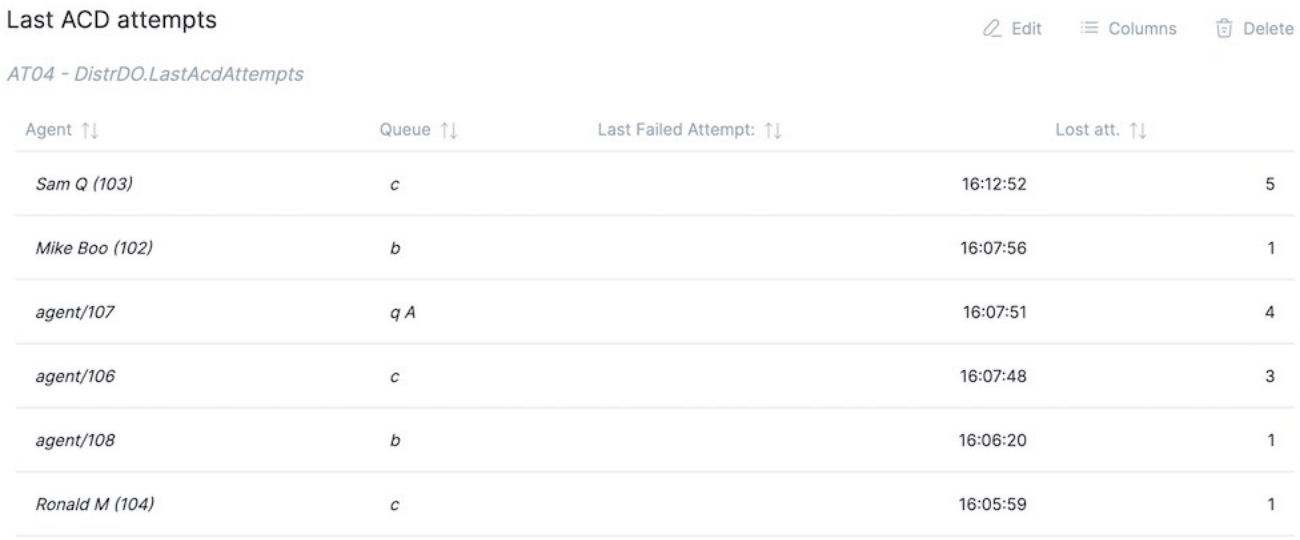


This graph breaks down agent attempts by the agent that was called. The following pieces of information are extracted for each agent:

- N. Zero: The number of lost attempts with a ring duration of 0 seconds.
- N. Lost: The number of lost attempts with a ring duration greater than 0 seconds.
- All Lost: The total number of lost attempts (the sum of N. Zero and N. Lost).
- Avg. Ring (Lost): The average ring time for lost attempts, calculated only for attempts with a ring duration greater than 0 seconds.
- Total Ring (Lost): The total ringing time for all lost calls.
- N. Taken: The number of taken agent attempts (i.e., calls answered).
- Avg. Ring (Taken): The average ring duration for taken calls.
- Total Ring (Taken): The total ring time for taken calls.

Available since	1.6.0
Default page	Distrib.
Shortcut code	AT03
API block name	DistrDO.AcdByTerminals
Parameters	-
See also	

AT04 - Last ACD Attempts



This data block shows the last failed ACD attempts for each agent and queue included in the report. It is computed on all closed calls (taken and lost).

For each agent and queue, you can see:

- when the last failed attempt was placed
- how may lost attempts for this agent and queue were made

They are sorted in reverse chronological order, so that the most recent entry is always on top (but you can, of course, change the way it is sorted).



Thi is useful as the basis of a monitoring system to be immediately aware of lost attempts when they are made - e.g. to be put on a wallboard, so that you can see in real-time which agents are piling up lost attempts.

Available since	24.11
Default page	Distrib.
Shortcut code	AT04
API block name	DistrDO.LastAcdAttempts
Parameters	-
See also	

Historical reports - Call distribution by day

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

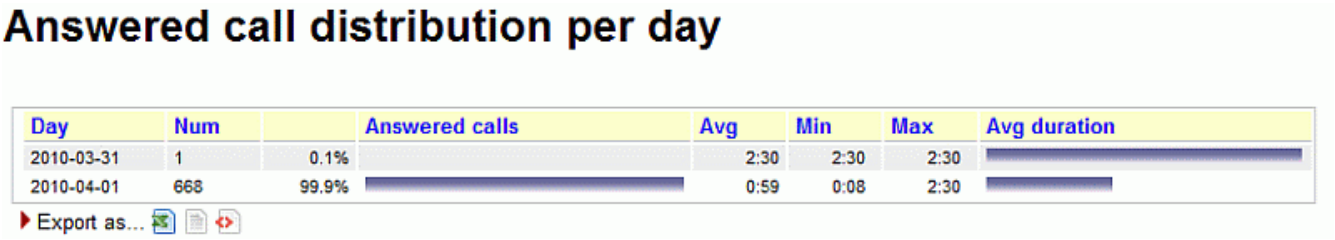
All percentages are calculated on the call class they belong to, i.e. a 50% of "Unanswered calls" on one day means that 50% of all unanswered calls for the period happened during that day, not that

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.

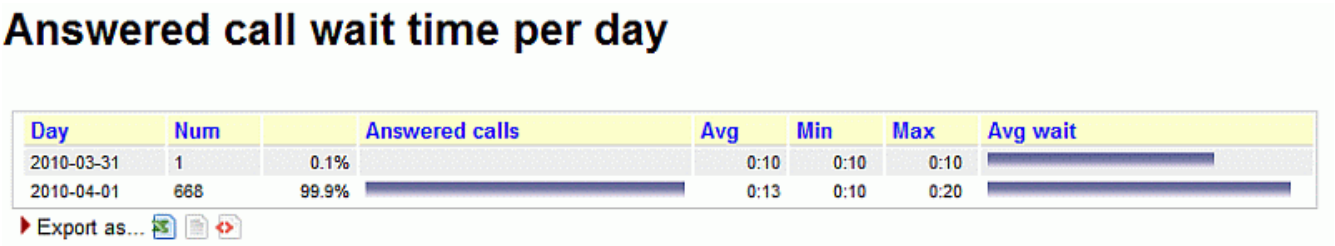
DD01 - Answered call distribution per day



Taken calls are shown per specific day. Days with no events are not shown.

Available since	1.6.0
Default page	Call distribution, by day
Shortcut code	DD01
API block name	CallDistrDO.AnsDistrPerDay
Parameters	-
See also	

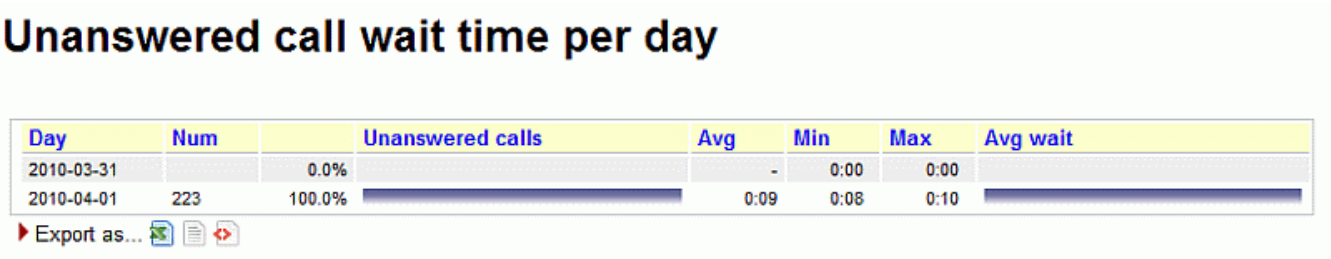
DD02 - Answered call wait time per day



The total numbers of call wait time for answered calls are plotted for each day.

Available since	1.6.0
Default page	Call distribution, by day
Shortcut code	DD02
API block name	CallDistrDO.AnsWaitPerDay
Parameters	-
See also	

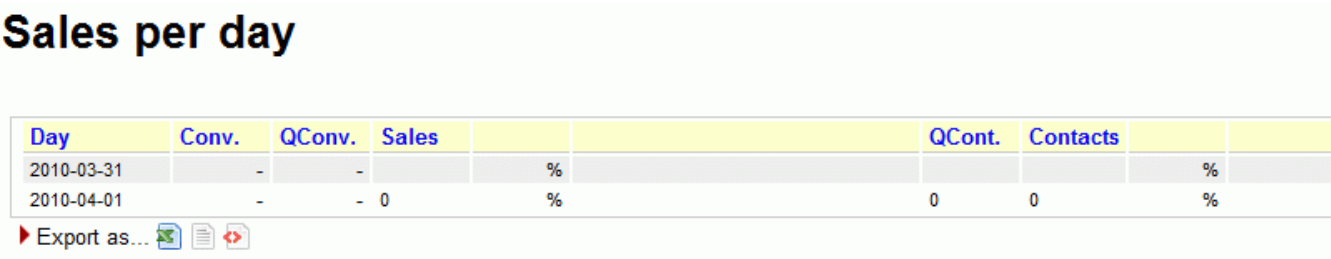
DD03 - Unanswered call wait time per day



The total numbers of lost wait time for lost calls are plotted for each day.

Available since	1.6.0
Default page	Call distribution, by day
Shortcut code	DD03
API block name	CallDistrDO.UnansWaitPerDay
Parameters	-
See also	

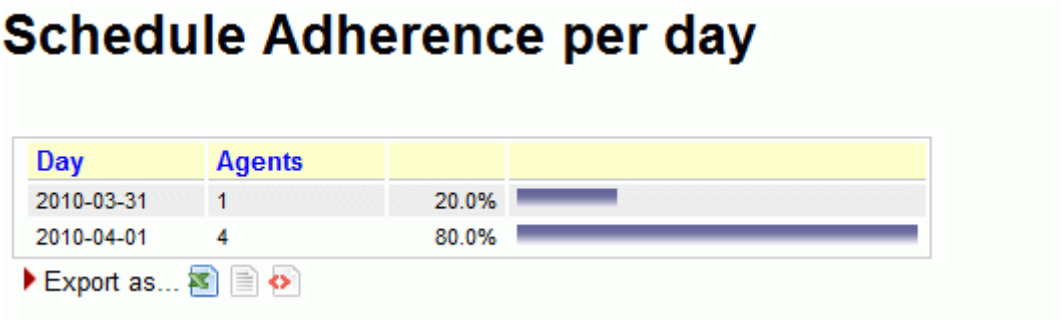
DD04 - Sales per day



Sales and contacts are shown on a daily basis.

Available since	1.6.0
Default page	Call distribution, by day
Shortcut code	DD04
API block name	CallDistrDO.SalesPerDay
Parameters	-
See also	

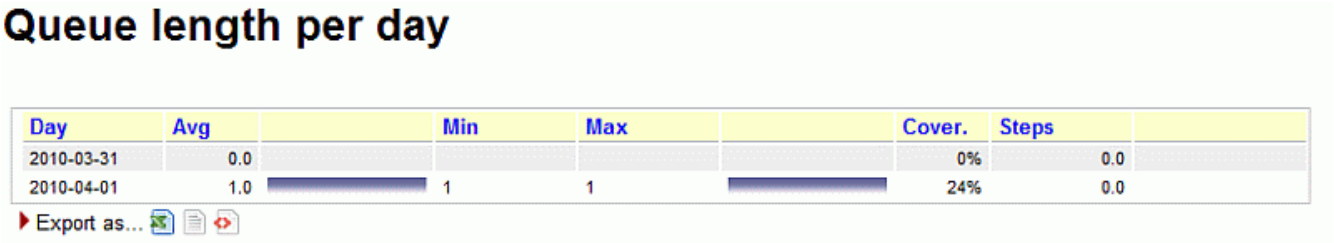
DD05 - Schedule Adherence per day



The Schedule Adherence report shows the number of distinct agents that were detected during the given period. This makes it possible to detect the number of different people that had been working on a given moment.

Available since	1.6.0
Default page	Call distribution, by day
Shortcut code	DD05
API block name	CallDistrDO.StaffPerDay
Parameters	-
See also	

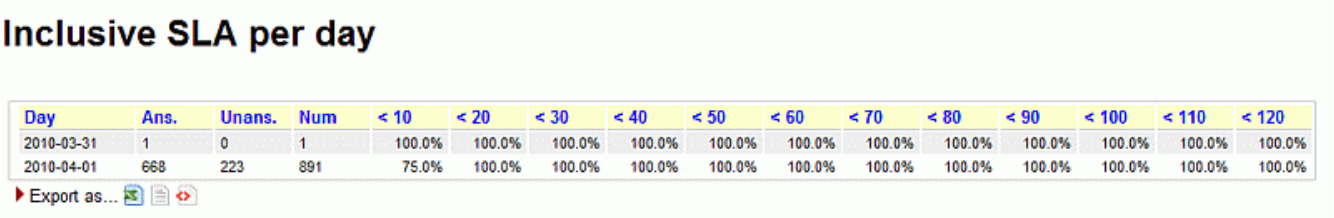
DD06 - Queue length per day



The Queue Length report shows the average length of the queue for each period, giving minimums and maximums. The Steps computation shows how fast each queue progresses during the period, expresses in steps per hour. Note: giving a meaning to the Queue Length reports may be hard in the case of composite queues.

Available since	1.6.0
Default page	Call distribution, by day
Shortcut code	DD06
API block name	CallDistrDO.QPosPerDay
Parameters	-
See also	

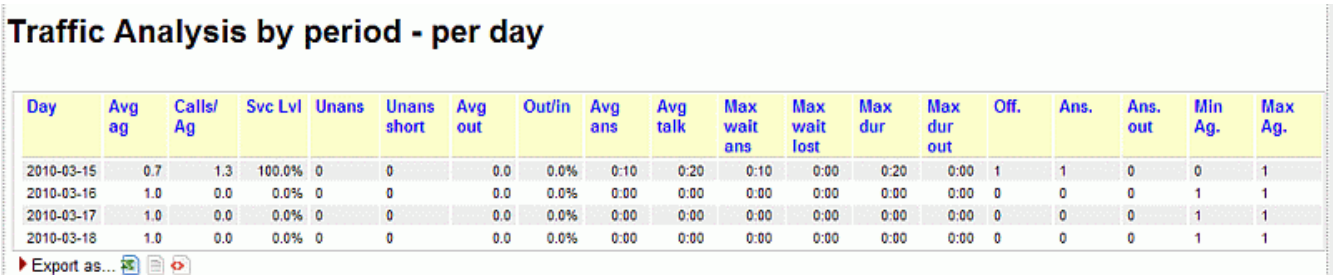
DD07 - Inclusive SLA per day



The inclusive SLA corresponds to the Service Level Agreement metrics shown on Service level agreement, grouped by day.

Available since	1.6.0
Default page	Call distribution, by day
Shortcut code	DD07
API block name	CallDistrDO.InclSlaPerDay
Parameters	-
See also	

DD08 - Traffic Analysis by period - per day



This report shows aggregate inbound/outbound activity per day.

- Date: the date, hour or day of week used for the report.
- Avg agents: the average number of agents, as calculated by agents logged in for each period versus the total period. E.g. if an agent logs in at 10:00 and logs off at 11:30, and a second agent logs in at 10:15 and logs off at 11:00, the period for 10:00 to 10:30 will show 1.5 available agents.
- Avg calls/agent: number of INCOMING calls (answered+unanswered) per period versus average available agents
- Service level: the SLA (see below), computed on INCOMING calls only
- Unans: Lost calls (INCOMING only)
- Unans short: Lost calls below X seconds (INCOMING only, as a percentage of all calls)
- Avg out: number of OUTGOING calls (completed and lost) per period versus number of available agents
- Out/in: ratio of outbound to inbound
- Avg ans. Average answer time (for INCOMING only)
- Avg talk time: Average talk time (for INCOMING only)
- Max wait ans: Maximum answer time in period (for INCOMING only)

- Max wait lost: Maximum wait time for lost calls in period (for INCOMING only)
- Max duration: Maximum talk time in period (for INCOMING only)
- Max duration OUT: Maximum talk time in period (for OUTGOING only)
- N. offered: Total number of INCOMING calls
- N Answered: Total number of answered INCOMING calls
- N Answered out: Total number of answered OUTGOING calls
- Min Agents: minimum number of agents logged on for the period
- Max Agents: maximum number of agents logged on in the period

The Service Level is measured against a time period that is specidfied in the *default.secondsServiceLevel* parameter (default is 20, as to say "percentage of calls answered within 20 seconds").

Short calls are defined as being shorter than the *default.shortCallsLimit* parameter - default is 5 seconds.

Available since	1.6.0.4
Default page	Call distribution, by day
Shortcut code	DD08
API block name	CallDistrDO.TrafficAnPerDay
Parameters	-
See also	DH08, DW08

DD09 - Skill sets per day

This reports shows the unique sets of skills during a specified time period.

Skills per Day

Day	Ans.	Unans.	Num	Within SLA: 10s.	Product1 english internet technical	Product1 english invoice land	Product1 english invoice mobile	Product1 english land others	Product1 english mobile others	Product1 english mobile technical
2016-12-04	0	0	0	-	0	0	0	0	0	0
2016-12-05	2,505	223	2,728	45.3%	0	2	8	7	4	8
2016-12-06	1,737	133	1,870	50.9%	0	0	4	2	4	5
2016-12-07	1,656	55	1,711	58.1%	1	0	3	0	4	3
2016-12-08	1,488	50	1,538	56.7%	1	2	5	0	2	1
2016-12-09	1,745	40	1,785	56.8%	3	0	9	0	6	0
2016-12-10	711	26	737	53.9%	0	0	0	0	4	0

The report is computed out of all calls in the current analysis. Time bands are set as for the current report (system property that can be overridden and is "remembered" across sessions). The SLA is set for a queue (or set of queues to be reported on at once) though the parameter "Service Level Agreement" on the Queue configuration page; if missing, the default value is taken from the property `sla.max_initial_delay`.

Each call is grouped into a "bucket" based on skills found; skills are sorted by **Skill Sort Weight** in order to create unique buckets for each permutation. Only buckets with at least one call are added to the report. Buckets are displayed with codes correctly decoded. To change the order of how the skills appear inside each bucket, make sure to adjust the **Skill Sort Weight** field in the **Agents Skill** editor. Skills with a lower sort weight are positioned before skills with higher sort weight. If two skills have the same sort weight, they are sorted alphabetically based on their description.

For example, a call with skills “inv en fix” is set to a bucket “en fix inv”, that is a separate bucket from e.g. “en fix” but is is the same as “en inv fix” . Such a call is displayed in a column named “English Fix Invoice”, where the names are based on the descripitions used for skills, sorted by the skills' sort weight.

The report shows the following columns:

- Time Slot: a call is banded in a time slot based on the time it is queued
- Answered calls: Total number of answered calls for this slot
- Unanswered calls: total number of lost calls for this slot
- Num: Total Answered + Unanswered
- Within SLA `X` sec: the number of calls answered within X seconds divided the total number of calls offered (answered+lost)
- "bucket": the total number of calls (taken+lost) that belong to this bucket

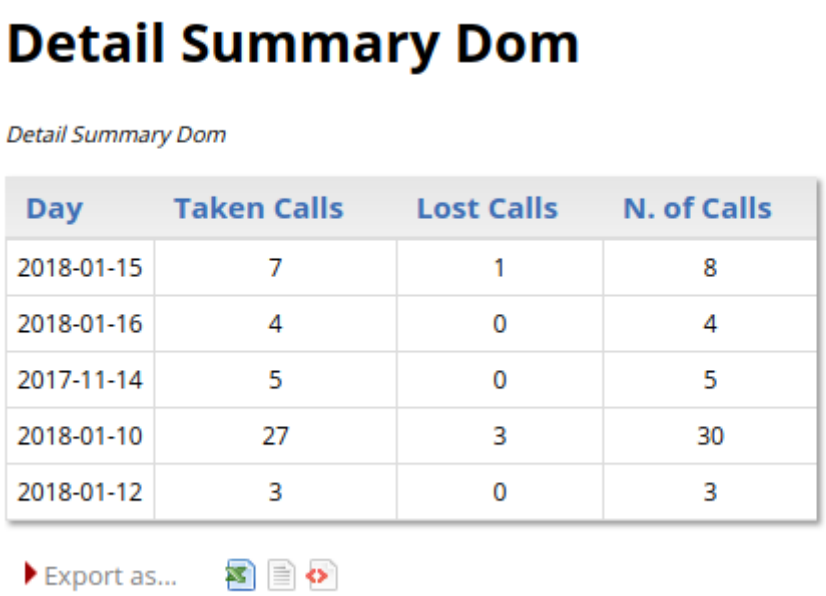


As the report may be very large, if you export it to PDF it may be cut on the right side. XLS report works without problems instead.

Available since	16.10.11
Default page	Call distribution, by day
Shortcut code	DD09
API block name	CallDistrDO.SkillsPerDay
Parameters	-
See also	DH09, DW09

DD10 - Detail Summary

This reports shows the number of taken and lost call by day.



This report comes in three version, that show the number of calls in different time slices. The available period of times are:

- Hourly [DH10]
- Days of the week [DW10]
- Days (with date) [DD10]

The information is the following

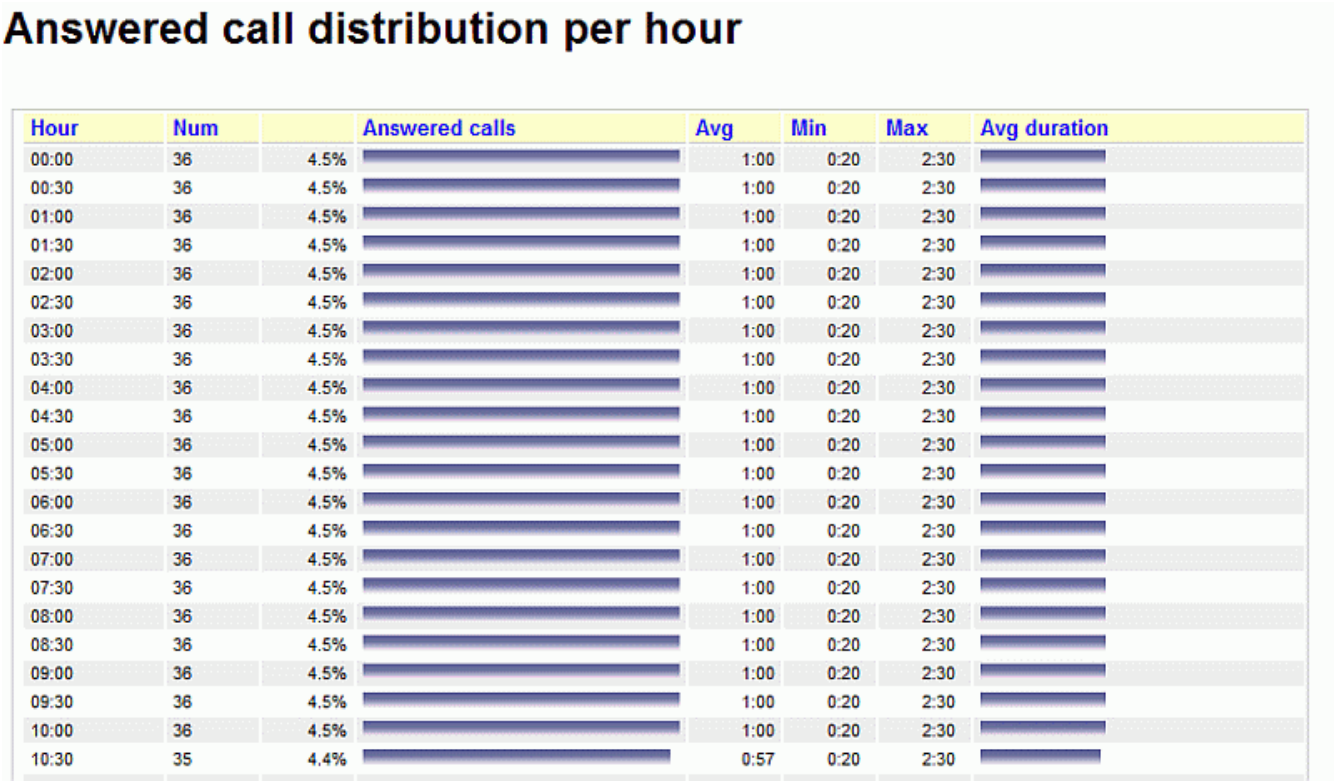
- Taken Calls
- Lost Calls
- N. of Calls

Available since	17.06.16
Default page	Detail Summary, by day
Shortcut code	DD10
API block name	CallDistrDO.DetailSummaryDom
Parameters	-
See also	DH10, DW10

Historical reports - Call distribution by hour

Events are shown on a 24-hour distribution.

DH01 - Answered call distribution per hour



Answered call distribution per hour

Hour	Num		Answered calls	Avg	Min	Max	Avg duration
00:00	36	4.5%		1:00	0:20	2:30	
00:30	36	4.5%		1:00	0:20	2:30	
01:00	36	4.5%		1:00	0:20	2:30	
01:30	36	4.5%		1:00	0:20	2:30	
02:00	36	4.5%		1:00	0:20	2:30	
02:30	36	4.5%		1:00	0:20	2:30	
03:00	36	4.5%		1:00	0:20	2:30	
03:30	36	4.5%		1:00	0:20	2:30	
04:00	36	4.5%		1:00	0:20	2:30	
04:30	36	4.5%		1:00	0:20	2:30	
05:00	36	4.5%		1:00	0:20	2:30	
05:30	36	4.5%		1:00	0:20	2:30	
06:00	36	4.5%		1:00	0:20	2:30	
06:30	36	4.5%		1:00	0:20	2:30	
07:00	36	4.5%		1:00	0:20	2:30	
07:30	36	4.5%		1:00	0:20	2:30	
08:00	36	4.5%		1:00	0:20	2:30	
08:30	36	4.5%		1:00	0:20	2:30	
09:00	36	4.5%		1:00	0:20	2:30	
09:30	36	4.5%		1:00	0:20	2:30	
10:00	36	4.5%		1:00	0:20	2:30	
10:30	35	4.4%		0:57	0:20	2:30	

The total numbers of call lengths for answered calls 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.

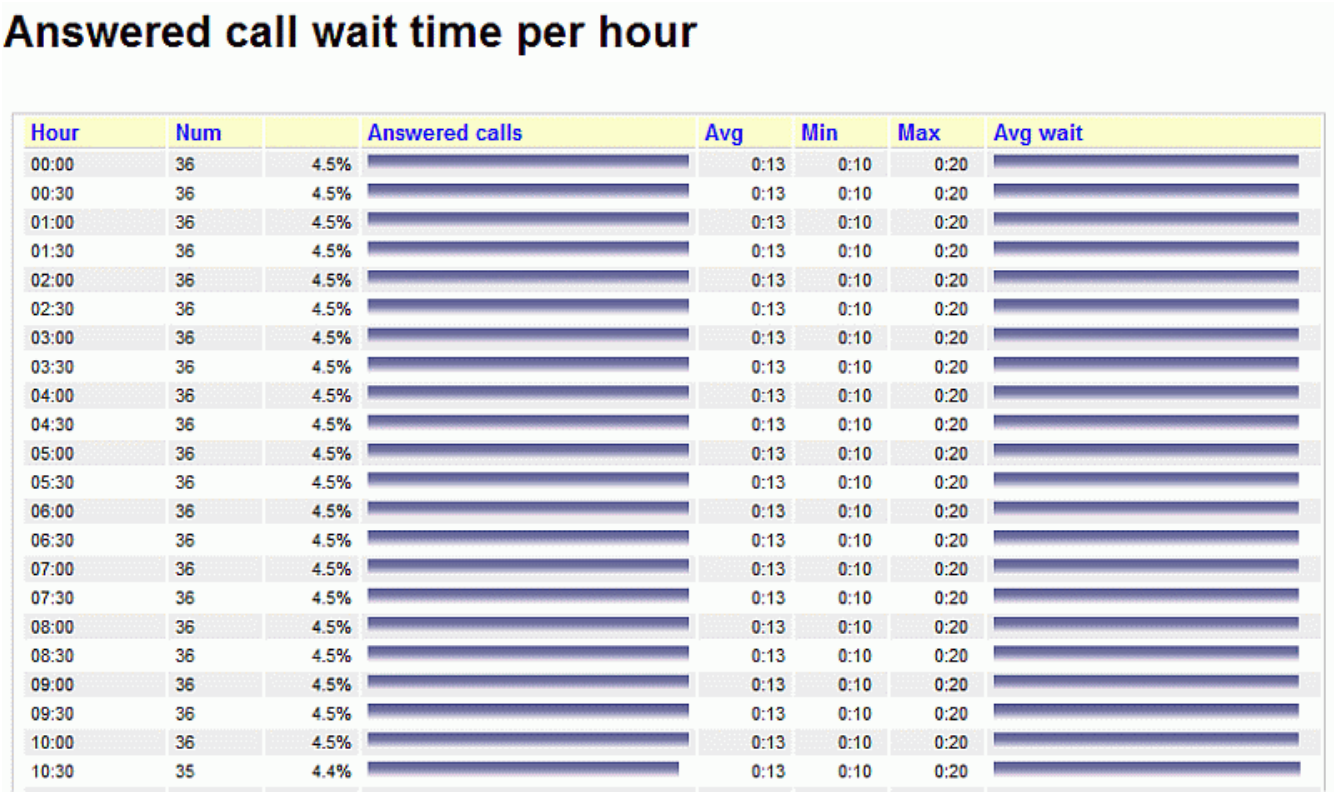
Enabling the *Hide empty hour slots* toggle in the Report settings panel, only time intervals that contain at least one answered call are shown. The report spans from the time of the first answered call to the time of the last answered call.

For example, if the first answered call occurs at 08:00AM and the last at 06:00PM, only the intervals from 08:00AM to 06:00PM (08:00-18:00) will appear in the report.

Available since	1.6.0

Default page	Call distribution, by hour
Shortcut code	DH01
API block name	CallDistrDO.AnsDistrPerHr
Parameters	-
See also	

DH02 - Answered call wait time per hour



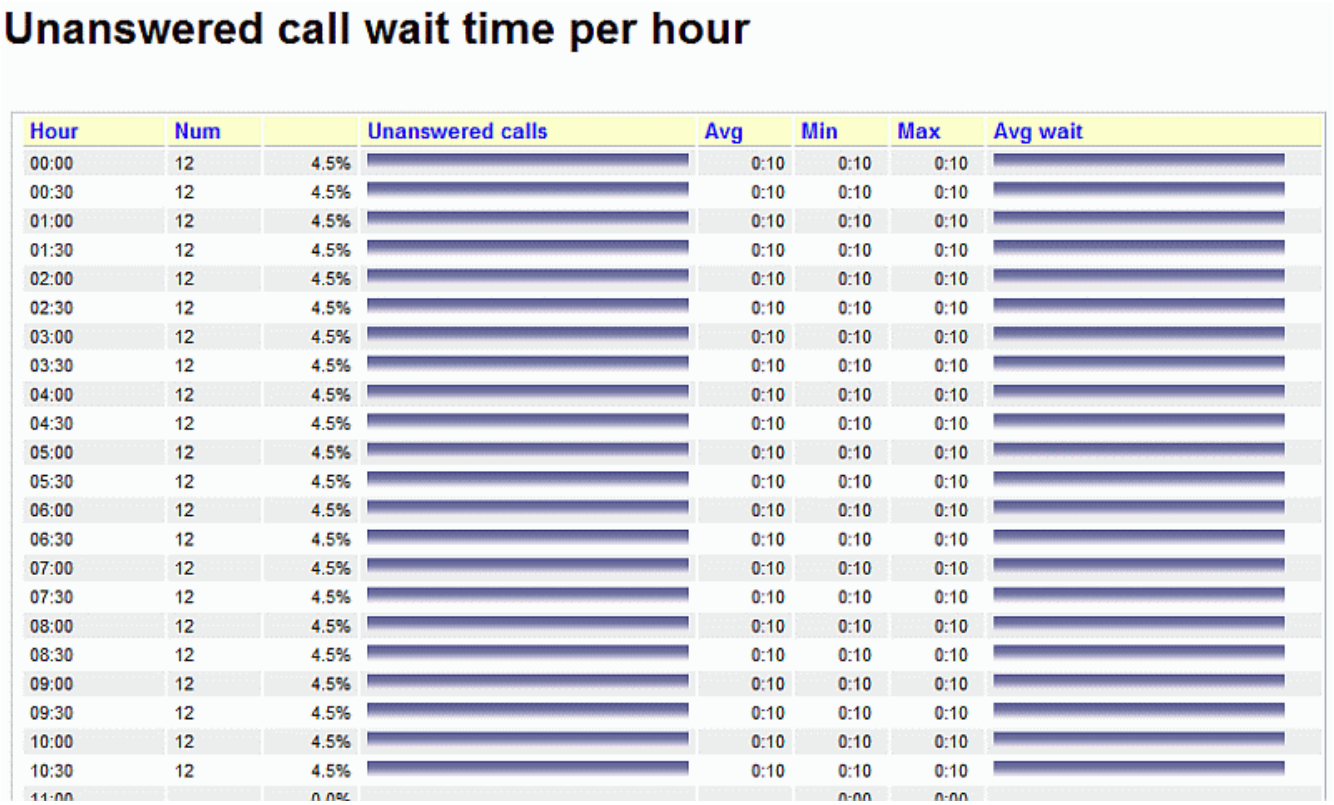
The total numbers of call wait time for answered calls are plotted for each hourly interval. The size of hourly intervals can be controlled by the *default.hourly_slot* configuration property, making it possible to run this reports based on 30-minute, 20-minute or 15-minute intervals.

Enabling the *Hide empty hour slots* toggle in the Report settings panel, only time intervals that contain at least one answered call are shown. The report spans from the time of the first answered call to the time of the last answered call.

For example, if the first answered call occurs at 08:00AM and the last at 06:00PM, only the intervals from 08:00AM to 06:00PM (08:00-18:00) will appear in the report.

Available since	1.6.0
Default page	Call distribution, by hour
Shortcut code	DH02
API block name	CallDistrDO.AnsWaitPerHr
Parameters	-
See also	

DH03 - Unanswered call wait time per hour



Unanswered call wait time per hour

Hour	Num		Unanswered calls	Avg	Min	Max	Avg wait
00:00	12	4.5%		0:10	0:10	0:10	
00:30	12	4.5%		0:10	0:10	0:10	
01:00	12	4.5%		0:10	0:10	0:10	
01:30	12	4.5%		0:10	0:10	0:10	
02:00	12	4.5%		0:10	0:10	0:10	
02:30	12	4.5%		0:10	0:10	0:10	
03:00	12	4.5%		0:10	0:10	0:10	
03:30	12	4.5%		0:10	0:10	0:10	
04:00	12	4.5%		0:10	0:10	0:10	
04:30	12	4.5%		0:10	0:10	0:10	
05:00	12	4.5%		0:10	0:10	0:10	
05:30	12	4.5%		0:10	0:10	0:10	
06:00	12	4.5%		0:10	0:10	0:10	
06:30	12	4.5%		0:10	0:10	0:10	
07:00	12	4.5%		0:10	0:10	0:10	
07:30	12	4.5%		0:10	0:10	0:10	
08:00	12	4.5%		0:10	0:10	0:10	
08:30	12	4.5%		0:10	0:10	0:10	
09:00	12	4.5%		0:10	0:10	0:10	
09:30	12	4.5%		0:10	0:10	0:10	
10:00	12	4.5%		0:10	0:10	0:10	
10:30	12	4.5%		0:10	0:10	0:10	
11:00		0.0%			0:00	0:00	

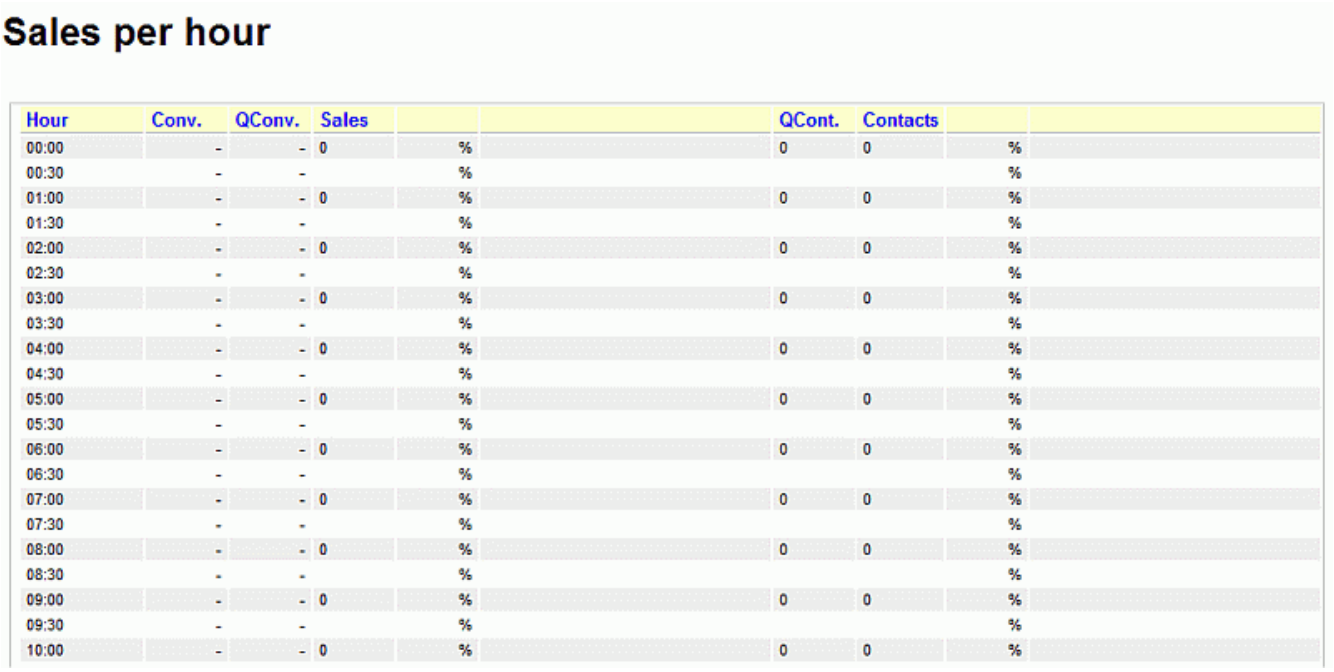
The total numbers of call lengths for unanswered calls are plotted for each hourly interval. The size of hourly intervals can be controlled by the *default.hourly_slot* configuration property, making it possible to run this reports based on 30-minute, 20-minute or 15-minute intervals.

Enabling the *Hide empty hour slots* toggle in the Report settings panel, only time intervals that contain at least one answered call are shown. The report spans from the time of the first answered call to the time of the last answered call.

For example, if the first answered call occurs at 08:00AM and the last at 06:00PM, only the intervals from 08:00AM to 06:00PM (08:00-18:00) will appear in the report.

Available since	1.6.0
Default page	Call distribution, by hour
Shortcut code	DH03
API block name	CallDistrDO.UnansWaitPerHr
Parameters	-
See also	

DH04 - Sales per hour



Sales per hour

Hour	Conv.	QConv.	Sales			QCont.	Contacts		
00:00	-	-	0	%		0	0	%	
00:30	-	-		%				%	
01:00	-	-	0	%		0	0	%	
01:30	-	-		%				%	
02:00	-	-	0	%		0	0	%	
02:30	-	-		%				%	
03:00	-	-	0	%		0	0	%	
03:30	-	-		%				%	
04:00	-	-	0	%		0	0	%	
04:30	-	-		%				%	
05:00	-	-	0	%		0	0	%	
05:30	-	-		%				%	
06:00	-	-	0	%		0	0	%	
06:30	-	-		%				%	
07:00	-	-	0	%		0	0	%	
07:30	-	-		%				%	
08:00	-	-	0	%		0	0	%	
08:30	-	-		%				%	
09:00	-	-	0	%		0	0	%	
09:30	-	-		%				%	
10:00	-	-	0	%		0	0	%	

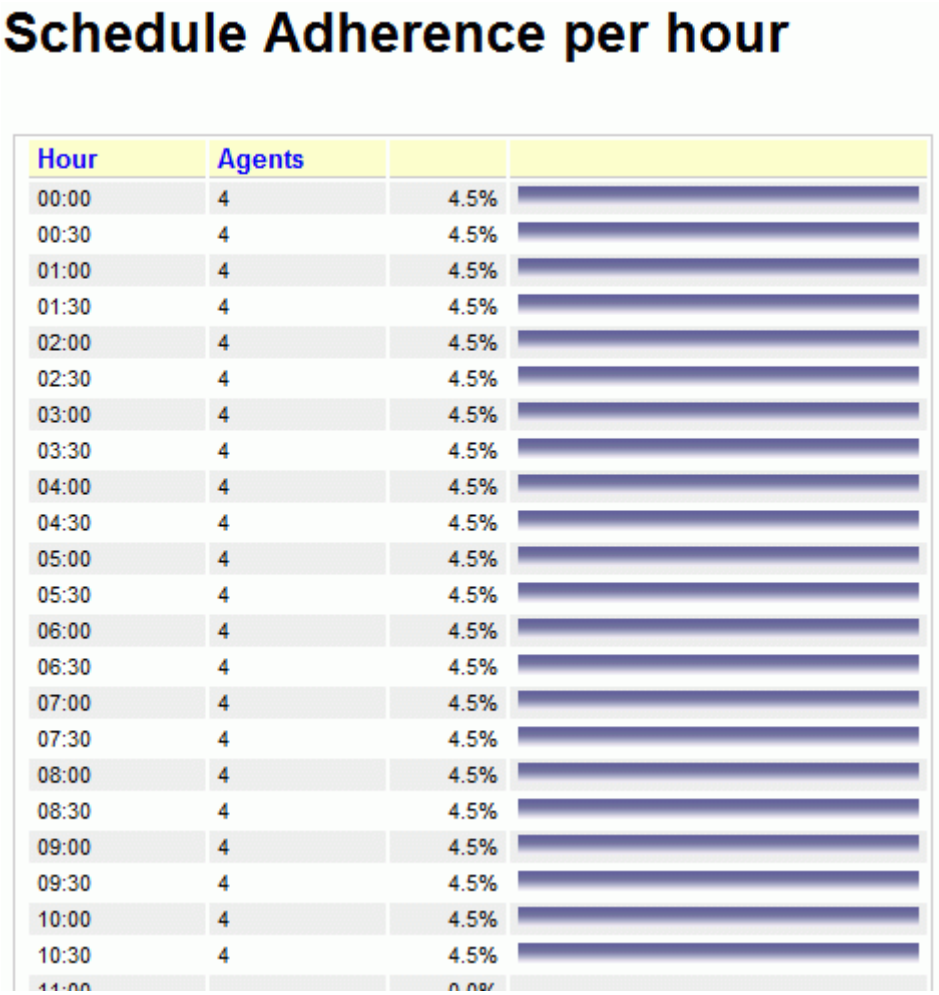
The total numbers sales and contacts, are plotted for each hourly interval. The size of hourly intervals can be controlled by the *default.hourly_slot* configuration property, making it possible to run this reports based on 30-minute, 20-minute or 15-minute intervals.

Enabling the *Hide empty hour slots* toggle in the Report settings panel, only time intervals that contain at least one answered call are shown. The report spans from the time of the first answered call to the time of the last answered call.

For example, if the first answered call occurs at 08:00AM and the last at 06:00PM, only the intervals from 08:00AM to 06:00PM (08:00-18:00) will appear in the report.

Available since	1.6.0
Default page	Call distribution, by hour
Shortcut code	DH04
API block name	CallDistrDO.SalesPerHr
Parameters	-
See also	

DH05 - Schedule Adherence per hour



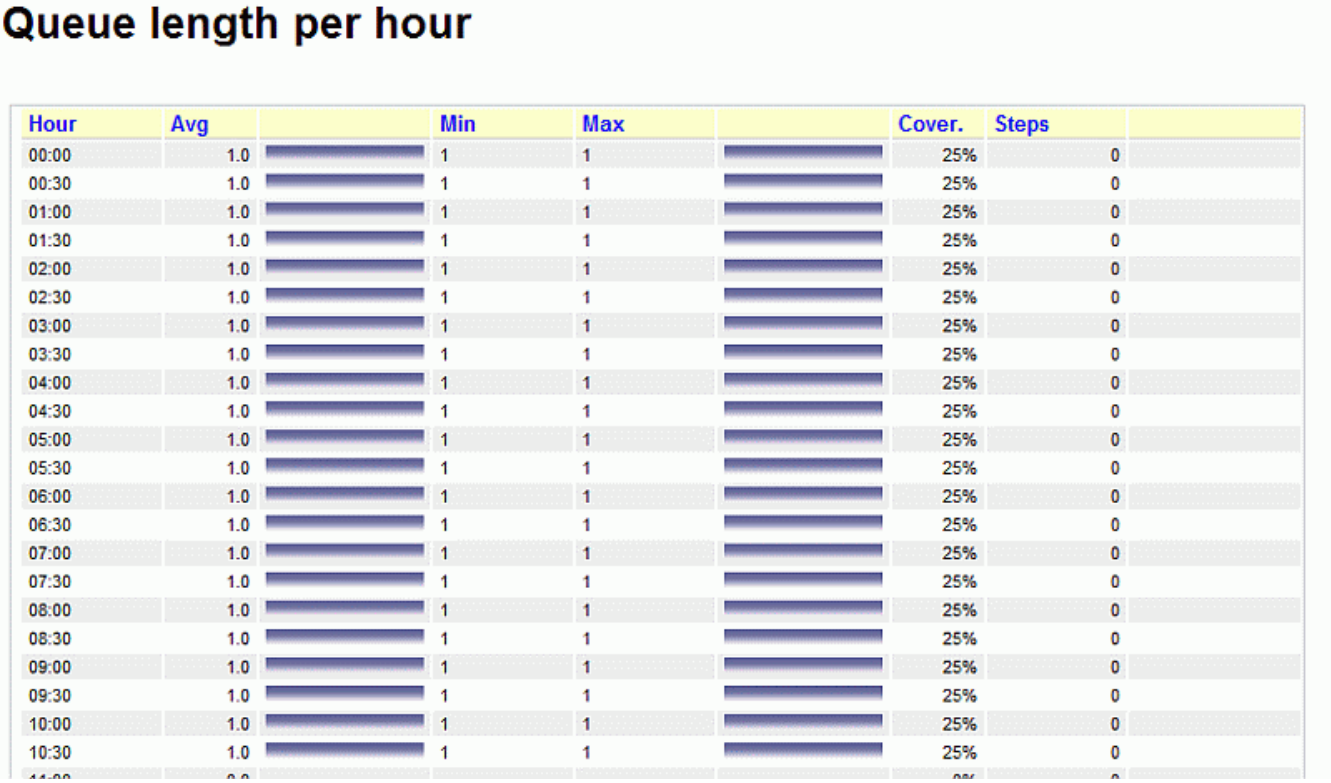
The Schedule Adherence report shows the number of distinct agents that were detected during the given period. This makes it possible to detect the number of different people that had been working on a given moment.

Enabling the *Hide empty hour slots* toggle in the Report settings panel, only time intervals that contain at least one answered call are shown. The report spans from the time of the first answered call to the time of the last answered call.

For example, if the first answered call occurs at 08:00AM and the last at 06:00PM, only the intervals from 08:00AM to 06:00PM (08:00-18:00) will appear in the report.

Available since	1.6.0
Default page	Call distribution, by hour
Shortcut code	DH05
API block name	CallDistrDO.StaffPerHr
Parameters	-
See also	

DH06 - Queue length per hour



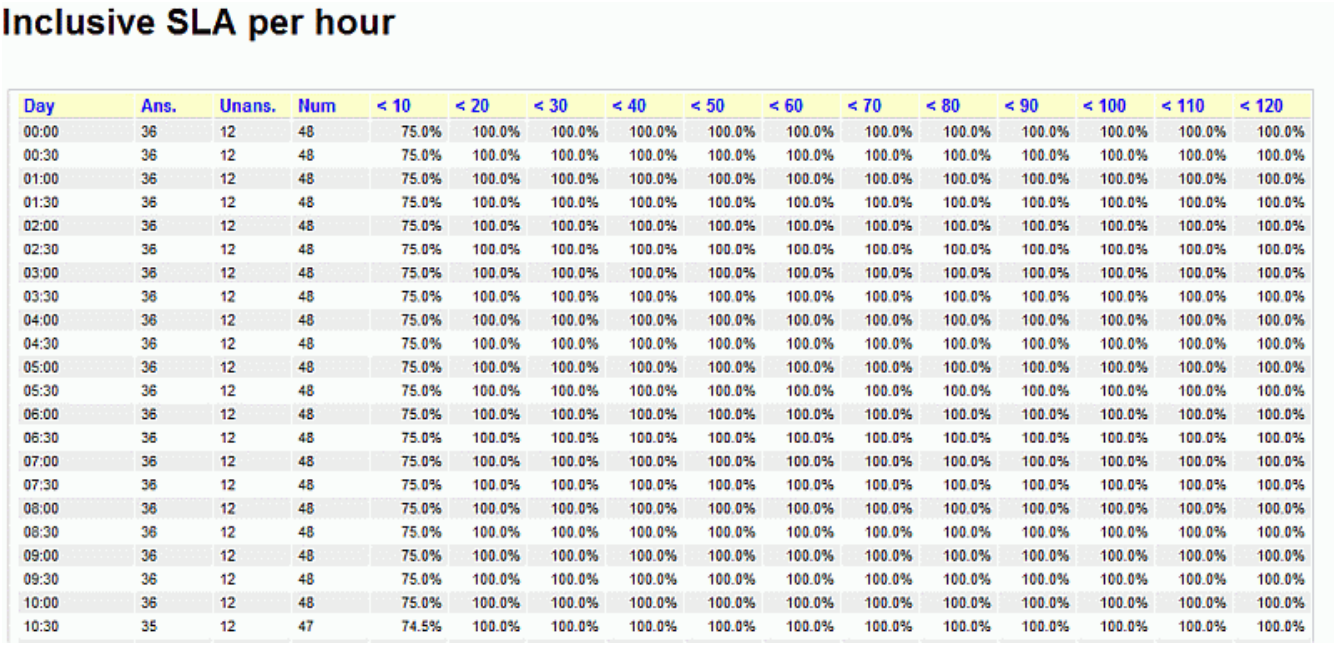
The Queue Length report shows the average length of the queue for each period, giving minimums and maximums. The Steps computation shows how fast each queue progresses during the period, expresses in steps per hour. Note: giving a meaning to the Queue Length reports may be hard in the case of composite queues.

Enabling the *Hide empty hour slots* toggle in the Report settings panel, only time intervals that contain at least one answered call are shown. The report spans from the time of the first answered call to the time of the last answered call.

For example, if the first answered call occurs at 08:00AM and the last at 06:00PM, only the intervals from 08:00AM to 06:00PM (08:00-18:00) will appear in the report.

Available since	1.6.0
Default page	Call distribution, by hour
Shortcut code	DH06
API block name	CallDistrDO.QPosPerHr
Parameters	-
See also	

DH07 - Inclusive SLA per hour



Inclusive SLA per hour

Day	Ans.	Unans.	Num	< 10	< 20	< 30	< 40	< 50	< 60	< 70	< 80	< 90	< 100	< 110	< 120
00:00	36	12	48	75.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
00:30	36	12	48	75.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
01:00	36	12	48	75.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
01:30	36	12	48	75.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
02:00	36	12	48	75.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
02:30	36	12	48	75.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
03:00	36	12	48	75.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
03:30	36	12	48	75.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
04:00	36	12	48	75.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
04:30	36	12	48	75.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
05:00	36	12	48	75.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
05:30	36	12	48	75.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
06:00	36	12	48	75.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
06:30	36	12	48	75.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
07:00	36	12	48	75.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
07:30	36	12	48	75.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
08:00	36	12	48	75.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
08:30	36	12	48	75.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
09:00	36	12	48	75.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
09:30	36	12	48	75.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
10:00	36	12	48	75.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
10:30	35	12	47	74.5%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%

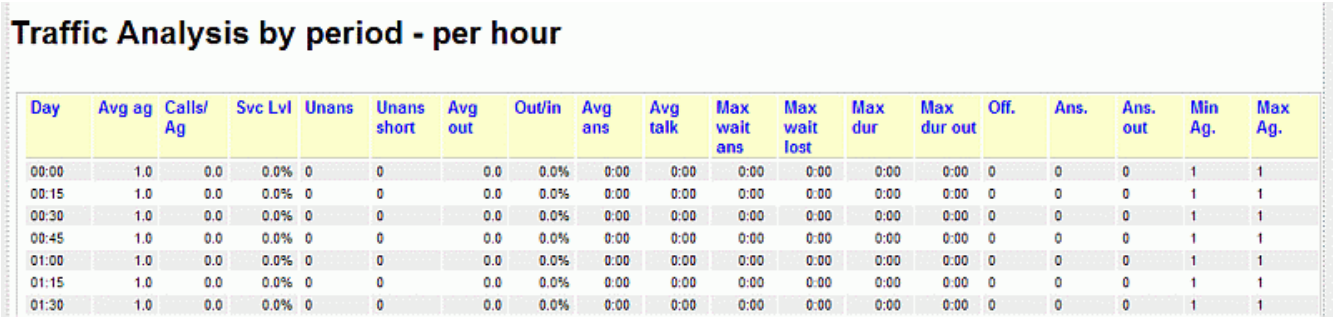
The inclusive SLA corresponds to the Service Level Agreement metrics shown on Service level agreement, grouped each hour.

Enabling the *Hide empty hour slots* toggle in the Report settings panel, only time intervals that contain at least one answered call are shown. The report spans from the time of the first answered call to the time of the last answered call.

For example, if the first answered call occurs at 08:00AM and the last at 06:00PM, only the intervals from 08:00AM to 06:00PM (08:00-18:00) will appear in the report.

Available since	1.6.0
Default page	Call distribution, by hour
Shortcut code	DH07
API block name	CallDistrDO.InclSlaPerHr
Parameters	-
See also	

DH08 - Traffic Analysis by period - per hour



Traffic Analysis by period - per hour

Day	Avg ag	Calls/ Ag	Svc Lvl	Unans	Unans short	Avg out	Out/in	Avg ans	Avg talk	Max wait ans	Max wait lost	Max dur	Max dur out	Off.	Ans.	Ans. out	Min Ag.	Max Ag.
00:00	1.0	0.0	0.0%	0	0	0.0	0.0%	0:00	0:00	0:00	0:00	0:00	0:00	0	0	0	1	1
00:15	1.0	0.0	0.0%	0	0	0.0	0.0%	0:00	0:00	0:00	0:00	0:00	0:00	0	0	0	1	1
00:30	1.0	0.0	0.0%	0	0	0.0	0.0%	0:00	0:00	0:00	0:00	0:00	0:00	0	0	0	1	1
00:45	1.0	0.0	0.0%	0	0	0.0	0.0%	0:00	0:00	0:00	0:00	0:00	0:00	0	0	0	1	1
01:00	1.0	0.0	0.0%	0	0	0.0	0.0%	0:00	0:00	0:00	0:00	0:00	0:00	0	0	0	1	1
01:15	1.0	0.0	0.0%	0	0	0.0	0.0%	0:00	0:00	0:00	0:00	0:00	0:00	0	0	0	1	1
01:30	1.0	0.0	0.0%	0	0	0.0	0.0%	0:00	0:00	0:00	0:00	0:00	0:00	0	0	0	1	1

Distribution of calls and agent availability per hour (or interval you specified). The whole 24h are mapped out for ease of comparison.

For a complete description of parameters, see DD08 - Traffic Analysis by period - per day

Available since	1.6.0.4
Default page	Call distribution, by hour
Shortcut code	DH08
API block name	CallDistrDO.TrafficAnPerHr
Parameters	-
See also	DD08, DW08

DH09 - Skill sets per hour

This reports shows the unique sets of skills during a specified time period.

Enabling the *Hide empty hour slots* toggle in the Report settings panel, only time intervals that contain at least one answered call are shown. The report spans from the time of the first answered call to the time of the last answered call.

For example, if the first answered call occurs at 08:00AM and the last at 06:00PM, only the intervals from 08:00AM to 06:00PM (08:00-18:00) will appear in the report.

Skills per Hour

Day	Ans.	Unans.	Num	Within SLA: 10s.	Product1 english internet technical	Product1 english invoice land	Product1 english invoice mobile	Product1 english land others	Product1 english mobile others	Product1 english mobile technical	Product1 english movie technica
01:00	0	0	0	-	0	0	0	0	0	0	
05:00	7,921	457	8,378	51.8%	13	1	10	5	17	15	
09:00	42,610	2,213	44,823	52.1%	39	10	103	42	160	78	
13:00	41,431	1,880	43,311	53.0%	40	17	83	49	129	64	
17:00	18,609	2,603	21,212	41.1%	27	10	39	17	67	38	
21:00	6	14	20	15.0%	0	0	0	0	0	0	

Export as...

For a complete description of parameters and behavior, see DD09 - Skill sets per day

Available since	16.10.11
Default page	Call distribution, by day
Shortcut code	DH09
API block name	CallDistrDO.SkillsPerHr
Parameters	-
See also	DD09, DW09

DH10 - Detail Summary

This reports shows the number of taken and lost call by hour.

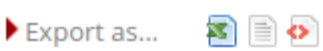
Enabling the *Hide empty hour slots* toggle in the Report settings panel, only time intervals that contain at least one answered call are shown. The report spans from the time of the first answered call to the time of the last answered call.

For example, if the first answered call occurs at 08:00AM and the last at 06:00PM, only the intervals from 08:00AM to 06:00PM (08:00-18:00) will appear in the report.

Detail Summary

Detail Summary

Hour	Taken Calls	Lost Calls	N. of Calls
0:00	0	0	0
1:00	0	0	0
2:00	0	0	0
3:00	0	0	0
4:00	0	0	0
5:00	0	0	0
6:00	0	0	0
7:00	0	0	0
8:00	0	0	0
9:00	0	0	0
10:00	0	0	0
11:00	7	0	7
12:00	2	0	2
13:00	2	0	2
14:00	10	0	10
15:00	17	3	20
16:00	6	1	7
17:00	1	0	1
18:00	1	0	1
19:00	0	0	0
20:00	0	0	0
21:00	0	0	0
22:00	0	0	0
23:00	0	0	0



For a complete description of parameters and behavior, see DD10 - Detail Summary by day

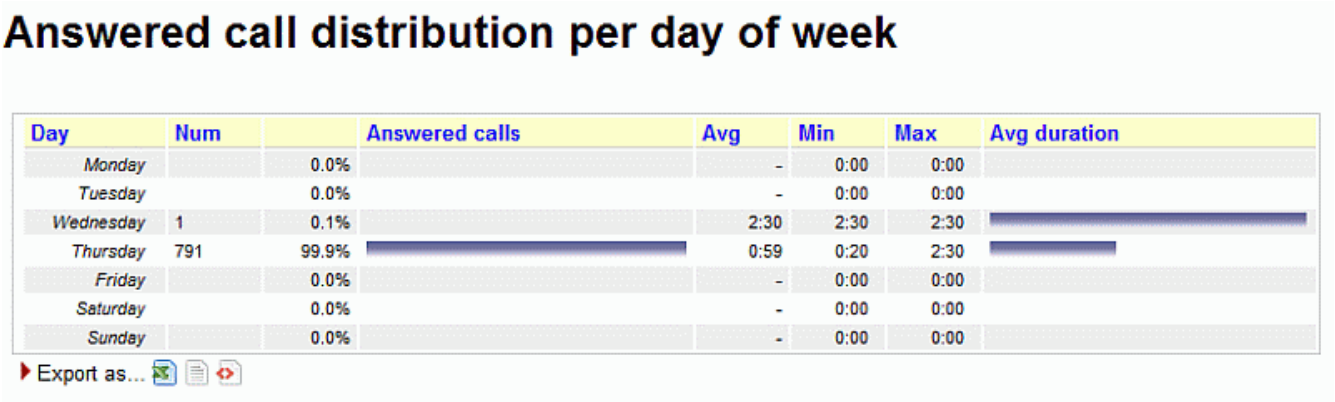
Available since	17.06.16
Default page	Detail Summary, by hour
Shortcut code	DH10

API block name	CallDistrDO.DetailSummary
Parameters	-
See also	DD10, DW10

Historical reports - Call distribution by day of week

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

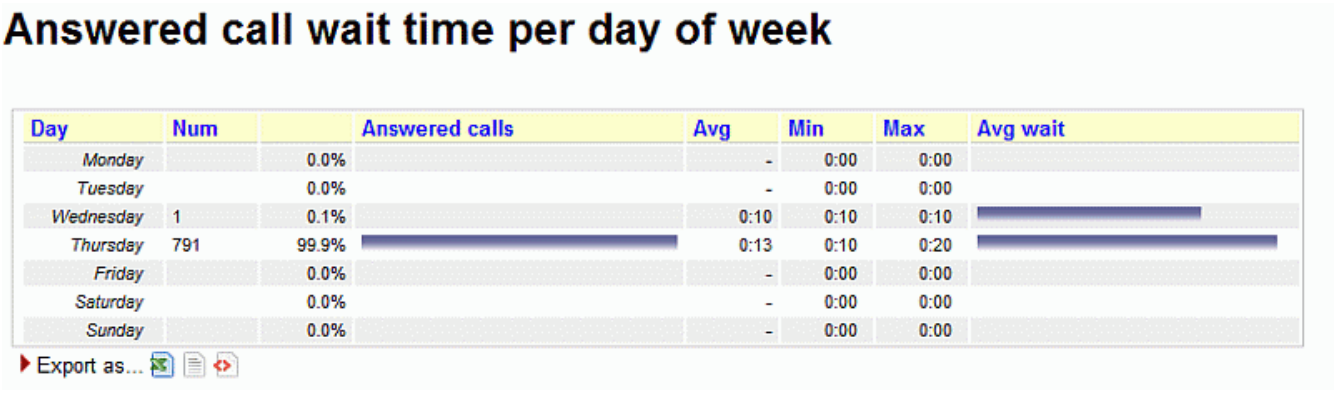
DW01 - Answered call distribution per day of week



The total numbers of call lengths for answered calls are plotted for each day of the week.

Available since	1.6.0
Default page	Call distribution, by day of week
Shortcut code	DW01
API block name	CallDistrDO.AnsDistrPerDOW
Parameters	-
See also	

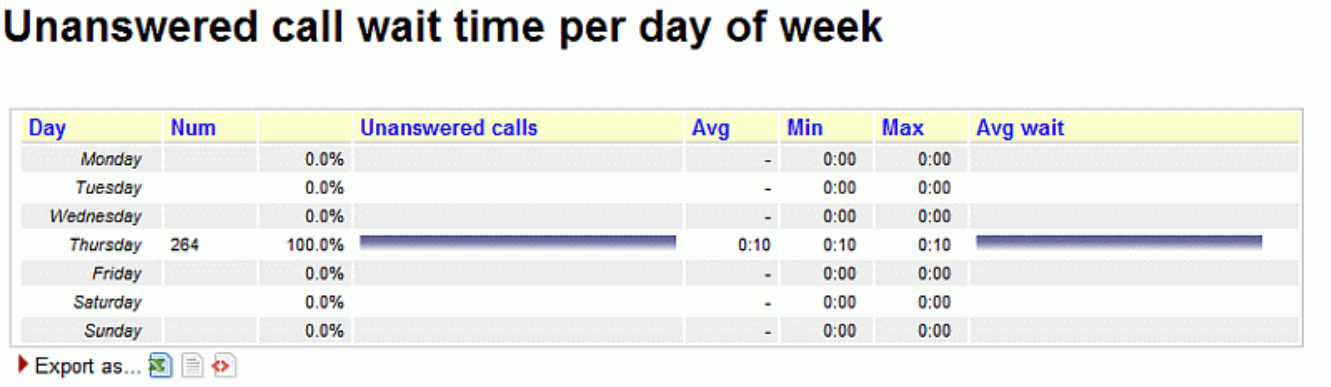
DW02 - Answered call wait time per day of week



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

Available since	1.6.0
Default page	Call distribution, by day of week
Shortcut code	DW02
API block name	CallDistrDO.AnsWaitPerDOW
Parameters	-
See also	

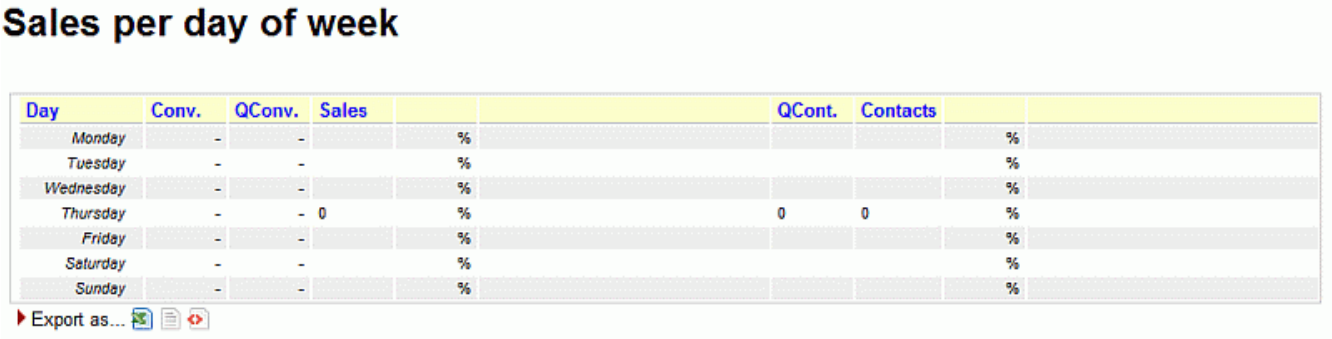
DW03 - Unanswered call wait time per day of week



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

Available since	1.6.0
Default page	Call distribution, by day of week
Shortcut code	DW03
API block name	CallDistrDO.UnansWaitPerDOW
Parameters	-
See also	

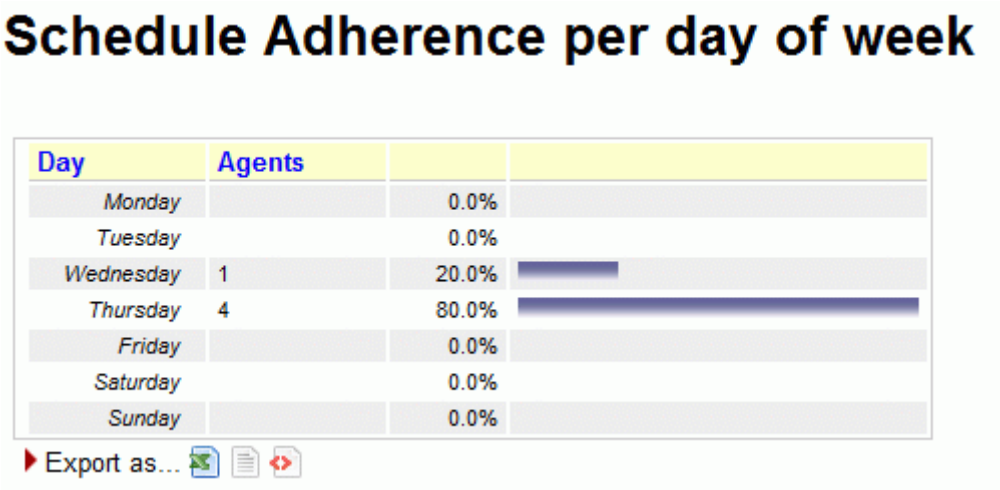
DW04 - Sales per day of week



The total numbers sales and contacts, are plotted for each day of the week.

Available since	1.6.0
Default page	Call distribution, by day of week
Shortcut code	DW04
API block name	CallDistrDO.SalesPerDOW
Parameters	-
See also	

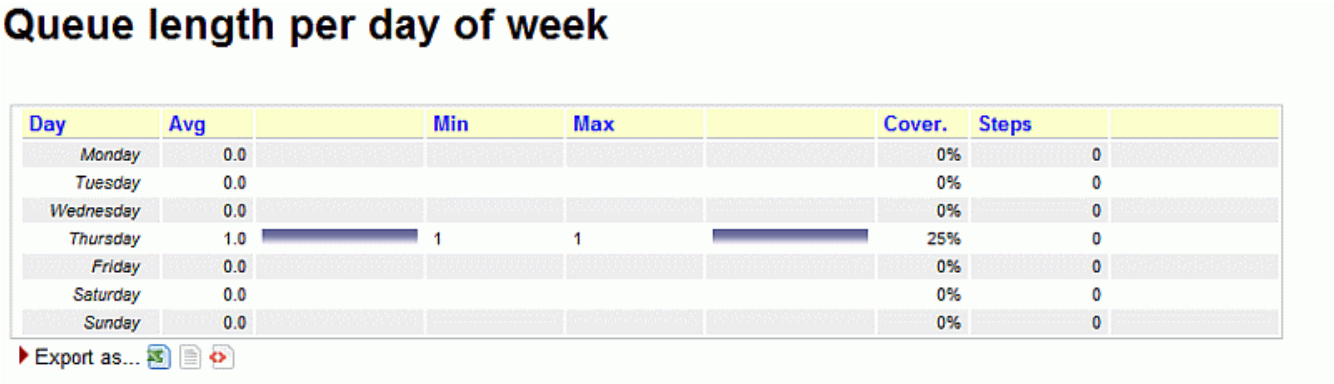
DW05 - Schedule Adherence per day of week



The Schedule Adherence report shows the number of distinct agents that were detected during the given period. This makes it possible to detect the number of different people that had been working on a given moment.

Available since	1.6.0
Default page	Call distribution, by day of week
Shortcut code	DW05
API block name	CallDistrDO.StaffPerDOW
Parameters	-
See also	

DW06 - Queue length per day of week



Queue length per day of week

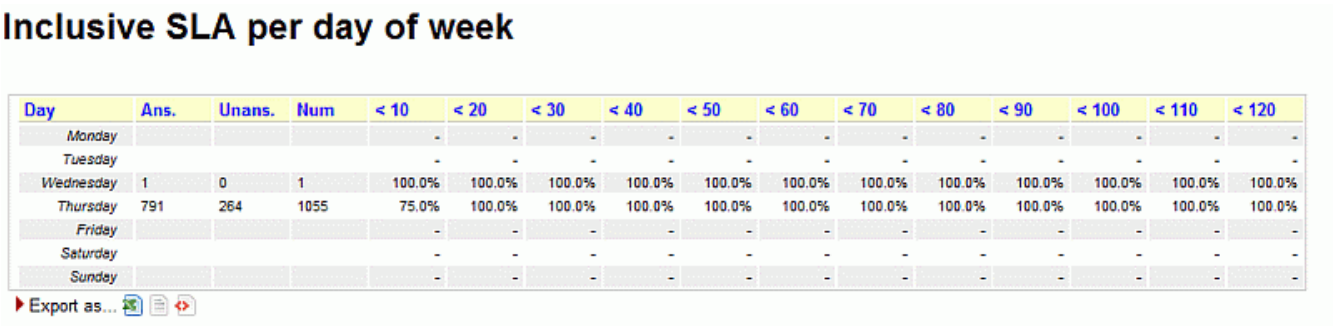
Day	Avg		Min	Max		Cover.	Steps	
Monday	0.0					0%	0	
Tuesday	0.0					0%	0	
Wednesday	0.0					0%	0	
Thursday	1.0		1	1		25%	0	
Friday	0.0					0%	0	
Saturday	0.0					0%	0	
Sunday	0.0					0%	0	

Export as...

The Queue Length report shows the average length of the queue for each period, giving minimums and maximums. The Steps computation shows how fast each queue progresses during the period, expresses in steps per hour. Note: giving a meaning to the Queue Length reports may be hard in the case of composite queues.

Available since	1.6.0
Default page	Call distribution, by day of week
Shortcut code	DW06
API block name	CallDistrDO.QPosPerDOW
Parameters	-
See also	

DW07 - Inclusive SLA per day of week



Inclusive SLA per day of week

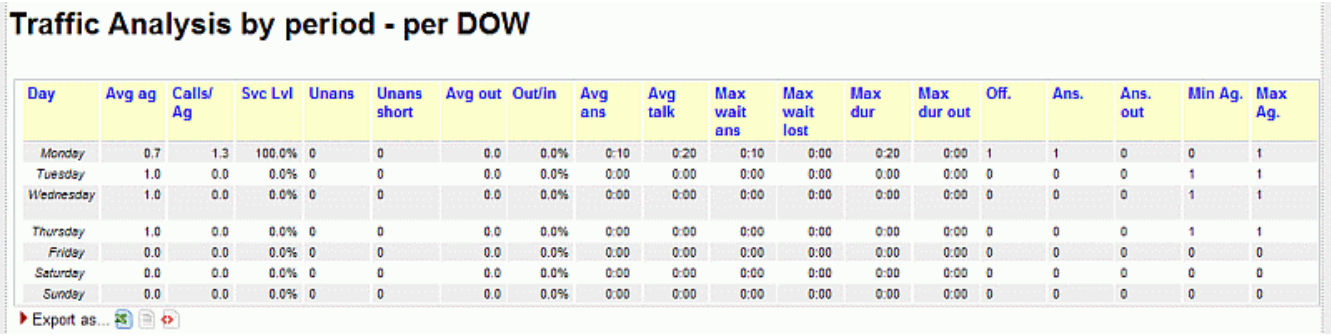
Day	Ans.	Unans.	Num	< 10	< 20	< 30	< 40	< 50	< 60	< 70	< 80	< 90	< 100	< 110	< 120
Monday				-	-	-	-	-	-	-	-	-	-	-	-
Tuesday				-	-	-	-	-	-	-	-	-	-	-	-
Wednesday	1	0	1	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
Thursday	791	264	1055	75.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
Friday				-	-	-	-	-	-	-	-	-	-	-	-
Saturday				-	-	-	-	-	-	-	-	-	-	-	-
Sunday				-	-	-	-	-	-	-	-	-	-	-	-

Export as...

The inclusive SLA corresponds to the Service Level Agreement metrics shown on Service level agreement, grouped for each day of week.

Available since	1.6.0
Default page	Call distribution, by day of week
Shortcut code	DW07
API block name	CallDistrDO.InclSlaPerDOW
Parameters	-
See also	

DW08 - Traffic Analysis by period - per day of week



Traffic Analysis by period - per DOW

Day	Avg ag	Calls/ Ag	Svc Lvl	Unans	Unans short	Avg out	Out/in	Avg ans	Avg talk	Max wait ans	Max wait lost	Max dur	Max dur out	Off.	Ans.	Ans. out	Min Ag.	Max Ag.
Monday	0.7	1.3	100.0%	0	0	0.0	0.0%	0:10	0:20	0:10	0:00	0:20	0:00	1	1	0	0	1
Tuesday	1.0	0.0	0.0%	0	0	0.0	0.0%	0:00	0:00	0:00	0:00	0:00	0:00	0	0	0	1	1
Wednesday	1.0	0.0	0.0%	0	0	0.0	0.0%	0:00	0:00	0:00	0:00	0:00	0:00	0	0	0	1	1
Thursday	1.0	0.0	0.0%	0	0	0.0	0.0%	0:00	0:00	0:00	0:00	0:00	0:00	0	0	0	1	1
Friday	0.0	0.0	0.0%	0	0	0.0	0.0%	0:00	0:00	0:00	0:00	0:00	0:00	0	0	0	0	0
Saturday	0.0	0.0	0.0%	0	0	0.0	0.0%	0:00	0:00	0:00	0:00	0:00	0:00	0	0	0	0	0
Sunday	0.0	0.0	0.0%	0	0	0.0	0.0%	0:00	0:00	0:00	0:00	0:00	0:00	0	0	0	0	0

Export as...

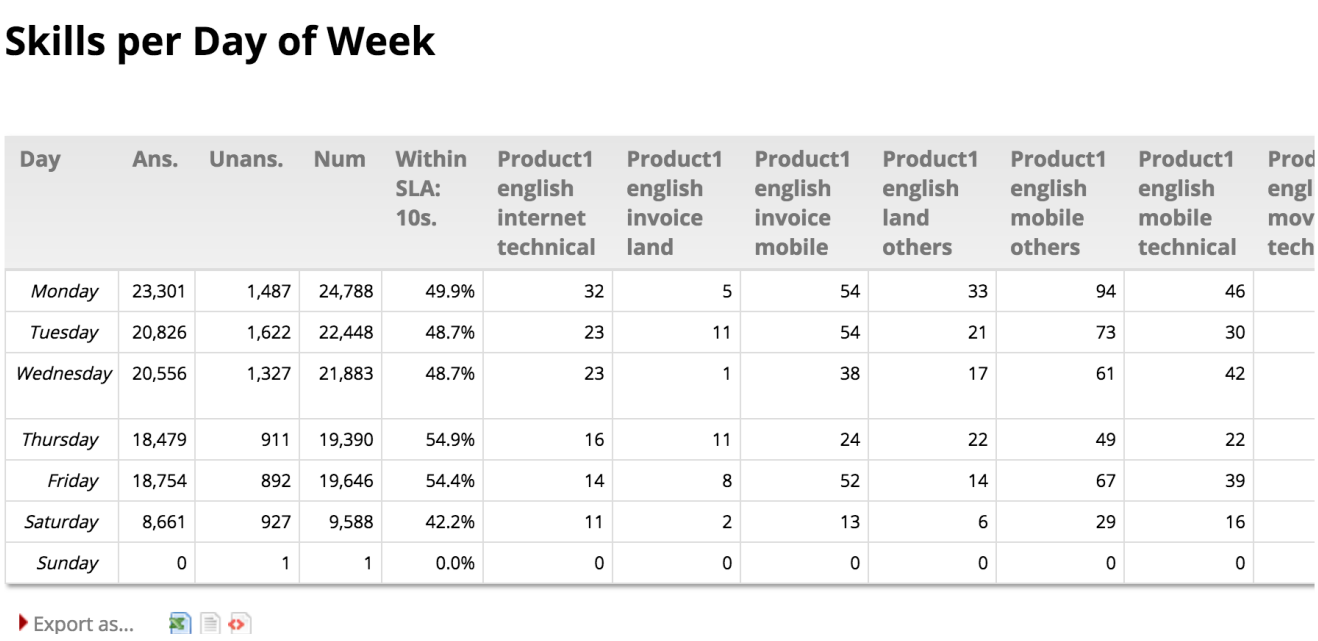
Distribution of calls and agent availability per day of week.

For a complete description of parameters, see DD08 - Traffic Analysis by period - per day

Available since	1.6.0.4
Default page	Call distribution, by day of week
Shortcut code	DW08
API block name	CallDistrDO.TrafficAnPerDOW
Parameters	-
See also	DD08, DH08

DW09 - Skill sets per day of week

This reports shows the unique sets of skills during a specified time period.



Skills per Day of Week

Day	Ans.	Unans.	Num	Within SLA: 10s.	Product1 english internet technical	Product1 english invoice land	Product1 english invoice mobile	Product1 english land others	Product1 english mobile others	Product1 english mobile technical	Prod engl mov tech
Monday	23,301	1,487	24,788	49.9%	32	5	54	33	94	46	
Tuesday	20,826	1,622	22,448	48.7%	23	11	54	21	73	30	
Wednesday	20,556	1,327	21,883	48.7%	23	1	38	17	61	42	
Thursday	18,479	911	19,390	54.9%	16	11	24	22	49	22	
Friday	18,754	892	19,646	54.4%	14	8	52	14	67	39	
Saturday	8,661	927	9,588	42.2%	11	2	13	6	29	16	
Sunday	0	1	1	0.0%	0	0	0	0	0	0	

Export as...

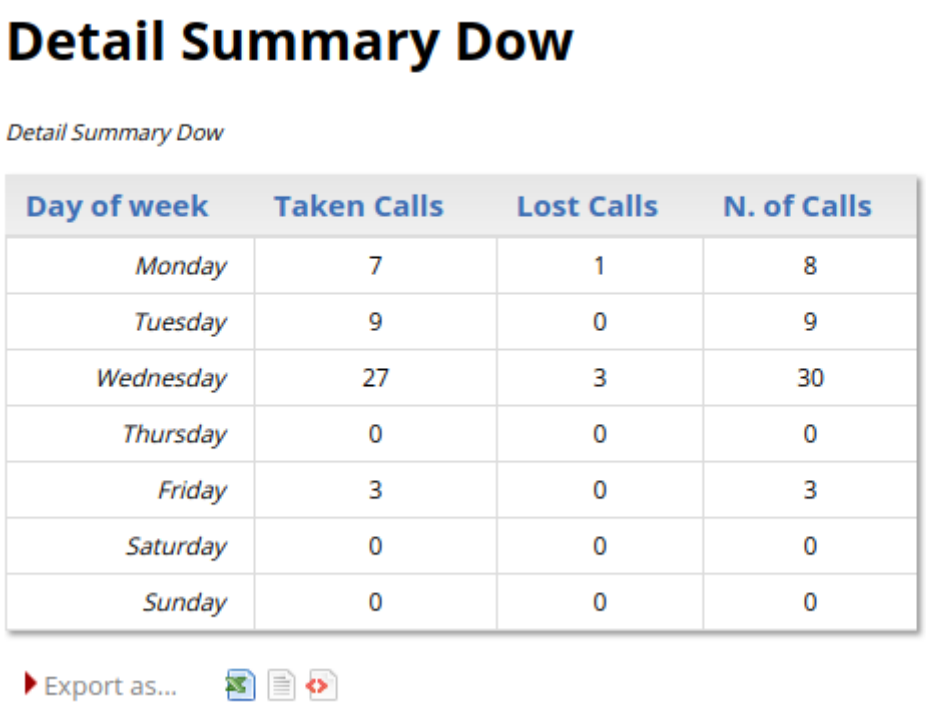
For a complete description of parameters and behavior, see DD09 - Skill sets per day

Available since	16.10.11

Default page	Call distribution, by day
Shortcut code	DW09
API block name	CallDistrDO.SkillsPerDOW
Parameters	-
See also	DD09, DH09

DW10 - Detail Summary by Day of the week

This reports shows the number of taken and lost call by day of the week.



Detail Summary Dow

Detail Summary Dow

Day of week	Taken Calls	Lost Calls	N. of Calls
Monday	7	1	8
Tuesday	9	0	9
Wednesday	27	3	30
Thursday	0	0	0
Friday	3	0	3
Saturday	0	0	0
Sunday	0	0	0

Export as...

For a complete description of parameters and behavior, see DD10 - Detail Summary by day

Available since	17.06.16
Default page	Detail Summary by day of the week
Shortcut code	DW10
API block name	CallDistrDO.DetailSummary PerDow
Parameters	-
See also	DD10, DH10

Historical reports - Agents and Sessions

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.

AG01 - Agent session detail



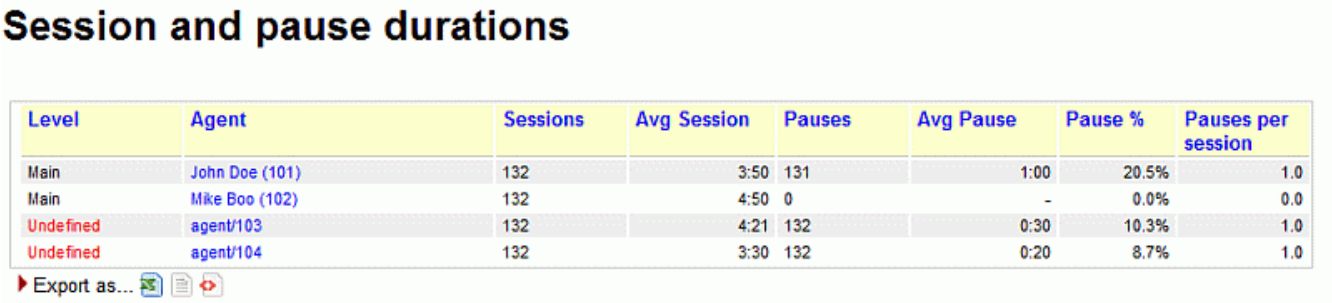
This report shows:

- The number of available agents in the considered period
- The average agent time availability
- The minumum agent time availability
- The maximum agent time availability
- The cumulated agents time availability

Available since	1.6.0
Default page	Agents and Sessions

Shortcut code	AG01
API block name	AgentsDO.ReportAgents
Parameters	-
See also	

AG02 - Session and pause durations



This report shows:

- The number of sessions for each available agent
- The number of pauses for each available agent
- The average pause time
- The percentage between the pause time and the availability time for each agent
- The number of pauses for each session, for each available agent

Available since	1.6.0
Default page	Agents and Sessions
Shortcut code	AG02
API block name	AgentsDO.SessionPauseDur
Parameters	-
See also	

AG03 - Agent availability (for all the queues they are member of)



This report shows:

- The agent available time
- The total billable and not billable pause times

In the latest release a new column has been added (Paused on Conversation), which displays occupancy values.

Available since	1.6.0
Default page	Agents and Sessions
Shortcut code	AG03
API block name	AgentsDO.AgentAvail
Parameters	-
See also	

AG04 - Answered calls (for selected queues)

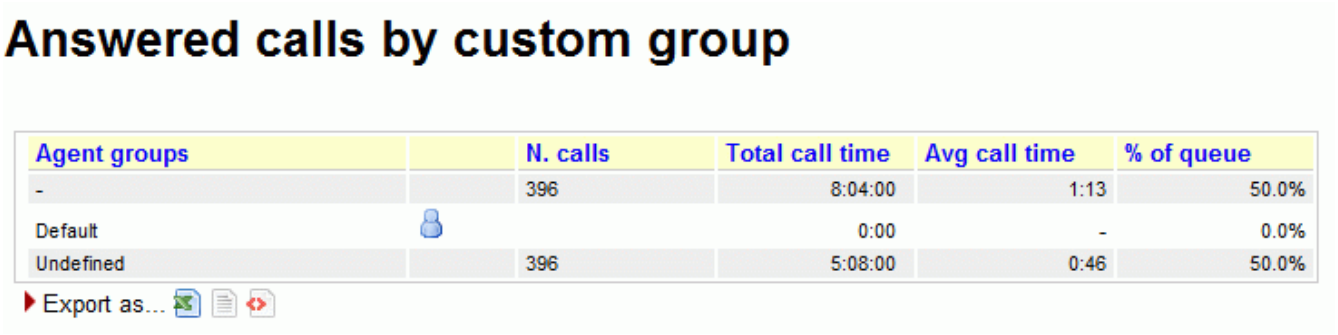


This report shows:

- The number of answered calls for each agent
- The cumulated call time for each agent
- The average call time for each agent
- The average wait time for each call taken
- The percentage of taken calls, related to the total queue calls, for each agent

Available since	1.6.0
Default page	Agents and Sessions
Shortcut code	AG04
API block name	AgentsDO.AnsCallsQueues
Parameters	-
See also	

AG05 - Answered calls by custom group



This report shows:

- The number of answered calls for each defined agent group
- The cumulated call time for each defined agent group
- The average call time for each defined agent group
- The percentage of taken calls, related to the total queue calls, for each defined agent group

The report displays separately the following cases:

- Agents that are known to QueueMetrics but have no custom group, are counted under the group "-"
- Agents that are not known to QueueMetrics but found in the analysis are counted under "Undefined".

Available since	1.6.0
Default page	Agents and Sessions
Shortcut code	AG05
API block name	AgentsDO.AnsCallsCG
Parameters	-
See also	

AG06 - Answered calls by location



This report shows:

- The number of answered calls for each defined location
- The cumulated call time for each defined location
- The average call time for each defined location

- The average wait time for each defined location
- The percentage of taken calls, related to the total queue calls, for each defined location

Available since	1.6.0
Default page	Agents and Sessions
Shortcut code	AG06
API block name	AgentsDO.AnsCallsLocation
Parameters	-
See also	

AG07 - Answered calls by service group

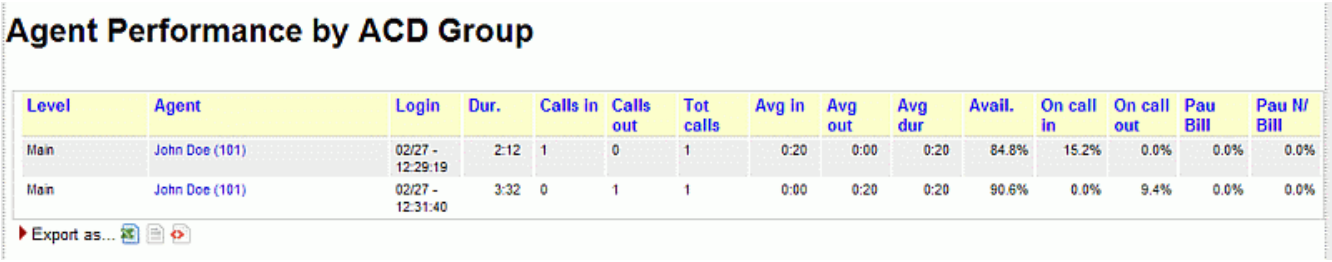


This report shows:

- The number of answered calls for each defined service group
- The cumulated call time for each defined service group
- The average call time for each defined service group
- The percentage of taken calls, related to the total queue calls, for each defined service group

Available since	1.6.0
Default page	Agents and Sessions
Shortcut code	AG07
API block name	AgentsDO.AnsCallsSG
Parameters	-
See also	

AG08 - Agent Performance by ACD Group



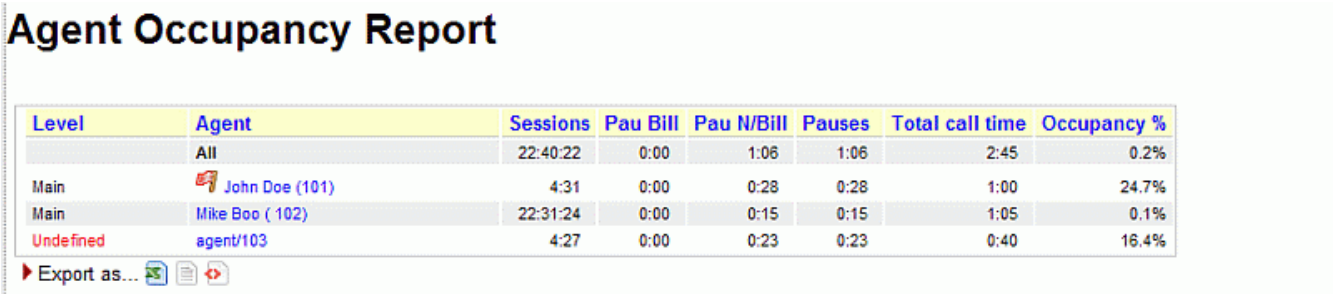
One entry is presented for each agent session. What makes this report different from most is that inbound and outbound activity for the agent is aggregated.

Items are computed as:

- Level: the agent level this agent belongs to
- Agent: agent name (if present) or Asterisk internal code if unknown. By clicking on it, the session details open.
- Login: Session login time
- Duration: Session duration
- N.calls in: Number of calls taken for this queue(s)
- N calls out: Number of calls outbound made for these campaign(s)
- Tot calls: calls in + calls out
- Average duration IN
- Average duration OUT
- Average duration for all (weighted)
- Available: precentage of idle time
- On call IN: percentage of time on call inbound
- On call OUT: percentage of time on call outbound
- Pause Billable: percentage of time on a pause code marked as billable
- Pause Unbillable: percentage of time on a pause code marked as unbillable, or not specified, or unknown. (a.k.a. Clerical time)

Available since	1.6.0.4
Default page	Agents and Sessions
Shortcut code	AG08
API block name	AgentsDO.PerformanceAcdGroups
Parameters	-
See also	

AG09 - Agent Occupancy Report



The report details the occupancy rate for each and all agents in the current analysis.

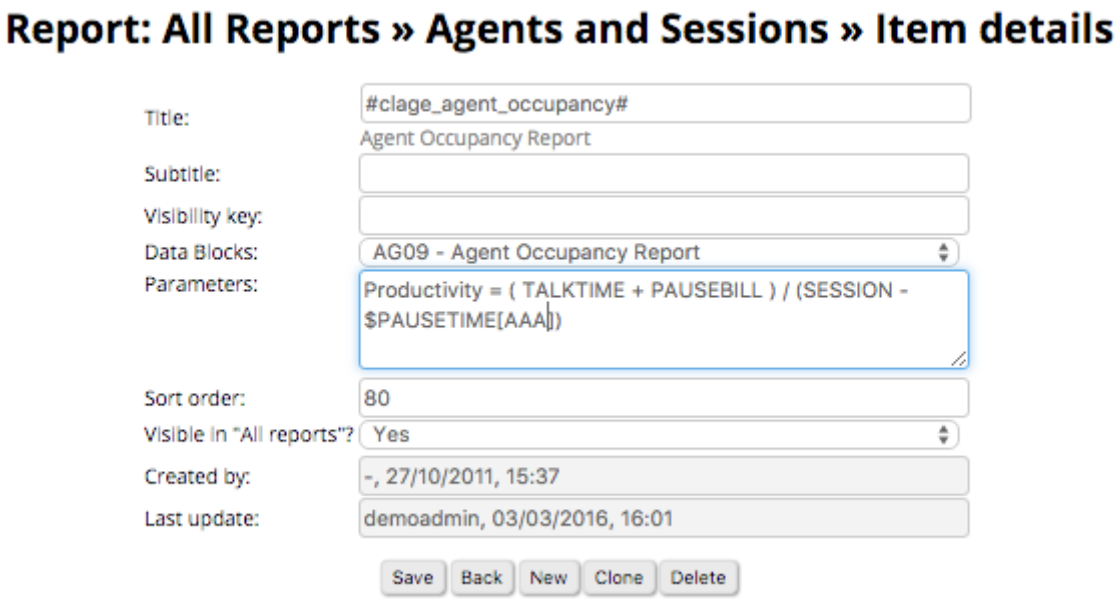
- The Agent column contains the decoded name, level and current group of the agent (if defined, expressed as an icon like elsewhere in QM). By clicking on the agent name, it is possible to open a popup with the session details for that agent.
- Total session time is the sum of the duration of all sessions in the current analysis, from start to end.
- Pause Billable is the total time on Billable pauses, for all sessions considered.
- Pause Non Billable is the total time on Non-Billable pauses, for all sessions considered.
- Total pause time is the sum of all pauses for all sessions considered.
- Total talk time is the sum of all time that the specified agent spent in conversation during the sessions considered.
- Occupancy is computed as: Talk time / (Total session - Total Pause) and expressed as a percentage, as per the numeric examples show in the table above.

In the latest release a new column has been added (Paused Conversation) which is subtracted from the total pauses time; this way Occupancy will never be able to be over 100%.

The first line is labeled "All agents" and is computed according to the following rules:

- Total session time, Pause billable, Pause non billable, Pause wrap, Total pause, Total talk time are computed as sums of the rest of data in each column
- Occupancy is computed as: Sum talk time / (Sum Total session - Sum Total Pause)

The Occupancy report allows you to define custom calculations by entering one or more custom formulas in the Parameters field when editing your report.



For example, if you add the formula:

```
Productivity = ( TALKTIME + PAUSEBILL ) / (SESSION - ($PAUSETIME[AAA] +
$PAUSETIME[BB]))
```

A new column "Productivity" will be computed based on the formula supplied. At the moment, the formula will always be displayed as a percentage.

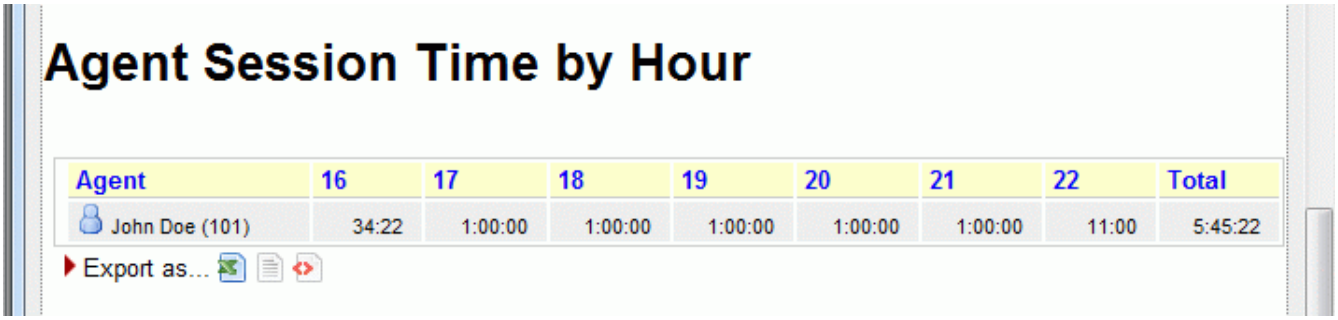
New columns can be programmed as explained in User-Programmable reports



A version of this data block meant to be completely customized is available as AG18.

Available since	1.6.2.4 - Formulas available since 15.10.5
Default page	Agents and Sessions
Shortcut code	AG09
API block name	AgentsDO.AgentOccupancy
Parameters	Agent parameters - See Variables bound for Agent blocks
See also	AG18

AG10 - Agent Session Time by Hour

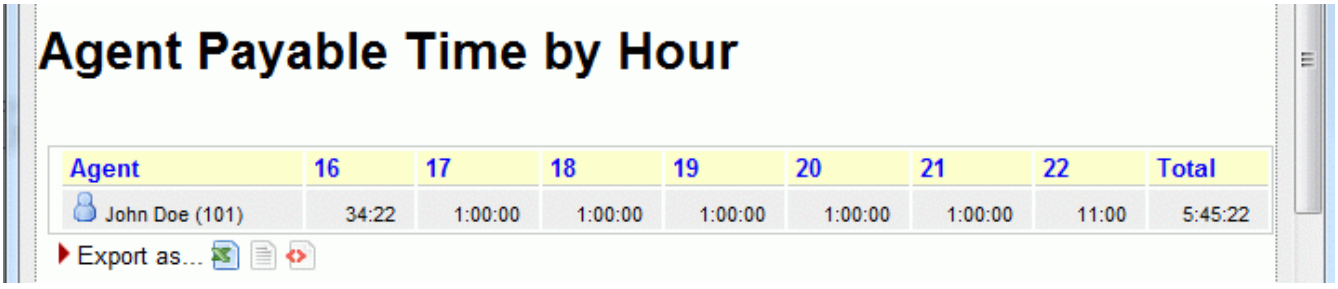


The report details the total presence time of each agent over the 24h for which you are running the current analysis.

- The Agent column contains the agent's decoded name, level and current group (if defined, expressed as an icon like elsewhere in QM).
- Total time is the sum of the duration of all sessions included in the current analysis, from start to end.
- One or more columns are present for each hour's timeframe where total session information is present

Available since	1.7.2
Default page	Agents and Sessions
Shortcut code	AG10
API block name	AgentsDO.AgentBillableTimeByHour
Parameters	-
See also	

AG11 - Agent Payable Time by Hour

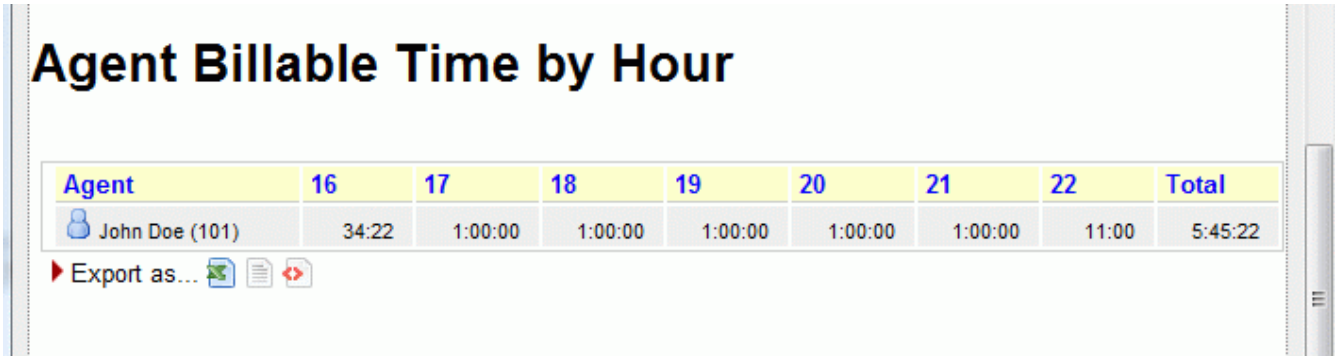


The report details the payable presence time of each agent over the 24h for which you are running the current analysis.

- The Agent column contains the agent's decoded name, level and current group (if defined, expressed as an icon like elsewhere in QM).
- Total time is the sum of the duration of payable session time within the current analysis, from start to end.
- One or more columns are present for each hour's timeframe where payable session information is present

Available since	1.7.2
Default page	Agents and Sessions
Shortcut code	AG11
API block name	AgentsDO.AgentPayableTimeByHour
Parameters	-
See also	

AG12 - Agent Billable Time by Hour



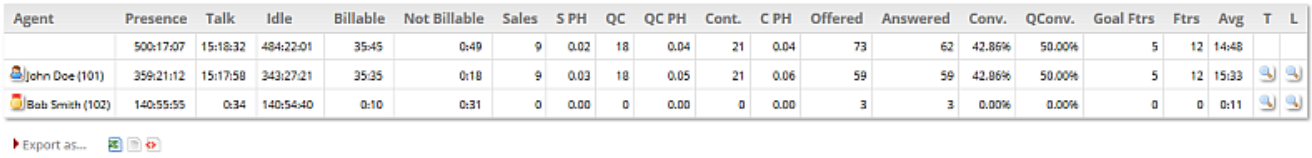
The report details the billable presence time of each agent over the 24h for which you are running the current analysis.

- The Agent column contains the agent's decoded name, level and current group (if defined, expressed as an icon like elsewhere in QM).
- Total time is the sum of the duration of billable session time within the current analysis, from start to end.
- One or more columns are present for each hour's timeframe where billable session information is present

Available since	1.7.2
Default page	Agents and Sessions
Shortcut code	AG12
API block name	AgentsDO.AgentBillableTimeByHour
Parameters	-
See also	

AG13 - Agent Session View

Agent Sessions View



Agent	Presence	Talk	Idle	Billable	Not Billable	Sales	S PH	QC	QC PH	Cont.	C PH	Offered	Answered	Conv.	QConv.	Goal Ftrs	Ftrs	Avg	T	L
	500:17:07	15:18:32	484:22:01	35:45	0:49	9	0.02	18	0.04	21	0.04	73	62	42.86%	50.00%	5	12	14:48		
John Doe (101)	359:21:12	15:17:58	343:27:21	35:35	0:18	9	0.03	18	0.05	21	0.06	59	59	42.86%	50.00%	5	12	15:33		
Bob Smith (102)	140:55:55	0:34	140:54:40	0:10	0:31	0	0.00	0	0.00	0	0.00	3	3	0.00%	0.00%	0	0	0:11		

Export as...

This report shows a set of calls aggregated by agents. It reports session information and shows, for each agent, the main figures for aggregated calls related to that agent. The first row reports the overall aggregated values.

Each row in the report shows:

- The agent name (the first line contains all agents information)
- Presence, calculated as sum of all agent sessions (so including pauses time)
- Talk time
- Idle time, calculated by subtracting pauses and talk time from the presence figure
- Idle time (No Wait), calculated by subtracting pauses talk time and Wait time from the presence figure
- Billable time
- Not billable time
- Total Wait Time
- Sales: the number of taken calls flagged "sales"
- S PH (Sales per hour): the number of sales per hour ratio
- QC: the number of taken calls flagged as qualified contacts
- QC PH: Qualified contacts per hour
- Cont: number of taken calls flagged as "contacts"
- CPH: Contacts per hour
- Offered: Overall taken + lost calls
- Answered: taken calls
- Conv: convertion ratio
- QConv: qualified convertion ratio
- Goal Ftrs: number of goal flagged feature codes found in the call dataset
- Ftrs: number of feature codes found in the call dataset
- Avg: average talk time
- T: by clicking on that icon, a popup with the taken call list (used to count the aggregated) is shown
- L: by clicking on that icon, a popup with the lost call list (used to count the aggregated) is shown



Regarding inbound calls, when looking at a single agent, the concept of "offered call" will coincide with the concept of "answered call". This is because this report

does not keep track of the various attempts for agent, but only registers the call as an answered call for the agent that actually answered the call. This will eventually result in the first row containing a number of offered calls that is higher than the number of answered calls, while on the single agents rows the offered and answered calls will be the same.



Regarding the "Total Wait Time" field, it's important to know that the wait time data refers to the sum of the wait time of all the calls that the agent answered. This does not mean that the wait time listed here is the same as the time the agent took before answering the call, but rather the time that the various calls have been waiting in the queue. This may include wait times that happened before the agent was free and thus unable to answer the call. To check the actual wait time before a particular agent answered a call, please refer to report AT03 "ACD attempts by terminal".

Available since	15.9
Default page	Agents and Sessions
Shortcut code	AG13
API block name	AgentsDO.AgentSessionsView
Parameters	-
See also	

AG14 - Queue Session View

Queue Sessions View

Queue	Presence	Talk	Idle	Billable	Not Billable	Sales	S PH	QC	QC PH	Cont.	C PH	Offered	Answered	Conv.	QConv.	Goal Ftrs	Ftrs	Avg	T	L
	500:17:07	15:18:32	484:22:01	35:45	0:49	9	0.02	18	0.04	21	0.04	73	62	42.86%	50.00%	5	12	14:48		
A300	498:30:21	15:15:16	482:38:39	35:37	0:48	9	0.02	18	0.04	21	0.04	68	59	42.86%	50.00%	5	10	15:30		
A400	1:46:45	3:16	1:43:21	0:07	0:00	0	0.00	0	0.00	0	0.00	5	3	0.00%	0.00%	0	2	1:05		

Export as...

This report shows a set of calls aggregated by queue. It reports queue information and shows, for each queue, the main figures for aggregated calls related to that queue. The first row reports the overall aggregated values.

Each row in the report shows:

- The queue name (the first line contains all queues information)
- Presence, calculated as sum of all agent sessions in the queue
- Talk time
- Idle time, calculated by subtracting pauses and talk time from the presence figure
- Billable time
- Not billable time

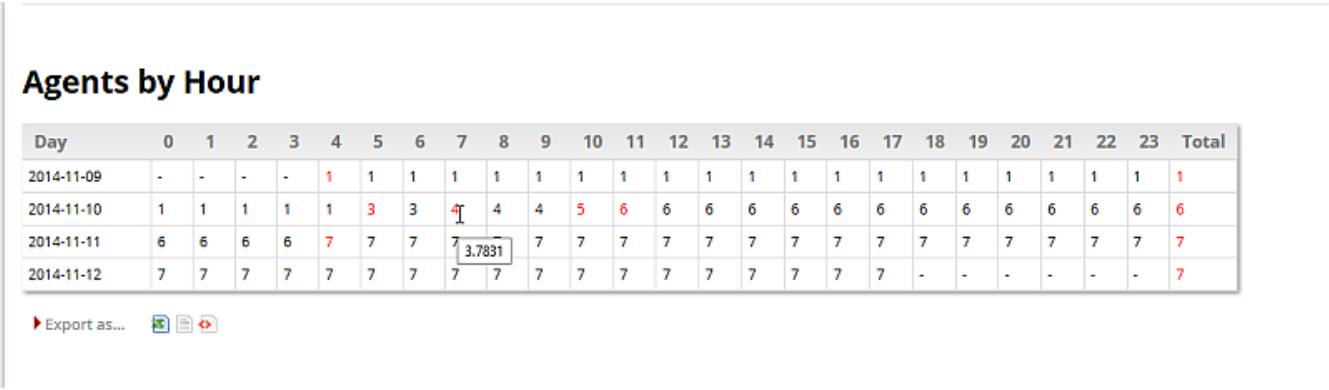
- Sales: the number of taken calls flagged "sales"
- S PH (Sales per hour): the number of sales per hour ratio
- QC: the number of taken calls flagged as qualified contacts
- QC PH: Qualified contacts per hour
- Cont: number of taken calls flagged as "contacts"
- CPH: Contacts per hour
- Offered: Overall taken + lost calls
- Answered: taken calls
- Conv: convertion ratio
- QConv: qualified convertion ratio
- Goal Ftrs: number of goal flagged feature codes found in the call dataset
- Ftrs: number of feature codes found in the call dataset
- Avg: average talk time
- T: by clicking on that icon, a popup with the taken call list (used to count the aggregated) is shown
- L: by clicking on that icon, a popup with the lost call list (used to count the aggregated) is shown



QueueMetrics uses an approach where it does not exactly track sessions per queue. If an agent joins and leaves multiple queues during the same session, QueueMetrics does not track where each session starts and ends. Therefore, while we may get accurate details of the time an agent was available, if an agent may cherry pick queues or work on multiple queues at once, we will have to find a way to estimate when its time was spent. In order to do this QueueMetrics counts the local time an agent was available per session, then it multiplies this value by the percentage of talk time that happened on each queue.

Available since	15.9
Default page	Agents and Sessions
Shortcut code	AG14
API block name	AgentsDO.QueueSessionsView
Parameters	-
See also	

AG15 - Agents by Hour



Agents by Hour

Day	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	Total
2014-11-09	-	-	-	-	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2014-11-10	1	1	1	1	1	3	3	4	4	4	5	6	6	6	6	6	6	6	6	6	6	6	6	6	6
2014-11-11	6	6	6	6	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
2014-11-12	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	-	-	-	-	-	-	7

3.7831

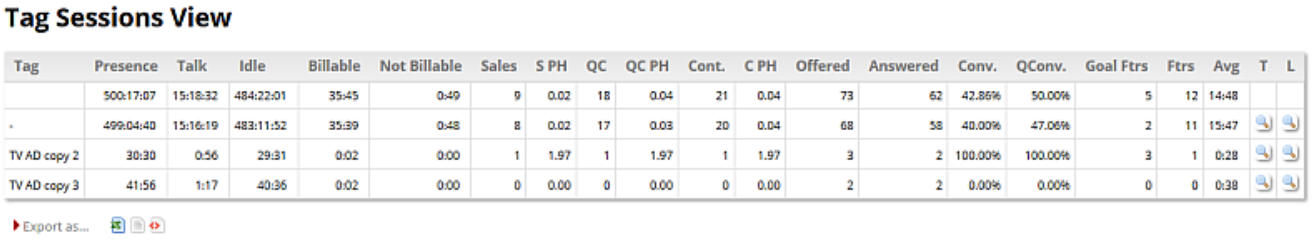
Export as...

The report details the number of agents available on the selected queues each hour for each day in the report.

- The first column (Day) reports the date who the line is referred to
- The next 24 columns report the number of agents found for at least one second in the associated hour. Numbers are reported black if the timeband has been covered completely by the agents, are reported red if at least one agent is not present for the whole hour. In this case, by flying over the red number the user can have an idea on what's the coverage percentage for the timeband. The coverage is counted as the sum of periods, in seconds, the agents cumulated in that timeband, divided by 3600 seconds.
- The last column reports the agents seen, for at least one second, during the day

Available since	14.10.5-938
Default page	Agents and Sessions
Shortcut code	AG15
API block name	AgentsDO.AgentByHour
Parameters	-
See also	

AG16 - Tag Session View



Tag Sessions View

Tag	Presence	Talk	Idle	Billable	Not Billable	Sales	S PH	QC	QC PH	Cont.	C PH	Offered	Answered	Conv.	QConv.	Goal Ftrs	Ftrs	Avg	T	L
	500:17:07	15:18:32	484:22:01	35:45	0:49	9	0.02	18	0.04	21	0.04	73	62	42.86%	50.00%	5	12	14:48		
-	499:04:40	15:16:19	483:11:52	35:39	0:48	8	0.02	17	0.03	20	0.04	68	58	40.00%	47.06%	2	11	15:47		
TV AD copy 2	30:30	0:56	29:31	0:02	0:00	1	1.97	1	1.97	1	1.97	3	2	100.00%	100.00%	3	1	0:28		
TV AD copy 3	41:56	1:17	40:36	0:02	0:00	0	0.00	0	0.00	0	0.00	2	2	0.00%	0.00%	0	0	0:38		

Export as...

This report shows a set of calls aggregated by tag. It reports tags information and shows, for each tag, the main figures for aggregated calls related to that tag. The first row reports the overall aggregated values.

Each row in the report shows:

- The tag name (the first line contains all tags information)

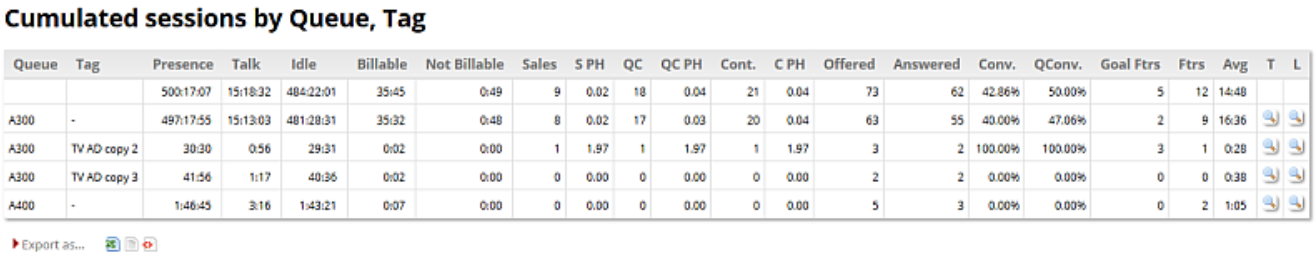
- Presence, calculated as sum of all agent sessions for agents taking calls with related tag
- Talk time
- Idle time, calculated by subtracting pauses and talk time from the presence figure
- Billable time
- Not billable time
- Sales: the number of taken calls flagged "sales"
- S PH (Sales per hour): the number of sales per hour ratio
- QC: the number of taken calls flagged as qualified contacts
- QC PH: Qualified contacts per hour
- Cont: number of taken calls flagged as "contacts"
- CPH: Contacts per hour
- Offered: Overall taken + lost calls
- Answered: taken calls
- Conv: convertion ratio
- QConv: qualified convertion ratio
- Goal Ftrs: number of goal flagged feature codes found in the call dataset
- Ftrs: number of feature codes found in the call dataset
- Avg: average talk time
- T: by clicking on that icon, a popup with the taken call list (used to count the aggregated) is shown
- L: by clicking on that icon, a popup with the lost call list (used to count the aggregated) is shown



QueueMetrics uses an approach where it does not exactly track sessions per tag. If an agent joins and leaves multiple queues during the same session, QueueMetrics does not track where each session starts and ends. Therefore, while we may get accurate details of the time an agent was available, if an agent may cherry pick queues or work on multiple queues at once, we will have to find a way to estimate when its time was spent. In order to do this QueueMetrics counts the local time an agent was available per session, then it multiplies this value by the percentage of talk time that happened on each queue.

Available since	15.09
Default page	Agents and Sessions
Shortcut code	AG16
API block name	AgentsDO.TagSessionsView
Parameters	-
See also	

AG17 - Cumulated sessions by Queue, Tag



Cumulated sessions by Queue, Tag

Queue	Tag	Presence	Talk	Idle	Billable	Not Billable	Sales	S PH	QC	QC PH	Cont.	C PH	Offered	Answered	Conv.	QConv.	Goal Ftrs	Ftrs	Avg	T	L
		500:17:07	15:18:32	484:22:01	35:45	0:49	9	0.02	18	0.04	21	0.04	73	62	42.86%	50.00%	5	12	14:48		
A300	-	497:17:55	15:13:03	481:28:31	35:32	0:48	8	0.02	17	0.03	20	0.04	63	55	40.00%	47.06%	2	9	16:36		
A300	TV AD copy 2	30:30	0:56	29:31	0:02	0:00	1	1.97	1	1.97	1	1.97	3	2	100.00%	100.00%	3	1	0:28		
A300	TV AD copy 3	41:56	1:17	40:36	0:02	0:00	0	0.00	0	0.00	0	0.00	2	2	0.00%	0.00%	0	0	0:38		
A400	-	1:46:45	3:16	1:43:21	0:07	0:00	0	0.00	0	0.00	0	0.00	5	3	0.00%	0.00%	0	2	1:05		

Export as...

This report shows a set of calls aggregated by queue and tag. It reports queue and tags information and shows, for each queue/tag pair, the main figures for aggregated calls related to that pair. The first row reports the overall aggregated values.

Each row in the report shows:

- The queue name (the first line contains all queues information)
- The tag name (the first line contains all tags information)
- Presence, calculated as sum of all agent sessions for agents taking calls with related tag
- Talk time
- Idle time, calculated by subtracting pauses and talk time from the presence figure
- Billable time
- Not billable time
- Sales: the number of taken calls flagged "sales"
- S PH (Sales per hour): the number of sales per hour ratio
- QC: the number of taken calls flagged as qualified contacts
- QC PH: Qualified contacts per hour
- Cont: number of taken calls flagged as "contacts"
- CPH: Contacts per hour
- Offered: Overall taken + lost calls
- Answered: taken calls
- Conv: convertion ratio
- QConv: qualified convertion ratio
- Goal Ftrs: number of goal flagged feature codes found in the call dataset
- Ftrs: number of feature codes found in the call dataset
- Avg: average talk time
- T: by clicking on that icon, a popup with the taken call list (used to count the aggregated) is shown
- L: by clicking on that icon, a popup with the lost call list (used to count the aggregated) is shown

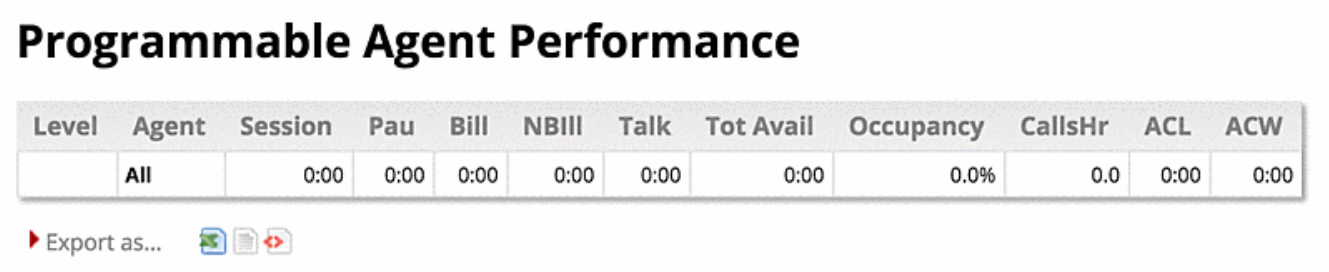


QueueMetrics uses an approach where it does not exactly track sessions per

queue/tag. If an agent joins and leaves multiple queues during the same session, QueueMetrics does not track where each session starts and ends. Therefore, while we may get accurate details of the time an agent was available, if an agent may cherry pick queues or work on multiple queues at once, we will have to find a way to estimate when its time was spent. In order to do this QueueMetrics counts the local time an agent was available per session, then it multiplies this value by the percentage of talk time that happened on each queue.

Available since	15.09
Default page	Agents and Sessions
Shortcut code	AG17
API block name	AgentsDO.SessionByQueueTagReport
Parameters	-
See also	

AG18 - Programmable Agent Performance

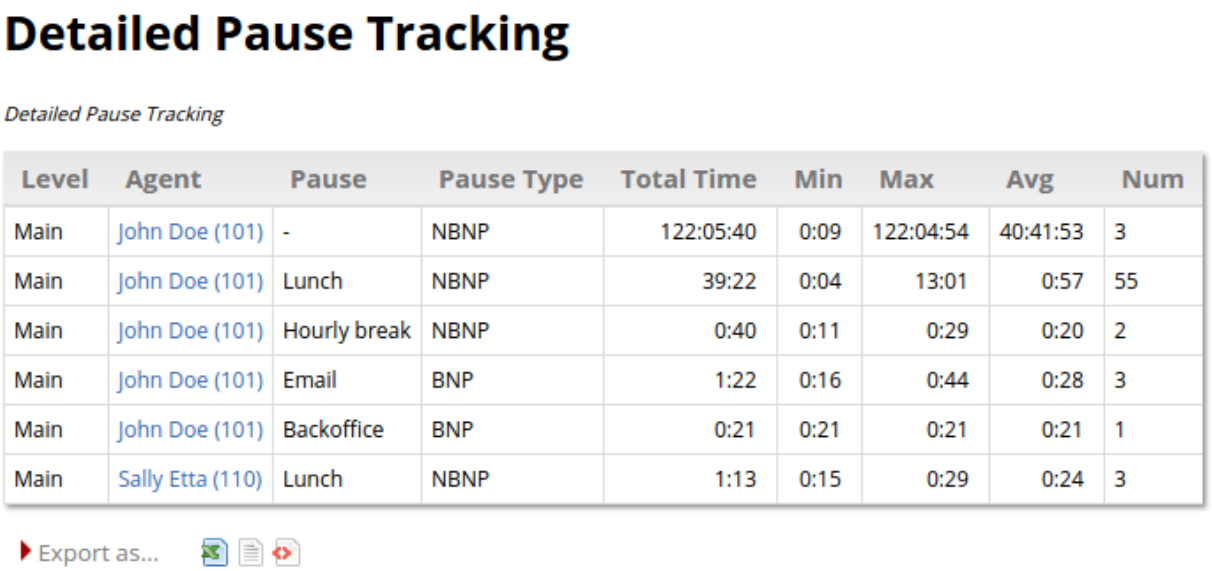


This block just includes the first two columns: Level and Agent. All other columns are to be programmed by the user as explained in User-Programmable reports .

This block is meant as a completyl configurable version of block AG09.

Available since	15.10.8
Default page	Agents and Sessions
Shortcut code	AG18
API block name	AgentsDO.ProgAgPerf
Parameters	Agent parameters - See Variables bound for Agent blocks
See also	AG09

AG19 - Detailed Pause Tracking

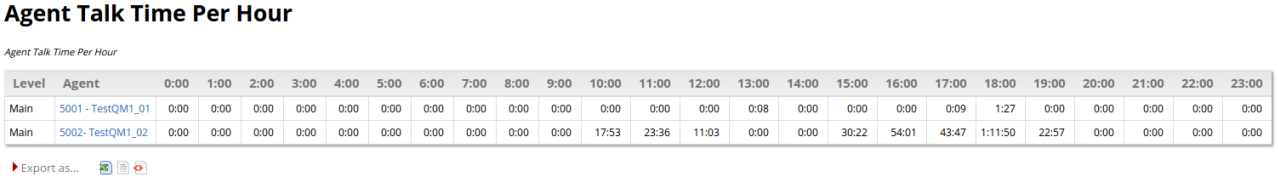


This block shows detailed information about agent pauses, including total pause time, average duration, maximum duration, minimum duration and number of pauses. This information is aggregated by agent and pause. For each agent/pause pair the agent's name and the pause code are displayed, together with the information described above.

The last two columns show the percentage of such pauses out of the the total session time for the given agent, and out of the total pause time for the given agent. For example, if an agent was logged in for four hours, two of which were on pause, and the current pause code has a total of one hour, it would show as 50% of all pauses and 25% of all session time for the agent.

Available since	16.10 - updated in 24.05
Default page	Agents and Sessions
Shortcut code	AG19
API block name	AgentsDO.DetailedPauses
Parameters	-
See also	AG23

AG20 - Agent Talk Time Per Hour



This block breaks down all the agents' talk times in one-hour long timeslots. This information is aggregated by agent, so you can keep track of how many minutes or hours of conversation your agents are having during the course of the day, across a given period of time. The name of the agent and the agent's level are displayed as well as the information listed above.

Available since	16.10.13
Default page	Agents and Sessions
Shortcut code	AG20
API block name	AgentsDO.AgentTTPerHour
Parameters	-
See also	

AG21 - Agent Outbound Productivity

Agent	Location	Total Calls	Unique calls	%	Unique answered	%	Unique effective	%	Total call time
Alina	Secondary location	148	148	100.0%	148	100.0%	135	91.2%	16:50:02
Anne B	-	126	125	99.2%	123	98.4%	114	92.7%	15:28:51
Tammy	-	117	117	100.0%	116	99.1%	111	95.7%	14:26:53
Jerry E	Secondary location	130	128	98.5%	126	98.4%	108	85.7%	12:26:54
Cindy L	-	125	124	99.2%	124	100.0%	112	90.3%	11:24:05
Leza	-	112	111	99.1%	109	98.2%	96	88.1%	8:47:57
Allison D	-	69	69	100.0%	68	98.6%	61	89.7%	6:21:43
Kelly N	-	49	47	95.9%	47	100.0%	43	91.5%	5:17:08
Wally O	-	32	32	100.0%	31	96.9%	30	96.8%	4:08:31
Candy L	-	14	14	100.0%	14	100.0%	11	78.6%	1:16:10
Betty L	-	1	1	100.0%	1	100.0%	1	100.0%	5:32

This block counts calls linked to an agent (both taken and lost); it is meant for outbound queues, though you can use it whenever it makes sense.:

- Agent: the code/name and group icon
- Location: the agent's location (if set)
- Total calls: all calls handled by this agent, answered or lost
- Unique calls: all calls handled by this agent that have a unique telephone number
- (Unique) %: the percentage of unique calls
- Unique answered: all answered calls that have a unique number
- (Answered) %: The percentage of unique answered to unique calls
- Unique effective: the number of answered calls that lasted at least `X` seconds that have a unique number - also used in OK16 and OK17.
- (Effective) %: the percentage of unique calls that are effective
- Total Call Time: the total amount of time the agent spoke with a customer

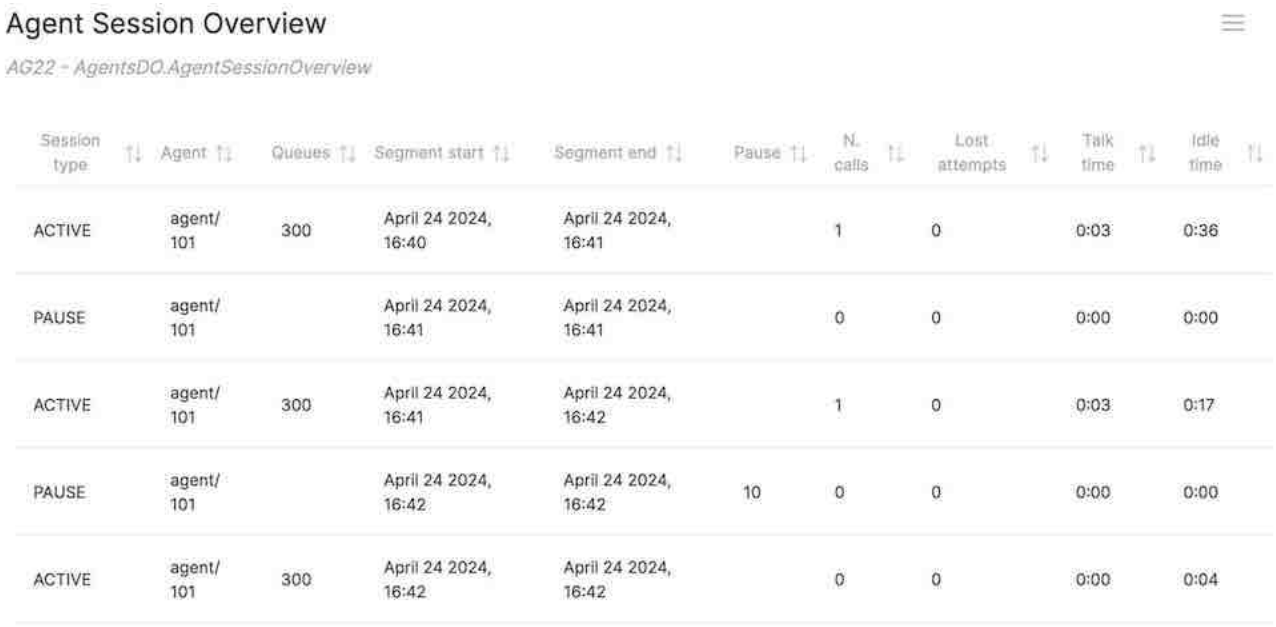
Call numbers are computed lexicographically with no further processing, as they appear on the `queue_log` table. If a call has an empty number, then it counted in the number of calls and total call time but not in unique calls.

The report is sorted by talk time (descending) but can be sorted by name or any other value by clicking on its column header.

The threshold for including calls into Unique Effective is the configuration parameter `default.effectiveCallThresold`; default is 150 seconds.

Available since	19.10.x
Default page	Agents and Sessions
Shortcut code	AG21
API block name	AgentsDO.OutboundProductivity
Parameters	-
See also	OK16, OK17

AG22 - Agent Session Overview



Session type	Agent	Queues	Segment start	Segment end	Pause	N. calls	Lost attempts	Talk time	Idle time
ACTIVE	agent/101	300	April 24 2024, 16:40	April 24 2024, 16:41		1	0	0:03	0:36
PAUSE	agent/101		April 24 2024, 16:41	April 24 2024, 16:41		0	0	0:00	0:00
ACTIVE	agent/101	300	April 24 2024, 16:41	April 24 2024, 16:42		1	0	0:03	0:17
PAUSE	agent/101		April 24 2024, 16:42	April 24 2024, 16:42	10	0	0	0:00	0:00
ACTIVE	agent/101	300	April 24 2024, 16:42	April 24 2024, 16:42		0	0	0:00	0:04

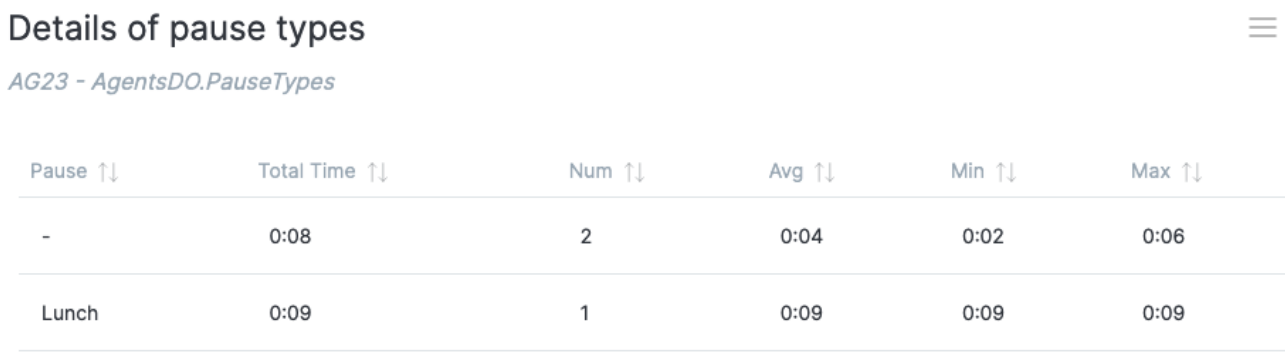
This data block is mostly useful in a JSON/RPC environment, where you have in a machine-friendly format the same data you get in the agent timeline. All agent activity is split into "segments" of uniform activity for each agent, and data has a minimal decoration for machine interoperability.

- Session type: whether the agent is active or paused
- Agent: the agent involved
- Queues: one or more queues the agent is logged on to
- Segment start and end
- Pause: the pause code
- N. calls
- N. lost attempts
- Talk time (if not paused)
- Idle time (if not paused)

Available since	24.05
Default page	Agents and Sessions

Shortcut code	AG22
XML-RPC code	AgentsDO.AgentSessionOverview
See also	

AG23 - Details of Pause Types



This report shows, for all of the pauses within the current report, given each pause code:

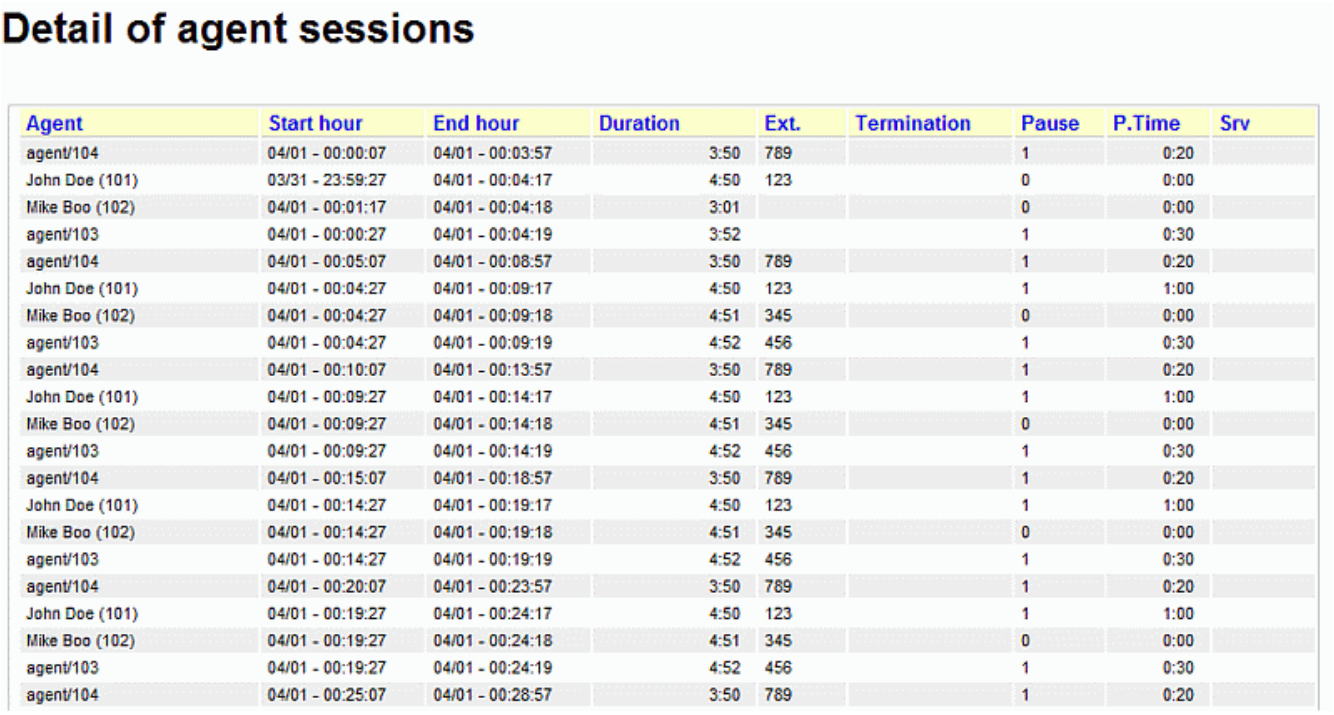
- the total and average time spent on a pause code
- the number of pauses
- the average, minimun and maximum duration

If the pause code is unset, it is shown as "-"

Available since	24.05
Default page	Agents and Sessions
Shortcut code	AG23
API block name	AgentsDO.PauseTypes
Parameters	-
See also	AG19

Historical reports - Details of agent sessions and pauses

AD01 - Detail of agent sessions



Detail of agent sessions

Agent	Start hour	End hour	Duration	Ext.	Termination	Pause	P.Time	Srv
agent/104	04/01 - 00:00:07	04/01 - 00:03:57	3:50	789		1	0:20	
John Doe (101)	03/31 - 23:59:27	04/01 - 00:04:17	4:50	123		0	0:00	
Mike Boo (102)	04/01 - 00:01:17	04/01 - 00:04:18	3:01			0	0:00	
agent/103	04/01 - 00:00:27	04/01 - 00:04:19	3:52			1	0:30	
agent/104	04/01 - 00:05:07	04/01 - 00:08:57	3:50	789		1	0:20	
John Doe (101)	04/01 - 00:04:27	04/01 - 00:09:17	4:50	123		1	1:00	
Mike Boo (102)	04/01 - 00:04:27	04/01 - 00:09:18	4:51	345		0	0:00	
agent/103	04/01 - 00:04:27	04/01 - 00:09:19	4:52	456		1	0:30	
agent/104	04/01 - 00:10:07	04/01 - 00:13:57	3:50	789		1	0:20	
John Doe (101)	04/01 - 00:09:27	04/01 - 00:14:17	4:50	123		1	1:00	
Mike Boo (102)	04/01 - 00:09:27	04/01 - 00:14:18	4:51	345		0	0:00	
agent/103	04/01 - 00:09:27	04/01 - 00:14:19	4:52	456		1	0:30	
agent/104	04/01 - 00:15:07	04/01 - 00:18:57	3:50	789		1	0:20	
John Doe (101)	04/01 - 00:14:27	04/01 - 00:19:17	4:50	123		1	1:00	
Mike Boo (102)	04/01 - 00:14:27	04/01 - 00:19:18	4:51	345		0	0:00	
agent/103	04/01 - 00:14:27	04/01 - 00:19:19	4:52	456		1	0:30	
agent/104	04/01 - 00:20:07	04/01 - 00:23:57	3:50	789		1	0:20	
John Doe (101)	04/01 - 00:19:27	04/01 - 00:24:17	4:50	123		1	1:00	
Mike Boo (102)	04/01 - 00:19:27	04/01 - 00:24:18	4:51	345		0	0:00	
agent/103	04/01 - 00:19:27	04/01 - 00:24:19	4:52	456		1	0:30	
agent/104	04/01 - 00:25:07	04/01 - 00:28:57	3:50	789		1	0:20	

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.

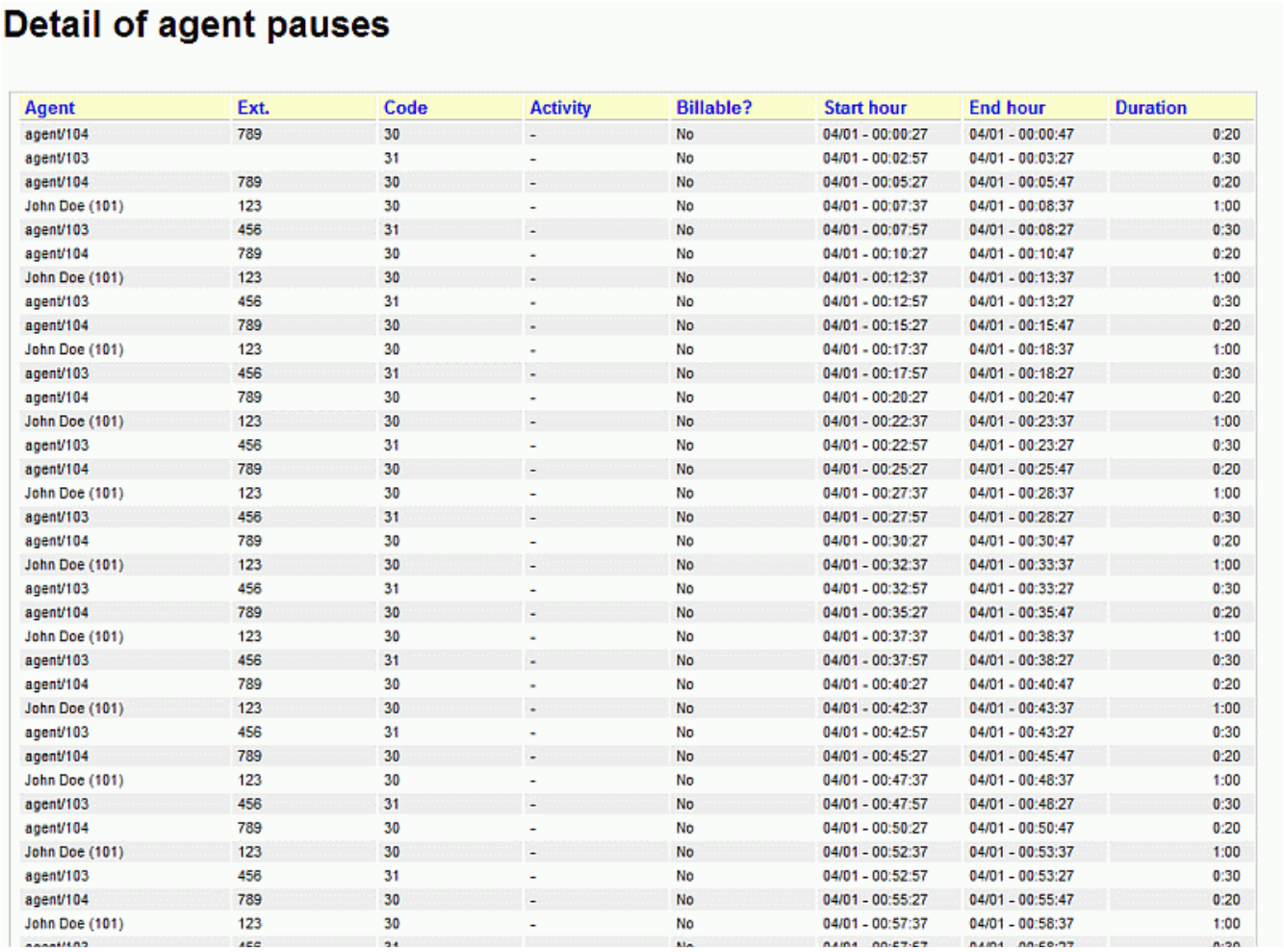
The number of pauses and the total pause time in seconds is shown.

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 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.

Available since	1.6.0
Default page	Details of Agent sessions and pauses
Shortcut code	AD01
API block name	DetailsDO.AgentSessions
Parameters	-
See also	

AD02 - Detail of agent pauses



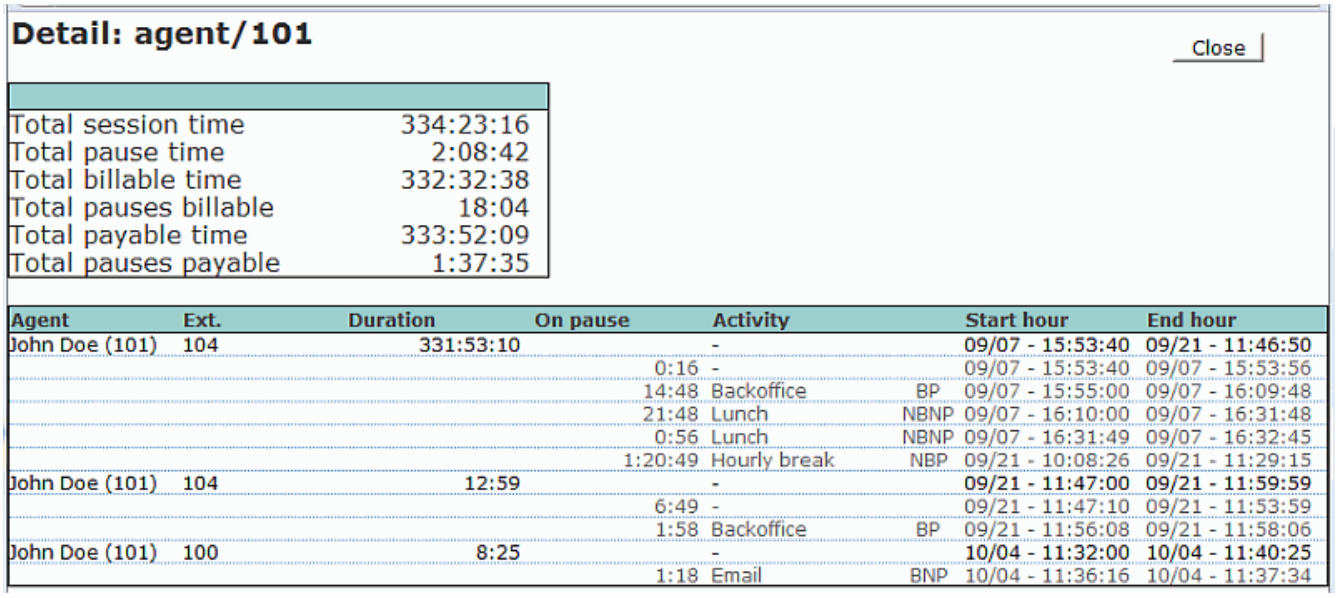
Detail of agent pauses

Agent	Ext.	Code	Activity	Billable?	Start hour	End hour	Duration
agent/104	789	30	-	No	04/01 - 00:00:27	04/01 - 00:00:47	0:20
agent/103		31	-	No	04/01 - 00:02:57	04/01 - 00:03:27	0:30
agent/104	789	30	-	No	04/01 - 00:05:27	04/01 - 00:05:47	0:20
John Doe (101)	123	30	-	No	04/01 - 00:07:37	04/01 - 00:08:37	1:00
agent/103	456	31	-	No	04/01 - 00:07:57	04/01 - 00:08:27	0:30
agent/104	789	30	-	No	04/01 - 00:10:27	04/01 - 00:10:47	0:20
John Doe (101)	123	30	-	No	04/01 - 00:12:37	04/01 - 00:13:37	1:00
agent/103	456	31	-	No	04/01 - 00:12:57	04/01 - 00:13:27	0:30
agent/104	789	30	-	No	04/01 - 00:15:27	04/01 - 00:15:47	0:20
John Doe (101)	123	30	-	No	04/01 - 00:17:37	04/01 - 00:18:37	1:00
agent/103	456	31	-	No	04/01 - 00:17:57	04/01 - 00:18:27	0:30
agent/104	789	30	-	No	04/01 - 00:20:27	04/01 - 00:20:47	0:20
John Doe (101)	123	30	-	No	04/01 - 00:22:37	04/01 - 00:23:37	1:00
agent/103	456	31	-	No	04/01 - 00:22:57	04/01 - 00:23:27	0:30
agent/104	789	30	-	No	04/01 - 00:25:27	04/01 - 00:25:47	0:20
John Doe (101)	123	30	-	No	04/01 - 00:27:37	04/01 - 00:28:37	1:00
agent/103	456	31	-	No	04/01 - 00:27:57	04/01 - 00:28:27	0:30
agent/104	789	30	-	No	04/01 - 00:30:27	04/01 - 00:30:47	0:20
John Doe (101)	123	30	-	No	04/01 - 00:32:37	04/01 - 00:33:37	1:00
agent/103	456	31	-	No	04/01 - 00:32:57	04/01 - 00:33:27	0:30
agent/104	789	30	-	No	04/01 - 00:35:27	04/01 - 00:35:47	0:20
John Doe (101)	123	30	-	No	04/01 - 00:37:37	04/01 - 00:38:37	1:00
agent/103	456	31	-	No	04/01 - 00:37:57	04/01 - 00:38:27	0:30
agent/104	789	30	-	No	04/01 - 00:40:27	04/01 - 00:40:47	0:20
John Doe (101)	123	30	-	No	04/01 - 00:42:37	04/01 - 00:43:37	1:00
agent/103	456	31	-	No	04/01 - 00:42:57	04/01 - 00:43:27	0:30
agent/104	789	30	-	No	04/01 - 00:45:27	04/01 - 00:45:47	0:20
John Doe (101)	123	30	-	No	04/01 - 00:47:37	04/01 - 00:48:37	1:00
agent/103	456	31	-	No	04/01 - 00:47:57	04/01 - 00:48:27	0:30
agent/104	789	30	-	No	04/01 - 00:50:27	04/01 - 00:50:47	0:20
John Doe (101)	123	30	-	No	04/01 - 00:52:37	04/01 - 00:53:37	1:00
agent/103	456	31	-	No	04/01 - 00:52:57	04/01 - 00:53:27	0:30
agent/104	789	30	-	No	04/01 - 00:55:27	04/01 - 00:55:47	0:20
John Doe (101)	123	30	-	No	04/01 - 00:57:37	04/01 - 00:58:37	1:00

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.

Available since	1.6.0
Default page	Details of Agent sessions and pauses
Shortcut code	AD02
API block name	DetailsDO.AgentPauses
Parameters	-
See also	

Popup of agent activity



Detail: agent/101

Close

Total session time	334:23:16
Total pause time	2:08:42
Total billable time	332:32:38
Total pauses billable	18:04
Total payable time	333:52:09
Total pauses payable	1:37:35

Agent	Ext.	Duration	On pause	Activity		Start hour	End hour
John Doe (101)	104	331:53:10		-		09/07 - 15:53:40	09/21 - 11:46:50
			0:16	-		09/07 - 15:53:40	09/07 - 15:53:56
			14:48	Backoffice	BP	09/07 - 15:55:00	09/07 - 16:09:48
			21:48	Lunch	NBNP	09/07 - 16:10:00	09/07 - 16:31:48
			0:56	Lunch	NBNP	09/07 - 16:31:49	09/07 - 16:32:45
			1:20:49	Hourly break	NBP	09/21 - 10:08:26	09/21 - 11:29:15
John Doe (101)	104	12:59		-		09/21 - 11:47:00	09/21 - 11:59:59
			6:49	-		09/21 - 11:47:10	09/21 - 11:53:59
			1:58	Backoffice	BP	09/21 - 11:56:08	09/21 - 11:58:06
John Doe (101)	100	8:25		-		10/04 - 11:32:00	10/04 - 11:40:25
			1:18	Email	BNP	10/04 - 11:36:16	10/04 - 11:37:34

This is not a normal data block but it is the popup that is displayed whenever you click on an agent's name.

- The top table reports session, pause, billable and payable time
- The bottom table shows the details of all agent sessions, with a break-up of all pauses that were made during that session. For each pause with a known pause code, their billable-payable status is displayed:
  - **BP**: Pause is Billable and Payable
  - **BNP**: Pause is Billable but not payable (be careful!)
  - **NBP**: Pause is not billable but Payable
  - **NBNP**: Pause is neither billable nor payable

In the latest release a new "Conversation" column has been added on, to the right of the "On Pause" column, which displays the pause time there may have been while actually in conversation.

You can close the popup by clicking on the Close button.

Historical reports - 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.

OU01 - Outcomes

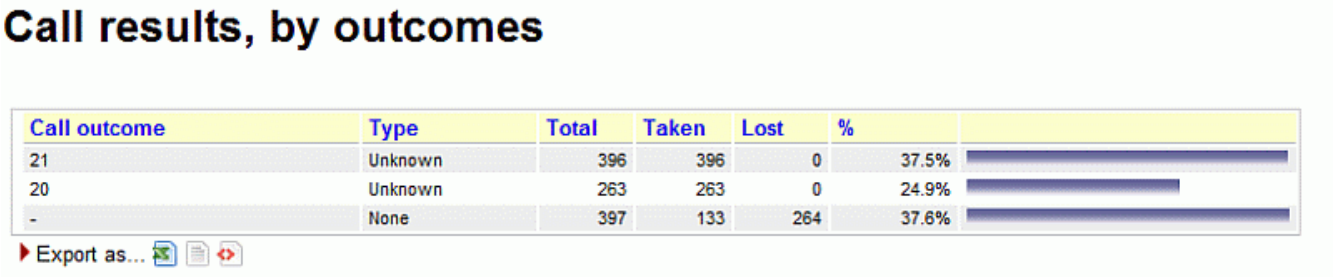


This report shows:

- 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, Qualified Contacts and Sales, as defined by call outcome codes
- The *Sales per Hour* (SPH), *Qualified Contacts per hour* (QCPH) and *Contacts per Hour* (CPH) ratios
- The *Conversion index*, that is the percentage of sales over the total number of sales and contacts.

Available since	1.6.0
Default page	Call outcomes
Shortcut code	OU01
API block name	OutcomesDO.GeneralRep
Parameters	-
See also	

OU02 - Call results, by outcomes



This report shows:

- The number of total calls grouped by outcome
- The number of answered calls grouped by outcome
- The number of unanswered calls grouped by outcome
- The percentage of calls related to each outcome
- The percentage of taken calls related to each outcome (available since 16.10.13)

Available since	1.6.0
Default page	Call outcomes
Shortcut code	OU02
API block name	OutcomesDO.CallResByOutcome
Parameters	-
See also	

OU03 - Billable activities



In this report you can find details explaining Billable activities, with average, minimum and maximum session durations, and a percentage on all activities of the same kind.

Available since	1.6.0
Default page	Call outcomes
Shortcut code	OU03
API block name	OutcomesDO.ActivBillable
Parameters	-
See also	

OU04 - Non billable activities

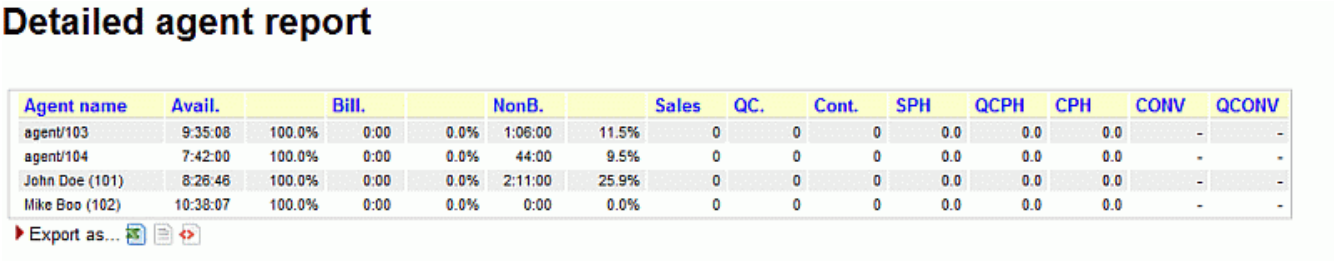


In this report you can find details explaining Non billable activities, with average, minimum and

maximum session durations, and a percentage on all activities of the same kind.

Available since	1.6.0
Default page	Call outcomes
Shortcut code	OU04
API block name	OutcomesDO.ActivNotBillable
Parameters	-
See also	

OU05 - Detailed agent report



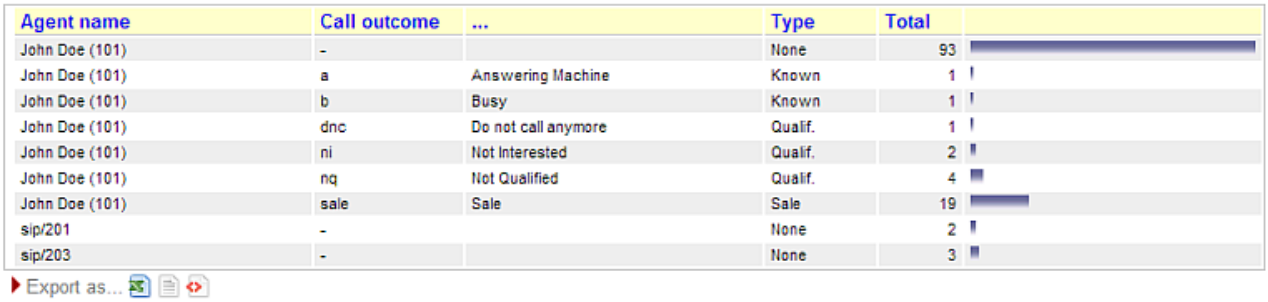
The Detailed Agent Report will show, for each agent:

- 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 a s 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.

Available since	1.6.0
Default page	Call outcomes
Shortcut code	OU05
API block name	OutcomesDO.AgentReportDetailed
Parameters	-
See also	

OU06 - Outcomes per agent

Outcomes per agent



Agent name	Call outcome	...	Type	Total
John Doe (101)	-		None	93
John Doe (101)	a	Answering Machine	Known	1
John Doe (101)	b	Busy	Known	1
John Doe (101)	dnc	Do not call anymore	Qualif.	1
John Doe (101)	ni	Not Interested	Qualif.	2
John Doe (101)	nq	Not Qualified	Qualif.	4
John Doe (101)	sale	Sale	Sale	19
sip/201	-		None	2
sip/203	-		None	3

Export as...

This data block displays all the status codes for each call handled by each agent within the current reporting selection.

If a call has no associated status code, then it is counted under a special status code called "-".

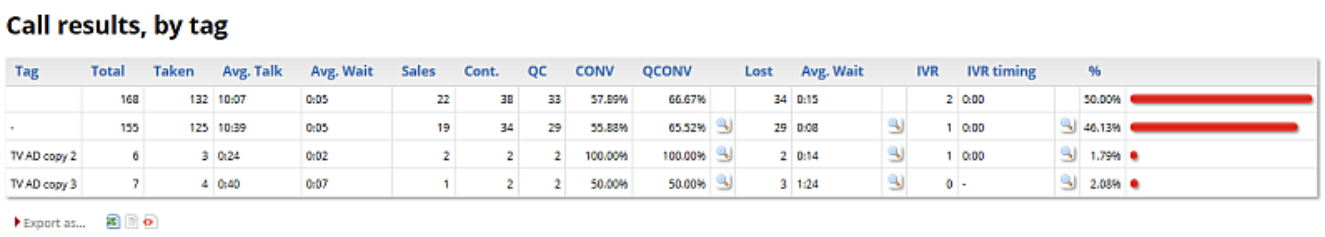
The columns show:

- The agent name. As an agent might have multiple status codes, the agent name will appear multiple times. If the agent belongs to an agent group, the proper icon is displayed.
- The call outcome, specified both as a code and a description (if it is present)
- The type currently associated with the status code
- The total number of calls for that agent under that status code

All columns are client-sortable.

Available since	13.04.3
Default page	Call outcomes
Shortcut code	OU06
API block name	OutcomesDO.AgentOutcomes
Parameters	-
See also	

OU07 - Call results, by tag



Call results, by tag

Tag	Total	Taken	Avg. Talk	Avg. Wait	Sales	Cont.	QC	CONV	QCONV	Lost	Avg. Wait	IVR	IVR timing	%
	168	132	10:07	0:05	22	38	33	57.89%	66.67%	34	0:15	2	0:00	50.00%
-	155	125	10:39	0:05	19	34	29	55.88%	65.52%	29	0:08	1	0:00	46.13%
TV AD copy 2	6	3	0:24	0:02	2	2	2	100.00%	100.00%	2	0:14	1	0:00	1.79%
TV AD copy 3	7	4	0:40	0:07	1	2	2	50.00%	50.00%	3	1:24	0	-	2.08%

Export as...

This data block displays calls aggregated by tag code. The first line reports the figures for all aggregated call. The second row, marked with the *dash* code as tag, counts aggregates for all calls with no tag code assigned.

The columns show:

- The tag code name
- Total number of calls aggregated
- Taken calls
- Average talk time
- Average wait time on taken calls
- Number of calls flagged as sales
- Number of calls flagged as contact
- Number of calls flagged as qualified contact
- Conversion ratio
- Qualified conversion ratio
- Number of lost calls
- Average wait on lost calls
- Average wait on IVR calls
- Time spent on IVR
- Ratio and histogram for total calls the row is reporting, compared with the overall total call in the report

All columns are client-sortable. Clicking on the lens icons present in the report will open a popup with call details (taken, lost and IVR) counted in the aggregation.

Available since	15.09
Default page	Call outcomes
Shortcut code	OU07
API block name	CallTagDO.CallResByTag
Parameters	-
See also	

OU08 - Call results, by feature code

Call results, by feature code

Feature	Goal	Total	Ftr. Conv.	Taken	Avg. Talk	Avg. Wait	Sales	Cont.	QC	CONV	QCONV	Lost	Avg. Wait	%
		166	436%	132	10:07	0:05	22	38	33	57.89%	66.67%	34	0:15	47.03%
-		145	381%	111	8:43	0:05	11	24	19	45.83%	57.89%	34	0:15	41.08%
Householder		10	26%	10	0:57	0:08	8	8	8	100.00%	100.00%	0	-	2.83%
Not sales rep.		11	28%	11	1:10	0:12	8	8	8	100.00%	100.00%	0	-	3.12%
Sales Target		1	2%	1	0:22	0:04	1	1	1	100.00%	100.00%	0	-	0.28%
Costs too much		13	34%	13	28:06	0:10	5	8	8	62.50%	62.50%	0	-	3.68%
Retry later		7	18%	7	51:33	0:17	4	4	4	100.00%	100.00%	0	-	1.98%

Export as...

This data block displays calls aggregated by feature code. The first line reports the figures for all

aggregated call. The second row, marked with the *dash* code as feature, counts aggregates for all calls with no feature code assigned.



The first line shows the number of unique calls. However, this number may not match the sum of the other rows, since a single call can have multiple features and may therefore appear in more than one row.

The columns show:

- The feature code name
- Goal/Not goal flag for feature code
- Total number of calls aggregated
- Feature conversion factor, counted as ratio between the total calls and the call flagged as contact
- Taken calls
- Average talk time
- Average wait time on taken calls
- Number of calls flagged as sales
- Number of calls flagged as contact
- Number of calls flagged as qualified contact
- Conversion ratio
- Qualified conversion ratio
- Number of lost calls
- Average wait on lost calls
- Ratio and histogram for total calls the row is reporting, compared with the overall total call in the report

All columns are client-sortable.

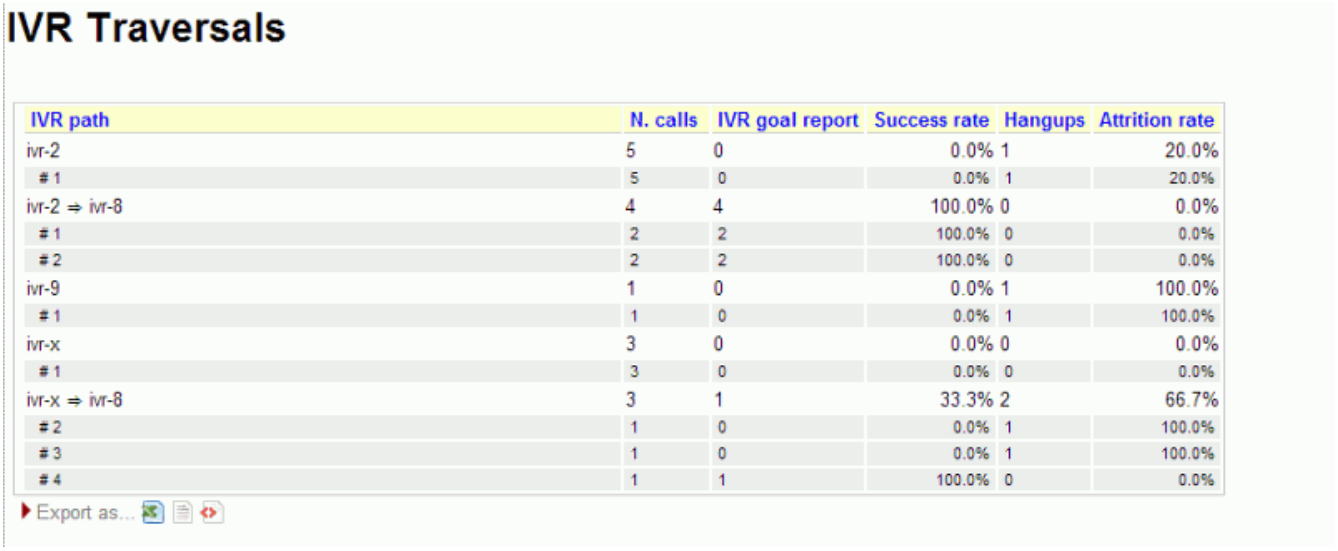
Available since	15.09
Default page	Call outcomes
Shortcut code	OU08
API block name	OutcomesDO.CallResByFeature
Parameters	-
See also	

Historical reports - IVR tracking

The following reports let you track information about IVR calls and goals. A detailed explanation of

IVR can be found in: Tracking IVR data

IV01 - IVR Traversals



This block shows:

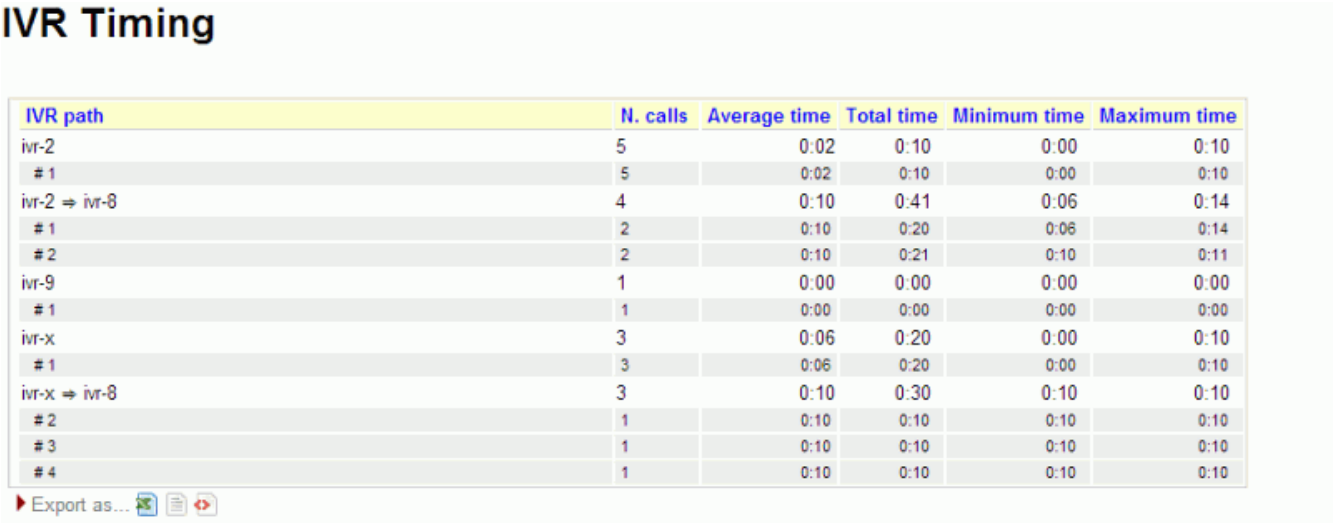
- The IVR paths examined, in total (as the sum of all choices)
- For each IVR path, the choices made on that path (prepended by #)

For each path, the following information is computed:

- The number of calls traversing that path
- The number of IVR goals reached immediately after that path (a goal can be a queue or an activity expressly earmarked as a goal)
- The success rate, as the percentage of goals on all calls that entered the menu
- The number of hang-ups made in that IVR menu
- The attrition rate, as the number of hang-ups versus total number of calls

Available since	13.03
Default page	IVR
Shortcut code	IV01
API block name	IvrDO.IvrReport
Parameters	-
See also	

IV02 - IVR Timing



IVR Timing

IVR path	N. calls	Average time	Total time	Minimum time	Maximum time
ivr-2	5	0:02	0:10	0:00	0:10
# 1	5	0:02	0:10	0:00	0:10
ivr-2 ⇒ ivr-8	4	0:10	0:41	0:06	0:14
# 1	2	0:10	0:20	0:06	0:14
# 2	2	0:10	0:21	0:10	0:11
ivr-9	1	0:00	0:00	0:00	0:00
# 1	1	0:00	0:00	0:00	0:00
ivr-x	3	0:06	0:20	0:00	0:10
# 1	3	0:06	0:20	0:00	0:10
ivr-x ⇒ ivr-8	3	0:10	0:30	0:10	0:10
# 2	1	0:10	0:10	0:10	0:10
# 3	1	0:10	0:10	0:10	0:10
# 4	1	0:10	0:10	0:10	0:10

Export as...

This block shows:

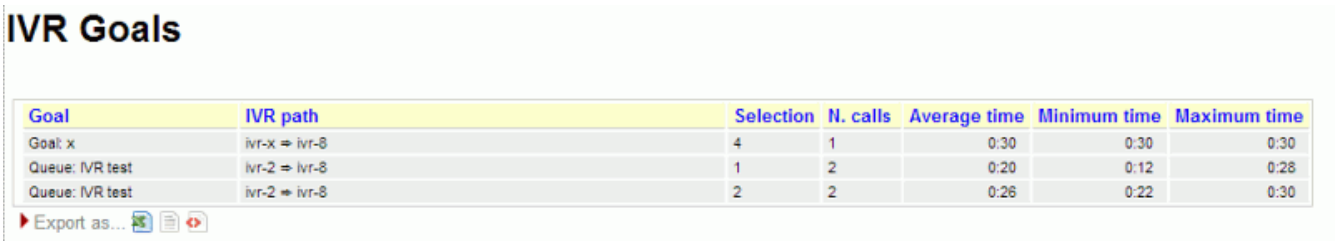
- The IVR paths examined, in total (as the sum of all choices)
- For each IVR path, the choices made on that path (prepended by #)

For each path, the following information is computed:

- The number of calls traversing that path
- The average traversal time for that IVR menu
- The total traversal time (as the sum of all times for each call)
- The maximum and minimum traversal times

Available since	13.03
Default page	IVR
Shortcut code	IV02
API block name	IvrDO.IvrTiming
Parameters	-
See also	

IV03 - IVR Goals



IVR Goals

Goal	IVR path	Selection	N. calls	Average time	Minimum time	Maximum time
Goal: x	ivr-x ⇒ ivr-8	4	1	0:30	0:30	0:30
Queue: IVR test	ivr-2 ⇒ ivr-8	1	2	0:20	0:12	0:28
Queue: IVR test	ivr-2 ⇒ ivr-8	2	2	0:26	0:22	0:30

Export as...

This block shows:

- For each goal, all the paths that lead to that goal and their selection (the digit that was pressed in order to reach the goal)

- The number of calls that reached each goal
- The average, minimum and maximum traversal time to to reach the goal. The time is computed since the call is first tracked, and not only on the final IVR menu.

Available since	13.03
Default page	IVR
Shortcut code	IV03
API block name	IvrDO.IvrGoals
Parameters	-
See also	

Historical reports - First Call Resolution (FCRs)

First Call Resolution lets you track cases that are not resolved in a single call and so the customer ends up calling multiple times in a specified time frame.

Repeated callers and clusters

At the basis of FCR, there is a definition of "repeated call". A call is considered "repeated" if:

- the caller-id is the same for multiple calls; e.g. if some user calls from a different number, we do not consider this a repeated call.
- the same caller-id is calling multiple times that are separated by less than a given time period, called "margin".
- the margin is controlled by a configuration property called *default.groupingMargin*, that can be changed in the system configuration and which default value is 168 hours (7 days).
- the caller-id is considered the same if after a normalization process (stripping of whitespaces and non-numeric characters) the same caller-ids match
- the caller-id is skipped (not clustered) if after the normalization it is blank – e.g. caller-ids like "" or "anonymous" will be skipped. Any calls with a blank caller-id will be counted as a single-call, as we do not have further information to cluster it.
- the normalized caller-id appears at least twice

For example, given the following set of calls and a marging of one week

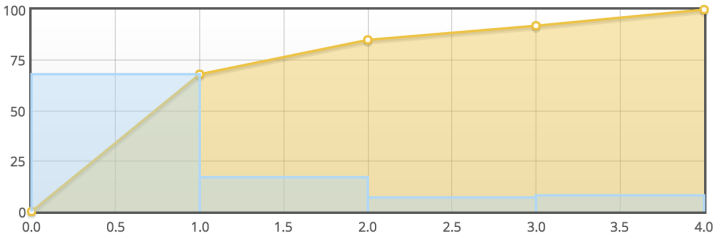
The data blocks are computed in three different flavors:

- Taken calls
- Lost calls
- All calls

FC01/2/3 - Break-down of recalls

Recalls on answered calls

	Clusters	Delta	%	Total calls	Total %
FCRs - 1	43		68.3%	43	44.3%
Up to attempts: 2	54	+ 11	85.7%	65	67.0%
Up to attempts: 3	58	+ 4	92.1%	77	79.4%
Up to attempts: 4	63	+ 5	100.0%	97	100.0%



Export as...

This report aims at letting you know how many calls were clustered.

This report is built to have a “shape” that is familiar for users used to SLAs, and is completed by a similar graph.

In detail:

- FCRs are calls that appear only once; while the other "classes" will be calls "up to" so many attempts
- "Up-to" lines will be present up to the maximum present in the report; considering all integers from 2 to the maximum included
- Clusters is the number of clusters that are "up to", so clusters that belong to the current class plus all previous ones
- Delta is the number of clusters that belong to the current class
- Percentage is the number of calls of the current class
- Total is the growing percentage of sorted classes, and always ends at 100%
- Total Calls is the number of calls that are "up to", so belong to the current class plus all previous ones

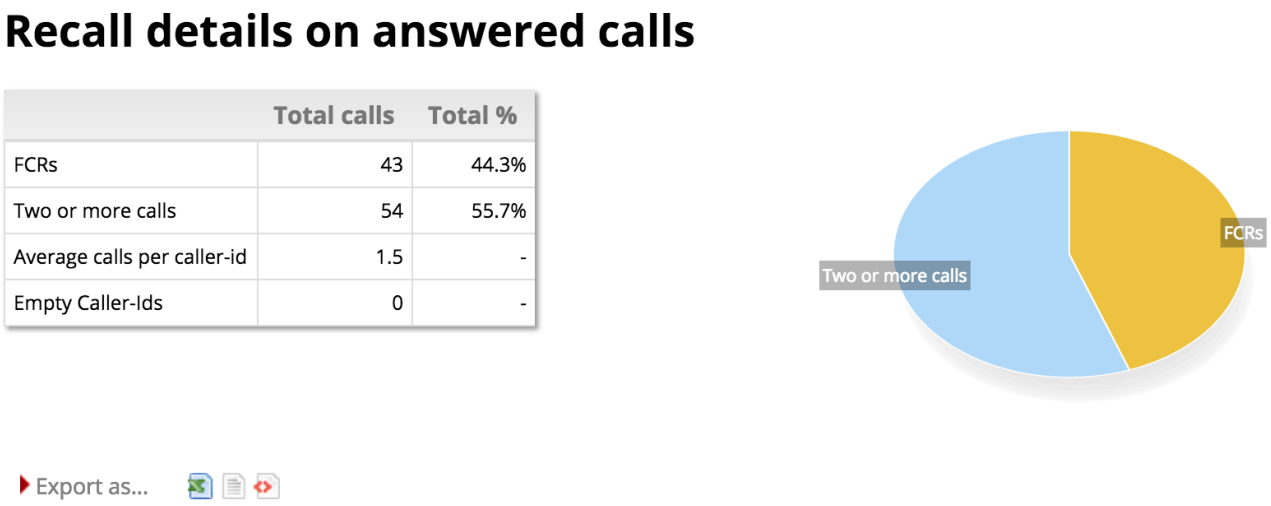
Available since	15.10.11
Default page	FCRs
Shortcut code	FC01
API block name	FcrDO.RecallsOk
Parameters	-
See also	Taken calls

Available since	15.10.11
Default page	FCRs
Shortcut code	FC02
API block name	FcrDO.RecallsKo

Parameters	-
See also	Lost calls

Available since	15.10.11
Default page	FCRs
Shortcut code	FC03
API block name	FcrDO.RecallsAll
Parameters	-
See also	All calls

FC04/5/6 - Recall details



This report shows the total number of FCRs, clusters and calls with an empty IDs.

It has a pie chart showing FCRs vs clusters.

Available since	15.10.11
Default page	FCRs
Shortcut code	FC04
API block name	FcrDO.ClidsOk
Parameters	-
See also	Taken calls

Available since	15.10.11
Default page	FCRs
Shortcut code	FC05
API block name	FcrDO.ClidsKo
Parameters	-

See also	Lost calls

Available since	15.10.11
Default page	FCRs
Shortcut code	FC06
API block name	FcrDO.ClidsAll
Parameters	-
See also	All calls

FC07/8/9 - Cluster details

List of recall clusters

N attempts	Number	First calls	End of last call	Taken calls	Lost calls
6	091 7	06/09 - 08:54:02	06/22 - 13:23:22	3	3
5	091 5	05/11 - 14:05:14	05/16 - 14:07:44	4	1
5	091 5	05/31 - 11:29:20	06/03 - 13:45:12	3	2
4	001 8010	06/07 - 05:58:12	06/07 - 07:39:19	3	1
4	004 30000	04/20 - 15:07:26	04/28 - 15:52:54	4	0
4	091 5	06/24 - 09:12:51	06/30 - 07:35:10	4	0
4	091 0	06/13 - 09:15:38	06/24 - 08:24:07	4	0
4	091 0	06/20 - 07:33:35	06/30 - 13:19:21	4	0
3	001 212	04/19 - 14:11:52	04/19 - 14:47:47	1	2
3	003 61261	05/12 - 08:39:01	05/13 - 12:09:29	3	0
3	091 5	06/10 - 16:01:12	06/17 - 07:09:07	2	1
3	091 0	05/04 - 09:14:48	05/10 - 06:20:35	2	1

This block is a list of all clusters, sorted by the number of attempts and showing the cluster's first and last call. Only clustered calls are included.

For all clusters including "taken" calls, the "Taken" column appears showing how many of those calls were counted and offering a link to a pop-up from which you can access the calls. The same goes for "Lost" calls. If both are counted, both columns are present.

Notes:

- When clicking on the icon to see calls, you will see all calls with the same caller-id belonging to the current report, whether they belong to the current cluster or not; so that you can have a general overview of all recalls in the current report

Available since	15.10.11

Default page	FCRs
Shortcut code	FC07
API block name	FcrDO.ClustersOk
Parameters	-
See also	Taken calls

Available since	15.10.11
Default page	FCRs
Shortcut code	FC08
API block name	FcrDO.ClustersKo
Parameters	-
See also	Lost calls

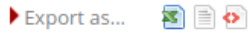
Available since	15.10.11
Default page	FCRs
Shortcut code	FC09
API block name	FcrDO.ClustersAll
Parameters	-
See also	All calls

FC10 - Retry Rate and Lost Caller Behaviour

Retry Rate and Lost Caller Behaviour

Retry Rate and Lost Caller Behaviour

Customer Attempts	Total Attempts	Long Calls	Short Calls	Taken Calls	Lost Calls
# 1	19	16	3	16	3
# 2	4	4	0	4	0
# 3	4	4	0	4	0
# 4	3	3	0	3	0
# 5	3	2	1	3	0
# 6	3	2	1	3	0
# 7	3	2	1	2	1
# 8	3	3	0	3	0
# 9	2	2	0	2	0
# 10	1	1	0	1	0
# 11	1	1	0	1	0
# 12	1	1	0	1	0
# 13	1	1	0	1	0
# 14	1	0	1	1	0
# 15	1	1	0	1	0



This report shows information about customers retry behaviour. We can see how many calls are registered as being the n-th attempt, based on the Customer Attempt number in the first column. The information is the following:

- Customer Attempts
- Total Attempts
- Long Calls
- Short Calls
- Taken Calls
- Lost Calls

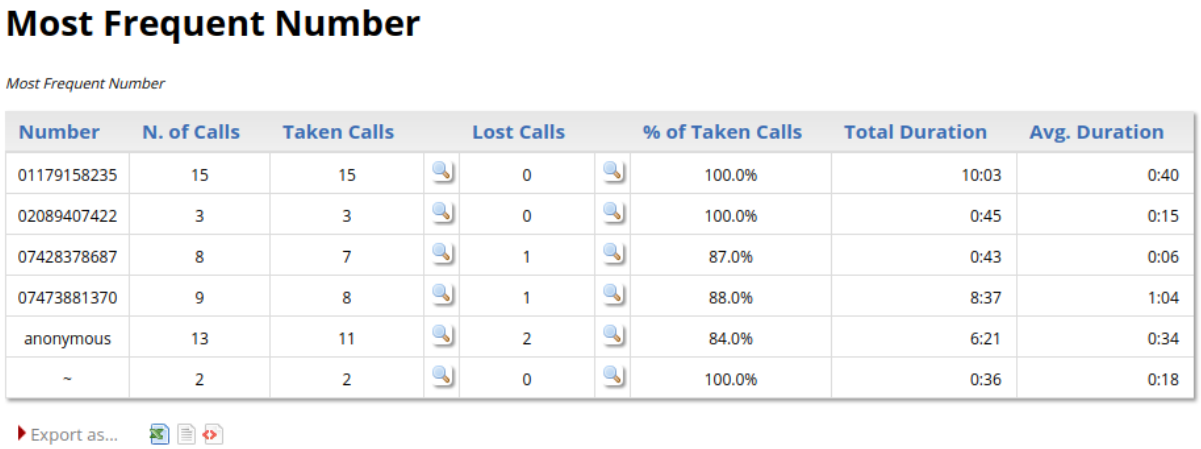
Notes:

- the number of seconds needed for a call to be classified as long is dictated by the following configuration property: default.shortcall_treshold = 30 . If the property is not specified, it will deafault at 30.

Available since	17.06.16
Default page	FCRs
Shortcut code	FC10
API block name	FcrDO.RetryRate
Parameters	-

See also	NA

FC11 - Most Frequent Called-Calling Number



Most Frequent Number

Most Frequent Number

Number	N. of Calls	Taken Calls	Lost Calls	% of Taken Calls	Total Duration	Avg. Duration
01179158235	15	15	0	100.0%	10:03	0:40
02089407422	3	3	0	100.0%	0:45	0:15
07428378687	8	7	1	87.0%	0:43	0:06
07473881370	9	8	1	88.0%	8:37	1:04
anonymous	13	11	2	84.0%	6:21	0:34
~	2	2	0	100.0%	0:36	0:18

Export as...

This report shows aggregated data by caller/called number. We can deduce by looking at this report, the most frequent callers and the duration of their calls. A special icon is also present to inspect the signular calls that make up the aggregated data.

The information is the following:

- Number
- N. of Calls
- Taken Calls
- Lost Calls
- % of Taken Calls
- Total Duration
- Avg. Duration

Notes:

- Calls that show "anonymous" as caller number are calls which have an empty or undeterminable caller id. these calls are all evaluated together. Similarly, calls that show "~" as caller id, are calls coming from caller numbers that have only called once, we can refer to them as FCR (First Call Resolution).

Available since	17.06.16
Default page	FCRs
Shortcut code	FC11
API block name	FcrDO.MostFrequentNumber
Parameters	-

See also	NA

Historical reports - Cases and Clients

CA01 - Case Distribution

This block show all cases referenced within the current report.

Ref. #	Case	Taken calls	Total calling time	Average time per call	Average wait time per call	Open Group
b123	Email server down	2	3:40	1:50	0:10	
c-1001	Connection error	1	0:05	0:05	0:03	
c-1002	DC route isue	1	0:15	0:15	0:03	

For each case, we show:

- The case's external reference and name
- How many calls reference this case
- The total handling time
- The average wait and talk time for those calls
- A detail of all calls (within the current report) that belong to this case

Available since	23.08
Default page	Cases
Shortcut code	CA01
API block name	CaseDO.CaseDistribution
Parameters	-
See also	CA02

CA02 - Client Distribution

This block show all clients referenced within the current report.

Ref. #	Client	Taken calls	Total calling time	Average time per call	Average wait time per call	Open Group
c102	Zebra Company Ltd	3	3:55	1:18	0:07	
k-1001	Stripes Point Inc	1	0:05	0:05	0:03	
unk7	unk7	1	0:20	0:20	0:10	

For each client, we show:

- The client's external reference and name
- How many calls reference this client
- The total handling time
- The average wait and talk time for those calls
- A detail of all calls (within the current report) that belong to this client



As explained in Configuring cases, a case may belong to a different client than its own, if set so by an agent.

Available since	23.08
Default page	Cases
Shortcut code	CA02
API block name	CaseDO.ClientDistribution
Parameters	-
See also	CA01

Historical reports - Microsoft Teams



These reports are meant for users running QueueMetrics for Microsoft Teams. If your data source is not MS Teams, then they may not work or be misleading.

In Microsoft Teams, there are different types of users/callers that can be seen within QueueMetrics. Specifically, they are:

- **Agents and Teams users** who belong to your tenant
- **Phone numbers** originating from the public network
- **Guest users**, i.e., users belonging to another tenant who have been added to the current tenant
- **External users**, i.e., people who connect directly but come from a different tenant, or people

who are not further identified and join an on-line conference by clicking on an anonymous link

The roles of a user can be:

- **Caller**: the one intiating the interaction. In an inbound queue, is the person who called that queue. In an outbound call, the agent who started it.
- **Callee**: in an inbound call, the agent who answered the call. In an outbound call, the number that was dialled.

So, in the case of outbound calls, caller and callee are automatically reversed.

TM01 - External Callers

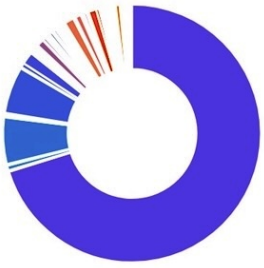
External Callers

Edit Columns Delete

TM01 - TeamsDO.ExternalCallersOnTaken

Caller domain	All Calls	Guest N Calls	Guest Avg Wait	Guest Avg Talk	Guest Tot Talk	Ext N Calls	Ext Avg Wait	Ext Avg Talk	Ext Tot Talk
-	**2,387**	**107**	**0:00**	**1:13**	**17:02:13**	**2,280**	**0:02**	**11:10**	**3:10:03**
somedomain.com	1	0	0:00	0:00	0:00	1	0:00	28:36	28:36
other-domain.com	1	0	0:00	0:00	0:00	1	0:00	9:09	9:09
web	5	0	0:00	0:00	0:00	5	1:18	24:11	2:00:59
friendly-scammer.com	1	0	0:00	0:00	0:00	1	0:00	4:12	4:12
another-one.com	2	2	0:00	0:57	1:55	0	0:00	0:00	0:00
loremipsum.com	5	0	0:00	0:00	0:00	5	0:00	3:01	15:07
hullo.com	1	1	0:38	1:15	1:15	0	0:00	0:00	0:00

10 of 10 Showing 136 - 142 of 142 15



This block shows all successfully completed calls where the caller is either an external user or a guest. The calls are aggregated by source domain, divided between external users and guests.

The purpose of this block of course to display inter-tenant activity, but also to let you quickly spot anomalous domains that are not supposed to be there, from which scams and phishing attempts may originate.

For each domain we see:

- The total number of calls (Guests + External users)
- The total number of calls, total time, average wait time and average talk time for Guests
- The total number of calls, total time, average wait time and average talk time for External users

The first line contains a recap of all acttivity for each column.



In the case of outbound calls, the caller and called party are automatically reversed: the agent field (which is the party making the call) is considered the caller, while the destination number is the called party. For inbound calls, naturally, the caller is the external number and the called party is the agent who answered.

Available since	24.11
Default page	-
Shortcut code	TM01
API block name	TeamsDO.ExternalCallersOnTaken
Parameters	-
See also	TM02 TM03 TM04

TM02 - External Callees

This is the same as block TM01 but computed by callee and not by caller.

Available since	24.11
Default page	-
Shortcut code	TM02
API block name	TeamsDO.ExternalCalleesOnTaken
Parameters	-
See also	TM01 TM03 TM04

TM03 - External Callers on All Calls

This is the same as block TM01 but computed on all calls (both answered and unaswered ones).

Available since	24.11
Default page	-
Shortcut code	TM03
API block name	TeamsDO.ExternalCallersOnAll
Parameters	-
See also	TM01 TM02 TM04

TM04 - External Callees on All Calls

This is the same as block TM01 but computed by callee on all calls (both answered and unaswered ones).

Available since	24.11
Default page	-
Shortcut code	TM04
API block name	TeamsDO.ExternalCalleesOnAll
Parameters	-
See also	TM01 TM02 TM03

TM05 - Teams Caller Classification



This block counts the statistics of all Teams calls (answered and missed) separated by caller type.

Call types are:

- Phone calls
- Other (usually agents/Teams users)
- Guest users
- External users

The first line contains a recap of all activity for each column.

Available since	24.11
Default page	-
Shortcut code	TM05
API block name	TeamsDO.CallerClassification

Parameters	-
See also	TM06

TM06 - Teams Callee Classification

This is the same as block TM05 but computed by callee and not by caller.

Available since	24.11
Default page	-
Shortcut code	TM06
API block name	TeamsDO.CalleeClassification
Parameters	-
See also	TM05

Externally generated reports

QueueMetrics is able to fetch and display data blocks generated by external systems. This may be used to enhance the standard QueueMetrics reports with data that is handled outside of QueueMetrics - for example, data from a CRM application or order tracking systems or anything else that makes sense to display together with queue data.

In order to use these blocks, you need to create external data providers that will generate the data that QueueMetrics is to display. Details on how to do this can be found in the *XML-RPC Manual* in the section called *Dynamic blocks in QueueMetrics*.

XS01 - Plain HTML

This data block includes a block of HTML right into the QueueMetrics report page.

For example, in order to include an HTML fragment generated by a server located at http://server/page.php you would set:

```
url=http://server/page.php
```

in the data block's parameter field.

Available since	13.12 - **Removed in 20.11**
Default page	None
Shortcut code	XS01
API block name	ExtSourceDO.HTML
Parameters	*url*: The URL that points to the page generating HTML

See also	

XS02 - XML-RPC data source

This data block includes a formatted table of data into the QueueMetrics report page. The data must be generated in a specified format by an XML-RPC server, as described on the *XML-RPC Manual*.

Available since	13.12
Default page	None
Shortcut code	XS02
API block name	ExtSourceDO.XMLRPC
Parameters	*url*: The URL that points to the XML-RPC server
See also	XS01

XS03 - JSON data source

This data block includes a formatted table of data into the QueueMetrics report page. The data must be generated in a given format by a JSON server, as described on the *XML-RPC Manual*.

Available since	13.12
Default page	None
Shortcut code	XS03
API block name	ExtSourceDO.JSON
Parameters	*url*: The URL that points to the page generating a JSON document
See also	XS01



You can find some easy-to-use examples at `https://github.com/Loway/OpenQueueMetricsAddOns/tree/master/xs03_json_example`.

User-Programmable Reports

User Programmable Reports are meant to be completely customized by the end-user. As such, they usually include a fixed set of columns that will be displayed, plus a set of facilities for creating new columns.

Those columns are defined through the "Parameters" field of the data block, where new columns are to be configured. For example, in order to add a column that displays the time an agent was available, you could enter a definition like:

```
Tot Avail [hms]= SESSION - (PAUSETIME + (TALKTIME + OOSCALL))
```

This definition is made up of three parts:

- A *column name*: how the column's title will appear in the reports;
- A *formatter*: how the resulting value is to be displayed. Formatting is expressed as a token between square brackets just before the equal sign;
- A *formula*: how the value is to be computed.

The *column name* is a free string; you may put anything you want in it.

The *formatter* must be one of the following values:

- *hms* - a number of seconds displayed as "[hours:]minutes:seconds"
- *%* - prints a float number into a percentage, es .365 is displayed as 36.5%
- *n* – prints an integer, right-aligned
- *f* – prints a float number

If no formatter is specified, the *percent* formatter is used. All numbers are printed according to the current rules set by your current language and locale (e.g. number of decimal digits, decimal separators, etc.)

The *formula* lets you insert an algebraic formula. This formula works on a set of bound variables (as specified below) that the block will replace with the correct values for each row.

- The following operators are supported, with natural precedence rules:
  - Addition: 2 + 2
  - Subtraction: 2 - 2
  - Multiplication: 2 * 2
  - Division: 2 / 2
  - Exponentation: 2 ^ 2
  - Unary Minus and Plus: +2 - (-2)
  - Modulo: 2 % 2
  - Rounded brackets are supported to force the order of evaluation. Use them freely.
- The following functions are supported:
  - abs: absolute value
  - acos: arc cosine
  - asin: arc sine
  - atan: arc tangent
  - cbrt: cubic root
  - ceil: nearest upper integer

    - cos: cosine
    - cosh: hyperbolic cosine
    - exp: exponent (e^x)
    - floor: nearest lower integer
    - log: logarithmus naturalis (base e)
    - log10: logarithm (base 10)
    - log2: logarithm (base 2)
    - sin: sine
    - sinh: hyperbolic sine
    - sqrt: square root
    - tan: tangent
    - tanh: hyperbolic tangent
- Specific data blocks may implement custom functions. Those are prefixed by the dollar sign and depend on the agent block.
- Any error in the formula (e.g. division by zero, incorrect formula, missing parentheses, referencing an unbound variable) will cause the formula to return zero. An error will be printed on the system log.

You can have multiple columns defined; it is also valid to have empty lines in the parameters block and comments, prefixed by "#".



When creating new formulas, it is often useful to print all referenced items by themselves, so you can double-check them, and comment them out when you don't need them anymore.

Variables bound for Agent blocks

The following variables are automatically bound and therefore usable in formulas:

- TALKTIME: The total conversation time
- SESSION: The total session time
- OVERLAPPING: The total overlapping conversation time
- OOSCALL: Out-of-session call time
- PAUSETIME: The total pause time
- PAUSEBILL: The total time of billable pauses
- PAUSENBILL: The total time of non-billable pauses
- PAUSEPAY: The total time of payable pauses
- PAUSENPAY: The total time of non-payable pauses
- NCALLS: The total number of calls for this agent

- NCALLSIN: The total number of inbound calls for this agent
- NCALLSOUT: The total number of outbound calls for this agent

The following custom functions are implemented:

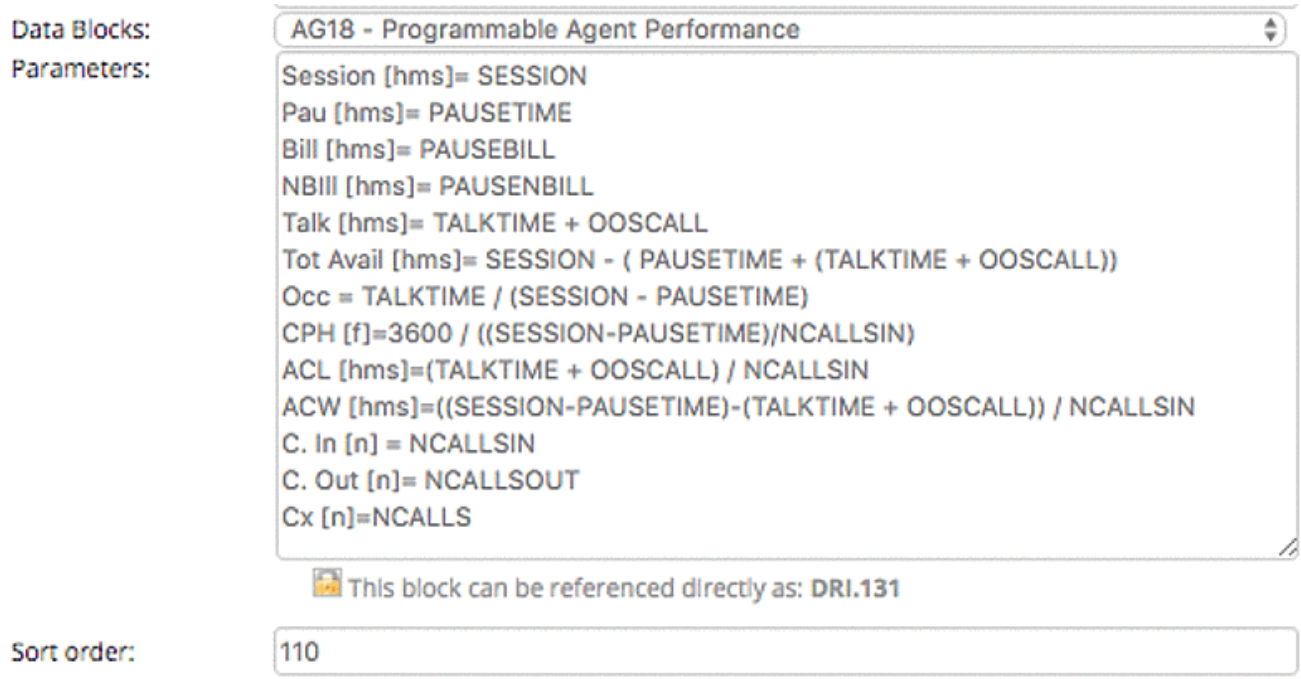
- $PAUSETIME[X]: returns the total duration of all pauses with the pause code "X" for the current agent

All times are expressed in seconds.

JSON and XML-RPC access

As Programmable blocks often do not include much interesting information by themselves, when referenced though their own block code will only return fixed columns.

It is still possible to access them as they are configured in a report by referencing their direct code; this will be displayed by the Reports editor as soon as you enter parameters for an existing data block. The format will usually be "DRI.num", where *num* depends on your QueueMetrics install (different QueueMetrics systems will report different DRI codes for a block defined in the same way).

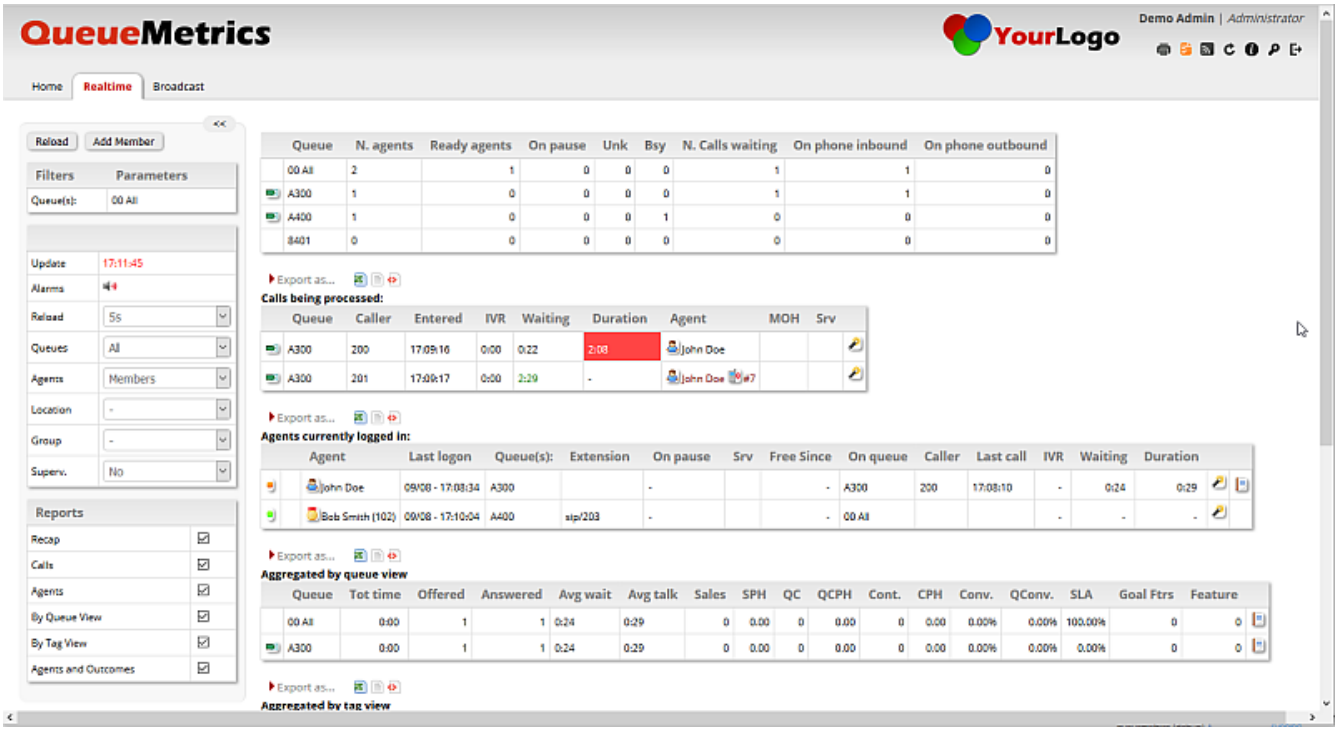




As accessing a data block that is a part of a report requires the accesing used to have the correct grants to access the report, screen and block, these security rules will be also enforced on "robot" users doing data queries. Make sure that, if you key-protect reports or elements in them, the "robot" user has all needed grants.

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:



On the top of the page there is a control table showing the last update timestamp and other dropdown selectors as specified below:

- **Reload**: It defines the update period will be used to refresh the shown data
- **Agents**: Shows or hides the *Agents currently logged in* table
- **Queues**: Shows all queues or only active queues
- **Agents**: Toggle between all agents or *members only* agents
- **Location**: If granted by user permissions, defines which location is shown
- **Group**: Defines which agent group is shown
- **Superv.**: Filter out agents not supervised by current user

The page is able to auto-refresh in background at the period specified in the first dropdown but you can anyway force a faster reload by clicking the "Reload" button.



If you enable filtering by Agents, Location or Group, statistics depending on lost calls (especially Lost attempts) may be undercounted in the view, as lost inbound calls will not be included in the data set to be considered, as they "belong" to no agent.

Next to the control table, there is the data section. Depending on the status of previously mentioned dropdown, three sections could be shown.

The first is a table showing a summary of all calls flowing through queues. Following that summary,

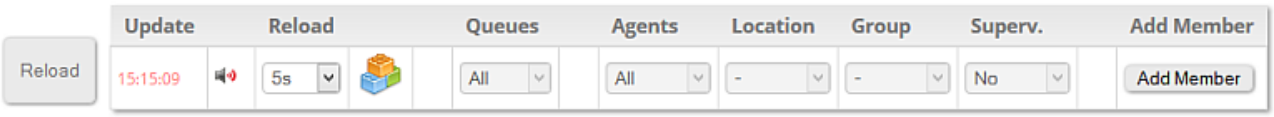
there is a table showing which calls are currently handled by the queue system, then the agents logged in at the moment.

This page is invaluable because 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 two calls and two 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 an yellow alarms for most numeric values that appear on screen - see the chapter Setting attention levels (Red and yellow alarms). You can also set sounds linked to yellow or red alarms, that will be played if a red or yellow alarm is present.

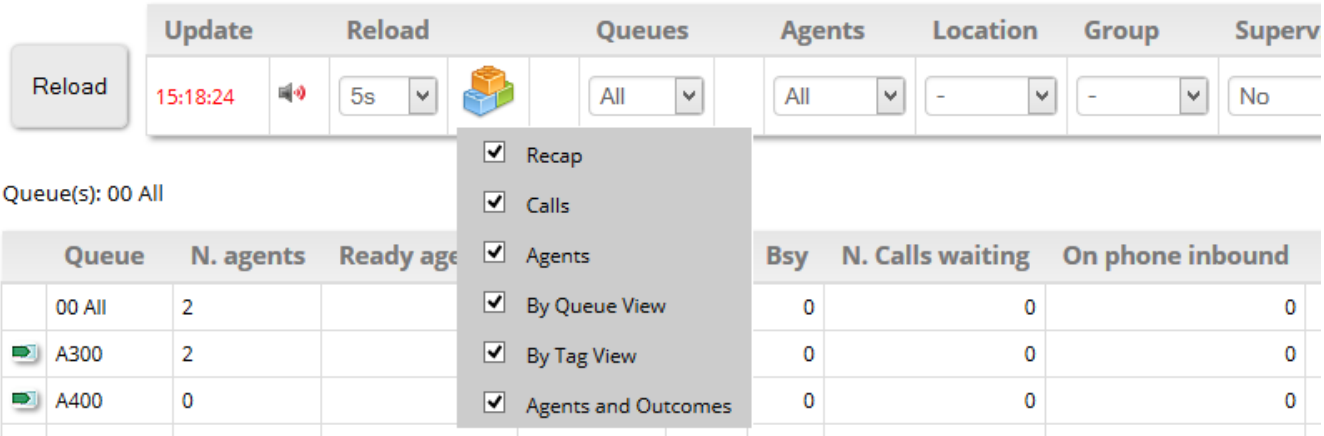
The "Add Member" button allows an administrator or supervisor to add an agent to a queue, as required. Users holding the RT_ADDMEMBER key will be able to add agents directly from the Realtime page.



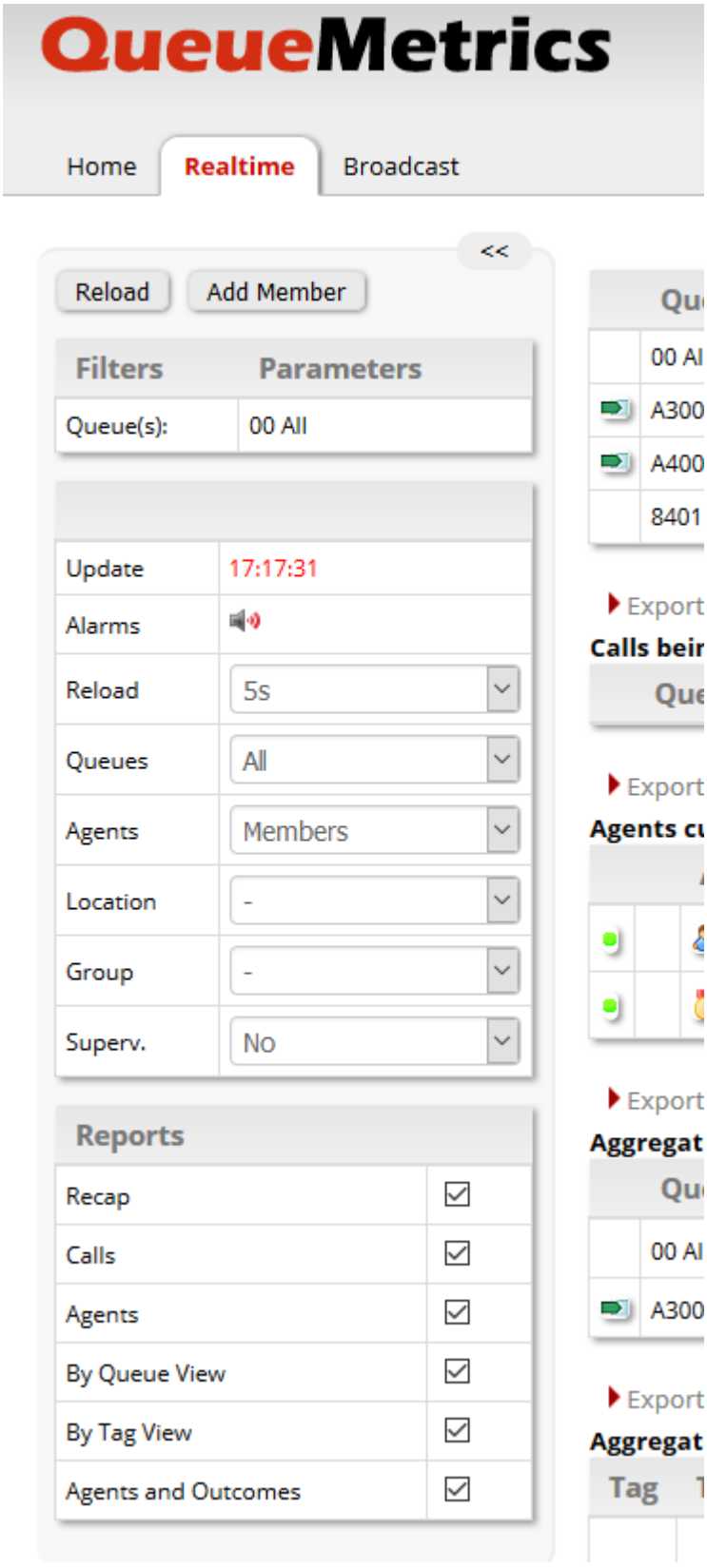
In order to change the default audio files, see "Appendix D: System Preferences". If you want to turn them off completely, just set them to blank.

The top menu bar shown in the above picture has been replaced by a left menu panel.

The top panel contains a *three bricks* icon. Flying over with the mouse allows the user to select the report blocks will be shown in the realtime page. The number of available blocks is dependent by a set of configuration keys.



Thre *three bricks* icon has been replaced by a checkbox list located on the new menu bar located on the left side, as reported below.



The menu bar can be optionally collapsed clicking on the << button to best fit realtime panel blocks.

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 "Members" option in the "Agents" dropdown is selected.

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. Depending on what type of information is present in the database (ATTEMPTS or RINGNOANSWER), is possible to have information about the last agent not picking up the call or the actual ringing agent's phone. The default configuration works with a standard Asterisk configuration and lets able to have RINGNOANSWER information. Please refer to Configuring Asterisk for QueueMetrics for the how to configure Asterisk generate ATTEMPTS information.

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, moving the mouse on the *wizard* icon at the end of

the line, some icons, like Call Monitor and VNC Monitor icons may be present. By clicking on one of these icons you activate the specified monitoring (see below).

If the call is ongoing and you have the special grants to do so, a red scissor icon might appear, moving the mouse on the *wizard* icon, to allow for brute-force call closure. See the section *Closing ongoing calls* for further details.

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

This panel can be turned on or off through the "Calls" dropdown on the top of the page.

The "MOH" field shows the amount of time a customer is on hold with music, during a call. If multiple HOLD instances took place during the call, this filed will show the total "on hold" time.

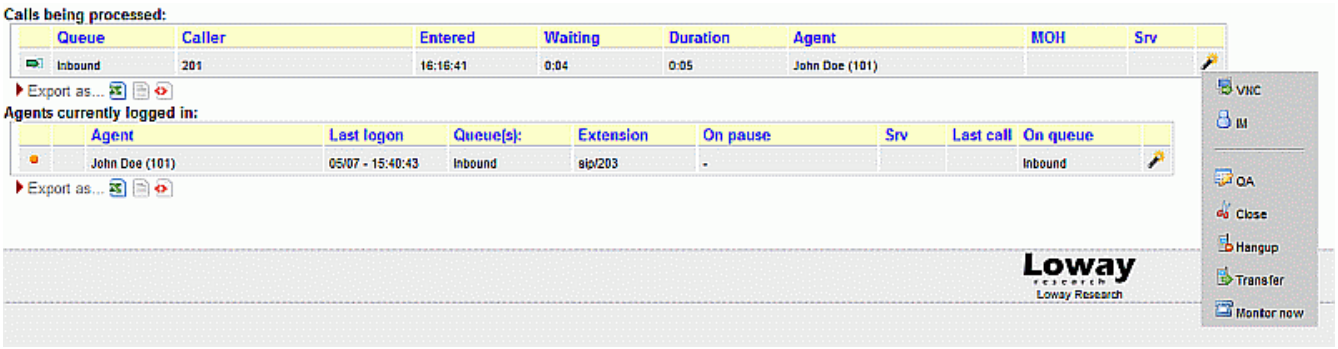




As Asterisk does not natively provide Music On Hold events on the queue_log, you need to run `uniloader` with MOH tracking enabled.

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

The last field contains a "wand" that on mouse-over displays a drop-down menu which allows to perform the following actions:



- *VNC*: Monitor agents via a VNC remote session
- *IM*: Begin an Instant Messaging session with an agent
- *QA*: Access the QA Form for the specific call (must have the QA_TRACK key enabled)
- *Close*: Close the call in the QM log, but not in Asterisk (must have the CLOSECALLS key enabled)
- *Hangup*: Close the live call in the PBX (must have the RT_HANGUPCALL key enabled)
- *Transfer*: Transfer the call to another extension (must have the RT_TRANSFERCALL key enabled).
- *Monitor now*: Start a listen-in *chanspy* Asterisk session (must have the MON_AUDIO key enabled)

To have precise realtime reporting a perfect clock synchronization is required. If your QM is on a

different server, make sure the clocks are exactly aligned or you may see strange values in all fields. The NTP protocol offers excellent clock synchronization precision and is available on most operating systems.

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

This panel can be turned on or off through the "Agents" dropdown 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 "All" option was selected in the "Agents" dropdown , all agents logged in on Asterisk will be shown, no matter to which queue they belong. When "Members" is selected, instead, only agents defined for the selected queue will be displayed.

By turning on the property *"realtime.show_incorrect_queue_sets"*, the Queues field will display each queue with a different color code:

- *in black*: queues the agent is a known member of
- *in brown*: queues the agent is currently working on but he's not a known member of
- *as a down-arrow*: queues the agent is a known member on but he's not currently logged on to. In order to see the list of queues, fly over the down-arrow symbol with your cursor.

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. A switch can be used to show the pause start time either as an absolute hour or a time increment.

The *Srv* field will contain the server identifier for QueueMetrics running in cluster mode.

The *Free Since* field will contain the length of period since the last taken call.

The *Last call* and *On queue* fields show the start 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.

The *IVR* field shows the call time spent on the IVR process before the call enter in the queue.

Waiting time and *Duration* fields are related to the last call taken.

The "wand" icon mouse-over displays a drop-down menu which allows to perform the following actions:



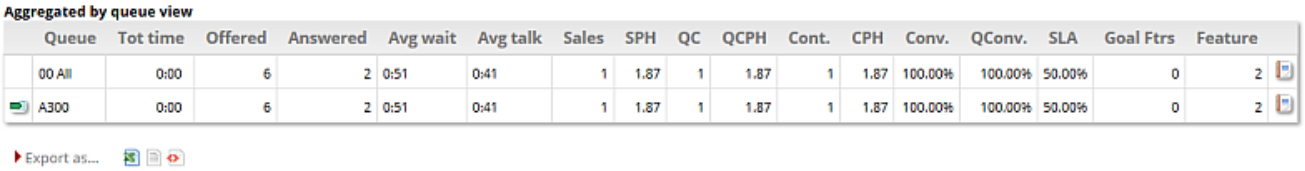
- *VNC*: Monitor agents via a VNC remote session
- *IM*: Begin an Instant Messaging session with an agent
- *Remove Member*: Remove an agent from a queue (must have the RT_REMOVEMEMBER key enabled)
- *Pause Agent*: Put an agent in a Pause state (must have the RT_PAUSEAGENT key enabled)
- *Unpause Agent*: Un-pause an agent so he/she is available to take calls (must have the RT_UNPAUSEAGENT key enabled)
- *Send Text Message*: Send a text message to an agent (must have the RT_SENDTEXTAGENT key enabled). The SMS functionality works only with asterisk 10+ versions.

The last icon is a *direct link to reports*. Clicking on this icon will run an historical report by populating the query with parameters similar to what generated the realtime panel results.

An optional field is available for the Agents Currently logged in table. This field is called *Paused For* and it shows the elapsed time of ongoing agent's pauses. To enable this field one needs to enable the realtime.relativePauseTimes System Preference. See Appendix C: System preferences.

A field is available, named "Since". This field is located right next to agent status colored icon. The number of seconds listed in this field represents the time that has elapsed since the agent entered it's current state.

Aggregated by queue view



Aggregated by queue view

	Queue	Tot time	Offered	Answered	Avg wait	Avg talk	Sales	SPH	QC	QCPH	Cont.	CPH	Conv.	QConv.	SLA	Goal Ftrs	Feature	
	00 All	0:00	6	2	0:51	0:41	1	1.87	1	1.87	1	1.87	100.00%	100.00%	50.00%	0	2	
	A300	0:00	6	2	0:51	0:41	1	1.87	1	1.87	1	1.87	100.00%	100.00%	50.00%	0	2	

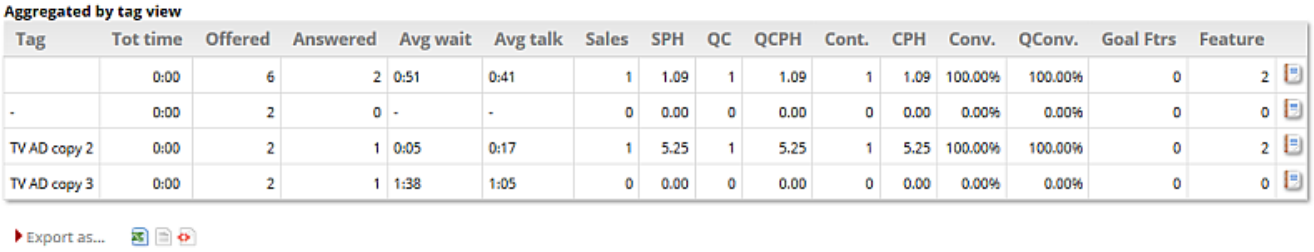
Export as...

This block reports realtime figures related to each selected queue. All values are aggregated by queue except for the first row where calls for the whole selected queue set is reported. On each column a user could find:

- An icon representing the queue direction (inbound/outbound)
- The queue name
- The total call time
- The number of calls sent to the queue
- The number of answered calls
- The number of lost attempts
- Average wait time for all calls in the queue
- Average talk time for answered calls
- Number of calls marked as sales
- SPH: Sales per hour figure
- Number of calls marked as qualified contact
- QCPH: Qualified contacts per hour figure
- Cont: number of calls marked as contact
- CPH: Contact per hour figure
- Conv: convertion ratio
- QConv: qualified conversion ratio
- SLA: percentage of calls matching the SLA time configured for the queue
- Goal Ftrs: number of goal marked feature codes found in the call set
- Feature: number of feature codes found in the call set

By clicking on the last icon, a user can start an historical report with query parameters matching the realtime page parameters that generated the selected row.

Aggregated by tag view



Aggregated by tag view

Tag	Tot time	Offered	Answered	Avg wait	Avg talk	Sales	SPH	QC	QCPH	Cont.	CPH	Conv.	QConv.	Goal Ftrs	Feature
	0:00	6	2	0:51	0:41	1	1.09	1	1.09	1	1.09	100.00%	100.00%	0	2
-	0:00	2	0	-	-	0	0.00	0	0.00	0	0.00	0.00%	0.00%	0	0
TV AD copy 2	0:00	2	1	0:05	0:17	1	5.25	1	5.25	1	5.25	100.00%	100.00%	0	2
TV AD copy 3	0:00	2	1	1:38	1:05	0	0.00	0	0.00	0	0.00	0.00%	0.00%	0	0

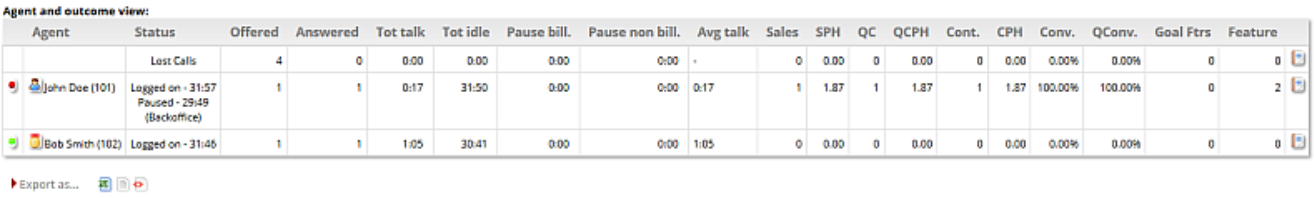
Export as...

This block reports realtime figures related to tags associated to processed calls. All values are aggregated by tag except for the first row, where calls for the whole tag set is reported, and the second row, where only calls with no tag are counted. On each column a user could find:

- Tag name (empty for all call set, dash for calls with no associated tags)
- The total call time
- The number of calls with associated tag
- The number of answered calls with associated tag
- The number of lost attempts for calls with a tag
- Average wait time for all calls with associated tag
- Average talk time for answered calls with associated tag
- Number of calls marked as sales with associated tag
- SPH: Sales per hour figure
- Number of calls marked as qualified contact with associated tag
- QCPH: Qualified contacts per hour figure
- Cont: number of calls marked as contact with associated tag
- CPH: Contact per hour figure
- Conv: convertion ratio
- QConv: qualified conversion ratio
- Goal Ftrs: number of goal marked feature codes found in the call set
- Feature: number of feature codes found in the call set

By clicking on the last icon, a user can start an historical report with query parameters matching the realtime page parameters that generated the selected row.

Agent and outcome view



Agent and outcome view:

Agent	Status	Offered	Answered	Tot talk	Tot idle	Pause bill.	Pause non bill.	Avg talk	Sales	SPH	QC	QCPH	Cont.	CPH	Conv.	QConv.	Goal Ftrs	Feature
	Lost Calls	4	0	0:00	0:00	0:00	0:00	-	0	0.00	0	0.00	0	0.00	0.00%	0.00%	0	0
John Doe (101)	Logged on - 31:57 Paused - 29:48 (Backoffice)	1	1	0:17	31:50	0:00	0:00	0:17	1	1.87	1	1.87	1	1.87	100.00%	100.00%	0	2
Bob Smith (102)	Logged on - 31:46	1	1	1:05	30:41	0:00	0:00	1:05	0	0.00	0	0.00	0	0.00	0.00%	0.00%	0	0

Export as...

This block reports realtime figures related agent sessions and calls. All values are aggregated agent except for the first row, where lost calls are shown. On each column a user could find:

- Agent status icon
- Agent name
- Agent status explained (Logged on time, pause status, pause time, pause code)
- Offered call number (that is: all calls received by the agent plus any outbound call they made)
- Answered call number (that is: all calls received by the agent plus any outbound call they made that was answered)
- Lost attempts to that agent
- Total talk time
- Total idle time (counted as session time purged from talk and pause time)
- Billable pause time
- Non billable pause time
- Average talk time
- Sales: number of calls markes as sale
- SPH: Sales per hour figure
- Number of calls marked as qualified contact
- QCPH: Qualified contacts per hour figure
- Cont: number of calls marked as contact
- CPH: Contact per hour figure
- Conv: convertion ratio
- QConv: qualified conversion ratio
- Goal Ftrs: number of goal marked feature codes found in the call set
- Feature: number of feature codes found in the call set

By clicking on the last icon, a user can start an historical report with query parameters matching the realtime page parameters that generated the selected row.

Using Locations

You can also assign each agent to al Location, i.e. a group of agents working together that you want to monitor as an 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.

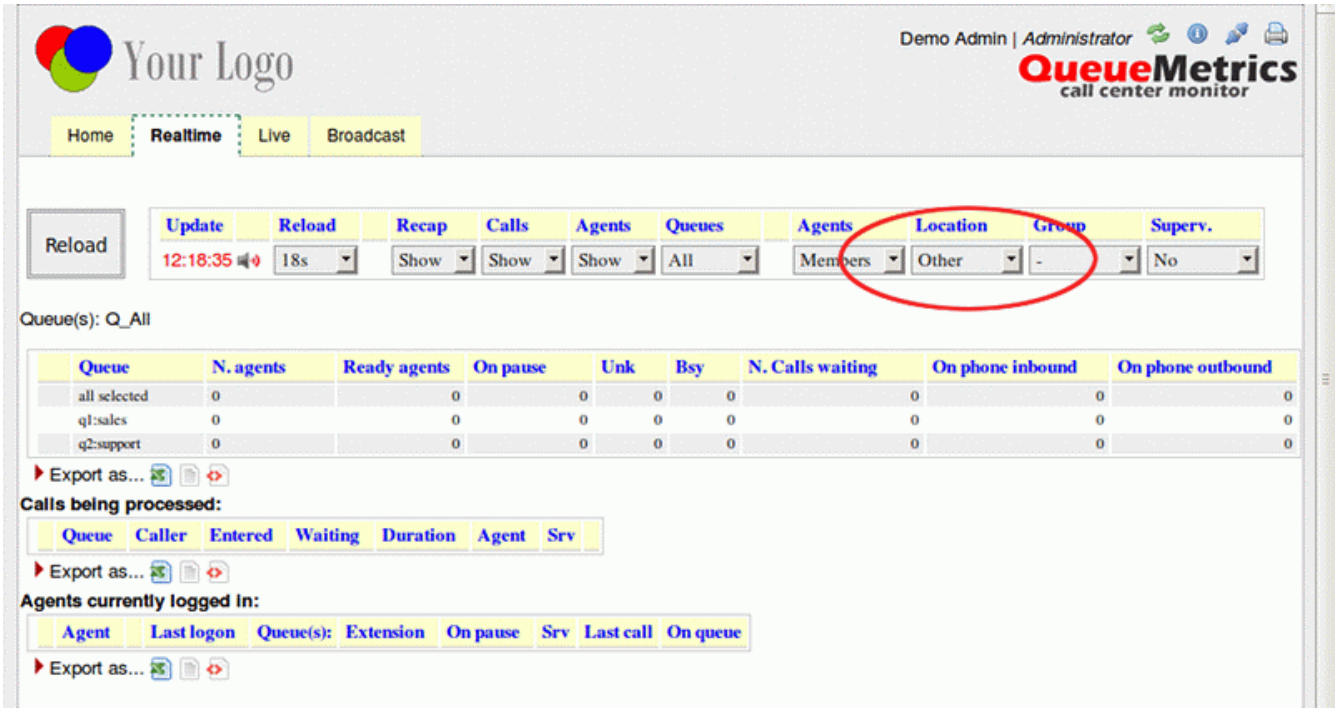
Note that when you filter by agents or locations you may see calls being queued and then disappear. This happens because all calls that wait to be answered on a queue are shown, but answered calls are shown only if the answering agents is a member of your defined filter conditions.

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.

It is also possible to have QM ask for a location to be selected in any case for Real-time reports. As locations can be key-protected, this is a handy way to determine which agents can be seen by which supervisors. In order to enable this feature, you must set the *realtime.assignedLocationsOnly* property. When this is set:

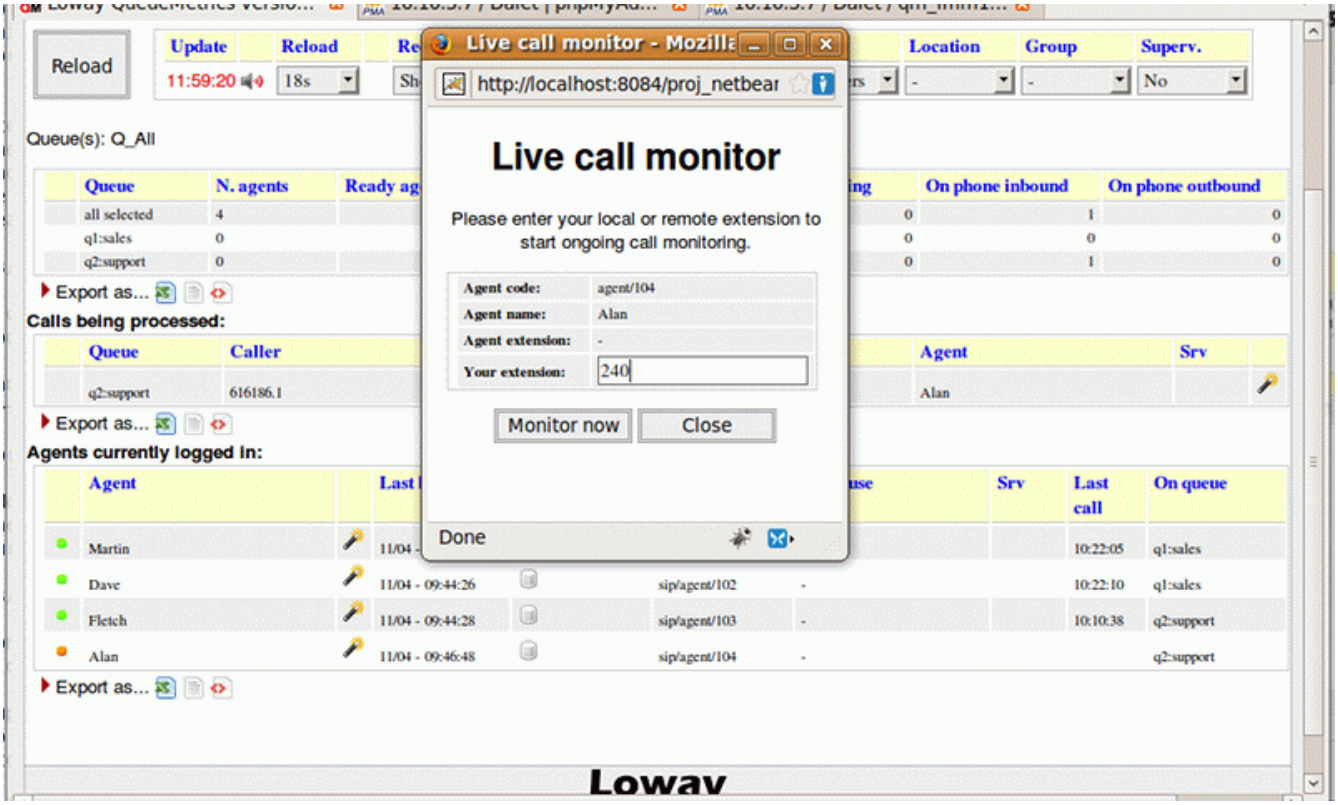
- When a user enters the real-time screen, the dash that means "Any location" is not shown anymore and the first location they have the grants to see is be selected.
- If a user has no grants to see at least one location, an error message is raised and they are not able to enter the real-time screen



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, that will appear moving the mouse under the *wizard* icon, a popup will be shown where you may enter your current extension or PSTN telephone number.



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 an the caller.

Depending on the kind of PBX platform you run, you may have one or more of the following options:

- *Spy*: listen to the call, without interacting with it
- *Whisper*: you can listen to the call and speak to the agent; the caller will not hear the conversation
- *Barge*: you can "break in" into the call and be on a conference with both parties

In order to set up this behaviour, please see the section Listening to live calls: Unattended Call Monitoring. It is possible to use different PMs to handle different audio needs - see Listening to calls using Pluggable Modules (PM).

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 of this manual.

Real-time performance considerations

Running a large set of real-time pages / agent pages / wallboards may impose a significant burden on a QueueMetrics system. But sometimes you have no choice. The best way to run them is to:

- Set the beginning of the analysis to the minimum usable time. For example, QM usually runs the

real-time report since the last midnight, but sometimes a sliding window of (say) 8 hours might be equally effective in your scenario and require a fraction of the load. You control this by setting the `realtime.startHour` property

- Tune Java memory for maximum performance - see the **Advanced Configuration** manual.

In order to further reduce the load above, there is a unified architecture (codenamed Presentialist) that aims to keep an up-to-date picture of the whole system and just send a snapshot to all interested parties. So, for example, an agent page and a real-time page sending in a status request at the same time would share the same underlying analysis.

This usually works seamlessly, but there is an important limitation - as all calls are processed no matter on which queue they appear to be on, if the same call is erroneously tracked multiple times (e.g. by a queue and by the Uniloader outbound tracker), the resulting logs happen to be unreadable. In order to bypass this issue, a new property `realtime.ignoredQueues` was introduced - it lets you blacklist a set of queues for display on real-time/agent-page/wallboard.



You can still run a report on any queue or queues you blacklist, but they won't be considered for real-time.

On older systems, you may also want to make sure that you are using the following defaults:

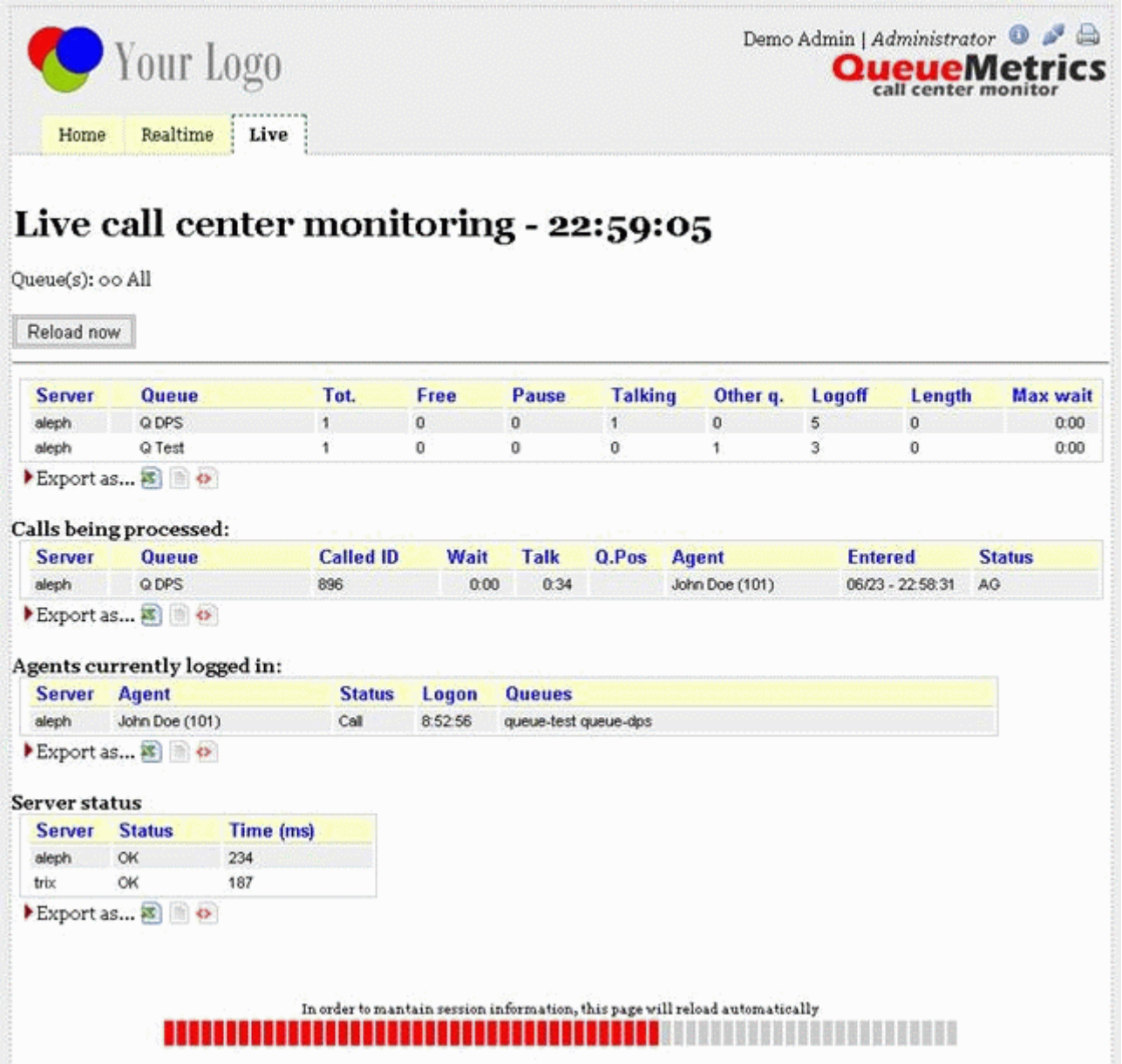
- Use SQL or CLUSTER storage types
- Turn on row caching - this allows QM to keep cached and pre-processed copies of objects it needs to run the Real-time and Agents Page. This may offer a significant speed up, ofter by an order of magnitude, and will reduce database load. You turn on this feature by setting *realtime.useRowCache* to true.

The real-time live page



This feature is deprecated.

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



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
- *Info*: contains all the Feature Codes, Variables, Call Tags, IVR choices if present.

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

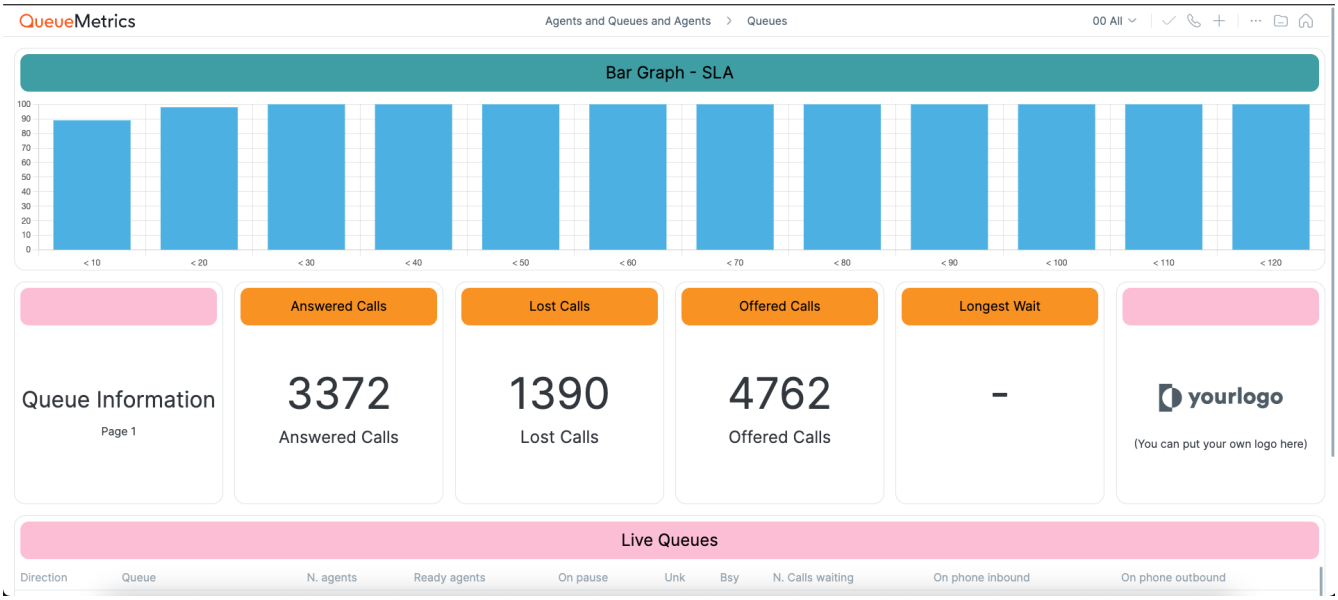
The last panel details the status of each server making up the cluster. If a server is not correctly set up, it will appear as KO.

- *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.

The Wallboard

QueueMetrics includes user-definable wallboard facilities that allow you to design your own wallboards and run them either interactively or as proper stand-alone wallboards.



Wallboards are built as a series of rotating **views**, each of which includes a number of **widgets**. Widgets are positioned as items on a square grid that is meant to fill the screen, and you can control the number of columns that you want to use.

Wallboards are meant to be customized - you can change colors, set alarms on specific widgets, include text and images and in general customize them to suit your specific needs. Wallboards can be private or public, in which case they can be shared between different users and can be used to run automated stations (see "Kiosk mode" below).

Understanding wallboards

A wallboard is controlled through the top bar.

The top bar:

- Displays the **current queue** or queues that are being used as data sources.
- Displays a number of circles, one for each **view**. By clicking on a widget, you can jump to a specific view. The current view is displayed as a dotted circle.
- Displays a **Play/Pause** button that lets you decide whether views rotate in order according to their expected durations or the current view remains visible. If you do any change to the page, rotation stops automatically until manually reactivated.
- Displays a **Load** button that lets you load other wallboards that are visible to you or create a new, blank wallboard.
- If the wallboard has unsaved data (e.g. because you adde some widgets or moved them around), a **Save** button apperas in the top bar that lets you save the current wallboard. Switching view when you have unsaved data will lose any unsaved data.

- There is a **Reorder** button that lets you change the sequence of views in the current wallboard and add/remove views.
- There is a share button that displays whether the wallboard you are looking at is **public or private**. By clicking on that icon within a public wallboard, you get the URL that you need to run the wallboard in kiosk mode copied to your clipboard.
- There is an **Add Widget** button that lets you add new widgets to the current view.
- There is a **title**, showing the name of the current wallboard and of the current view. By clicking on it, you can customize some properties for both wallboard and view.
- There is an **Filter** button that lets you add filters to all the wallboard view
- There is an **Exit** button that gets you back to your QueueMetrics session.



In order to access the wallboard, you need the security key *REALTIME*. To edit a wallboard you need the key *WALLBOARD_ADD* and to create a public wallboard you need the key *WALLBOARD_FORALL*. You can easily add those keys though the User editor or by adding them to a security class.

Saved Wallboards

The default wallboard is accesible though the link "Wallboard" on the home page of QueueMetrics, using the current queue(s) selected. From the wallboard itself you can load other wallbords though the **Load** button.

Your wallboards, and all public wallboards visible to you, will appear on the home page in the "Wallboards" section. By clicking on them, they will open with the current selected queue. You can also delete existing wallboards by clicking on the "Remove" icon hovering over their home page link.

Settings

By clicking on the title of the wallboard, you can edit the following properties:

For the whole wallboard:

- **Name** that the wallboards appears under. You can have multiple wallboards with the same name
- **Comment** is a free text comment you may want to save
- **Public** controls whether the wallboard appears on every user's home page (as restricted by the security key) or is private for you
- **Security key** lets you protect this wallboard from users who do not hold the specified key

For the current view:

- **Name** is the name of the current view. Multiple views within a wallbord may have the same name, though it would be better to use different names
- **Duration** is the duration in seconds that a wallbard is displayed on the screen before the next

wallboard. Data is updated live when a wallboard is being displayed.

- **Number of Columns** is the size of the grid that the screen is horizontally split into. This is the minimum size for a widget, though you can stretch a widget to use multiple grid cells as you best see fit.
- **Filters** allow you to add or edit data filters that apply only to the current view.

Editing Widgets

Widgets can be added, moved, cloned or removed freely within a view. You should position a widget in a way that makes sense for your use case, giving it enough room to display its data in a way that is easy to see.

Most widgets will resize automatically in the space they are given to offer a good readability and to make full use of the space they are assigned.

If you can edit a widget, when you fly over it with your mouse it will display an **Edit widget** icon and it will go in editor mode. Though the editor you can:

- Clone the widget
- Remove the widget
- Change the title
- ...and edit any special configuration it has (see below for a description of all available widgets)

Most of the widgets can be filtered by queue. If you are using the wallboard to monitor multiple queues at the same time, you can use the queue filtering function to apply the widget to a single queue instead of applying it to the aggregation of all the queues data.

All the widgets that can be filtered by queue, can be filtered by multiple queues, instead of a single one at a time.

Starting with QueueMetrics 26.01.1, a variety of filters can be applied to all widgets, with the exception of Live Queues.

When you save a widget, the whole wallboard is saved automatically.

Alarms

Most widgets have a set of possible **custom alarms** that you can use to display them in a different color when an alarm condition is met. Custom alarms are expressed as simple expressions, e.g. ">10" or "<15".

If QueueMetrics has a system alarm set on the queue shown and no custom alarm set, then the correct system alarm is used. Custom alarms always take precedence on system alarms.

Running in kiosk mode

If you want to run a wallboard in kiosk mode, the first thing that you need is to get its kiosk URL. You can obtain this by clicking on the share button. The URL might appear like the one below, for

queues 300 and 301:

```
http://127.0.0.1:8080/QM/qm_wab2.do?user=robot&pass=***
    &queues=300%7C301&wallboardId=21
```

As you can see, you can edit which (atomic) queues you want to use for your wallboard by separating them with an URL-encoded pipe "|" character, that is `%7C`.



The URL will always include a user called *robot* with password ***; edit it as necessary given the current settings of your "robot" user. Make sure that the *robot* user is enabled, as it is usually present but disabled on new QueueMetrics installs.

If you connect a small-cost Linux box (e.g. a Raspberry PI or similar) to a large screen or a video beamer and set it in your call-center where it will be visible by your agents, you can 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 at night.

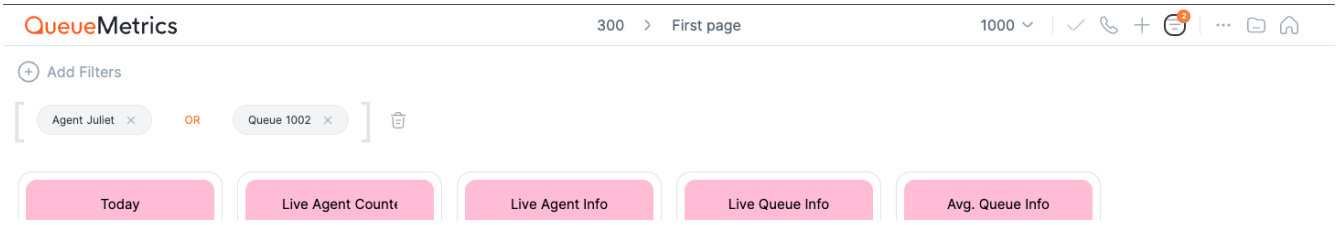
The Kiosk mode requires a user with the `REALTIME` and `ROBOT` grants - they are already set with the default `ROBOT` user.

Wallboard filters

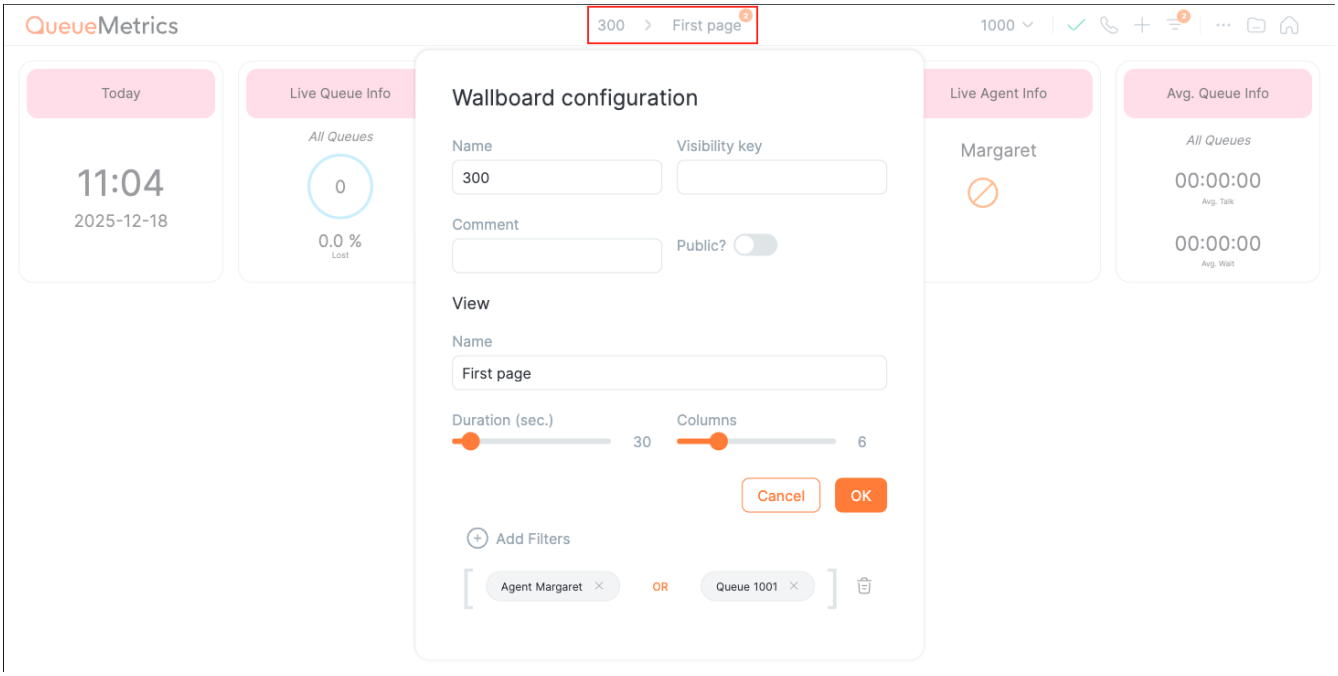
Starting from QueueMetrics 26.01.1, it is possible to add a vast variety of filters as already present in the report.

The wallboard filters are divided in three level:

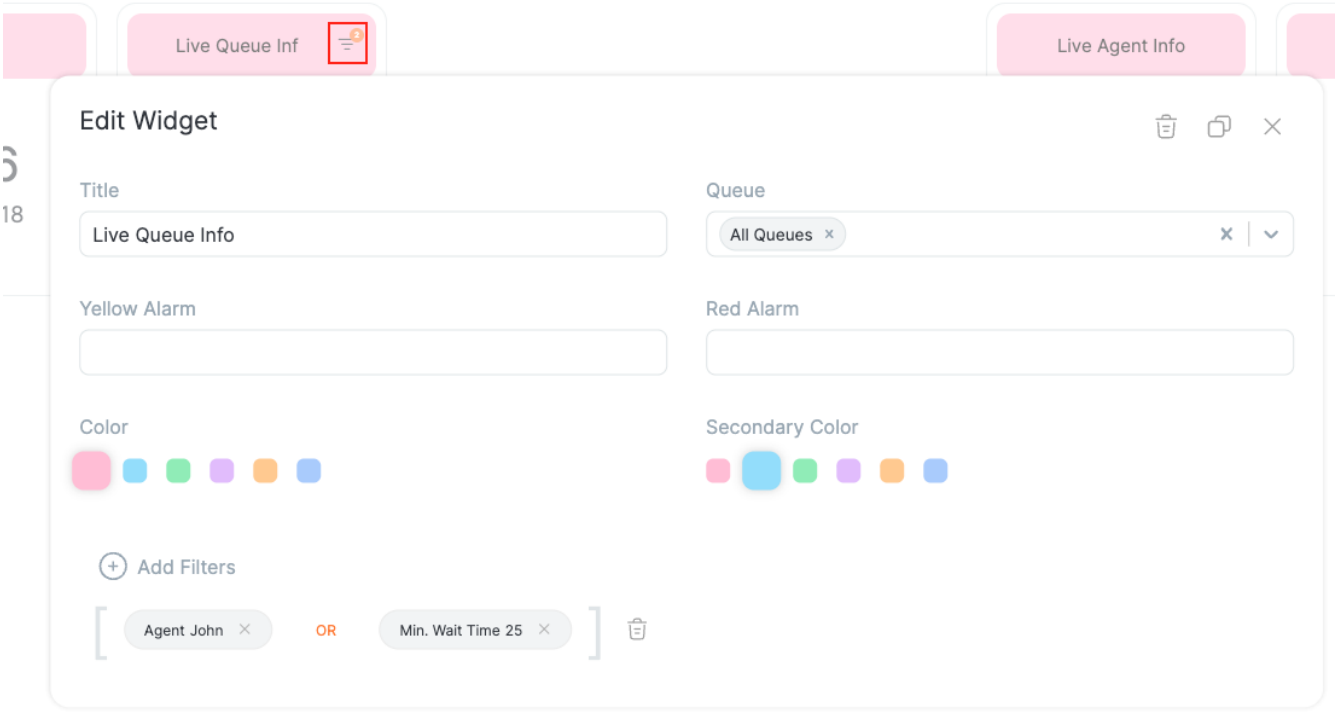
- **General Wallboard**: These filters affect all views and widgets and are accessible via the top bar button.



- **View Level**: These filters affect only the widgets within the current view and are managed through the **View Settings**.



- **Widget Level**: These filters apply exclusively to the selected widget and are accessible via the individual **Widget Settings**.



Understanding Filter Logic

It is important to understand how filters interact across these three levels. Filters are combined using a logical "AND" condition. This means that if you apply filters at all three levels, you must ensure they do not create a null set (no data).

For example, if you filter for Agent X at the general level, Agent Y at the view level, and Agent Z at the widget level, no data will be displayed because a single record cannot satisfy all three criteria simultaneously.

Best Practice: Apply broader filters at the wallboard level and become increasingly specific as you

move toward the widget level.

```
Example: Apply a filter for Agents X, Y, and Z at the wallboard level, narrow it down
to Agents X and Y at the view level, and finally isolate Agent X at the widget level.
```



When filters are applied, a numerical badge appears over the filter icon to indicate the number of active selections.

Wallboard actions

The wallboard supports agent actions and call actions.

To access agent and call actions, you need to use the contextual menu to select one of the two:

- Agent actions
- Call actions

Agent actions

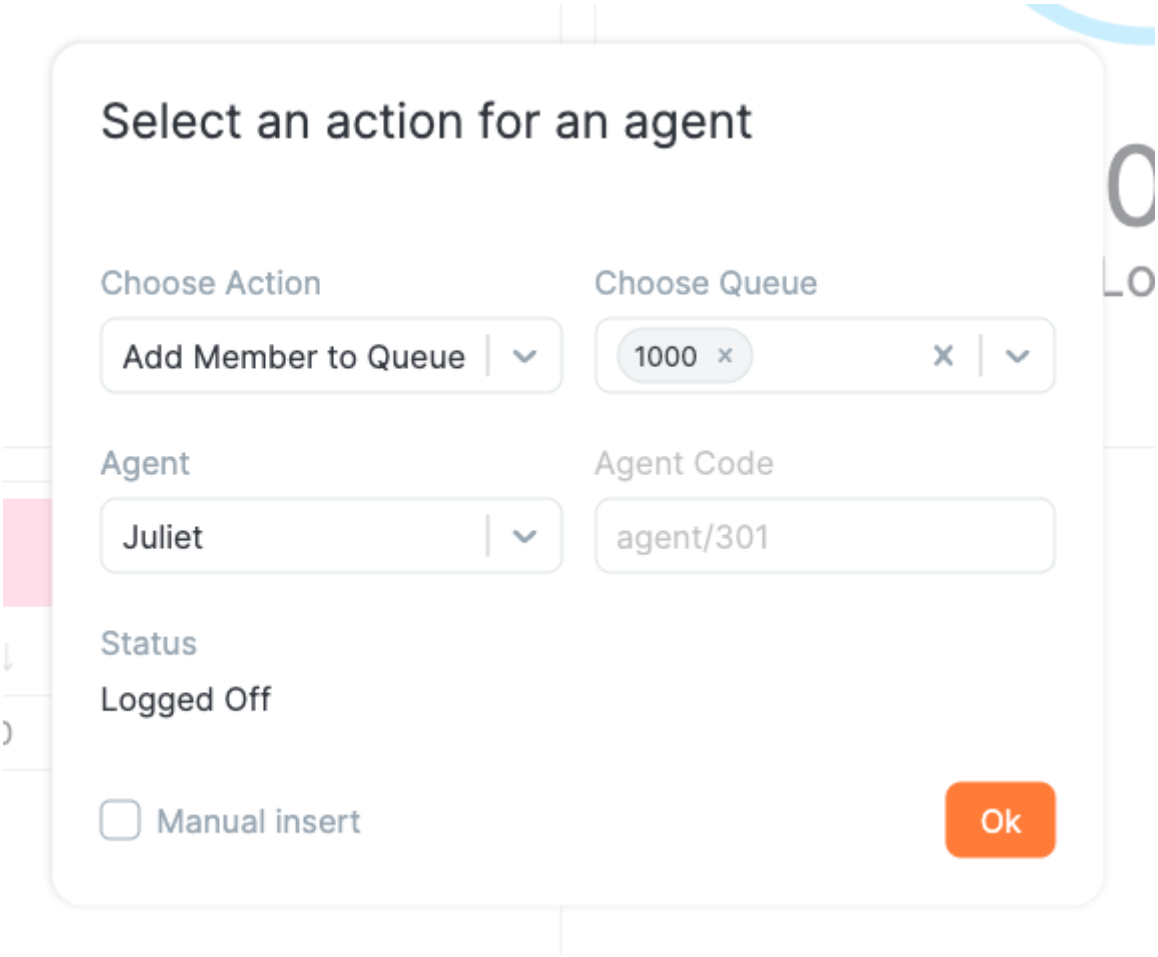
By selecting the Agent actions option, you will open the **Agent action panel**.

In this panel you can select one of the following actions:

- Add member to queue
- Remove member from queue
- Pause agent
- Unpause agent
- Change pause

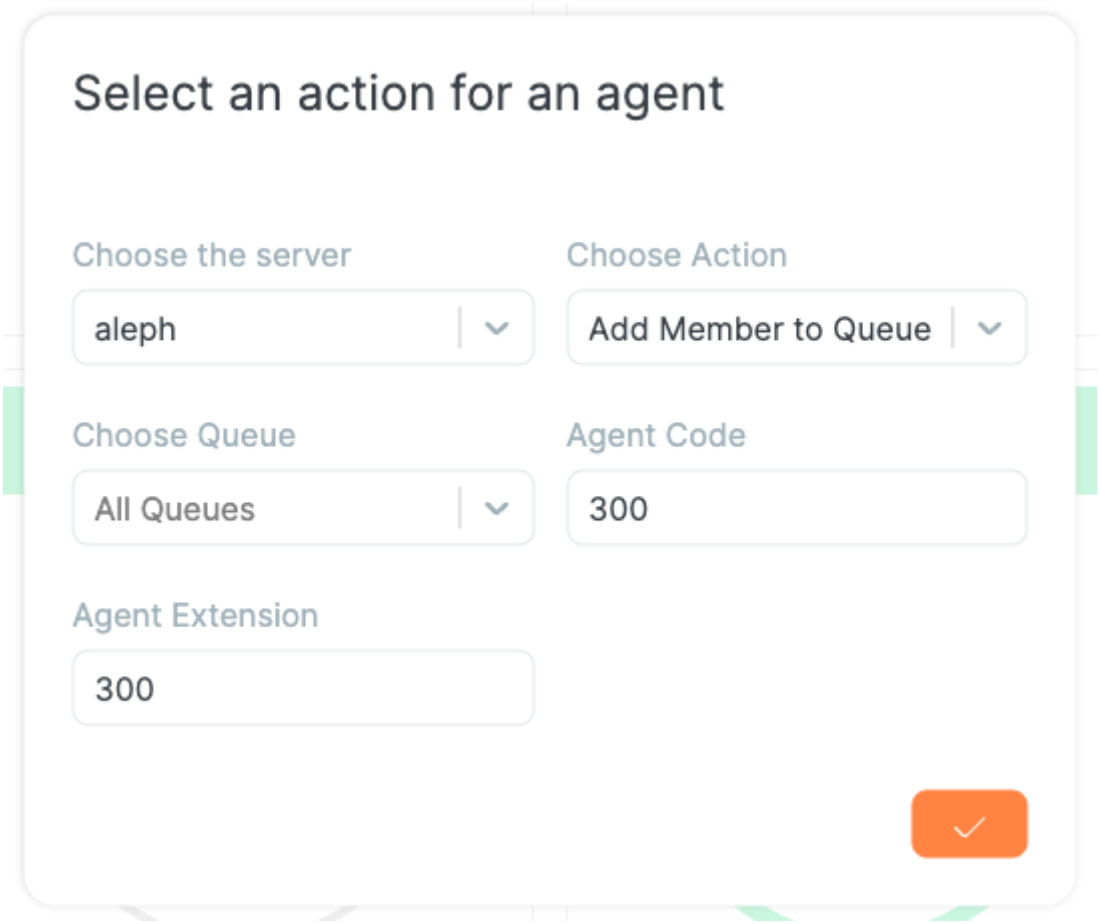
You also need to specify different parameters for each action, that will show in the rest of the panel:

- Queue (if needed)
- Agent code
- Agent extension (if needed)
- Server (if needed)
- Pause code (if needed)



Starting from QueueMetrics 26.01.1, you can either select an agent from the dropdown menu, which will automatically fill in the required fields. Based on the agent's current status, different actions will be available, for example: If the agent is not logged in, you can only add the agent as a member, otherwise if the agent is logged in, you can pause or remove the agent, in this case, all queues in which QueueMetrics detects the agent as logged in will be preselected.

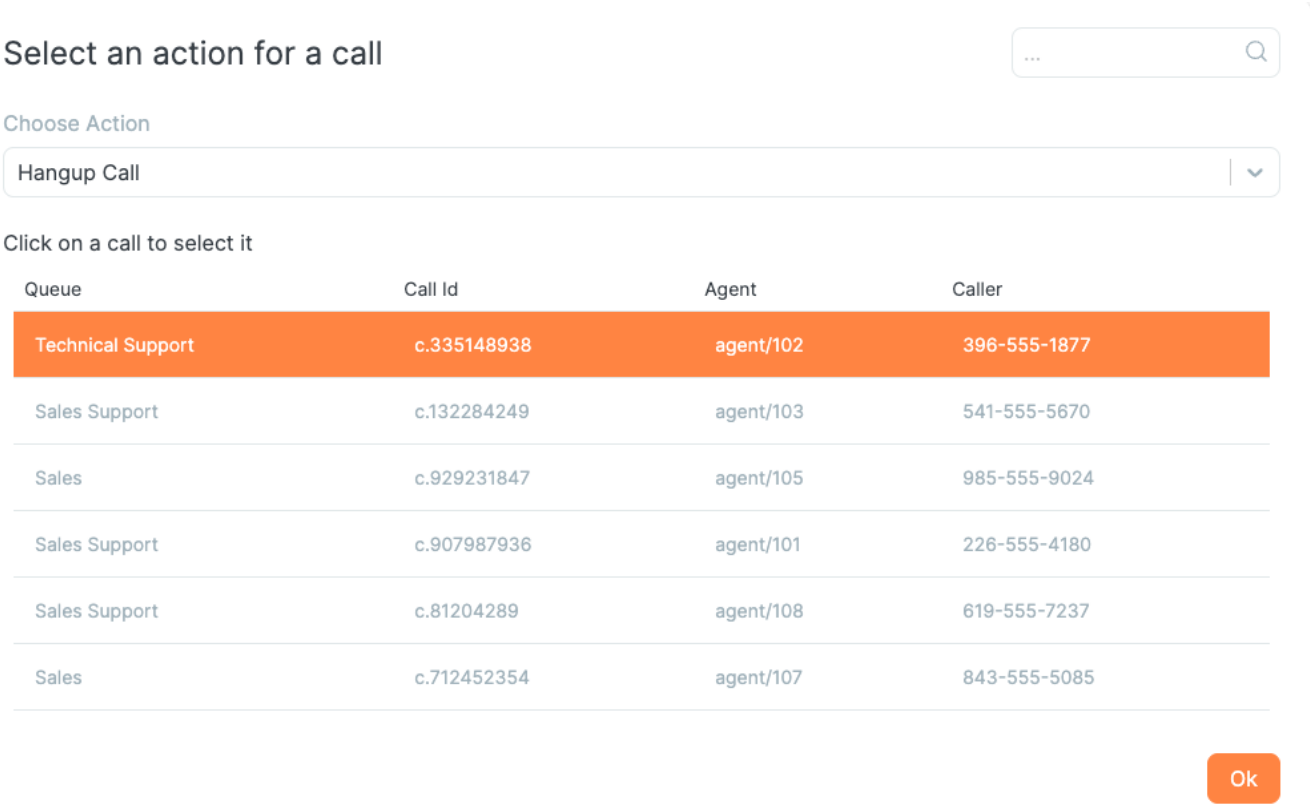
If, for any reason, the agent cannot be found or the detected status is incorrect, you can manually enter the data by selecting the **Manual Insert** checkbox.



Starting from QueueMetrics 26.01.1, clicking an agent's name or status in the Live Agents widget opens the agent action panel with the selected agent already preselected.

Call actions

By selecting the call actions option, you will open the **Call action panel**.



In this panel you can select one of the following actions:

- Hangup call
- Monitor call
- Transfer call

A table will also appear in the panel, showing the current calls. To select the call you want to execute the action upon, click on the desired call in the table and it will highlight.

You also need to specify different parameters for each action, that will show in the rest of the panel:

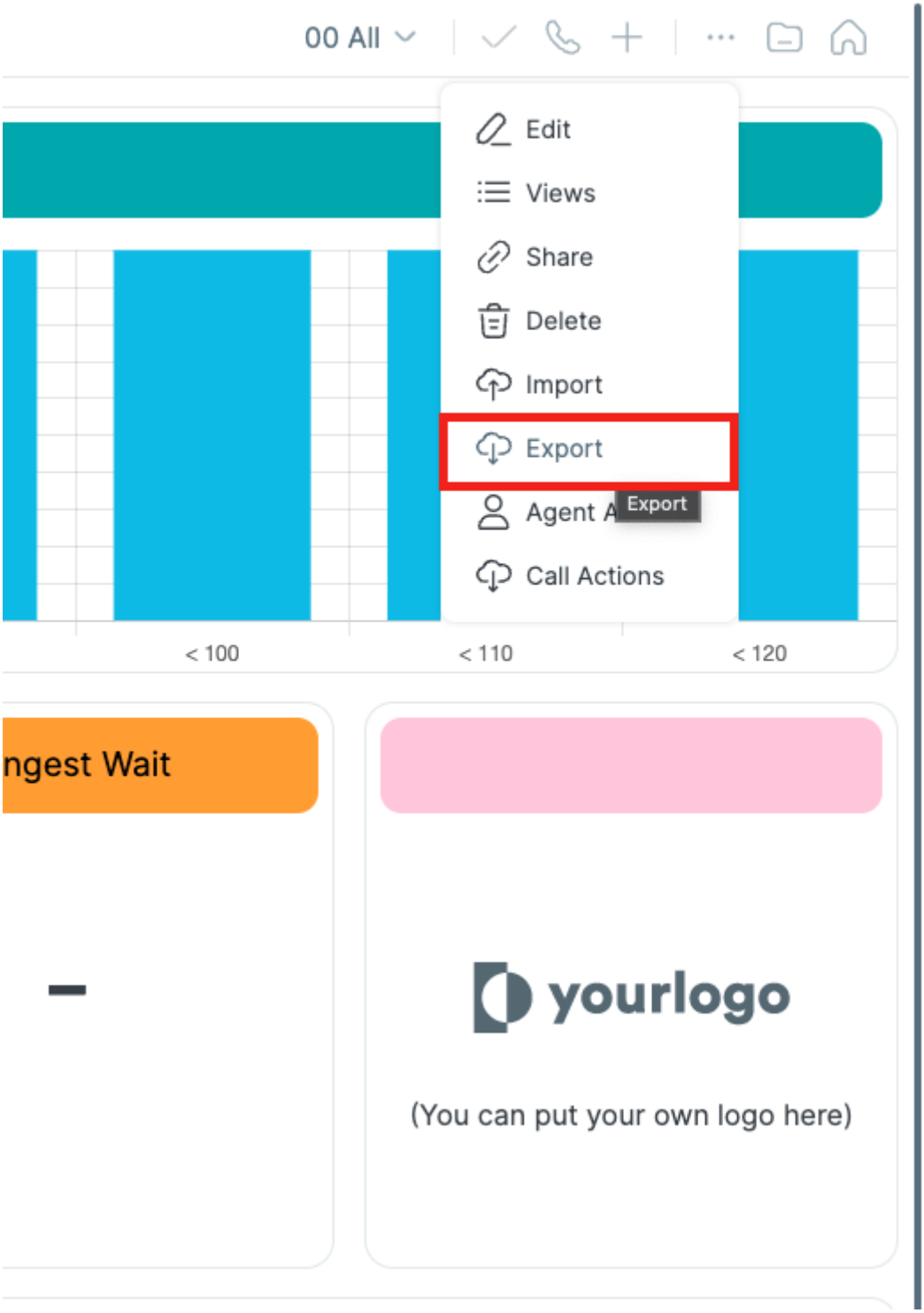
- Server (if needed)
- Ext. to transfer to (if needed)
- Local Extension (if needed)
- Monitoring mode (if needed)

Importing and Exporting Wallboards

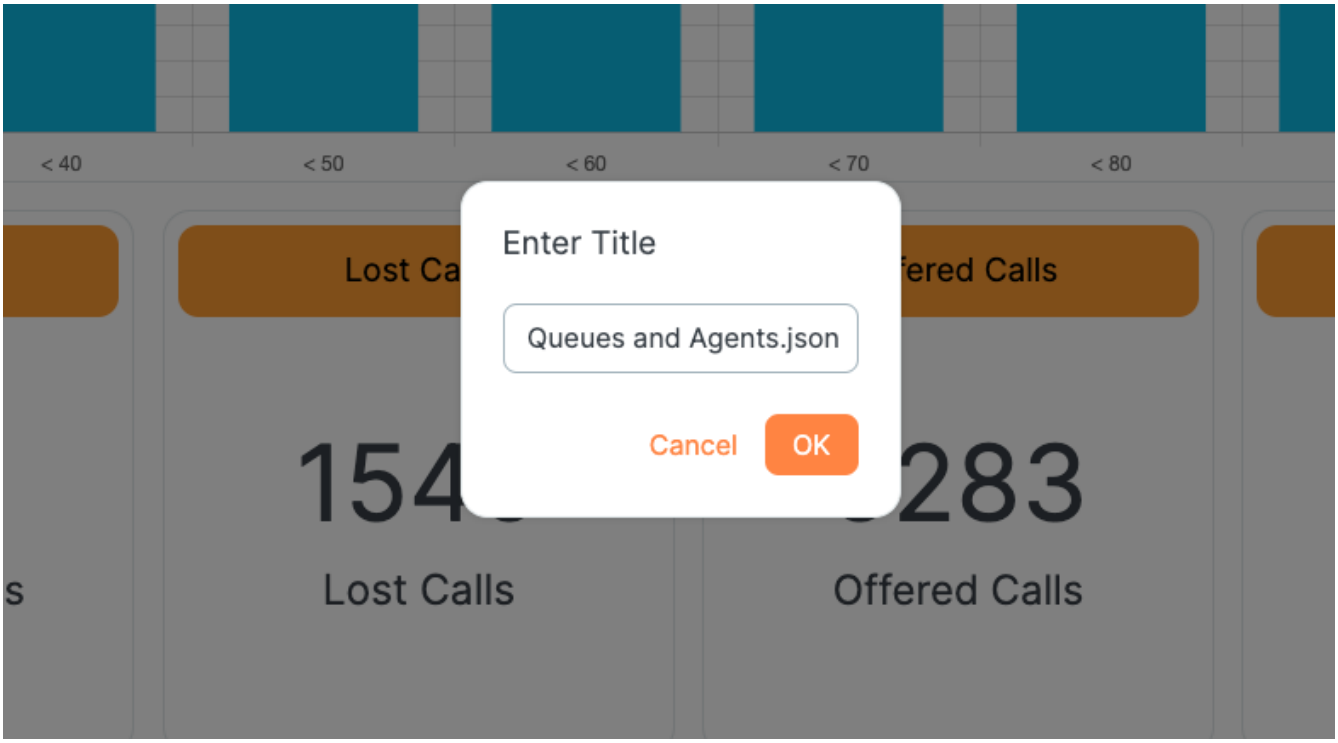
The wallboard allows users to export and import wallboards as JSON files that can be shared between different systems and users.

Exporting a Wallboard

To export a Wallboard, move your mouse over the top bar menu as shown in the screenshot, and click on "Export".



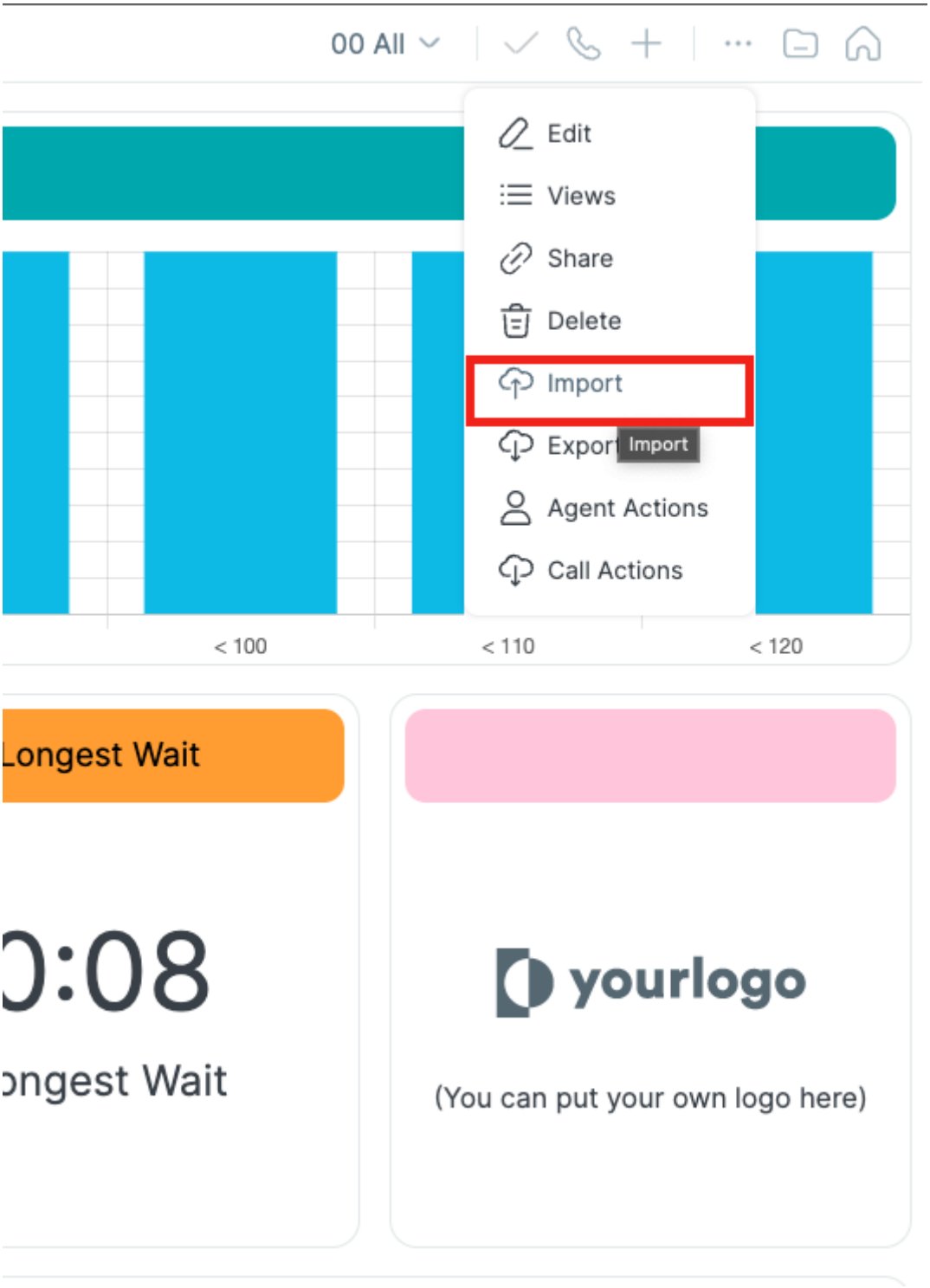
This should open the Wallboard Export panel, where you can pick a file name for the exported Wallboard.



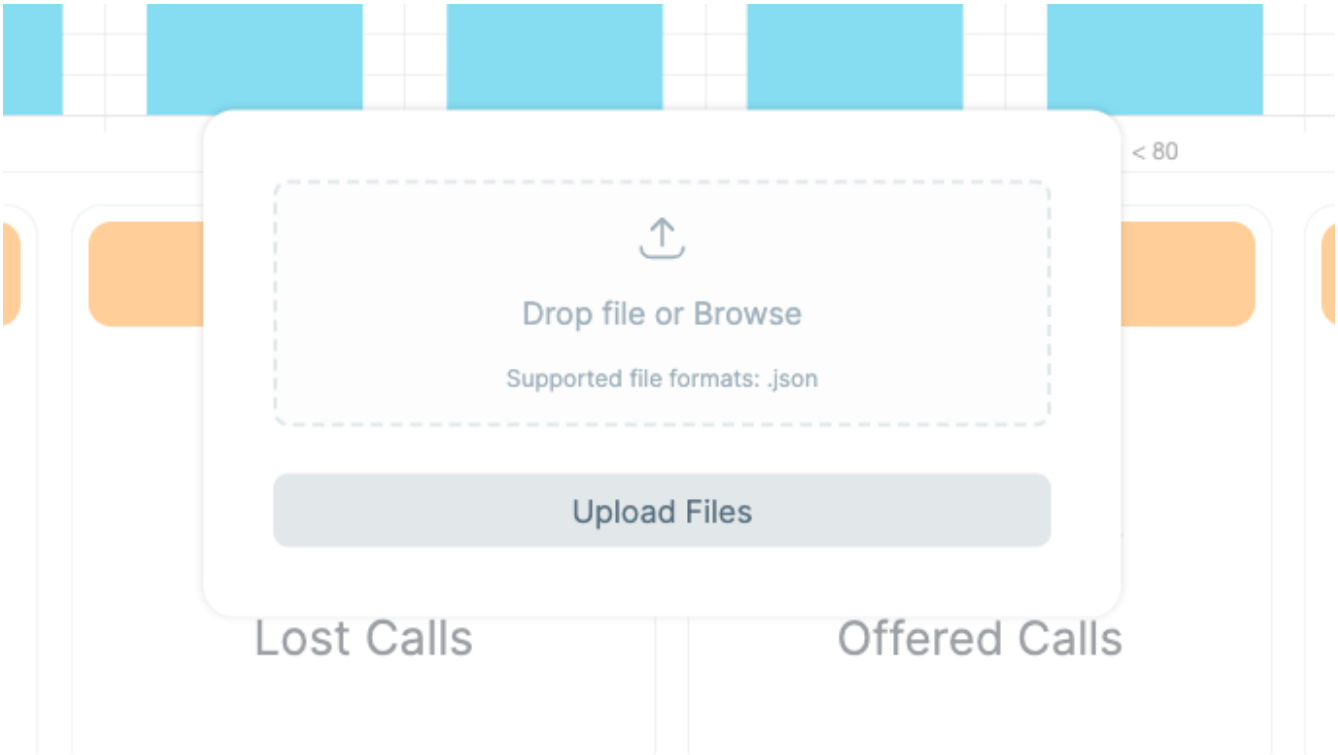
Once you have selected a name, click on "OK". This will export the Wallboard as a JSON file that can be imported by any QueueMetrics system (22.11 or newer).

Importing a Wallboard

To import a Wallboard, move your mouse over the top bar menu as shown in the screenshot, and click on "Import".



This should open the Wallboard Import panel, where you can drag a JSON Wallboard file. You can also click on the drop zone to open your system's file explorer and pick a file from there.



Once you have selected the file, click on the confirm button in the bottom. If the imported file is a valid QueueMetrics Wallboard, it will be saved on your system and loaded immediately.



if the wallboard contains some custom DataBlocks that are not present on the importing QueueMetrics instance, QueueMetrics is going to follow this process for each one:

A) Look for a Custom DataBlock with the same ID. If it exists, go to step B. If not, go to step C.

B) Checks if the existing DataBlock is the same as the one from the other system If so, it keeps it, if not, go to step C.

C) Creates a new DataBlock with the same properties as the one on the old system. If the ID is already in use, it creates a new one.

Wallboard softphone

The wallboard offers an integrated **WebRTC softphone**.

To enable the wallboard softphone for a user, the user must be able to access the wallboard and must have the `WALLBOARD_PHONE` security key.

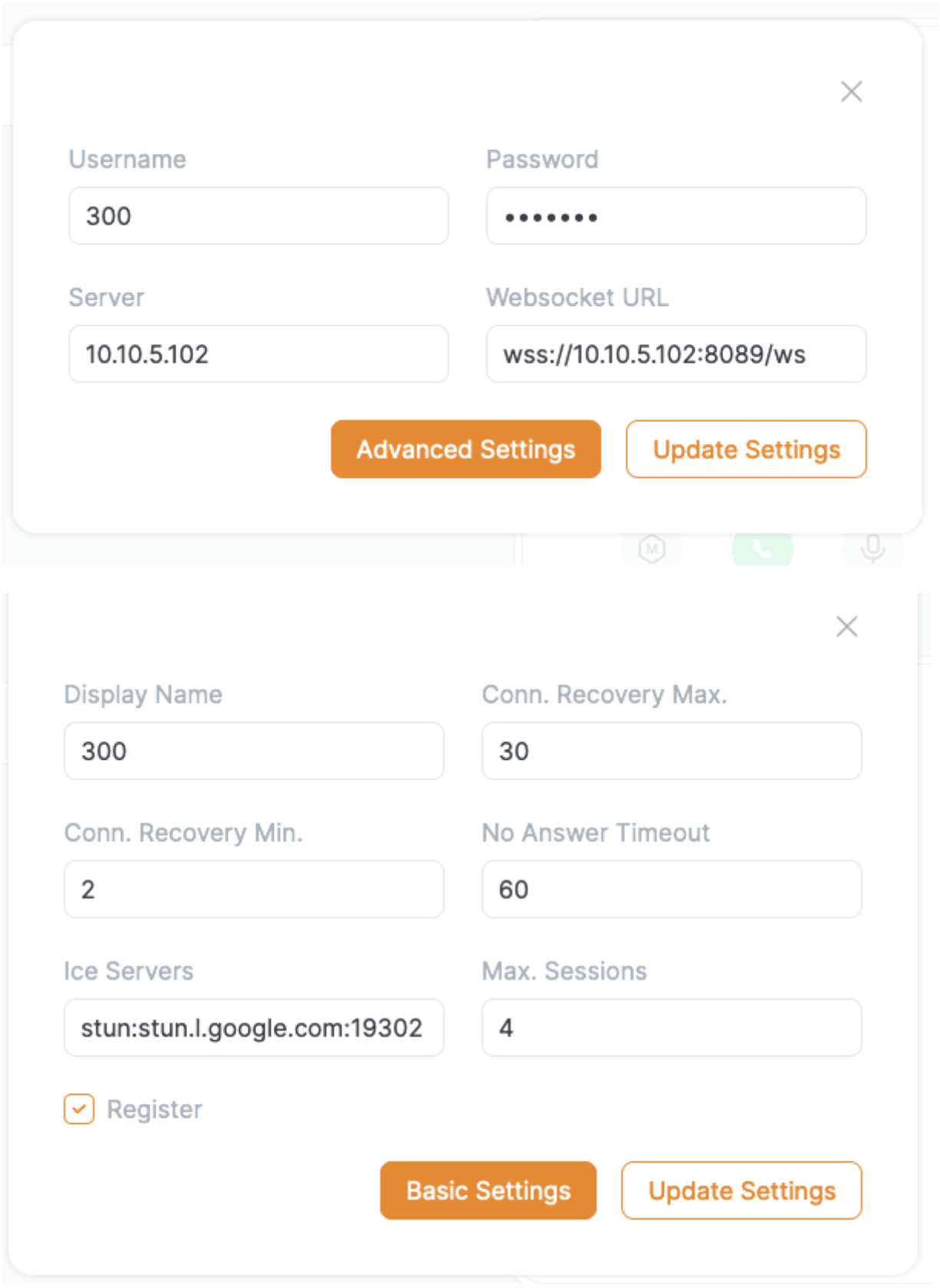
This softphone is configurable through:

- The **Edit System Parameters** page
- The **Explore System Parameters** page
- The **Softphone Settings Panel** inside the wallboard

System parameters act as default values, while the **Softphone Settings Panel** provides user-specific

settings that override defaults and persist across sessions. For personalized configuration, always use the Wallboard's **Softphone Settings Panel** unless you intend to change defaults globally.

The **Softphone Settings Panel** is accessible via the three-dot icon in the wallboard's top bar.



The following settings are available in the configuration panel:

- **Username**: SIP username for the extension

- **Password**: SIP password for the user
- **Server**: PBX server address
- **WebSocket URL**: WebSocket address used by the phone for registration and signaling
- **Display Name**: Caller ID displayed when this extension places a call
- **Connection Recovery Max Interval**: Maximum delay before retrying connection recovery
- **Connection Recovery Min Interval**: Minimum delay before retrying connection recovery
- **No Answer Timeout**: Timeout in seconds before an unanswered call is automatically ended
- **Register**: Whether the softphone should register to the PBX (`yes` or `no`)

System Parameters Defaults

The same settings can be defined as defaults in the **Edit System Parameters** page:

```
default.wallboardphone.connection_recovery_max_interval=30
default.wallboardphone.connection_recovery_min_interval=2
default.wallboardphone.display_name=200
default.wallboardphone.no_answer_timeout=60
default.wallboardphone.password=password200
default.wallboardphone.username=200
default.wallboardphone.server=10.10.5.181
default.wallboardphone.websocketurl=wss://10.10.5.181:8089/ws
```



An additional system-wide-only setting is:

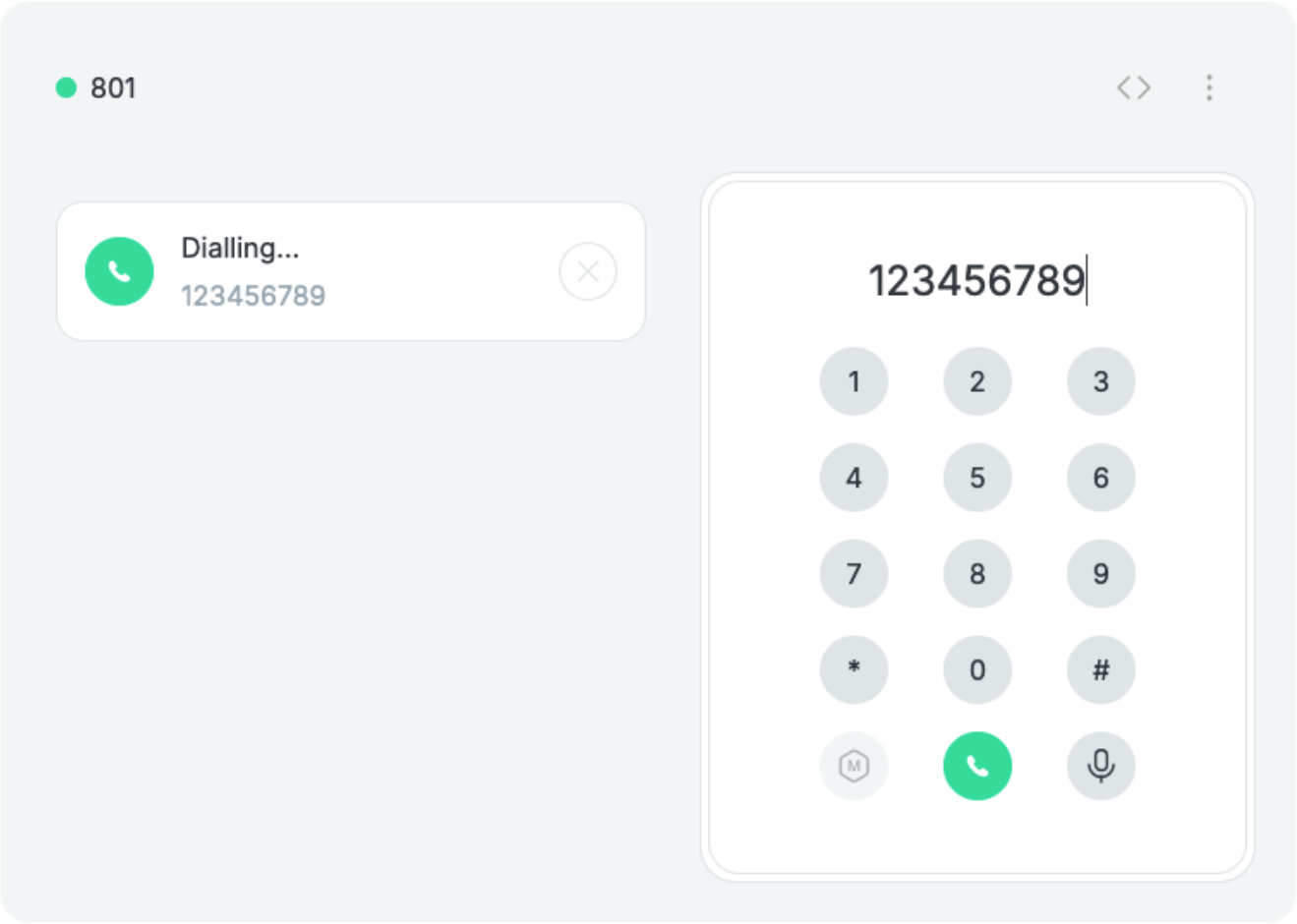
`default.wallboardphone.iceservers=stun:stun01.sipphone.com,stun:stun01.sipphone.com,stun:stun.fwdnet.net`

Use this to specify STUN/TURN servers for WebRTC media negotiation.

The Softphone Panel

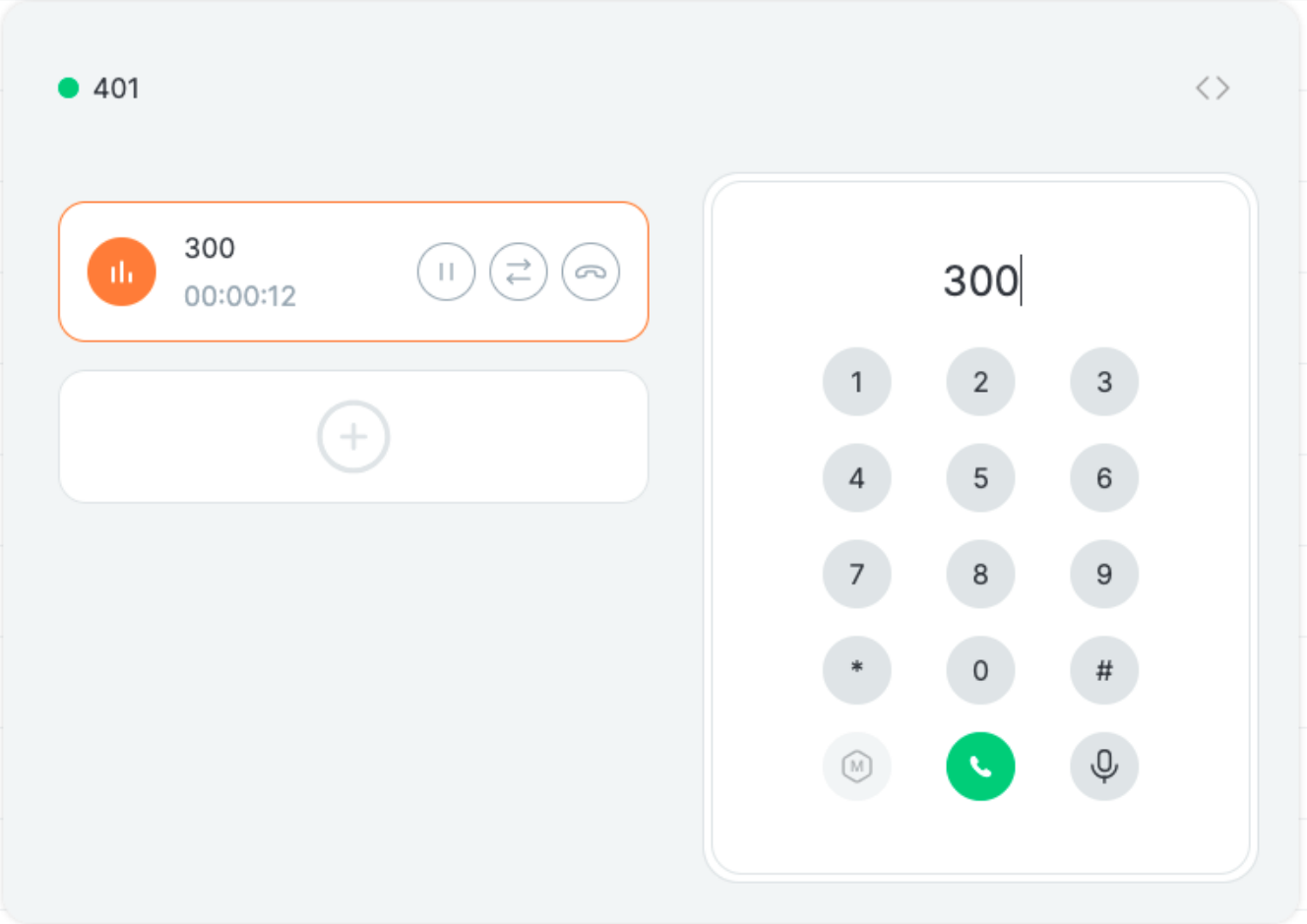
To open the actual **Softphone Panel**, click on the **Phone** icon in the top bar.

Upon opening, the phone will attempt to register to the PBX unless already registered.



From this panel, users can:

- Dial a number
- Start a call
- Pause a call
- Transfer a call
- End a call



Call Transfer

Pressing the **Transfer** button opens a panel to select an extension and choose:

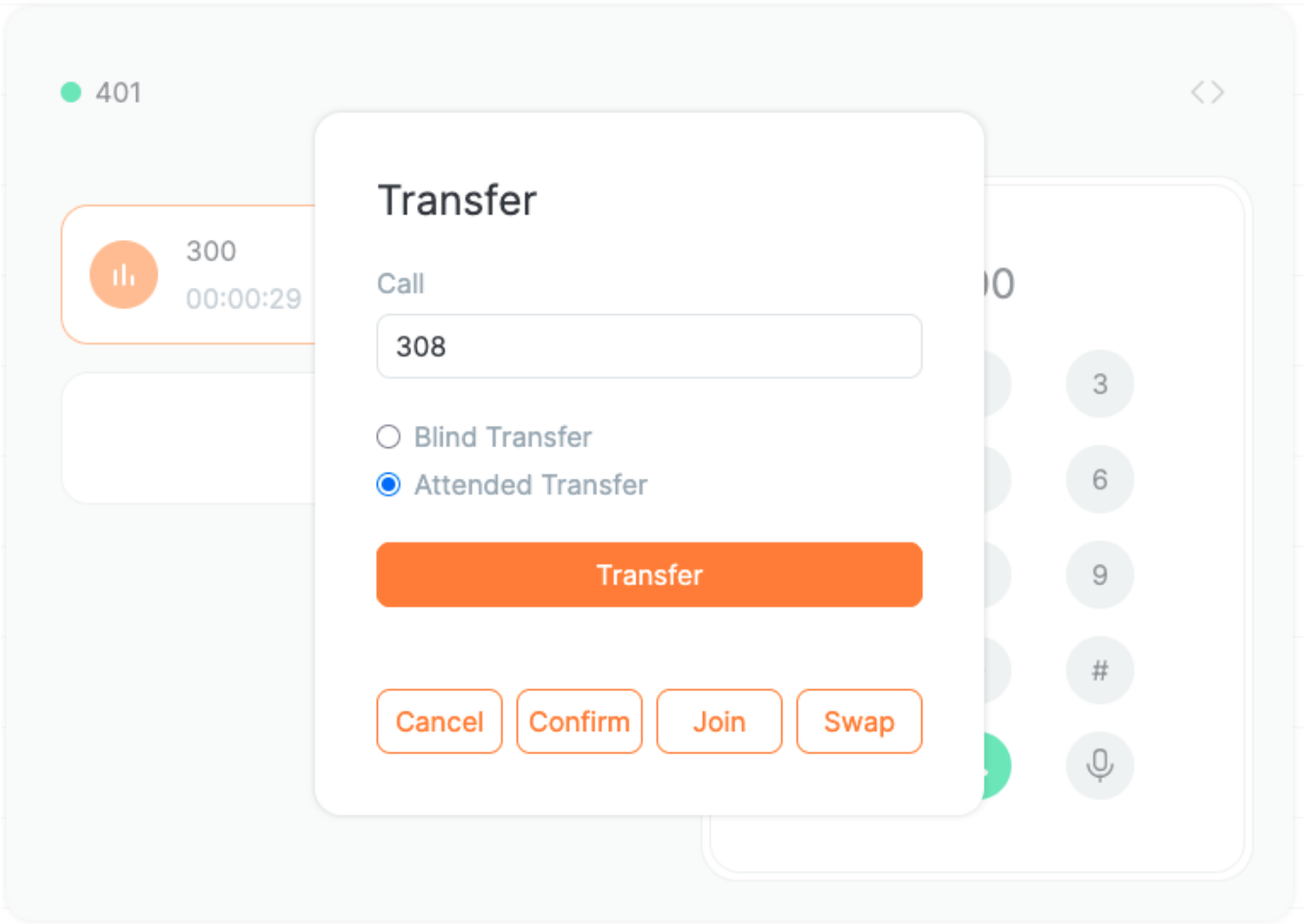
- Blind Transfer
- Attended Transfer

In an **attended transfer**, once the destination answers, the panel remains open for the following actions:

- **Cancel** – Abort transfer and resume the original call
- **Confirm** – Complete the transfer and hang up the original call
- **Join** – Merge all parties into a 3-way call
- **Swap** – Alternate between the original and transfer calls



On **FreePBX 16 and later**, DTMF Feature Codes must be manually enabled for these actions to work. Refer to the **QueueMetrics Advanced Manual**, Chapter **Setting up QueueMetrics WebRTC Softphone**, section **Enable DTMF Feature code**.



DTMF feature codes can be customized via the **Explore System Parameters** section.

feature code

Save

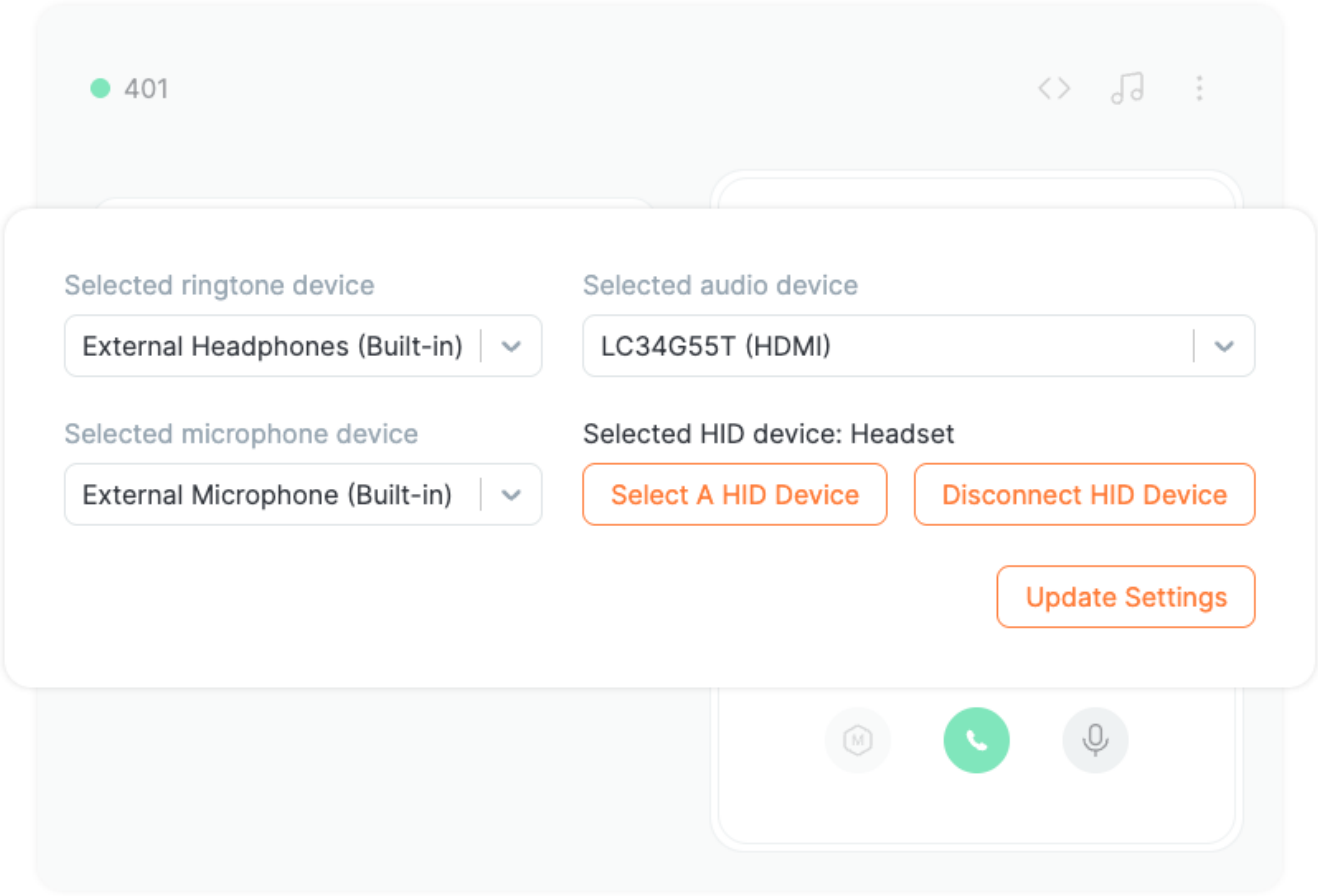
WebRTC Softphone

SoftPhone Global Defaults

Blind Transfer Feature Code	##
Attended Transfer Feature Code	*2
Abort Attended Transfer Feature Code	*3
Complete Attended Transfer Feature Code	*4
Join Attended Transfer Feature Code	*5
Swap Attended Transfer Feature Code	*6

Advanced Audio Device Selection

The softphone supports advanced selection and management of audio and device routing through a dedicated audio configuration panel.



Audio Device Routing Options

Once the softphone is in a **registered** state, an music icon becomes available in the interface. Clicking it opens a configuration panel that provides the following device selection options:

- **Ringtone Output** – Dropdown to select the device for ringtone playback.
- **Call Audio Output** – Dropdown to select the device used to listen to call audio.
- **Microphone Input** – Dropdown to select the microphone used during calls.

These selections allow users to route audio precisely across available devices (e.g., send ringtone to speakers but calls to a headset).

Clicking the **Update Settings** button applies the selected devices and stores their configuration in the browser's local storage. These settings are saved per-browser and per-computer, and cannot be transferred between browsers or machines. If the same browser is reopened later, the last-used configuration will be reapplied automatically, if the devices are still available.



If no microphone (input device) is detected on the system, the audio configuration panel will not display any selectable devices.

> A working input device is required for the softphone to function properly.
>
> Audio, microphone, and HID devices can be changed **even during an active call**. The new configuration is applied immediately without interrupting the call session.
>
> In some cases, an error may occur when changing the output device (e.g., due to device compatibility or disconnection). If this happens, a notification will appear.
>
> To resolve the issue, you can: Replug the previously used audio device, try selecting a different available device from the dropdown in the audio menu panel and once the call ends, refresh the page to reset the audio routing environment.

HID Device Management

The "USR_HID" key is required for each user to enable this feature.

A dedicated section is available for managing **HID-compatible devices** (e.g., USB headsets with physical call control buttons). It is composed of a label showing the name of the currently connected HID device, if any. A **Select HID Device** button, which opens a native browser prompt to choose an available HID device.(Upon successful selection, a confirmation notification is shown, otherwise an error message is displayed)

If a device is connected, an additional button **Disconnect HID Device** appears, allowing users to manually disconnect the device and removes it from localstorage.

HID device selection is only supported in browsers based on the Chromium engine(eg.Google Chrome, Microsoft Edge,Opera,Chromium)

HID Device Button Behavior

If a connected HID device has physical buttons (e.g., a call-answer button), those buttons will answers the call if a call is incoming otherwise will hangup the active call session. After an action to hang up or answer is executed, this behavior is locked for 5 seconds. This prevents unwanted automatic answering and hanging up, which can be caused by some devices sending multiple events from a single click.

Device Disconnection and Fallback

If any selected audio device is physically disconnected from the computer during a session the softphone will revert to the browser’s default input/output devices.

Device selection can then be modified again through the system’s audio settings or the softphone panel.

Persistence and Memory

Audio and HID device configurations are stored using browser **localStorage** (The configuration is

unique to each browser and computer combination) When reloading or reopening the softphone interface, the system attempts to reconnect the previously saved devices automatically if they are available.

Known Limitations

On Linux, **Chromium-based browsers** (eg. Google Chrome,Microsoft Edge,Opera,Chromium (open-source base)) currently do not support dynamic detection of newly added audio/HID devices. To view and use newly connected devices, users must **reload the page**. The WebHID API receives every event sent from the headset. However, when other applications like Teams, Skype, or the headset's proprietary software are running simultaneously with the QM softphone, they can interfere by also trying to communicate with the device. This conflict may result in unwanted behavior, such as causing the QM softphone to answer or hang up calls unexpectedly.

ICE Gathering Optimization

To reduce call setup time, the softphone uses a partial ICE gathering strategy. Instead of waiting for every possible network path, it prioritizes usable (routable) candidates and moves forward as soon as it has enough information to establish a connection.

ICE Gathering Mechanism

The gathering process is driven by a simple timing strategy that balances speed with reliability:

- Initial discovery: The browser starts collecting ICE candidates as usual.
- Hard timeout: A maximum limit (controlled by default.maxGatheringTime) ensures the process doesn't take too much time. Even in slow or restricted networks, the call will eventually proceed.
- Soft Timeout (Routable Discovery): If a STUN/TURN candidate is found, the system identifies this as an ideal candidate and instead of waiting for the full hard timeout, a shorter "Post-Gathering" delay is triggered.
- Finalization: Once the soft timeout expires, or if the browser naturally signals that gathering is complete (sending a null candidate), the call proceeds immediately using the best available candidate.

Configuration

The behavior of the gathering process is controlled by two specific configuration properties:

- **default.maxGatheringTime**: (Expressed in ms) Sets the maximum total time allowed for the ICE gathering process.
  - If set to 0, or not set, the optimization is disabled and the softphone reverts to the browser's default natural gathering.
- **default.icePostGatheringTimeout**: (Expressed in ms) The duration of the soft timeout triggered after a routable STUN/TURN candidate is detected.
  - This allows a brief window for even better candidates to be discovered before finalizing the connection.
  - Defaults to 500 ms.

Behavioral Logic Flow

When a call is initiated, the following sequence is executed:

- The system listens for ICE candidates
- If maxGatheringTime > 0, the optimized logic is enabled.
- When a STUN or TURN candidate is detected, the post-gathering timer starts.
- If Soft Timeout timer expires before the Hard timeout, gathering stops early and the call

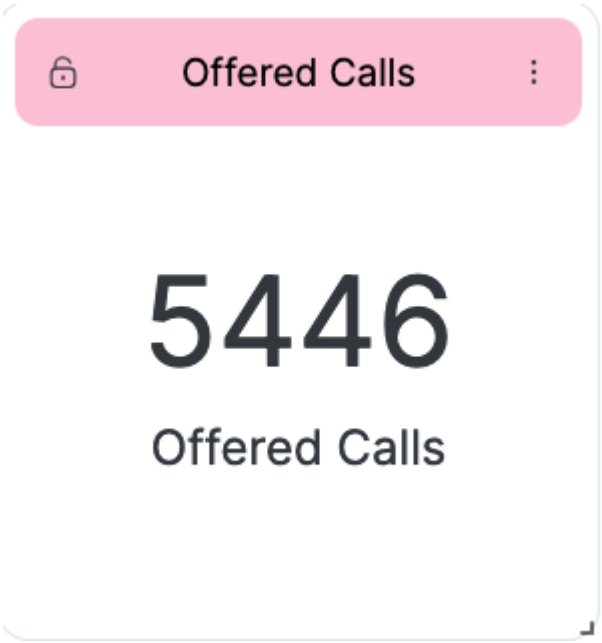
proceeds.

- If no routable candidates appear before the hard timeout, the system logs a warning and continues with whatever candidates are available, avoiding indefinite delays.

Component Details

Each widget is draggble within the wallboard so that it can be customized as desired.

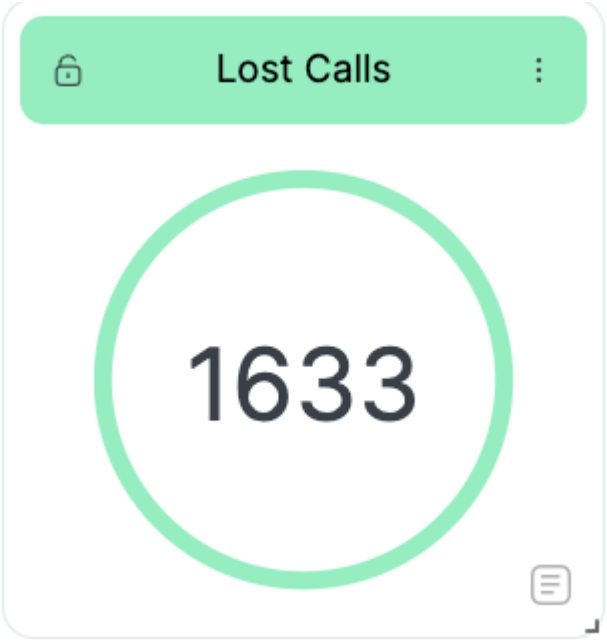
When hovered the widget will show two icons:



- Lock icon: when pressed indicates whether a widget is draggable or whether you want to fix in a specific place on the wallboard
- Three points icon: open a model where you can change the settings of the widget

Additional details

Some widgets have the ability to open a modal window that shows some additional details of the widget. These widgets, when hovered, have a detail icon in the lower right corner.



If you click on the icon the modal will open.



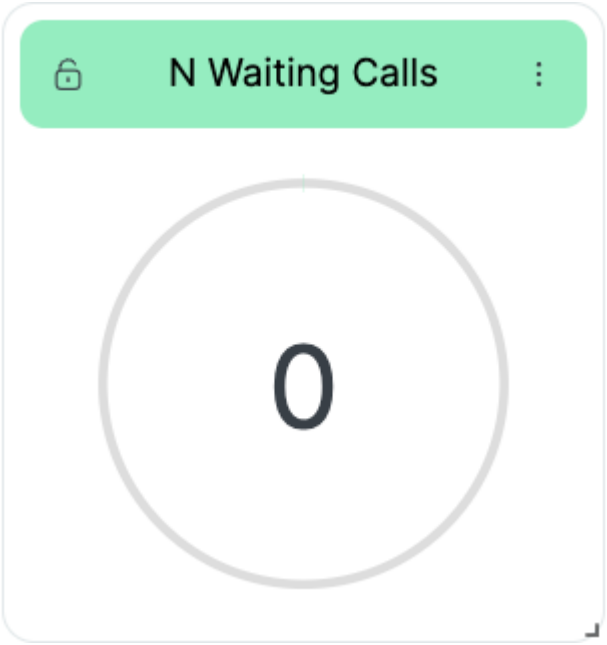
Date	Agent	Caller	Queue	Disconnection	Wait	Pos.	Attempts	Code	Key	Stints	Srv	Call Unique ID	IVR duration	IVR path	DNIS	IVR	Tag	Feat	Vars	Fea
11/14 - 17:17:22		467-555-3977	Sales Support	Abandon	0:02	0	0			0		c.891070213	0	6		0:01		0	0	
11/14 - 17:17:11		686-555-5904	Sales Support	Abandon	0:02	0	0			0		c.24626795	0	6		0:02		0	0	
11/14 - 17:17:03		452-555-1811	Sales	Abandon	0:01	0	0			0		c.847608149	0	3		0:01		0	0	
11/14 - 17:16:52		352-555-7725	Sales	Abandon	0:02	0	0			0		c.889666468	0	1		0:02		0	0	
11/14 - 17:15:45		391-555-2232	Technical Support	Abandon	0:06	0	1			0		c.579253367	0	2		0:02		0	0	
11/14 - 17:14:46		294-555-6083	Sales Support	Abandon	0:02	0	0			0		c.753668226	0	0		0:03		0	0	
11/14 - 17:14:23		710-555-8560	Sales Support	Abandon	0:04	0	1			0		c.914432337	0	2		0:02		0	0	
11/14 - 17:13:50		826-555-3483	Sales Support	Abandon	0:08	0	1			0		c.151491829	0	8		0:01		0	0	
11/14 - 17:13:42		774-555-3940	Technical Support	Abandon	0:10	0	1			0		c.118694102	0	6		0:01		0	0	
11/14 - 17:13:08		709-555-9045	Sales	Abandon	0:03	0	0			0		c.285519963	0	5		0:03		0	0	
11/14 - 17:12:52		339-555-5956	Technical Support	Abandon	0:01	0	0			0		c.57049934	0	5		0:03		0	0	
11/14 - 17:11:31		949-555-4974	Sales Support	Abandon	0:03	0	0			0		c.268999944	0	1		0:01		0	0	
11/14 - 17:11:23		550-555-9689	Technical Support	Abandon	0:01	0	0			0		c.328304644	0	1		0:03		0	0	
11/14 - 17:10:38		211-555-5109	Sales	Abandon	0:01	0	0			0		c.222603733	0	5		0:03		0	0	
11/14 - 17:10:25		222-555-3332	Sales	Abandon	0:02	0	0			0		c.994640064	0	4		0:01		0	0	

Here is an overview of the components currently available in Queuemetrics' Wallboard.

Circular Panels

PC01 - Waiting Calls

This Component displays how many calls are currently waiting to be answered.



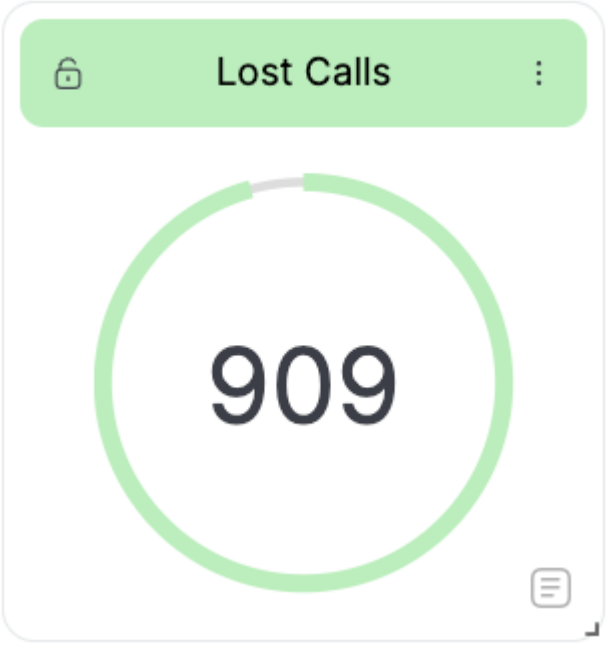
The following properties can be modified:

- **Title**: The title of the component
- **Color**: The color of the stroke
- **Red alarm**: The expression used to trigger a **red alarm** (eg. >20)
- **Yellow alarm**: The expression used to trigger a **yellow alarm** (eg. >20)
- **Queue**: The queue you want to filter the data for

Available since	17.06
Name	Waiting calls
Shortcut code	PC01
Parameters	-
See also	

PC02 - Lost Calls

This Component displays the number of lost calls.



The following properties can be modified:

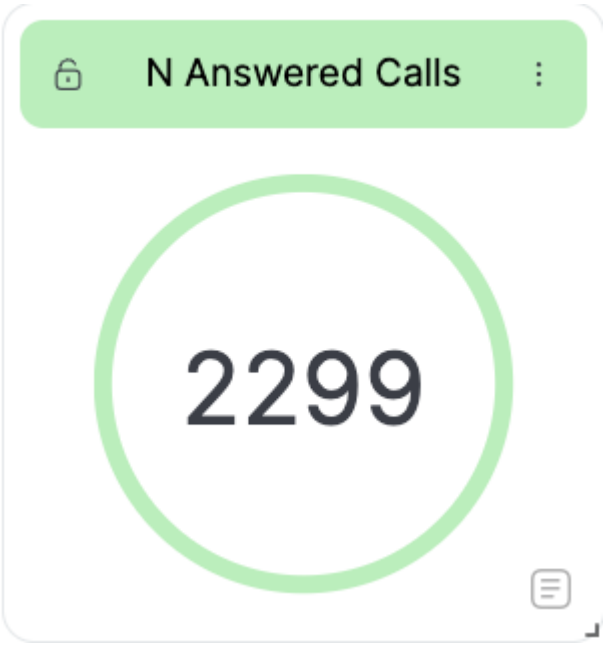
- **Title**: The title of the component
- **Color**: The color of the stroke
- **Red alarm**: The expression used to trigger a **red alarm** (eg. >20)
- **Yellow alarm** The expression used to trigger a **yellow alarm** (eg. >20)
- **Queue**: The queue you want to filter the data for

Available since	17.06
Name	Lost calls
Shortcut code	PC02
Parameters	-
See also	

- **Additional details shown**: 15 most recent calls

PC03 - Answered Calls

This Component displays the number of answered calls.



The following properties can be modified:

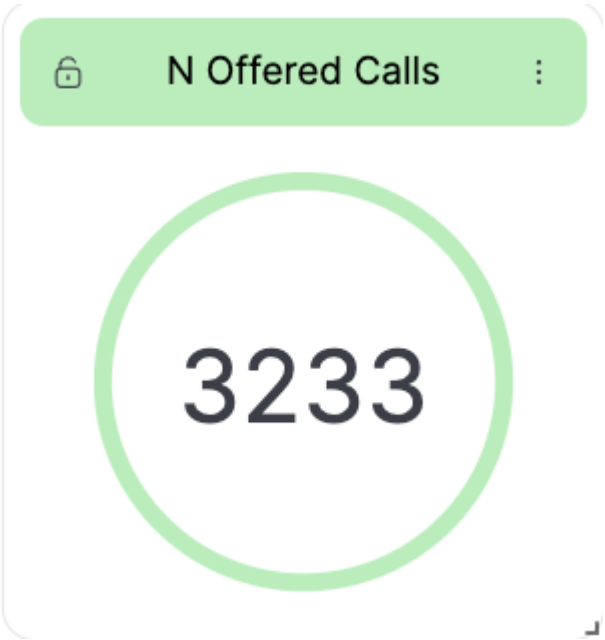
- **Title**: The title of the component
- **Color**: The color of the stroke
- **Red alarm**: The expression used to trigger a **red alarm** (eg. >20)
- **Yellow alarm** The expression used to trigger a **yellow alarm** (eg. >20)
- **Queue**: The queue you want to filter the data for

Available since	17.06
Name	Answered calls
Shortcut code	PC03
Parameters	-
See also	

- **Additional details shown**: 15 most recent calls

PC04 - Offered Calls

This Component displays the number of offered calls.



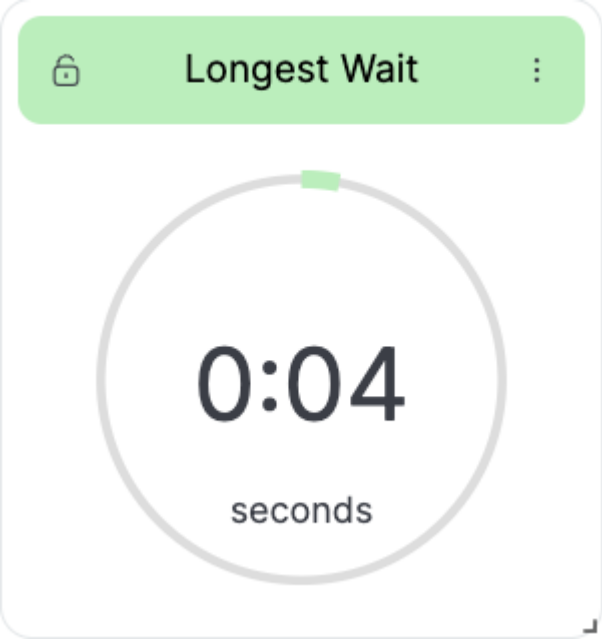
The following properties can be modified:

- **Title**: The title of the component
- **Color**: The color of the stroke
- **Red alarm**: The expression used to trigger a **red alarm** (eg. >20)
- **Yellow alarm** The expression used to trigger a **yellow alarm** (eg. >20)
- **Queue**: The queue you want to filter the data for

Available since	17.06
Name	Offered calls
Shortcut code	PC04
Parameters	-
See also	

PC05 - Longest Wait

This Component displays the duration of the longest waiting time currently happening on the selected queues.



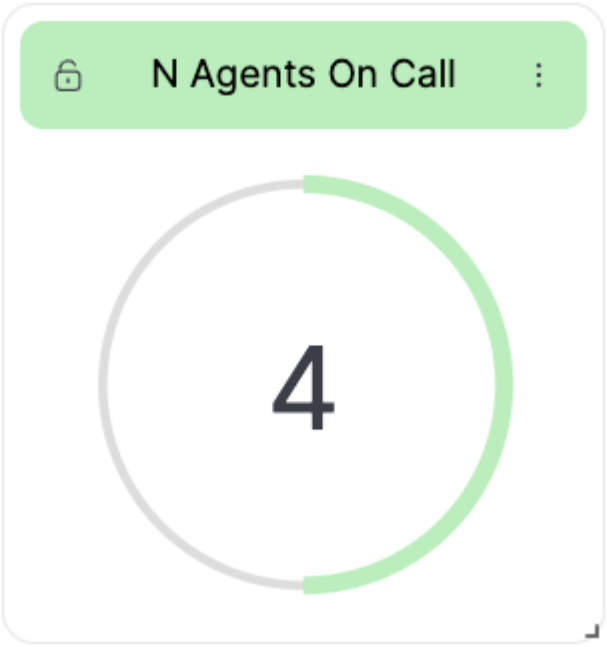
The following properties can be modified:

- **Title**: The title of the component
- **Color**: The color of the stroke
- **Red alarm**: The expression used to trigger a **red alarm** (eg. >20)
- **Yellow alarm** The expression used to trigger a **yellow alarm** (eg. >20)
- **Queue**: The queue you want to filter the data for

Available since	17.06
Name	Longest Wait
Shortcut code	PC05
Parameters	-
See also	

PC06 - Agents On Call

This Component displays the number of agents on call on the selected queues.



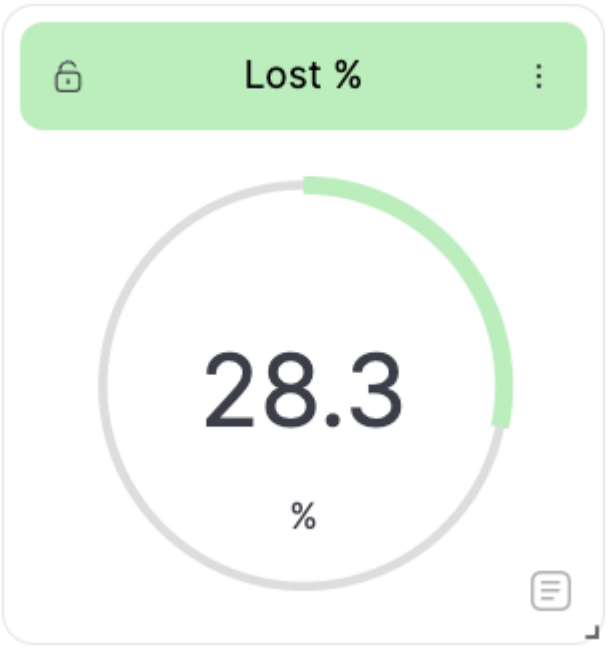
The following properties can be modified:

- **Title**: The title of the component
- **Color**: The color of the stroke
- **Red alarm**: The expression used to trigger a **red alarm** (eg. >20)
- **Yellow alarm** The expression used to trigger a **yellow alarm** (eg. >20)
- **Queue**: The queue you want to filter the data for

Available since	17.06
Name	Agents On Call
Shortcut code	PC06
Parameters	-
See also	

PC07 - Lost Calls %

This Component displays the percentage of lost calls.



The following properties can be modified:

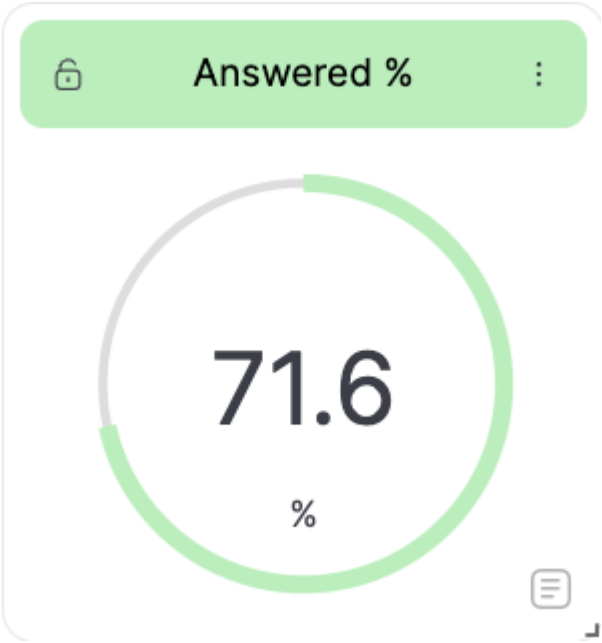
- **Title**: The title of the component
- **Color**: The color of the stroke
- **Red alarm**: The expression used to trigger a **red alarm** (eg. >20)
- **Yellow alarm** The expression used to trigger a **yellow alarm** (eg. >20)
- **Queue**: The queue you want to filter the data for

Available since	17.06
Name	Lost Calls %
Shortcut code	PC07
Parameters	-
See also	

- **Additional details shown**: 15 most recent calls

PC08 - Answered Calls %

This Component displays the percentage of answered calls.



The following properties can be modified:

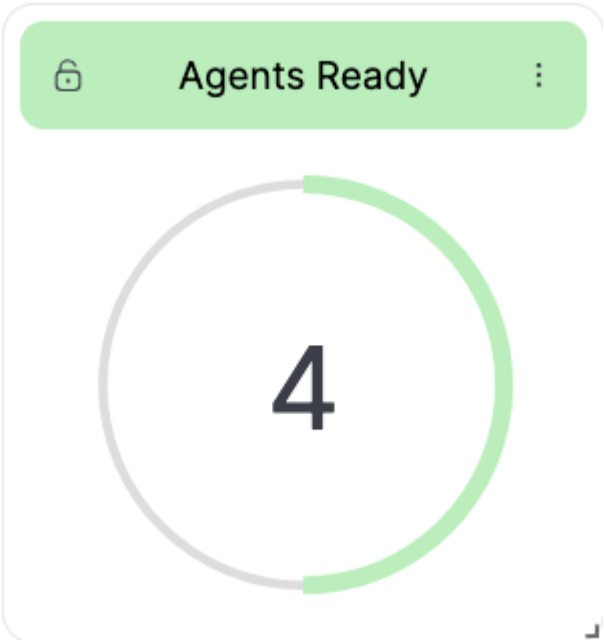
- **Title**: The title of the component
- **Color**: The color of the stroke
- **Red alarm**: The expression used to trigger a **red alarm** (eg. >20)
- **Yellow alarm** The expression used to trigger a **yellow alarm** (eg. >20)
- **Queue**: The queue you want to filter the data for

Available since	17.06
Name	Answered Calls %
Shortcut code	PC08
Parameters	-
See also	

- **Additional details shown**: 15 most recent calls

PC09 - Agents Ready

This Component displays the number of agents ready on the selected queues.



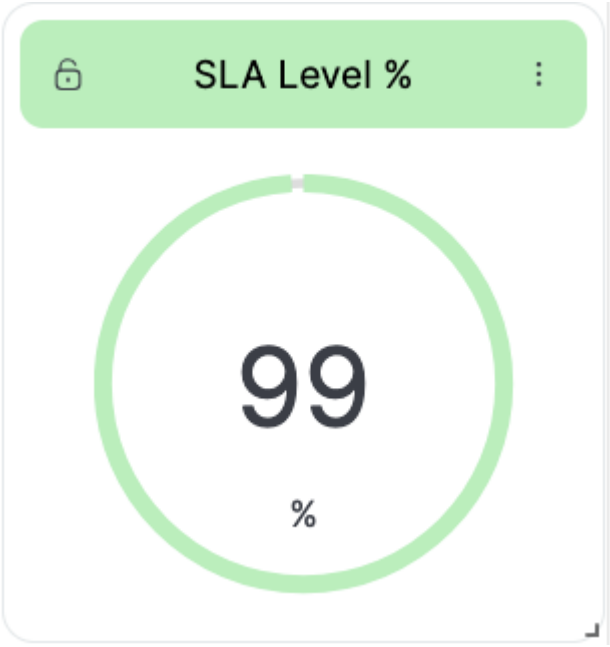
The following properties can be modified:

- **Title**: The title of the component
- **Color**: The color of the stroke
- **Red alarm**: The expression used to trigger a **red alarm** (eg. >20)
- **Yellow alarm** The expression used to trigger a **yellow alarm** (eg. >20)
- **Queue**: The queue you want to filter the data for

Available since	17.06
Name	Agents Ready
Shortcut code	PC09
Parameters	-
See also	

PC10 - SLA Level %

This Component displays the percentage of calls answered within the selected SLA level.



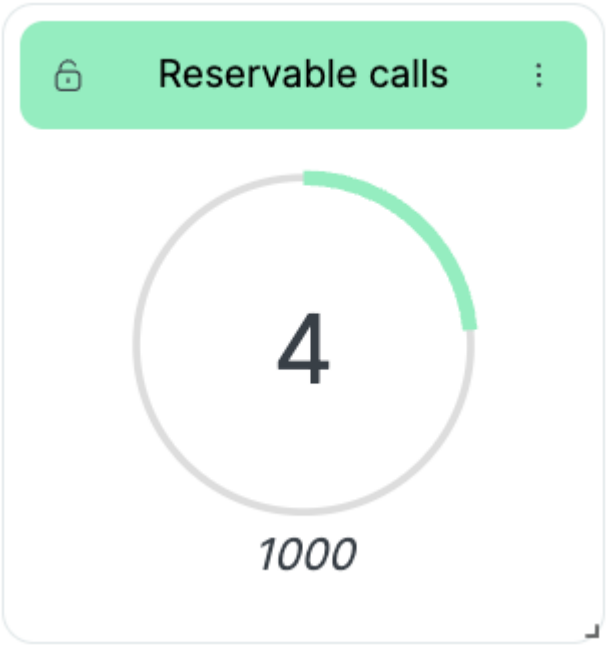
The following properties can be modified:

- **Title**: The title of the component
- **SLA Level**: The SLA time interval
- **Color**: The color of the stroke
- **Red alarm**: The expression used to trigger a **red alarm** (eg. >20)
- **Yellow alarm** The expression used to trigger a **yellow alarm** (eg. >20)

Available since	17.06
Name	SLA Level %
Shortcut code	PC10
Parameters	-
See also	

PC11 - Reservable Calls

This Component displays the number of Wombat reservable calls available.



The following properties can be modified:

- **Title**: The title of the component
- **Queue**: The queue you want to see the data for (**MANDATORY**)
- **Color**: The color of the stroke
- **Red alarm**: The expression used to trigger a **red alarm** (eg. >20)
- **Yellow alarm** The expression used to trigger a **yellow alarm** (eg. >20)
- **Max Val** The upper limit used to calculate the percentage for the circular progress animation

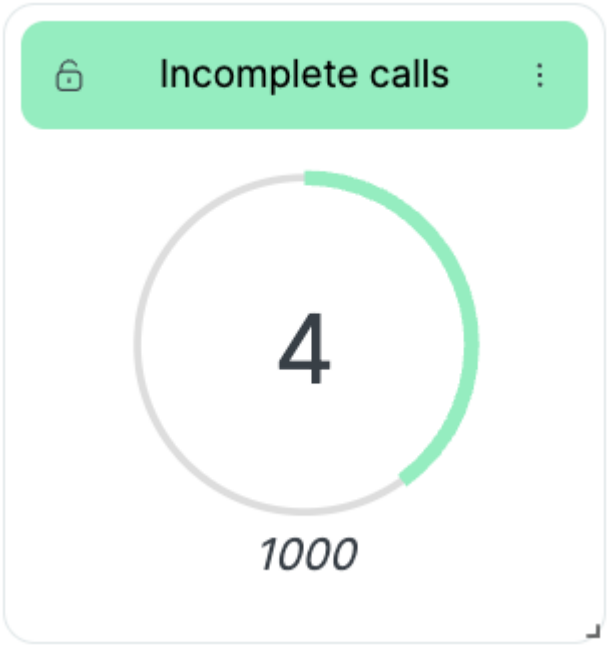
This widget enter an "expired" mode if attribute data becomes stale or missing. (See. Expired Widget)

If the retrieved value is invalid or non-numeric, the widget transitions to a disabled state and displays `NaN`.

Name	Reservable calls
Shortcut code	PC11
Attributes	WBT.RESERVABLE_CALLS
See also	WombatDialer

PC12 - Incomplete Calls

This Component displays the number of wombat incomplete calls.



The following properties can be modified:

- **Title**: The title of the component
- **Queue**: The queue you want to see the data for (**MANDATORY**)
- **Color**: The color of the stroke
- **Red alarm**: The expression used to trigger a **red alarm** (eg. >20)
- **Yellow alarm** The expression used to trigger a **yellow alarm** (eg. >20)
- **Max Val** The upper limit used to calculate the percentage for the circular progress animation

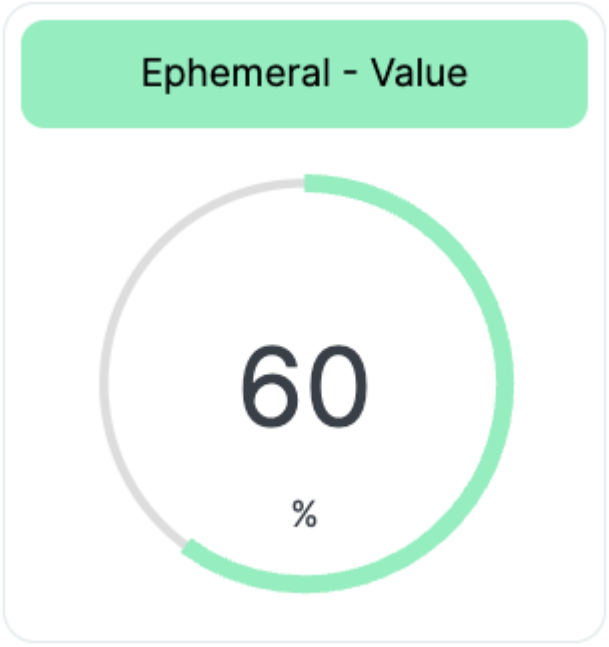
This widget enter an "expired" mode if attribute data becomes stale or missing. (See. Expired Widget)

If the retrieved value is invalid or non-numeric, the widget transitions to a disabled state and displays `NaN`.

Name	Incomplete calls
Shortcut code	PC12
Attributes	WBT.INCOMPLETE_CALLS
See also	WombatDialer

PC13 - Ephemeral - Value

This Component displays a specific numeric attribute.



The following properties can be modified:

- **Title**: The title of the component
- **Attribute**: Select the specific attribute to track. The "EphemeralDO". prefix is automatically prepended to the selection.
- **Queue**: The queue you want to see the data for (can be left empty if the attribute is global)
- **Color**: The color of the stroke
- **Display Mode**: Choose between absolute number or percentage
- **Red alarm**: The expression used to trigger a **red alarm** (eg. >20)
- **Yellow alarm** The expression used to trigger a **yellow alarm** (eg. >20)
- **Max Val** The upper limit used to calculate the percentage for the circular progress animation

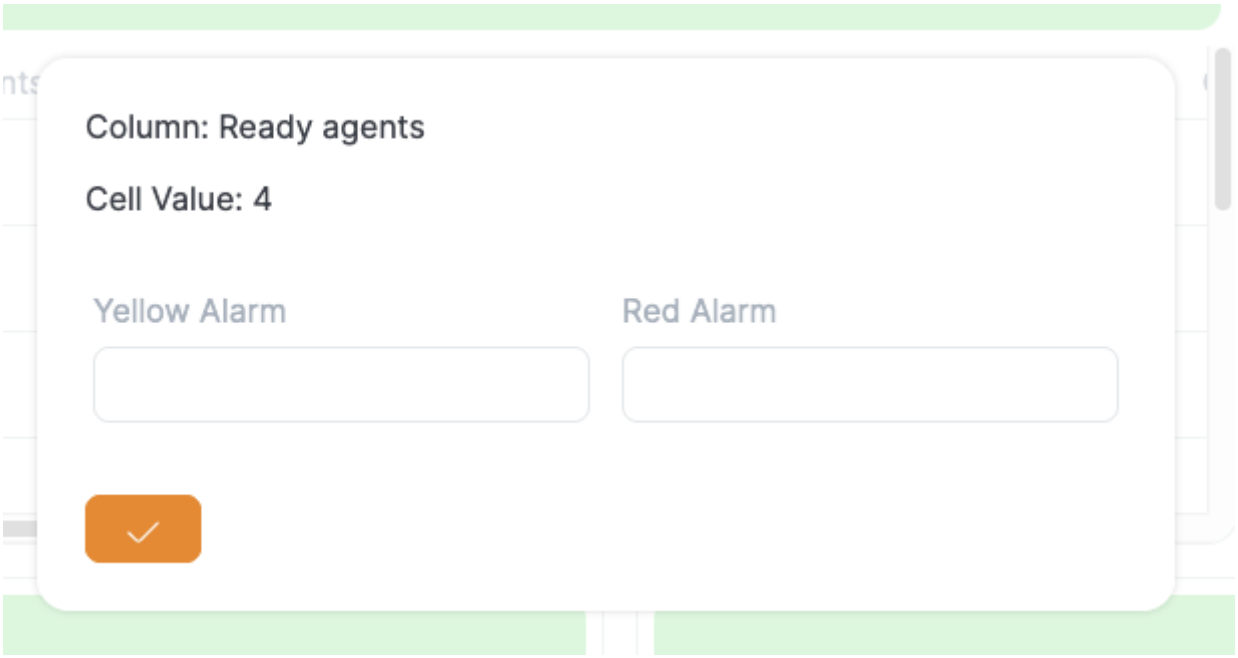
This widget enter an "expired" mode if attribute data becomes stale or missing. (See. Expired Widget)

If the retrieved value is invalid or non-numeric, the widget transitions to a disabled state and displays `NaN`.

Name	Incomplete calls
Shortcut code	PC13
See also	WombatDialer

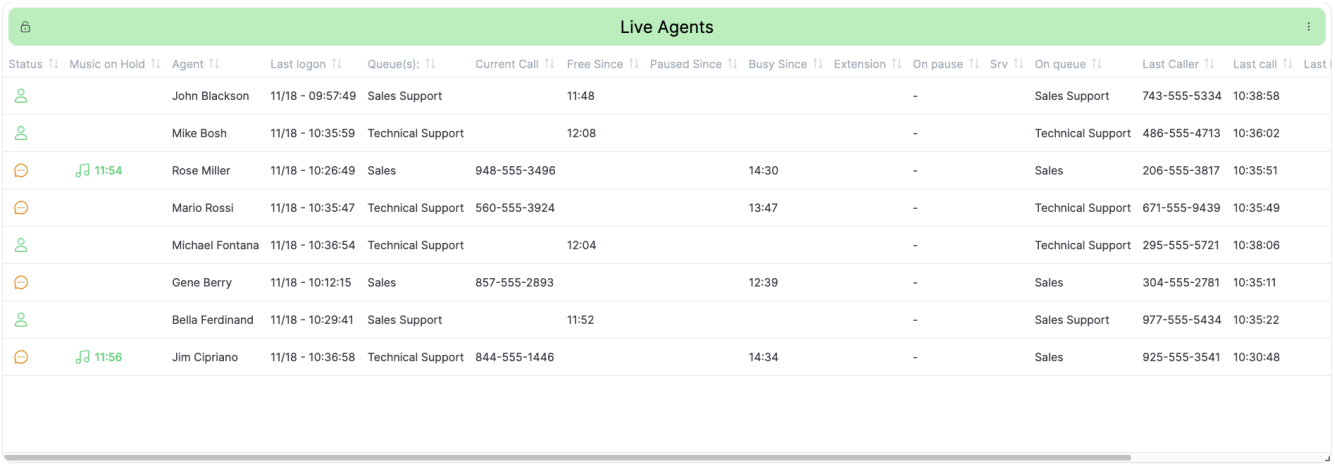
Tables

Table widget alarms can be set by clicking on a table's cell.



TC01 - Live Agents

This Component displays a table showing all the agents logged in on the selected queues.



The following properties can be modified:

- **Title**: The title of the component
- **Queue**: The queue you want to filter the data for
- **Columns**: The columns of the table you want to show

The table will also be responsive to any **Red** or **Yellow alarm** that has been set in the **Edit Queues** page under the following fields:

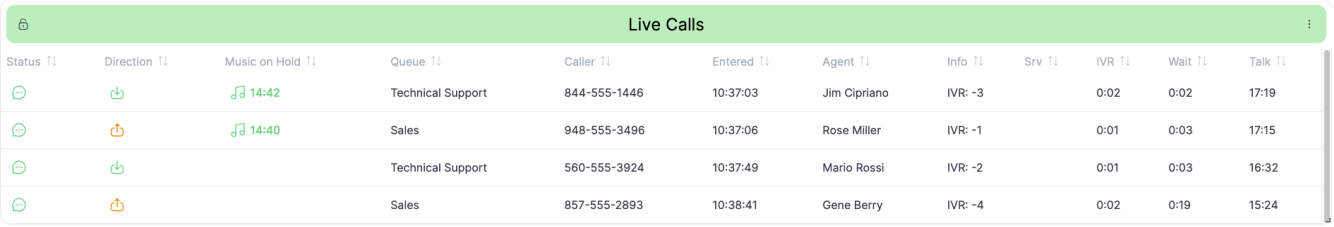
- **Call talking duration**

Available since	17.06
Name	Live Agents
Shortcut code	TC01

Parameters	-
See also	

TC02 - Live Calls

This Component displays a table showing all the calls in progress on the selected queues.



The following properties can be modified:

- **Title**: The title of the component
- **Queue**: The queue you want to filter the data for
- **Columns**: The columns of the table you want to show

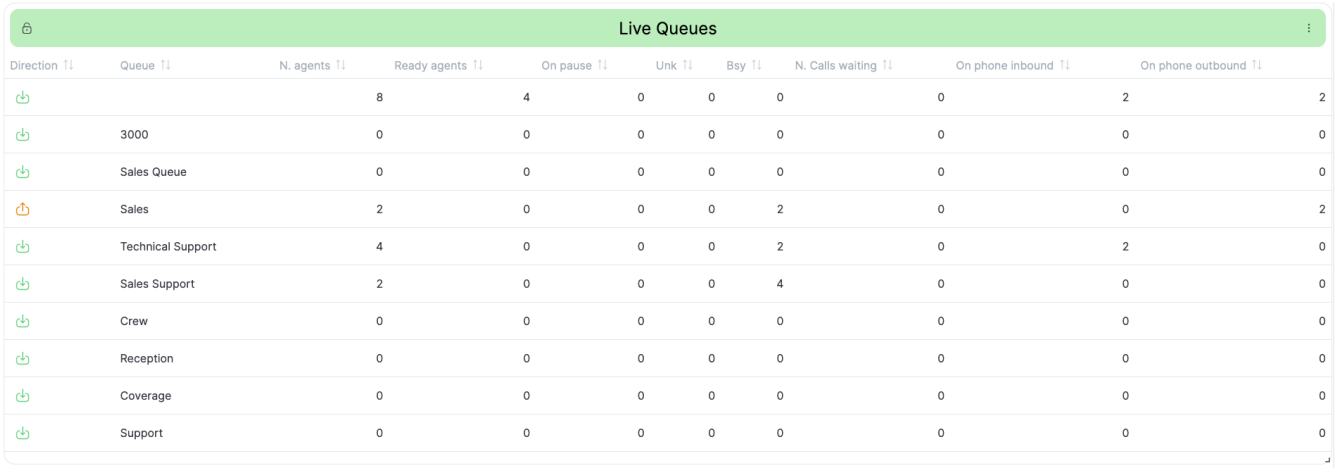
The table will also be responsive to any **Red** or **Yellow alarm** that has been set in the **Edit Queues** page under the following fields:

- **Call wait duration**
- **Call talking duration**

Available since	17.06
Name	Live Calls
Shortcut code	TC02
Parameters	-
See also	

TC03 - Live Queues

This Component displays a table showing all the selected queues, and their relative information.



Live Queues

Direction	Queue	N. agents	Ready agents	On pause	Unk	Bsy	N. Calls waiting	On phone inbound	On phone outbound
		8	4	0	0	0	0	2	2
	3000	0	0	0	0	0	0	0	0
	Sales Queue	0	0	0	0	0	0	0	0
	Sales	2	0	0	0	2	0	0	2
	Technical Support	4	0	0	0	2	0	2	0
	Sales Support	2	0	0	0	4	0	0	0
	Crew	0	0	0	0	0	0	0	0
	Reception	0	0	0	0	0	0	0	0
	Coverage	0	0	0	0	0	0	0	0
	Support	0	0	0	0	0	0	0	0

The following properties can be modified:

- **Title**: The title of the component
- **Queue**: The queue you want to filter the data for
- **Columns**: The columns of the table you want to show

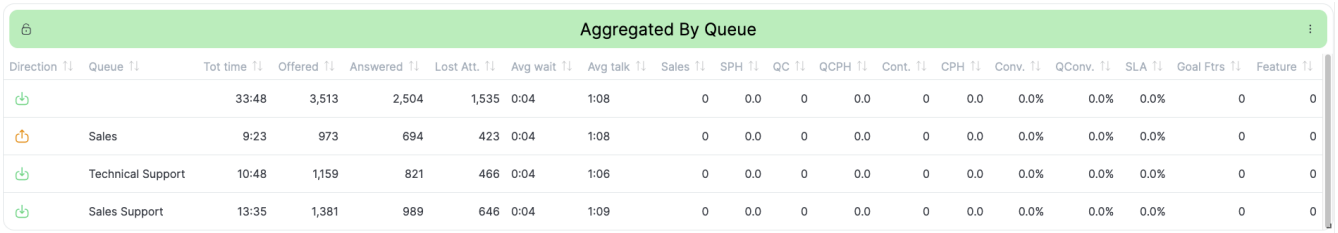
The table will also be responsive to any **Red** or **Yellow alarm** that has been set in the **Edit Queues** page under the following fields:

- **Number of agents waiting**
- **Number of agents paused**
- **Number of agents on call**
- **Number of calls in queue**

Available since	17.06
Name	Live Queues
Shortcut code	TC03
Parameters	-
See also	

TC04 - Aggregated By Queue

This Component displays a table showing the aggregate information for the day, grouped by queue.



Aggregated By Queue

Direction	Queue	Tot time	Offered	Answered	Lost Att.	Avg wait	Avg talk	Sales	SPH	QC	QCPH	Cont.	CPH	Conv.	QConv.	SLA	Goal Ftrs	Feature
		33:48	3,513	2,504	1,535	0:04	1:08	0	0.0	0	0.0	0	0.0	0.0%	0.0%	0.0%	0	0
	Sales	9:23	973	694	423	0:04	1:08	0	0.0	0	0.0	0	0.0	0.0%	0.0%	0.0%	0	0
	Technical Support	10:48	1,159	821	466	0:04	1:06	0	0.0	0	0.0	0	0.0	0.0%	0.0%	0.0%	0	0
	Sales Support	13:35	1,381	989	646	0:04	1:09	0	0.0	0	0.0	0	0.0	0.0%	0.0%	0.0%	0	0

The following properties can be modified:

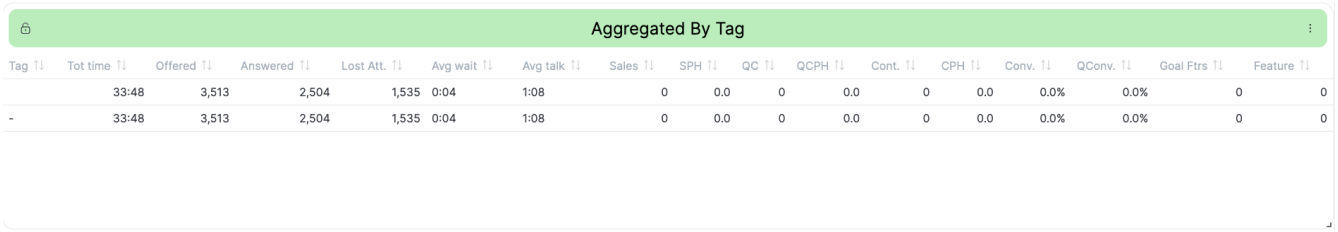
- **Title**: The title of the component

- **Columns**: The columns of the table you want to show

Available since	19.04.1
Name	Aggregated By Queue
Shortcut code	TC04
Parameters	-
See also	

TC05 - Aggregated By Tag

This Component displays a table showing the aggregate information for the day, grouped by tag.



The following properties can be modified:

- **Title**: The title of the component
- **Columns**: The columns of the table you want to show

Available since	19.04.1
Name	Aggregated By Tag
Shortcut code	TC05
Parameters	-
See also	

TC06 - Agents and Outcomes

This Component displays a table showing the aggregate information for the day, regarding the agents and the outcomes of their calls.



Status	Status	Agent	Offered	Answered	Lost Att.	Tot talk	Tot idle	Pause bill.	Pause non bill.	Avg talk	Sales	SPH	QC	QCPH	Cont.	CPH	Conv.	QConv.
	Lost Calls		1,009	0	0	0:00	0:00	0:00	0:00	-	0	0.0	0	0.0	0	0.0	0.0%	0.0%
	Logged on - 1:02:26	John Blackson	316	316	186	5:40:33	5:12:26	0:00	4:11	1:04	0	0.0	0	0.0	0	0.0	0.0%	0.0%
	Logged on - 24:16	Mike Bosh	322	322	178	5:57:53	4:53:33	0:00	4:10	1:06	0	0.0	0	0.0	0	0.0	0.0%	0.0%
	Logged on - 33:26	Rose Miller	288	288	195	5:38:52	5:05:01	0:00	4:56	1:10	0	0.0	0	0.0	0	0.0	0.0%	0.0%
	Logged on - 24:28	Mario Rossi	307	307	184	6:20:32	4:28:09	0:00	4:05	1:14	0	0.0	0	0.0	0	0.0	0.0%	0.0%
	Logged on - 23:21	Michael Fontana	336	336	197	5:34:46	5:12:33	0:00	4:19	0:59	0	0.0	0	0.0	0	0.0	0.0%	0.0%
	Logged on - 48:00	Gene Berry	333	333	192	6:05:26	4:49:28	0:00	3:40	1:05	0	0.0	0	0.0	0	0.0	0.0%	0.0%
	Logged on - 30:34	Bella Ferdinand	296	296	209	5:57:25	4:55:22	0:00	3:47	1:12	0	0.0	0	0.0	0	0.0	0.0%	0.0%
	Logged on - 23:17	Jim Cipriano	306	306	194	6:03:05	4:46:20	0:00	4:40	1:11	0	0.0	0	0.0	0	0.0	0.0%	0.0%

The following properties can be modified:

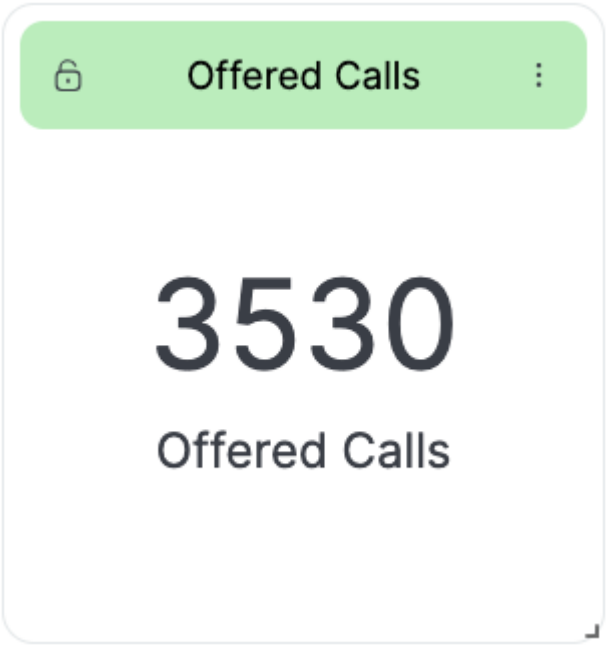
- **Title**: The title of the component
- **Columns**: The columns of the table you want to show

Available since	19.04.1
Name	Agents and Outcomes
Shortcut code	TC06
Parameters	-
See also	

Items

IT01 - N. of Calls Offered

This Component displays a text item showing the number of calls offered on the selected queues.



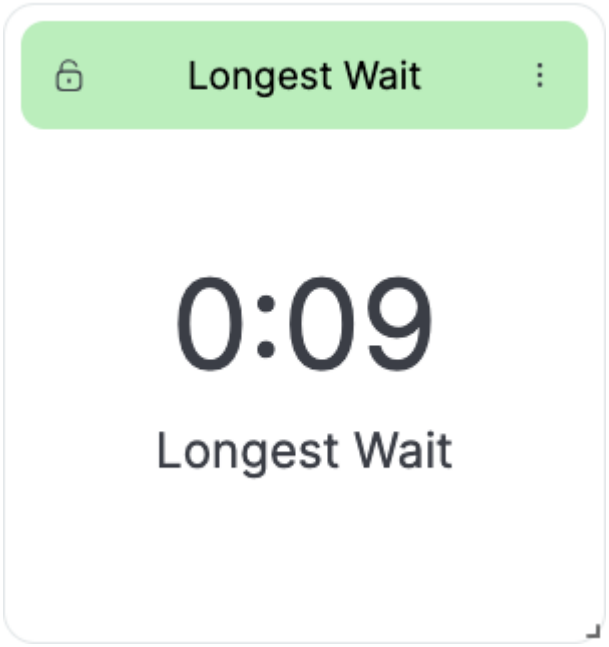
The following properties can be modified:

- **Title**: The title of the component
- **Red alarm**: The expression used to trigger a **red alarm** on the number of offered calls (eg. >20)
- **Yellow alarm**: The expression used to trigger a **yellow alarm** on the number of offered calls (eg. >20)
- **Color**: The color scheme for the background and text.
- **Queue**: The queue you want to filter the data for

Available since	17.06
Name	N. of Calls Offered
Shortcut code	IT01
Parameters	-
See also	

IT02 - Longest Wait

This Component displays a text item showing the longest wait time among the ongoing calls on the selected queues.



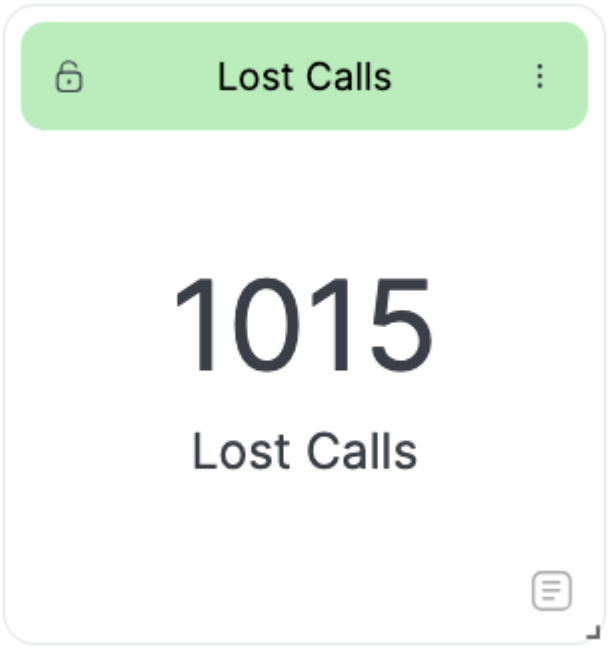
The following properties can be modified:

- **Title**: The title of the component
- **Red alarm**: The expression used to trigger a **red alarm** on the longest wait value (eg. >20)
- **Yellow alarm**: The expression used to trigger a **yellow alarm** on the longest wait value (eg. >20)
- **Color**: The color scheme for the background and text.
- **Queue**: The queue you want to filter the data for

Available since	17.06
Name	Longest Wait
Shortcut code	IT02
Parameters	-
See also	

IT03 - N. of Calls Lost

This Component displays a text item showing the number of calls lost on the selected queues.



The following properties can be modified:

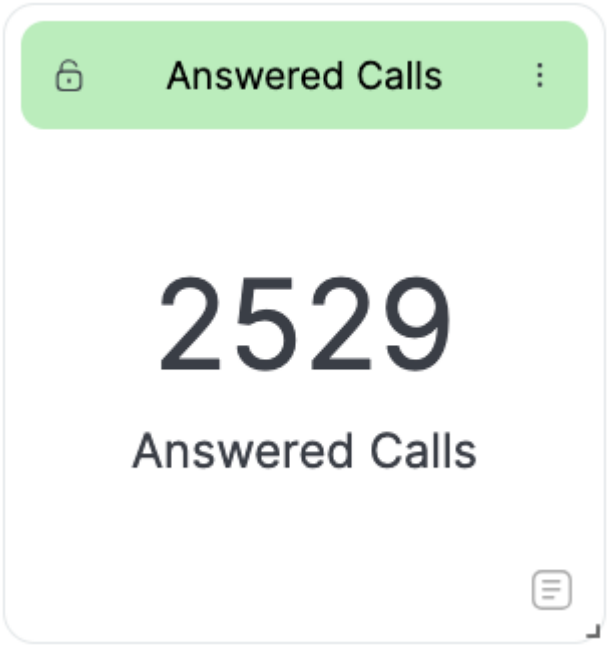
- **Title**: The title of the component
- **Red alarm**: The expression used to trigger a **red alarm** on the number of answered calls (eg. >20)
- **Yellow alarm**: The expression used to trigger a **yellow alarm** on the number of answered calls (eg. >20)
- **Color**: The color scheme for the background and text.
- **Queue**: The queue you want to filter the data for

Available since	17.06
Name	N of Calls Lost
Shortcut code	IT03
Parameters	-
See also	

- **Additional details shown**: 15 most recent calls

IT04 - N. of Calls Answered

This Component displays a text item showing the number of calls answered on the selected queues.



The following properties can be modified:

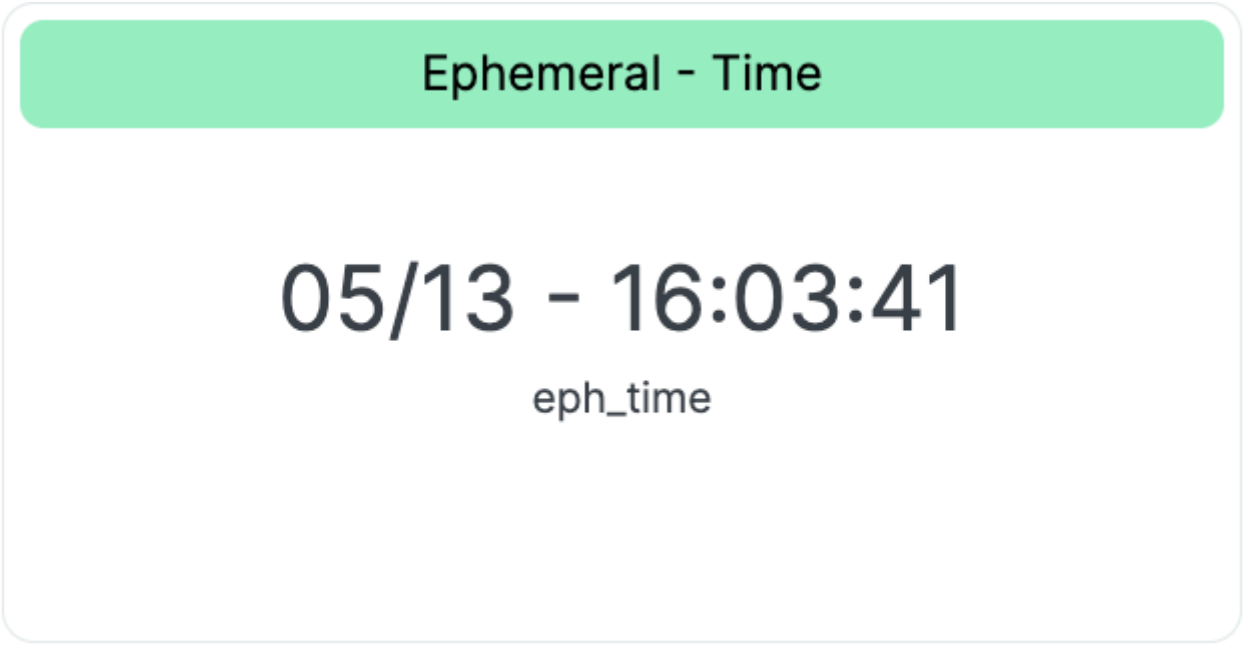
- **Title**: The title of the component
- **Red alarm**: The expression used to trigger a **red alarm** on the number of answered calls (eg. >20)
- **Yellow alarm**: The expression used to trigger a **yellow alarm** on the number of answered calls (eg. >20)
- **Color**: The color scheme for the background and text.
- **Queue**: The queue you want to filter the data for

Available since	17.06
Name	N of Calls Answered
Shortcut code	IT04
Parameters	-
See also	

- **Additional details shown**: 15 most recent calls

IT05 - Ephemeral - Time

This Component displays a specific unix timestamp into local dateformat.



The following properties can be modified:

- **Title**: The title of the component
- **Attribute**: Select the specific attribute to track. The "EphemeralDO". prefix is automatically prepended to the selection. (**MANDATORY**)
- **Queue**: The queue you want to see the data for (can be left empty if the attribute is global)
- **Color**: The color of the stroke
- **Red alarm**: The expression used to trigger a **red alarm** (eg. >20)
- **Yellow alarm** The expression used to trigger a **yellow alarm** (eg. >20)



This widget enter an "expired" mode if attribute data becomes stale or missing. (See. Expired Widget)

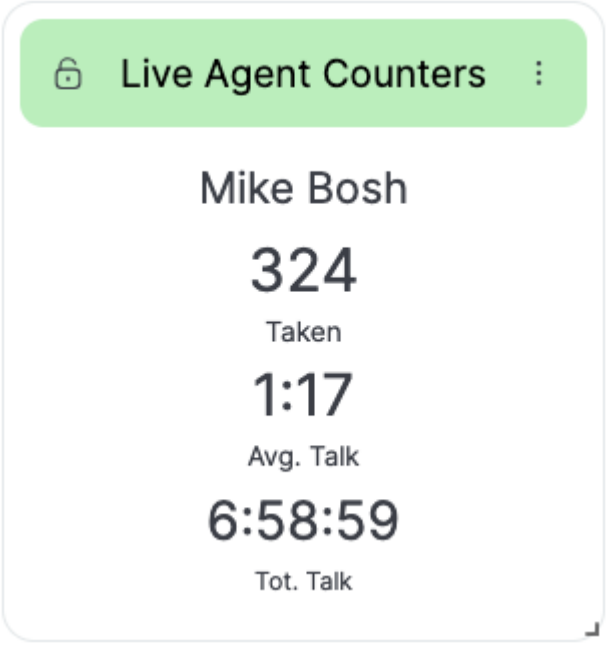
Name	Ephemeral - Time
Shortcut code	IT05
See also	WombatDialer

Composite Items

MI01 - Live Agents Counters

This Component is composed of three sections, each detailing a different information about an agent.

- The **Taken calls** is shown in the **uppermost part** of the component.
- The **middle part** of the component shows the **average talk time** value for the selected agent.
- The **bottom part** of the component will show the **Total talk time** for the selected agent.



The following properties can be modified:

- **Title**: The title of the component
- **Taken Red alarm**: The expression used to trigger a **red alarm** on the number of taken calls (eg. >20)
- **Taken Yellow alarm**: The expression used to trigger a **yellow alarm** on the number of taken calls (eg. >20)
- **Avg. Talk Red Alarm *: The expression used to trigger a *red alarm** on the average talk time (eg. >20)
- **Avg. Talk Yellow Alarm *: The expression used to trigger a *yellow alarm** on the average talk time (eg. >20)
- **Tot. Talk Red Alarm *: The expression used to trigger a *red alarm** on the total talk time (eg. >20)
- **Tot. Talk Yellow Alarm *: The expression used to trigger a *yellow alarm** on the total talk time (eg. >20)

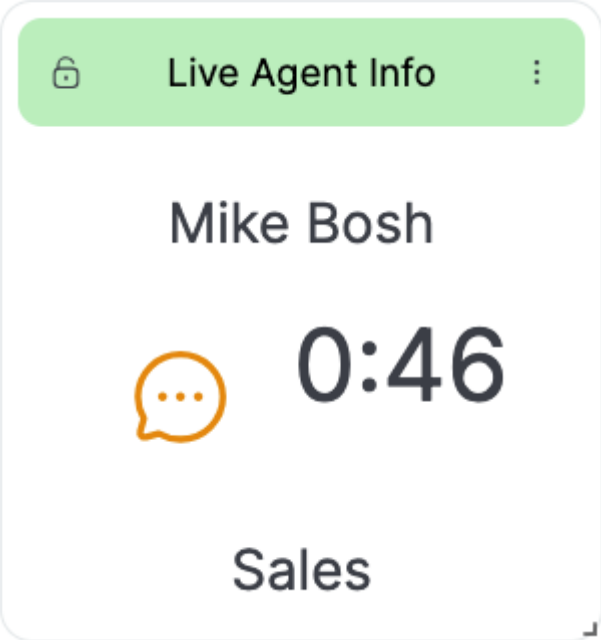
Available since	19.04.1
Name	Liva Agents Counter
Shortcut code	MI01
Parameters	-
See also	

MI02 - Live Agent Info

This Component is composed of three sections, each detailing a different information about an agent.

- The **agent's name** is shown in the **uppermost part** of the component.

- The **agent's state** is shown in the **middle part** of the component. To represent the agent's status there are different status badges. To read a description of the status you can hover on the status badge. Together with the status badge you will see a time counter, detailing how long the agent has been in that particular state.
- The **bottom part** of the component will show the **name of the queue** the agent is receiving a call from, if the agent is in a BUSY status. If the agent is on PAUSE, the **pause name** will be shown.



The following properties can be modified:

- **Title**: The title of the component
- **Pause Red alarm**: The expression used to trigger a **red alarm** on the pause time of an agent (eg. >20)
- **Pause Yellow alarm**: The expression used to trigger a **yellow alarm** on the pause time of an agent (eg. >20)
- **Call Red alarm**: The expression used to trigger a **red alarm** on the call time of an agent (eg. >20)
- **Call Yellow alarm**: The expression used to trigger a **yellow alarm** on the call time of an agent (eg. >20)
- **Idle Red alarm**: The expression used to trigger a **red alarm** on the idle time of an agent (eg. >20)
- **Idle Yellow alarm**: The expression used to trigger a **yellow alarm** on the idle time of an agent (eg. >20)

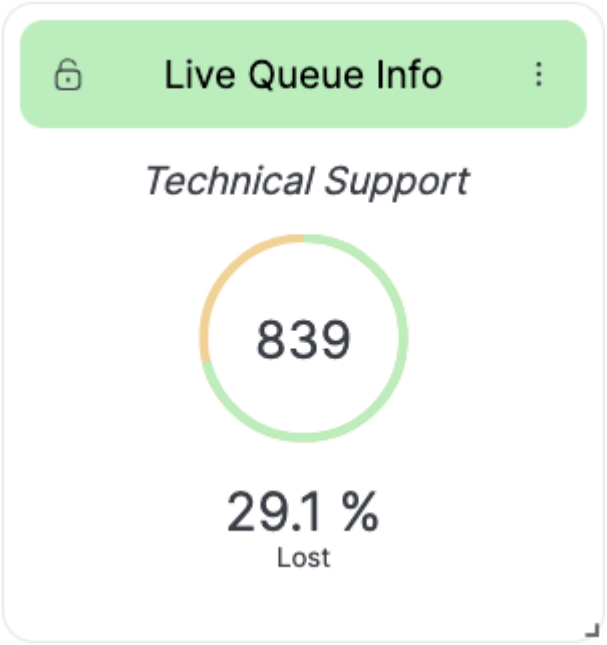
Available since	17.06
Name	Live Agent Info
Shortcut code	MI02
Parameters	-

See also	

MI03 - Live Queue Info

This Component is composed of three sections, each detailing a different information about a queue.

- The **queue's name** is shown in the **uppermost part** of the component.
- The **middle part** of the component is comprised of a circular panel detailing the number of **answered calls** on the selected queue.
- The **bottom part** of the component will show the percentage of **lost calls** on the selected queue.



The following properties can be modified:

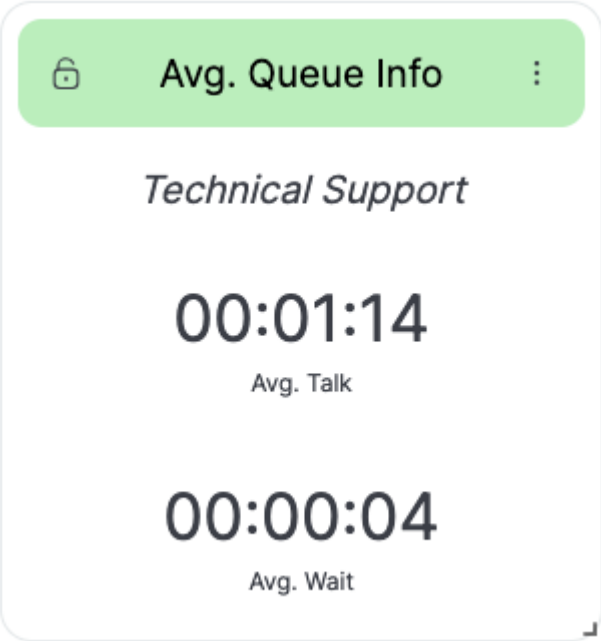
- **Title**: The title of the component
- **Red alarm**: The expression used to trigger a **red alarm** on the number of answered calls (eg. >20)
- **Yellow alarm**: The expression used to trigger a **yellow alarm** on the number of answered calls (eg. >20)
- **Color**: The color of the circular panel's stroke.
- **Secondary Color**: The secondary color for the circular panel's stroke.

Available since	17.06
Name	Live Queue Info
Shortcut code	MI03
Parameters	-
See also	

MI04 - Average Queue Info

This Component is composed of three sections, each detailing a different information about a queue.

- The **queue's name** is shown in the **uppermost part** of the component.
- The **middle part** of the component shows the **average talk** value for the selected queue.
- The **bottom part** of the component will show the **average wait time** for a call on the selected queue.



The following properties can be modified:

- **Title**: The title of the component
- **Talk Red alarm**: The expression used to trigger a **red alarm** on the average talk time (eg. >20)
- **Talk Yellow alarm**: The expression used to trigger a **yellow alarm** on the average talk time (eg. >20)
- **Wait Red alarm**: The expression used to trigger a **red alarm** on the average wait time (eg. >20)
- **Wait Yellow alarm**: The expression used to trigger a **yellow alarm** on the average wait time (eg. >20)
- **Color**: The color of the circular panel's stroke.
- **Secondary Color**: The secondary color for the circular panel's stroke.

Available since	17.06
Name	Average Queue Info
Shortcut code	MI04
Parameters	-
See also	

MI05 - Campaign Stats

This Component provides a high-level overview of the selected WombatDialer campaign status.



The following details are displayed:

- **Name**: Name of the selected queue/campaign
- **Status**: Current state badge (Running, Paused, Stopped)
- **Calls**: Calls Placed vs. Calls Terminated
- **Estimation**: Estimated calls remaining in the campaign

The following properties can be modified:

- **Title**: The title of the component
- **Queue**: The queue where the campaign is currently running (**MANDATORY**)
- **Red alarm**: The expression used to trigger a **red alarm** (eg. >20)
- **Yellow alarm** The expression used to trigger a **yellow alarm** (eg. >20)
- **Color**: The Widget color.
- **Secondary Color**: The secondary color.



This widget enter an "expired" mode if attribute data becomes stale or missing. (See. Expired Widget)

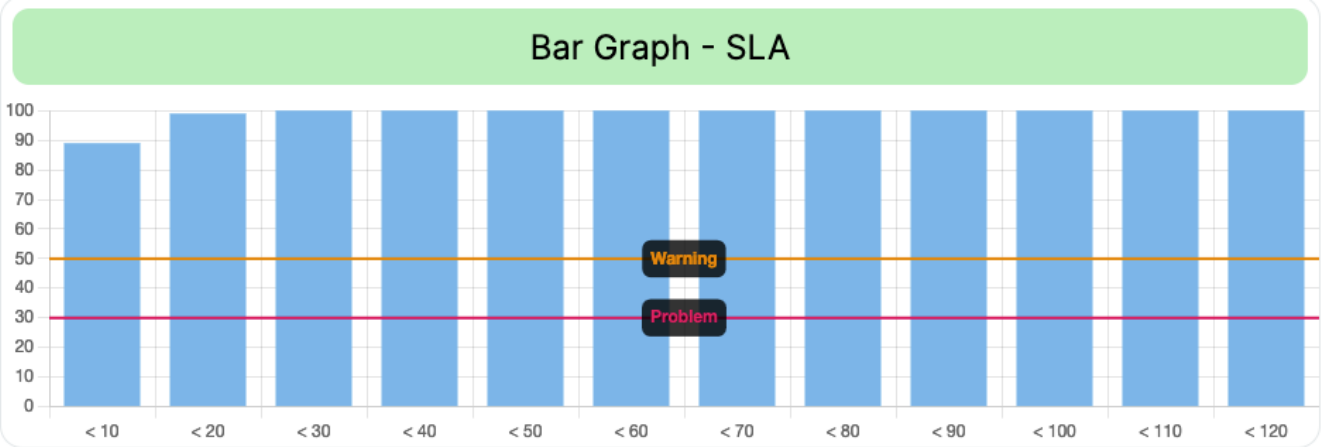
Name	Campaign Stats
Shortcut code	MI05

Attributes	WBT:CAMPAIGN_STATUS, WBT:CAMPAIGN_CALLS_PLACED, WBT:CAMPAIGN_CALLS_TERMINATED, WBT:CAMPAIGN_EST_LEFT
See also	WombatDialer

Charts

BC01 - Bar Graph SLA

This Component is a **Bar Chart** detailing the calls fulfilling of the setted SLA answering time requirements.



The following properties can be modified:

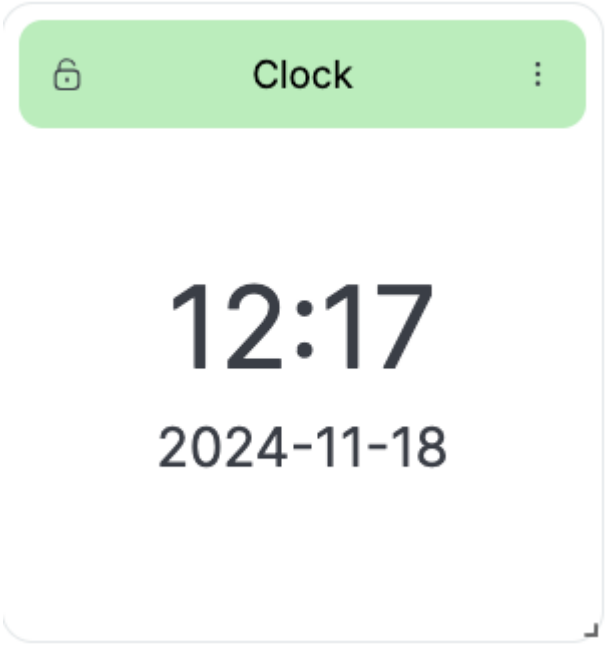
- **Title**: The title of the component
- **Color**: The color of the circular panel's stroke.
- **Yellow Line Limit**: A numerical value between 0-100 that set's the Y coordinates for a yellow limit line.
- **Red Line Limit**: A numerical value between 0-100 that set's the Y coordinates for a red limit line.

Available since	17.06
Name	Bar Graph SLA
Shortcut code	BC01
Parameters	-
See also	

Other

CL01 - Clock

This Component is a Clock component showing the current Date and Time.



The following properties can be modified:

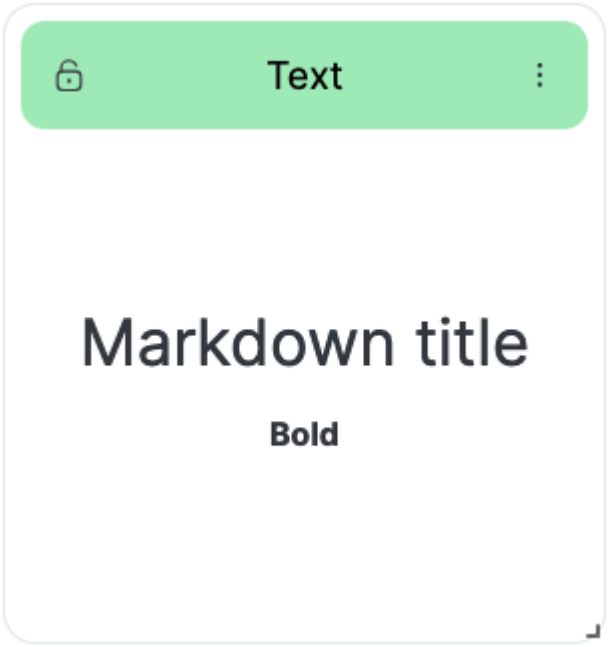
- **Title**: The title of the component

Available since	17.06
Name	Clock
Shortcut code	CL01
Parameters	-
See also	

TX01 - Text

This Component is a Text component showing a string of text. The string of text can be formatted in MD markup format, allowing the user to display images, gifs and links.

HTML tags are not supported



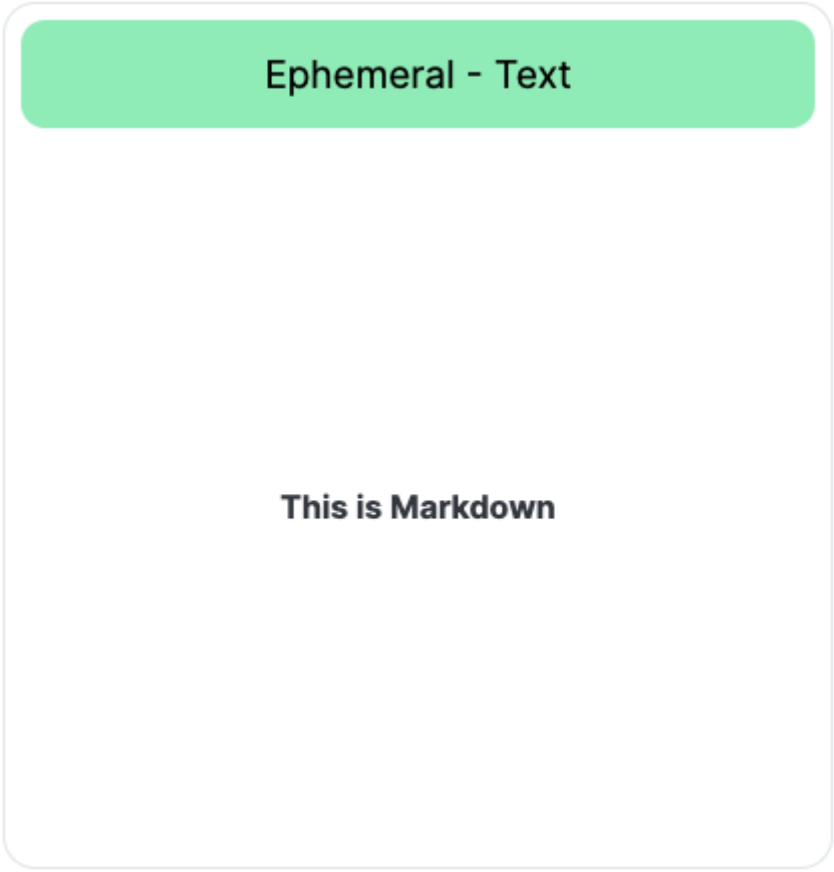
The following properties can be modified:

- **Title**: The title of the component
- **Text**: The text to display

Available since	17.06
Name	Text
Shortcut code	TX01
Parameters	-
See also	

TX02 - Ephemeral - Text

This Component displays Ephemeral attribute data in a text-based format or in markdown format.



The following properties can be modified:

- **Attribute**: Select the specific attribute to track. The "EphemeralDO". prefix is automatically prepended to the selection.(**MANDATORY**)
- **Queue**: Optional selector (can be left empty if the attribute is global)
- **Markdown**: Select whether the data should be rendered as Plain Text or Markdown



This widget enter an "expired" mode if attribute data becomes stale or missing. (See. Expired Widget)

Name	Ephemeral - Text
Shortcut code	IT05
See also	WombatDialer

HTML01 - HTML

This Component is an Iframe component showing any Web page that allows Iframes to display it. This widget requires the user to configure it by inputting a valid URL or it won't show anything.



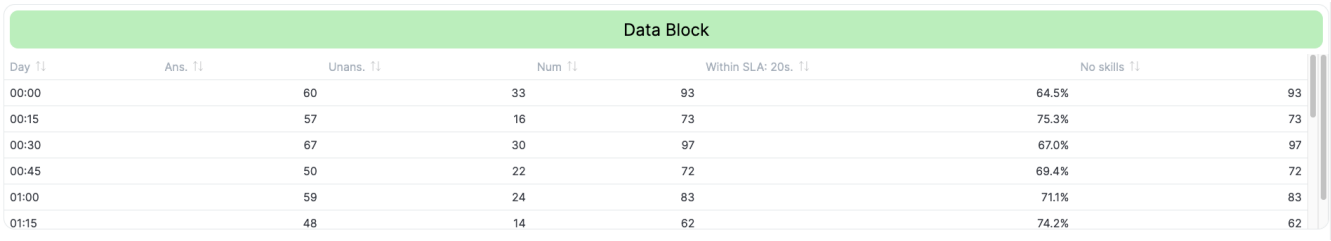
The following properties can be modified:

- **Title**: The title of the component
- **URL**: The URL of the web page to display

Available since	22.11
Name	Text
Shortcut code	HTML01
Parameters	-
See also	

DB01 - DataBlock Table

This component shows any DataBlock table available in QueueMetrics. It requires the user to select a valid DataBlock by configuring it, or it will show nothing.



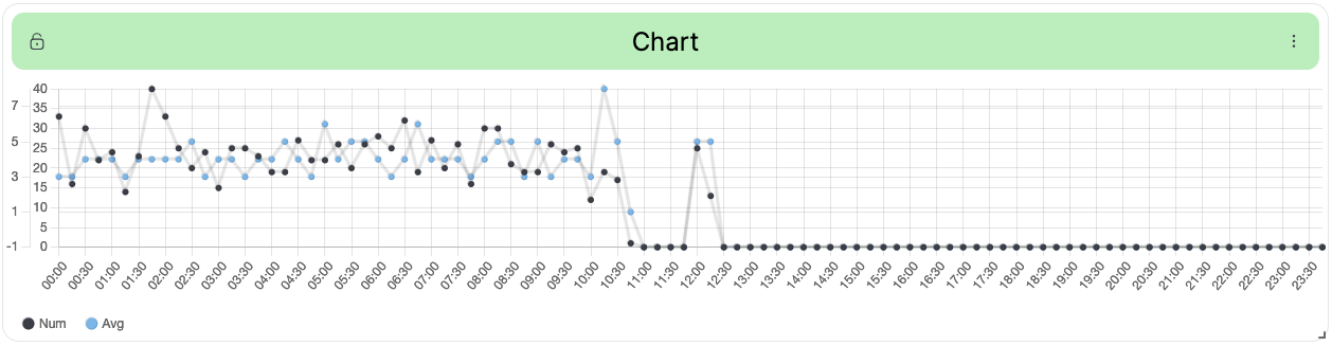
The following properties can be modified:

- **Title**: The title of the component
- **DataBlock**: The Source DataBlock
- **Columns**: The columns of the table that will be shown

Available since	22.11
Name	Text
Shortcut code	DB01
Parameters	-
See also	

DB02 - DataBlock Chart

This component shows any DataBlock char tavailable in QueueMetrics. It requires the user to select a valid DataBlock by configuring it, or it will show nothing.



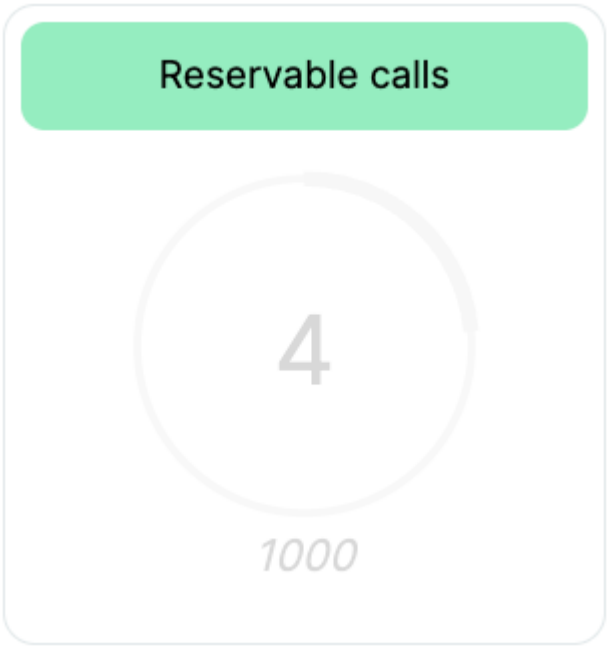
The following properties can be modified:

- **Title**: The title of the component
- **DataBlock**: The Source DataBlock

Available since	22.11
Name	Text
Shortcut code	DB02
Parameters	-
See also	

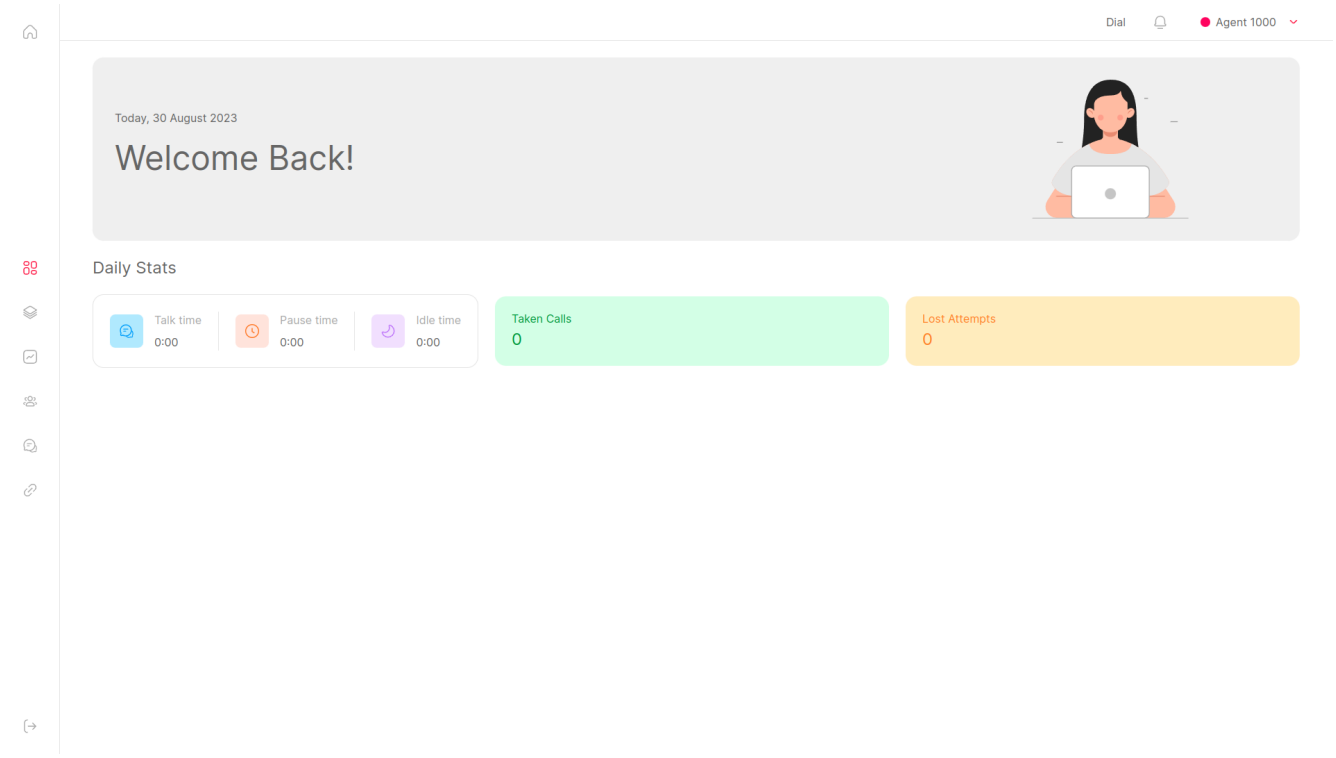
Expired widget

When the attribute data requested by a widget becomes stale or missing, the widget will transition to grayscale with reduced opacity to indicate that it is in an *expired* state.



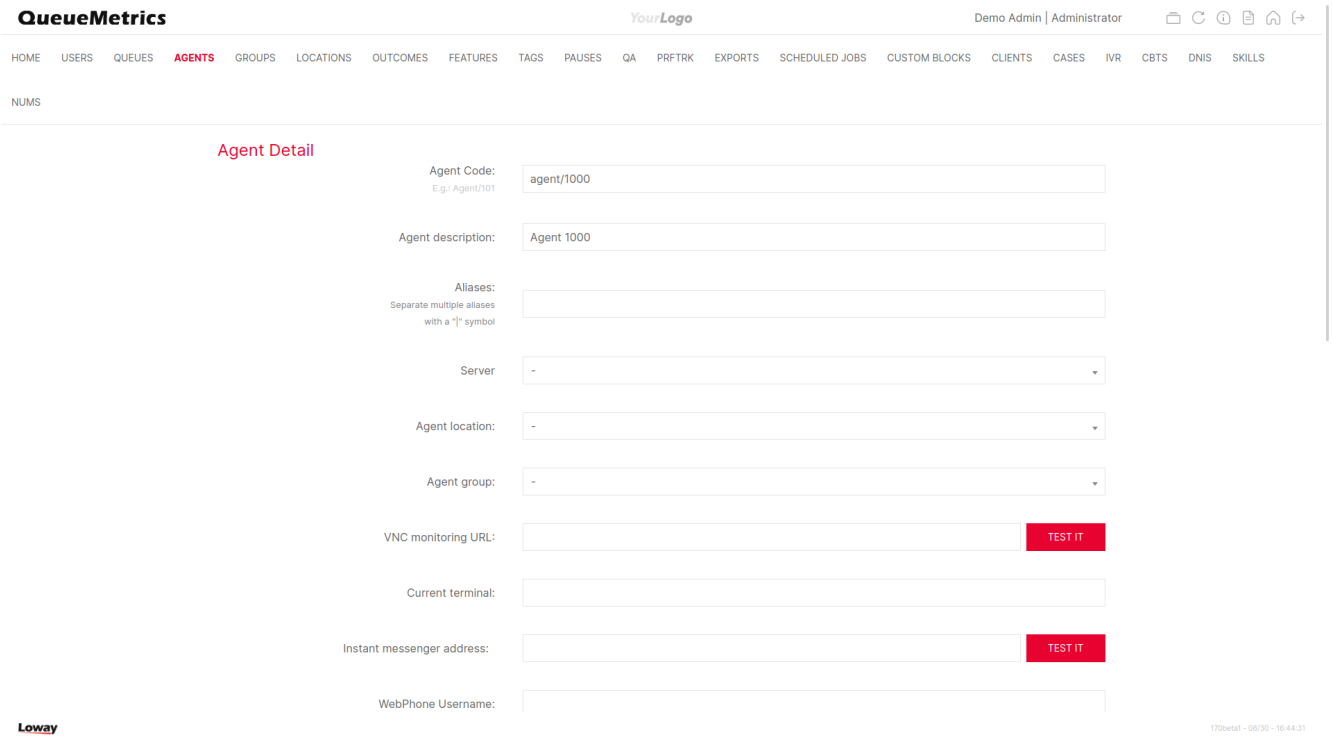
Agent page

Through QueueMetrics' agent interface, agents can manage queue presence and handle their complete workflow directly.

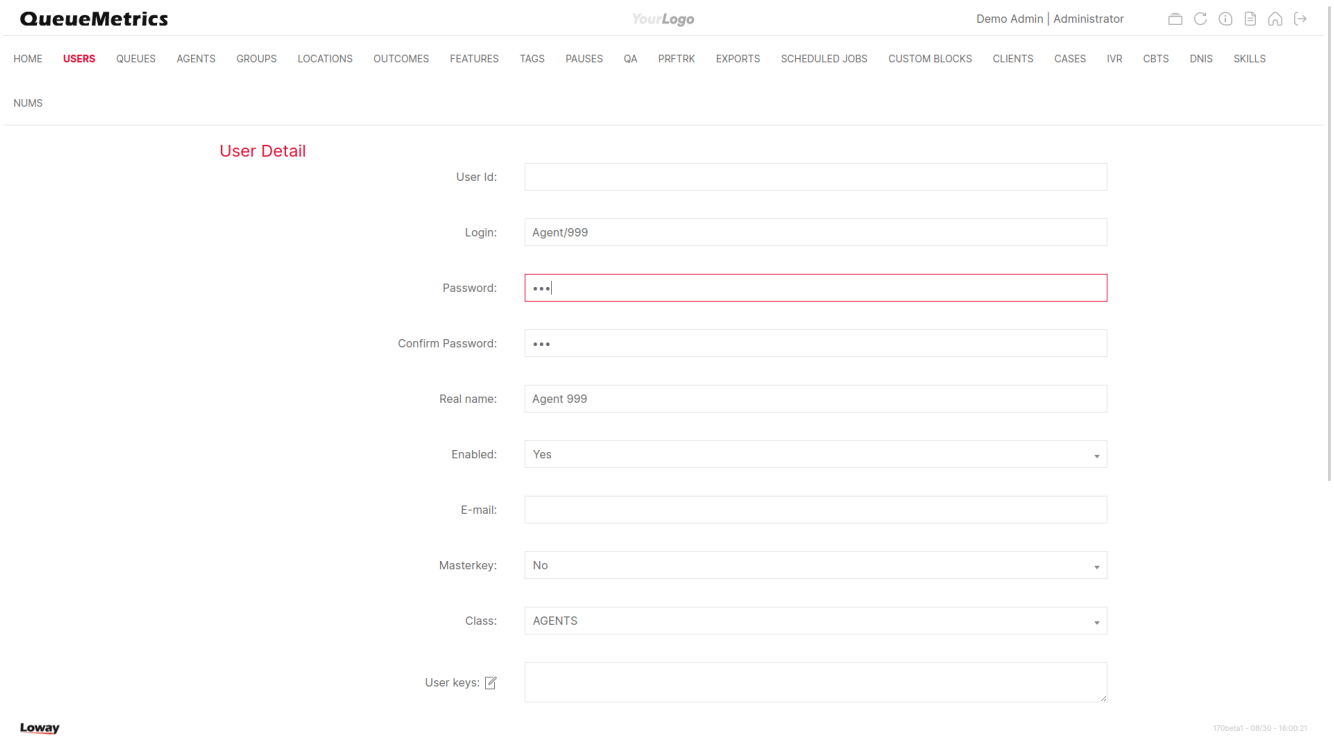


Login

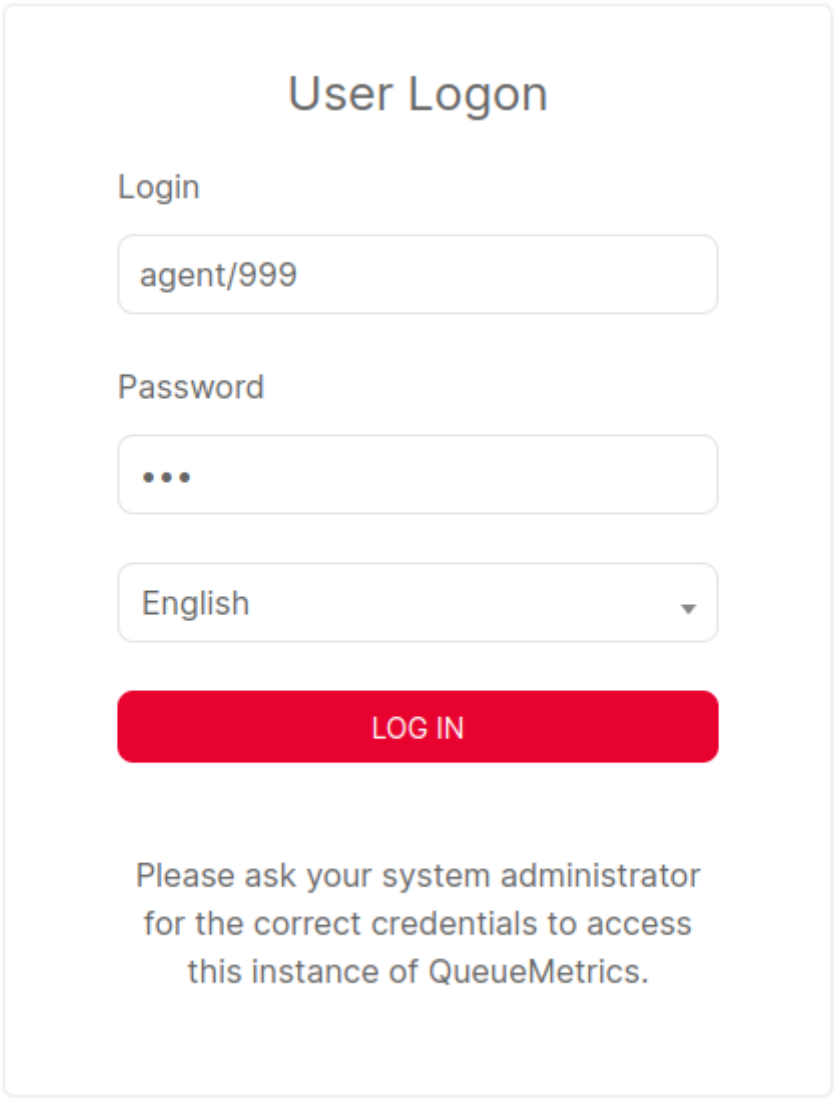
To login into the Agent Page, you will need to configure an Agent User in QueueMetrics. In order to do this, please create an Agent and a User, both with the same Agent Code.



Remember to make sure that the User class for this User is AGENTS.

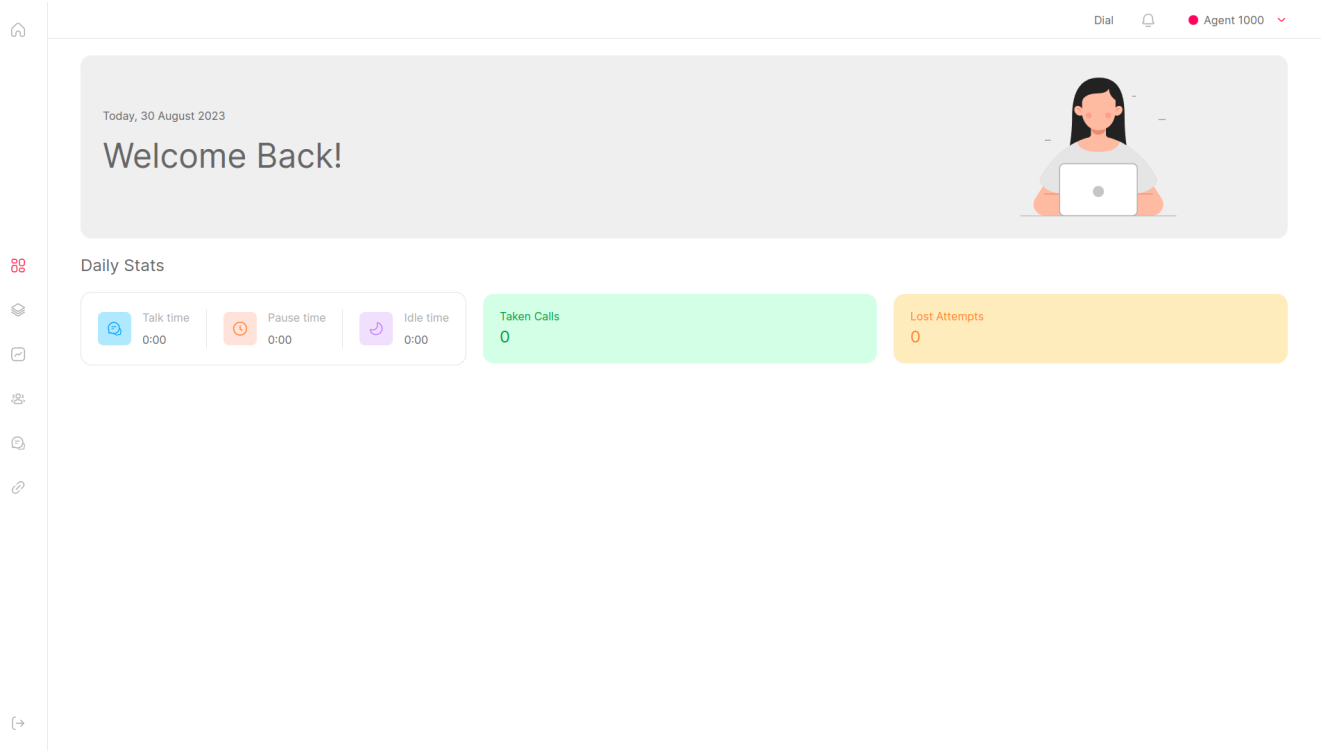


Once you have created the Agent and User, please log out and log into QueueMetrics using your new credentials.



Overview

The first view that will show when looking into the Agent Page, is the Overview.



The Overview will present the Agent with a rundown of their Daily Stats. Namely:

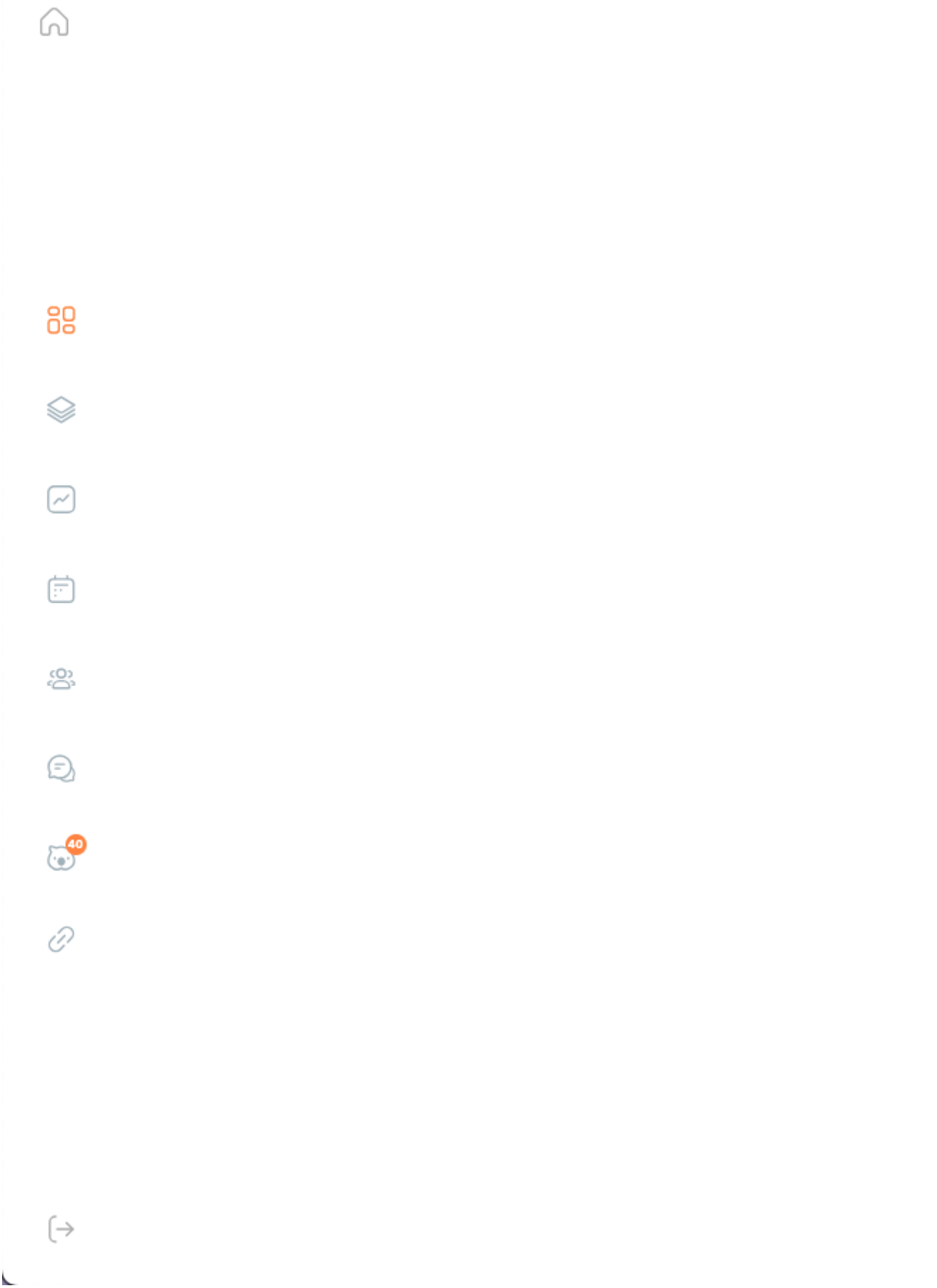
- Talk Time: How much time the agent has spent on call today.
- Pause Time: How much time the agent has spent on pause today.
- Idle Time: How much time the agent has spent in an idle state today. Idle time is defined as time the Agent was available to receive calls on at least one queue, without being on a break.
- Taken Calls: How many calls the agent has answered today.
- Lost Attempts: How many times did the PBX try to call the Agent with no success.

Start/End Work

The Start/End work button is a quick way for the agent to start working. This button will only appear if the agent has logged in in the past, and QueueMetrics knows which extension the Agent uses.

By clicking on this button, the agent will be taken to the Control view, and automatically join all the queues they are assigned to. An Administrator can assign queues to an agent from the Queue Configuration page.

Navigation Menu



The Navigation Menu on the left side of the screen allows the Agent to move between different views. There are different available views, listed below:

- Overview: Shows a general overview of the Agent State and Stats.
- Control: Allows the Agent to join/leave queues, and allows them to pause/unpause themselves.
- Interactions: Shows the list of Calls/Interactions the agent has handled throughout the day, with actions.
- Clients: Search engine for Clients and Client creation/editing.
- Cases: Search engine for Clients and Client creation/editing.
- WombatDialer Preview Dialing: If a valid URL for a WombatDialer instance, a WombatDialer icon will appear, allowing the agent to access the Preview Dialing Panel. (If some call can reserved on the queues you are logged in, the sum of those will be display as a counted badge on on the top right corner of wombat icon)
- CRM: Shows the user-defined links to external CRM systems.

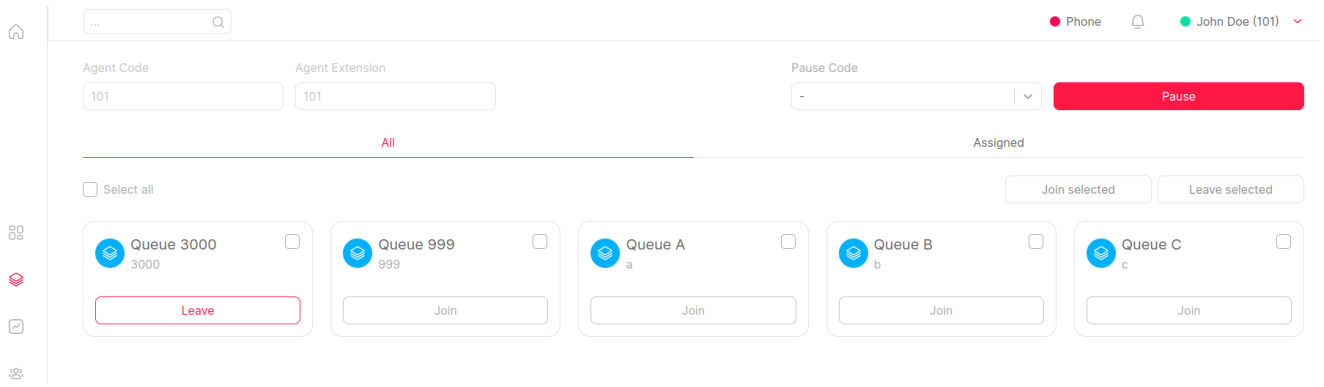
The Navigation menu also has a logout button at the bottom, that will log the agent out of the Agent Page.



Logging out of the Agent Page does NOT log the agent out of any queues he might be working on. Make sure the Agent logs out of the queues from the Control view before logging out of the Agent Page, if they don't plan on being available on queues any longer. Unless you set the property agent.leaveAllQueuesOnLogoff to true.

Control

The Control view is the main view for the Agent to join/leave queues, check their current status and pause/unpause themselves, as well as picking which extension they want to use.



Extension

By changing the extension field, an Agent will be able to change which extension they are using when joining queues. Agents can change their extension only if hotdesking is enabled in the System Parameters.



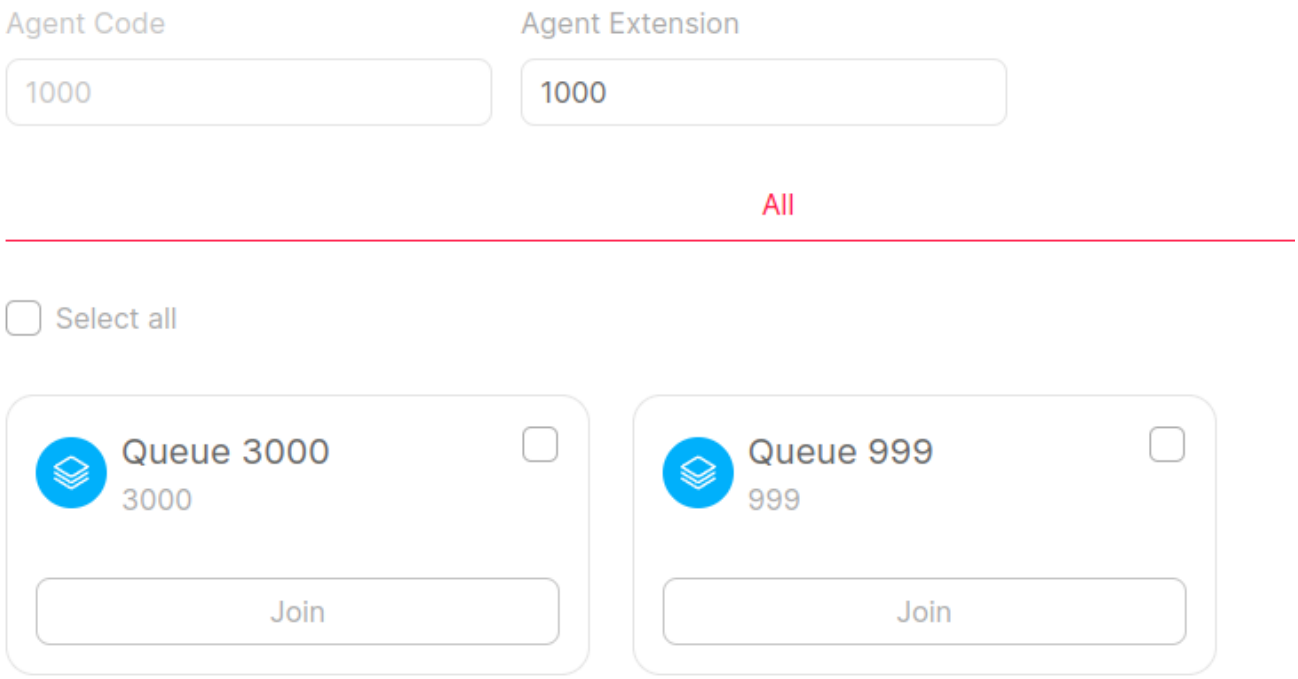
If this field is empty, it means that QueueMetrics doesn't know the Agent's extension. Without this information, it will be impossible for QueueMetrics to let the Agent join queues, so Agents should always ensure their extension is properly

set.

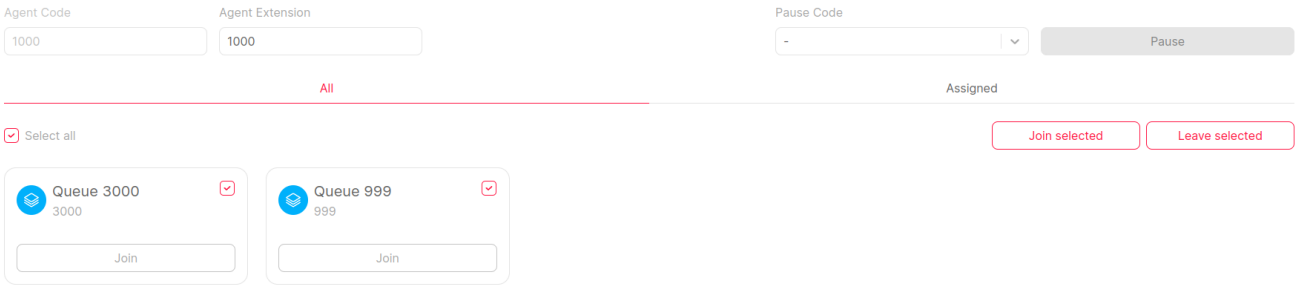
Joining queues

Once the Extension is set, the Agent can join one or more queues by either

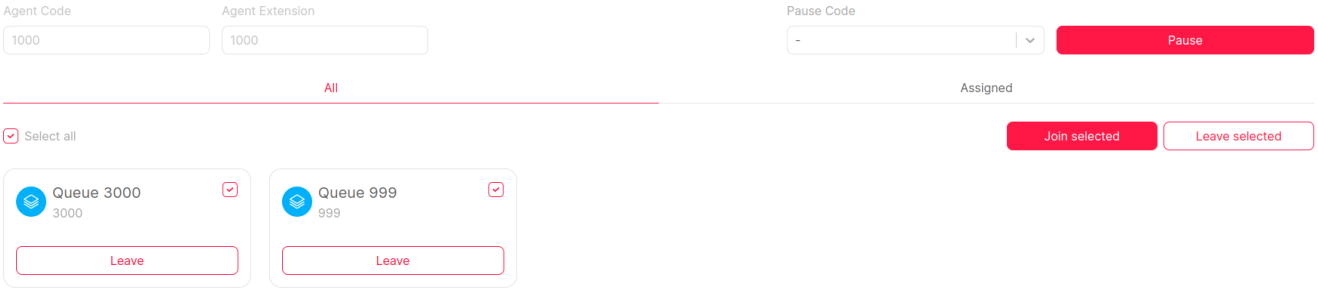
- Clicking on the Join button for each individual queue they want to join.



- Selecting queues using their checkboxes (or the Select All checkbox on the top left), then clicking on "Join Selected".

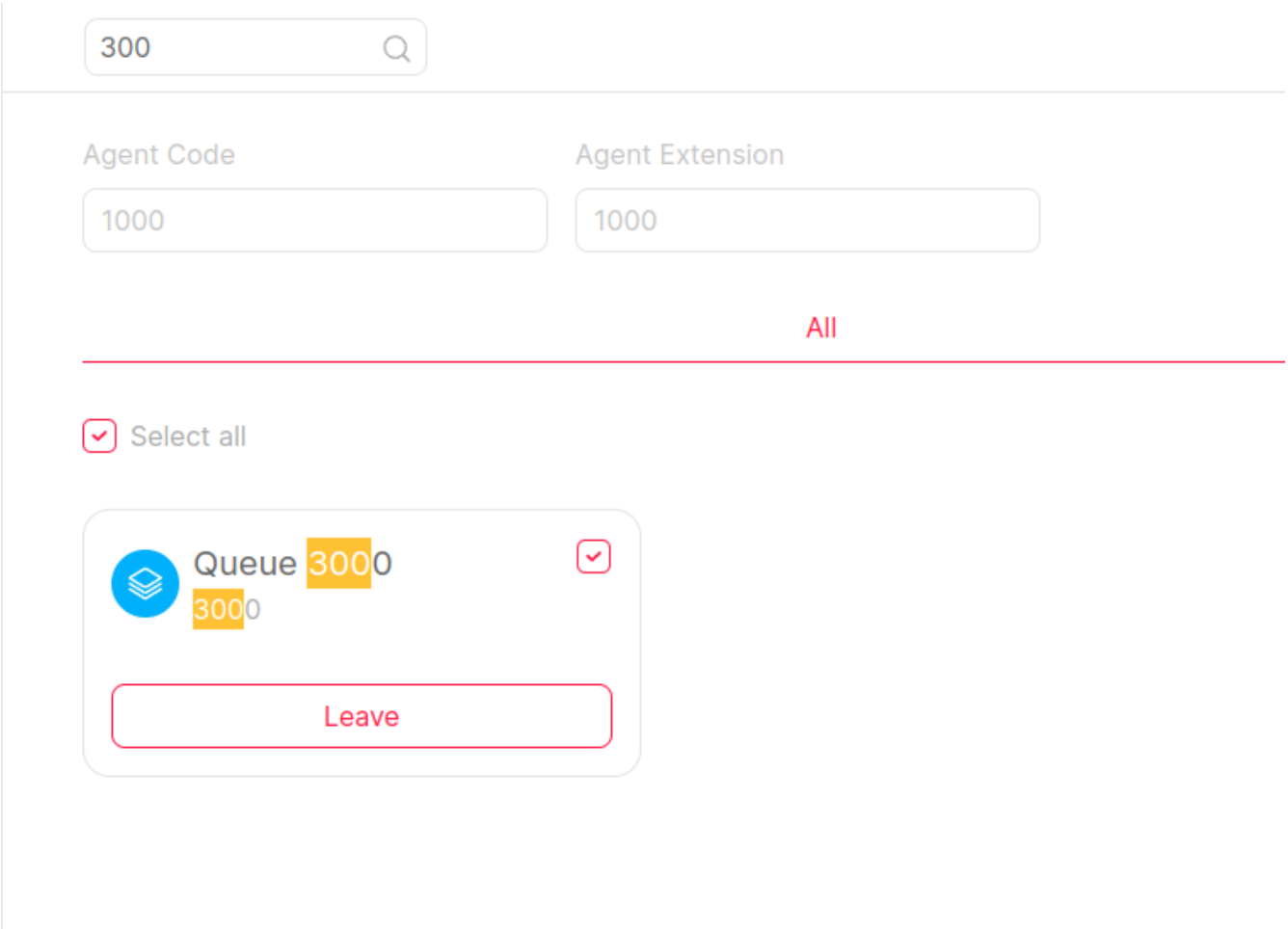


By doing this, QueueMetrics will send a Join request to the PBX, and will update the Agent's status on said queues as soon as it obtains a response from the PBX.



If the agent has too many queues, they can always use the searchbar in the top left to filter the queues that appear in the page.

Queues on which the agent is working have a green border to make them easily identifiable.



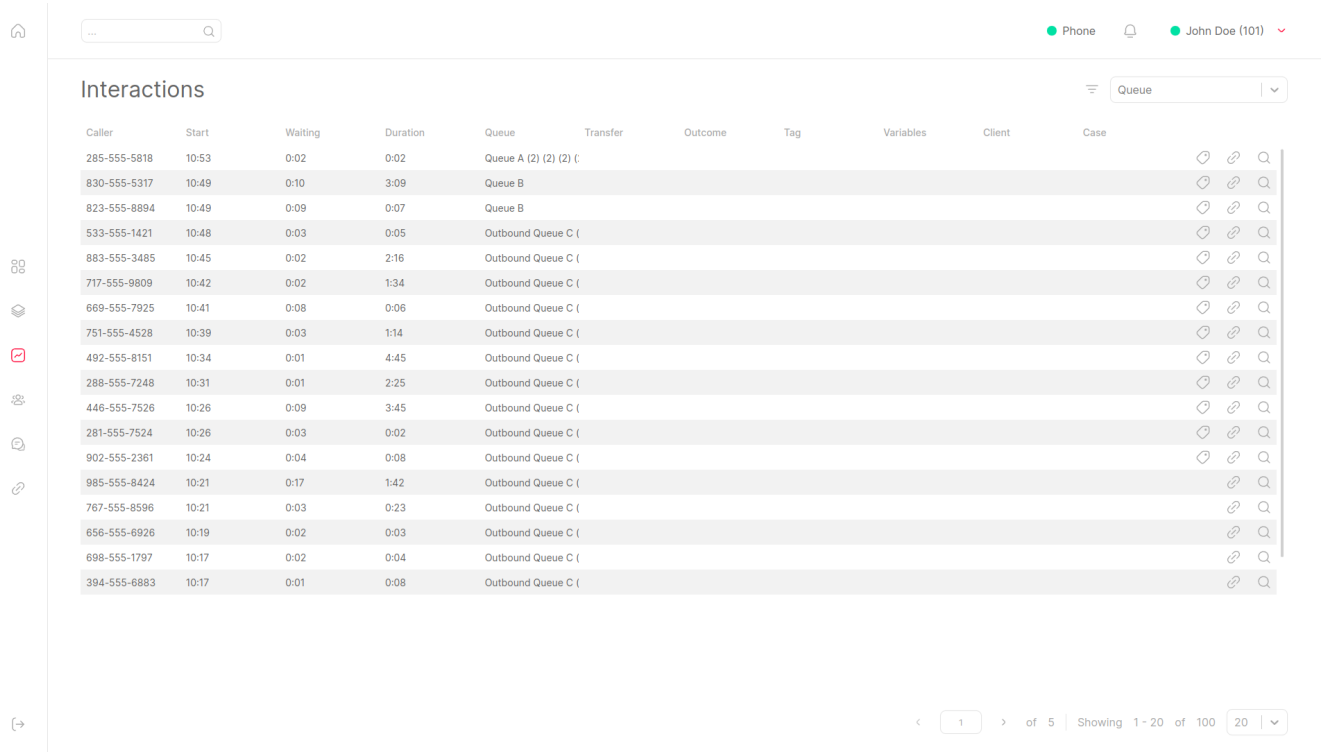
Changes to queue membership and pause status take effect immediately, as long as the agent isn't busy in a conversation or on an outbound call. If the agent **is** busy, the change won't take effect until the current call ends. In this case, QueueMetrics keeps the agent's current status active until the call terminates.

For example, suppose an agent's shift ends at 18:00, but they're on a call that started at 17:59 and runs until 18:05. Even if they click "Queue log off" at 18:00, the change won't be reflected until the call ends at 18:05.

This behavior applies consistently across the agent page, wallboards, and reports.

Interactions

The Interactions view provides agents with a comprehensive overview of all the interactions they have handled during the current day.

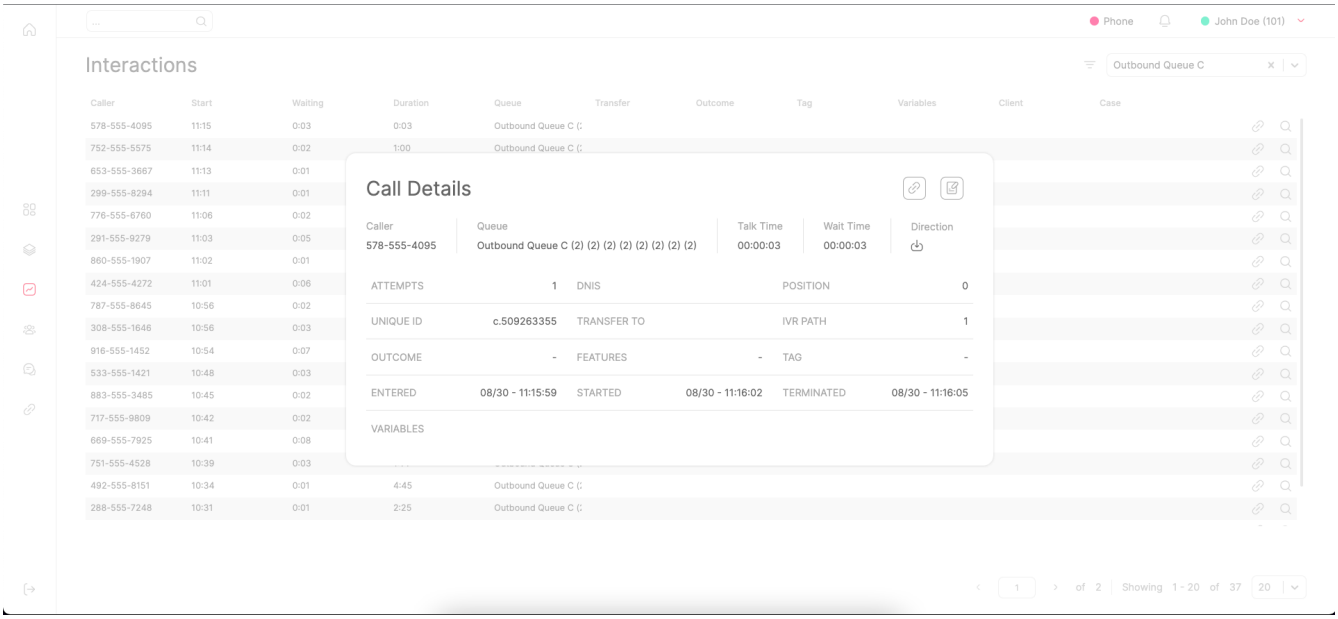


For each call, QueueMetrics presents detailed information.

To the right of each entry, you will find three buttons:

1. `label` icon: This button allows you to tag a call with an outcome and one or more feature codes.
2. `url` icon: Clicking this button opens the call's associated URL.
3. `magnifier` icon: This button provides access to the call's detailed information.

Interaction Details



In this section, users can access comprehensive details about the interaction they are reviewing, which includes the following information:

- Caller
- Queue
- Talk Time
- Wait Time
- Direction
- Number of Attempts
- DNIS (Dialed Number Identification Service)
- Position
- Unique ID
- Transfer To
- IVR Path
- Outcome
- Features
- Tags
- Time Entered
- Time Started
- Time Terminated
- Custom Variables

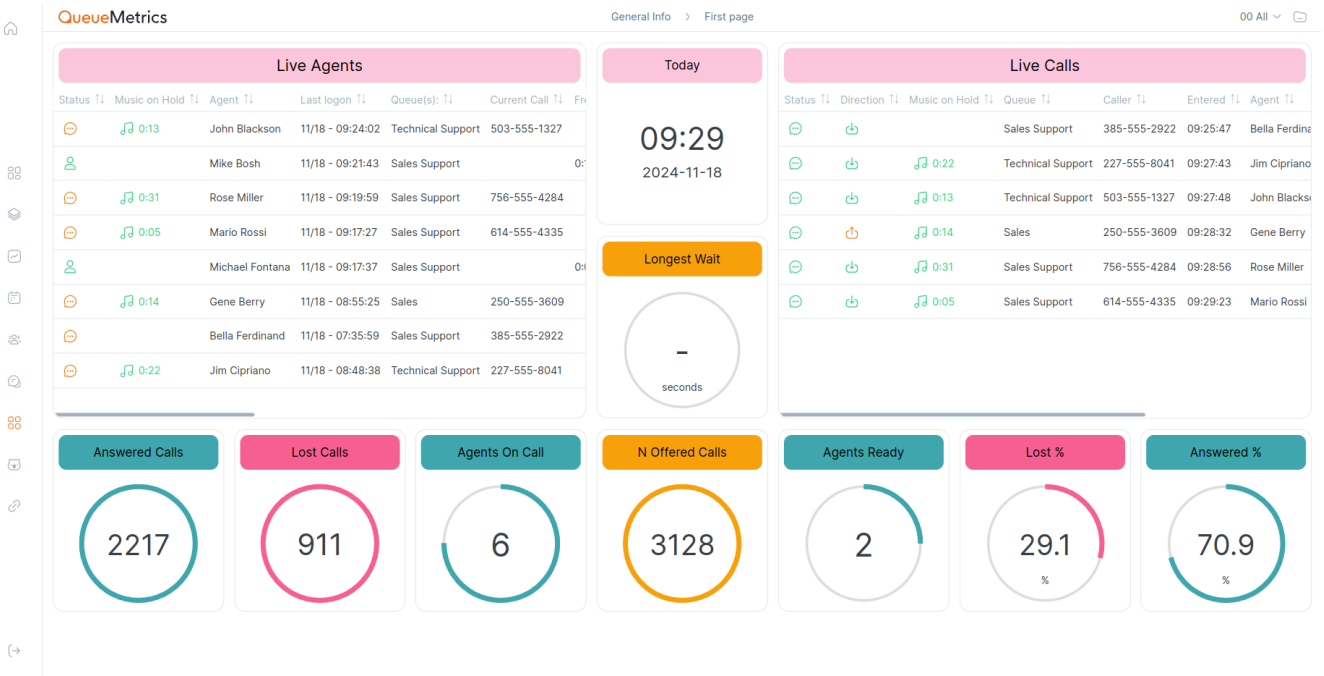
From this interface, users can perform various actions on interactions, such as:

1. Open the QA Form
2. Label a Call

3. Open the URL
4. Assign the Call to a Client
5. Assign the Call to a Case
6. Initiate a Wombat Recall

Wallboard

QueueMetrics 24.11 integrates the QueueMetrics Wallboard with the agent page, allowing agents with the appropriate permissions to view the Wallboards they have access to, which gives them visibility into the activity of the queues they are working on.

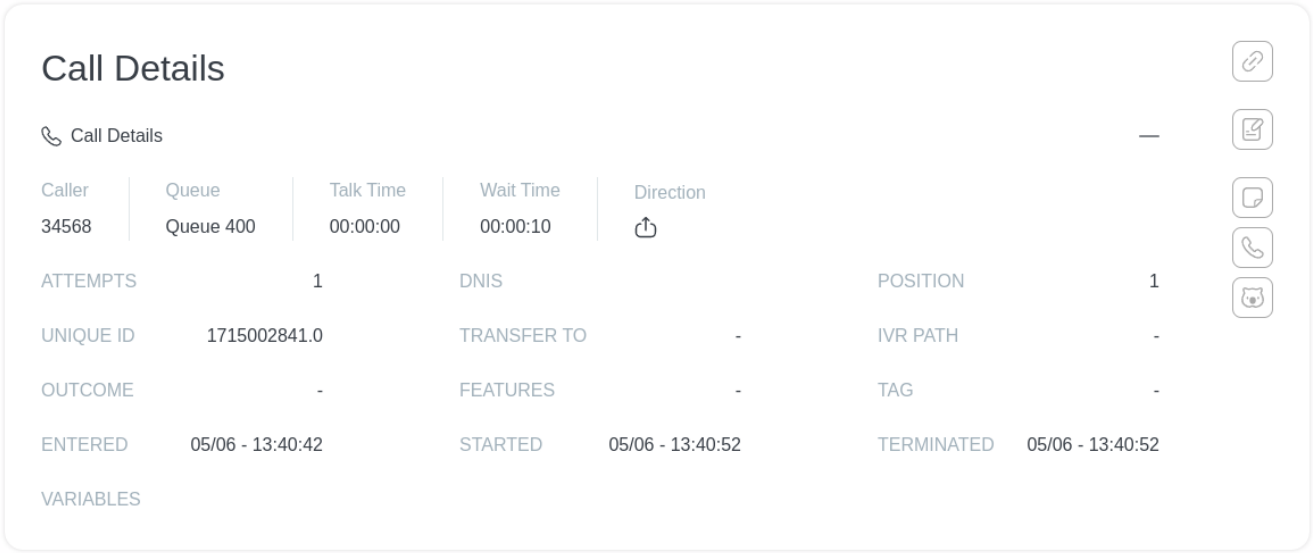


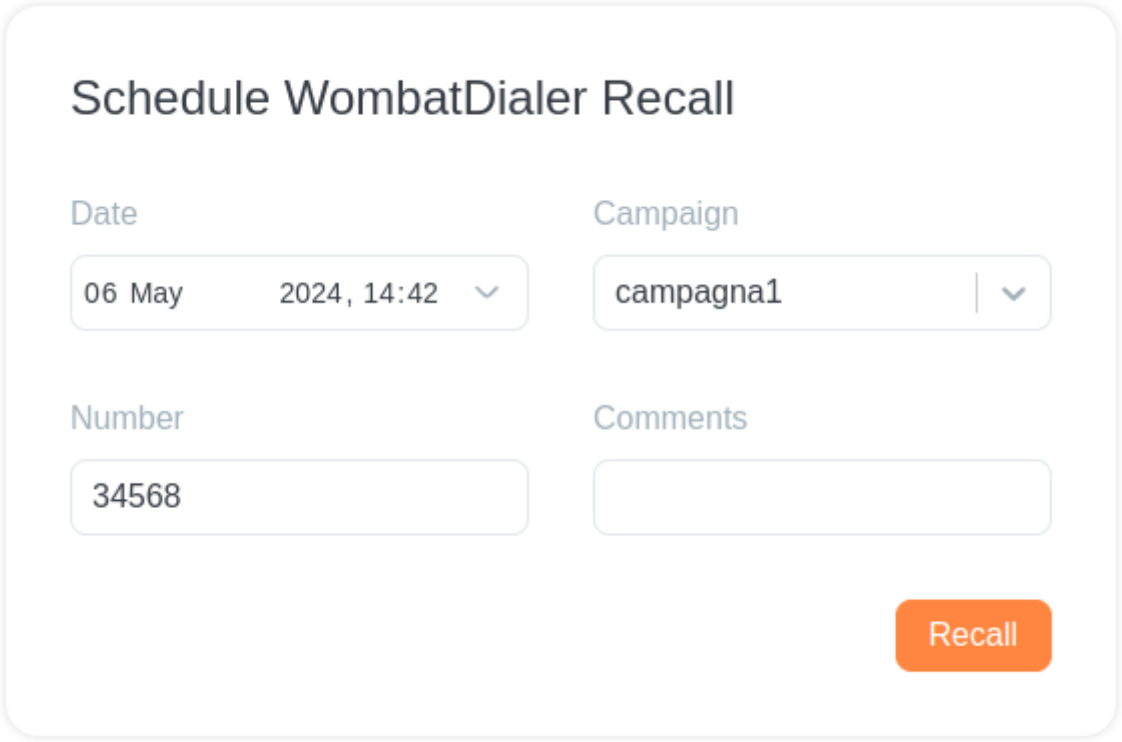


In order to access the Wallboard, agents need to hold the security key: REALTIME. This will show the wallboard icon in the navigation menu on the left side.

WombatDialer Recalls

QueueMetrics 24.05 introduces the WombatDialer Recall Panel, which enables the user to schedule a recall for the examined call through Loway's Dialer: WombatDialer. To initiate a recall, press on the WombatDialer recall at the bottom of the right menu of icons.





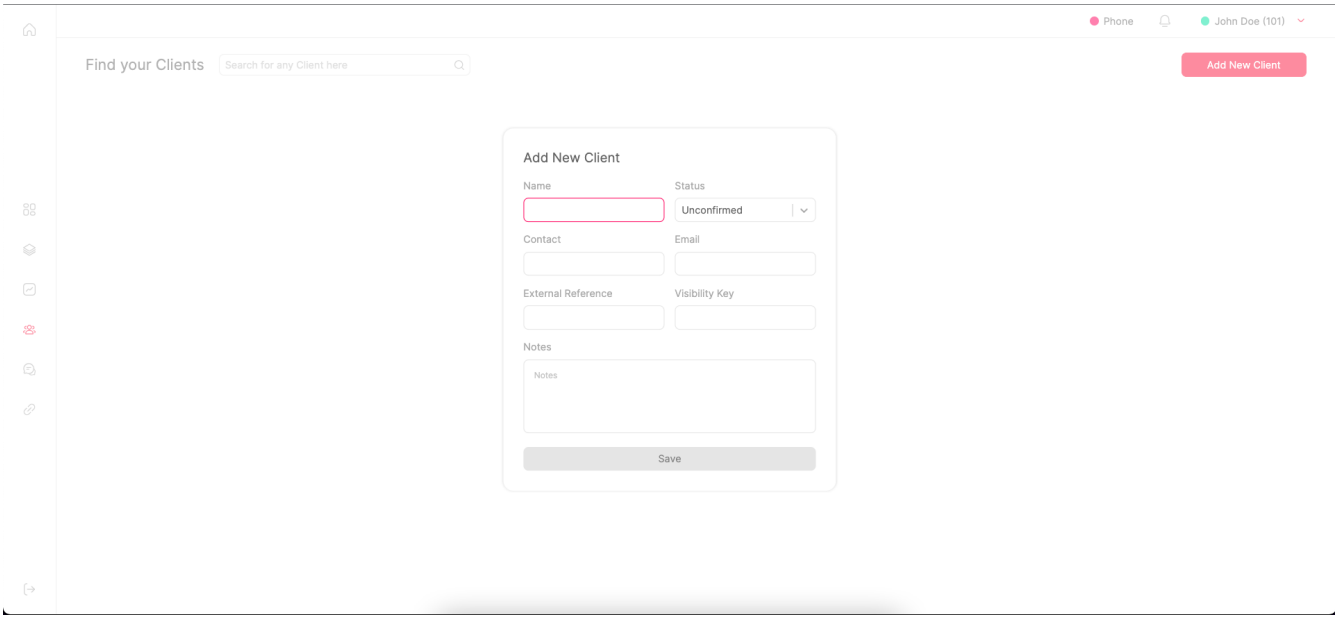
Clients and Cases

QueueMetrics 23.09 introduces two new concepts to QueueMetrics: Clients and Cases. While the description below addresses how they look like from an agent's point of view, an administrator's point of view is available in Configuring clients and cases.

Clients

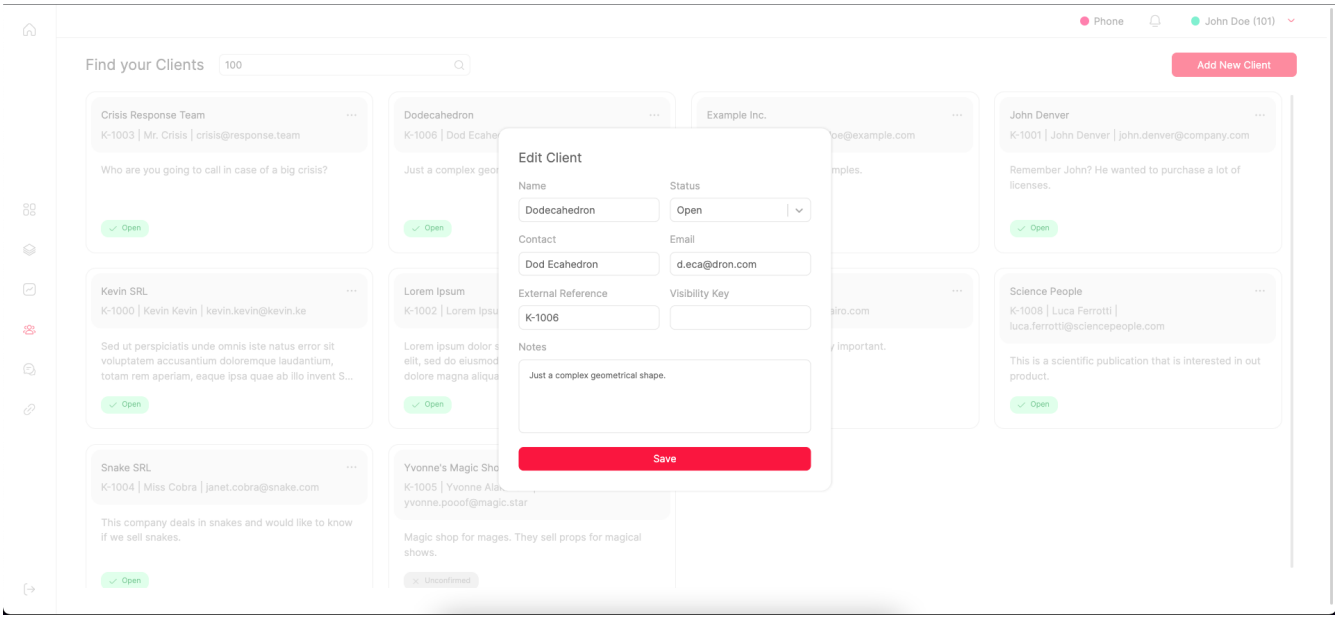
To create/edit Clients, users need the key `USR_CLIENTS_EDIT`.

Clients are data structures that can be created to save information about customers, functioning as a minimalist CRM inside QueueMetrics.

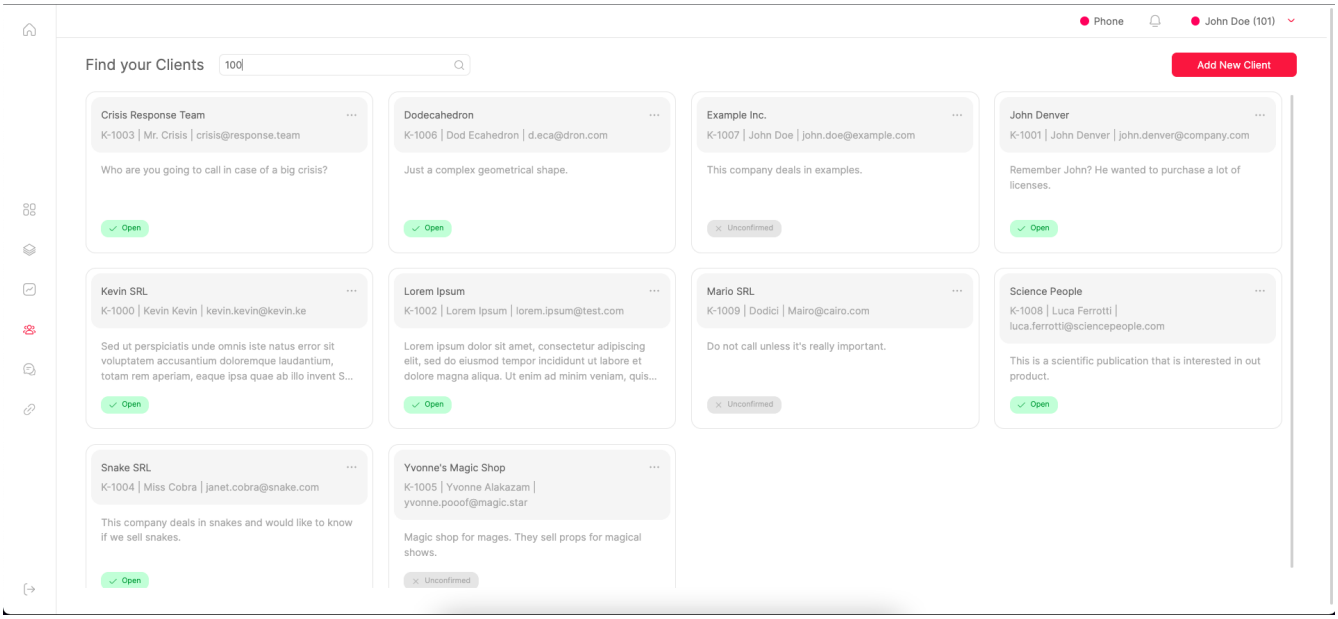


The information that can be saved about Clients is the following:

- Name
- Status (Open, Closed, Unconfirmed)
- Contact
- Email
- External Reference (Will be filled automatically if left blank)
- Visibility Key
- Notes



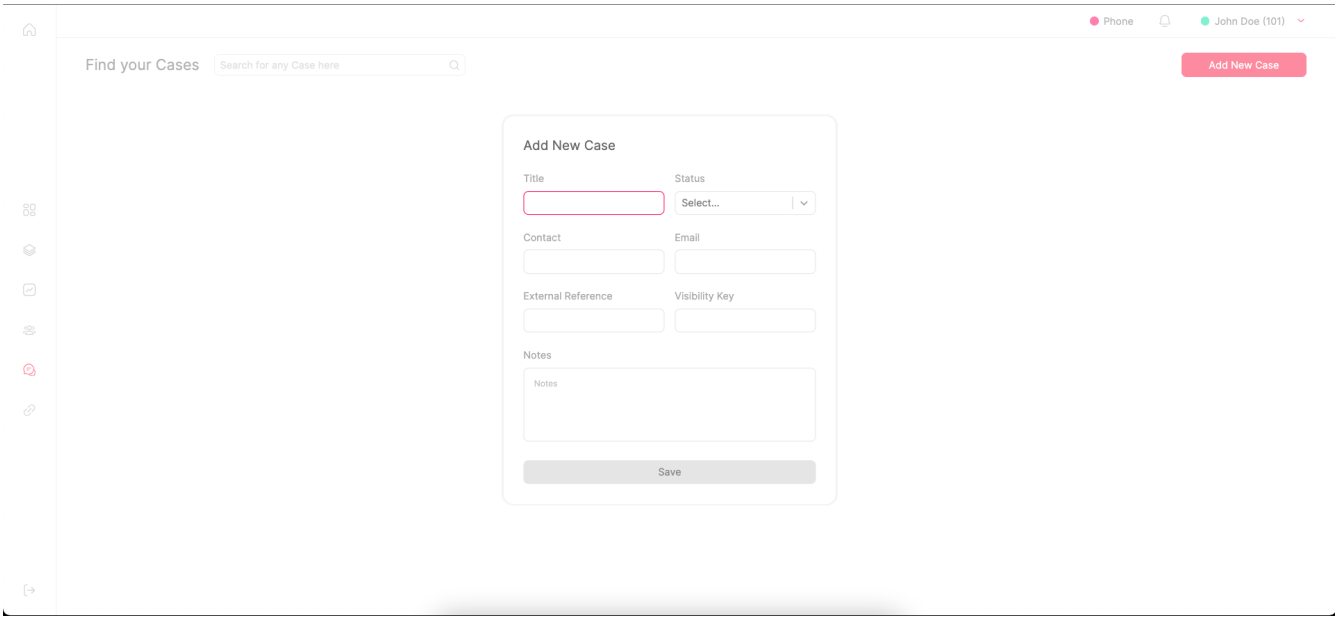
Once a client has been created, it can be accessed by searching for clients in the search bar, and edited by clicking on the card.



Cases

To create/edit Cases, users need the key `USR_CASES_EDIT`.

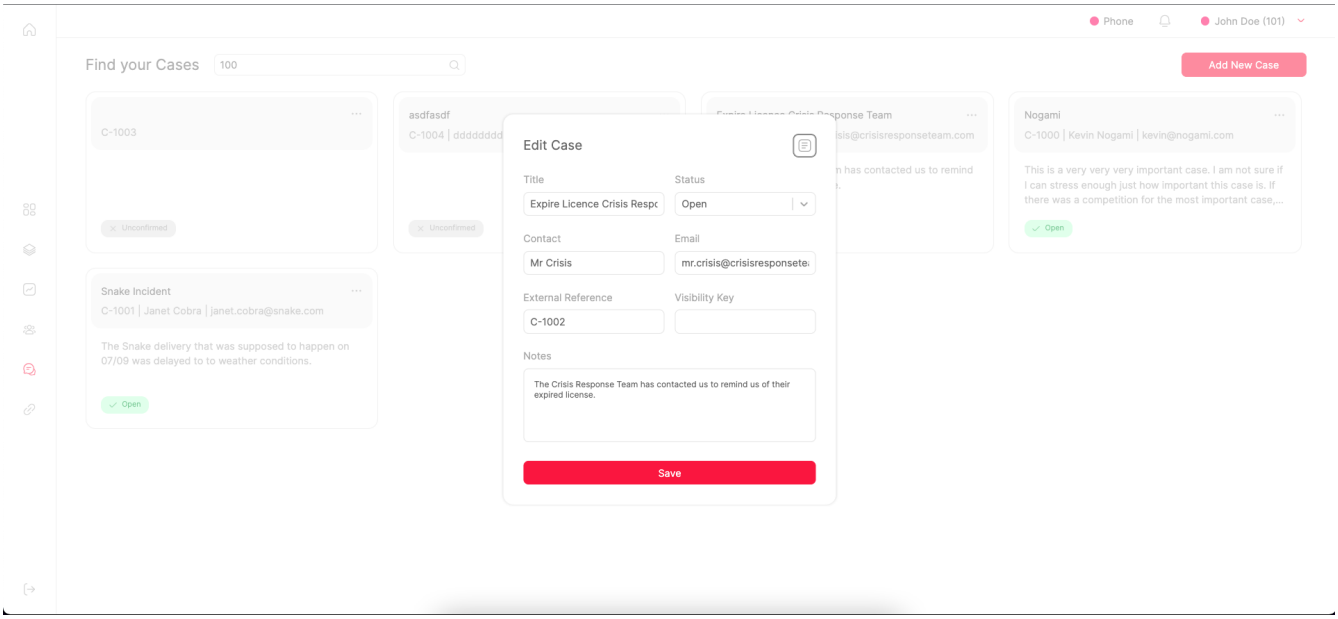
Cases are data structures that can be created to save information about specific situations, cases, or anything that involves one or more calls, functioning as an entity for the minimalist CRM inside QueueMetrics.



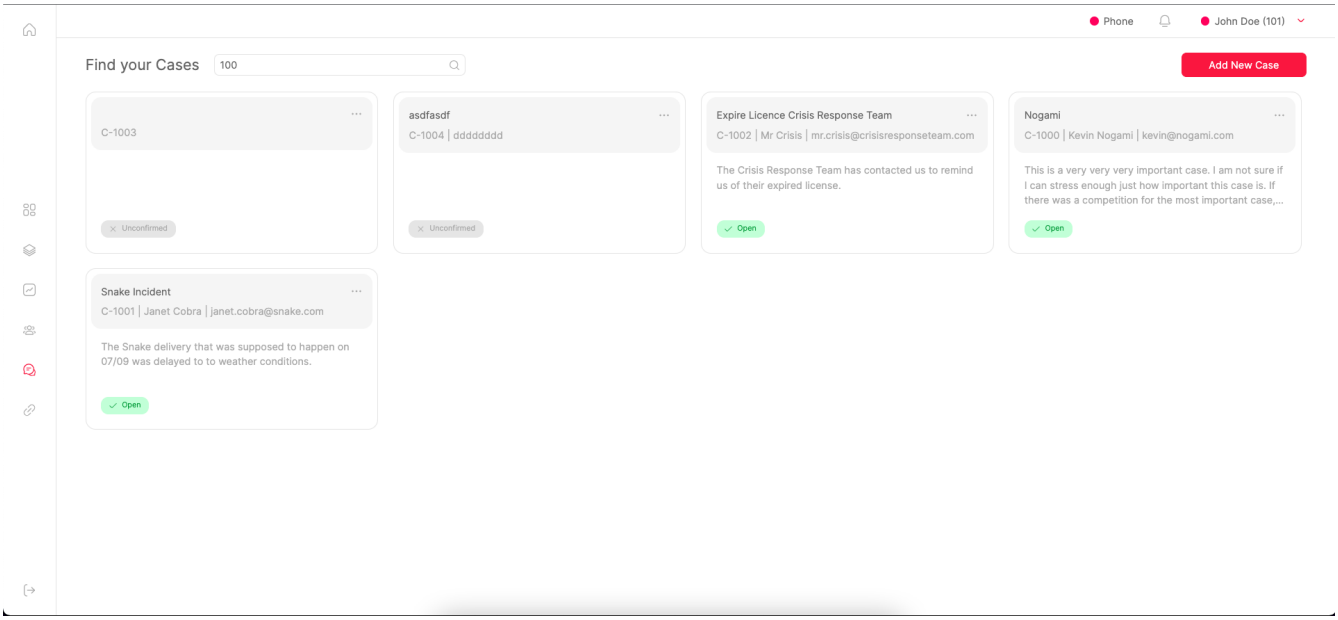
The information that can be saved about Clients is the following:

- Name
- Status (Open, Closed, Unconfirmed)
- Contact
- Email
- External Reference (Will be filled automatically if left blank)
- Visibility Key

- Notes



Once a case has been created, it can be accessed by searching for clients in the search bar, and edited by clicking on the card.



CRM Integration

In QueueMetrics, you can seamlessly integrate your CRM (Customer Relationship Management) system by configuring CRM links through the navigation menu. These links are defined in QueueMetrics' system parameters.

The properties that regulate these CRM links are as follows:

- `realtime.agent_web1_url`: This property specifies the URL used as the first CRM URL in the QueueMetrics Agent page. The key can be populated with a URL where certain tokens will be expanded by the QueueMetrics engine. Valid tokens include [A], which will be expanded with the agent code. See also `default.crmapp` and `default.crmlabel`.

- `realtime.agent_web1_label`: If present, this property is used by the Realtime Agent page to identify this specific background on the dropdown menu.
- `realtime.agent_web2_url`: Similar to the first URL, this property specifies the URL used as the second in the Realtime Agent page. It can also contain expandable tokens like [A].
- `realtime.agent_web2_label`: If present, this property is used to label the second background on the dropdown menu within the QueueMetrics Agent page.

These configurations allow you to link to CRM resources directly from QueueMetrics, enhancing your workflow and productivity.

Screen-pops to your CRM application

Each time the call history changes, and if there is an URL associated to the call shown on top of the list, a new window will automatically be opened pointing to the specific URL. This feature could be disabled modifying the value associated to the configuration key *realtime.agent_autoopenurl*.

URLs are passed to the Queue() command in Asterisk, or - if missing - they can defined in the QueueMetrics queue configuration as a default. If not present, the URL defined in the *default.crmapp* property will be used (make sure the *default.crmlabel* property is also defined). The URL can include a set of placeholders that are expanded with the details of the call being handled, as in the following table.

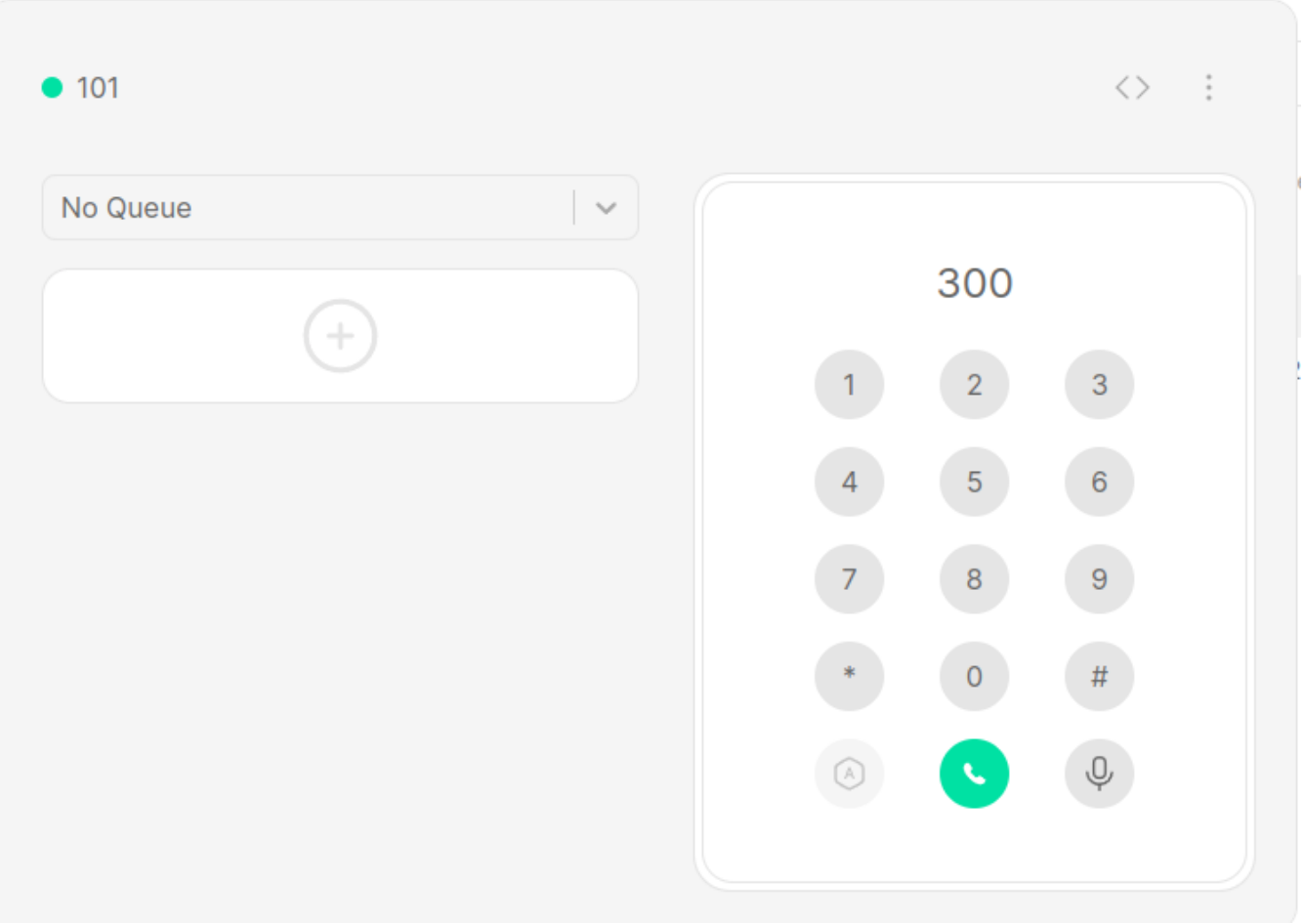
Placeholder	**Meaning**
[A]	The agent's numeric code, e.g. *101* for *Agent/101*
[X]	The agent's login extension
[U]	The call's Asterisk UniqueId
[S]	The Server ID (for clusters)
[Q]	The name of the queue
[T]	The timestamp of the call
[C]	The caller-id, if present
[D]	The DNIS, if present
[I]	The IVR Path, if present
[O]	The outcome code, if present
[P]	The position in the queue, if present
[M]	The number of attempts, if present
[TAG]	The call tag, if present. See Adding Tags to a call.
[F:x]	If the call has a feature called *x*, the text of feature *x*; otherwise blank. See Adding Feature codes to a call.

Placeholder	Meaning
[FX:x]	If the call has a feature called *x*, it is set to *1*; otherwise *0*
[V:x]	If the call has a variable called *x*, the value of *x*; otherwise blank. See Custom variables.

SoftPhone

The new real-time agents page integrates a softphone based on the JsSIP library. This softphone is expected to work with all recent Asterisk releases (13+) and recent browsers.

The softphone automatically registers to the SIP/WebRTC server with the SIP credentials specified in the user administration page and a green bullet is shown in the softphone panel. The agent can start calls typing the number to be dialed then clicking on the "Start Call" button.



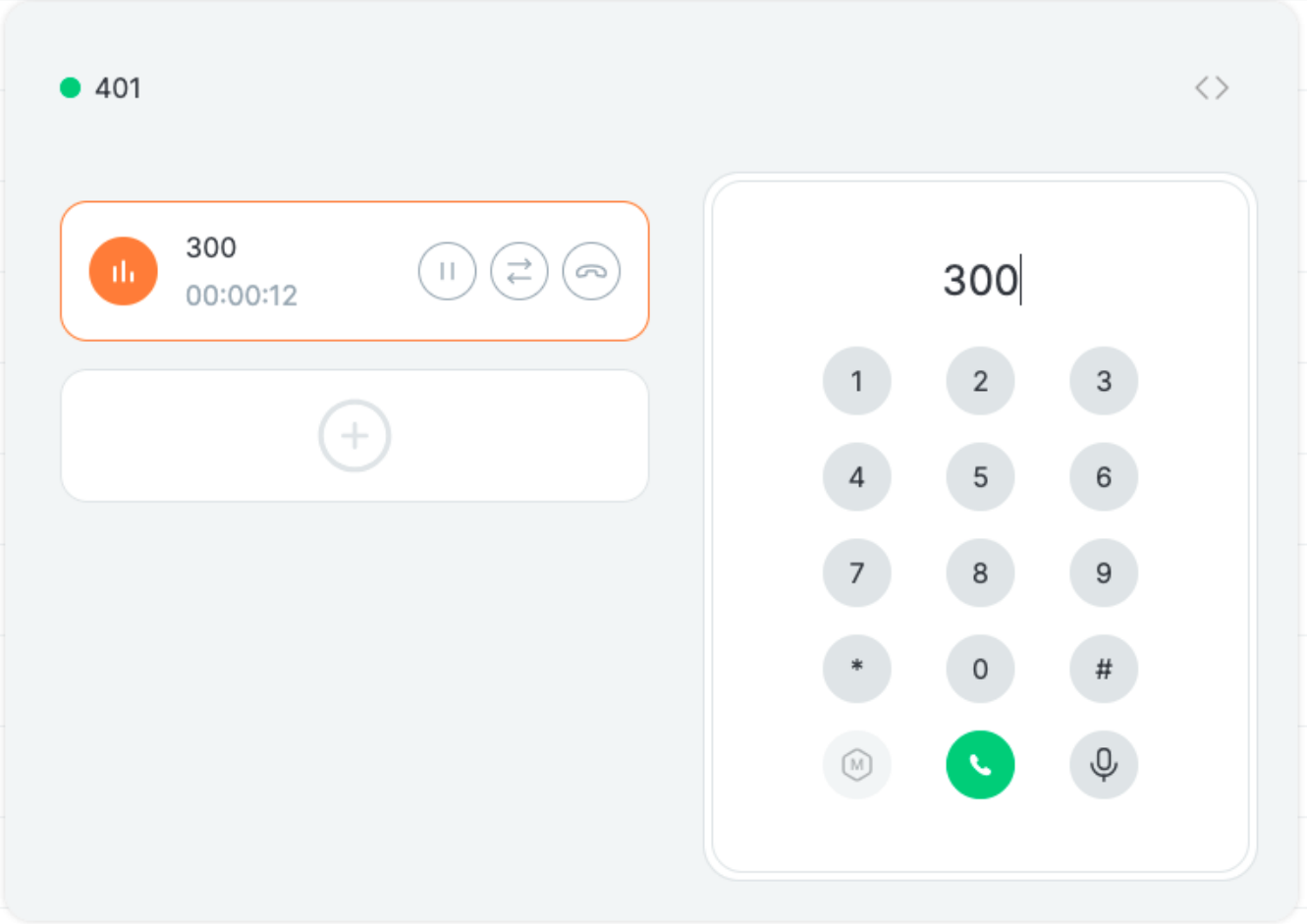
Each time the agent receives a call from the queue, the softphone pops up and will prompt the user to answer. Clicking on the "Start Call" the agent could answer to the call. During the call touch tones could be sent by clicking on the dial pad provided in the panel. The softphone shows the callerId SIP username (usually this information matches the incoming caller ID).

For more information on how to configure the softphone, please refer to Configuring the new Real-time Agents Page and Configuring agents



The softphone uses the local PC microphone. Chrome security policies allow permanent use of the local microphone only for sites secured with HTTPS. Please make sure your QueueMetrics instance is running over HTTPS and relax all security policies for the QueueMetrics URLs. This could be done by clicking on the shield located in the Chrome URL bar and selecting "allow not secure scripts for this site". Be sure to allow microphone usage for the QueueMetrics URL at the very first call.

Once a call is active, you have the option to **pause**, **transfer**, or **end** the call.



Pressing the **Transfer** button opens a panel where you can select the extension to transfer the call to, and choose between a **blind transfer** or an **attended transfer**.

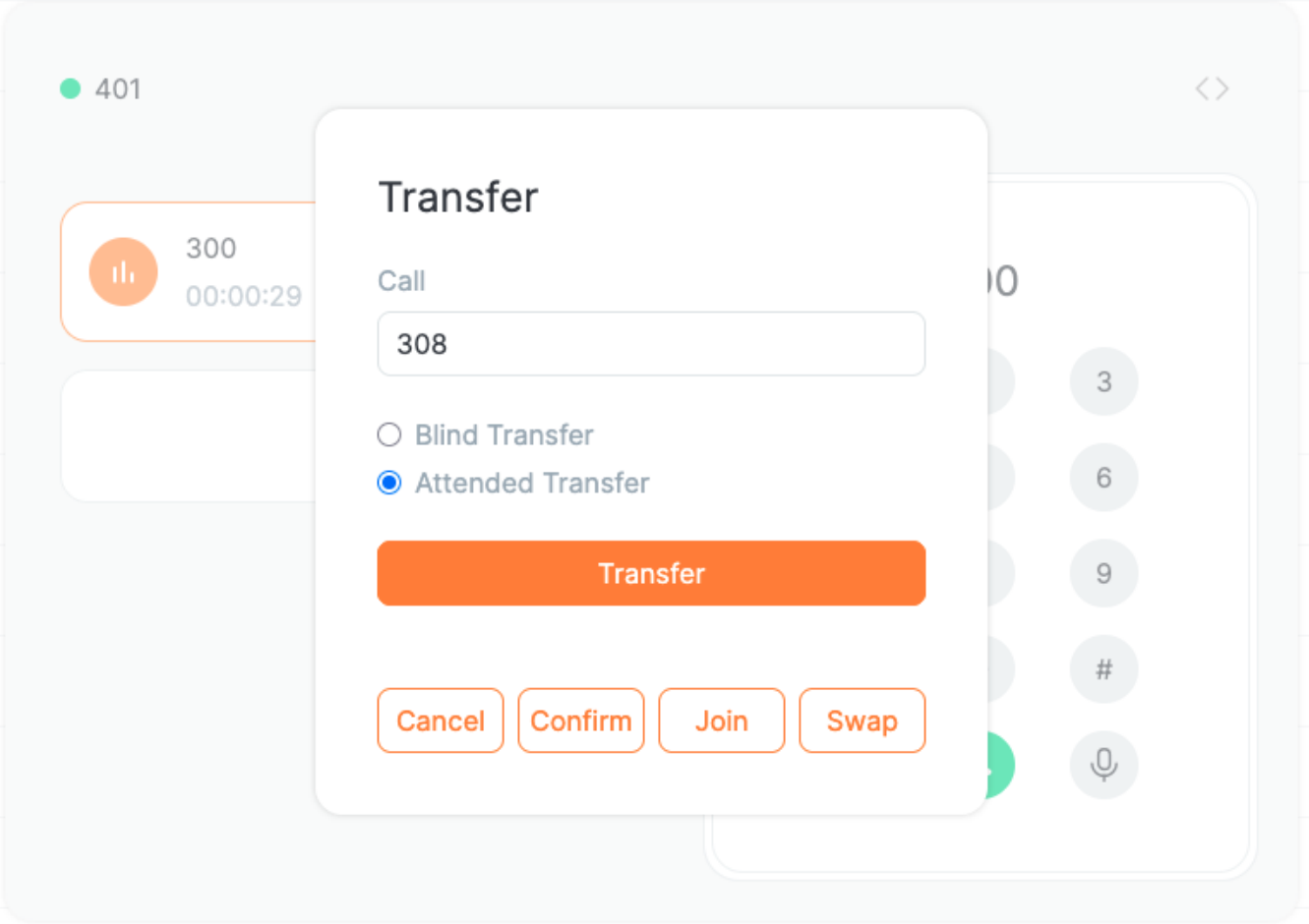
If you select an **attended transfer**, the panel will remain open after the destination answers. From this panel, you can choose one of the following actions using the buttons at the bottom:

- **Cancel** – Cancel the transfer and return to the original call.
- **Confirm** – Complete the transfer and end the original call.
- **Join** – Merge all participants into a three-way call.
- **Swap** – Toggle between the original caller and the transfer recipient by pressing the button repeatedly.

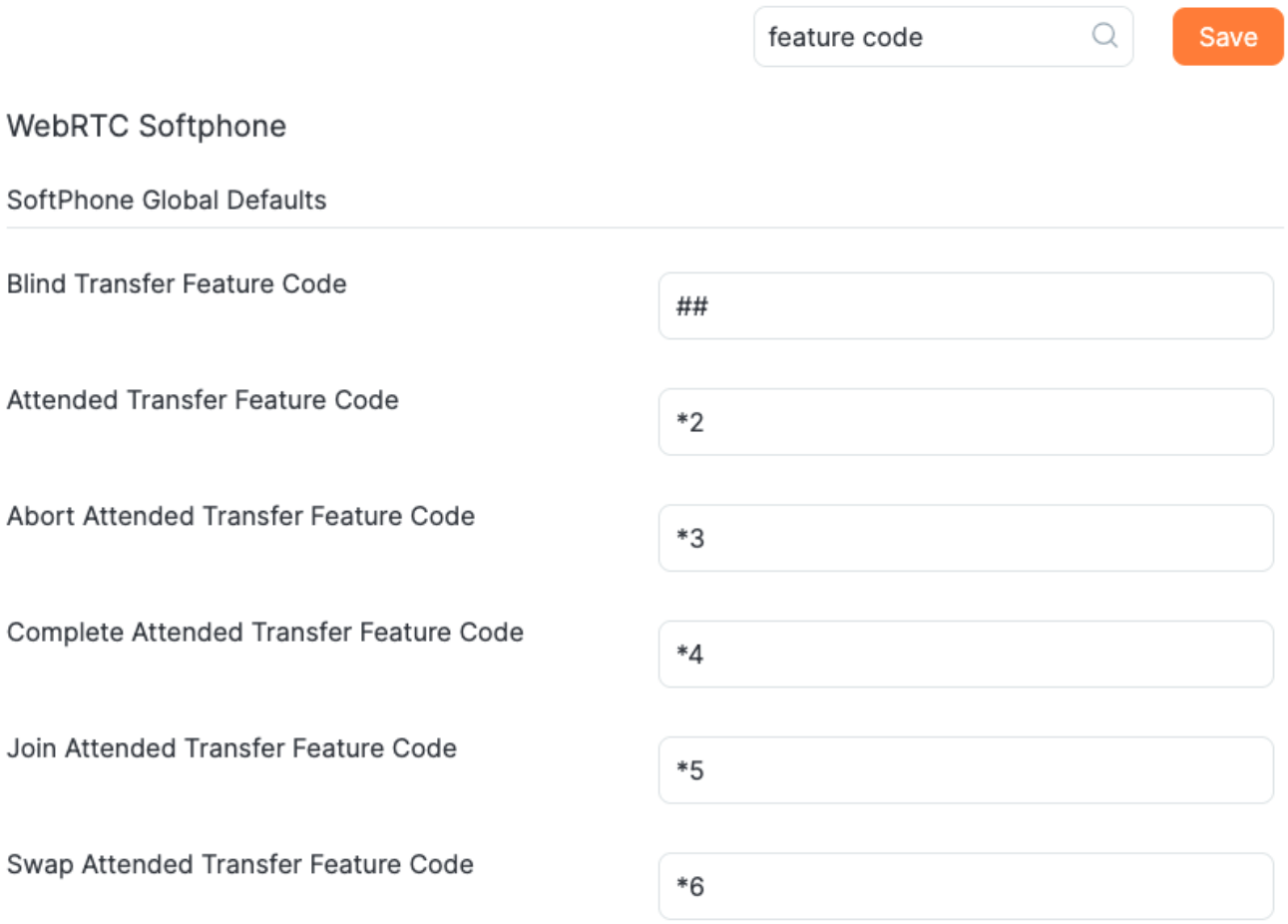


Starting with **FreePBX 16 and later**, you must manually enable the required DTMF Feature Codes for transfer-related actions (e.g., Cancel, Confirm, Join, Swap)

> to function properly. For more information see **QueueMetrics Advanced Manual** under Chapter **Setting up Queuemetrics WebRTC Softphone**, section **Enable DTMF Feature code**



By default, QueueMetrics uses Asterisk DTMF codes to perform transfer actions. You can customize these codes to match your own setup by modifying them in the **explore system parameters**, as shown in the screenshot below.



Softphone settings

The integrated softphone uses a combination of **agent settings** and **system parameters** to fetch it's own settings.

The settings that are available in the **Edit system parameters** page and in the **Explore system parameters page** are the following:

- default.connection_recovery_max_interval: Maximum time interval before attempting a reconnection in case of connection failure
- default.connection_recovery_min_interval: Minimum time interval before attempting a reconnection in case of connection failure
- default.no_answer_timeout: How long should a call ring with no answer before being hanged up.
- default.iceservers: List of STUN / TURN servers to use (e.g. stun:stun01.sipphone.com,stun:stun01.sipphone.com,stun:stun.fwdnet.net)
- default.websocketurl: URL for the web socket that the softphone uses to send data (e.g. wss://10.10.5.181:8089/ws)



These settings will be shared across all agents using the softphone and are not individually configurable.

The softphone settings that are individual for each agent are the following, and can be found in the Agent configuration page, under "Agents" in the Administrator homepage.

- WebPhone Username: the SIP username for the extension
- WebPhone Password: the SIP password
- WebPhone Realm: the SIP realm (eg: 10.10.5.181)
- WebPhone SIP Uri the SIP URI (eg: sip:200@10.10.5.181)
- Auto answer details if the softphone should operate in auto answer mode (Default is no)

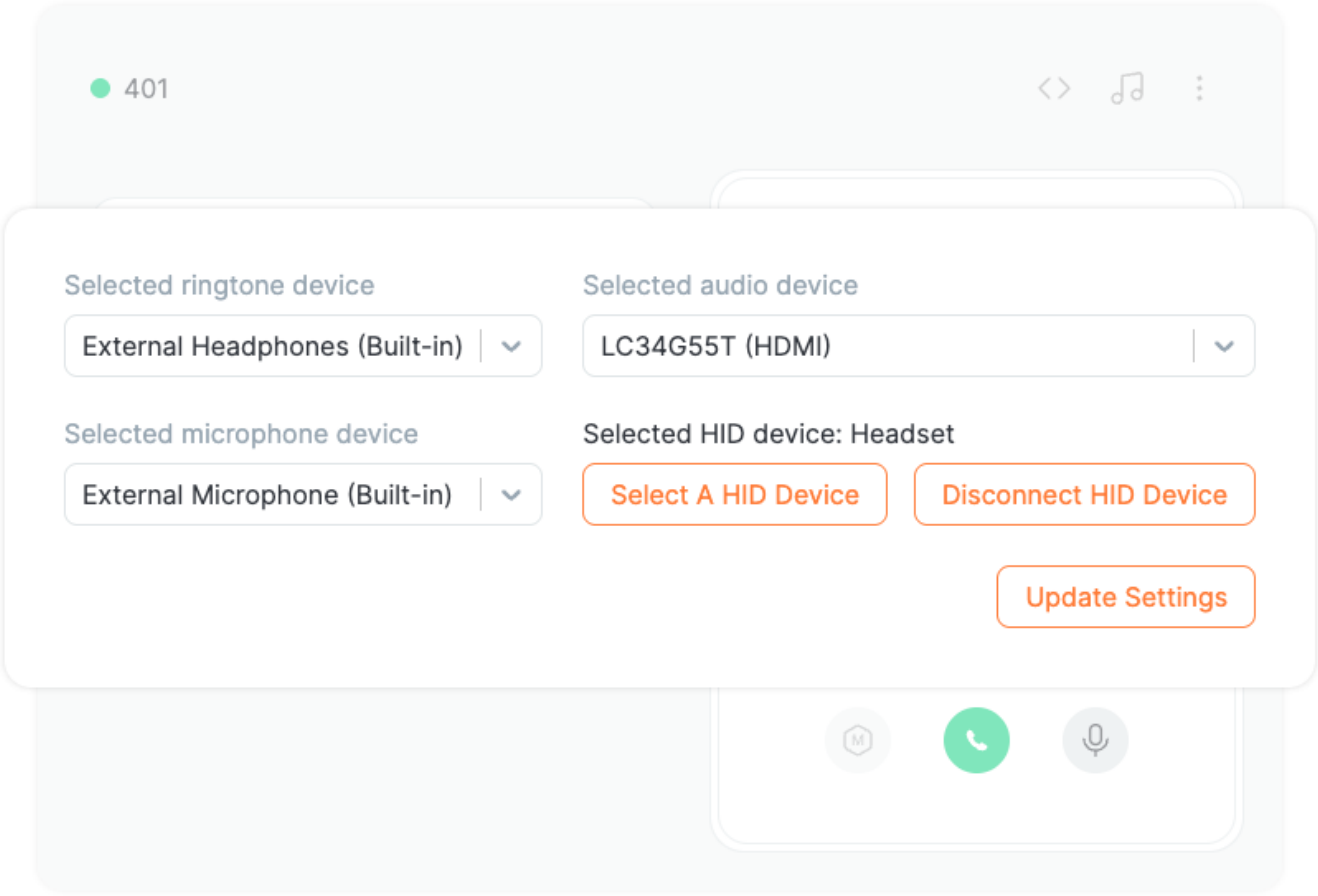
The AutoAnswer Feature, if enabled, allows the softphone to answer calls automatically. This applies only if there are no other active calls at the moment. If the agent is already on a call, and receives another call, this call will ring normally, regardless of auto answer being enabled.

Users with the security key AUTOANSWER_PHONE can enable/disable the auto answer feature by clicking on the auto answer icon (Speech bubble icon). If the icon is green, auto answering is enabled, if it’s black auto answering is currently disabled.

Users that do not hold the security key cannot change the auto answer status, that will be decided by the administrator in the agent settings page.

Advanced Audio Device Selection

The softphone supports advanced selection and management of audio and device routing through a dedicated audio configuration panel.



Audio Device Routing Options

Once the softphone is in a **registered** state, an music icon becomes available in the interface. Clicking it opens a configuration panel that provides the following device selection options:

- **Ringtone Output** – Dropdown to select the device for ringtone playback.
- **Call Audio Output** – Dropdown to select the device used to listen to call audio.
- **Microphone Input** – Dropdown to select the microphone used during calls.

These selections allow users to route audio precisely across available devices (e.g., send ringtone to speakers but calls to a headset).

Clicking the **Update Settings** button applies the selected devices and stores their configuration in the browser's local storage. These settings are saved per-browser and per-computer, and cannot be transferred between browsers or machines. If the same browser is reopened later, the last-used configuration will be reapplied automatically, if the devices are still available.



If no microphone (input device) is detected on the system, the audio configuration panel will not display any selectable devices.

> A working input device is required for the softphone to function properly.
>
> Audio, microphone, and HID devices can be changed **even during an active call**. The new configuration is applied immediately without interrupting the call session.
>
> In some cases, an error may occur when changing the output device (e.g., due to device compatibility or disconnection). If this happens, a notification will appear.
>
> To resolve the issue, you can: Replug the previously used audio device, try selecting a different available device from the dropdown in the audio menu panel and once the call ends, refresh the page to reset the audio routing environment.

HID Device Management

The "USR_HID" key is required for each user to enable this feature.

A dedicated section is available for managing **HID-compatible devices** (e.g., USB headsets with physical call control buttons). It is composed of a label showing the name of the currently connected HID device, if any, a **Select HID Device** button, which opens a native browser prompt to choose an available HID device.(Upon successful selection, a confirmation notification is shown, otherwise an error message is displayed)

If a device is connected, an additional button **Disconnect HID Device** appears, allowing users to manually disconnect the device and removes it from localstorage.

HID device selection is only supported in browsers based on the Chromium engine(eg.Google Chrome, Microsoft Edge,Opera,Chromium)

HID Device Button Behavior

If a connected HID device has physical buttons (e.g., a call-answer button), those buttons will answers the call if a call is incoming otherwise will hangup the active call session. After an action to hang up or answer is executed, this behavior is locked for 5 seconds. This prevents unwanted automatic answering and hanging up, which can be caused by some devices sending multiple events from a single click.

Device Disconnection and Fallback

If any selected audio device is physically disconnected from the computer during a session the softphone will revert to the browser’s default input/output devices.

Device selection can then be modified again through the system’s audio settings or the softphone panel.

Persistence and Memory

Audio and HID device configurations are stored using browser **localStorage** (The configuration is

unique to each browser and computer combination) When reloading or reopening the softphone interface, the system attempts to reconnect the previously saved devices automatically if they are available.

Known Limitations

On Linux, **Chromium-based browsers** (eg. Google Chrome,Microsoft Edge,Opera,Chromium (open-source base)) currently do not support dynamic detection of newly added audio/HID devices. To view and use newly connected devices, users must **reload the page**. The WebHID API receives every event sent from the headset. However, when other applications like Teams, Skype, or the headset's proprietary software are running simultaneously with the QM softphone, they can interfere by also trying to communicate with the device. This conflict may result in unwanted behavior, such as causing the QM softphone to answer or hang up calls unexpectedly.

ICE Gathering Optimization

To reduce call setup time, the softphone uses a partial ICE gathering strategy. Instead of waiting for every possible network path, it prioritizes usable (routable) candidates and moves forward as soon as it has enough information to establish a connection.

ICE Gathering Mechanism

The gathering process is driven by a simple timing strategy that balances speed with reliability:

- Initial discovery: The browser starts collecting ICE candidates as usual.
- Hard timeout: A maximum limit (controlled by default.maxGatheringTime) ensures the process doesn't take too much time. Even in slow or restricted networks, the call will eventually proceed.
- Soft Timeout (Routable Discovery): If a STUN/TURN candidate is found, the system identifies this as an ideal candidate and instead of waiting for the full hard timeout, a shorter "Post-Gathering" delay is triggered.
- Finalization: Once the soft timeout expires, or if the browser naturally signals that gathering is complete (sending a null candidate), the call proceeds immediately using the best available candidate.

Configuration

The behavior of the gathering process is controlled by two specific configuration properties:

- **default.maxGatheringTime**: (Expressed in ms) Sets the maximum total time allowed for the ICE gathering process.
  - If set to 0, or not set, the optimization is disabled and the softphone reverts to the browser's default natural gathering.
- **default.icePostGatheringTimeout**: (Expressed in ms) The duration of the soft timeout triggered after a routable STUN/TURN candidate is detected.
  - This allows a brief window for even better candidates to be discovered before finalizing the connection.
  - Defaults to 500 ms.

Behavioral Logic Flow

When a call is initiated, the following sequence is executed:

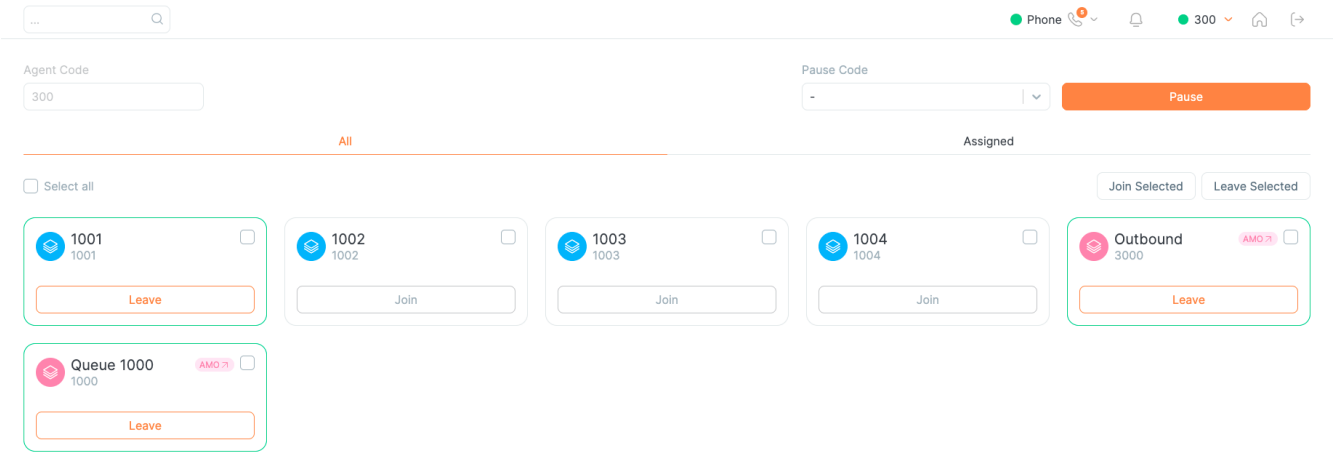
- The system listens for ICE candidates
- If maxGatheringTime > 0, the optimized logic is enabled.
- When a STUN or TURN candidate is detected, the post-gathering timer starts.
- If Soft Timeout timer expires before the Hard timeout, gathering stops early and the call

proceeds.

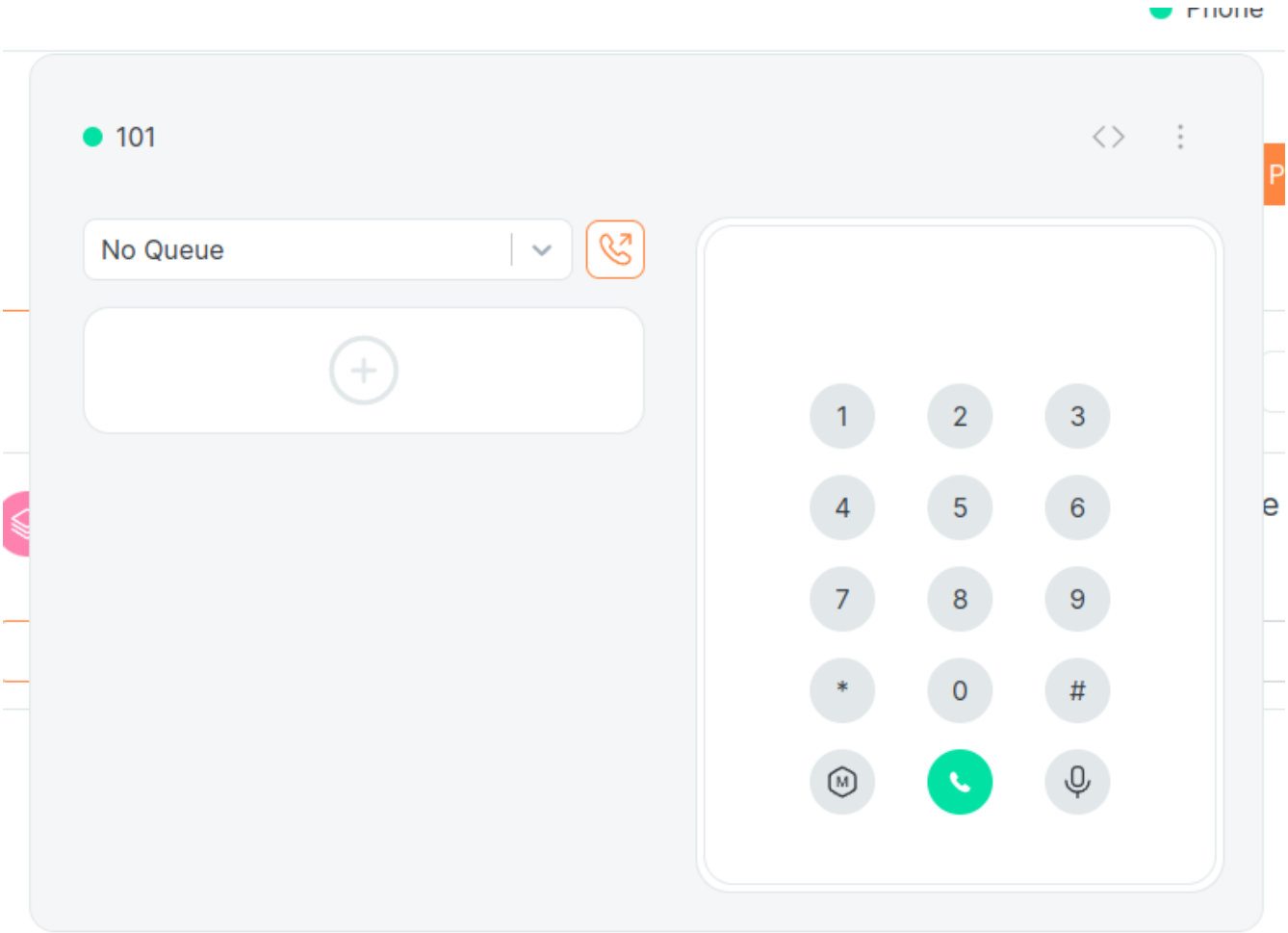
- If no routable candidates appear before the hard timeout, the system logs a warning and continues with whatever candidates are available, avoiding indefinite delays.

AMO Campaigns

If your agent has visibility on one or more AMO campaigns, they will see them in the Control view.



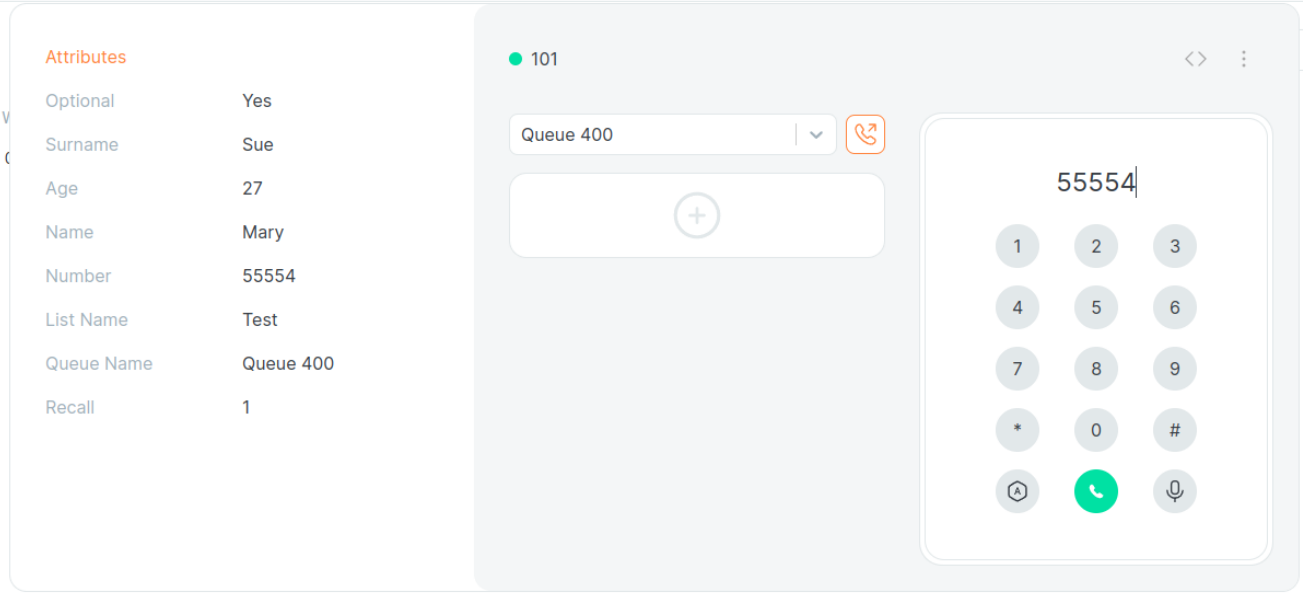
Once an agent has logged in on an AMO queue, they will be able to make AMO calls directly from the softphone or the dialout panel:



By clicking on the AMO button next to the queue selector, the user will trigger a request to the QueueMetrics server for an AMO call to be made.

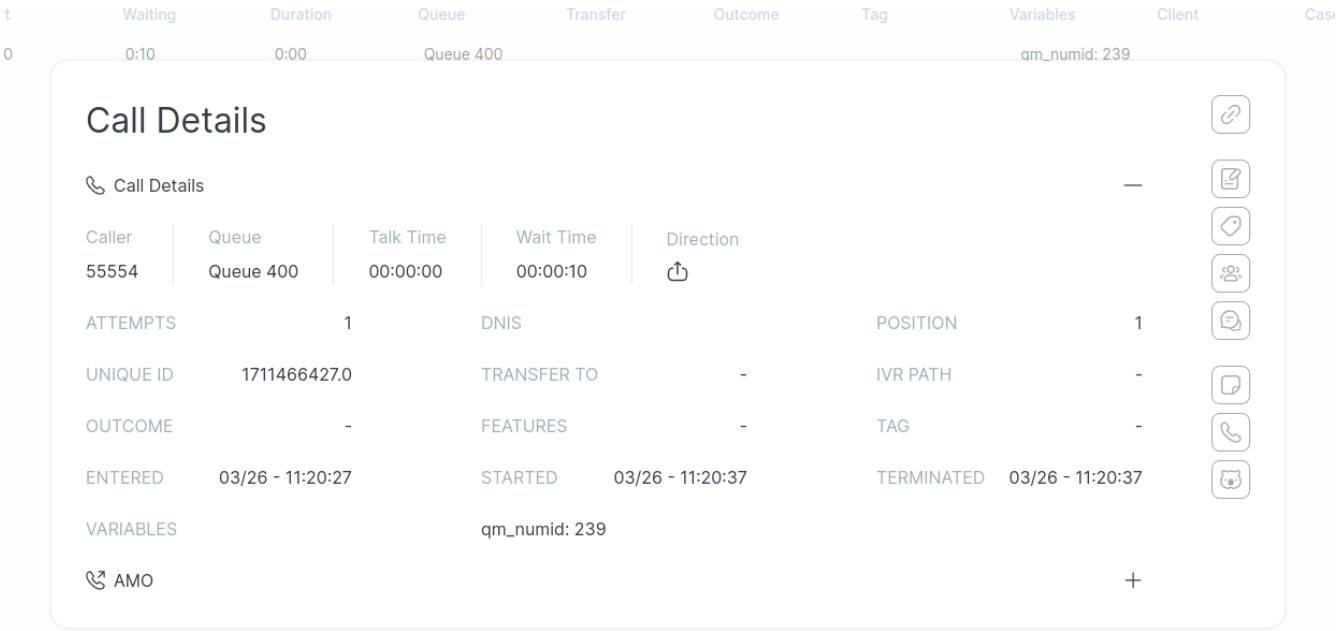
Clicking on the AMO Call button will ask QueueMetrics for an AMO call in the list of numbers that are still available in the AMO pool. It's not certain that a call will be available to be made, depending on the state of the AMO campaigns and their lists. If an available call is present, it will be dialled automatically, if not, a message will inform the user of the inability to place an AMO call.

When an agent is logged into AMO queues with pending calls, a badge displays the total call count. The call will initiate as soon as the user clicks the button.

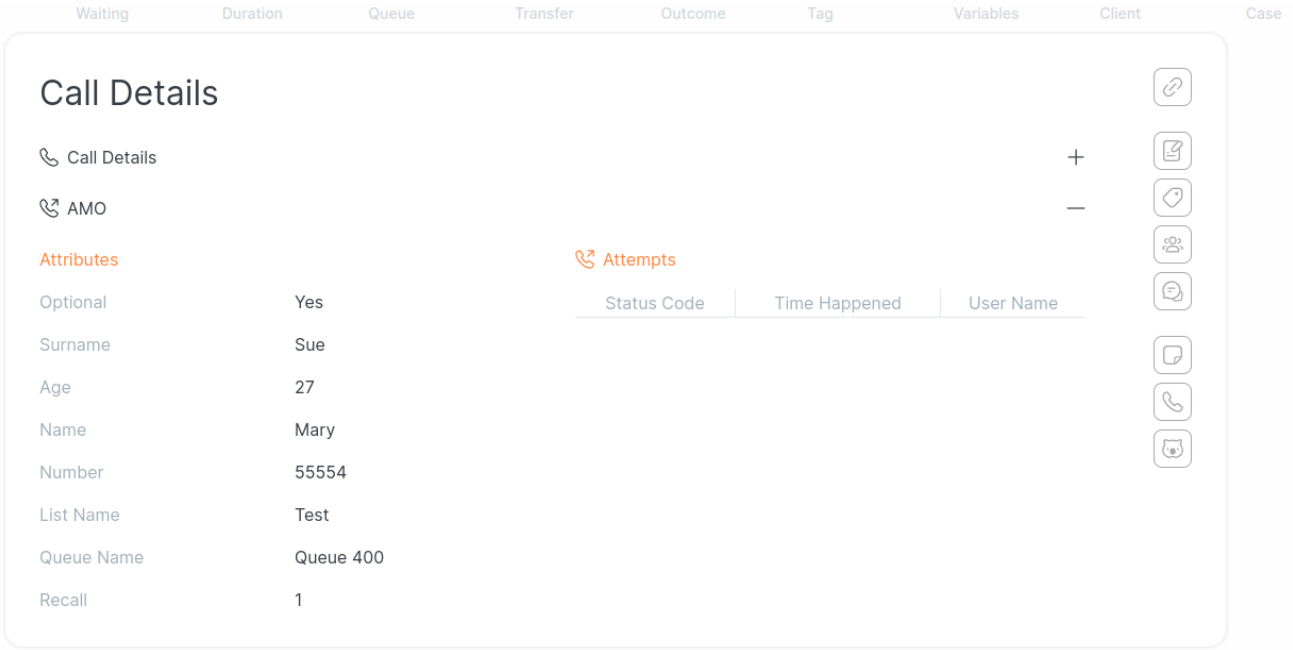


Once an AMO call is initiated, a side panel will open to the left of the Softphone, showing the attributes and relevant information about the number that is being dialled.

Once the call connects, the Call Details panel will be opened automatically, to offer the agent the possibility to work on the call, taking advantage of the functions of the Call Details panel (Outcomes, Clients, Cases, etc.).



When inspecting the details of an AMO call, an additional section of the Call Details modal is added, as evidenced by the AMO icon at the bottom. Clicking on it will expand the Call Details section, showing the attributes of the call, and the list of attempts made for that number.



Disable outbound autonomous calls

If you want to prevent an agent from placing outbound calls autonomously, you can assign the following security key to their user profile:

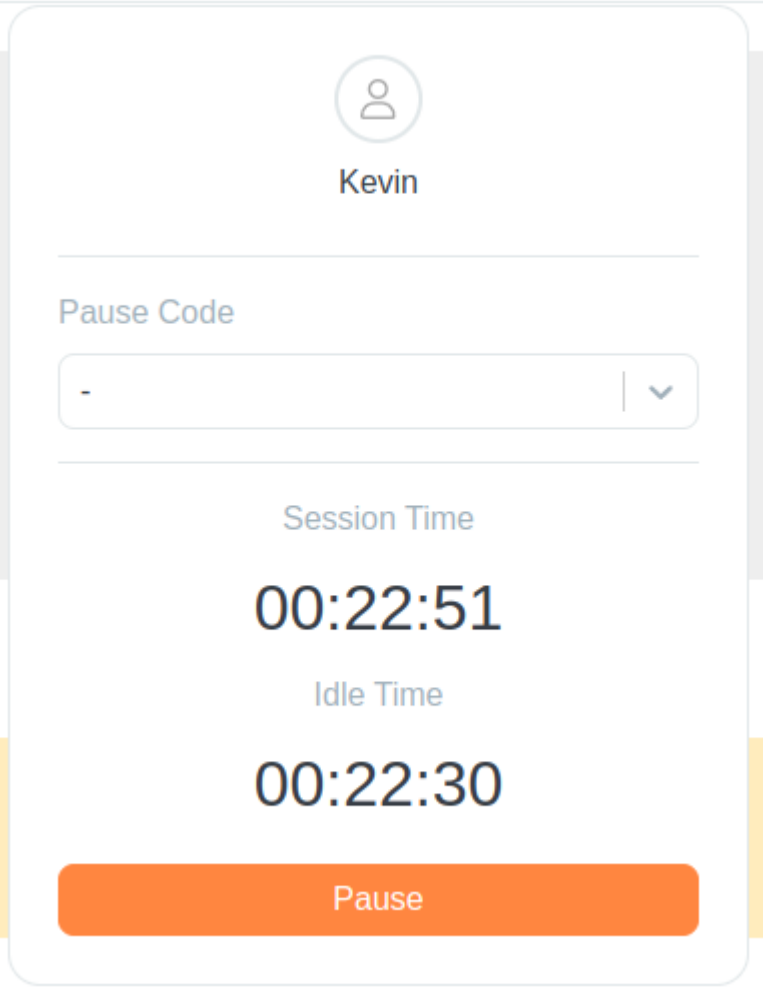
NODIALOUT

With this key enabled, the agent will not be able to open the dial-out panel and in the softphone they cannot select a queue, or choose an AMO campaign. However, they will still be able to receive and place inbound calls. This setup allows you to manage outbound calls programmatically via the

API or a custom dialplan. This approach allows you to control this behavior on a per-user basis. If you want to restrict it for all users, you should assign the security key at the class level.

Agent Status Panel

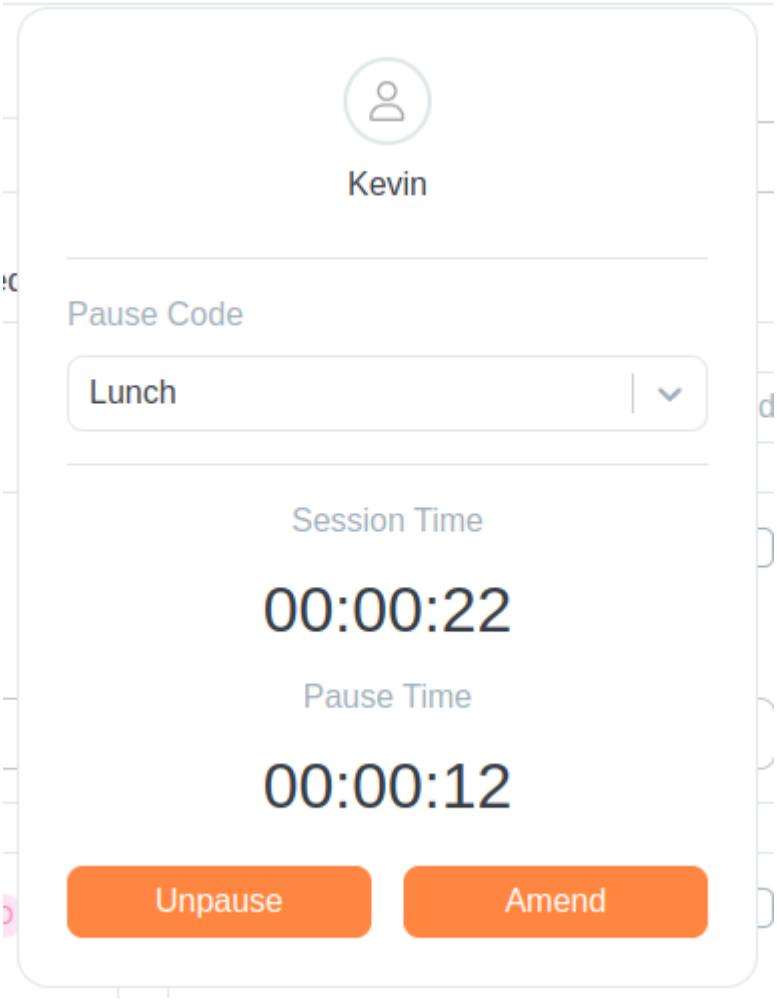
The Agent Status Panel provides a convenient overview of the agent's current status and offers several functions that can be accessed by clicking on the agent's name in the top right corner.



From this panel, users can:

- View the agent's current status, session time and idle time.
- Pause or unpause themselves.
- Change or amend the pause code.

Please note that the appearance of the "Change/Amend" button may vary depending on the value of the system parameter: `default.agentCanAmendPause`. For more details, refer to Appendix C: System Preferences.



Additionally, if the agent is on a break, the panel displays the amount of time the agent has spent in the current pause.

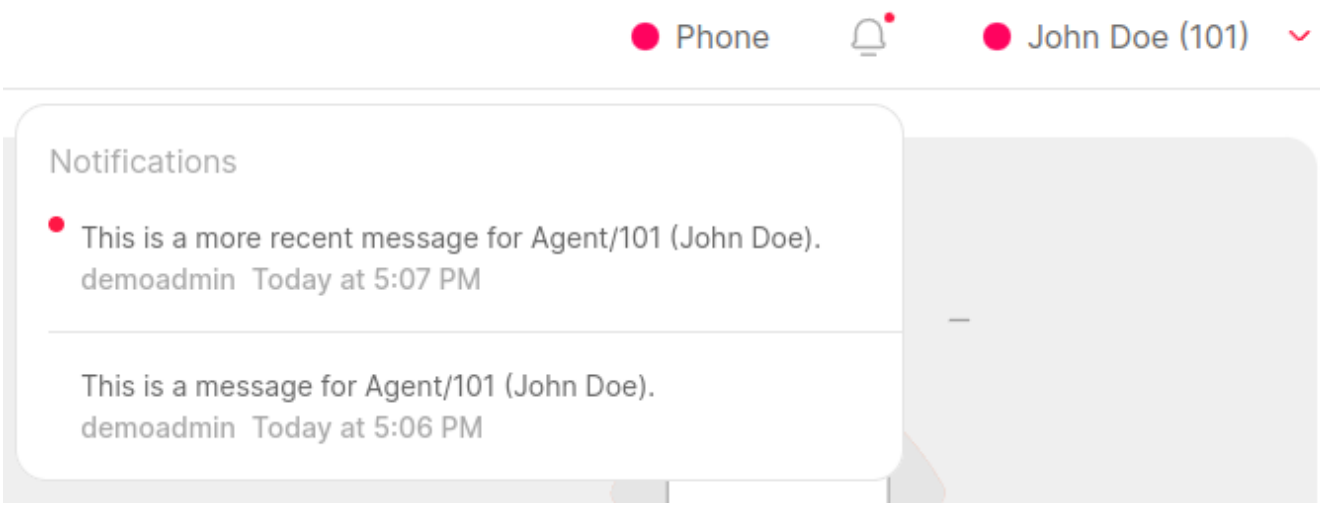
Notifications

The Notifications panel can be accessed from the Bell icon in the top bar on the right-hand side. It serves as a platform to display broadcasted messages to agents.



Key features of the Notifications panel include:

- **Red Dot Indicator**: A red dot appears on the Bell icon when new messages are received, providing a visual cue to agents.
- **Message Text**: Agents can view the content of received messages directly within the panel.
- **Timestamp**: Each message is timestamped to indicate when it arrived.



QueueMetrics Memo System

QueueMetrics 24.05 introduces a new Memo System to aid Agents and Supervisors during their workflow.

The QueueMetrics Agent Page (QAP) offers a specific view for users to browse, create, update and edit their Memos, and three different tabs to further organize their Memos.

Memos can be of several types, inlcuding:

- Notes
- ToDo
- Meeting
- QA
- Recall
- Flag
- Training

Specifically, some of these are generated in response to specific events, like QA Memos that are sent to the Agents when a Supervisor completes a QA form on one of their calls, while others are just different types of notes that Agents can send to others or themselves, like Notes and ToDos.

Recall Memos can be created from the interaction details of the QAP, and they allow for some special commands in the agent page, namely opening the softphone and initiating a recall for that number automatically, as long as the agent clicks on the recall button. (In order for it to work, the agent must **not** have the **NODIALOUT** security keys)

Memos can have different statuses, that keep track of their completion and acceptance in case of tasks that are assigned to agents, namely:

- TBD (The task has been created but not completed)
- OK (The task was successfully completed)
- Rejected (The task was rejected by the receiver)

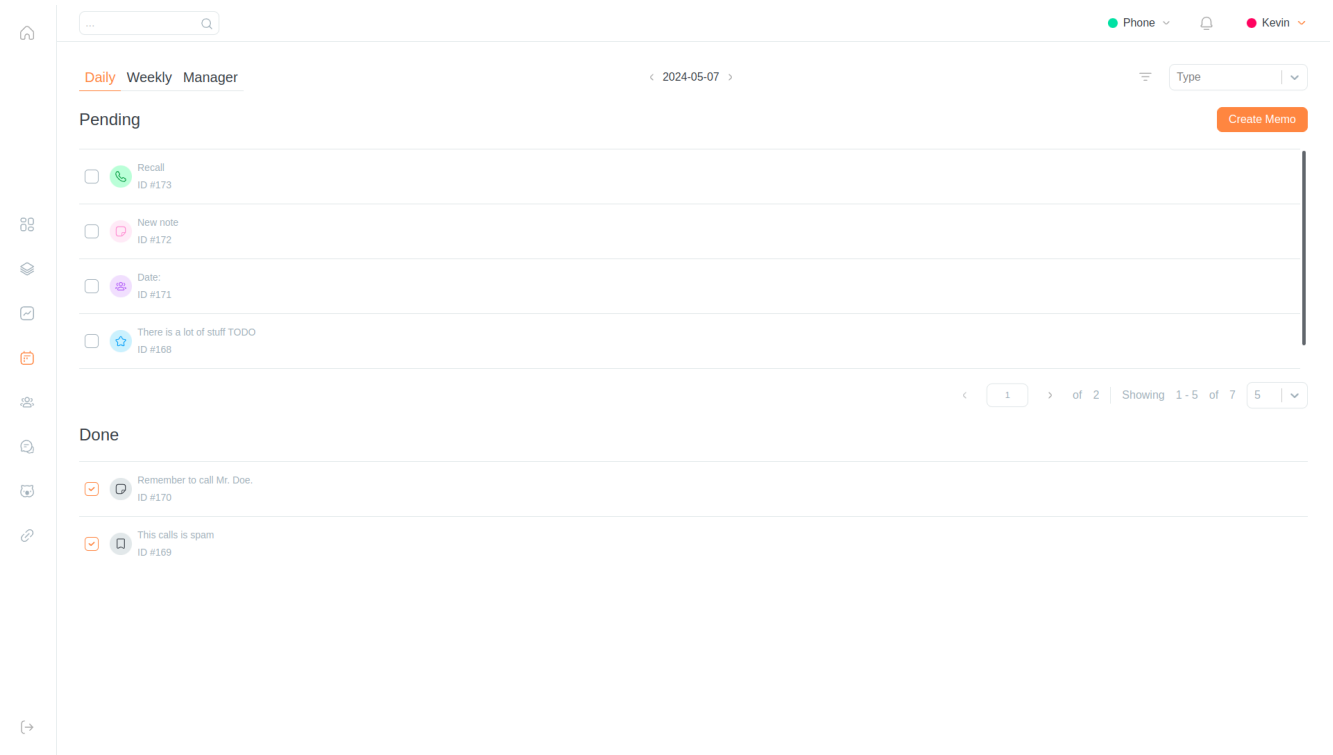
- Refused (The task was refused by the receiver)
- Rejected (The task was rejected by the receiver)

If these definitions seem a little vague, that's by design, since most organizations will find their own internal mapping to define what the three different types of rejection mean in their context.

Daily View

In this view, Agents can see their Memos organized by day.

Using the selector on the top right, the user can change the day they are looking at, filtering memos by their expiration date. If a Memo is expiring on a specific date, the user will see it listed in the daily view of that day.



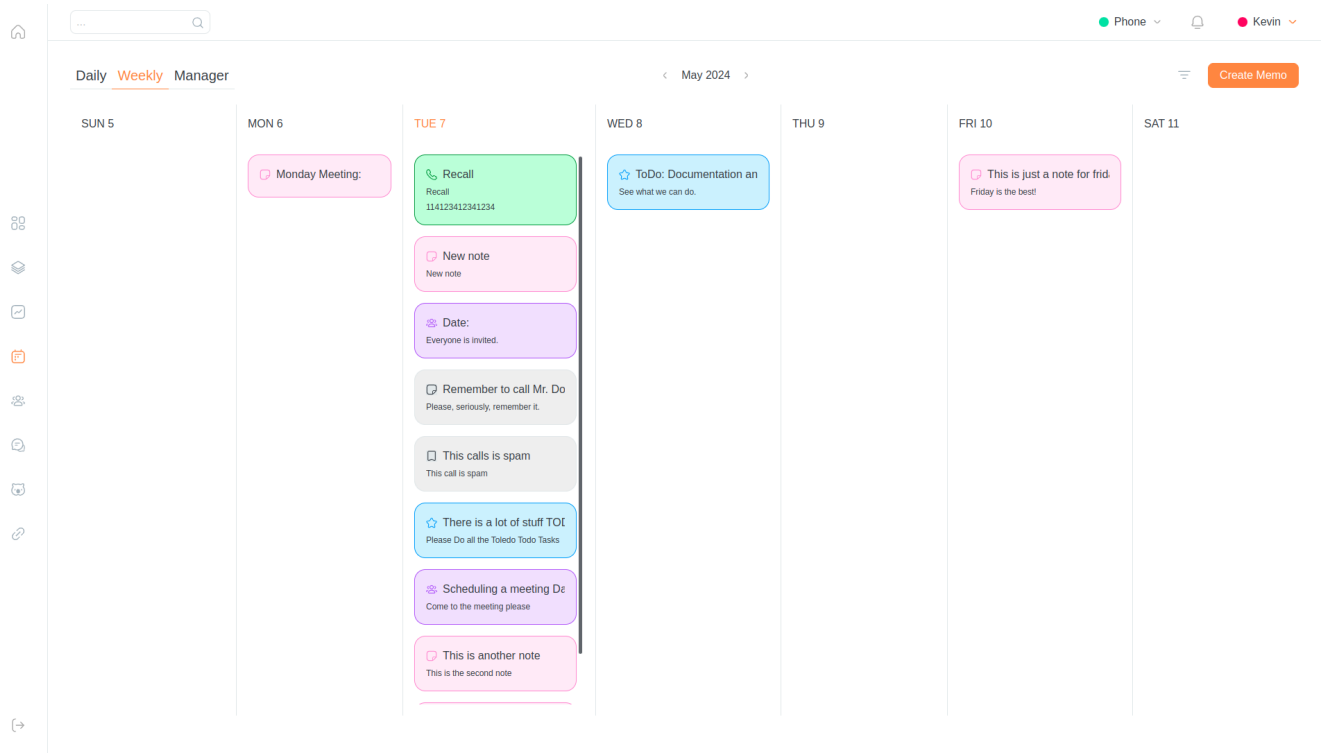
The View is divided in two. A list of pending Memos, whose state is listed as "TBD", and a list of completed ones with status "OK". There is a "Create Memo" button in the top right that will bring the User to the Memo Creation Editor.

Both tables are paginated, and clicking on the checkbox on the left of a pending memo will complete it, while clicking on the same spot for a completed memo will put it back in the Pending list.

Clicking on a Memo will take the user to the Edit Memo page.

Weekly View

In this view, Agents can see their Memos organized by week, in a simil-calendar mode. While the individual days are not strictly divided by hour, they are divided by day, creating different individual lists, one for each day of the week. By using the top selector, the user can change week, moving backwards and forward in time.



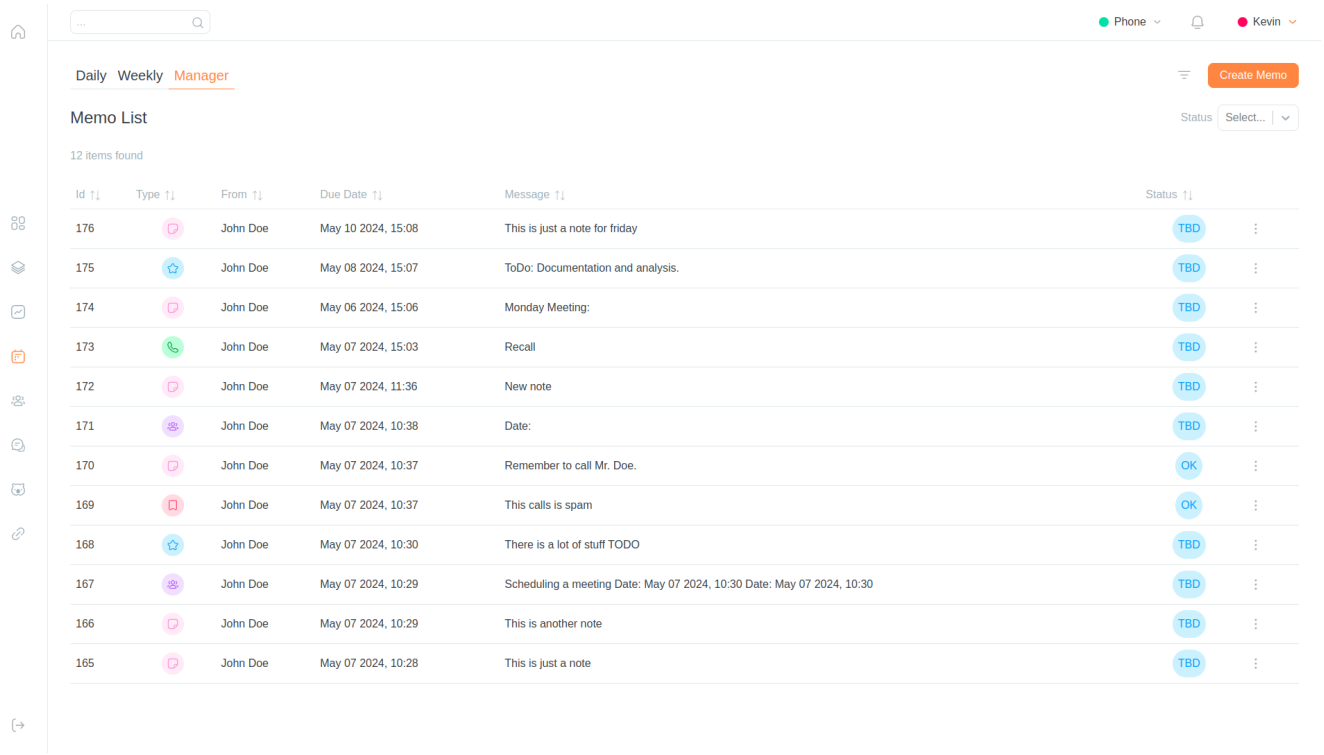
Memos are color coded based on the type, and each type of memo has its own icon. Hovering with the mouse over an icon will tell the user the Memo's type.

The user will find the Create Memo button here as well, with a filter icon on its left. Clicking on this icon will show two drop-down menus that let them filter Memos by Status and Type.

Clicking on a Memo will take the user to the Edit Memo page.

Manager View

In this view, Agents can see their Memos in full, filtered by many different types of filters. The page appears as a table that lists all Memos that satisfy the User-defined filters.



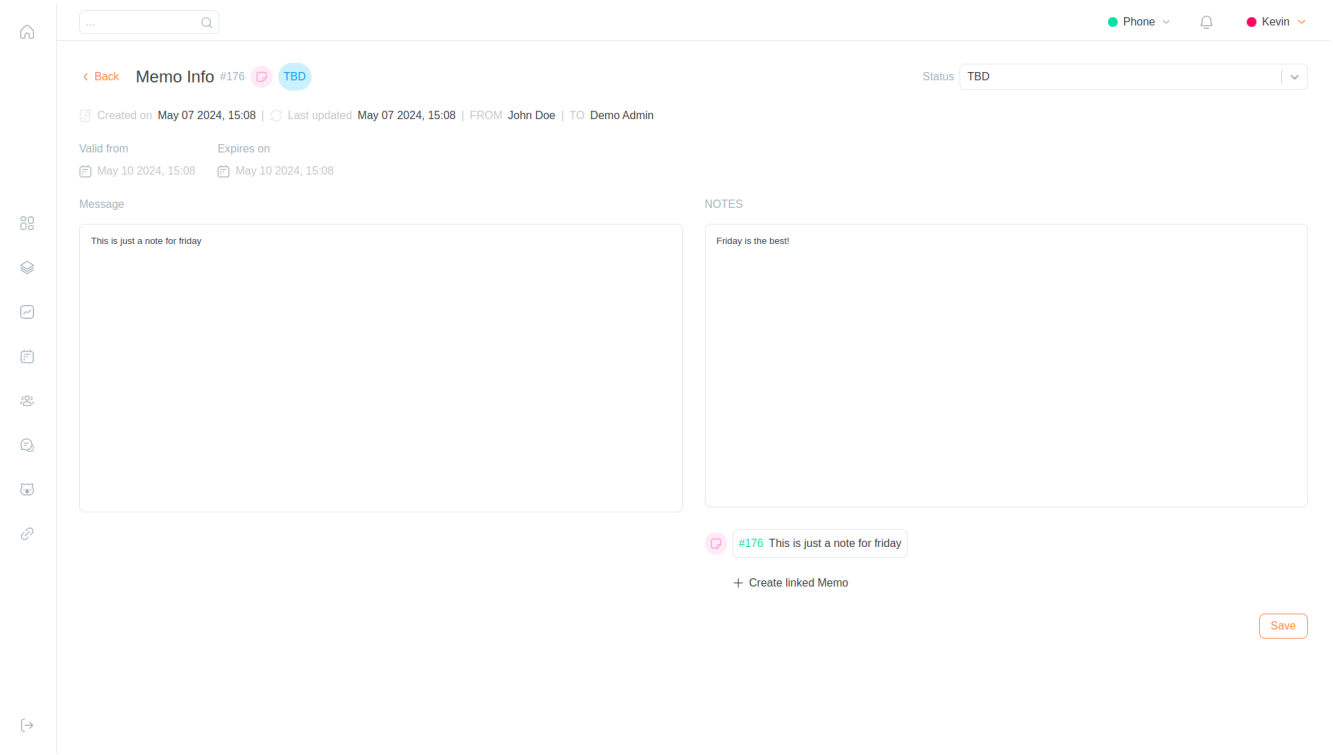
The user will find the Create Memo button here as well, with a filter icon on its left. Clicking on this icon will show two drop-down menus that let them filter Memos by Category and Type, while the Status filter is always visible on the page. Clicking on the filter button will also show two Date selectors that allow the user to indicate a time period for the Memos, showing only memos that are expiring during that period of time.

Clicking on the icon at the right end of every Memo, the user can edit, complete, reject, dispute or refuse the Memo.

Memo Editor

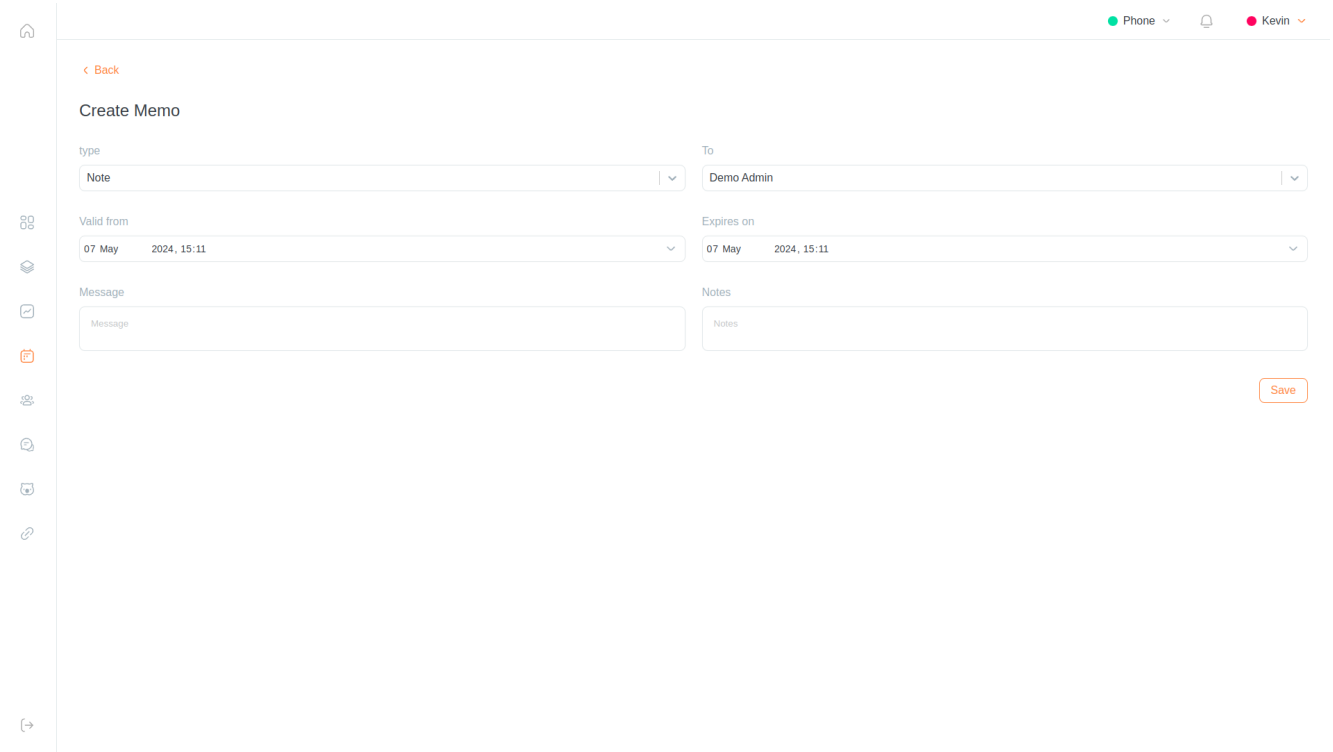
Memos can be edited in different parts, depending on the ownership (Assigned to the Memo creator). If a user is the Owner of a Memo, they can modify the "Message" part of the Memo, as well as their status. If the user is the receiver of the Memo, they are able to modify the Notes section.

Both are allowed to edit the status of the Memo, and to create a linked Memo by using the button at the botton of the page.



Memo Creator

The Memo Creator can be accessing by clicking on any of the "Create Memo" buttons. From here, Agents or Supervisors can create Memos and send them to their recipients.

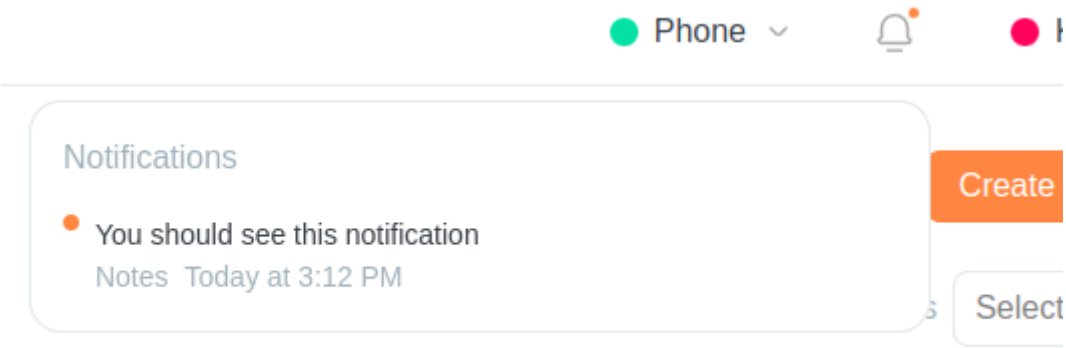


Here, the user can create Memos adding notes and a message, while also picking the recipient and type of memo they want to create, as well as the validity and expiration dates.

Notifications

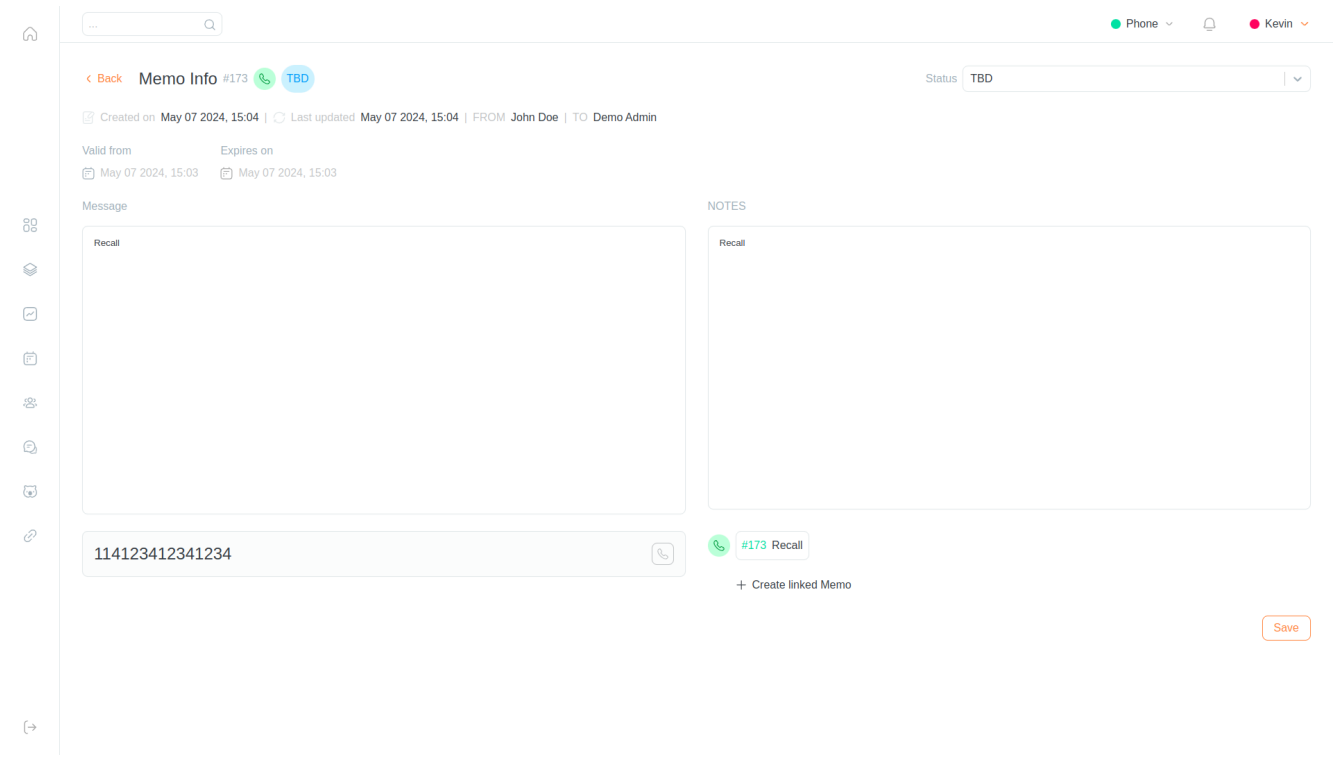
When a Memo is created by a user, or sent to them, it will appear in the top right Notification menu.

Clicking on the notification for a Memo will take the user to the Edit Memo page for that Memo.



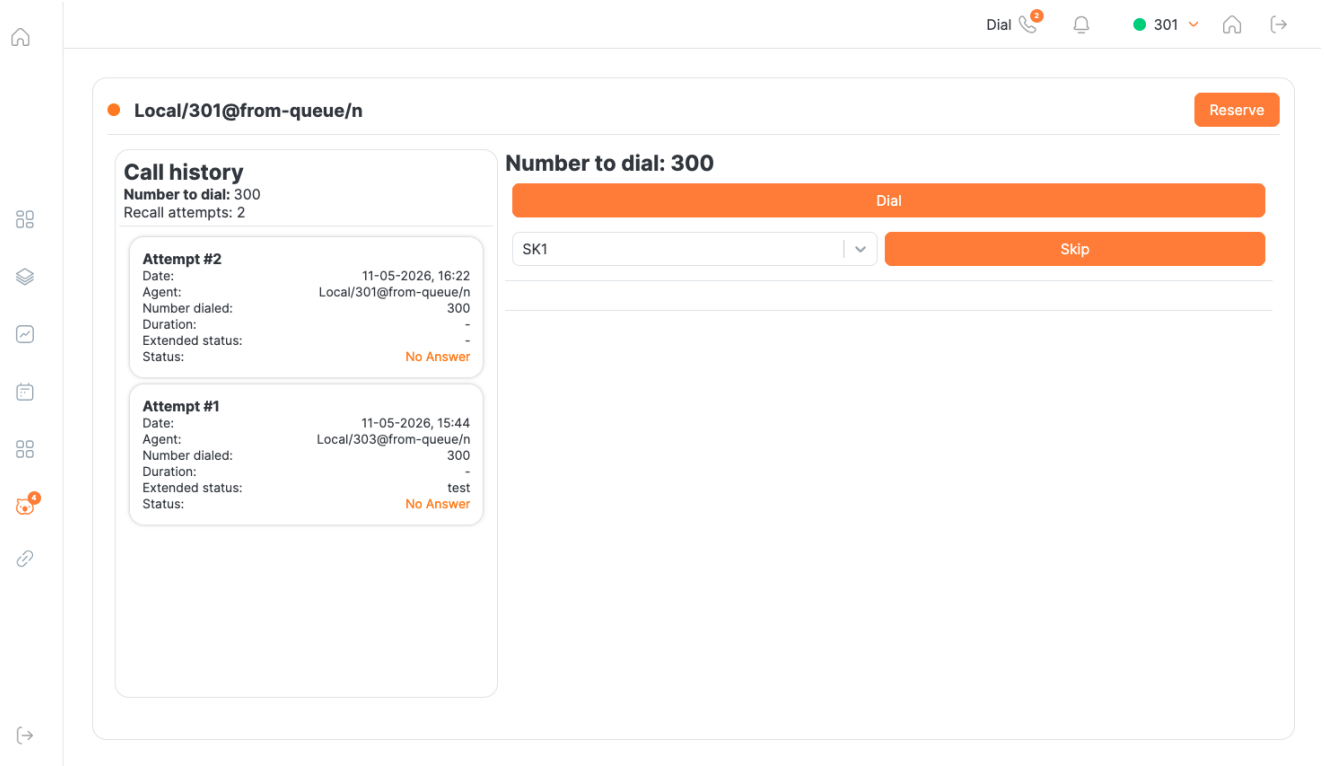
Recall Memos

From the Interaction Detail view, the user can create a Recall Memo for that specific Interaction. Those Memos will offer a specific recall button in their Edit Memo page, that allows user to initiate a call directly, if their softphone is enabled.



WombatDialer Preview Dialing

The native integration between QueueMetrics and WombatDialer enables agents to manage preview dialing directly from the Agent Page. This seamless approach removes the need for external iframes, resulting in a faster and more user-friendly interface.



Configuration

To enable the Preview Dialing panel, you must configure the following properties in the System Parameters:

- `wombat.previewDialingUrl`: Set this to your WombatDialer instance URL. (eg: `wombat.previewDialingUrl=https://{wombat-url}/wombat/`)
- `agent.wombatPreviewChan`: Defines how the agent is identified on the PBX. Use the ${agent} placeholder to dynamically insert the agent's current extension. (eg: `agent.wombatPreviewChan=PJSIP/${agent}`)
- `agent.wombatToken`: Define this if your Wombat API is protected by a token.

The Dialing Workflow

When the integration is active, a Wombat icon appears in the navigation menu. A badge on the icon displays the sum of all reservable calls across the agent's active queues. This number is retrieved from the ephemeral data table using the WBT:RESERVABLE_CALLS key. It is filtered by the queues the agent is logged into and restricted to values that have not yet expired.



In the panel the agent can:

- Reserve a Call: Clicking the Reserve button pulls a record from the dialer. If successful, the agent's status updates, and the interface displays the destination number along with the call history for that specific contact.
- Dial: Once a call is reserved, the agent can click Dial to initiate the outbound call.

- Skip a Call: If an agent needs to skip a reserved record, they must select a reason from a predefined list. This list is configured via: `agent.wombatAttr.extStatusSkip=SK1,SK2`



If the agent code is not found on WombatDialer (e.g., because the agent is not logged into any valid queues), the page will display an appropriate error message explaining why the reservation failed.

Managing Input Attributes and External URLs Integration

The Preview Dialing panel can display data fields (Attributes) associated with the contact being called. These attributes are pulled directly from WombatDialer and may include data imported from a list or values saved by agents during previous calls (Output attributes).

Input Attribute Visibility

By default, QueueMetrics show all contact attributes. You must explicitly list which variables you want the agent to see using the `agent.wombatAttrsIn` property.

- Property: `agent.wombatAttrsIn`
- Usage: Provide a comma-separated list of outcomes.

(eg. `agent.wombatAttrsIn=name,surname,caseid` Even if Wombat has 20 variables, the agent will only see these three: `name`,`surname`,`caseid`)



the variable are case sensitive

Individual External Links per Attribute

You can also transform any displayed attribute into a clickable link. This allows you to create multiple "entry points" to external systems (CRMs, ..) based on the specific data in that field.

To create a link for an attribute, create a new property following this syntax: `agent.wombatUrl.{attributeName}`.

- Placeholder Logic: Use ${attributeName} within the URL. QueueMetrics will dynamically replace this placeholder with the actual value, lowerCase, from the current call record if available in the `agent.wombatAttrsIn` property.
- Multi-Variable Links: You can use multiple placeholders in a single URL to build complex queries.

To create a link for...	Use this Property Configuration
caseid	agent.wombatUrl.caseid=https://crm.local/ticket/id=${caseid}&user=${surname}
surname	agent.wombatUrl.surname=https://crm.local/search?q=${surname}



For example if a call has the following input attributes caseId=5,surname=Rossi follwing the above table, each attribute will have the following url - caseId url →

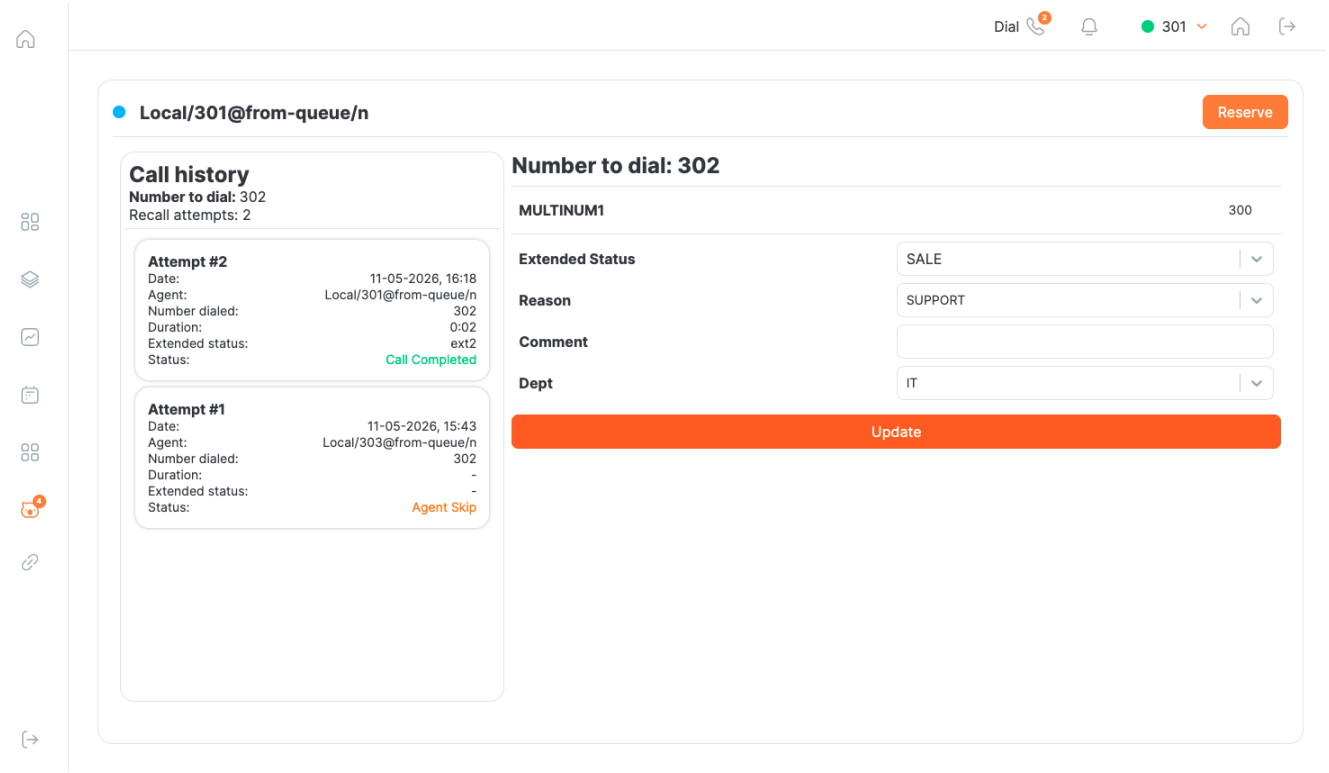
https://crm.local/ticket/id=5&user=rossi - surname url → https://crm.local/search?q=rossi



Attribute links are designed to open automatically in a new browser tab.

Call Handling and Disposition

When a call is connected, agents can submit outcomes and data to the call.



This is handled through two types of fields: Extended Status and Output Attributes.

Setting the Call Outcome (Extended Status)

The "Extended Status" is a dropdown menu used to define the final result of the call (e.g., a Sale or a Don’t call or a Recall).

- Property: `agent.wombatAttr.extStatus`
- Usage: Provide a comma-separated list of outcomes.

Eg. `agent.wombatAttr.extStatus=SALE,DONTCALL,RECALL`

If you want to use the integrated QM call outcomes, simply leave `agent.wombatAttr.extStatus` empty. The dropdown will then populate with QM call outcomes based on the configured security keys.



While this configuration respects the call outcomes' security keys, it does not adhere to queue visibility restrictions.

Defining Output Attributes (Custom Data Fields)

Output variable are additional data points the agent can add to the call (e.g., comments, department names, or reason codes).

- Step 1: Define the Fields. Configure `agent.wombatAttrsOut` property to list the "variables" that agent can add to the call separated by comma. (eg. `agent.wombatAttrsOut=comment,dept,reason`)
- Step 2: Configure the Input Type (Text or Dropdown). By default, any attribute listed above appears as a text field (limited to 50 characters). To turn a text field into a dropdown menu, you must create a new property specifically for that attribute using the syntax: `agent.wombatAttr.{attributeName}`.

Attribute Name	Property to set	Resulting UI Component
comment	(None set)	Text Field (Agent types manually)
dept	agent.wombatAttr.dept=IT,SALES,HR	Dropdown Menu (Agent selects from list)
reason	agent.wombatAttr.reason=PRICE,SUPPORT	Dropdown Menu (Agent selects from list)

In this scenario, the agent will see a free-text box for comments and a fixed dropdown menu for the dept and for the reason.



Agents can add output attributes only if an extended status for the call has been set; otherwise, the update button is disabled.

Reference: WombatDialer Integration Properties

The following table summarizes all configurable properties related to the WombatDialer Preview Dialing integration.

Property	Explanation
`wombat.previewDialingUrl`	Sets the URL of your WombatDialer instance (e.g., `https://{wombat-url}/wombat/`). Required for connectivity.
`agent.wombatToken`	The security token required to authenticate API requests if your WombatDialer instance is protected.
`agent.wombatPreviewChan`	Defines the PBX identification for the agent. Uses the `${agent}` placeholder to dynamically insert the extension (e.g., `PJSIP/${agent}`).
`agent.wombatAttr.extStatusSkip`	A comma-separated list of reasons an agent can select when skipping a reserved call (e.g., `SK1,SK2`).
`agent.wombatAttrsIn`	Comma-separated list of input attributes from WombatDialer visible to the agent. If omitted, all attributes are shown.
`agent.wombatUrl.{attributeName}`	Defines an external link for an input attribute. Supports `${attributeName}` placeholders for dynamic URL generation.

Property	Explanation
`agent.wombatAttr.extStatus`	Defines the dropdown values for the "Extended Status" field, setting the final call outcome (e.g., `SALE,DONTCALL,RECALL`).
`agent.wombatAttrsOut`	Comma-separated list of output attributes (custom fields) the agent can fill in (e.g., `comment,dept,reason`).
`agent.wombatAttr.{attributeName}`	Converts an output attribute into a dropdown menu by providing a comma-separated list of allowed values.

Full Configuration Example

```
# Connection to WombatDialer
wombat.previewDialingUrl=https://wombat.example.com/wombat/
agent.wombatToken=mySecureToken123
agent.wombatPreviewChan=PJSIP/${agent}

# Input attributes (visible to agent)
agent.wombatAttrsIn=name,surname,caseid

# CRM links per attribute
agent.wombatUrl.caseid=https://crm.local/ticket/id=${caseid}&user=${surname}
agent.wombatUrl.surname=https://crm.local/search?q=${surname}

# Call outcome (Extended Status)
agent.wombatAttr.extStatus=SALE,DONTCALL,RECALL

# Skip reasons
agent.wombatAttr.extStatusSkip=SK1,SK2

# Output attributes (custom data the agent fills in)
agent.wombatAttrsOut=comment,dept,reason
agent.wombatAttr.dept=IT,SALES,HR
agent.wombatAttr.reason=PRICE,SUPPORT
```

Keyboard Shortcuts

Action	Shortcut	Description
NAVIGATION_OVERVIEW	`ctrl + alt + o`	Navigates to the Overview page
NAVIGATION_INTERACTIONS	`ctrl + alt + i`	Navigates to the Interactions page
NAVIGATION_CONTROL	`ctrl + alt + c`	Navigates to the Control page
NAVIGATION_CALENDAR	`ctrl + alt + l`	Navigates to the Memo Calendar page
NAVIGATION_CLIENTS	`ctrl + alt + t`	Navigates to the Clients page

Action	Shortcut	Description
TOPBAR_PHONE	`ctrl + alt + p`	Toggles the Softphone panel
TOPBAR_NOTIFICATIONS	`ctrl + alt + n`	Toggles the Notifications panel
TOPBAR_STATUS	`ctrl + alt + a`	Toggles the Agent Status panel
OVERVIEW_START_WORK	`ctrl + alt + w`	Toggles the Start/End work button on the Overview page
CONTROL_PAUSE_UNPAUSE	`ctrl + alt + b`	Controls the Pause/Unpause function on the Control page.
CONTROL_CHANGE_AMEND	`ctrl + alt + m`	Controls the Change/Amend pause function on the Control page.
CONTROL_JOIN_SELECTED	`ctrl + alt + j`	Joins the selected queues on the Control page.
CONTROL_LEAVE_SELECTED	`ctrl + alt + u`	Leaves the selected queues on the Control page.

Softphone Keyboard Shortcuts

Here are the keyboard shortcuts for the Softphone feature in QueueMetrics:

- Press `Enter` while typing in the input box: Dial a number.
- `Ctrl + e`: Hang up the current call.
- `Ctrl + +`: Add a new session.
- `Ctrl + a`: Answer an incoming call.
- `Ctrl + b`: Access the Transfer panel.

Softphone System Parameters

To customize the Softphone experience, you can configure the following system parameters:

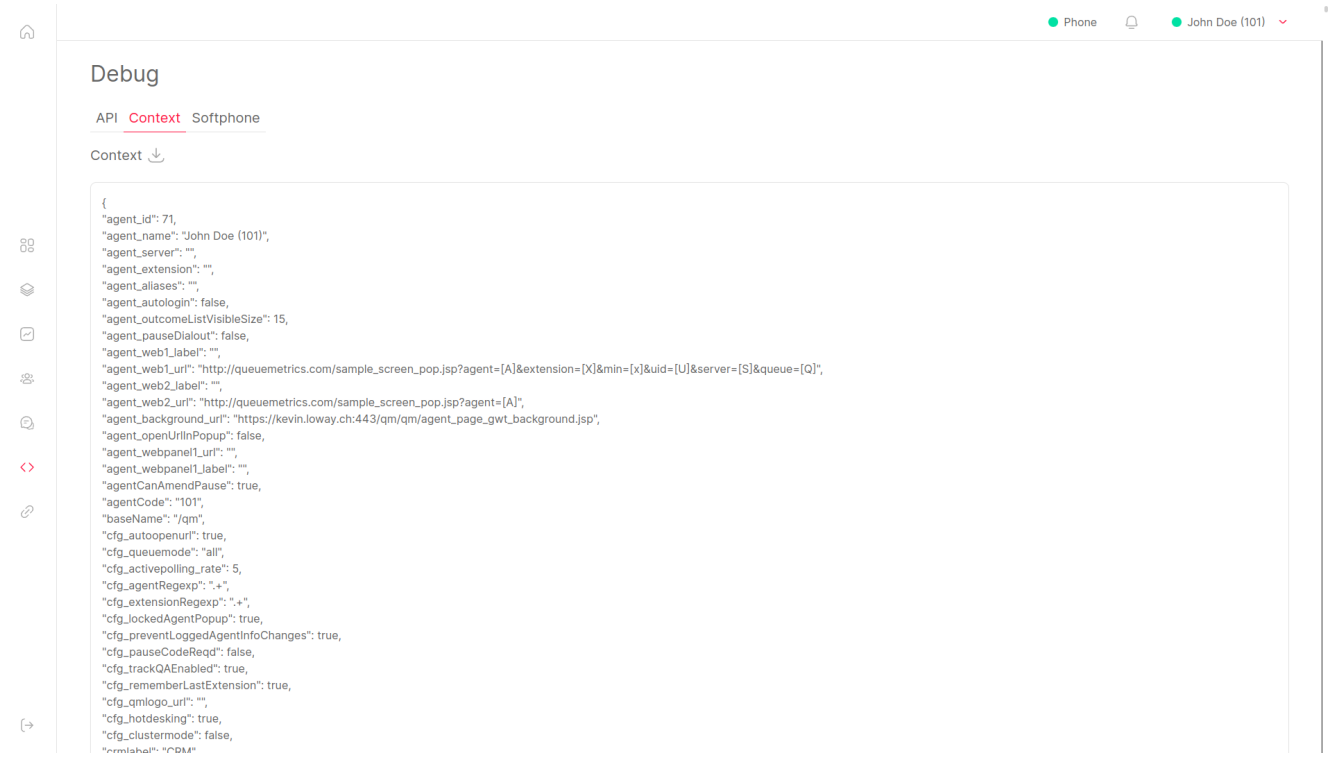
- `qap.answer_src`: Specifies the source path of the answer tone audio file.
- `qap.hangup_src`: Specifies the source path of the hangup tone audio file.
- `qap.dialtone_src`: Specifies the source path of the dial tone audio file.
- `qap.ringtone_src`: Specifies the source path of the ringtone audio file.

These parameters allow you to tailor the audio and behavior of the Softphone to your specific needs.

Debug Page Activation

This section explains how to enable the debug page for issue diagnosis or support purposes. The

debug page provides valuable information for troubleshooting.



To enable the debug page, follow these steps:

- Open the QueueMetrics agent page in a browser.
- Modify the URL by adding `&debug=true` just before the "#" symbol.

For example, if your original URL looks like this:

```
https://myipaddress/queuemetrics/qm_qap_load.do#/overview
```

Modify it to include debug mode as follows:

```
https://myipaddress/queuemetrics/qm_qap_load.do?debug=true#/overview
```

Once you've added `&debug=true` to the URL, reload the page. The debug page will now be accessible, providing valuable information for debugging any issues.

Enabling debug mode is particularly useful when you encounter problems and need to diagnose them or when seeking assistance from the support team.

Assisted Manual Outbound

Assisted Manual Outbound, shortened as AMO, is a simple way to manage list of calls to be processed manually by your agents.

AMO is distinct from, though in a sense complementary to, other kinds of manual outbound and recall memos.

- **Passive outbound**: as tracked by Uniloader. An agent simply places an outbound call on her phone, and it is tracked as `q-outbound`. Nothing special happens with it.
- **Manual outbound**: agent dials a number that she got from an external source (post-it, copy and paste from another aplication) She knows this is a work call for campaign `Cars`, she flags it as such on QM, so the supervisor can see her correct status on the RT page and it is tracked as a work activity
- **Recall memo**: during a conversation, the agent schedules a recall for the next day. She puts the information she has on a recall memo so she can see it in her daily calendar for the next day, start the call in one click, know who/what it's for, and not have to depend on her notes.
- **WombatDialer-based recall**: if WombatDialer is present, it is possible to schedule a recall to be made at a specific date-time through WombatDialer. This works like a recall memo, but usually happens automatically.
- **WombatDialer-based preview dialing**: agent uses WombatDialer to dial calls; by clicking on the right sub-page on QAP (the one with the WombatDialer icon), she accesses WombatDialer's own preview dialing panel, sees the details of a call to be made and either accepts or refuses it.
- **Assisted manual outbound**: agent just pulls the "next" call she has to make. The system knows on which campaigns she is supposed to be working, so gets her an available call. She can start the call in one click, see all relevant information (including previous call history), and can work on the call with the minimum number of clicks. She cannot choose which call to make, though it's up to her to request for a new call or do something else. The call is started an logged with all relevant information, and can easily be connected to a CRM.

All outbound modes work with a physical phone, a soft-phone, or QueueMetrics' own WebRTC phone embbedded on the Agent page.

Assisted campaigns

Any campaign (that is, an outbound queue) can be used for AMO dialing. This is set in the configuration of the queue. So, when an agent is logged into one or more outbound queues that have been set for AMO, their outbound panel/phone shows an icon for "Next call".

This call will come randomly from any one relevant campaign that the agent is logged on to. It is possible that there are no next calls to be made, in which case a suitable message is displayed. The AMO button is always available to the agent, as numbers may be loaded dynamically to any existing campaign.

The mixer

Each assisted campaign has a "mixer", that is a set of call lists that are linked to it, a weight for each list, and a toggle to pause/restart each list. The next call is pulled from a list by choosing randomly from all available lists, based on their current weight.

By using the mixer, a supervisor can change the weight of a list during operation to prioritize certain calls over others.

The mixer marks the campaign as "completed" if all calls from active list are done, that is, pulling a number from it was impossible and all calls are fully processed (all calls have reached a final state, and no further recalls are pending). Completed campaigns are logged on the system log. A completed campaign will become open if:

- a new list is added to it
- new numbers are added to an active list
- new numers are returned to an active list - e.g. agent reserved a numer, called it, but it needs to be recalled.
- a paused call list in unpaused

Rescheduling

The campaign has fields for rescheduling failed calls; this specifies the minimum validity interval for next recall and whether the retry period is fixed, linear or exponential. It also specifies the set of outcomes that are accepted as a successful attempt. The outcome code "NONE" matches calls that have no outcome.

Rescheduling happens:

- when a call has a final status, that is, an outcome is explicitly set for that call, so AMO can decide whether recalling is to be done or not
- when a call status times out; for example when a number that was reserved has not been used in a few minutes, or after one hour a call has been made but not marked with an outcome. This means that when doing AMO, it is very important that agents enter an outcome for all calls, so that they are rescheduled quickly.

Attributes

Each assisted campaign also has a number of attributes that are taken from the call information and logged to the outbound call that was made, so that it is possible to "spot" it in reports. Usually only a subset of call attributes are actually logged to the `queue_log` - some are used only as context for the agent.

For example, a call list may include a case-id and a the first and the last name of the person so that the agent makes sure she's talking to the right person, but only the case-id is actually sent to the `queue_log` so that it will be visible in reports as a variable.

In any case, for AMO calls, the attribute `QM_NUMID` is always saved. This makes it possible to spot

AMO calls even when those calls are completed, and to link back to the call's AMO history.

Call lists

Call lists are single-use, and are "consumed" as numbers are pulled from them. Each list has a name and belongs to one campaign.

Lists are fed:

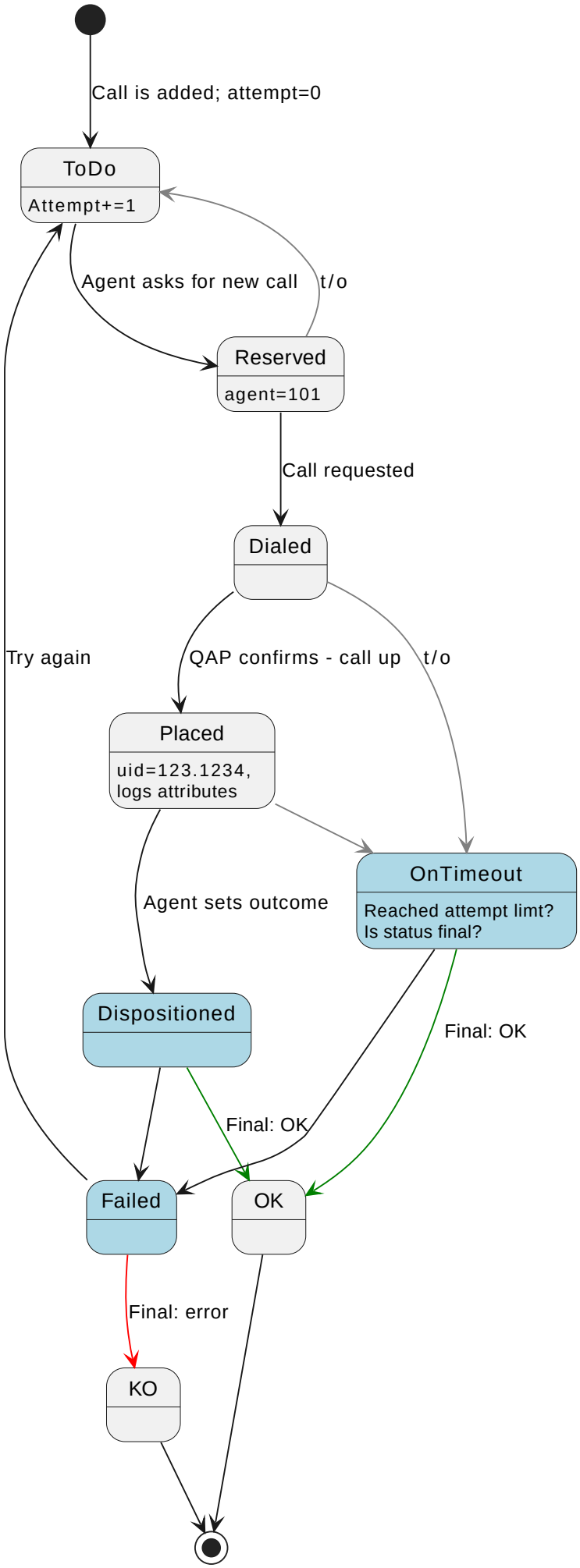
- manually: numbers are edoted through the AMO Numbers editor.
- through a CSV: the AMO Lists editor allows uploading a complete CSV file, all at once..
- from other lists on other campaigns
- through API calls, as references on the JSON API manual.

As in WombatDialer, a list contains information for a call to be made, and for each number:

- the number to be called
- a validity date for the next recall - empty to imply any time is valid.
- a set of attributes to be shown to the agent when dialing and possibly be logged
- the recall history, that is the list of previous attempts (who/when/uid/status)

A call's lifecycle

- Agent asks for a new call. By checking the AMO campaigns that they are logged on to, first a campaign and then a list is selected. The list's weight gived the probability that a given list is chosen. If another agent asks for a call, the one that agent reserverd got won't be available, even if she's not started it yet.
- When agent actually dials the call, she gets a screen showing all call details, including previous recalls, so she is ready for the callee picking up.
- As soon as the Agent's page "spots" the call on the call list, the call is confirmed, and any required attributes are logged. Note that we have no way of knowing the UID of a call before it is made.
- When an agent enters an outcome for the call, it is checked about the campaign's setting to decide whether the state is final or not.


Call is added; attempt=0
ToDo
Attempt+=1
Agent asks for new call
t/o
Reserved
agent=101
Call requested
Dialed
Try again
QAP confirms - call up
t/o
Placed
uid=123.1234,
logs attributes
OnTimeout
Reached attempt limt?
Is status final?
Agent sets outcome
Dispositioned
Final: OK
Final: OK
Failed
OK
Final: error
KO

Failures and recalls

There is a background process that handles call time-outs. Agents may or may not enter an outcome for a call.

- Call reserved: is recycled back in 5 minutes if it does not progress, does not increment attempt counter
- Call dialed: goes to reschedule in 5 minutes - this means no logs were received, and should never happen
- Call placed, no further information: Reschedule by reading property "amo.placedCallTimeout", default 3600 seconds.
- Call has outcome:
  - Positive outcome: Call completed
  - Negative outcome: Reschedule if needed

QueueMetrics Users

Default users

The following users come pre-configured in the default database.

Table 2. Default QueueMetrics Users

Login	Password	Enabled by default?	Explanation
demoadmin	demo	Yes	The sample admin user
demouser	demo	Yes	The sample CC manager
demovisitor	demo	No	A sample visitor
demosupervisor	demo	No	A sample supervisor
robot	robot	No	A sample robot
webqloader	qloader	No	Access point for Uniloader
Agent/101	999	Yes	A sample agent
Agent/102	998	Yes	Another sample agent

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

Please note that some users are present but NOT ENABLED by default with the default database schema supplied with QM. You need to enable them manually if you need them.

Password reset

By using Uniloader, it is possible to "lock" a QueueMetrics user so that they must access QueueMetrics through a special link to have their password reset. This way the administrator never knows about the actual password used by the user, that is stored in an encrypted format and not recoverable.

Links have a definite validity (usually 5 days, including the current one) and when they expire, you need to repeat the process again to generate a new one if it was not used before its expiry date.

The procedure is documented in the Uniloader user manual, see section "Password Reset" in chapter "User information".

Restricted user administrators

A user holding the key `USRADMIN` can edit all users and classes. If the user also holds the key `USRADMIN_RESTRICTED`, the key `USRADMIN` is limited as follows. The classes the user can manage are set in the property `admin.classesWhenRestricted`, a list of class names separated by spaces, where the default is just the class `AGENTS`.

Restricted user administrators can see all users and classes, but:

- They can create users only in the allowed classes, with no custom keys and with the master key turned off.
- They can save, clone and delete users in the allowed classes, but they can never change a user's custom keys or master key.
- They can save a user that belongs to any other class, but only to change its name, e-mail or comment, to disable it, or to move it to an allowed class. They cannot change its password or token, enable it, or move it to another class that is not allowed.
- They cannot save, clone or delete classes.

Any other operation is refused: an error is shown and the operation cannot proceed.

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.



For security reasons, this user must be manually enabled in the standard database.

Using an external auth source for QueueMetrics

QueueMetrics can accept an external authentication source. By using a simple JSON/XML-RPC/LDAP interface, it is possible to write third party scripts that are able to authenticate against a chosen authentication system (e.g. Kerberos, Microsoft Active Directory, a Microsoft domain server, etc).

Overview

When an user logs in to QueueMetrics, QM checks to see if there is an auth source defined. QueueMetrics will execute a query to that third-party server passing along the user-id and the

password given for authentication, plus a given service-id that will be defined in the configuration file.

The server will respond passing along a fixed set of data:

- An auth response (see below)
- A set of login information for that user (e.g. real name, email, and possibly a different log-in)
- The current class and user keys for that user

If the class passed along is empty, only the user keys will be computed. If the class is not empty, it will be searched by name on the QueueMetrics storage and it will be loaded. Trying to load a non-existent class will result in auth failure.

The auth server may return four different responses:

- `S`: Access successful allowed
- `A`: Access allowed with supplied user data
- `F`: Access is forbidden
- `D`: Access is fully delegated to QM

The following table explains the relationship between those states:

	Auth OK	**Auth KO**
User data on auth server	AUTHORITATIVE	FORBIDDEN
User data on QM	SUCCESSFUL	DELEGATED

If auth is **SUCCESSFUL**, the local QM database is checked for that user name. If such a user is present, the user details, class and key information are loaded from QM. If such a user is not present, the details are taken from the ones supplied by the service.

If auth is **AUTHORITATIVE**, the details are taken from the ones supplied via RPC. Then they are copied to the local user database (with a random password) so that although the user cannot login manually, it is possible to decode the user name for all logged operations (e.g. Updating a queue). If a user with the same name is present, credentials are forcibly updated with the authoritative credentials.

The following values are implied in authoritative replies:

- Enabled = yes
- Masterkey = no

The actual user data sent by the auth server is only read by QM in case "A"; otherwise it's ignored, whatever is passed.

If auth is **FORBIDDEN**, no other check is done and the user is rejected access.

If auth is **DELEGATED**, the standard QM logon procedure is done; so the remote auth source is basically telling QM to repeat the log-on process as if there was no external auth involved. This way, you can first check on the remote system, and you delegate any failing user to QM, so you can have a mix-and-match situation. Some drivers let you input a list of ignored users that are in any case delegated, effectively bypassing them.

The whole procedure is totally transparent to the user, so they do not need to know which is the authority that grants or denies access.

Driver: JSON

By setting the properties:

```
auth.externalSource=json
auth.verboseLog=true
default.authSystemId=sys1

auth.jsonServerUrl=https://myserver.....
auth.jsonPost=false
auth.jsonIgnoreUsers=webqloader,robot
```

A request will be sent to the URL specified, with a JSON payload in parameter `query`, as below:

```
{
  "login" : "demoadmin",
  "password" : "password",
  "systemId" : "sys1"
}
```

The result must have the following shape:

```
{
    "response": "A",
    "login": "demoadmin",
    "realname": "John Doe",
    "email": "john@home",
    "classname": "AGENTS",
    "keys": "A B C"
}
```

In the case of "A" (Authoritative) you need to fill all parameters, while in all other cases the `response` will be enough.

If there is a `login` field in the reply, then the returned login will be used on QM - this is especially useful for agents, where QM demands a format like `Agent/123` while you might use a free form, and return the correct agent code if authentication is successful.

The *ignore* parameter is useful for JSON services / web loading, to avoid hitting the server on each request, or if you don't want to have QM-specific users on your server (eg. a company's LDAP server you don't manage).



You can test JSON auth by using ready-made answers, so you don't have to write a server yourself - see in: https://github.com/Loway/OpenQueueMetricsAddOns/tree/master/json-auth-examples

Driver: LDAP

The LDAP driver lets you query an external LDAP/LDAPS server; you define both the server and the user query to be checked.

```
auth.externalSource=ldap
auth.verboseLog=true
default.authSystemId=sys1

auth.ldapServerUrl=ldap://ldap.forumsys.com:389
auth.ldapBind=cn=${login},dc=example,dc=com
auth.ldapFailureDelegates=true

auth.ldapLoginAttr=msDS-cloudExtensionAttribute7
auth.ldapBase=CN=Users,dc=example,dc=com
auth.ldapQuery=(sAMAccountName=${login})

auth.ldapIgnoreUsers=webqloader,robot
```

All strings can be expanded with the following placeholders:

- `login`: the login string enter by the user in QM
- `password`
- `systemid`: the `authSystemId`

The `ldapBind` parameter is the log-in that lets your user access LDAP; it is usually the Distinguished Name (e.g. `CN=John Doe,CN=Users,DC=example,DC=com`) or the Account Name or the user's e-mail, depending on system configuration.

LDAP can be used in two ways:

- you use the same log-ins in QM and LDAP; in this case you only need to set the `ldapBind` parameter to your "login"
- you cannot use the same log-ins in QM and LDAP, e.g. because users have a company-wide LDAP policy set, and you have no control over their names and logins. This may be a problem for agents, where QM needs the `Agent/101` format that matches their extension.

In this second case you need to specify an attribute on your users to contain the login that will be used in QM for them. In ActiveDirectory, for example, the default schema has a set of empty

Extension Attributes that may be set freely. They do appear in LDAP with a name like `msDS-cloudExtensionAttribute7`. In one of them, you should specify the QM login to use.

To use them, you need to set `ldapLoginAttr` to the attribute to use, the `ldapBase` to where to start searching in the LDAP tree (typically, `Users`), and a LDAP query that is supposed to return only one record that finds the right user on the basis of the user-supplied login. Missing attributes, users, or multiple results will be logged on the system log.

At the end of your LDAP authentication, the only positive auth will be SUCCESS - so a user with the same name is supposed to be present in the database.

If the auth fails, we check `auth.ldapFailureDelegates`: if true, auth fails with DELEGATES; if false, as by default, it is FORBIDDEN.

As for JSON, it is possible to have a set of users that are ignored for LDAP and effectively checked only on the local database; this is helpful for webservices or data loaders, especially when you cannot create service users in LDAP for third-party applications.



To get started with LDAP, you will need a tool to access it, as names and attributes are different on different systems. A tool like `ldapsearch`, available on most Linux systems, will let you test binding parameters from the command line; once they work, you can start configuring it in QM.

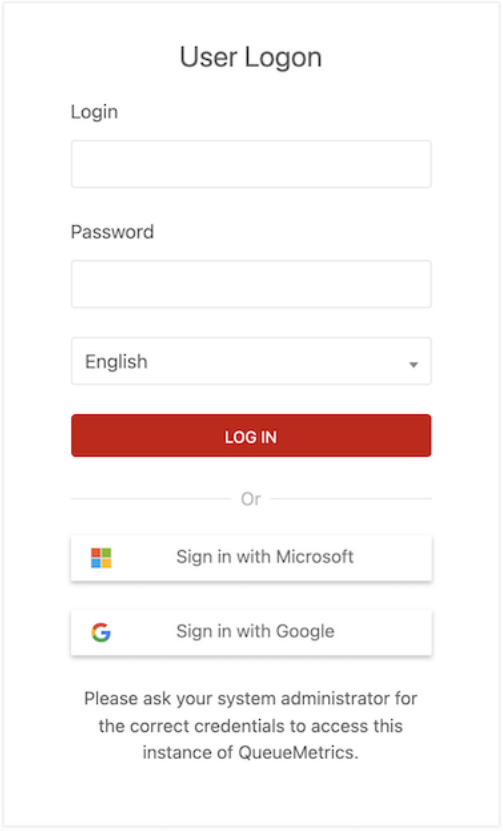
Driver: XML-RPC

This driver is obsolete, see the XML-RPC manual for more details.

```
auth.externalSource=xmlrpc
default.authRpcServerUrl=http://myserver/auth-xmlrpc
```

Using Single Sign-On with QueueMetrics

QueueMetrics is compatible with some Single Sign-On (SSO) providers that are based on OAuth2, and can offer a very simple SSO flow where the QueueMetrics user can just click on the right button to authenticate.



So, when an user clicks on the provider's button, they are asked for consent to allow log-in to QueueMetrics; from them on, if they are logged on to the provider, the just click on the button and find themselves within QueueMetrics.

Overview

Prerequisites

- A system administrator creates an **application** in the OAuth provider of choice. QueueMetrics will act on behalf of this application to authenticate users. The QM instance will not share any data with Loway.
- A system administrator enables one or more OAuth drivers in QM by setting some configuration properties.
- The QM server has internet HTTP/S connectivity towards the required auth source
- The end-user who needs to log-in has direct HTTP/S connectivity towards the required auth source
- QM is served through HTTPS

Flow

- A system with OAuth enabled will display the familiar (say) "Sign-in with Google" button below the main login form in QM, depending on which driver(s) are active.
- When a user wants to use an OAuth source, they click on the button and are sent to a transaction where the user consent is gathered to share data with the application (and QM running it). Data to be shared is the current user's email and name. If the user is already logged

on the platform and has already given consent, no manual action has to be performed.

- If the user consents, then a secure token is returned to QM:
  - QM will then issue a request to the service, using the secure token, to read the user's profile: user-name, unique identifier and email
  - If an existing user with the same email is present in the QM database, and is flagged for OAuth access, and is enabled, then that user's profile will be used for log-in and its password will be ignored.
  - If a user with the same email exists, but the user is not flagged for OAuth access or is not enabled, then login will be refused as if the user was blocked
  - If no user with that email address is present, a user is created in a placeholder class, so that its name, email and unique-id will be created on the database. They won't be able to log anyway in until an administrator approves them. See Creating users.
- If the user does not consent, or the login procedure fails on othe OAuth provider for any other reason, then a log is written and the user goes back to the login page
- Ordinary users will co-exist, and they will log in normally from the login form. If an external auth source (LDAP, JSON...) is set, it will be used normally.

In practice: a Google flow

- Alice Doe is an agent in QM, where she is known as `Agent/103`. She has an account on Google as `alice@compa.ny`
- She clicks on the Google button, and goes to a consent form. On the consent form she will see the name of her company, as that name was set up in the application in Google. She logs in with her email and password as she usually does.
- If all goes well, QM is returned a secure token
- QM queries her Google profile via API (therefore asserting that the token is indeed what is is supposed to be), and finds out that her full name is "Alice Doe", her email is "alice@compa.ny" and her Google ID is "123456"
- QM goes looking for a user which email field is set to `alice@compa.ny` and who has the token field set to `google:123456` in their profile:
  - QM finds such a user; their id happens to be `Agent/103`; at this point a full login is performed and Alice sees herself logged in as `Agent/103`
  - QM does not find such a user: then it will create a new user `Alice.Doe`, with email `alice@compa.ny`, and flag `google:123456` in a designated class. User will be displayed an error stating that log-in is not possible.

Why flagging users for access through the token field

The format of log-ins for agents in QM is constrained; `Agent/103` will work but `John.Doe` won't. That's why QM uses the email address as a pivot to find the right user to log in. This is turn causes a possible security issue - if you have `alice@compa.ny` and create a second account with the same e-mail on a different provider, then you would be able to log-in with a different set of credentials.

QM instead validates the presence of a token `google:12345` or `microsoft:12345` so that we can assert

that:

- Only that driver can be used for logging in
- The correct provider-id is used (unless you user e.g. `google:*` to allow any Google account)
- Multiple assertions may be present, e.g. `google:* microsoft:*`

Creating users for SSO

To enable a QM user to log in with SSO:

- you set their email address to the one returned by the SSO provider,
- and set their `token` field to the correct value, e.g. `google:*`

As this is sometimes unwieldy, you can have QM automatically create a template user when someone accesses QM through SSO.

Automatic creation of foreign users

To create a new user, the sysadmins must know not only the email, but also the service and possibly the exact reference id for the user in the exact auth provider.

To make their life easier, we automatically create unknown users into a placeholder class, usually `AGENTS`. Therefore in the example above, to enable Alice for proper access, the sysadmin will only have to change her class and possibly the log-in name (for agents), but the full name and the identity assertions will already be present.

When a user tries to log in with a user that is correct on the OAuth side but does not exist in QM:

- If a placeholder class is set in the configuration:
  - A user is created in QM with:
    - its log-in is set as `Agent/` plus the name of their e-mail account
    - its E-mail and Full name set as specified by the OAuth provider
    - its password is set to an invalid value
    - the user is in any case set as blocked, so that it is impossible to log in
    - the user is pre-set with the correct service token, e.g. `google:12345`
  - The user is shown an error message that specifies that the log-on is not possible.
  - An administrator will have to approve the user by unblocking her and (optionally) assigning it to the right class.
  - If a user with the same login is already present, then no new user is created.
- If no placeholder class is set, the login simply fails.



The administrator needs to manually unlock any such user, after editing their profile - for example by assigning them to the correct class or modifying their login. Just make sure that the email and token fields are left unaltered.

Security considerations

An OAuth user exist on the QM database, but cannot be used for log ins as their password is invalid. Therefore, if a user has the grants to set their own password **they can bypass the OAuth flow entirely**. Therefore we strongly recommend removing the key `CHPASSWD` before allowing direct log-ins.

On the other hand, this can be used by an administrator in an emergency where the OAuth server is unavailable to allow access to QM by setting ad-hoc passwords and changing them after the event.

It is strongly advisable to keep at least an admin user as QM-only to allow emergency access in case the OAuth service should be unavailable.

Logging

The following items are logged on the Audit logs:

- OAuth consent obtained (before authentication is done, so it could still fail)
- OAuth consent failed (if the provider can signal it)
- OAuth profile access failed
- User locked out / missing grants
- New user created from OAuth

Limitations

As OAuth users depend on human interaction, such users **won't work for webservices** (unless a password is set). This is not a major limitation, as you would usually create local users for webservice acces.

OAuth users can be instead users to run scheduled reports, as QM impersonates them without the need of password authentication.

The return URL

In order to "jump back" from the auth flow, QM creates a return URL, that is basically the public name of the server QM instance plus the string "/qm_sso_return.do": for example, it might be `https://test.loway.ch:443/queuemetrics/qm_sso_return.do`.

As most SSO providers will require this to be an HTTPS URL, QM rebuils the public name based on the headers it receives from its upstream proxy implementing HTTPS, e.g. NGINX.

To see the current URL, as an administrator, go to "Diagnostics" → "View configuration" → "HTTP Configuration" and check the value that appears in "Q: getPublicQmBaseUrl".

SSO drivers: Google

The driver is called `google`, and will work in the case you use Google to manage your organization's credentials.

Creating credentials

First you need to log in to Google Cloud Platform, select "Dashboard" and then "Create project".

- Project name: anything you want
- Organization and Parent resource: your organization

Now select your new project, and select "APIs and Services":

- first select "OAuth consent screen" (left column). Enter App information (name - that is what your users will see as the entity requesting access - and email), select Audience: Internal, then your contact email, then you are done.
- then click on "Clients", "Create OAuth Client ID", select type "Web application". Now add an "Authorized redirect URI", as `https://my.server.x/queuemetrics/qm_sso_return.do` - it must be the full HTTPS name of your QM instance plus `qm_sso_return.do`
- once the client is created, you will be prompted to download a JSON file. Do not lose it! it contains the credentials you need to set up QM.

Configuring QueueMetrics

First, take the values from `client_id` and `client_secret` from the JSON file you downloaded, and set your configuration properties like:

```
sso.driver=google
sso.placeholderClass=AGENTS
sso.verbose=false

sso.google.clientId=XXXXXXX.apps.googleusercontent.com
sso.google.clientSecret=GOCSPX-xxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Now restart QM and the home page will present a new Google button. By clicking on it, you can gather user consent and a new user will be created in QM. You can then unlock this user, change the login as needed, and have them log in again.

SSO drivers: Microsoft

The driver is called `microsoft`, and will work in the case you use Microsoft (e.g. Teams) to manage your organization's credentials.

Creating credentials

Log in into the Azure portal, click on "App registrations", click on "New registration".

- As `Name` enter the name you want to show in Azure
- For "Supported account types" select "Accounts in any organizational directory (Any Azure AD directory - Multitenant)"
- For "Redirect URI", select type "Web" and add

`https://my.server.x/queuemetrics/qm_sso_return.do` - it must be the full HTTPS name of your QM instance plus `qm_sso_return.do`

- Now click on "Register"; you will be driven to the newly created app
- Now click on "Certificates and secrets", select "Client secrets", then click on "New client secrets", add a name and a duration, and a new secret will be created for you. You can now copy the "Value" and paste it somewhere, but just once, so store it securely.
- Now go to "Authentication", find "Implicit grant and hybrid flows" and check both "Access tokens (used for implicit flows)" and "ID tokens (used for implicit and hybrid flows)", and make sure that "Supported account types" is set to "Multitenant".
- If you want, from "Branding and properties" you can set up a consent screen with your logos.

Configuring QueueMetrics

You will need the Application ID and the Tenant/Directory ID that you can find on the home page of the app you just created, plus the secret.

```
sso.driver=microsoft
sso.placeholderClass=AGENTS
sso.verbose=false

sso.microsoft.applicationId=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
sso.microsoft.tenantId=yyyyyyyy-yyyy-yyyy-yyyy-yyyyyyyyyyyy
sso.microsoft.password=xxxxx~xxxxxxxxxxxxxxxxxxxxxx.xxxxxxx
```

Now restart QM and the home page will present a new Microsoft button. By clicking on it, you can gather user consent and a new user will be created in QM. You can then unlock this user, change the login as needed, and have them log in again.

Generate PDF and XLS reports and send them by e-mail



The information here is obsolete - see Scheduled jobs instead.

By using the ROBOT profile is possible to generate PDF and XLS reports and send them to a list of e-mail recipients. The report parameters, like queue, period, report set and the list of recipients should have been already defined in the Report ExportConfiguration pages.



To generate the PDF and XLS reports the ROBOT user should hold the USR_REPORTS_EXPORT key and all optional security keys assigned to each export job.

You will usually run these jobs automatically by setting up a Scheduled job, though you can have reports generated on-demand by downloading the following URL (you will probably use `wget` or a similar tool, and will script this as a cron job so that it will be run daily or weekly):

`http://server:8080/queuemetrics/qm_export_stats_run.do?username=robot&password=robot&exportId=X`

where the exportId *X* parameter is the job number identifier that could be found on the Report ExportConfiguration pages.

The server generates the required report and send it to a list of recipients. It also returns a copy of the generated document.

If you don't want QueueMetrics to send e-mails but just need to retrieve the file, avoid configuring any e-mail recipients.

If there is an error during the document generation or e-mail sending phase, then an error message is returned instead.



Please note that the some configuration settings are required for QueueMetrics when sending e-mails. Further details could be found on the System Preferences. We suggest turning on SMTP debug when sending your first e-mails to understand what is going on, or use the SMTP debug page at SMTP test .



As QueueMetrics has no way to know which language you want to use for the report, and the character set to use in it, make sure that you set the parameters *default.language* and *default.country* to the correct values.

Setting up a self-service wallboard



The information here is obsolete - see Running a wallboard in kiosk mode instead.

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
```

or, for the ajax based page version

```
http://server/queuemetrics/qm_wab_ajax.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* or the *qm_rt_ajax.do* commands to create a very simple real-time monitor running all day long for your supervisors.



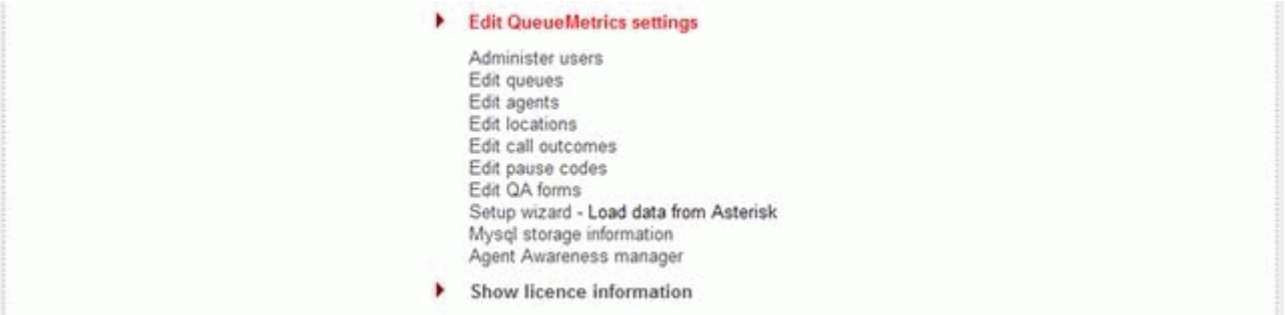
Please note that there is a difference between results produced by the XML rpc realtime calls and the realtime statistics produced through the QueueMetrics GUI

when the key *realtime.members_only* is equal to true. The difference is related to the agents list shown. Being the list of queues, in the XML-RPC call, specified by a list of names instead of a list of queue unique identifiers, is not possible to correctly identify elementary queues from macro queues having the same name. In this situation the agent list will always be calculated as sum of all agents associated to all elementary queues composing the macro queue, even if the macro queue has directly assigned agent.

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:

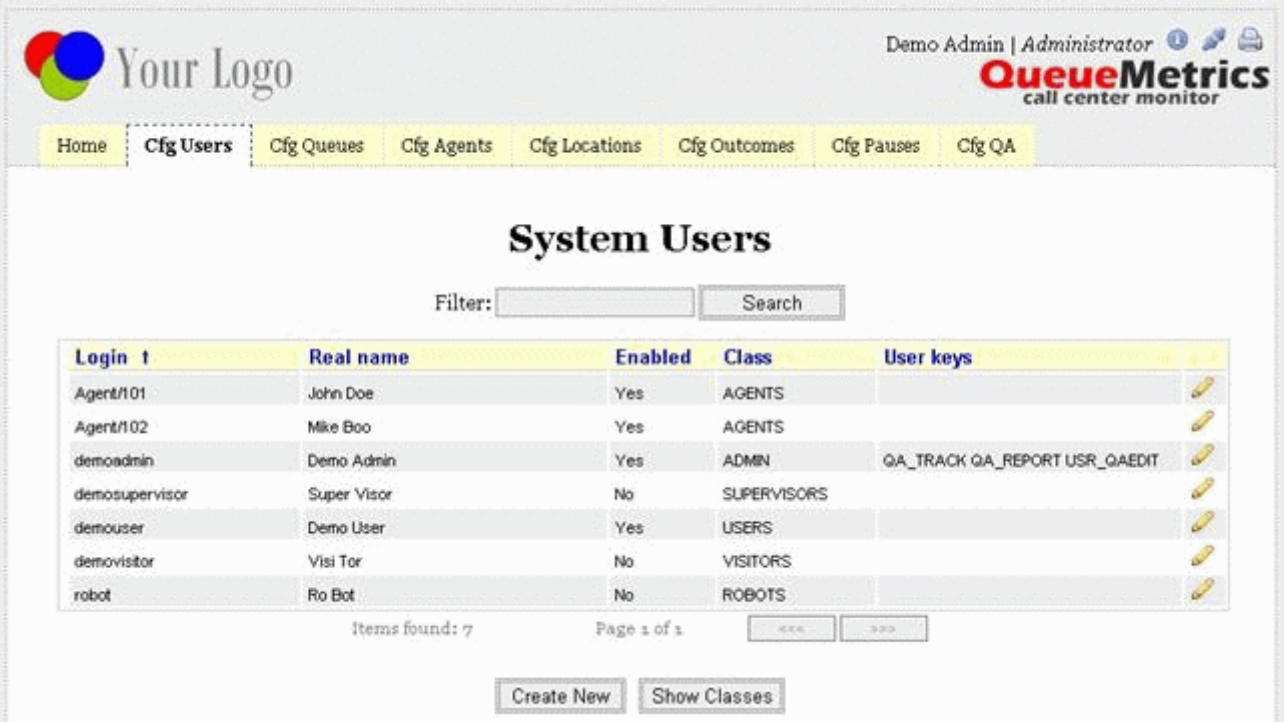


Editing a setting, or even displaying the full detail of a specific item, is always protected by security keys, and in any case is logged on the Audit Log for further inspection - see The Audit log for more details.

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.

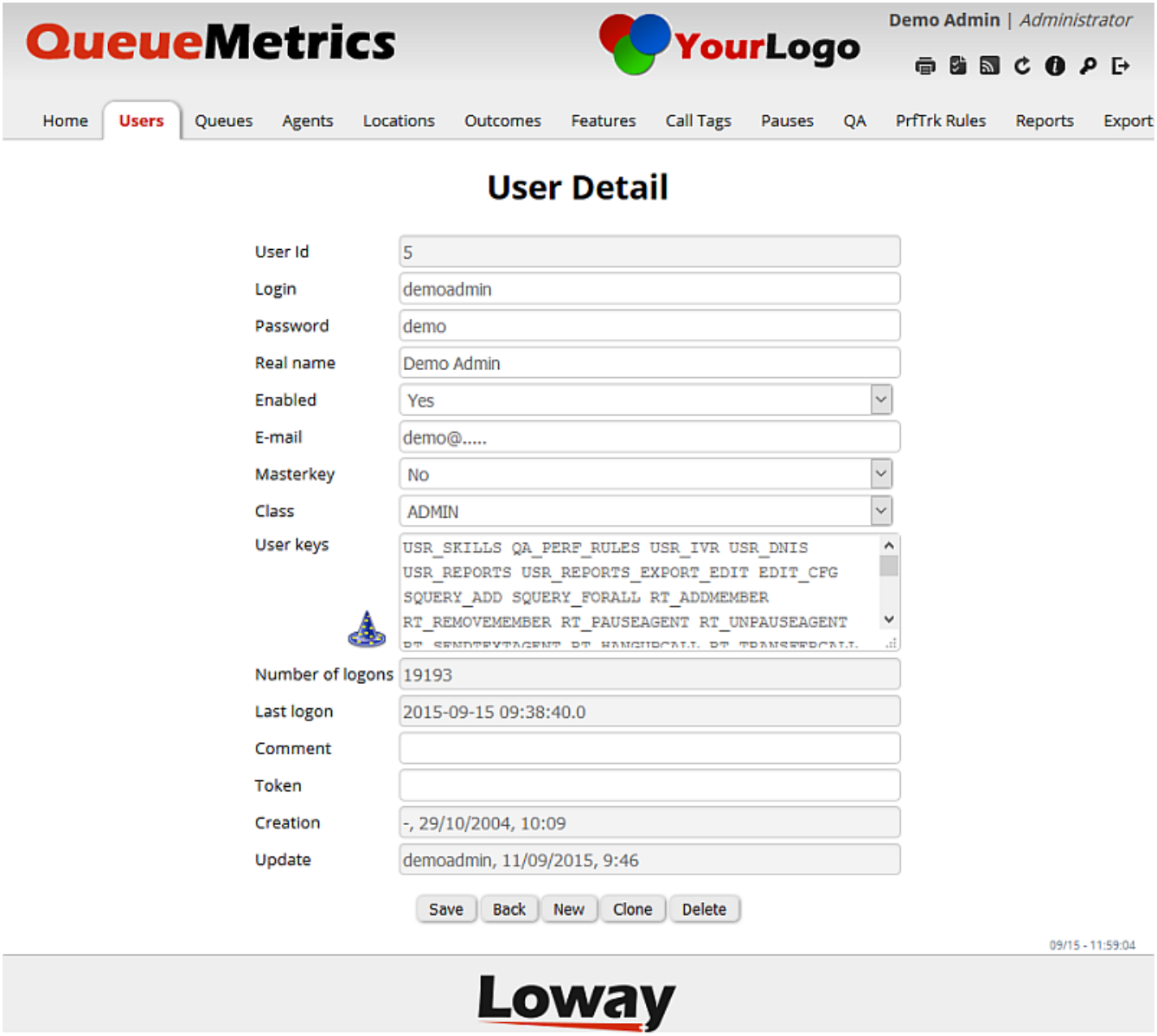


For each user, the login and full name are shown, together with the current class and any additional user keys. A user must be enabled in order to log on, so if you want to prevent somebody

from logging on without deleting its user information, you can simply disable it. *A number of default users are shipped with Enable: No in order to prevent unauthorized access.*

The list of users is paged and you can use the top box named "Filter" in order to search for a specific user or a set of users that match the entered substring. You can also click on the column name to toggle ascending versus descending sort order.

The "Create new" button lets you add new users while the "Show classes" button leads to the class editor.



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

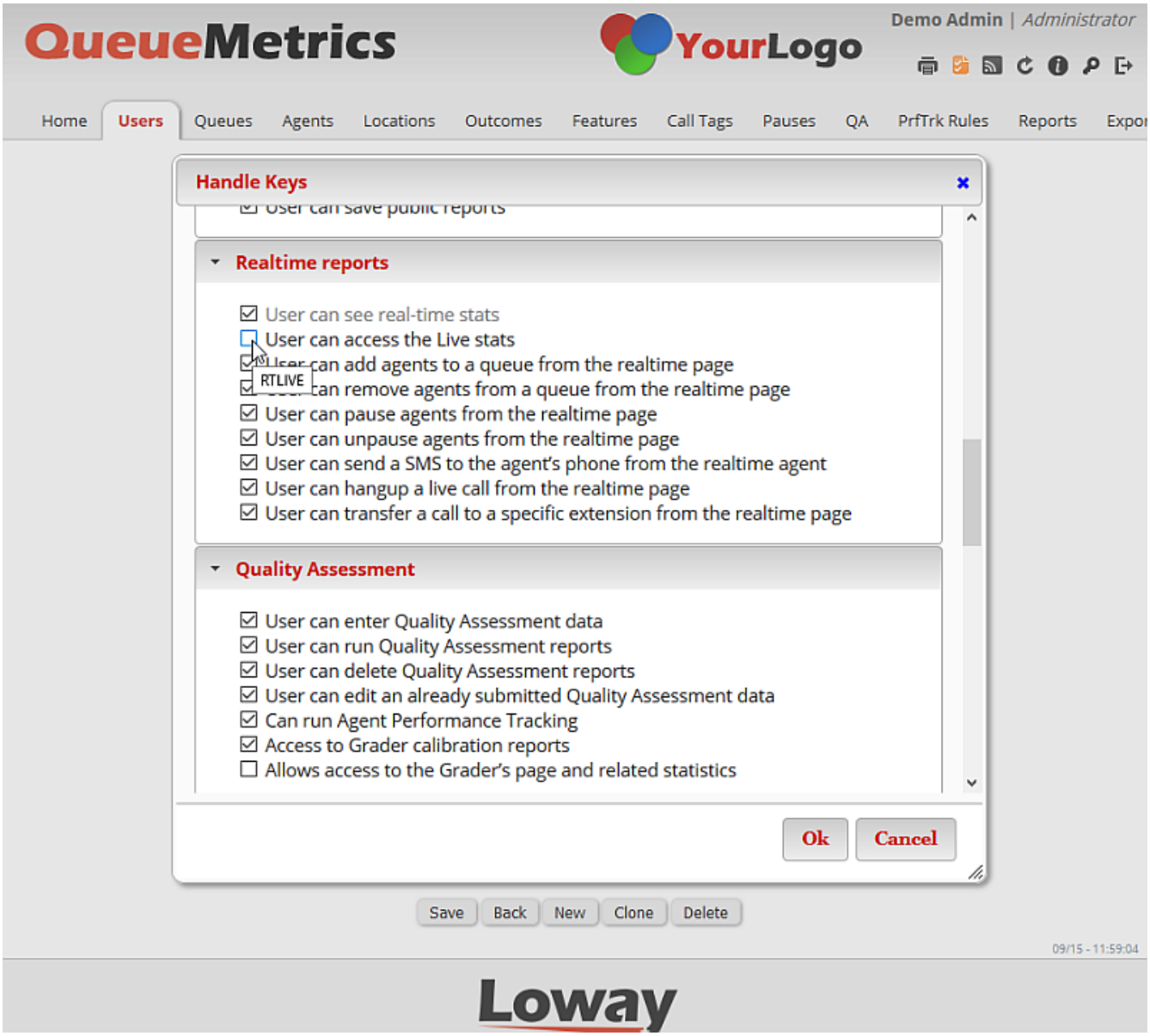
- *Userid* is a technical reference used internally. Read only.
- *Login* is the login string.
- *Password* is the password, shown in clear text.
- *Real name* is the name shown in the top part of the screen
- *Enabled* lets you temporarily disable somebody from using QM.

- *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. The pushbutton *HandleKeys* starts a wizard helping administrators to proper handle keys for users (see below).
- *Number of logons* tells how many times the user logged on in QM. Read only.
- *Comment* is an optional free comment.
- *Token* is used for Single Sign-on.
- *Creation* and *Update*: the user and date/time when the record was first created and then last updated. Read only.



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!

The "wizard's hat" icon located under the User keys field opens a popup where all available pre-defined security keys are listed together with a description. By clicking on each checkbox is possible to assign/unassign the associated key to the user. The wizard helps the administrator to generate a valid keyset by computing key dependencies and/or uncompatibles key pairs and checking or unckecking keys accordingly.



Keys inherited from the class where the user is belonging to are greyed out. Is, moreover, possible to uncheck an inherited security key thus generating the proper negative definition for the edited user.

Flying over a checkbox with the mouse, a tooltip containing the associated security key value is shown.

Pressing *Save* closes the wizard and updates accordingly the *User key* list in the user definition page.

Editing user classes

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



Each class has a set of keys that can be freely edited in much the same way as users by clicking on the "edit" icon (the pencil).



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



The administrator can start a security key wizard by clicking on the button *HandleKeys*. For more information on this wizard, please refer to the previous

> paragraph.

Password Encryption

QueueMetrics allows for user passwords to be encrypted, in order to guarantee greater security.

To manage the process of password encryption, two new system parameters have been added to the system configuration.

- pwd.minAllowedLevel
- pwd.defaultLevel

The first parameter, `pwd.minAllowedLevel`, dictates if the system allows users with non encrypted passwords to access the system. The two different behaviours are associated with two different values for the system parameter:

- **pwd.minAllowedLevel=1** Users with non-encrypted passwords can login onto the system.
- **pwd.minAllowedLevel=2** Users with non-encrypted passwords cannot login onto the system.

The second parameter, `pwd.defaultLevel`, indicates if the new passwords that are set will be encrypted or not. When saving a new password for a user, this property decides if the password is to be encrypted:

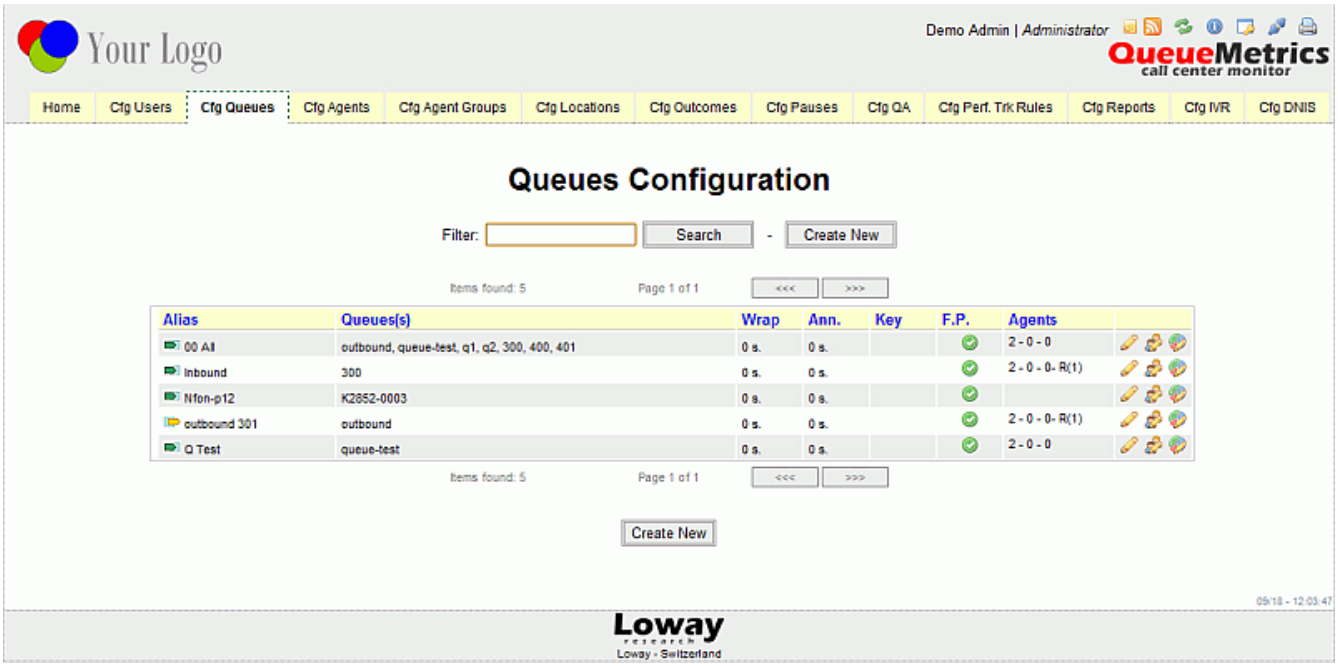
- **pwd.defaultLevel=1** New User's passwords will NOT be encrypted.
- **pwd.defaultLevel=2** New User's passwords will be encrypted.

To encrypt a user's password, the system administrator should first set `pwd.defaultLevel` to a value of 2 then, by accessing the **Edit User** page, a new password for the user must be set. With QueueMetrics 18.04 the user editing page has been modified to include a password confirmation field.

When the new password is set, and the modifications are saved, the new password will be saved as an encrypted password.

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.

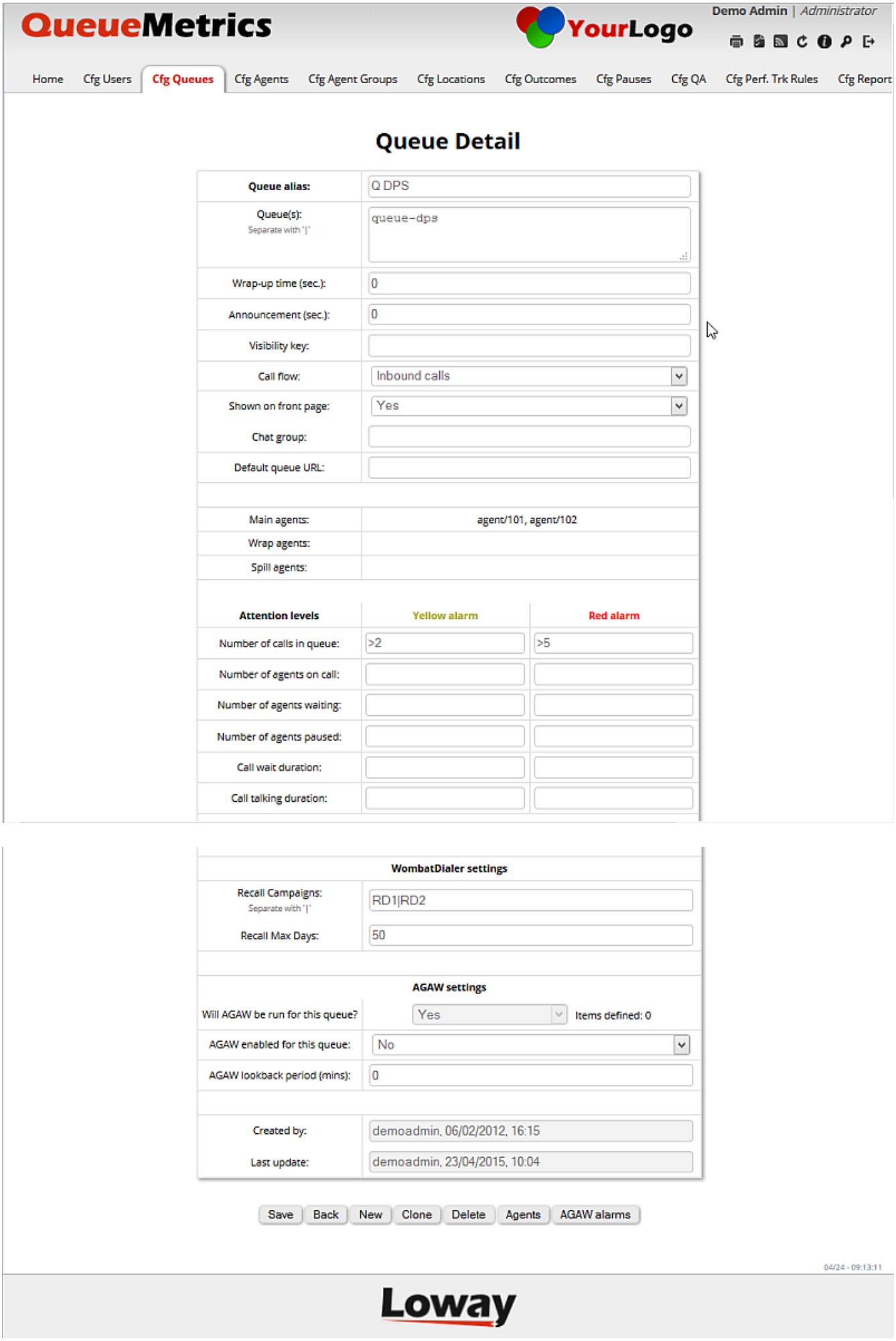


You can search for a specific queue by entering data into the Filter box on top of the page, or change the default sort order for the list by clicking on one of the column names.

The default page shows:

- The queue Alias and Composition
- The Wrap and Announcement durations
- The key protecting the queue, if any
- FP - Front Page: Whether that queue will be visible from the queue selection boxes
- The number of known agents that are member of the queue, by service level (as Main - Wrap - Spill). If the queue is associated to some agents that are not currently defined, for example because you deleted them from the Agent configuration page, an asterisk will be appended to this metric and those agents will appear in red on the queue detail page.

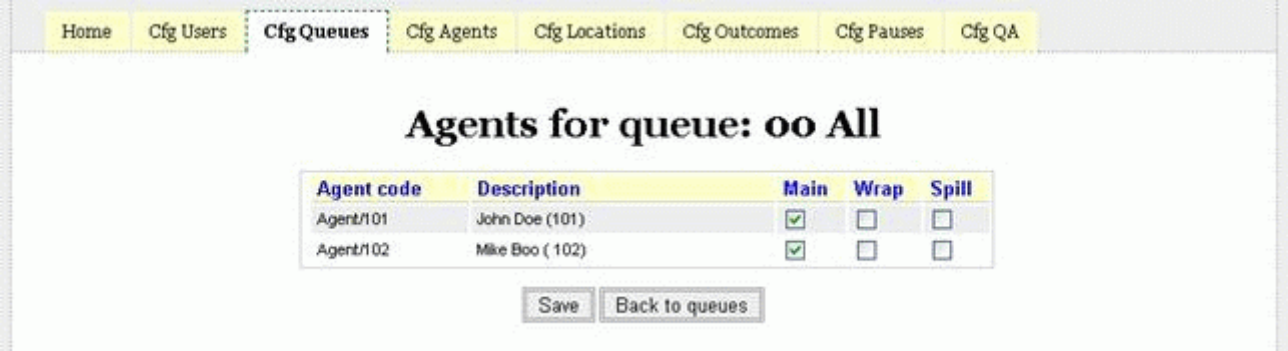
The pencil icon will let you edit the queue, while the "People with pencil" icon lets you change queue associations. The *pie chart* icon at the end allows to define/assign the set of queues on which agents can run reports. No specific key or property are required. For an administrator to make this feature available for the agent, it is necessary to have at least one queue and a report assigned to the agent.


QueueMetrics
YourLogo
Demo Admin | Administrator
Home
Cfg Users
Cfg Queues
Cfg Agents
Cfg Agent Groups
Cfg Locations
Cfg Outcomes
Cfg Pauses
Cfg QA
Cfg Perf. Trk Rules
Cfg Report
Queue Detail
Queue alias:
Q DPS
Queue(s):
Separate with '|'
queue-dps
Wrap-up time (sec.):
0
Announcement (sec.):
0
Visibility key:
Call flow:
Inbound calls
Shown on front page:
Yes
Chat group:
Default queue URL:
Main agents:
agent/101, agent/102
Wrap agents:
Spill agents:
Attention levels
Yellow alarm
Red alarm
Number of calls in queue:
>2
>5
Number of agents on call:
Number of agents waiting:
Number of agents paused:
Call wait duration:
Call talking duration:
WombatDialer settings
Recall Campaigns:
Separate with '|'
RD1|RD2
Recall Max Days:
50
AGAW settings
Will AGAW be run for this queue?
Yes
Items defined: 0
AGAW enabled for this queue:
No
AGAW lookback period (mins):
0
Created by:
demoadmin, 06/02/2012, 16:15
Last update:
demoadmin, 23/04/2015, 10:04
Save
Back
New
Clone
Delete
Agents
AGAW alarms
04/24 - 09:13:11
Loway

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. You can also use wildcards - see below. Queue names can contain spaces, though we suggest using simple strings.
- 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). Select "any call" when you aggregate queues with different call flow settings.
- If it's *Shown on front page*, that is, in the main Queue selector combo box (if not, the queue is said to be *invisible*).
- A *Chat group*, that is, the XMPP address of a queue manager
- A *Default URL* to be used on the Agent's page if no URL is passed in the queue log (see below) in order to open a third-party application.
- A *Service Level Agreement* (in seconds), that is the point in time where you want SLA measured for reports using this metric, e.g. DD09, DH09, DW09
- *External reference*: An identifier for the queue that is used at the PBX level and is opaque to QueueMetrics. Used by some interaction or autoconfiguration modules.
- *Update protection*: If this option is set to "Yes", the queue cannot be updated by the "Configuration Synchronizer". Default is set to yes.
- *Delete protection*: If this option is set to "Yes", the queue cannot be deleted by the "Configuration Synchronizer". Default is set to yes.
- A number of *Attention levels*, see below.
- The current known *Service groups* for that queue, i.e. which agents are linked to that queue
- The current *WombatDialer campaings* for the queue (see below)
- The current *AGAW settings* for the queue (see below)
- The current *AMO settings* for the queue (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.



If, usually because of a misconfiguration, a queue has a known member that does not exist within the agent list, then the queue name is shown in red in the queues list and the code for such agent(s) is shown in red on the queue details. This may happen, for example, if you create an agent, add it to a queue as a member, and then go back and delete such agent.

The Queue URL

The rules followed by QueueMetrics when displaying a link icon (and linking to the associated URL for your call) are as follows.

On report pages:

- If the reporting queue is not a composed queue and the queue has an associated URL (coming from the queue definition page), then this URL is shown.
- If the reporting queue is a composed queue and/or no URL is associated to that queue in the queue definition page, then:
  - If the call has an associated URL in the queue_log, this will be shown
  - If the call has not an associated URL in the queue_log then the URL specified in the configuration properties key *default.crmapp* will be shown. If no *default.crmapp* URL has been specified in the configuration properties file, the icon will not be shown.

On the live agent page:

- If the queue where the call comes has an associated URL in the queue configuration page then this URL is shown for each call
- If the queue where the call comes has not an associated URL in the queue configuration page then:
  - If the call has an associatd URL in the queue_log, this will be shown
  - If the call has not an associated URL in the queue_log then the URL specified in the configuration properties key *default.crmapp* will be shown. If no *default.crmapp* URL has been specified in the configuration properties file, the icon will not be shown.

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

Using wildcards in queue names

QueueMetrics allows a limited use of wildcards to group together queue names. Wildcards work by matching the composition of known single queues, so if a queue is not defined in QueueMetrics (even if it is present on Asterisk) it will NOT be matched by a wildcard of "*".

On the other side, a hidden single queue will be matched by a non-hidden front-page queue whose definition is "*".

With wildcard matching:

- The Asterisk * stands for any number of characters. E.g., "open*" as the queue composition will match any atomic queue starting with "open". Just entering "*" as a queue composition will match any atomic queue on the system.
- The question mark ? stands for a single character - e.g. "open?" will stand for "open1" and "openq", but not for "open99". You can group together a number of question marks to match multi-character sequences of known length, e.g. "open??" will match "open99".
- Any string starting with a caret ^ will be interpreted as a case-insensitive regular expression that can be used to create very complex rules. For example, if you want all queues that do not

start witn an underscore, you could use `^(?!_).+`.

AMO Settings

To create an AMO campaign:

- Make sure that the queue is set for Outbound
- Set "AMO enabled" to Yes

Once a campaign is set for AMO, it is visible in AMO Campaigns.

You can then edit the details of how the campaign is supposed to behave.

- **Max attempts** is the maximum number of times a number is recycled in front of negative outcomes.
- **Seconds before recall** is the minimum time that will be interleavled between further attempts. As calls are "pulled" by agents, you have no guarantee that this time will be respected - that is, this is a minimum value, but the actual value may be way higher. Measured since the the reschedule decision is taken.
- **Recall stepping** might be Fixed, Linear or Exponential. In fixed mode, recalls are always spaced by the same distance. In linear mode, recalls are multiplied by the number of recalls it is being attempted - that is, the first time the recall distance is applied, then two times, then three times, and so on. In exponential mode, the distance is multiplied by 2 raised to the number of attempt - so the first time it's the basic distance, the second time two times that, then four times that, then eight, and so on.
- **Valid outcomes** are the set of call outcomes that will be considered signs of a successful call. "*" means any outcome, "NONE" means no outcome set, or you can enter a set of coutcome codes separated by space.
- **Recall attributes** are the attributes that will be copied from the number into the `queue_log`, as QueueMetrics variables. Separate with space.



To understand how AMO works, see Assisted Manual Outbound.

Configuring redial services through WombatDialer

WombatDialer is highly scalable, multi-server, easy-to-use telecasting platform produced by Loway. Queues in QueueMetrics can be associated to dial campaigns in WombatDialer thus allowing easy implementation of recall services. By specifying a set of pipe separated campaings and a maximum recall day period, an agent can schedule a recall by clicking on an icon on the call details in their own agent's page. For further clarifications, see section WombatDialer Recalls

Configuring queues to be processed by the AGAW Runner

The AGAW Runner will use the following rules to decide which queues and agents to process:

- All queues that are "simple", i.e. not composite, are taken into consideration for processing by the runner. All queues that have the AGAW runner enabled will be actually processed (you must

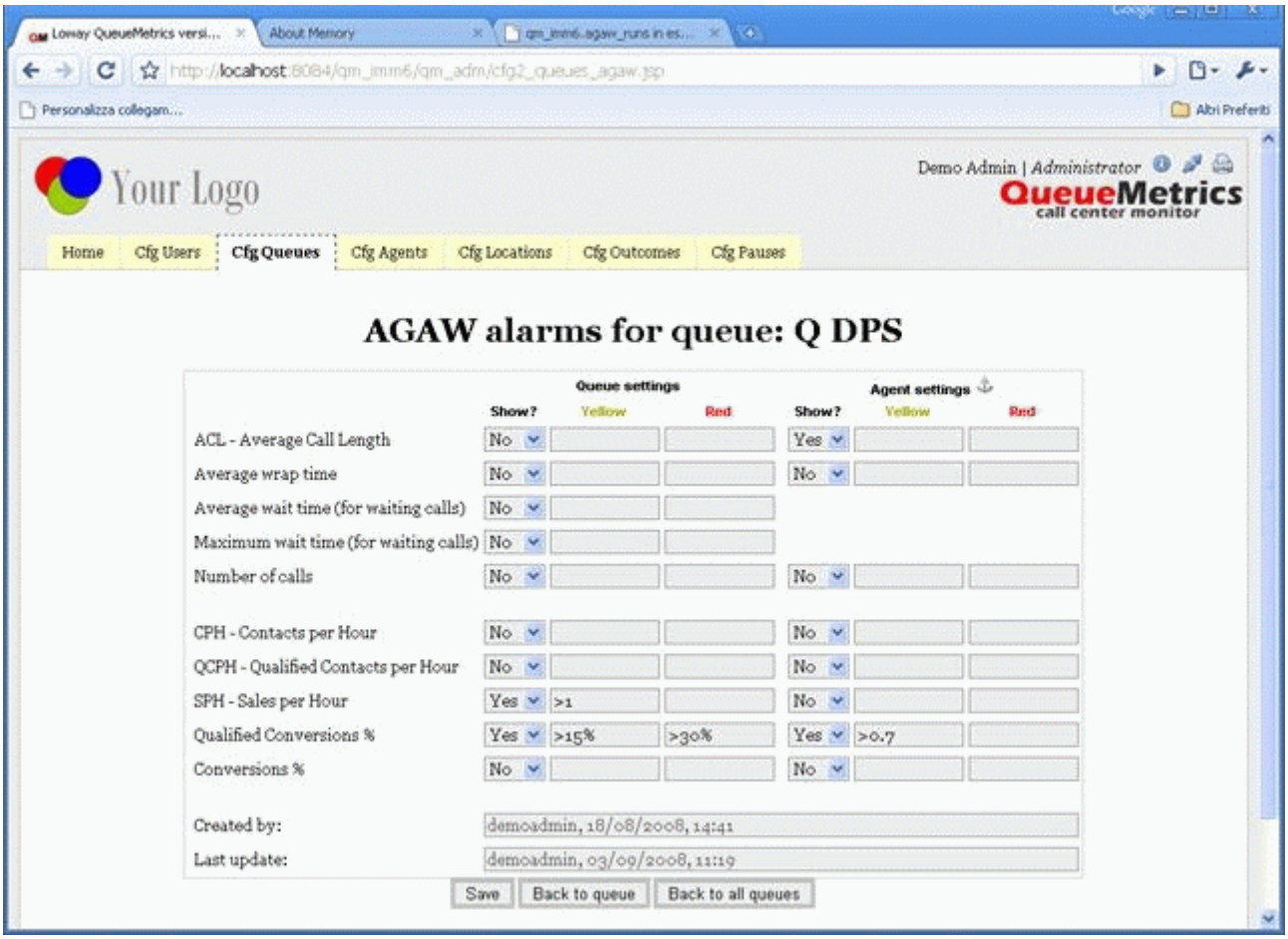
enable that manually in the queue config page).

- All agents that are linked to a queue are processed for that queue, even if there is no data for them; plus, any unknown agent that is detected working on a queue is processed for that queue..

In order for a queue to be processed by the runner, and show visible metrics to the user:

- The *"Will AGAW be run..."* must be set to "yes"
- The *Items defined* must be > 0 (or the agent will see no metrics)
- The *AGAW enabled* should be "Yes"
- The *AGAW look back* period can be left blank (default). This is the size of the look-back period the AGAW runner uses.

The set of metrics that is enabled and their alarms is defined in the AGAW queue configuration screen (click on the "AGAW alarms" button):



As you can see, for each metric there are a couple of switches that decide:

- Whether that metrics is to be shown at all
- Red and yellow alarm levels for the whole queue (to be shown in AGAW, so they might differ from the ones used for the real-time page)
- Red and yellow alarm levels for each agent separately

These settings are applied from the next run of the AGAW runner, so they can be modified while

the AGAW runner is active and will be picked up when the relevant queue is processed.

All values are always computed; you can toggle visibility of values on and off (if they are "Off" they are visible in the AGAW monitor but NOT to the user).

Alarms can be expressed as:

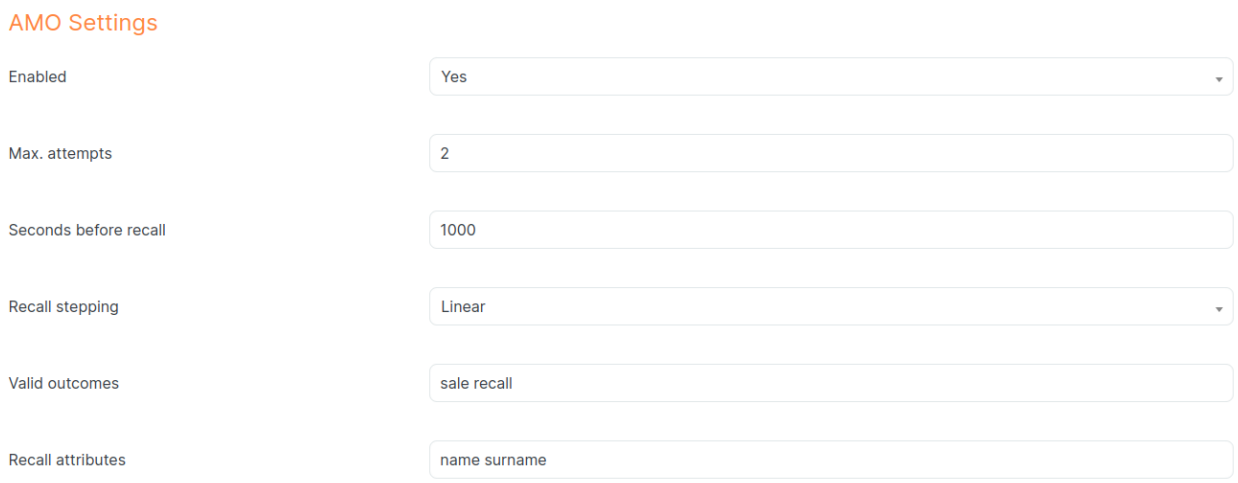
- Integers (for time periods and n. calls)
- Floating-point values (e.g. > 3.7)
- Percentages (e.g. > 10%).

While percentages for the Queue part translate to the corresponding ratio (e.g. 10% means 0.1) in the Agent part they are anchored to the Queue metrics - that is if number of calls is at a given moment 1000 and there is an agent alarm at "< 1%" , if the number of calls taken by that agent are less than 1% of the queue the alarm will be triggered.

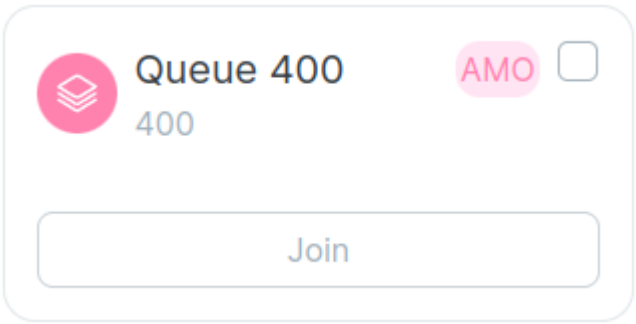
If you need to express a fixed percentage in the Agent part, use the corresponding ratio, as in the Qualified Conversion example above.

Configuring AMO campaigns

To enable AMO (Assisted Manual Outbound) calls on a queue, you will need to open the Queue Settings page of a given Queue, and set the "Enabled" flag to "Yes" in the "AMO Settings" section.



This will mark the queue as an AMO queue, which will reflect in the Agent Page by showing the AMO tag on the selected queue.



Other important setting you can configure here are:

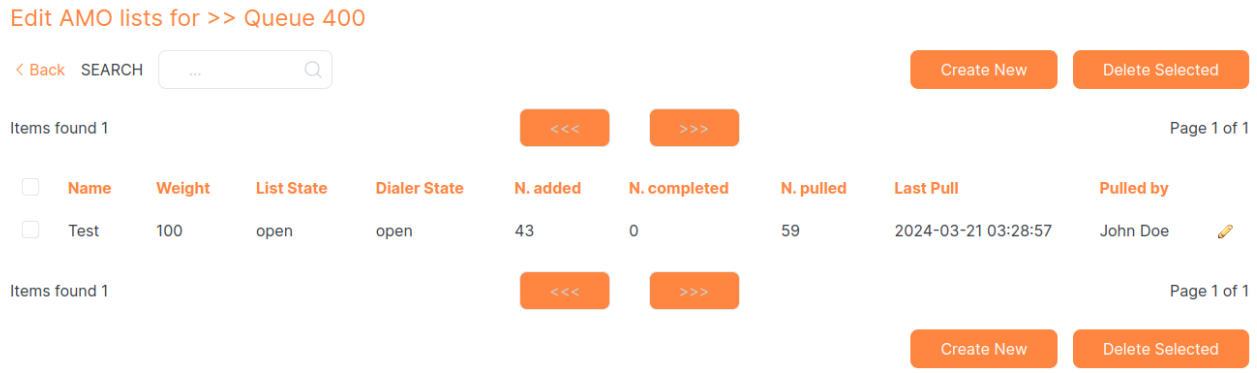
- Max. Attempts: The maximum numbers of times an AMO number can be retried before being discarded.
- Seconds before recall: The number of seconds that needs to pass between a failed AMO call and its re-shuffling in the AMO queue.
- Recall stepping: The type of Recall Stepping algorithm to use.
- Valid outcomes: The call outcomes that are counted as valid towards marking an AMO call as completed.
- Recall attributes: The attributes for the AMO call that QueueMetrics will keep track of. These should match the attributes associated to a number in an AMO list.

Once the AMO queue is configured correctly, you can move to the AMO editor, where you should immediately see the Queues that are AMO enabled, listed as AMO Campaigns (Each queue counts as a single AMO Campaign).

Clicking on the Queue name will navigate you back to the Queue Setting for the Queue associated with that AMO campaign.

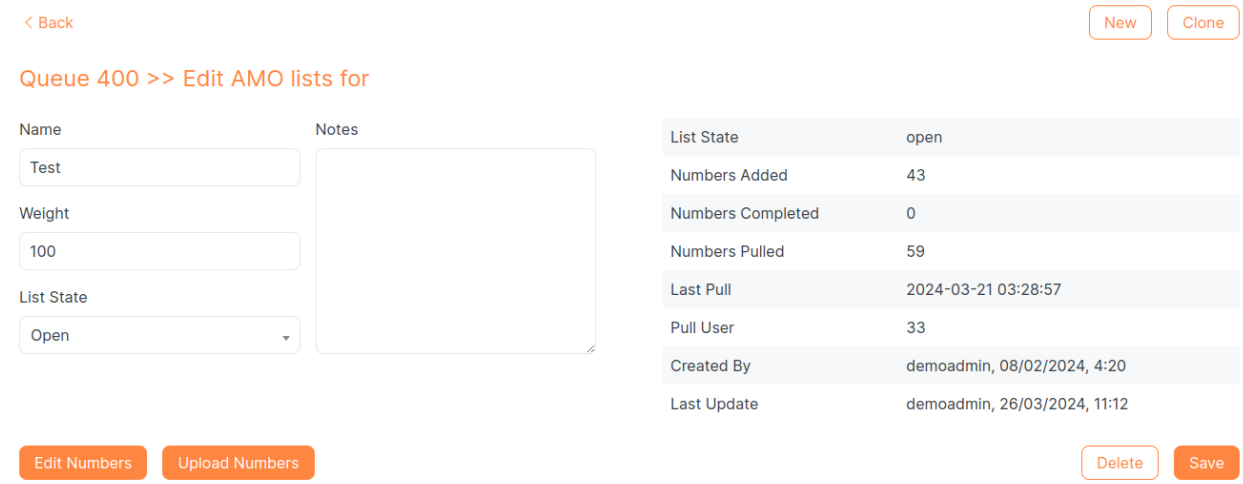
Clicking on the pencil icon will navigate you to the List editor for that specific AMO campaign, where you can work on the list of numbers linked to the AMO Campaign in question.

Configuring AMO lists



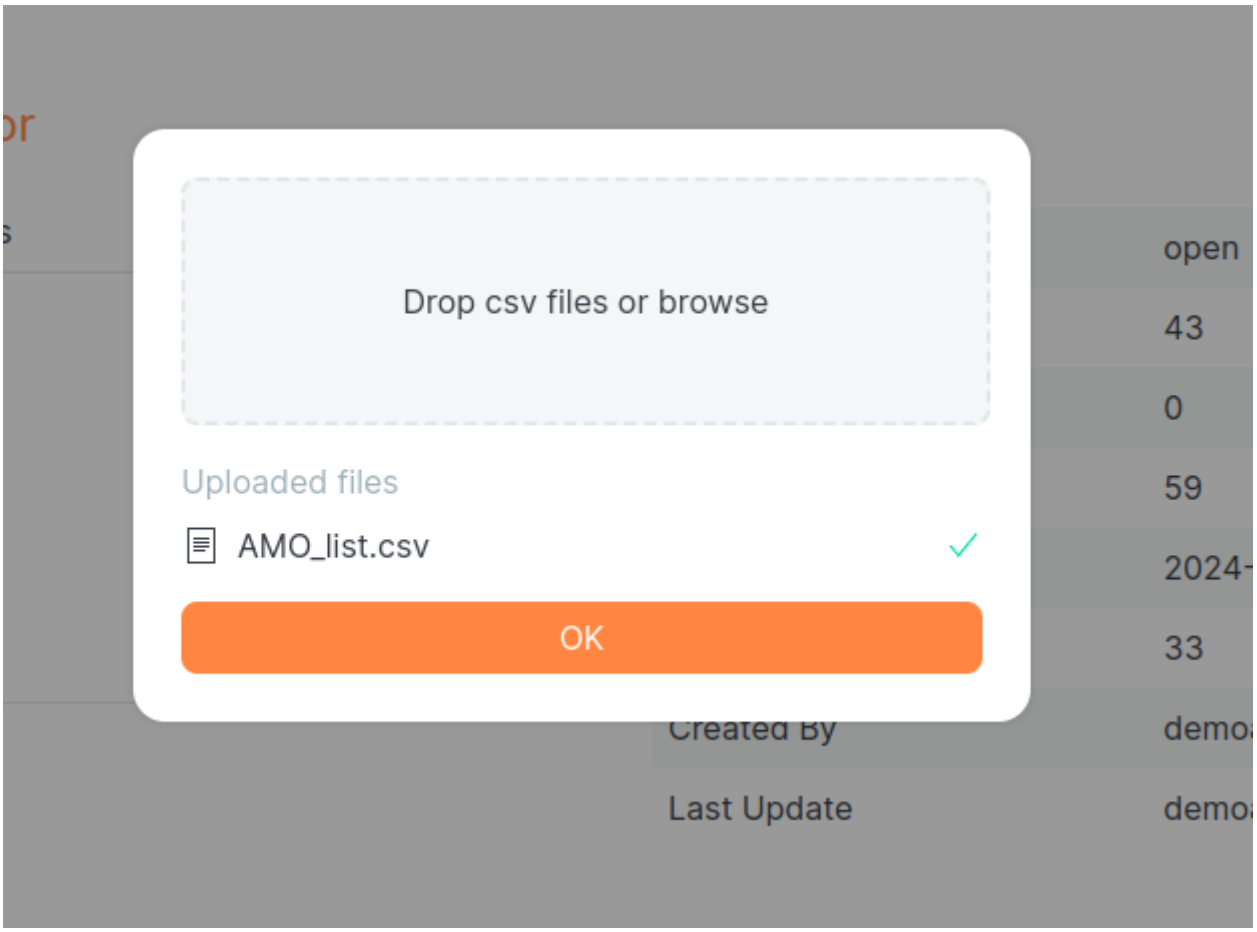
Here, you will see a list of AMO lists that are related to the currently selected AMO Campaign. You can always navigate back to the AMO campaigns list by clicking on the "< Back" button in the top-left of the screen.

Clicking on the pencil icon will navigate you to the editor for that specific AMO list. Here, you can edit the list itself, see its usage statistics, and upload numbers by uploading a CSV file.



Configuring AMO numbers

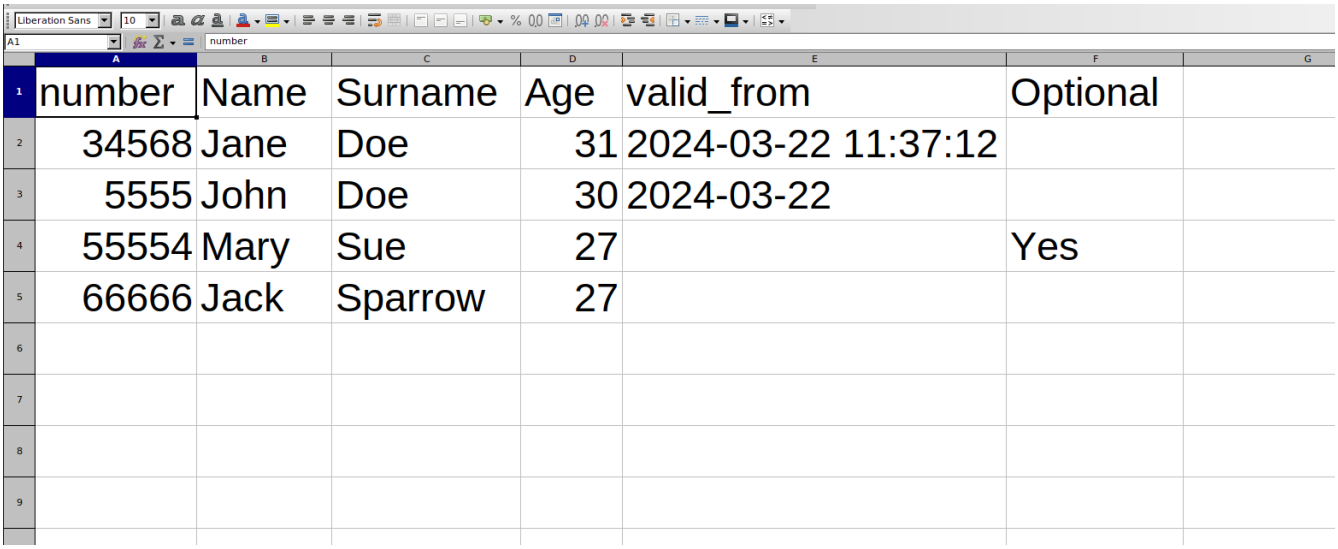
The most convenient way to upload numbers into an AMO list, is to use the "Upload Numbers" button in the List editor. This will open a modal that allows you to drag and drop or click to upload a .csv file containing the numbers in question.



By clicking "OK", the file will be processed and uploaded to QueueMetrics, adding the numbers to the current list.

The file MUST be a .csv file, and it needs to contain:

- A "number" attribute.
- At least one record.
- Any number of arbitrary attribute/key pair (Remember to also write them in the Queue AMO Settings, under "Recall Attributes".)
- The valid_from attribute (Optional) in the format "AAAA-mm-gg HH.mm.ss" or "AAAA-mm-gg". If the hour is missing, it will be automatically set to 00:00:00. If the whole day is missing, the number will be immediately marked as valid for shuffling in the AMO Campaign.



number	Name	Surname	Age	valid_from	Optional
34568	Jane	Doe	31	2024-03-22 11:37:12	
5555	John	Doe	30	2024-03-22	
55554	Mary	Sue	27		Yes
66666	Jack	Sparrow	27		

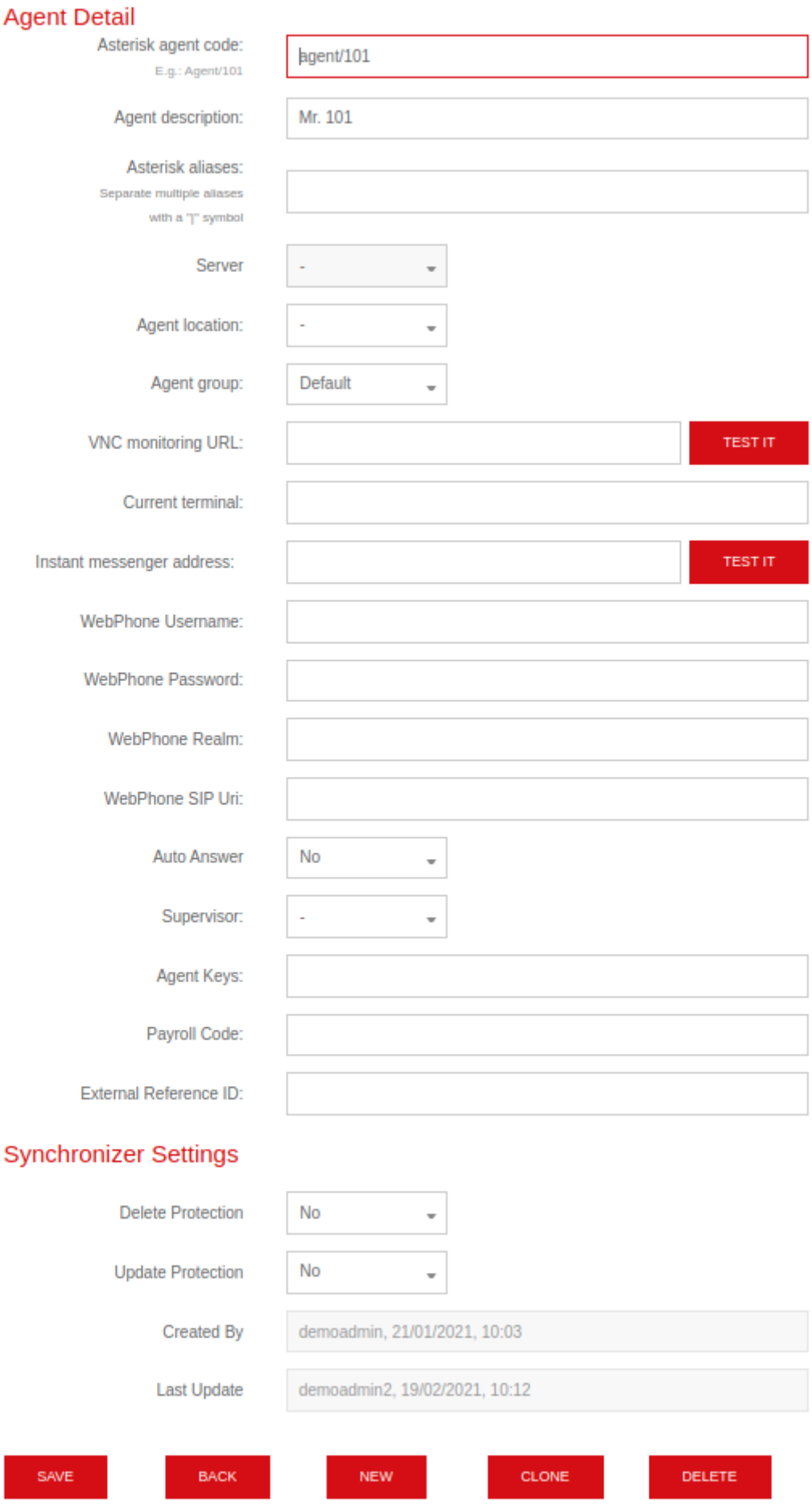
Configuring agents

Agents should be configured so that they:

- are decoded to their own name when they are found in reports
- can be set as members of service levels for queues.
- can be assigned an optional Location, that can also be used as a filter condition.
- can be assigned an optional User Group, that can also be used as a filter condition.
- can be assigned to a Supervisor
- can have a VNC URL defined
- can have a Current Teminal defined.



When editing an agent, the following screenshot appears:



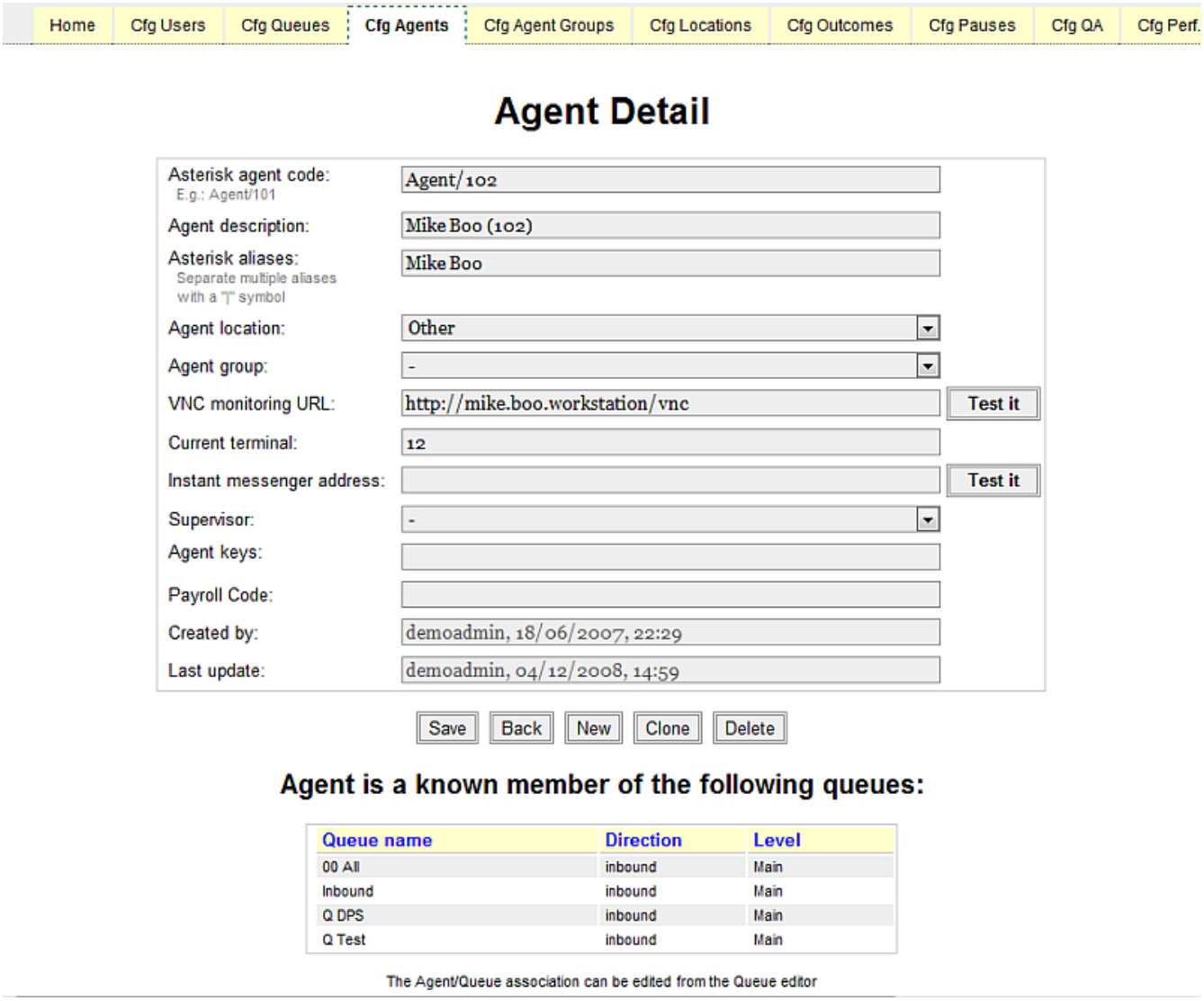
For each agent in use, enter:

- *Agent code* as the Asterisk agent code, e.g. Agent/101;
- *Agent description* as the agent's own name.
- *Asterisk aliases* is a list of aliases that the agent appears on the `queue_log` as. See below.
- *Default server*: for cluster mode, the default server for the agent. This information is used to pre-populate the related field in the realtime agent page logon panel. Leave empty if the agent is not associated to a specific server.
- *Agent location* and *Agent group* can be selected from a drop-down list of defined locations. Leave blank if not needed.
- *VNC Monitoring URL:* the URL that will launch the VNC monitoring app for the given agent
- *Agent Keys*: A user must have these keys in order to perform actions on or with the agent, but this does not affect their visibility.
- *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. This field is used to pre-populate the "Current extension" field in the agent real-time logon panel.
- *Instant messenger address*: an XMPP address associated to the agent. Used in the real-time page for supervisors to initiate a chat with the agent.
- *WebPhone Username*: the SIP username used by the softphone in the realtime agent page to authenticate the agent
- *WebPhone Password*: the SIP password used by the sofphone in the realtime agent page to authenticate the agent
- *WebPhone Realm*: The authentication SIP Realm used by the softphone in the realtime agent page to authenticate the agent
- *WebPhone SIP Uri*: An optional SIP URI used by the softphone. If blank, the username@sipserver is used
- *Auto Answer*: If set to yes, the agent's Softphone (in the agent page) will automatically answer incoming calls, if no other calls are currently active.
- *Supervisor*: the supervisor for this agent. This can be selected between all users holding the key SUPERVISOR.
- *External reference*: An identifier for the agent that is used at the PBX level and is opaque to QueueMetrics. Used by some interaction or autoconfiguration modules.
- *Update protection*: If this option is set to "Yes", the agent cannot be updated by the "Configuration Synchronizer". Default is set to yes.
- *Delete protection*: If this option is set to "Yes", the agent cannot be deleted by the "Configuration Synchronizer". Default is set to yes.

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

On the bottom of the page, the current association of that agent to a set of queues is shown.

The Agent Detail page is accessed by selecting the "Edit agents" link on the QueueMetrics Home page and then clicking on the "Edit" icon for a specific agent.



Asterisk aliases

It is possible to add one or more "friendly names" for agents within the "Asterisk aliases" field, which can be found in the Agent Detail page. Such names are alternate ways that the same agent is logged on the queue_log.

This may be useful if the code for the same agent was changed (e.g. up to a certain date you used two digits for agents, and at some point you made them tree digits, so the same person who had code "20" now appears as "120" - but it is always the same person). Another common occurrence is FreePBX logging the agent's full name instead of their code.

You may have multiple aliases, all separated by the pipe symbol. If an alias ends with a space, you have to enter a trailing pipe symbol too (e.g. `"john doe |"`). Note that when an unknown alias that has trailing spaces appears on reports, it is printed with dots instead of spaces (e.g. "`agent/john·doe·`"), so that you can see those spaces are there.

When you enter an alias that does not start with "agent", QueueMetrics will automatically add a second alias starting with "agent", so that when you define e.g. alias "120" you do not need to

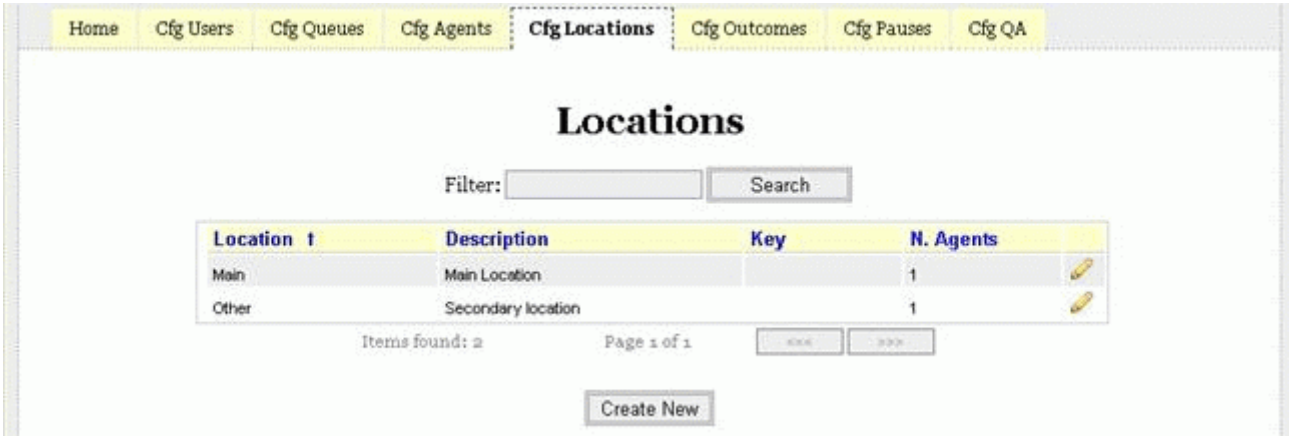
explicitly define "agent/120" too (but the opposite does not apply!).



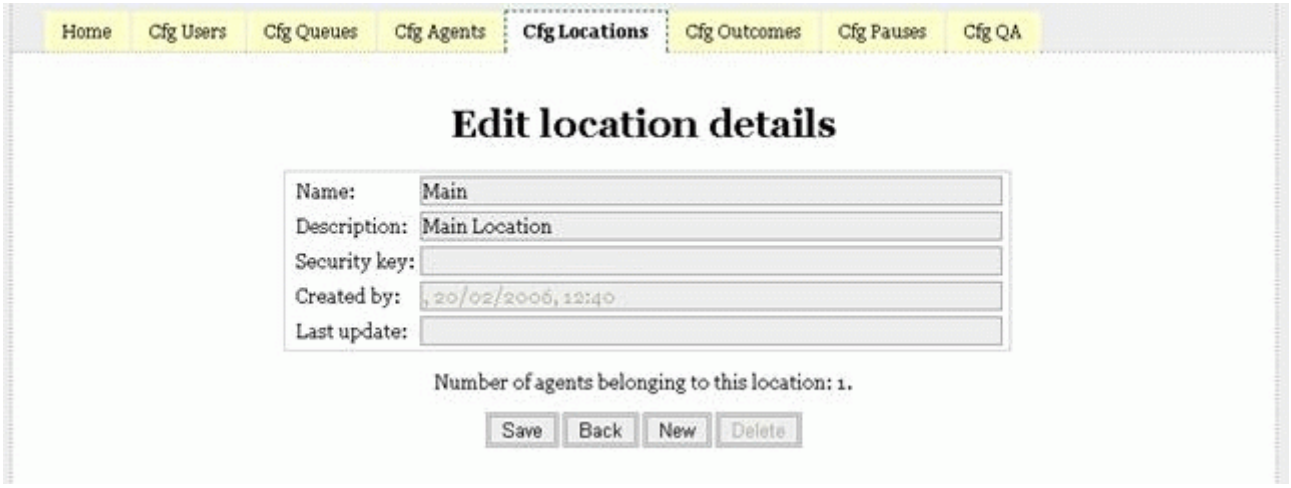
If you use the hot-desking mode, you must make sure that **no extension used for hotdesking is ever set as an agent alias**; otherwise the agent will be logged on corrrectly, but it will be impossible to pause or log them off (as the extension will not be found, and shown as empty). You will notice such cases because, on the Real-time page, the agent who logged on with hot-desking will have a blank Extension field. If this happens, to fix the issue, just remove the alias.

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.



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.



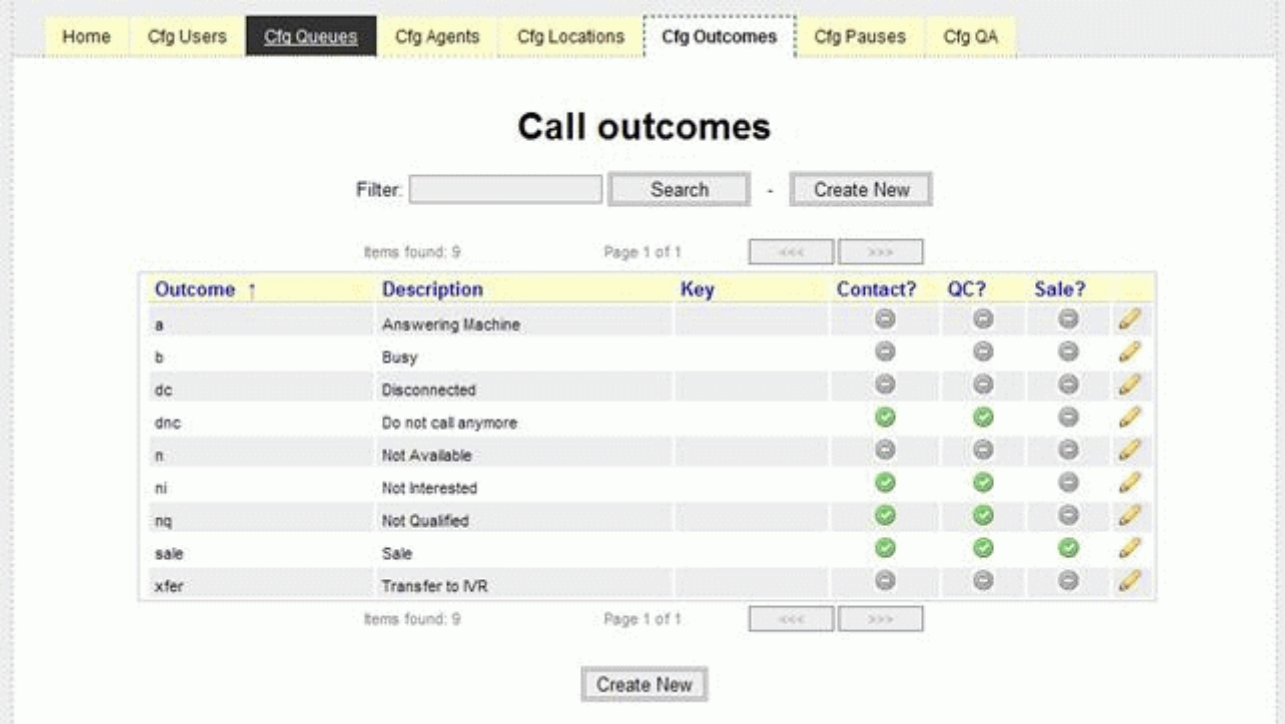
A location cannot be deleted if at least one agent is defined for that location.

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.

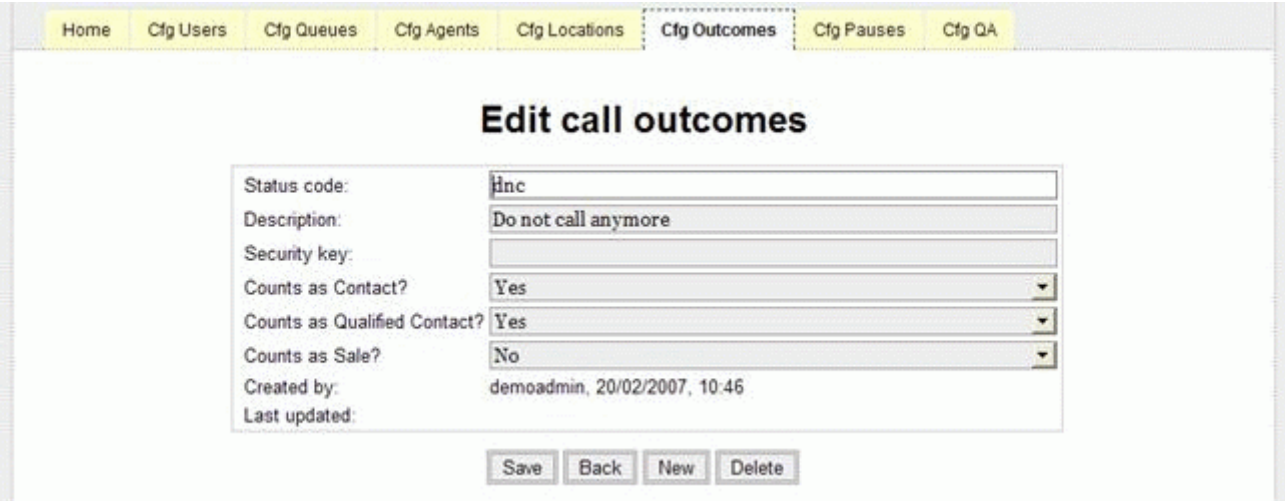
The call outcome will be defined by a numeric sequence that the agent will either key in on their telephone terminal or report through QueueMetrics itself through the Agent's page. QueueMetrics will not consider how the sequence is entered, as long as it's present in the *queue_log* data it runs on. 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.



As you can see, each outcome can set two flags: a "This call qualifies as contact?", "The call qualifies as Qualified Contact" and a "This call qualifies as sale?" flag. This will be used in order to produce statistics on traffic (see section How are Call Outcomes calculated?).

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.



The editor page lets you set:

- A numeric code for that outcome. The system will check that it will not be duplicated on the list (The code *should* be numeric so it may optionally be keyed-in using the rep's terminal)
- 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.
- A *Queue visibility* list - you can list a set of atomic queues for which this code will be displayed. Queues are separated by whitespaces. The field accepts wildcards to match multiple queues. If the field is left blank, then the code is displayed for all queues.

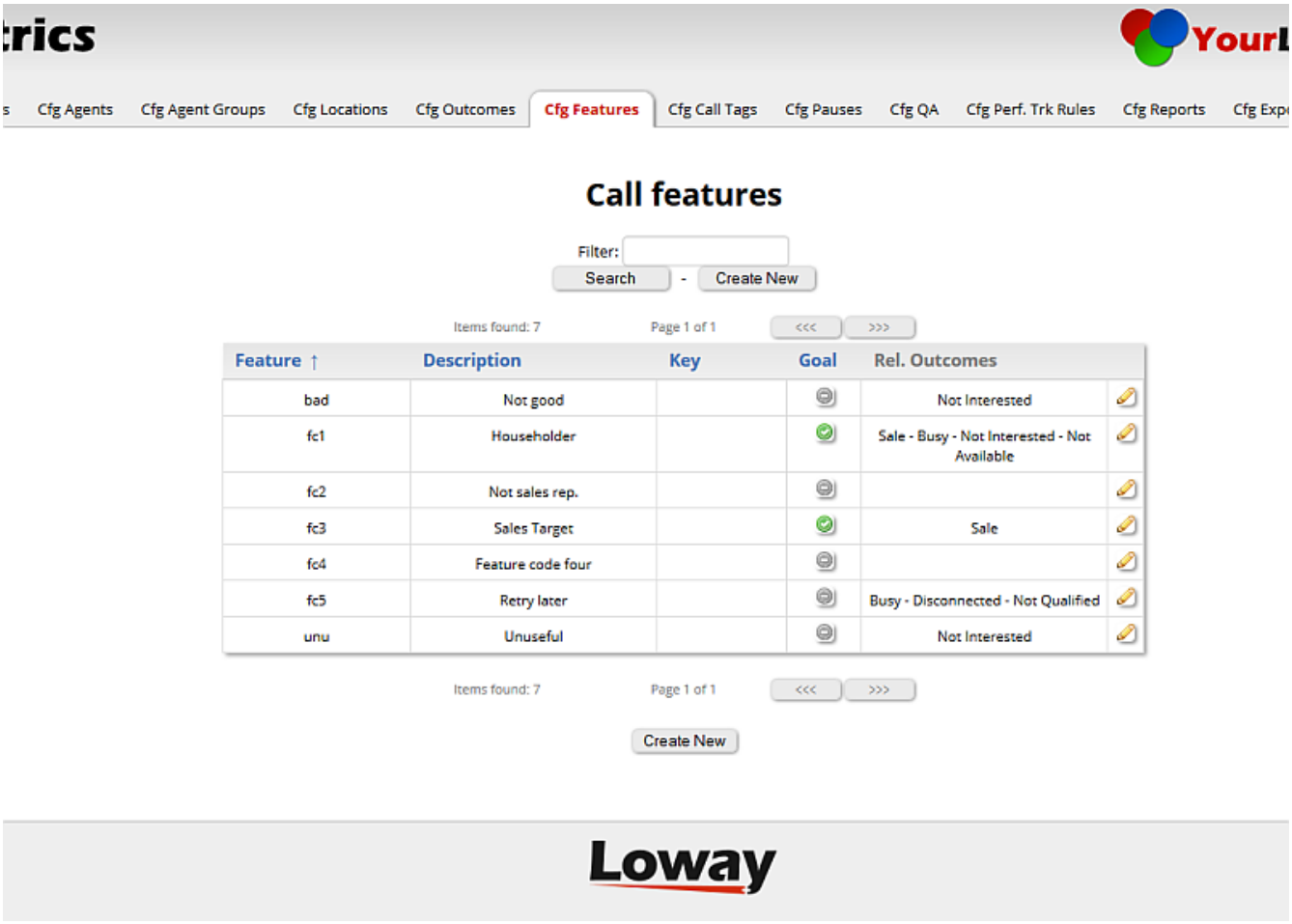


Outcome status codes MUST be defined only with numbers and lowercase letters. The following characters are also allowed: _-.@/()=?^ . QueueMetrics will automatically delete wrong characters and change uppercase characters to their lowercase equivalent.

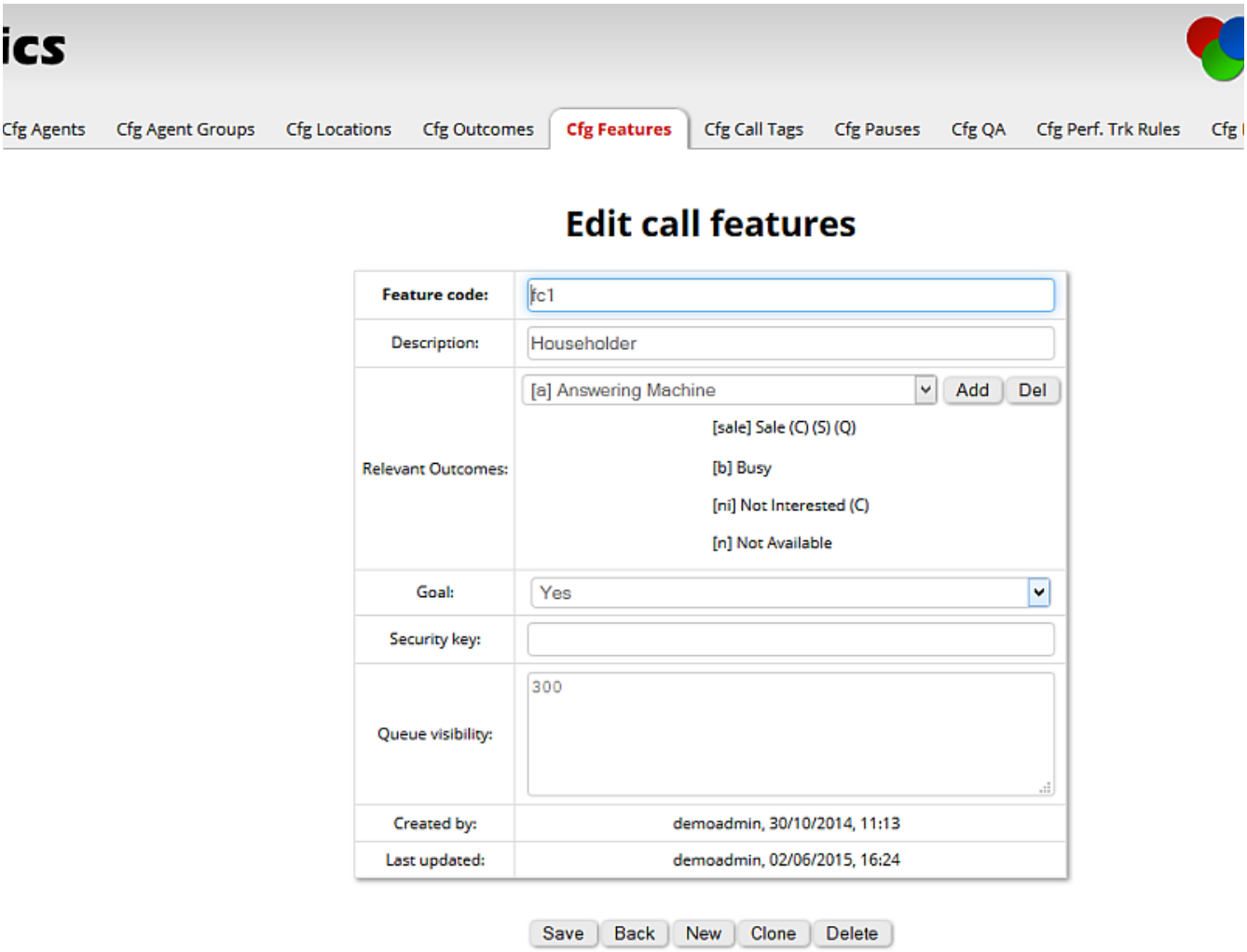
Configuring Feature Codes

For call centers that require to analyze agent performances at a level below call outcome, the concept of *feature codes* has been added. A user can think about feature code like a sub-outcomes. They can, optionally, have hierarchical relationships with the primary outcome code. A simple example of the feature code concept is the pair of outcome/feature like "Not Interested"/"costs too much". A call may have an unlimited number of features associated with. Some features may be flagged as *goal*. Goal features are counted separately and generate separated statistical figures in reports.

Editing feature codes requires holding the key USR_OUTCOME.



By clicking on *Create New* button or on the pencil icon present on each feature code row in the list, a user can create or edit an already present feature code.



As told before, tag features can be linked with a specific outcome code set. To link a feature code with a specific outcome or set of outcomes, a user should select the outcome from the dropdown, then click on the *Add* button. The list of already associated outcomes are shown below the dropdown. To remove an outcome association, a user should select the proper outcome from the dropdown, then click on the *Del* button.

Clicking on *Save* updates the feature code on QueueMetrics database.

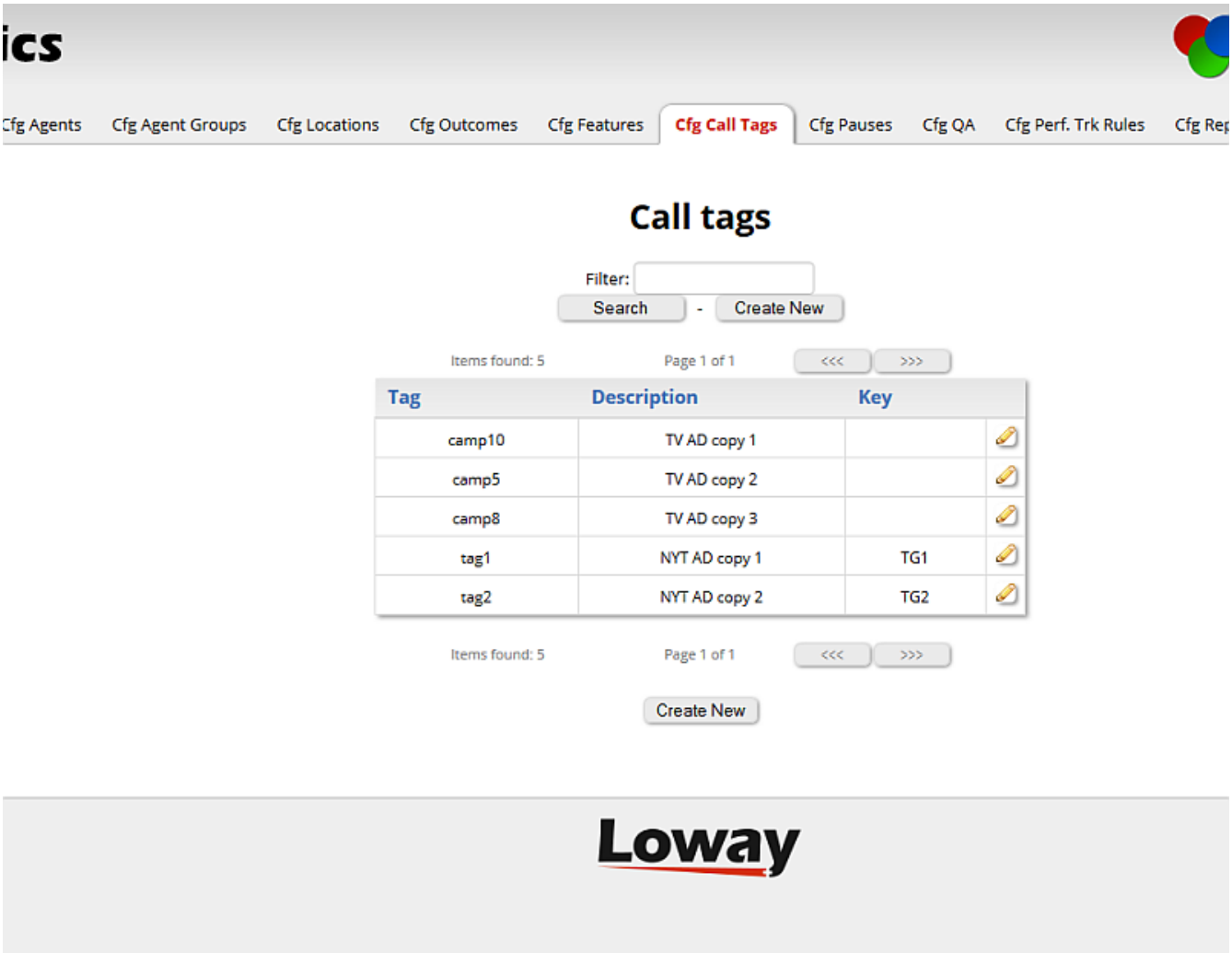


Feature status codes MUST be defined only with numbers and lowercase letters. The following characters are also allowed: _-.@/()=?^ . QueueMetrics will automatically delete wrong characters and change uppercase characters to their lowercase equivalent.

Configuring Tag Codes

Some call centers require to report call set at the list level below campaign. This is frequently used for outbound campaings. The idea is to add the concept of generic *tag* to calls, so that, if assigned to each calls, it could be used to create a new level of reporting, parallel to the queue. When each call

is associated to a single tag, the queue would be the *physical* medium for handling calls, while the tag could bee freely set. Imagine, for example, to set up a special number for a Tv Ad, or a printed advertising, that makes it possible to discriminate where the caller saw the Ad. Tags are set by the system (i.e. there is no way for QueueMetrics to add a tag on a specific call but is up to the PBX/dialplan to add a tag signature in the queue log); for more information on that, please refer to this paragraph. Tags associated to a call will be shown by QueueMetrics and decoded following the tag list defined by the administration page, as reported below.



By clicking on *Create New* button or on the pencil icon present on each tag code row in the list, a user can create or edit an already present tag description.

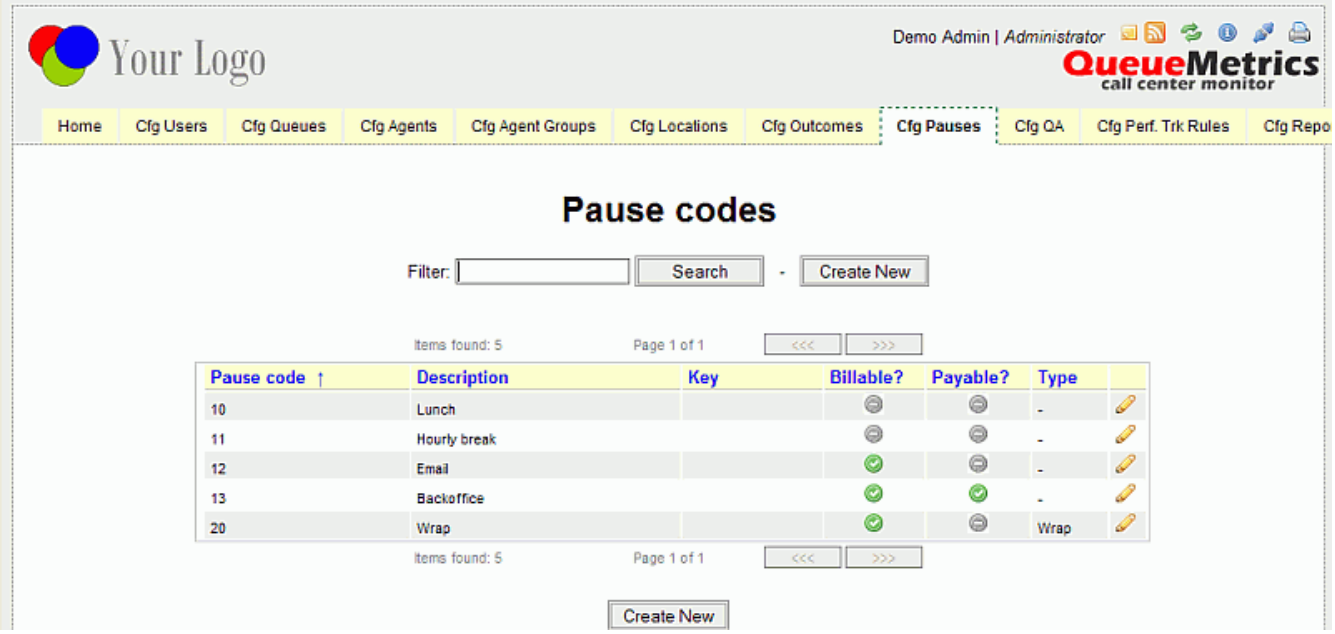
Tags definition/listing is available only for users holding the USR_CALLTAG security key.

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 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.

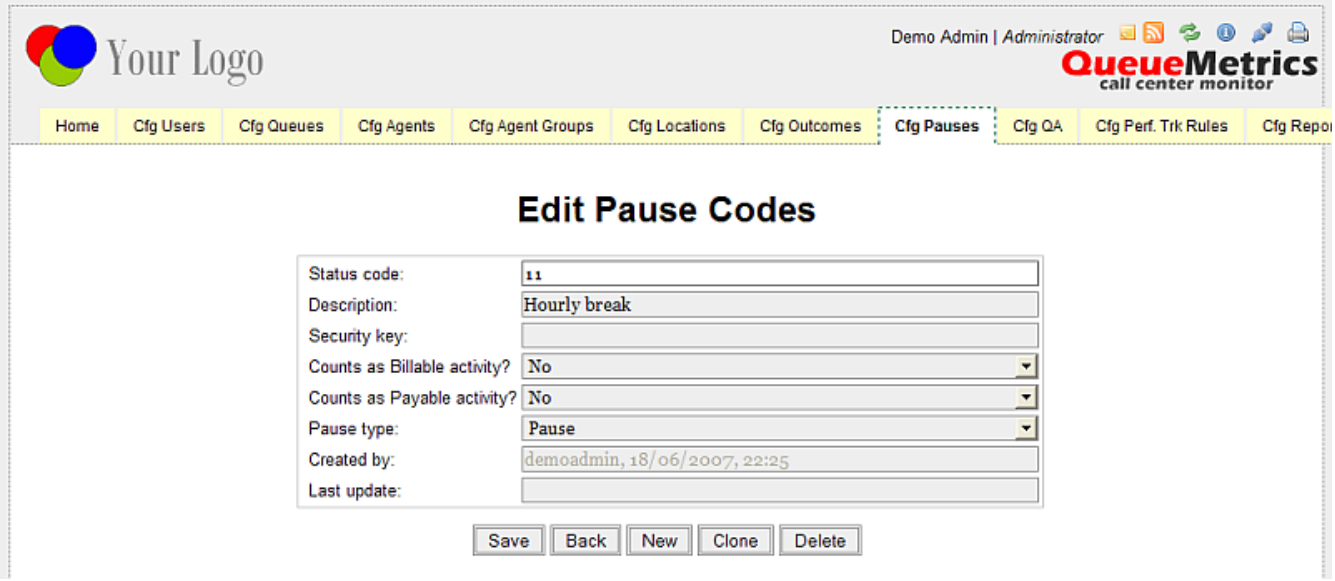


For each pause code, it is possible to tell QueueMetrics whether that time is:

- billable or non-billable - whether the pause will be counted for client billing
- payable or non-payable - whether the pause will be counted for payroll generation

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

- A numeric code for that activity. The system will check that it will not be duplicated on the list (The code *should* be numeric so it may optionally be keyed-in using the rep's terminal. This is not a technical requirement anyway)
- A text label for the activity (e.g. "Lunch")
- A flag telling the system whether that activity is: Billable or Not Billable
- A flag telling the system whether that activity is: Payable or Not payable
- A flag showing the type of pause. A pause can be a standard pause, or a pause of the ACD made to produce outbound calls, or a call wrap-up pause (this is partially implemented only in the AGAW sub system) or an administration type pause.
- 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.



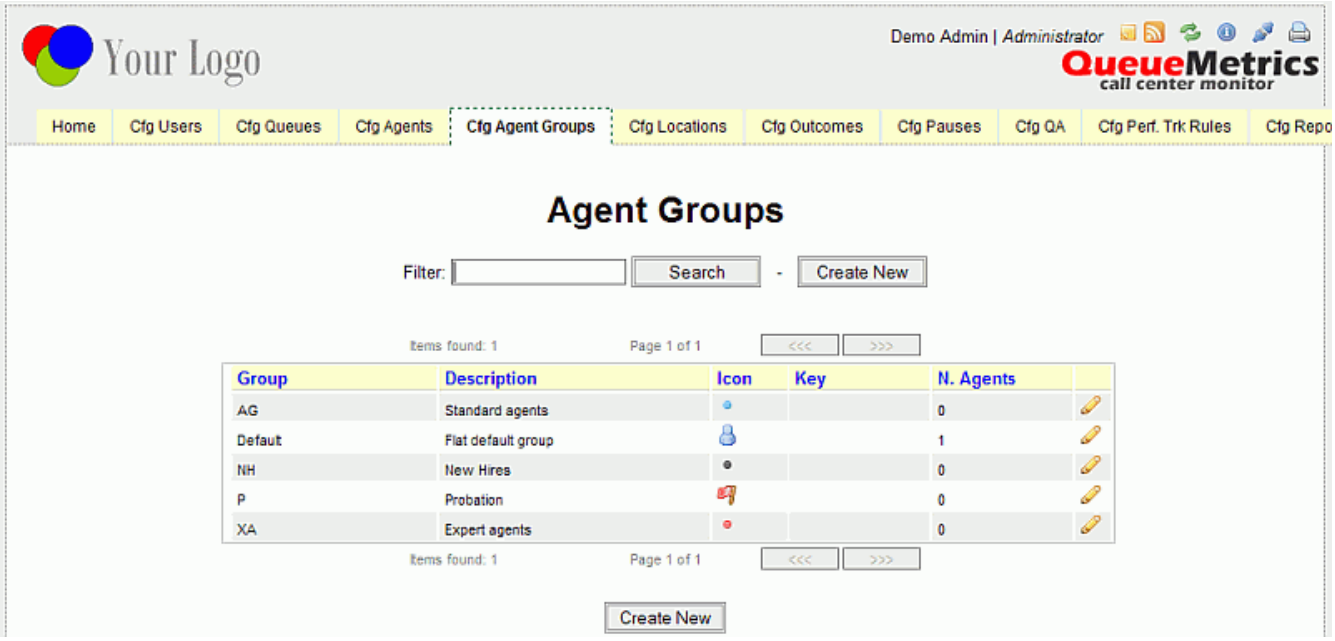


pause status codes MUST be defined with no whitespaces. Apart from numbers, only lowercase letters and the following characters are also allowed: _-.@/()=?^ . QueueMetrics will automatically delete wrong characters, change uppercase characters to their lowercase equivalent and trim the whitespaces.

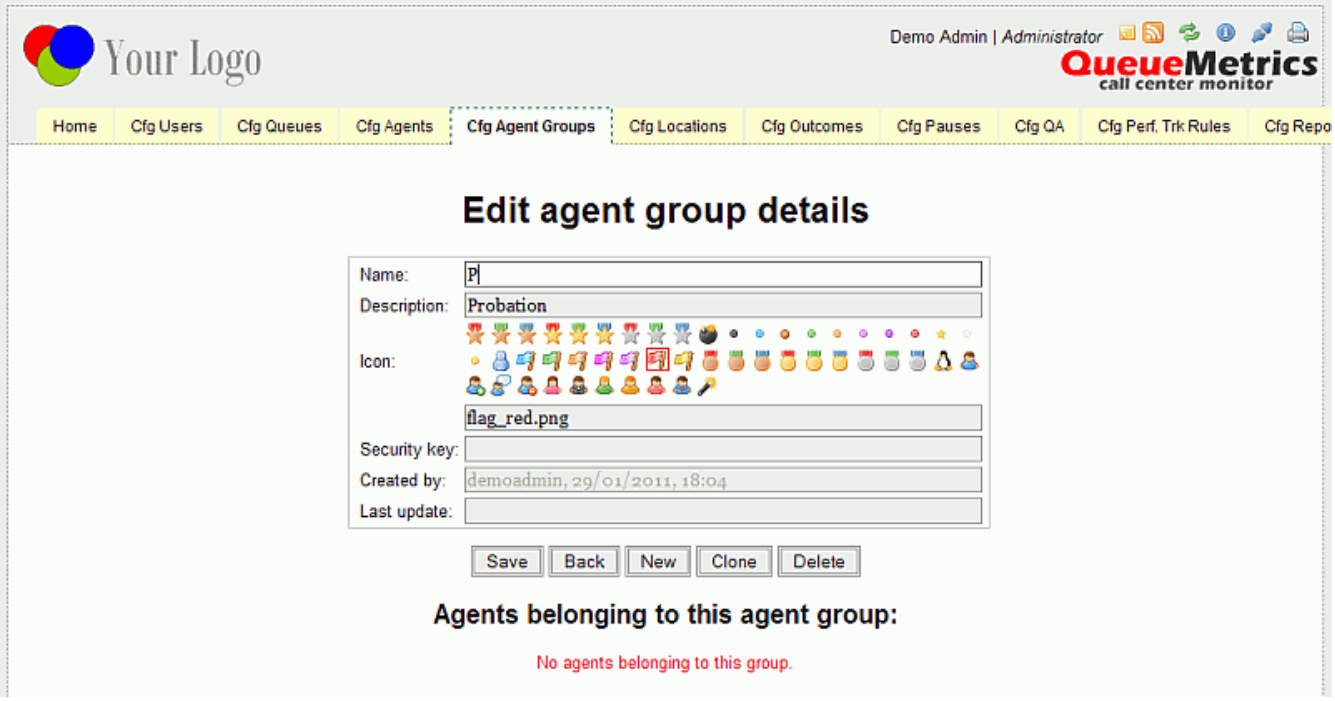
Configuring agent groups

An agent group is an attribute that is applied to an agent in order to keep track of their life-cycle. This is used as a filtering criterion in QM and is developed in the QA monitoring subsystem.

The idea here is that you can have e.g. "New hires", "Regular agents", "Expert agents"; each agent group has a different icon, that is displayed throughout QM whenever the agent name is displayed.



You can select different icons by clicking on them in the editor.

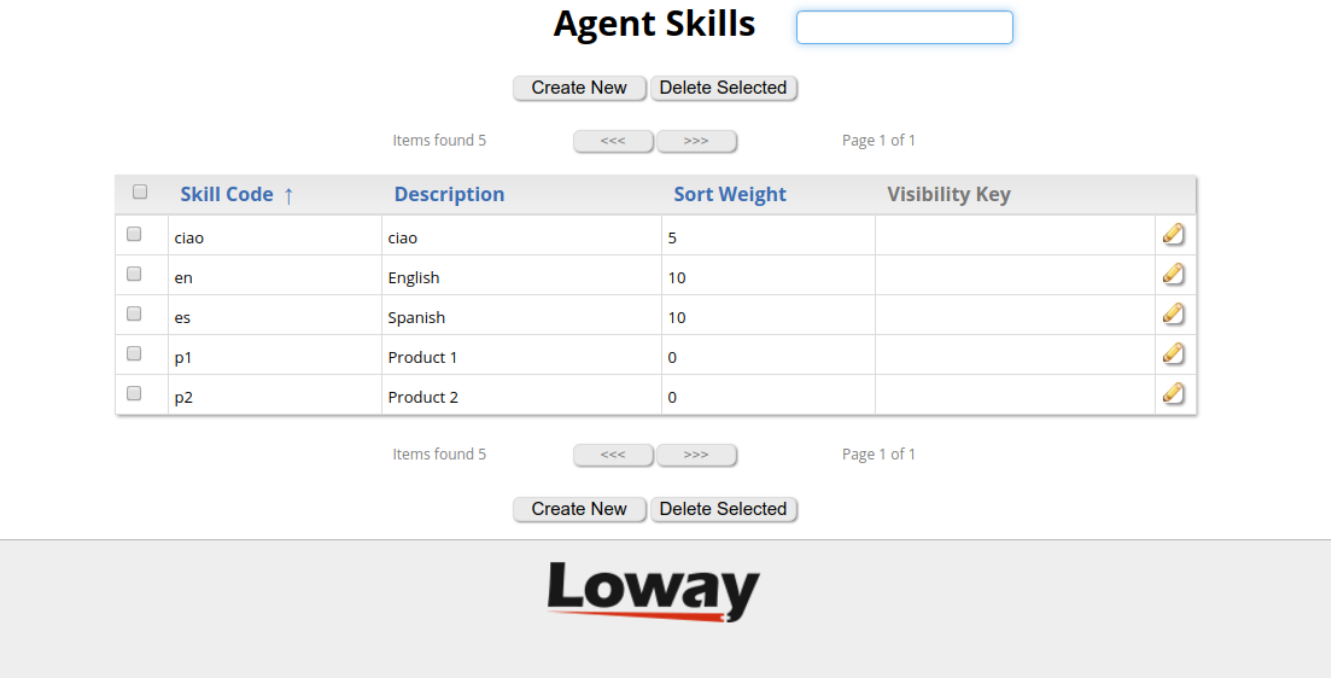




you can add more icons that better suit your liking by uploading them to the */img/agent_groups* folder in QueueMetrics.

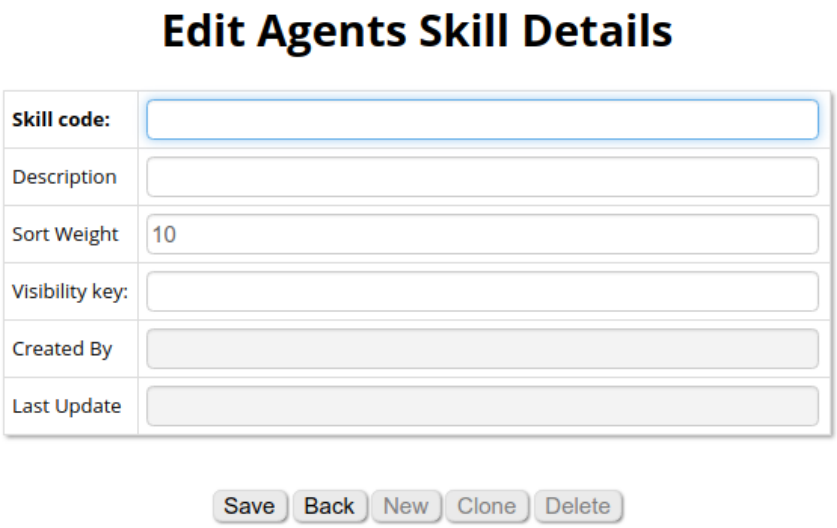
Configuring skills

Skills are attributes that can be associated to a call and can be used to implement somethig useful like skill based routing on Asterisk queues. QueueMetrics reports can be filtered by a set of specific skills through the custom report page. Skill codes can be associated to labels through the skill configuration page available to all users holding the *USR_SKILLS* security key.



The list of available skills can be reached from the main page.

Clicking on the *Create New* button or the pencil icon on each skill reported in the table, is possible to create or edit a skill entry.



Apart from the skill code and the description for the skill that can be defined, a field named **Sort Weight** can be found. The weight field is used to define the order in which skills will be shown in report columns. Skills with a lower weight will always be placed before skills with a higher weight. If two skills have the same weight, they will be ordered based on the alphabetical order of their description.

The **Visibility key** field indicates that a user must possess the chosen key to use this particular skill.

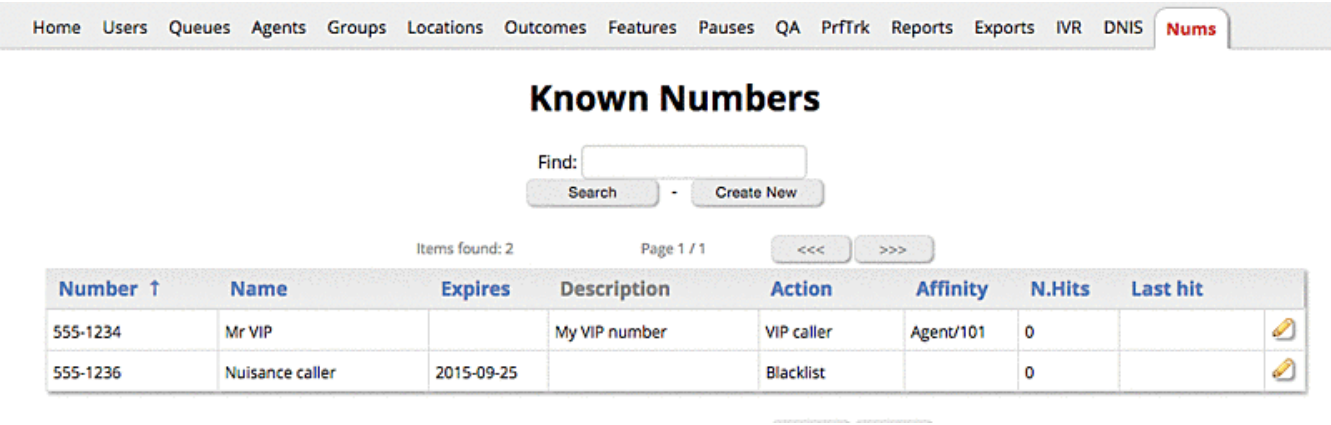
Configuring known numbers (VIP callers / blacklists)

Known numbers are phone numbers that are known to QueueMetrics and can have properties associated to them. They are meant to be used by the numberLookup API in order to implement strategies based on the calling number - see Using known numbers - and will be used to "improve" the display of said numbers as described in Displaying numbers.

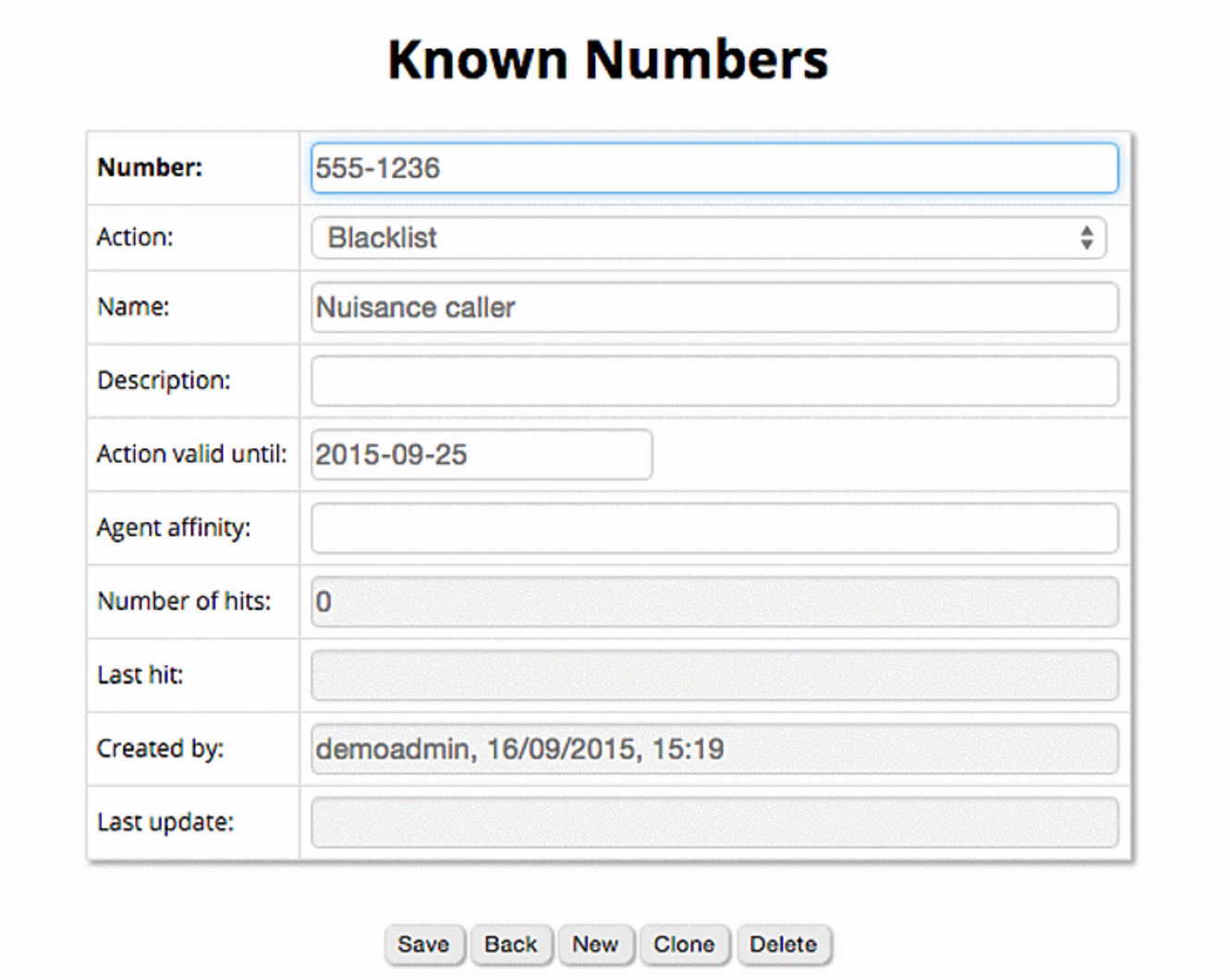
QueueMetrics will use "known numbers" to:

- Decode the number when displayed in reports and real-time pages
- Earmark specific calls as BLACKLIST or VIP, where at the dialplan level a proper function is implemented to route thise numbers to specific high priority / low priority queues
- Store a known description for the number that can be used to set a caller-id for the number in a format that will make the number "stand out". It coud be the name of a client, or their CRM IDs
- Store a known agent affinity for the number, where at the dialplan level there is an attempt to route the call to a specific agent before queuing the number on a "general purpouse" queue

In order to access this page, users must hold the security key *USR_KNUMBERS*.



When editing a record, you see a form like the following one:



The editor will track:

- A number. This can be input freely, es "(555) 555-5555", and will be internally normalized to the sequence of DTMF digits it uses.
- A name. This will be used by the service "name"
- An action, currently belonging to the set:

    - BLACKLIST
    - VIP
    - None (used in the case you just want affinity or name decoding)
- A free text descripton
- An agent id, to set the agent affinity for this number
- A record expiry date. When the record is expired, all associated attributes are inactive (action, name and affinity). This does not affect number rewriting.

The editor will also display, in read-only mode:

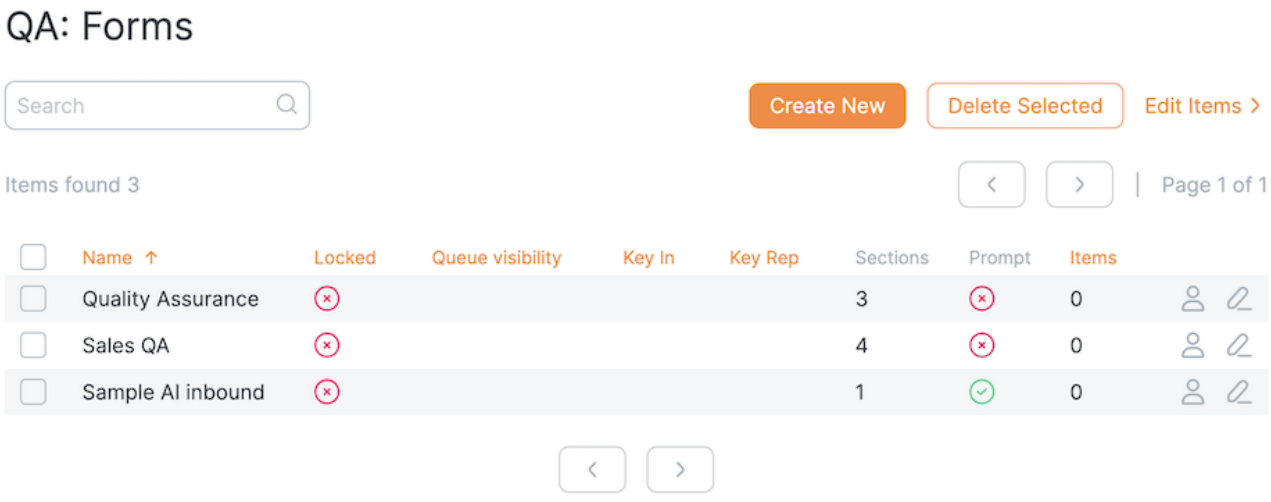
- The total number of hits that were received for this number. A hit is only defined as a successful lookup as performed by the PBX though its webservice, and won't include the number being decoded.
- The time this number was last hit.



Each number can only appear once (as per its normalized form) in the editor; attempts to have it listed multiple times will be rejected.

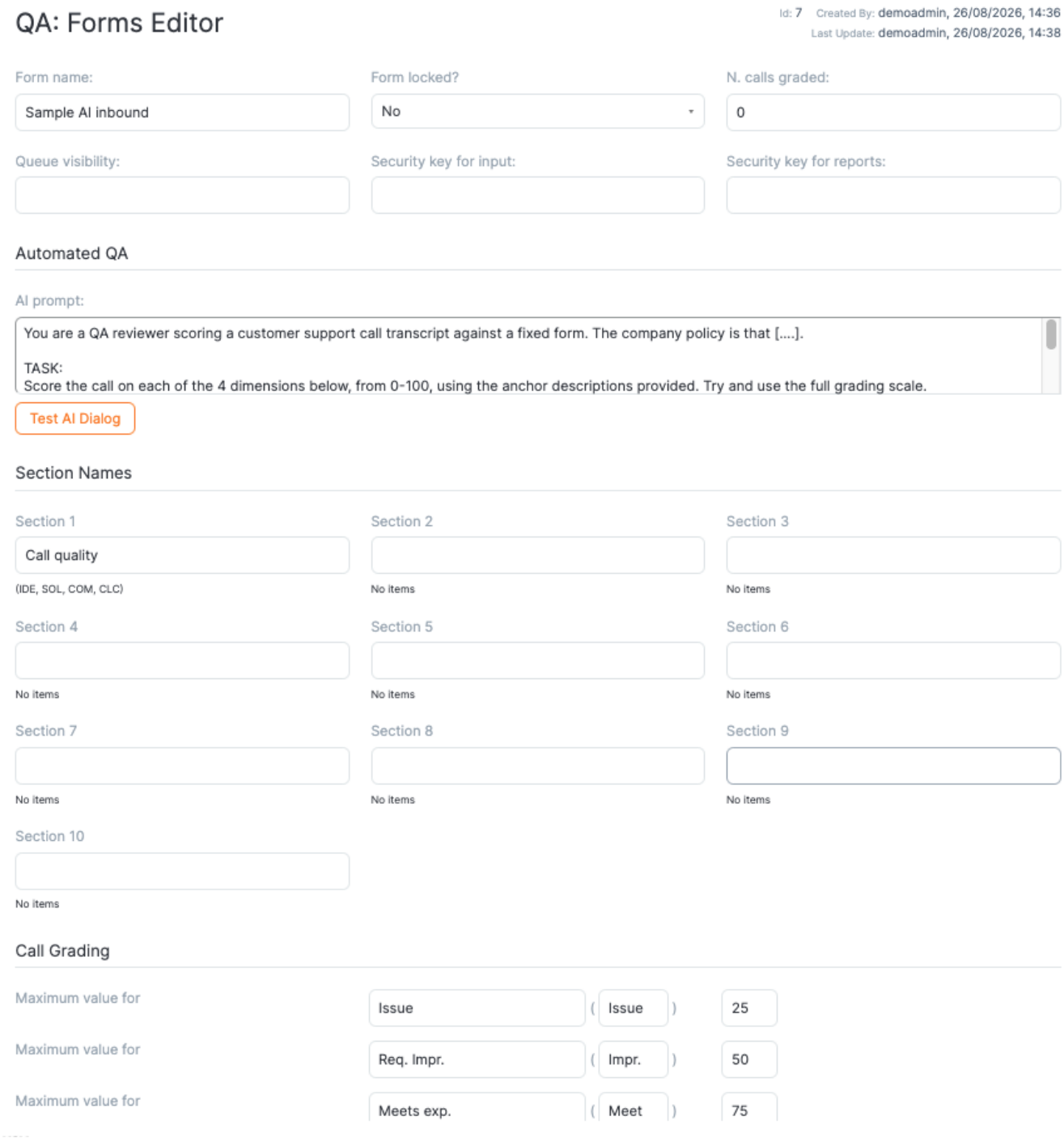
Configuring QA forms

The set of current QA forms can be configured through the QA form editor. It shows the current set of defined forms and lets you performs the usual operations (search, filter, sorting, paged listing...).



The names of each section and the number of items that have been input for that form are shown on the front page. If a form has any number of items input, it is considered "locked" and cannot be modified anymore, though you can create a different form with the same set of items.

The form editor looks like the following page:



You can enter:

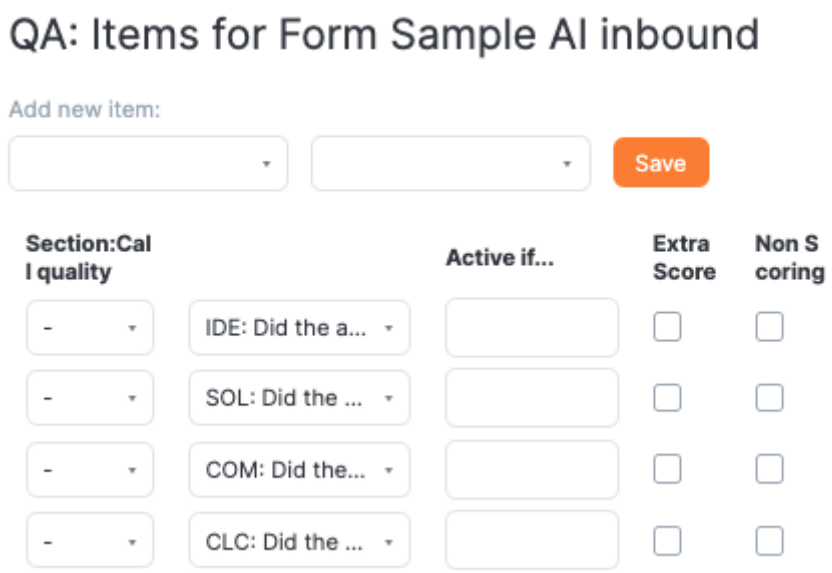
- The form name
- Whether the form allows new input or not; the total number of calls graded using this form is shown
- The queue visibility: it limits the queues on which this form is displayed. It may contain a set of queue identifiers, separated by space.
- The security key required (in addition to the basic one) to grade calls using this form
- The security key required (in addition to the basic one) to run reports on this form
- The AI prompt used for Automated QA, i.e. the instructions given to the LLM when a call is graded automatically instead of by a human grader. Once a prompt is entered, you can verify how it behaves by clicking the "Test AI Dialog" button; see Troubleshooting automated QA for

details.

- The names of the section. There can be up to ten sections. Any section MUST have a name, and there must be at least one section.
- The threshold values for Call grading (the maximum for "Exceeds expectations" is fixed at 100).
- The labels associated to each level in the Call grading (defaults are "Issue", "Req.Impr.", "Meets exp.", "Exceeds exp."). Please note that a short label, used on the report pages, could be also defined (defaults are "Issue", "Impr.", "Meet", "Exc.").
- A *Queue visibility* list - you can list a set of atomic queues for which this form will be displayed. Queues are separated by whitespaces. The field accepts wildcards to match multiple queues and/or subqueues. If the field is left blank, then the form is displayed for all queues.

To edit the set of items that belong to a form, you should have no data reported for that form. If you have no data, the item editor icon will show from the main form page.

The editor looks like the following page:



To add a new element, just select an element on the top form and a section it should be added to.

On the main part of the page, you can edit the elements by changing them, moving them up or down and removing them.

The "Active if..." field allows to enter a dependent rule, as described within the following chapter: "Handling dependent questions".

The Extra Score checkbox allows to give extra scoring to a particular question, in order to highlight an agent's higher performance within a specific section, or to balance his/her averages. The question in the form will have two main characteristics: it can have a score above the 100 standard value (up to a maximum of 120) and the score assigned to the question will not be calculated within the denominator part, when totalling averages for the specific section and the overall QA form.

The "Non Scoring" checkbox, if checked, forces the engine to ignore the inserted question score when calculating the section and form average. Questions flagged as "Non Scoring" are not

classified as "under or over performing items" in the QA form. Non scoring questions are aggregated on reports based on the number of times a particular score is found in the query period.



When only one extra score question is within a section, the average will show as zero, given that within the denominator there are no values, but it is null.

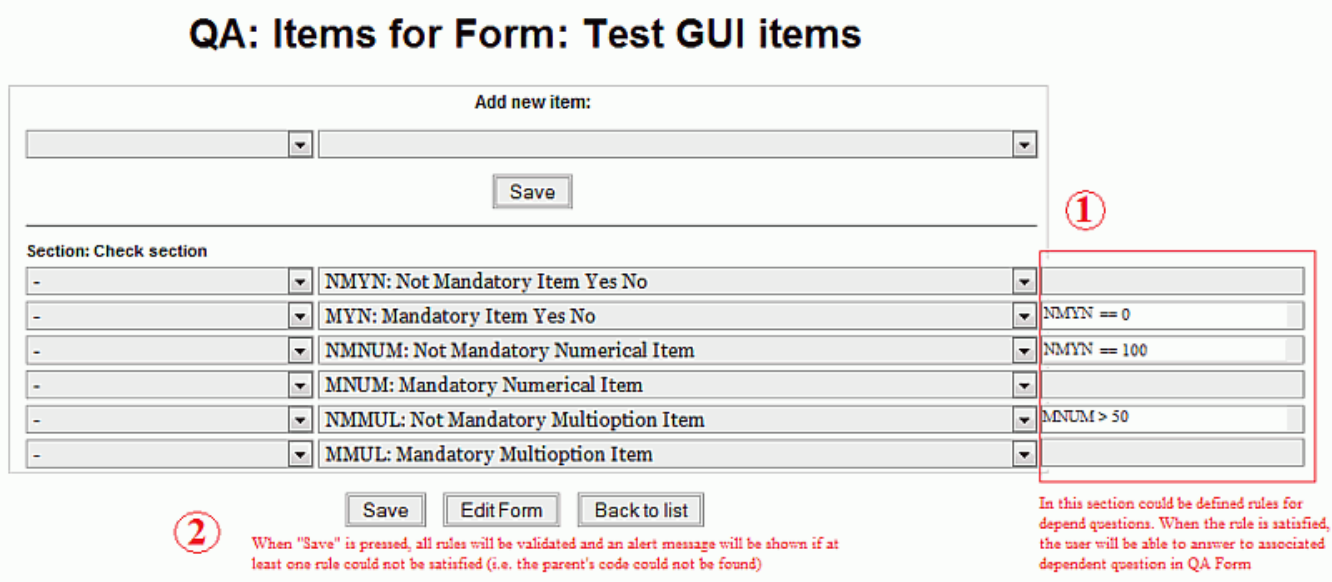
Handling dependent questions

It is possible to have **Dependent Questions**, i.e. questions that are enabled or disabled according to values input in a different question.

In order to enable this, a new input area is defined on the right side of each item. Here the administrator can insert a rule that associated the question to a particular parent question. There are several restrictions in the rule definitions:

- A rule should be defined with the sequence LABEL, OPERATOR and VALUE where:
  - LABEL is the short code associated to each item in the section
  - OPERATOR is a mathematical operator in the subset: <, >, ⇐, >=, ==, !=
  - VALUE is a numerical value in the interval 0 (included) to 100 (included)
- A rule cannot be composed by other rules

When the administrator saves the form, the server will parse each inserted rule and, if an invlalid rule is be found, an exception is raised specifying the first incorrect rule found. The administrator must then modify the wrong rule before submitting the form definition again.



For example, a few rules are defined in the picture above:

- The item NMYN is parent of two items: MYN and NMNUM. The MYN item is enabled only if the value inserted is equal to 0; the NMNUM item is enabled only if the value associated to the NMYN question is equal to 100.
- The item NMMUL is related to the itemMNUM. This question is enabled only if the score given to the MNUM item is greater then 50.

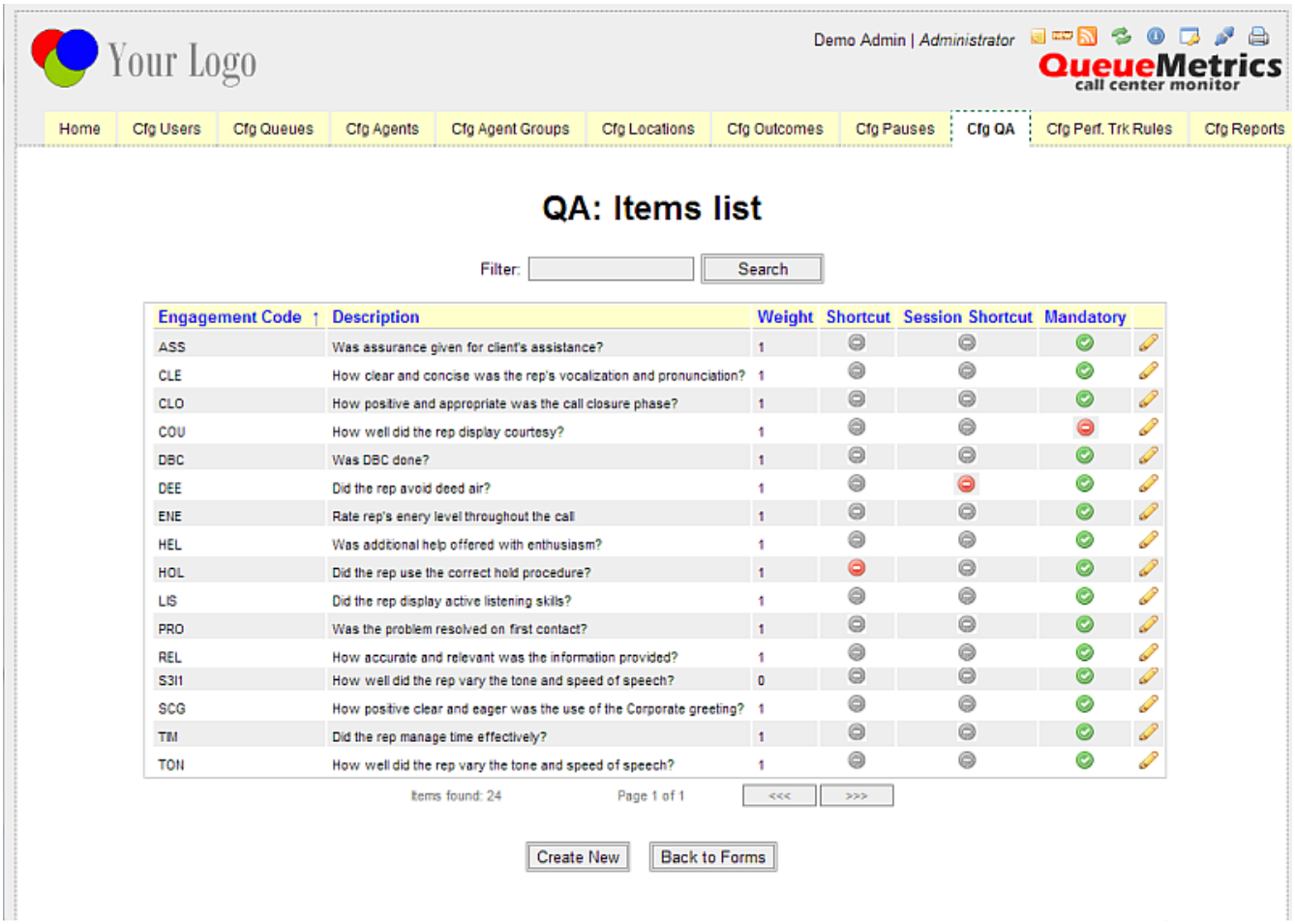
- The question MMUL is a normal question and is always enabled.

A couple of rules govern the way values for dependent question are handled when the question is disabled:

- Dependent questions, when saved, receive a "N/A" value, so they behave like non-madatory items when you check the "N/A" box. The value will be counted in the grading report statistics following the same rule.
- The "N/A" value works with the same rule even if the dependent question is a shortcut one: if it is disabled it is not considered a shortcut.

Configuring QA items

The set of items that are selectable as members of a form can be configured by the user by clicking on the "Edit items" button at the bottom of the QA forms editor.



New items can be added and the description can be edited if needed.

The **weight** of an item is the number of times that an item must be counted in comparison to other items. All form level scores consider the item's weight.



If you average an item with a score of 50 and one with a score of 100, each weighting one item, you have an average of (50+100)/2 = 75. If you average an item with score 50 and weight 2 and one with score 100 and weight one, you get ((50 + 50)+100)/3 = 66!

A **shortcut** item is an item that, if failed, will fail the entire form. They are marked with a red icon when present.



If a shortcut item fails, the whole form will have an overall score of zero, no matter what other scores are. While averages at the form level are affected by shortcuts, averages at the item level are not affected.

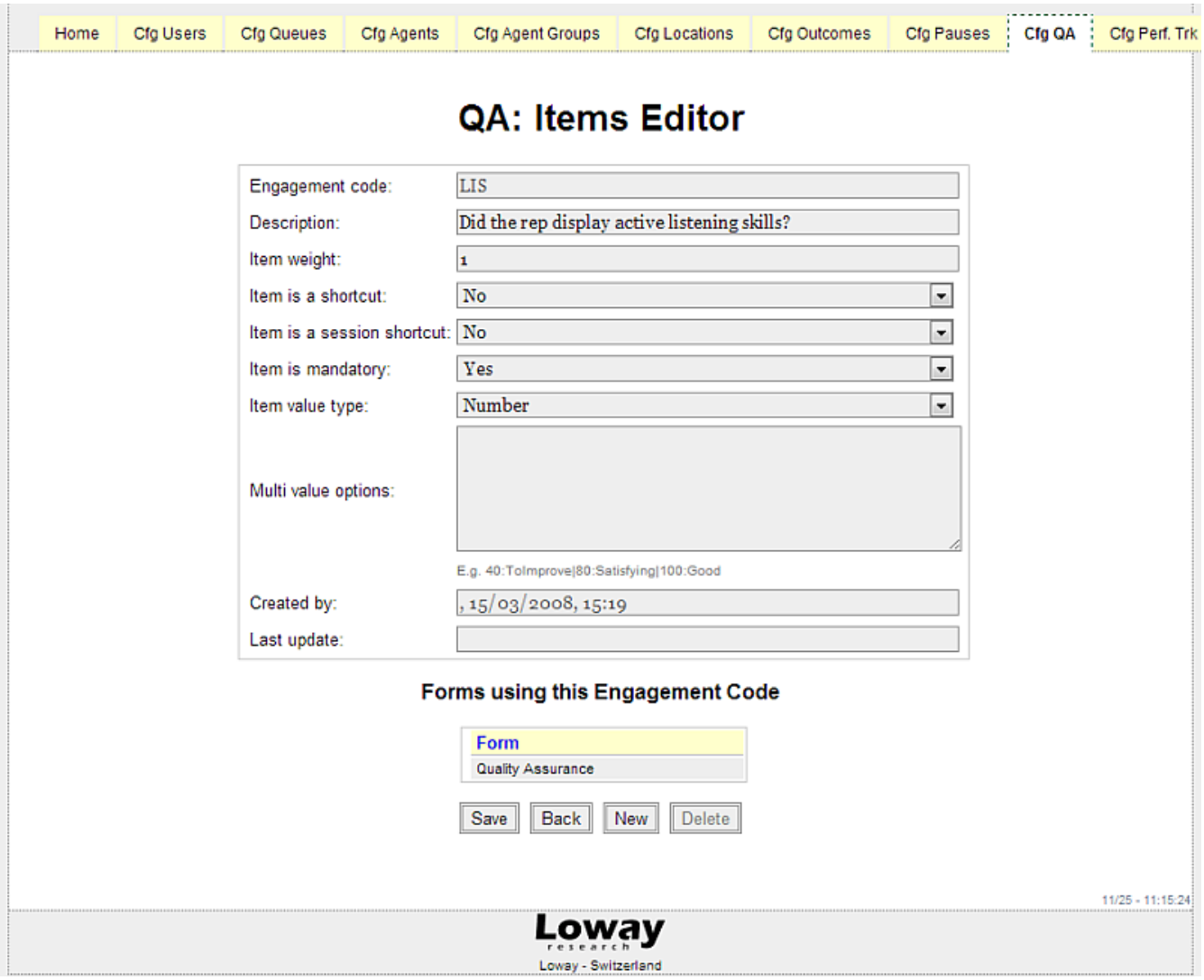
A **session shortcut** item is an item that, if failed, will fail the entire section. They are marked with a red icon when present.



If a session shortcut item fails, the whole section will have an overall score of zero, no matter what other scores are.

A **mandatory** field requires you to select a valid value; if it is not checked, the user can optionally leave those items blank. Mandatory fields are marked with a green icon.

Different kinds of input can be accepted by the item - see below.



The Engagement code cannot be modified once input, and there can be no two items with the same engagement code.

Items cannot be deleted if they are in use by at least one form. You can see the set of forms that are

using the chosen it at the bottom of the editor's page.

Item value type: Numeric

A numeric value type must be an integer number between 0 and 100, extremes included.

Item value type: Yes/No

A Yes/No value stores 0 when set to No and 100 if set to Yes.

Item value type: Combo

This lets you create a drop-down selector, asscociating each entry with a given value.

You enter the list of values in "Multi value format" text box, with the format:

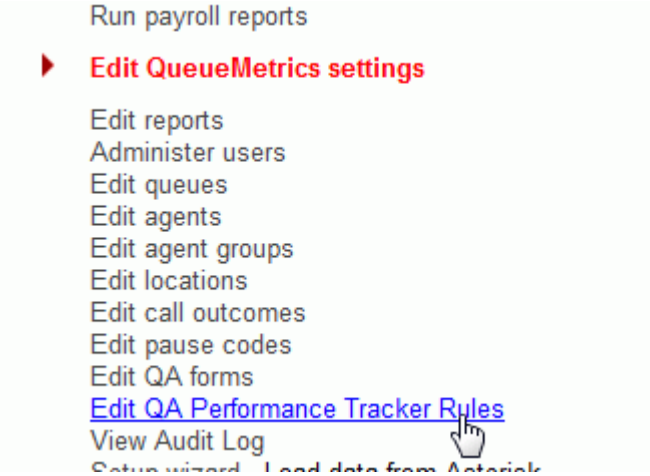
```
40:ToImprove|80:Satisfying|100:Good
```

Each option is made up of a numeric value plus the colon symbol ":" plus the text to be displayed. Multiple options are separated by the pipe "|" character.

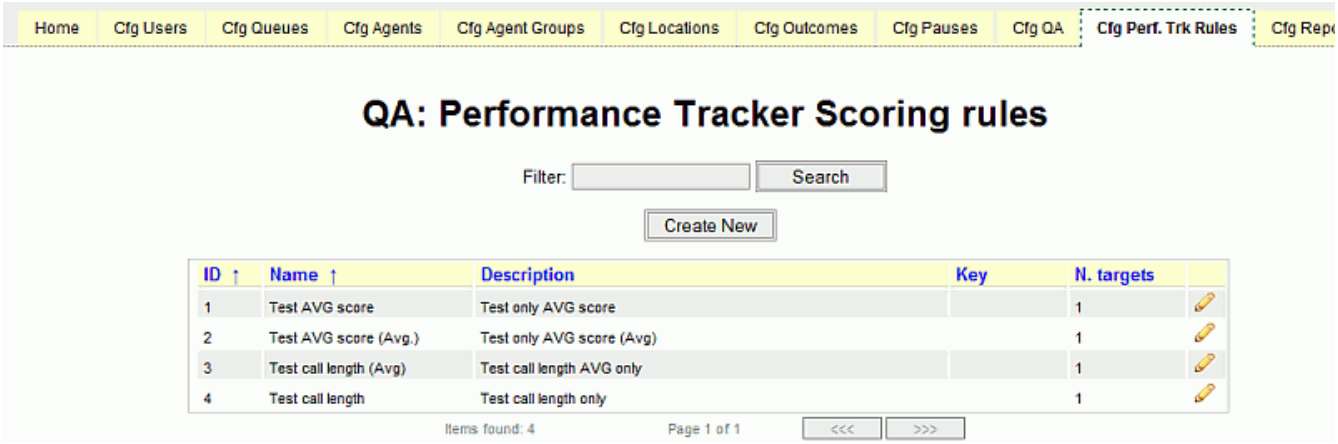
Defining agent performance rules

As explained above, in order to track performance you first have to express a set of **business targets** that express what is expected from your agents and how much deviations from each rule are comparatively worth, expressed in **review points**. This is called a **ruleset**.

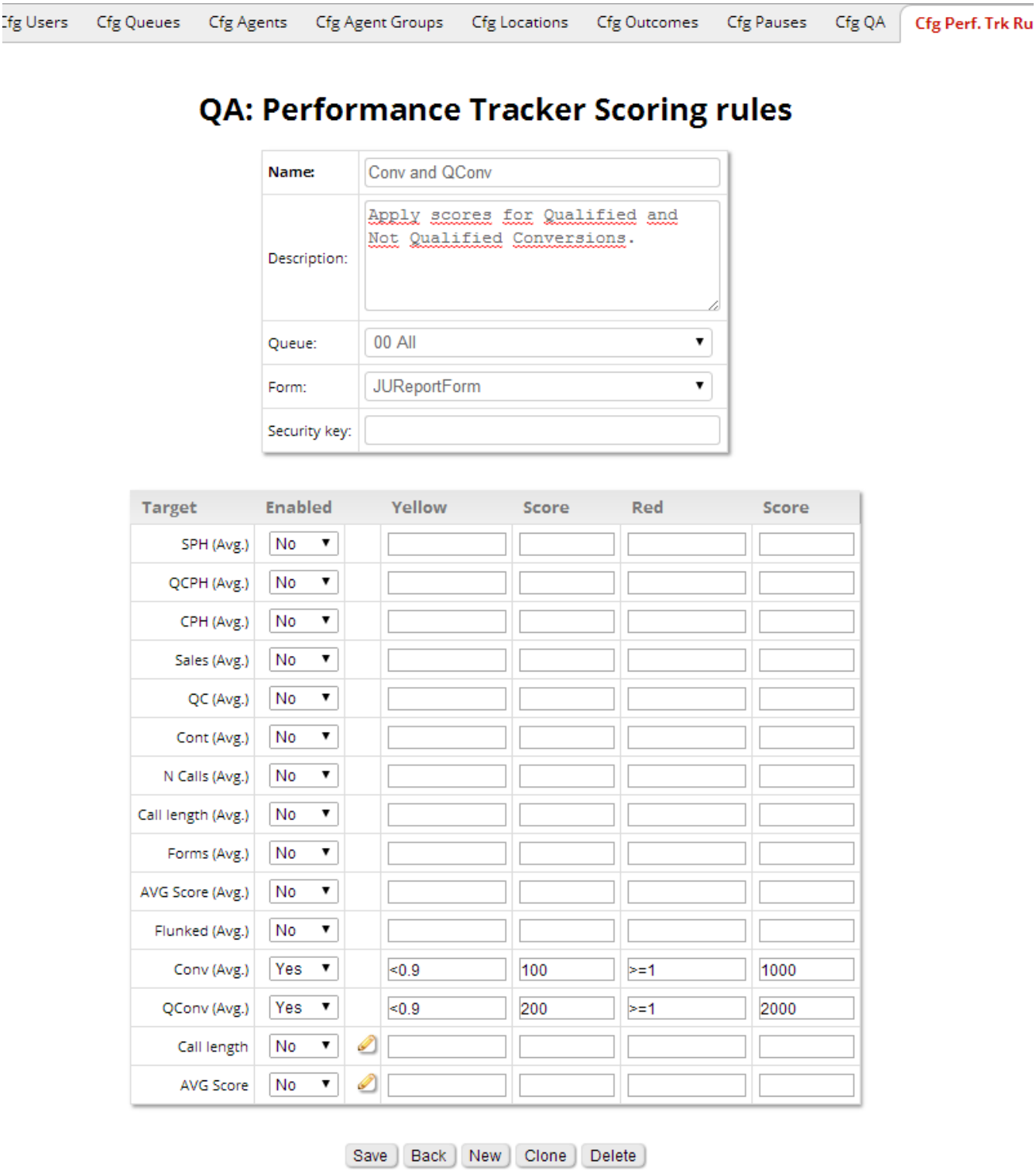
This can be done through the proper configuration page by users holding the key "QA_PERF_RULES"; they will see a new link from the home page:



Selecting the link, a new page is shown listing rulesets already defined. In order to define a new ruleset, you press the "Create New" button.



The "Create New" button opens a new page where an empty rule is shown, like in the picture below.





Targets will not be displayed until you first save the ruleset.

The creator should assign to the new ruleset a name, a short description, and optionally a security key.

A rule is usually linked to a specific queue (or set of queues) and form. This is because we expect to have homogeneous statistic distributions in the same queue and form items. This might not be true outside a specific form and/or queue. The user should select a specific queue and form before pressing "Save".

Available targets

When editing a ruleset, you see it is actually built out of a number of targets. It is important to understand that there are basically two different kinds of targets:

- "Aggregated" targets - identified by **AVG** - that are computed once per agent, and
- "Atomic" targets that are computed for each call handled by the agent

When computing the review score for an agent, first each call is checked against atomic targets and a first score is computed, then averages for the dataset are taken, and they are computed against aggregated targets and an aggregate score is computed; the final score is the sum of both scores.



You can use either type of target, or both as once, as you best see fit. Try and run some tests to make yourself familiar with the ruleset.



It is important to note that some targets are not available as atomic targets. Examples are the QCPH, Sales, Number of calls, (Qualified) Conversion etc. that are obviously associated to a set of calls and make no sense in relation to a single call.

Setting targets

For each possible target within the rule set, you can:

- Enable or Disable a specific target rule
- Insert an algebraic expression defining the rule for the "yellow" theshold
- Specify a score that the engine will assign to the target when matching the "yellow" expression
- Insert an algebraic expression defining the rule for the "red" level
- Specify a score that the engine will assign to the target when matching the "red" expression

The algebraic expressions that can be used to define a threshold are:

- simple mathematical expressions formed by an operator (in the set of "<", ">", "⇐", ">=") and a value.
- X << Y: defines the internal interval between the values X and Y (excluded)
- X ⇐ ⇐ Y: defines the internal interval between the values X and Y (included)
- X >> Y: defines the external interval outside the values X and Y (excluded)
- X >⇒= Y: defines the external interval outside the values X and Y (included)

Valid examples are:

- "< 10" is triggered by a number lower than 10
- ">= 40" is triggered by a number greater or equalling 40
- "40 << 80" is triggered by a number between 40 and 80

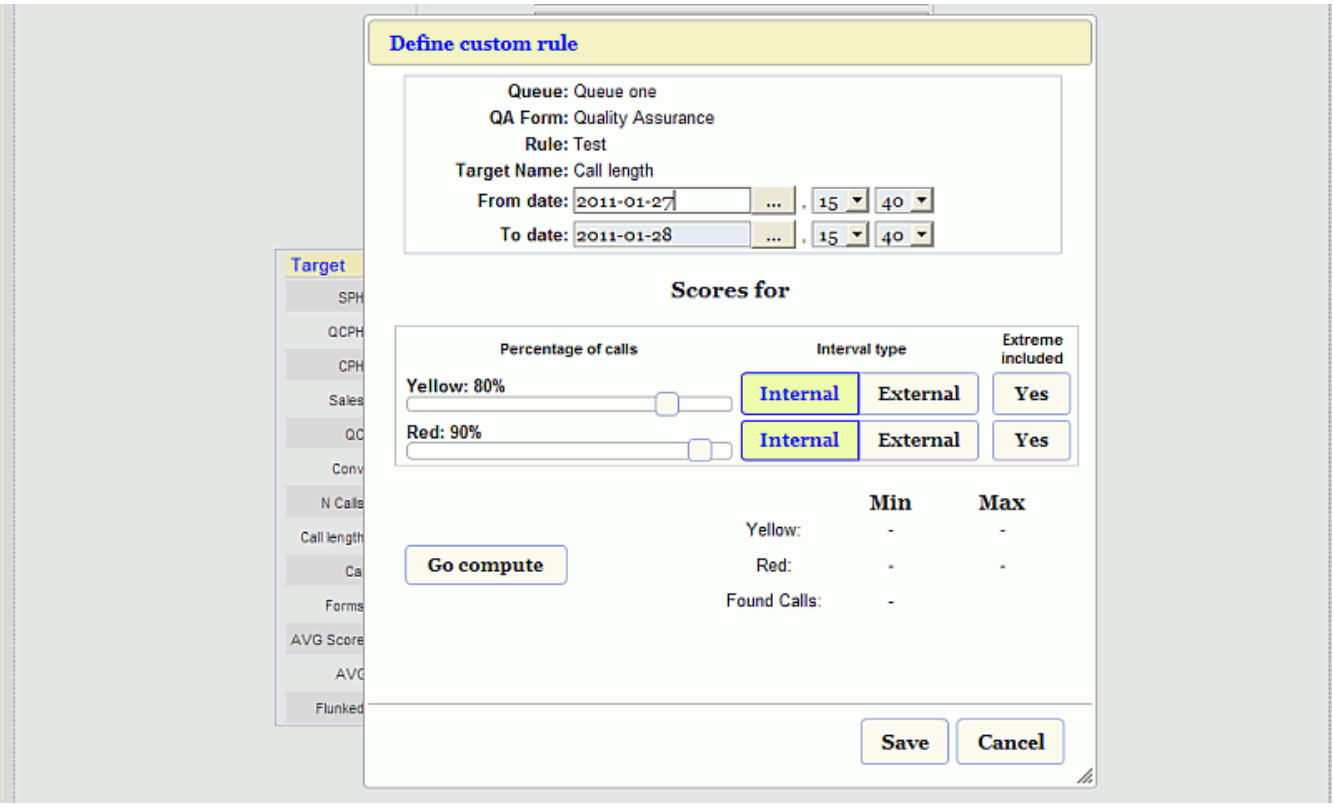
For not-averaged rules the user can access a wizard that simplifies definition of interval-based

rules.

Dataset-based agent performance wizard

A rule set can be inferred from the measured properties of a given data set; this basically lets you express differences in terms of a percentage of outliers expressed on the total number of calls.

In order to access the wizard, you click on the "pencil" icon:



In this modal dialog you define a start and end time period and the "yellow" and "red" percentage of calls the user wants to include in the resulting rule, the type of interval (internal or external) and whether the interval extreme values should be included or not in the resulting rule.

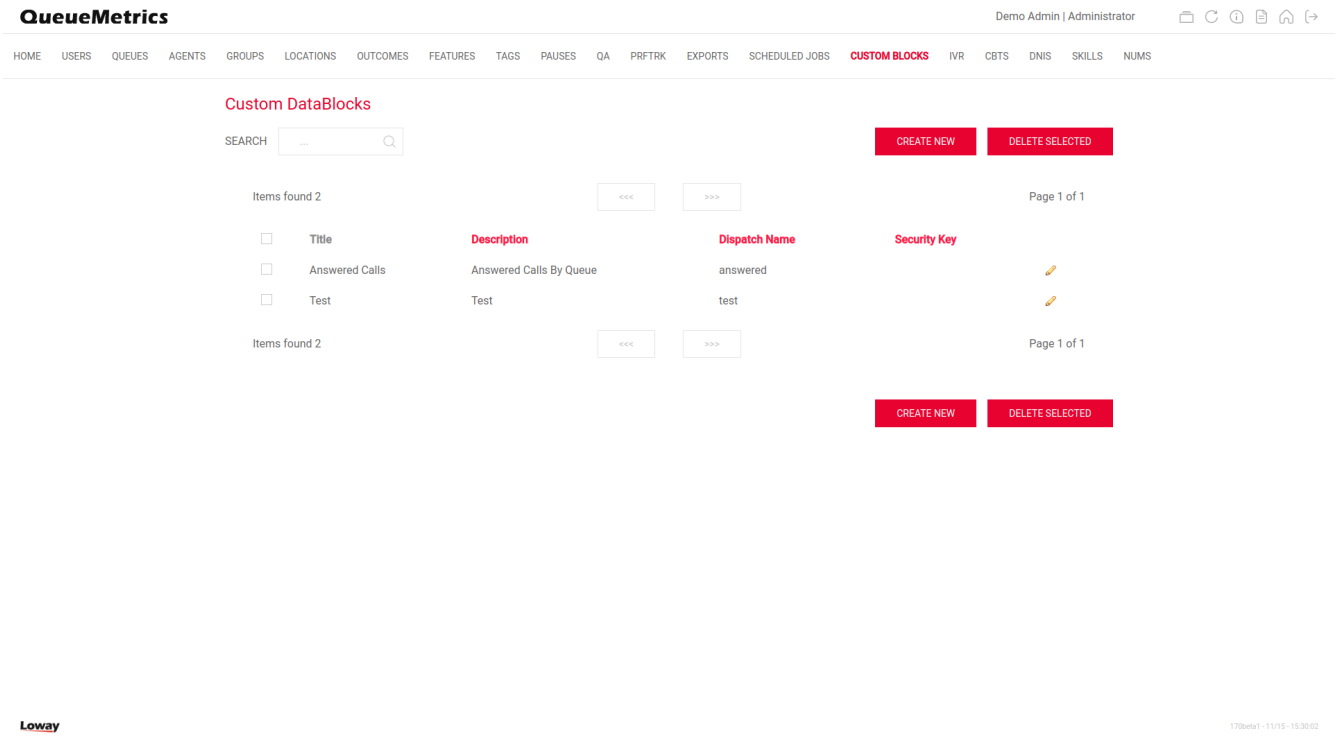


Imagine you want to consider "yellow" the 10% of calls that are too long or too short and red the 5% of calls that are way too long or way too short in relation to the average length. You would set the "yellow" slider to "90% external" (meaning you want the external tails) and the "red" slider to "98% external".

The "Go compute" button runs an internal analysis that reports, in the lower right table present in the dialog, the minimum and maximum values representing the interval fulfilling the inserted parameters and the number of calls analyzed. You can repeat the calculation until satisfied, then press "Save" to insert the rule in the rule-set or press "Cancel" to forget it.

Configuring Custom DataBlocks

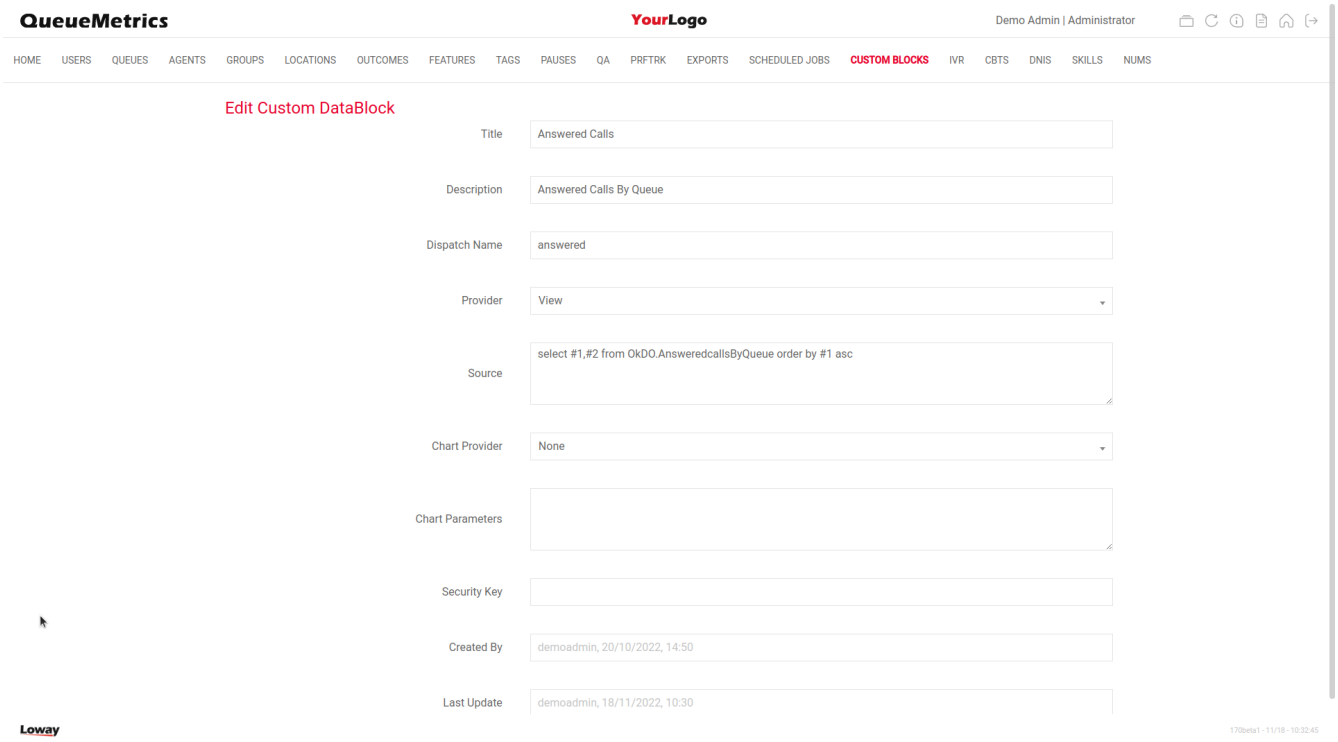
Users holding the key USR_CUSTOMBLOCKS_EDIT can edit the list of Custom DataBlocks.



QueueMetrics allows the user to create their own Custom DataBlocks, by specifying a source and a list of columns to show.

Custom DataBlocks are filtered views of existing DataBlocks. For this reason, these DataBlocks need to be based on an existing DataBlock (including another Custom DataBlock).

When creating a new Custom DataBlock, the user needs to input the following options:



- Title
- Description

- Dispatch Name: This will be the actual name of the DataBlock. Has to be unique.
- Provider: Type of DataBlock. Only View can be selected.
- Source: The source code of the DataBlock (More info below).
- Chart Provider: The type of Chart, if any. Only React is available.
- Chart Parameters: More information below.
- Security Key: The security key for the block, in case access needs to be restricted to some users.

DataBlock Source Code

A Custom DataBlock's source code is the formal definition of said custom DataBlock. By defining a Custom DataBlock, you can create a view of another existing customblock, by specifying the origin DataBlock and the colums you want to pick from that DataBlock. You can also specify an order (ascendant or descendant) for a specific field.

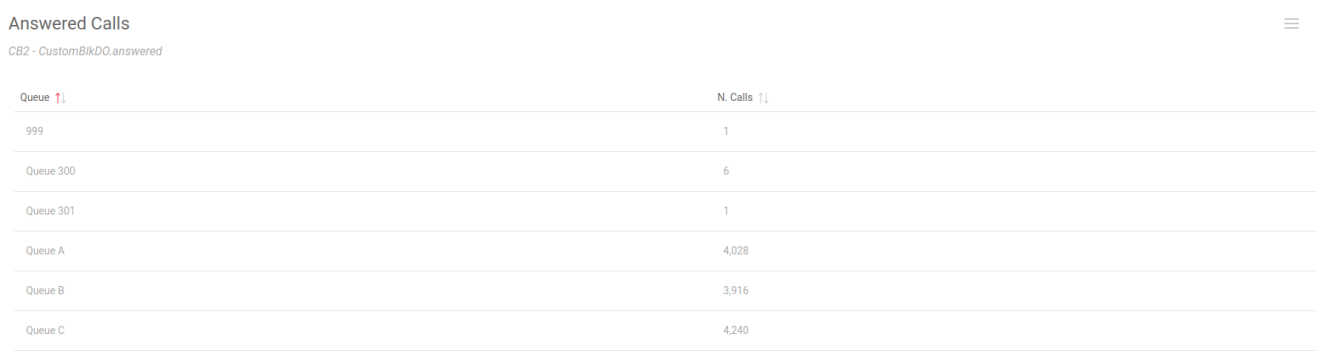
A well formed DataBlock is defined as such:

```
select {COLUMNS} from {DATABLOCK} order by {COLUMN} {DIRECTION}
```

so, for example:

```
select #1,#2 from OkDO.AnsweredcallsByQueue order by #1 asc
```

Will generate a DataBlock like this one:



Where only Column #1 and Column #2 of the original DataBlock are visible, sorted by field #1 in a ascending order.

The sorting part of the source can be omitted, but the select and from clauses are mandatory.

```
select {COLUMNS} from {DATABLOCK}
```

is a perfectly valid DataBlock.

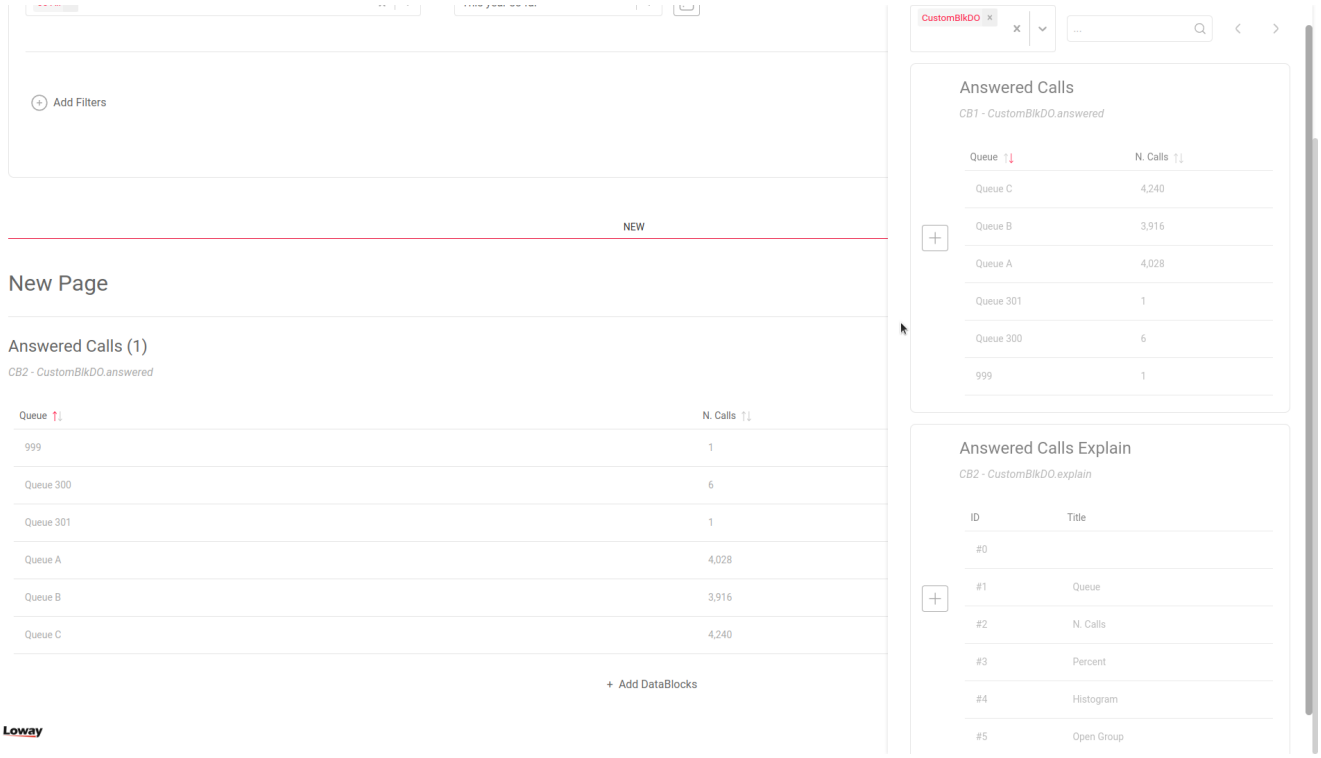
```
select #1,#2 from OkDO.AnsweredcallsByQueue
```

If you want to select all the columns of the original DataBlock, you can use the following syntax:

```
select * from OkDO.AnsweredcallsByQueue
```

Adding Custom DataBlocks to a report

To add Custom DataBlocks to a report, you need to look for them in the DataBlock Selector in QueueMetrics' Report Page. You can access the DataBlock Selector by clicking on the "Add Datablock" button at the end of every page, or from the Reports' Page command menu



Explain DataBlocks

Since not all DataBlocks will have columns with ID #1 and #2, to create a view of an existing DataBlock it's usually better to create an "Explain" DataBlock for the DataBlock we want to create a view of.

An Explain DataBlock is a DataBlock that, when added to a report, will show the name and ID of the various columns of a DataBlock, so that the user can see the IDs of the columns to use in the Select clause of a Custom DataBlock's source.

The syntax to define a Custom DataBlock is as follows:

```
explain * from {DATABLOCK}
```

for example:

```
explain * from OkDO.AnsweredcallsByQueue
```

Edit Custom DataBlock

Title	Answered Calls Explain
Description	Answered Calls By Queue Explain
Dispatch Name	explain
Provider	View
Source	explain * from OkDO.AnsweredcallsByQueue
Chart Provider	None
Chart Parameters	
Security Key	

will yield the following result.

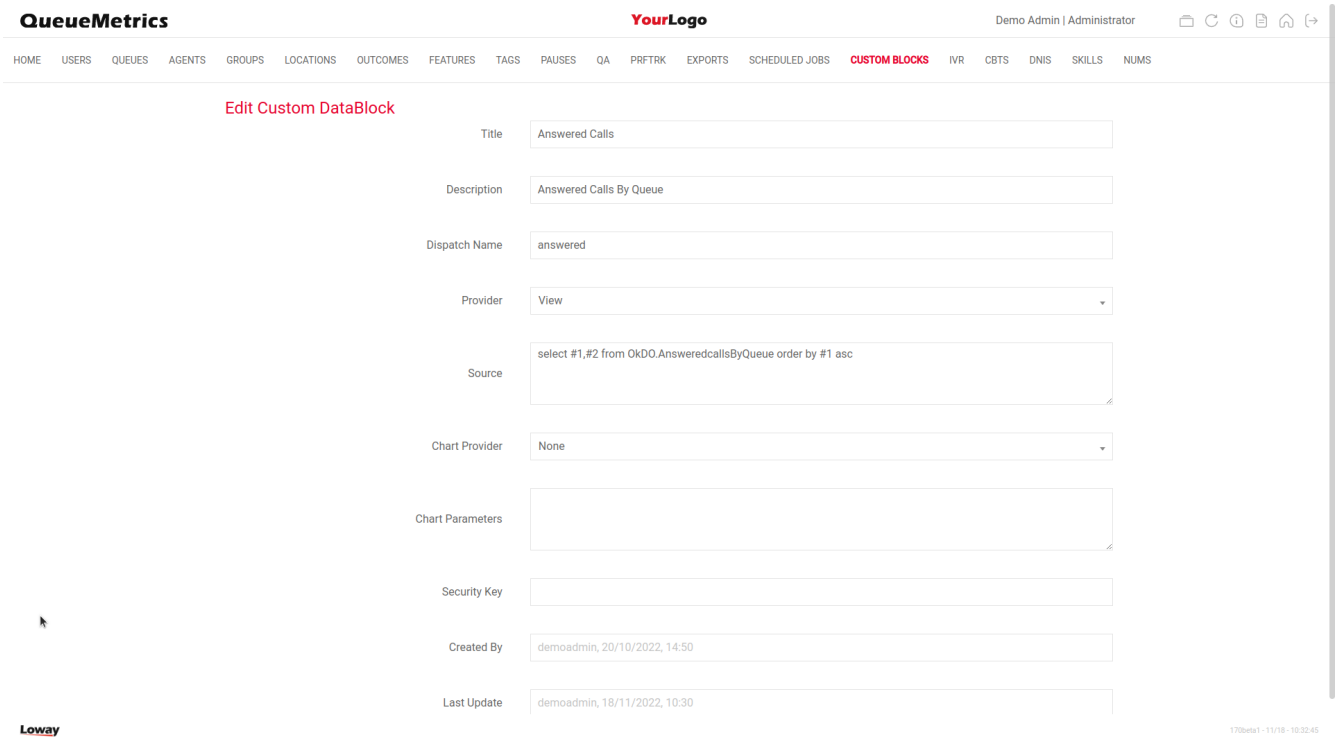
Answered Calls Explain

CB3 - CustomBlkDO.explain

ID	Title
#0	
#1	Queue
#2	N. Calls
#3	Percent
#4	Histogram
#5	Open Group

Custom DataBlocks Charts

Custom DataBlocks offer the ability to define customizable charts to go with them.



When defining a custom DataBlock, the user can select "React" as a Chart Provider, and set the parameters for the chart.

The chart parameters are built as follows:

- CHART: Type of chart. Can be PIE, BAR, LINE, RADAR.
- DATATYPE:_Type of data. Can be NUMBER, DATA or TIME.
- XCOLUMN: The column that will be used as the X Axis
- YCOlUMN: The column that will be used as the Y Axis



Data type NUMBER is to be used when the Y Column is comprised of numerical values. DATA will take the values as they are, while TIME is to be used if the values are durations in the format 00:23.

Some valid chart parameter examples for the above custom DataBlock are:

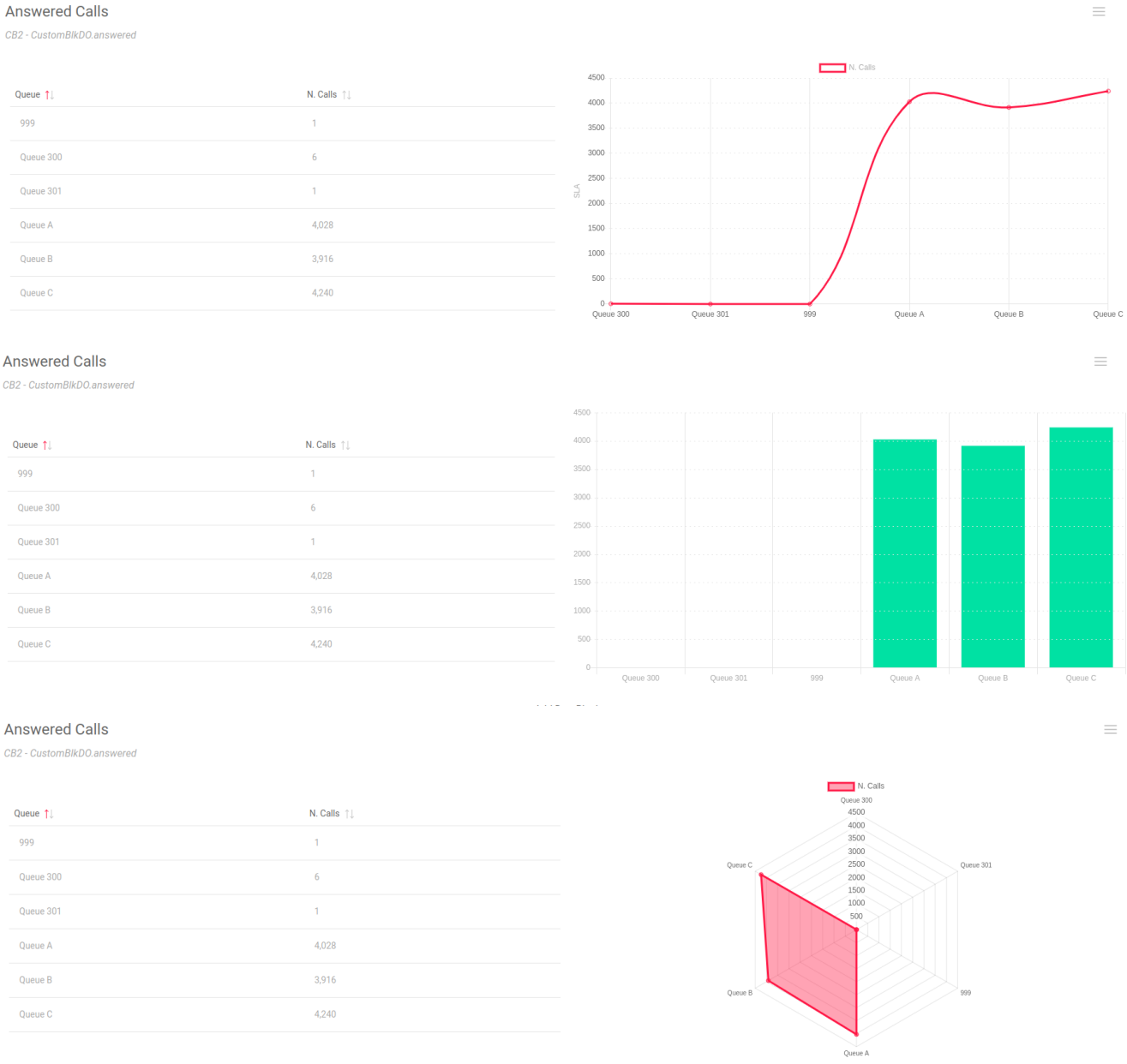
```
CHART=LINE DATATYPE=NUMBER XCOLUMN=#1 YCOLUMN=#2
```

```
CHART=BAR DATATYPE=NUMBER XCOLUMN=#1 YCOLUMN=#2
```

```
CHART=RADAR DATATYPE=NUMBER XCOLUMN=#1 YCOLUMN=#2
```

```
CHART=PIE DATATYPE=NUMBER XCOLUMN=#1 YCOLUMN=#2
```

As you can see below, these recipes will result in different charts for the DataBlock.



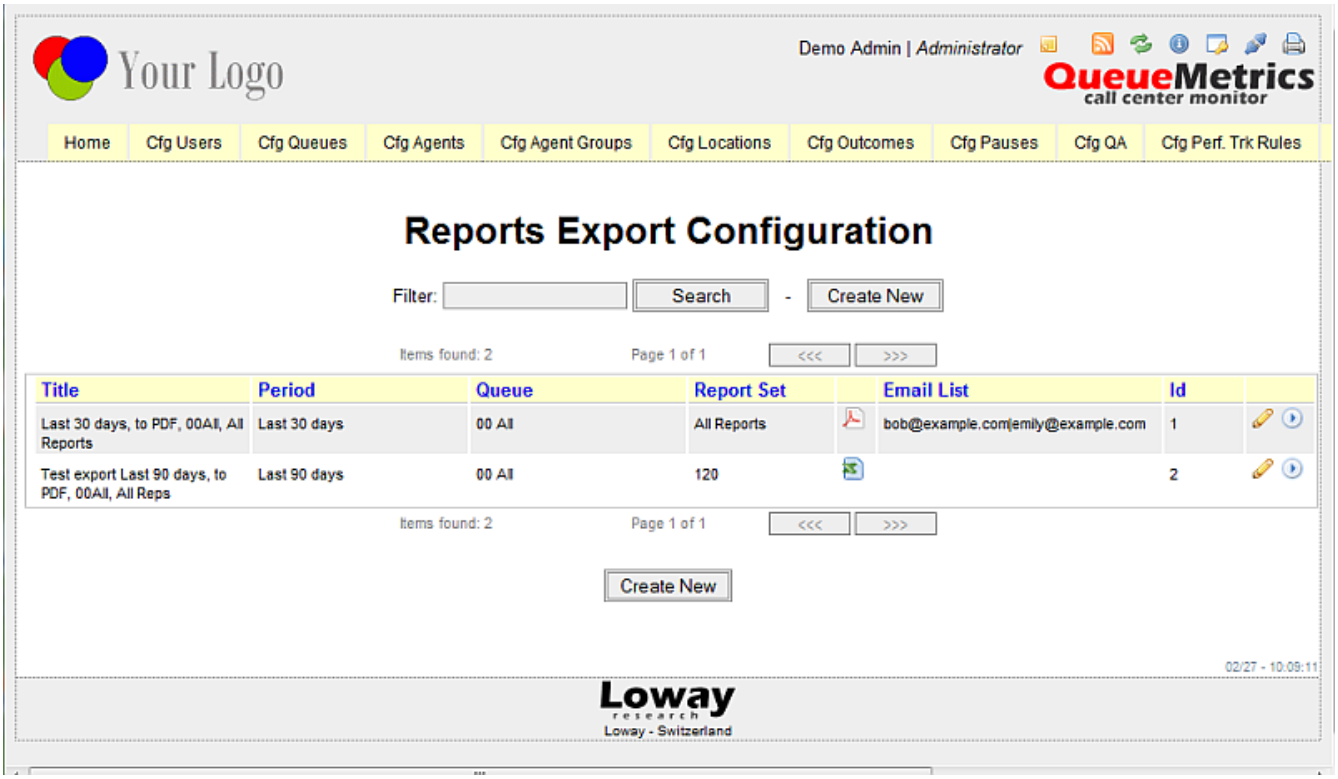
If there's an error with the chart configuration, you will see a placeholder picture in its place as such:

Queue	N. Calls
999	1
Queue 300	6
Queue 301	1
Queue A	4,028
Queue B	3,916
Queue C	4,240



Report Export Configuration pages

Users holding the key USR_REPORTS_EXPORT_EDIT can edit the list of PDF and XLS export jobs.



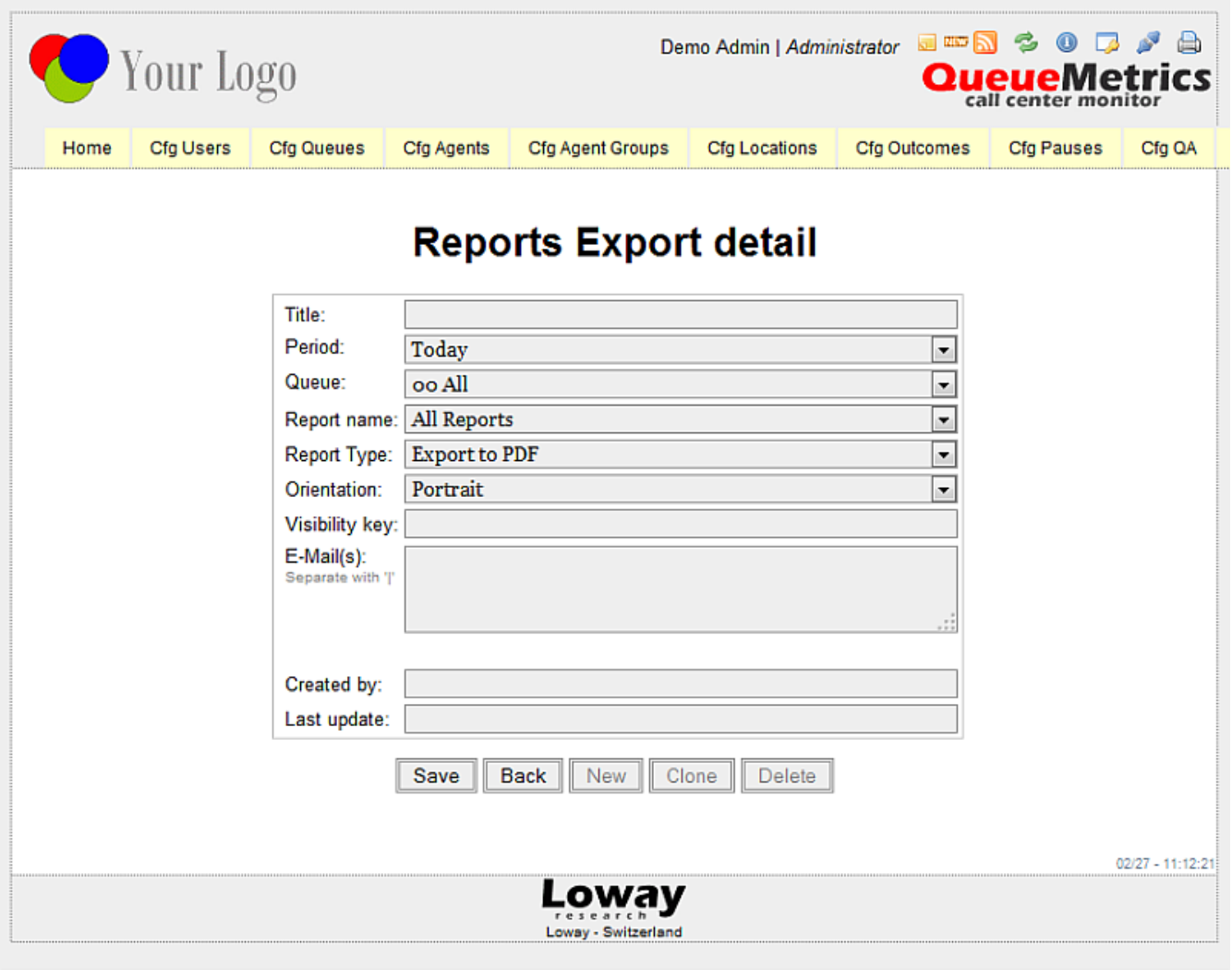
Each job is associated to a specific query period, queue and report set meaning that, each time the report is run, QueueMetrics calculates the results based on the input parameters and exports the whole report set to the required format. The generated document is sent to the recipients list specified for the job. Each job is identified by a numeric id generated by QueueMetrics and shown in the *Id* column. As defined in the chapter Generate PDF and XLS reports and send them by e-mail this id has to be used to run the report by an external ROBOT profile.

The user can edit a specific job, by clicking on the pencil icon, or can run it immediately, by clicking on the blue button shown on the list.



The user is allowed to run the report only if holding the USR_REPORTS_EXPORT key.

Each time a user creates or edits a job, the next screen will be shown:

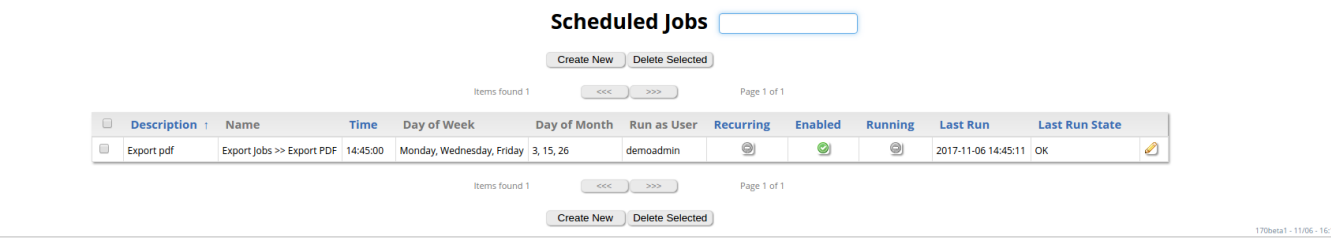


On this page the user is allowed to specify a title, needed to identify the job in the job list, and other relevant parameters like period, queue, report name (an already defined report set), report type (XLS or PDF) and the orientation (valid only for PDF documents). Each reports export job could be associated to optional visibility keys used to hide/show the job to a set of users. Finally, a list of e-mail recipients can be added - if present, on generation, the resulting report will be sent to them.

If an export job is included in a scheduled job, a clock icon will appear next to it in the export jobs list page. By clicking on the clock icon, the user will be taken to the scheduled jobs list where he can view the various scheduled jobs.

Scheduled Jobs Configuration page

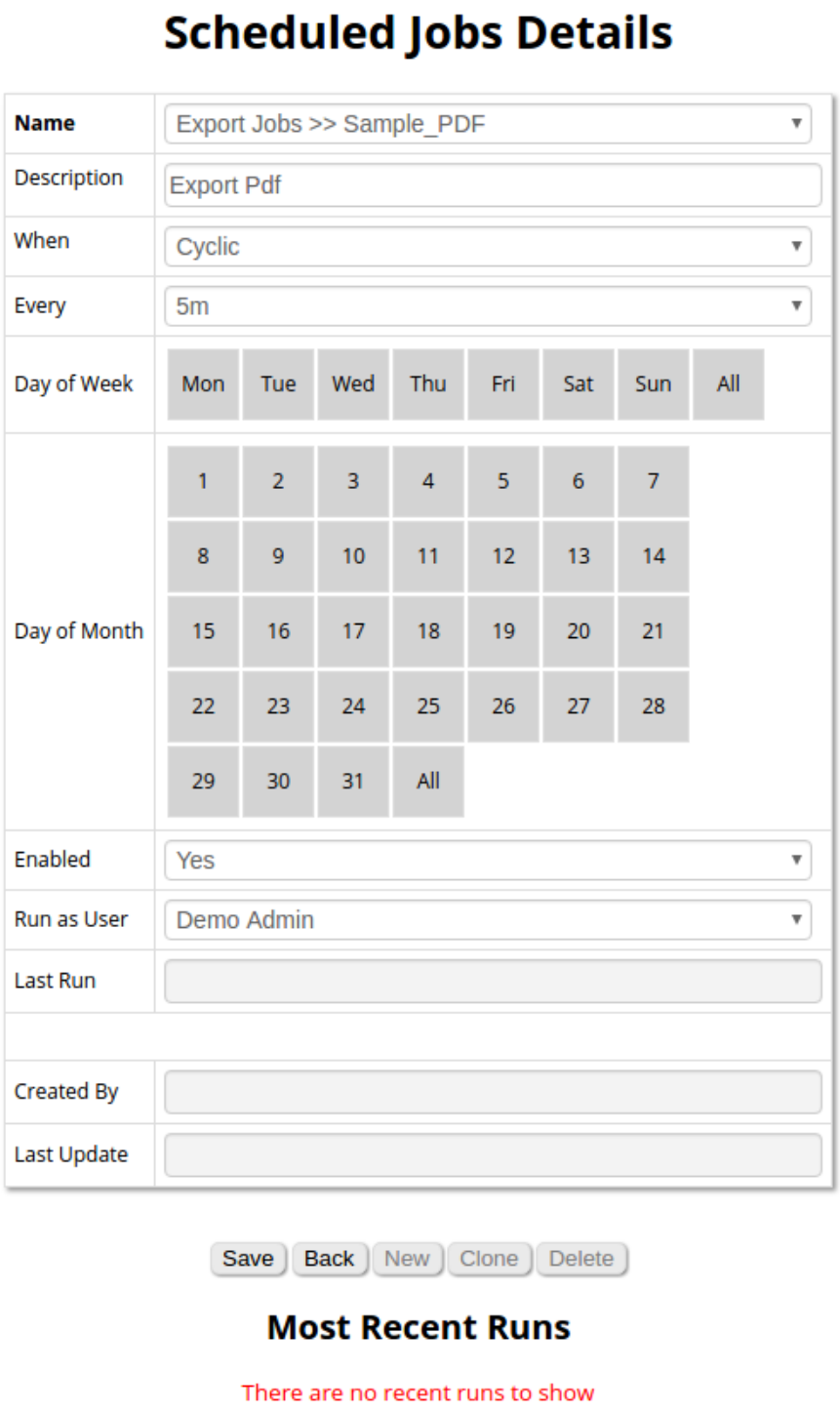
Users holding the key USR_CRONJOBS_EDIT can edit the list of available scheduled jobs.



Each Job represents a specific task, chosen between those available, that will be executed automatically at a specific point in time, or regularly at set intervals of time. The user can also specify particular days of the week or days of the month as a requirement for executing the job.

The user can edit a specific job, by clicking on the pencil icon shown on the list.

Each time a user creates or edits a Scheduled Job, the next screen will be shown:



On this page the user is allowed to choose a job type, and other relevant parameters like the time, the day requirements, and if the job is enabled or not.

One relevant parameter here is the "When" parameter. If this parameter is set to "Cyclical", the

selected "Every" value will be intended as the interval between different executions of the Job. If "When" is set to "Once a day" instead, the "Time" value is intended as the actual time of intended execution.

So for example,

- If "When" is set to "Cyclical" and "Every" is set to "5m" the job will be executed every five minutes.
- If "When" is set to "Once a Day" and "Time" is set to "00:05", the job will be executed at five minutes past midnight.

Another relevant parameter is the "Day of Week" parameter. If we select only "Monday" for example, the Job will be executed only on mondays.

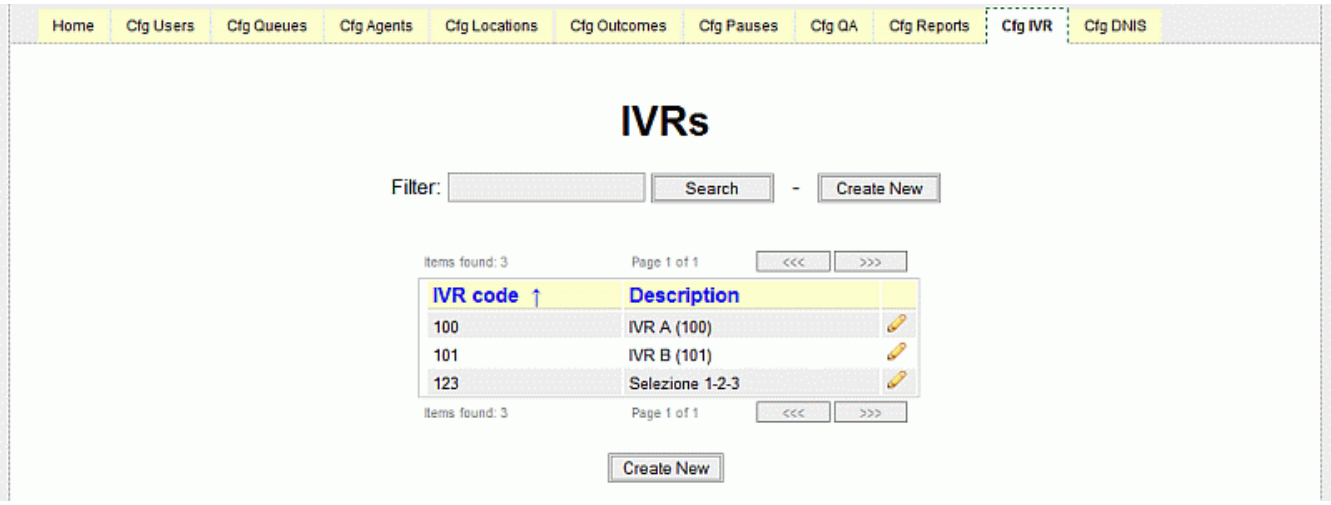
The same goes for the "Day of Month" parameter. If we choose 2,5,13 for example, our Job will only be executed on the 2nd 5th and 13th day of the month.

The "Day of Week" and "Day of Month" conditions are mutually exclusive, so if we set "Monday" and then "5", our "Monday" choice will be deleted and the scheduled job will only consider the Month Day option "5".

The "As User" parameter indicates which user will be impersonated when the Job is executed automatically. The user can only set itself as the user or, if the user is editing a Scheduled Job created by another user, the creator of the Job.

Configuring IVR and DID/DNIS names

Users holding the keys USR_IVR and USR_DNIS can edit the list of known IVR and DNIS names.



This list is used to decode the display of known IVR selections and DNIS numbers. Both configuartion pages behave the same way.



If you know that your *Support* IVR selection is 1-3-4, you could create an IVR entry of *134* that decodes to *Support (1-3-4)*. This surely makes the display easier to read.

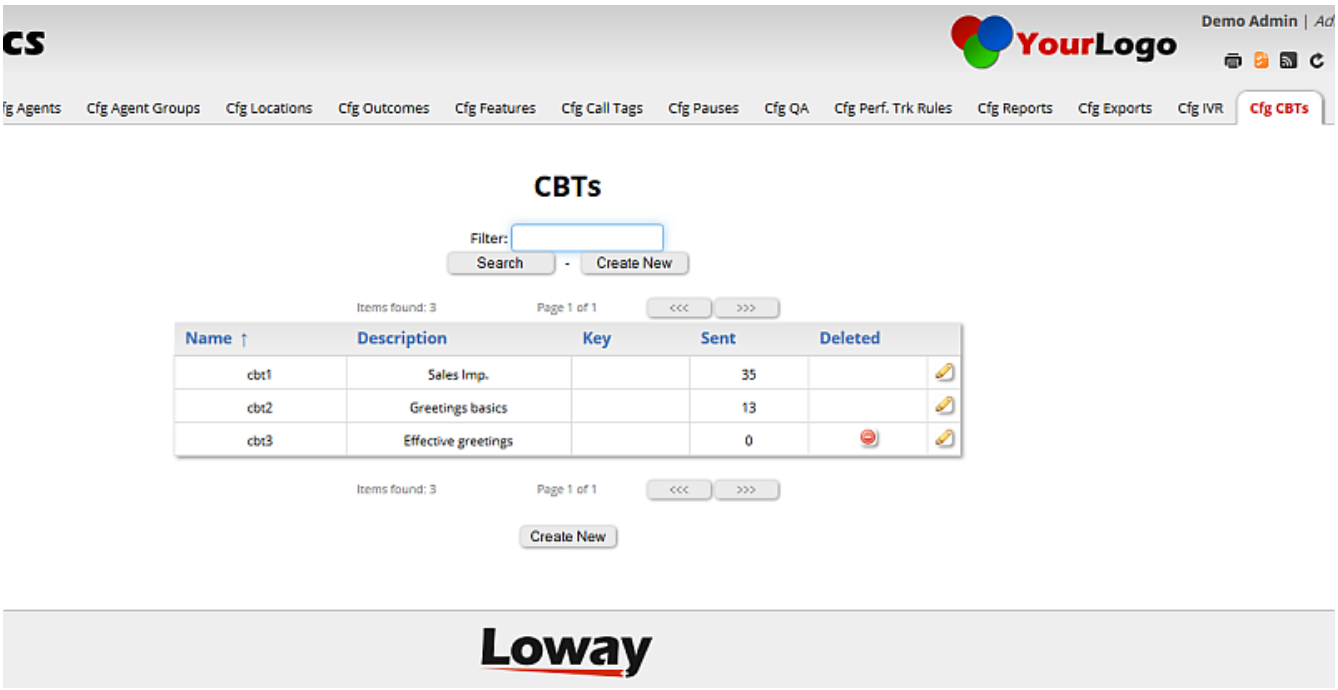
Elements that are not listed in the editor are displayed with the string they are recorded with at the

Asterisk level.

IVR entries are also used to decorate IVR menus in IVR path traversal analysis. You simply add the name of the IVR menu and the decoded value.

Configuring CBTs (Computer Based Training)

Users holding the key USR_CBTS can edit the list of known CBT URLs and names.



This list is used to populate the selection of available CBTs and to define the common URLs will be used in the task detail for each CBT sent. The URL can be used to point to any kind of material, as appropriate (e.g. PDFs, videos, interactive courses...). The URL may include a [U] token field that will be substituted with the login of the agent filling in the CBT, so that an external CBT system may keep track of why is filling it in or personalize the courseware.



If you modify the URL of an existing CBT on the list, all CBT URLs matching the same URL on the tasks tables, for all tasks not completed, are updated as well. This means that you can create a post-hoc descriptive CBTs out of URLs that are already used on the system.

Each CBT includes a read-only field that tracks the number of times the CBT was sent; this value is reported in the CBTs list. CBTs can be set active or not, so it's possible to put CBTs off-line when they become obsolete.

Configuring Cases and Clients

Clienst and Cases act as a kind of mini-CRM within QueueMetrics; they can be created by agents and supervisors, and can be easily added to calls through the Agent's page - to see how it works from there, see Clients and Cases in the Agent's page.

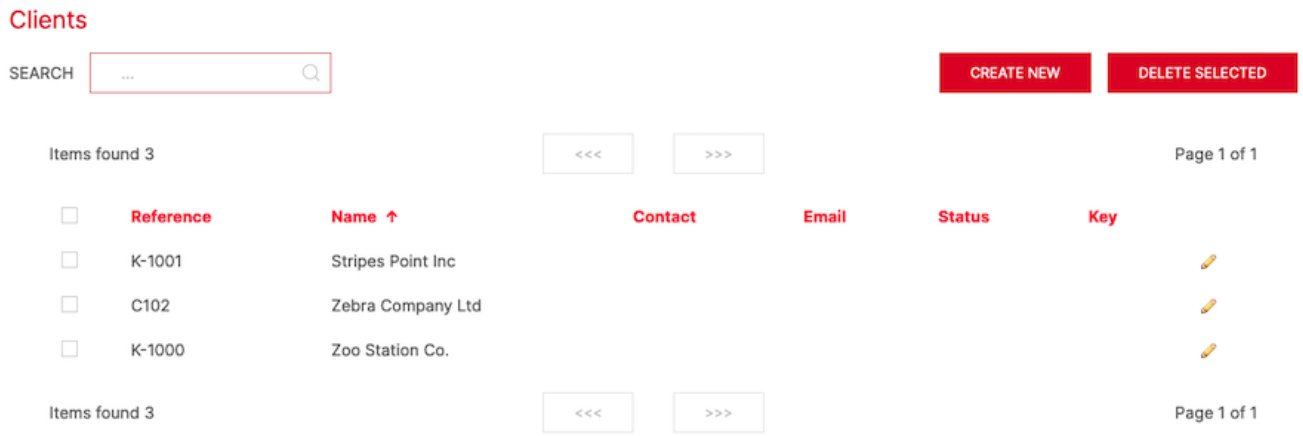
They share a set of common principles:

- Both have:
  - A **title** - that is mandatory
  - An **external reference**, that is a client-id or a case-id as used in your company. If you provide none, then QueueMetrics will create one for you. But if you have a different identifier, you are free to use it instead. The external reference must be unique, though you can change it.
  - A **status**, that can be `Open` or `Closed`. Only items with an open status can be used to be linked to calls or, in the case of clients, to new cases. You are generally not allowed to delete elements that are in use.
  - A **security key** that determines which agents can select it. If blank, all agents can see it.
  - A number of other fields, that you can use as you best see fit (email, notes, contacts etc).
- You can use both Clients and Cases, or just Clients or Cases, depending on your preferred workflow.
- A Case has an implied Client, though this can be overridden on a call-by-call basis
- Both Cases and Clients are stored, though their external references, on the `queue_log` file along with other call data

Both Cases and Clients will appear in reports, under data block show in Historical reports - Cases and Clients.

Configuring Clients

To configure clients, you must hold the key `USR_CLIENTS_EDIT`.



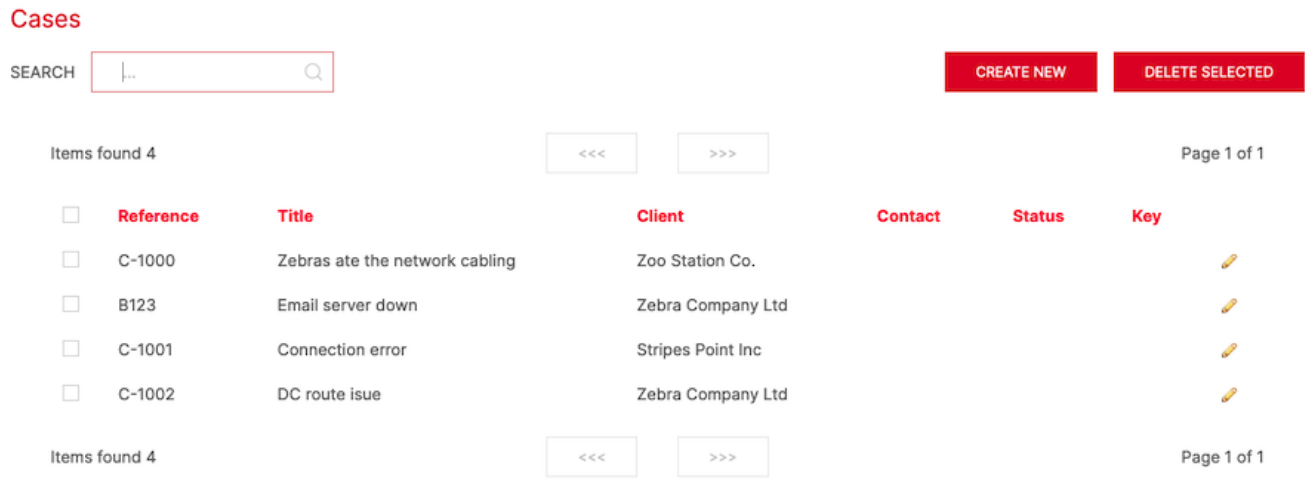
The editor works in a similar way to all other editors. If you leave the field **Reference** blank, a new reference will be created for you.

The default format for clients' external references that QueueMetrics generates for you is `K-1234`. But you are free to use your own, as long as they are at least three characters long.

Under the details of a client, there is a list of recent cases that this client is referenced to. For each case, you can see their reference, the title and whether it is open or closed.

Configuring Cases

To configure cases, you must hold the key `USR_CASES_EDIT`.



The editor works in a similar way to all other editors. If you leave the field **Reference** blank, a new reference will be created for you.

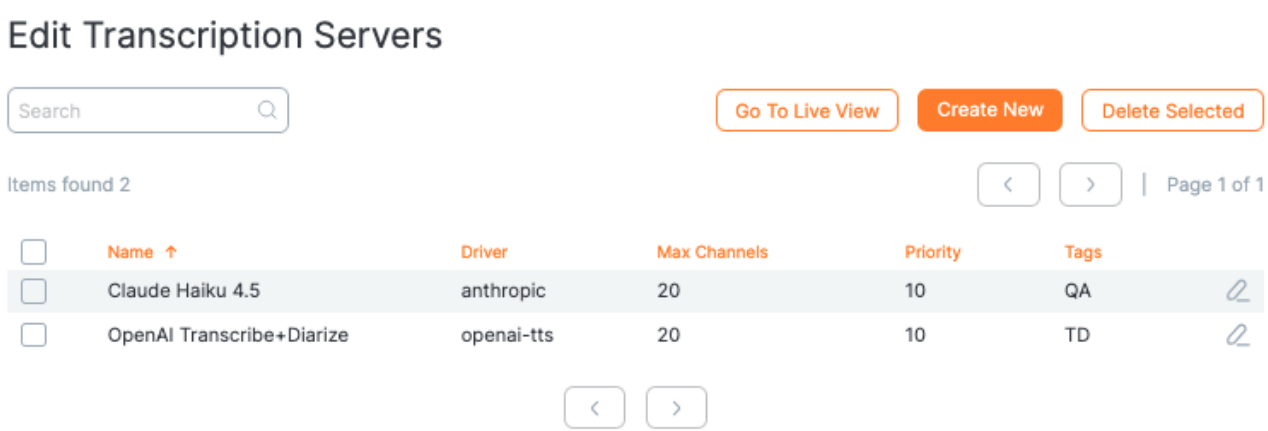
The default format for cases' external references that QueueMetrics generates for you is `C-1234`. But you are free to use your own, as long as they are at least three characters long.

When creating a case, you must link it to a client. Of course, a client may have multiple cases linked to it. In general, if there is only case information for a call, QM assumes that the referenced client is the case's. But the agent can override this setting, so that a call may appear under a case and under a client that is not the case's own client, by manually setting a client on a call where a case is set.

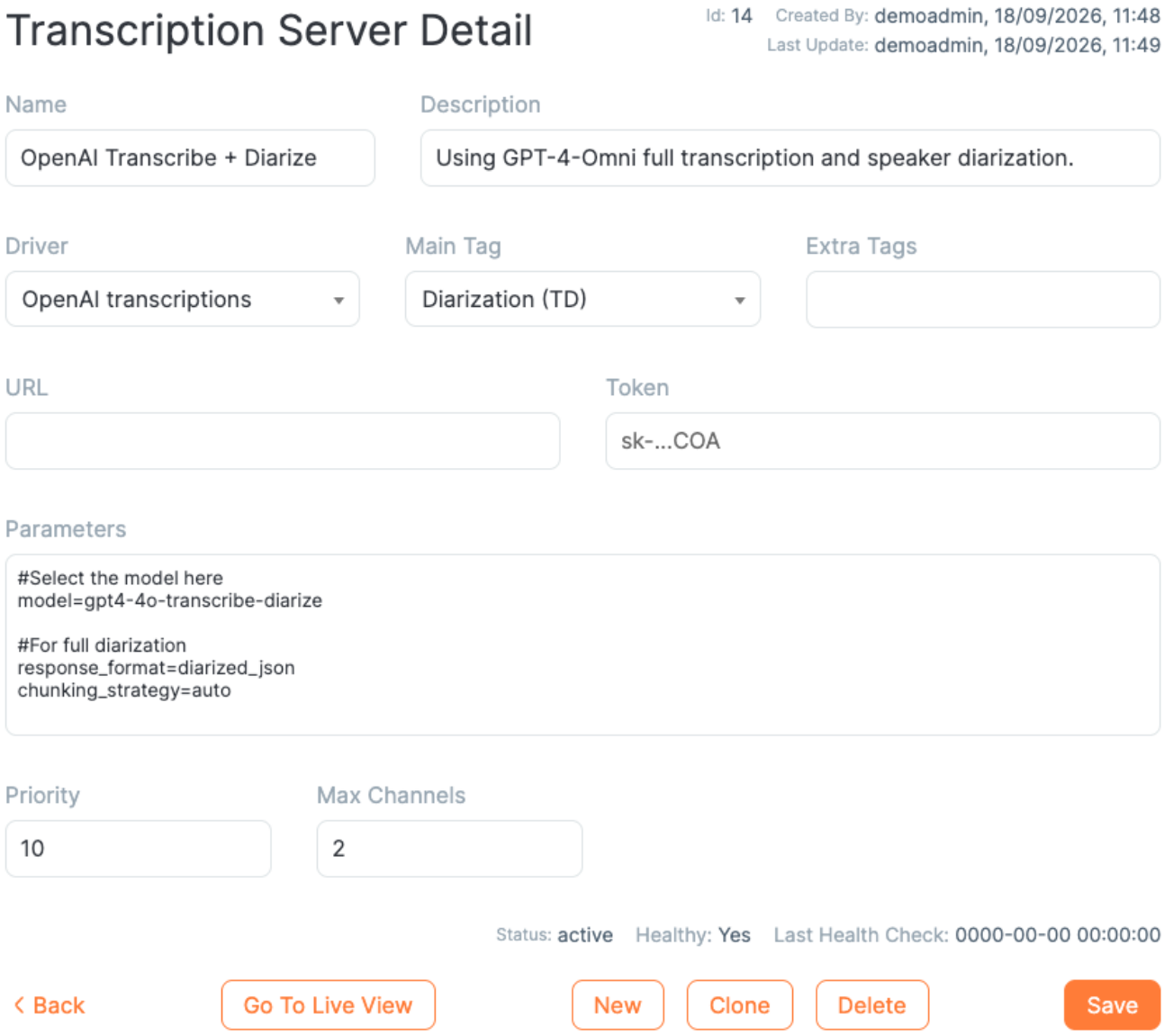
Under the details of a client, a list of all activity for the case is displayed. This consists of all agents linking the case to some call, or adding comments to it.

Configuring Transcription Servers

Transcription servers are the pool of transcription and AI-grading engines that QueueMetrics dispatches work to; see Setting up the server pool for how the pool and its channels are used at runtime. This page lists the transcription servers currently configured and provides access to their detail editor. The editor requires the `TRANS_SERVER` security key.



The list shows, for each server, its name, driver, maximum number of channels, priority and tags.



The detail editor exposes the following fields:

- **Name**: a unique name identifying the server.
- **Driver**: the transcription or LLM engine this server represents. See Transcription drivers for the list of supported drivers and the requirements specific to each of them.
- **URL**: the address QueueMetrics contacts to reach the service. Whether this field is required depends on the selected driver.
- **Token**: the access credential for the service. The field is rendered as a password input; once a token has been saved, the field only displays a masked hint of the stored value (its first and last characters), and leaving it blank on a subsequent save preserves the token already on record. Whether this field is required depends on the selected driver.
- **Priority**: a number used, together with the number of free channels, to decide which server is preferred when more than one server can handle a given kind of work. Lower values are given precedence. Defaults to `10` for a new server.
- **Main Tag**: the kind of work this server accepts (`TR`, `TD`, `QA`),

- **Extra Tags**: Any additional tags that Uniloader uses to tag this server, written as free text, conventionally as a comma-separated list. Matching against a requested tag is performed by substring search, not by parsing the field into discrete values. See Setting up the server pool for the meaning of each tag.
- **Description**: a free-text description of the server.
- **Parameters**: further, driver-specific configuration, entered as one `key=value` pair per line. Blank lines and lines starting with `#` are treated as comments and ignored. A value starting with `{` is parsed as a JSON object rather than taken as a literal string, so that it is possible to enter a complete JSON object if the upstream server requires it. Which keys are recognised, and their meaning, depend on the selected driver; see Transcription drivers.
- **Max Channels**: the maximum number of operations this server can run in parallel. Defaults to `2` for a new server.
- **Status**: a read-only field reporting whether the server is currently active or paused. It is not changed from this editor - see Monitoring for how a server is paused and resumed.
- **Healthy**: a read-only field reporting whether the server is currently seen as reachable and functioning. It is updated from health notifications sent by Uniloader, but not every driver sends them.
- **Last Health Check**: a read-only field reporting the time of the last health notification received for this server from Uniloader. As with **Healthy**, it is only updated for drivers that send health notifications.

A server can be created, cloned or deleted from its detail editor like any other configuration record. The *Go to live view* link opens the live monitoring page for the pool, described in Setting up the server pool.

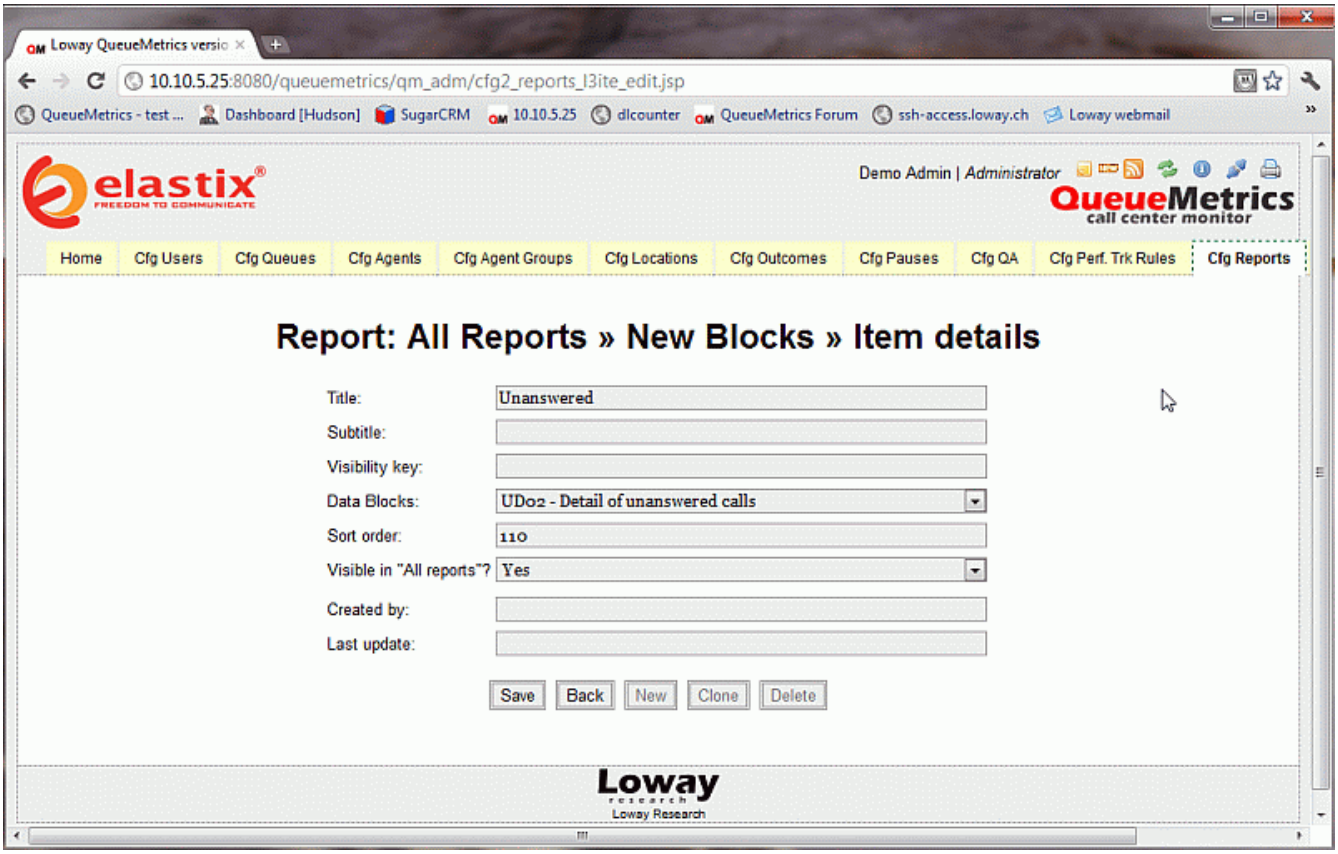
Configuring paged call lists

It is possible to view details of calls (answered, unanswered) in a paginated order, rather than as a long list of data on a single page, allowing better readability when running a large result set.

To set this up, from the Home page you select "Edit Reports" which leads to the Configure QueueMetrics reports" page (Cfg Reports tab). Click on the "All reports" link and you will be shown all current reports. At the bottom of the screen select "Create New".

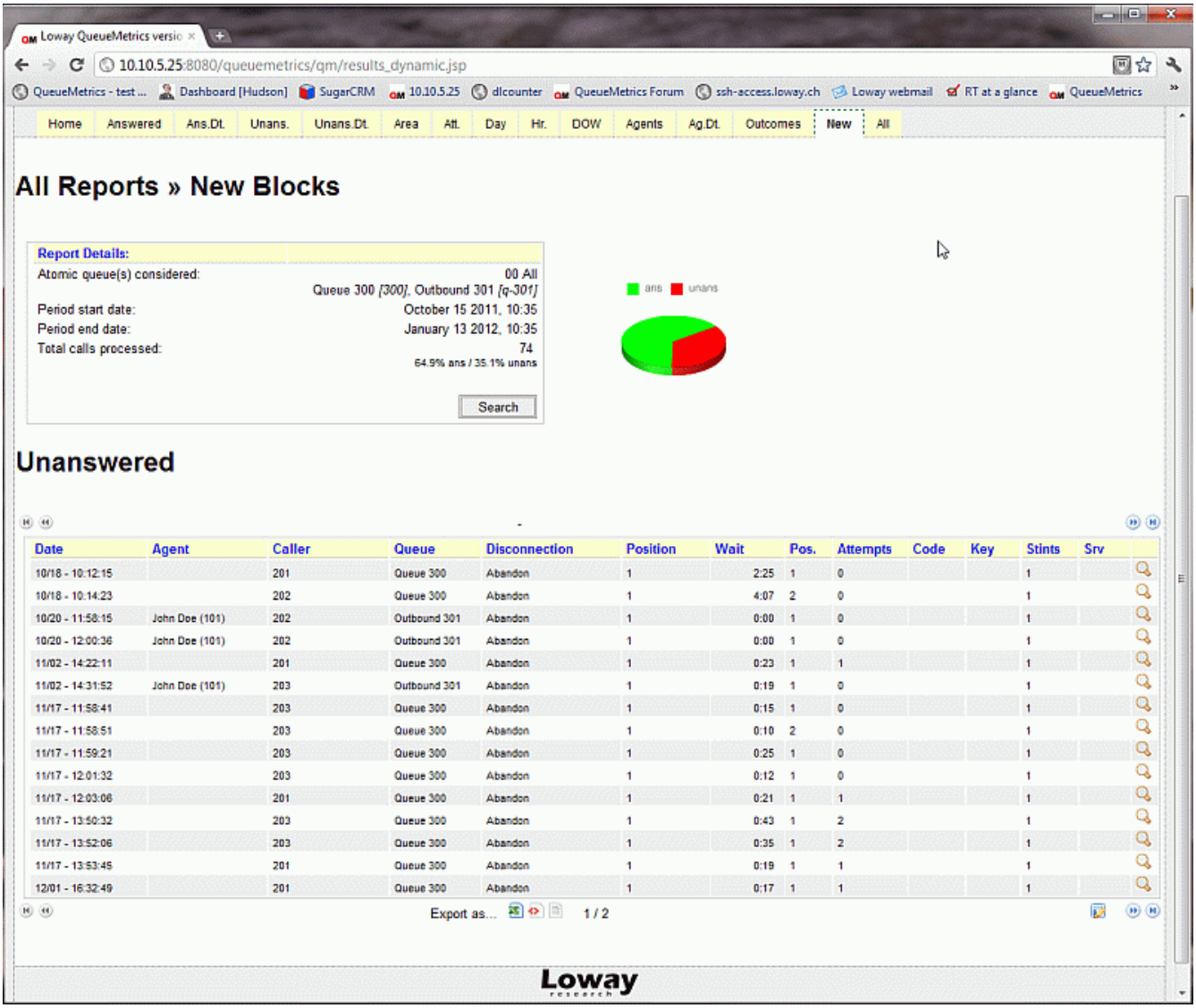
Enter a Short Name such as "New" and a Title such as "New Blocks" - you can choose different Name/Title if necessary. Also, the Visible in All Reports field should be set to "Yes". Now Save it.

Go back and select "New Blocks" which leads to the "Report: All Reports >> New Blocks >> Items" screen. Select "Create New".

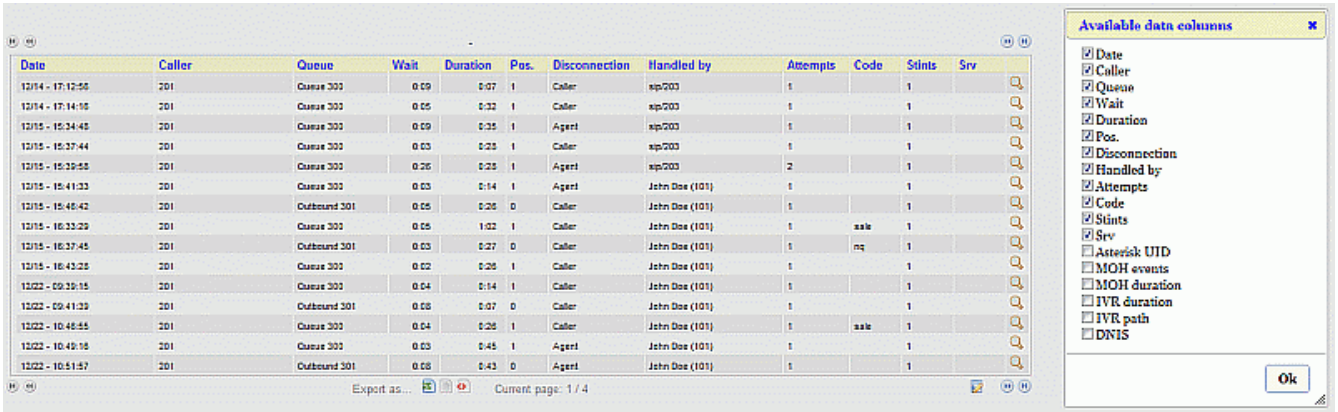


Save it and select "Back" - you will now see the newly-created item.

If we run a Report we will see a new tab called "New", as shown below, where the calls are showing in pages, rather than as a listing (note the buttons to go forward/back and that the page is 1 of 2 pages)



In the image above, on the bottom right of the paginated list we can see a small icon just before the "next page" buttons. this icon allows us to add or remove columns within the displayed paginated listing. It is now possible to add the Music on Hold (MOH) columns "MOH events" and "MOH duration" which display the number of events where a caller was put on Hold with music and the total duration of such events. You can also add the columns "IVR duration/IVR path" which displays the time the caller spent within the IVR selections and the choices the caller made while going through the process. Adding the "DNIS" column allows to display the number dialled to reach the queue.



Configuring the new Real-time Agents Page

Configuring the new realtime page involves to set some configuration keys that are inherited from the old agent realtime page and a new set of keys specifically designed to operate with the new realtime page. This chapter summarize the most relevant keys needed to properly configure the new realtime page and the integrated softphone. For more details on each key, please refer to the Appendix C: System preferences

Enable/Disable the hotdesking:

```
default.hotdesking=86400
```

A value different than zero enables the hotdesking.

Defines a custom CRM page that will be open on each new taken call:

```
# Asterisk Unique ID visibility and related callback URL associated to the icon
default.crmapp=http://server/app?agent=[A]&unique=[U]&server=[S]&queue=[Q]&tst=[T]&cid
=[C]&dnis=[D]&ivr=[I]&outcome=[O]&position=[P]&attempts=[M]
default.crmlabel=CRM Page
default.showAstClid=true
realtime.agent_autoopenurl=true
```

Defines two other optional custom backgrounds:

```
# Custom web pages for the realtime agent page
realtime.agent_web1_url=http://www.queuemetrics.com/?agent=[A]
realtime.agent_web1_label=QueueMetrics
realtime.agent_web2_url=http://www.loway.ch/?agent=[a]
realtime.agent_web2_label=Loway
```

Optional backgrounds can be opened as popups by setting the property agent.openUrlInPopup to true in the Edit System Parameters page.

Custom background pages can also expand a set of placeholders that carry useful information.

Placeholder	**Meaning**
[A]	The agent's numeric code, e.g. *101* for *Agent/101*
[X]	The agent's extension, e.g. 300
[U]	Unique
[S]	Server
[Q]	Queue
[T]	tst

Placeholder	Meaning
[C]	cid
[D]	dnis
[I]	IVR
[O]	Outcome
[P]	Position
[M]	Attempts

If CRM and/or custom backgrounds are not needed, it's possible to override the default background changing the key:

```
realtime.agent_background_url=http://serverurl/backgroundpage.htm
```

In order to use the integrated softphone, the steps required are: 1. Configure the SIP server and the WebRTC URL 2. Add, for each agent, a SIP username, password and SIP Realm authentication.

Depending on our setup (normal or clustered mode), the SIP server and the WebRTC urls are set by changing the below configuration keys. For a not clustered QueueMetrics install:

```
# Default keys for the sofphone definition
default.sipaddress=10.10.5.49
default.websocketurl=ws://10.10.5.49:8088/ws
default.rtcWebBreaker=true
```

For a clustered QueueMetrics install, be sure you're setting the following keys for each server in your setup:

```
cluster.serverX.sipaddress=10.10.5.49
cluster.serverX.websocketurl=ws://10.10.5.49:8088/ws
cluster.serverX.rtcWebBreaker=true

where serverX is the name of each server defined in the cluster configuration key.
```

Then, for each agent, set the SIP username, password and SIP Realm authentication through the Agent Configuration Pages, as explained in Configuring agents

The Logon Panel allows to pre-validate the agent code and the agent extension needed by the login/logout process. In order to validate these codes, a two set of keys have to be set:

```
# The regexp to validate agent codes
realtime.agentRegexp=1\d\d

#A regexp to validate extension codes
```

```
realtime.extensionRegexp=\d\d\d
```

The Logon Panel allows to mark as read only the agent code field. This could be accomplished by setting the following key:

```
default.lockedAgentPopupCode=true
```

The list of available queues for each agent is defined by the following key:

```
realtime.dynamicLoginQueues=all
```

Relevant values for this key are:

- *assigned* or *registered*: the list of available queues shows only the queues where the agent plays a defined role (i.e. is a main/spill/wrap for these queues)
- *all*: the list of available queues shows all the defined queues, without taking care of the agent role (but filtered by visibility key)

Agents can insert pauses through the Pauses panel. Pauses without a specified pause codes can be rejected by setting the following key:

```
default.pausecoderequired=true
```

Agents can grade taken calls only if they hold the QA_TRACK key.

QueueMetrics configuration wizard

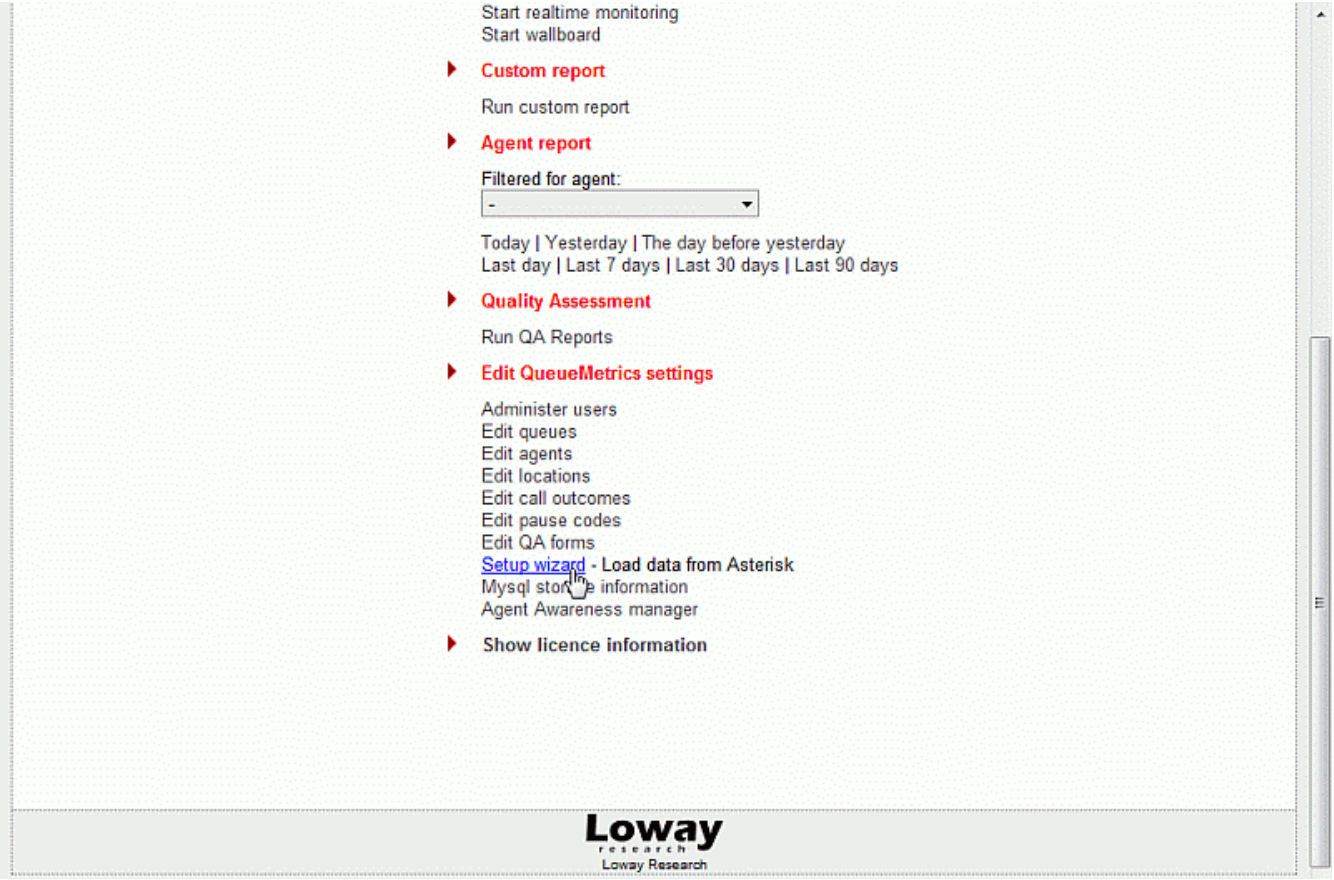
In order to save time and make sure that QM is always up-to-date with the underlying Asterisk configuration, it is possible to run a wizard that will load the following data straight from Asterisk configuration files:

- 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, edit agents, and must hold the CONFIG key 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 the first step of the wizard.

At the top of this page is a dropdown menu that defines where asterisk configuration could be found. Actually the wizard is able to read information from:

- File
- Single machine Asterisk Manager Interface
- Clustered machines through Asterisk Manager Interface
- Asterisk realtime database
- Asterisk queue log file

By selecting the "File" source, the three edit boxes will let the administrator able to specify the local paths for the agents.conf, queues.conf and the users.conf file.

By selecting "Queue Log File" as source, the associated edit box will let the administrator able to specify the local path for the queue_log file.



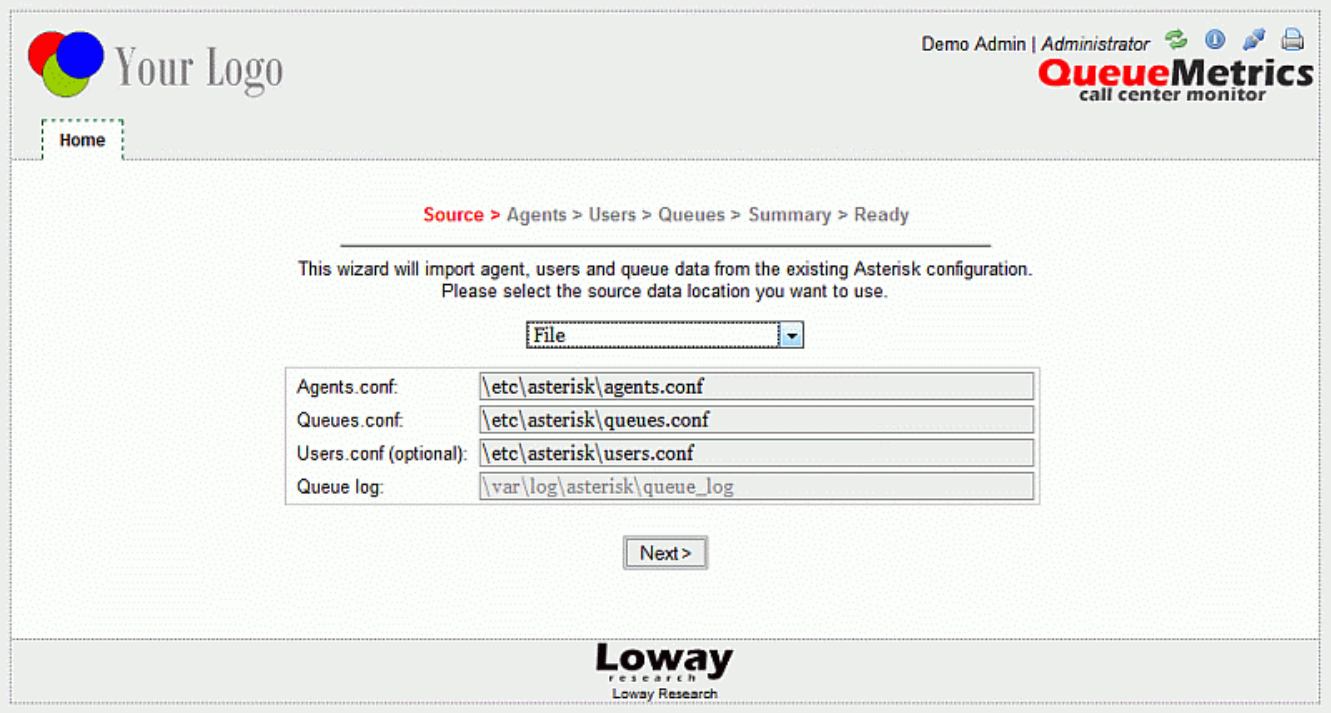
The users.conf file is optional and could integrate the information stored in the agents.conf file. The agents.conf file, instead, is not required only if the users.conf is present.



If you don’t have the users.conf or the agents.conf file, you can leave in the edit boxes their default values and the system will be able to skip it if not found.

For sources different than "File", or "Queue Log File", the wizard will use some configuration

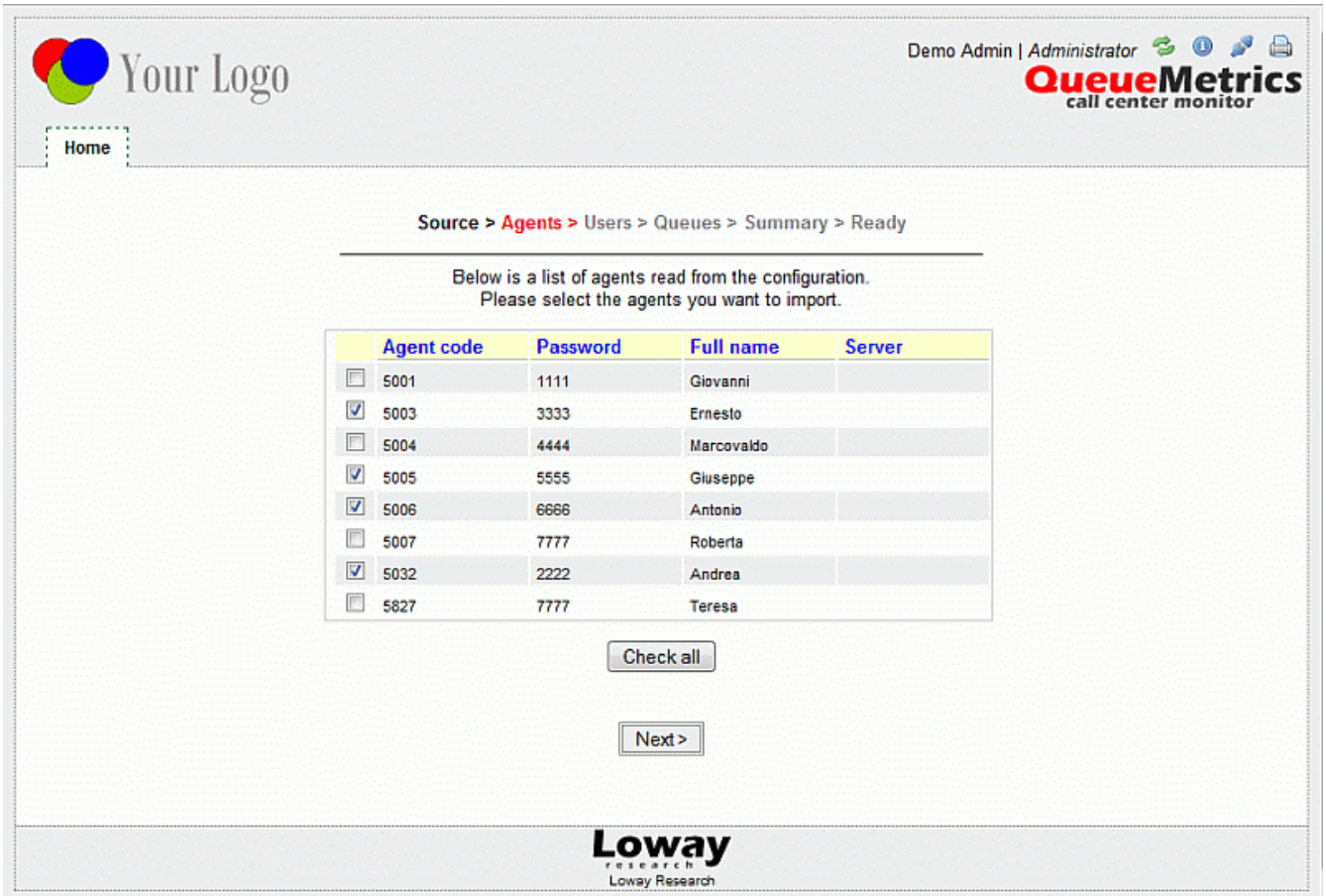
options to know how to reach the required information. More details can be found in Configuring system preferences and in Appendix C: System preferences.



When you have selected the source you want to be read, click on *Next* button. You will be redirected on the validation page. This page will inform you if the provided sources were successfully read or, in the worst case, it will show you a message reporting an explanation of the error found.

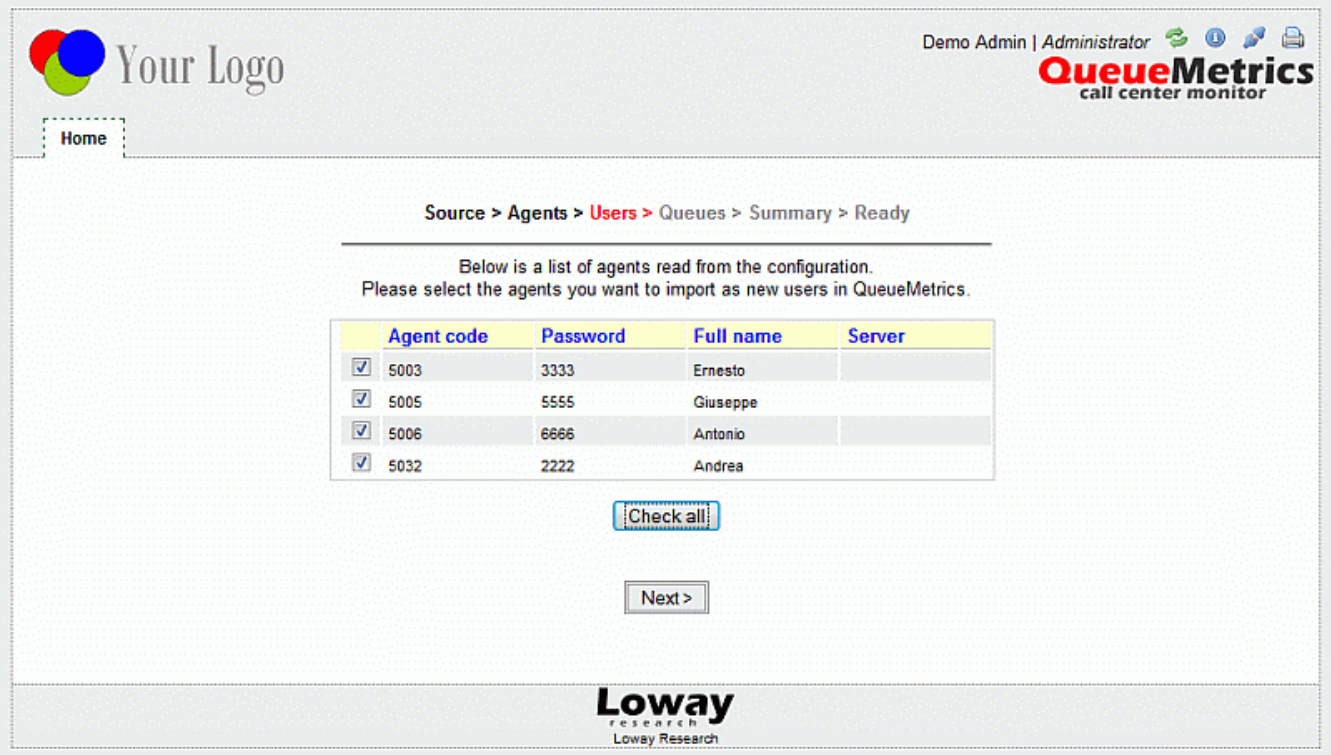


If the validation fails, clicking on *Next* button you'll be forwarded back to the first step, otherwise, you'll be redirected to the next step.



The wizard will scan the available agents and presents 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.

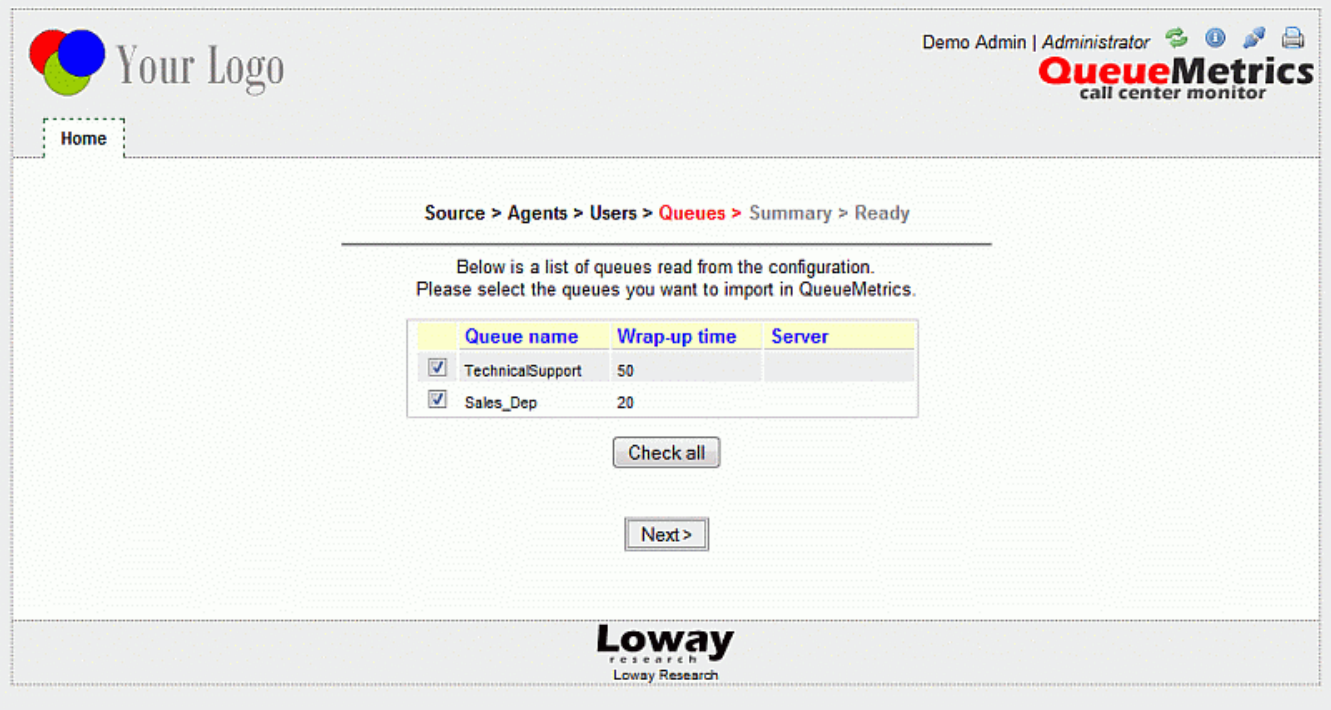
If no agents will be selected, by clicking on *Next* button the wizard will skip the next step and will forward you directly to the queue selection step. If at least one agent was selected, instead, when you click on *Next* button you'll be redirected to the window shown below.



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

Please note that if the wizard is not able to read the password associated to a specific user (because the password is not specified in the configuration files or because the wizard is reading information from AMI or realtime, or the queue log file, where password for agents are not shown) it will use the following rules:

- For each new user added, a default password will be forced to be equal to their agent code
- For each user to be updated (i.e. already present in the QueueMetrics database) a default password will be shown in the mask but it will never used to overwrite the already present one.



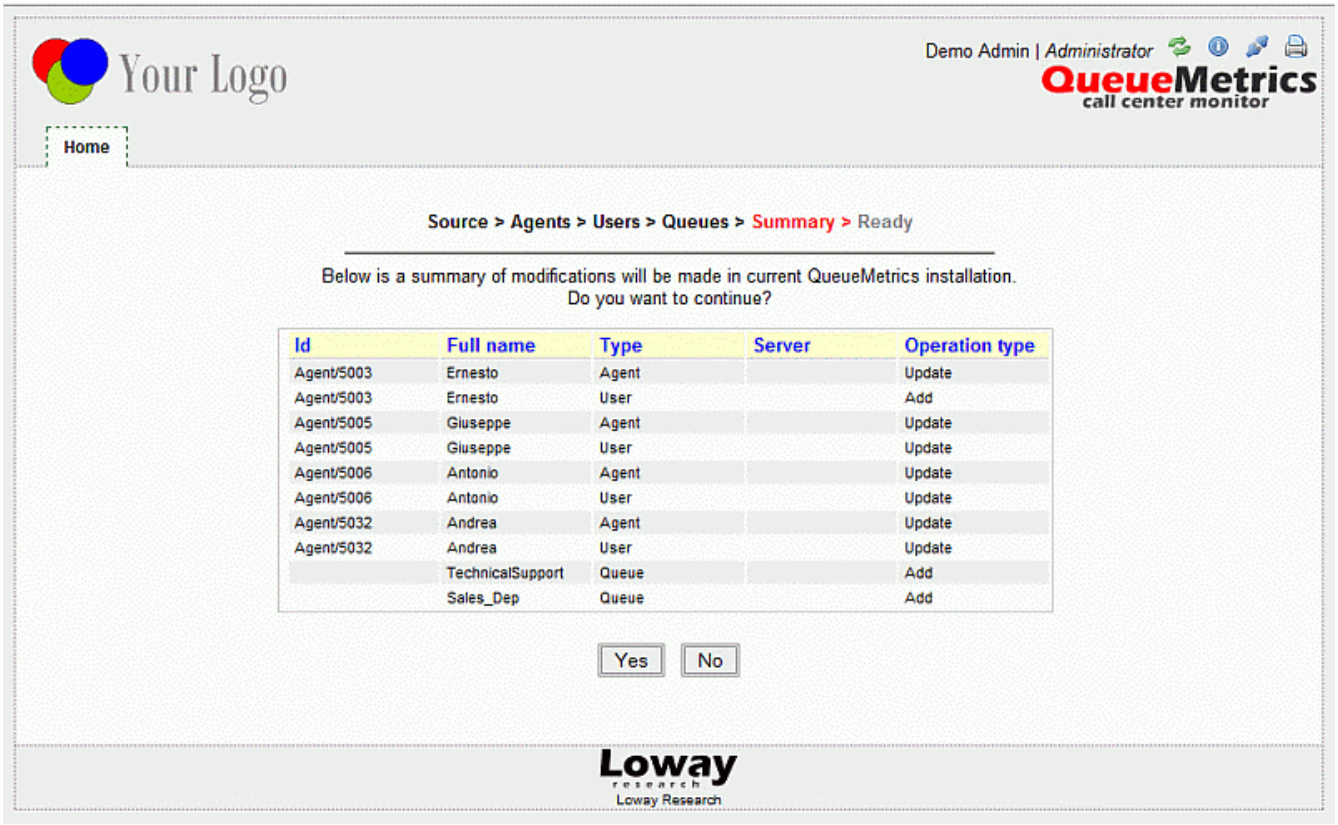
The queues will be created or updated as needed; existing queues will not usually be overwritten without explicit user permission.



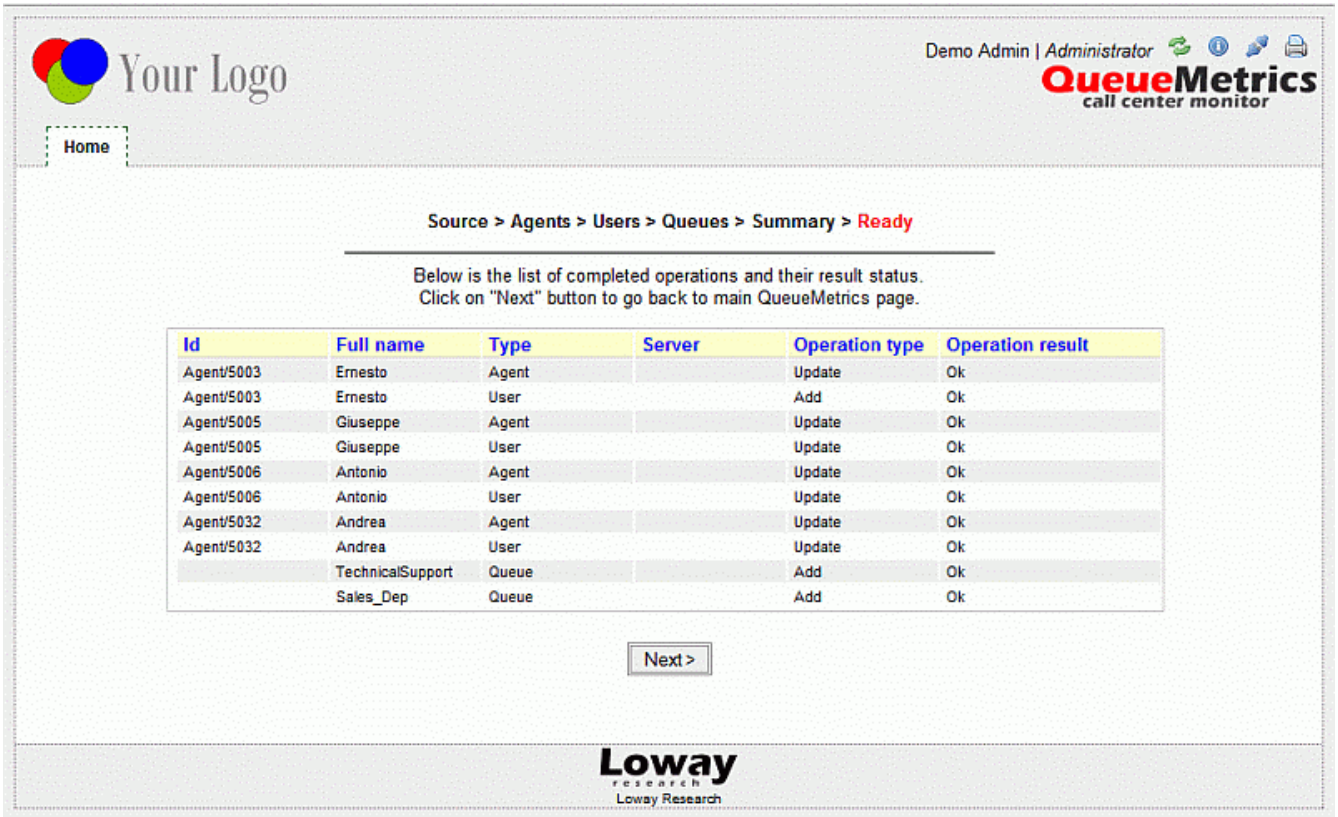
A queue will be automatically checked to be updated if at least one of its agent member was selected to be updated and/or added.



When updating a queue, the spilloff and queue members lists will be generated looking at the penalties associated to the agents read from the sources. If an agent is already present in a member or spilloff list, but it was not selected to be updated, he will not removed and/or moved from any list.



If you click to the *Next* button you'll be redirected to the page above reported. This page will display a summary of the QueueMetrics database updates that have been scheduled to be performed. Clicking on the *Yes* button, the scheduled actions will be run and you will be redirected to the last page where a table listing the related operations results will be presented. Clicking on *No* button, instead, you'll be forwarded back to the first wizard step.



The QueueMetrics database is now updated with the information found in the selected sources. You can go back to the home page clicking on *Next* button.

Unattended QueueMetrics configuration and update

QueueMetrics could be updated and configured by means of external http queries made in a known format. This is really interesting for setting up a cron job to be completed sometimes during the day. When QueueMetrics receives external http queries, it will perform all the configuration wizard steps together (see QueueMetrics configuration wizard) assuming default answers. This will result in a background synchronization between your asterisk boxes and the QueueMetrics database. To be able to run periodic QueueMetrics update, you need:

- A QueueMetrics user holding the CONFIG key
- A command line script able to perform http queries

The URL to be used to start the unattended configuration system has to be formatted as follow:

```
http://qmaddress/queuemetrics/autoconf_Robot.do?user=demoadmin&pass=demo \
 &stype=0&agents=/etc/agenti.conf&queues=/etc/code.conf&users=/etc/users.conf
```

The meaning of specified parameters is below reported:

- *user*: the username to be responsible for the update process
- *pass*: the password associated to specified username
- *stype*: defines what type of source you want to use and it could assume the following values:

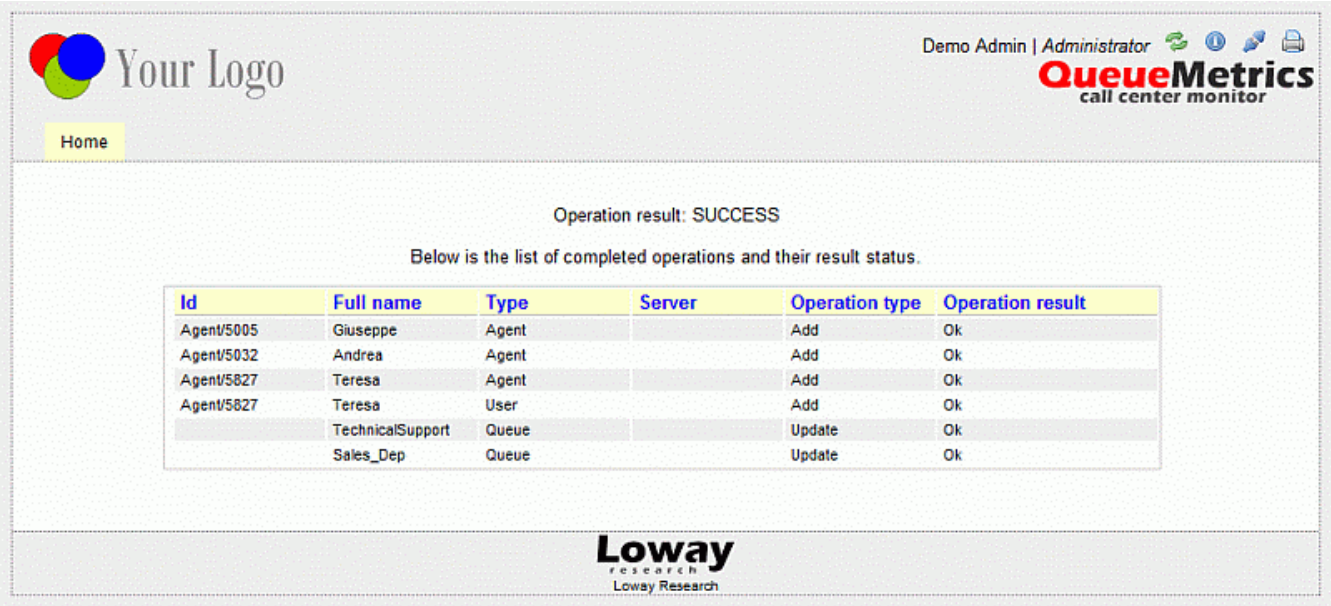
- **0**: File. If no other parameters were specified, the wizard will read the files defined in the default configuration.
- **1**: Single Machine Asterisk Manager Interface. The wizard will read information from the machine specified in callfile.dir key.
- **2**: Clustered Machines Asterisk Manager Interface. The wizard will read information from the machines specified in the standard cluster definition.
- **3**: Asterisk realtime. The wizard will read information from the database specified in the configuration.
- **4**: Asterisk queue log file. The wizard will read information from the provided queue log file.

The *user*, *pass* and *stype* are mandatory; the other parameters are optional and have no meaning when the requested source is different from "file".

The other parameters are:

- *agents*: specifies the asterisk agents configuration file (and it's read only when the "File" source is specified)
- *queues*: specifies the asterisk queues configuration file (and it's read only when the "File" source is specified)
- *users*: specifies the asterisk users configuration file (and it's read only when the "File" source is specified)
- *qlog*: specifies the asterisk queue log file (and it's read only when the "Asterisk Queue Log" source is specified)

When QueueMetrics terminates the procedure, it will answer with a result page where the term "SUCCESS", or "FAIL", will be present reflecting the operation success status. In this page will be also present a list of the performed operation (and their result status). An example page is reported below:



Configuring system preferences

System preferences are stored on a database table, and can be edited either though a guided transaction or through a free-form page. You can check the current set of system preferences from the Using the DbTest Diagnostic Tools page.

A complete list of preferences recognised by QM can be found in the chapter Appendix C: System preferences.

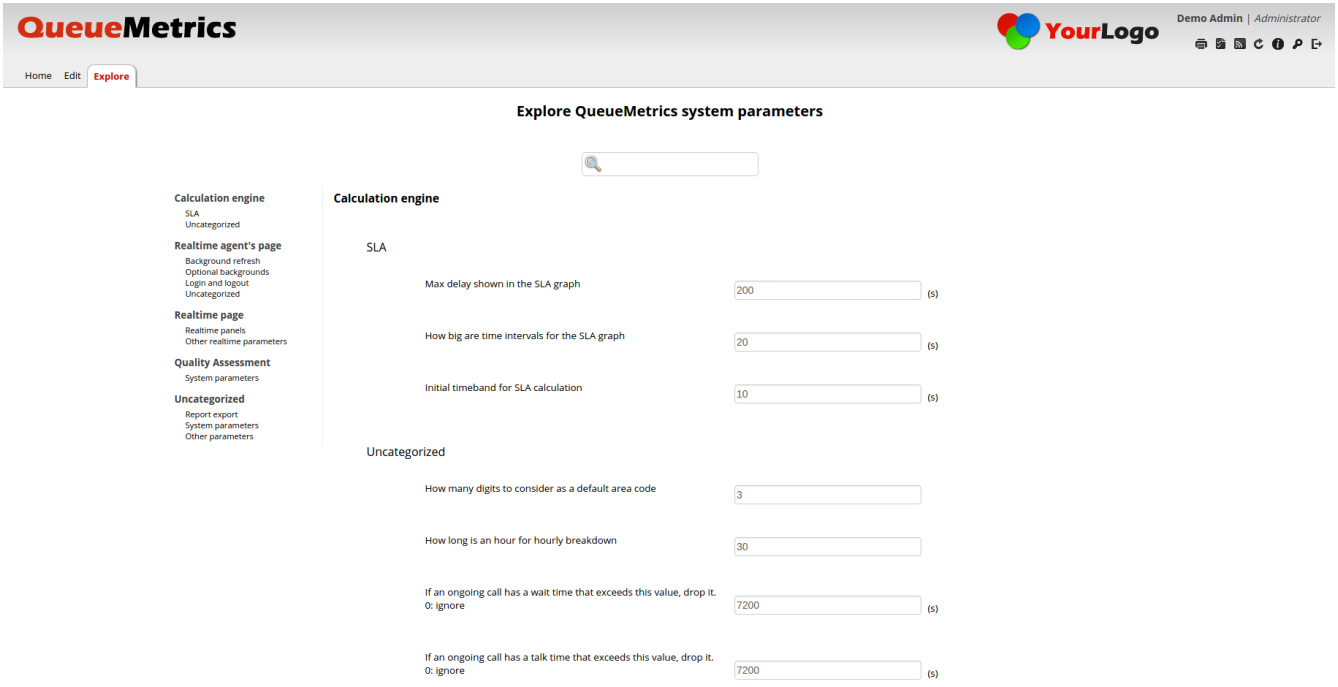
Once a preferences is changed, it is enough for the user to log off and log on again; restarting the servlet container is not needed.



These properties used to be held in a file called `configuration.properties`, and older documentation may still advise to modify it directly. This is not true any longer – the file is now unused and all configuration is stored on the database. If you need to make a change, use the forms described here.

Explore system parameters

The esiest way to set preferences is though the "Explore system parameters" page under the Administrative tools, available to users holding the `EDIT_CFG` security key.



This page lists the most usefull configuration options and allows an administrator to change their values through a set of gui elements. Allowed values are already listed and default values are marked with an asterisk. Through the search box located on the top center side is possible to limit the number of shown options to easily focus on the wanted options.

The system Parameters can be filtered by value, by the parameter description or by the actual parameter name. When searching for a parameter, QueueMetrics will filter out all the non-matching parametrs, displaying only the ones relevant to the research.

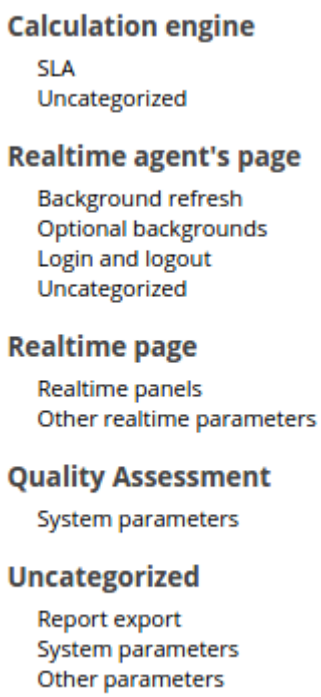
The various settings are organized in a hierarchical fashion, with all the keys divided in categories and features. Categories are the most comprehensive groupings, they are written in bold and represent the different aspects of your QueueMetrics system. Each category is then divided in Features resembling the different functionalities that belong to a particular category.

At the bottom of this hierarchy, we find the actual parameter keys and their linked current values.

There are three different types of Parameters:

- Multiple Choice Values: this type of parameter key can only accept a specific list of values, these will be listed in a drop-down menu.
- Numeric Values: this type of parameter key can only accept numerical values, sometimes even within a certain specific range.
- Free-Text Values: this type of parameter key can accept any value.

On the left side of the screen there is the System Parameters Sidebar. This sidebar is an overview of all the Categories and Features in the page. To jump to a specific category one can simply click on one of the different category names.



Edit system parameters

This page is also available to users holding the `EDIT_CFG` key, and lets you edit freely the contents of the configuration. This way, you can set any property you like.



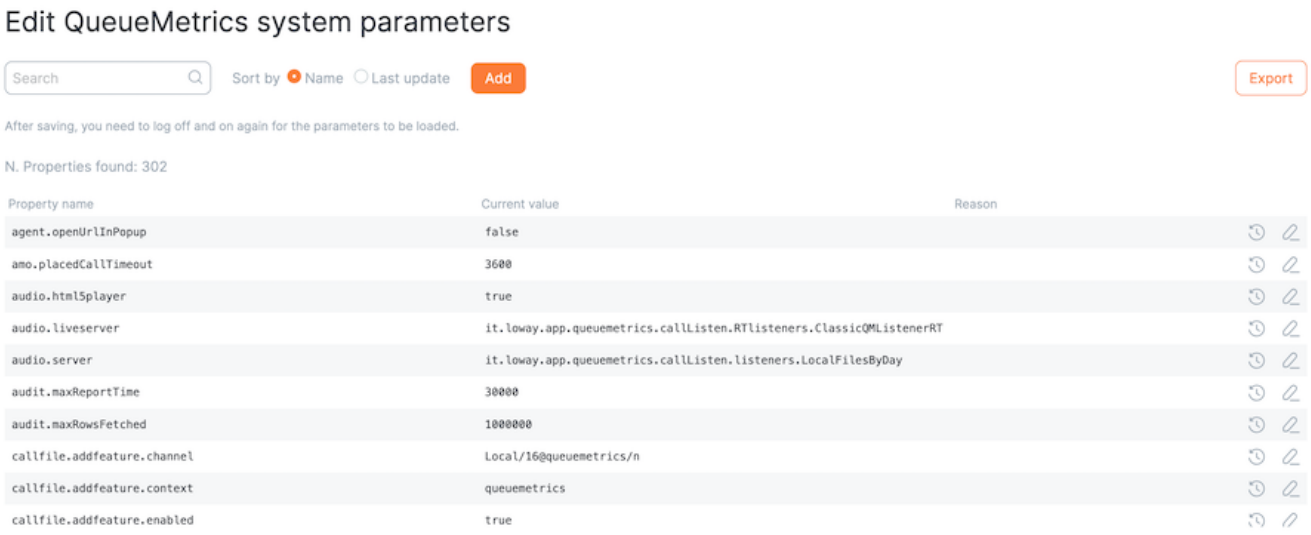
Configuration properties are not structured – you can basically enter any property as you like. There is no check that they make any sense when you do; any unknown property will be silently ignored. That's why it's usually better to use the

"Explore system parameters" transaction shown above.



A complete set of configuration properties, written as a big text file with comments, can be found at Example configuration properties.

When you access the page, you can see a list of properties with their current values. You can search within properties, their names and comments by entering substring in the box on the top left.



It is possible to display the properties, either in alphabetical order, or in reverse order of last update – so that the most recent changes appear on top.

For each property, you can see its history by clicking on the "time" icon. It shows:

– the current value, the time when this was changed or created, and the comment (if any) that was saved with it - a history of preview modifications, showing previous values and who did them

Editing properties

To edit an existing property, the easiest thing to do is to click on the "pencil" icon and make your changes in the window. The current value will be shown, and you can edit it. If you want to add other properties as well, you may add them within the same editing window on separate lines. You may also enter a comment for your changes.

If you want to add a block of properties at once (as you often find on the manual or blogs) the easiest thging is to click on the "Add" button and paste them there.

There is usually no need to delete an existing property, unless you made a typo and so the property just won't go away. QM won't care, but it ayb be a nuisance. Another possible reason for deleting a property is to force QM to use its default value.

Either case, to delete an existing property, you just prepend it with a hash. For example, if you have the property `aaa=bbb` and you want to delete it, you change it to `#aaa=bbb` and save. You can see the property disappear from the list and it will not be visible anymore. QM anyway retains the full change history – in fact, if you create again a property you just deleted, you will see it's full history, including its deletion!

Remember that whenever you make changes to a configuration property, this activity is logged on this is system log.

Passwords and Secrets

Some properties are known to be secrets. Those, once entered, cannot be displayed. To make your life easier, short passwords would be completely blanked out (e.g. `abc` would become `***`), but for longer password the beginning and end of their value will be shown. This is especially useful for identity, tokens, so you can understand if it's the right one without completely disclosing it. When editing the property, it will be always shown as a blank (so you basically have to enter it every time).

Exporting properties

A button lets you export the current set of properties as a text file, for reference or to be sent to technical support. Any known passwords will be blanked out as described above.

Using the DbTest Diagnostic Tools

The DbTest page, available at the address http://127.0.0.1:8080/queuemetrics/dbtest , will not only let you update the database, but also check a number of QueueMetrics subsystems. It is invaluable for debugging QueueMetrics installations where you suspect some problem may be.

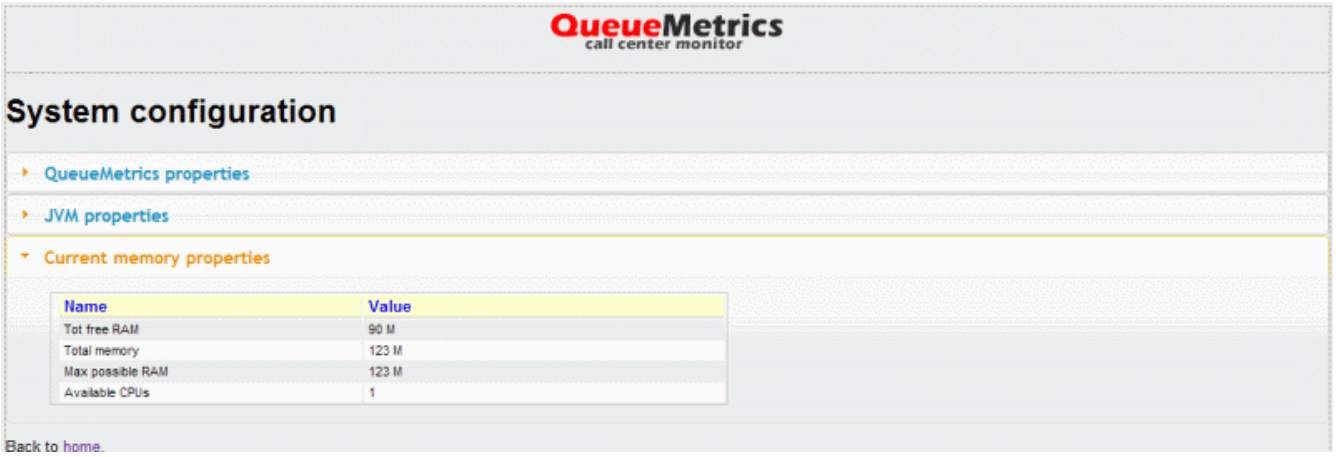


It is also possible to access this page directly from the QueueMetrics home page by selecting the *System diagnostic tools* link, which can be found within the available Administrative tools.



It is possible to completely turn off the DbTester page when not needed by toggling the *default.viewTechInfo* system property. You should do so if not needed by your users.

Checking the current system configuration

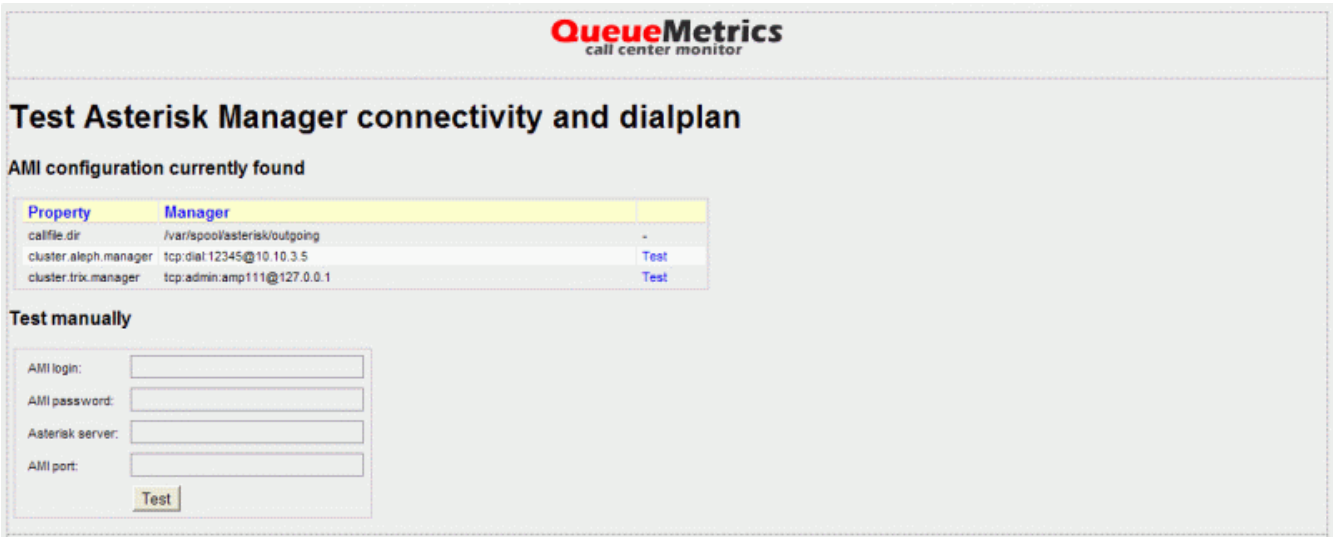


From this page, you can see:

- The current settings for all system configuration properties, as written in the configuration.properties file
- The current Java environment variables, usually defined at the JVM level
- The current memory and CPU settings for QueueMetrics, and the current memory usage. Note that Java will usually try to use all memory availble before doing a cleanup, so seeing most memory in use does not necessarily mean that QM needs more.
- The current HTTP/HTTPS environment and the HTTP headers passed to QueueMetrics. This is very useful to debug a QueueMetrics system that is behind one or more web proxies.

Checking an Asterisk Manager connection

It is possible to check an AMI connection to an Asterisk server.



As you can see, the AMI connections in your system preferences (see Configuring system preferences) are automatically read and can be configured at the touch of a button. As an alternative, you can manually enter the configuration parameters and see what happens.



In case the connection (like in the example above) displays an error, the complete stack trace is easily available for inspection. In case everything goes OK, QM will try to originate a call in order to check that the privileges are correct.

If the connection is possible, QueueMetrics will try to:

- Download and display the complete dial-plan, as displayed under "Configured Dialplan"
- Download and display the current queue configuration, as displayed under "Configured Queues". This shows the configuration as defined in queues. conf plus the current agent membership (static and dynamic).
- Download and display the current agent configuration (this only applies to agents as defined in agents.conf)

For further information on the AMI connection, see Configuring the AMI connection .



If you use a modern Asterisk system, the download of the dial-plan will likely fail, because it would require a special flag `live_dangerously = yes` that you don't really want to set. This is expected and is completely harmless in terms of the working status of your AMI connection.

Checking the AMI Queue

If you send commands to your PBX by proxying them through Uniloader, e.g. in platforms `CLASSIC_LIVE`, `DIRECTAMI_LIVE` or `FREESWITCH_LIVE`, you can see the status of the command queue and the current platform being used.

- *On*: is the moment when the command was queued for execution
- *Cmd* is the command
- *Ext*, *Agent*, *Queue* are the parameters recorded
- *State*: can be `WAIT` (command is queued but not yet processed), `SENT` (command was received by Uniloader and is being applied), `OK` if it was successfully applied or `KO` if command execution went bad. If any error was reported back by the PBX - as it usually happens - it is also shown.

- *Sent*: is the time it took before the comamnd was received by Uniloader.
- *Proc*: is the time it took for Uniloader to apply the command and send back a status.
- *Srv*: is the server being used
- The last columnn displays a button to read the actual dialog that was generated to be applied by the PBX. The details depend on the platform being used, and only appear after the call is in state SENT.

The command queue is kept for a limited time (usually one day or so) and then deleted.

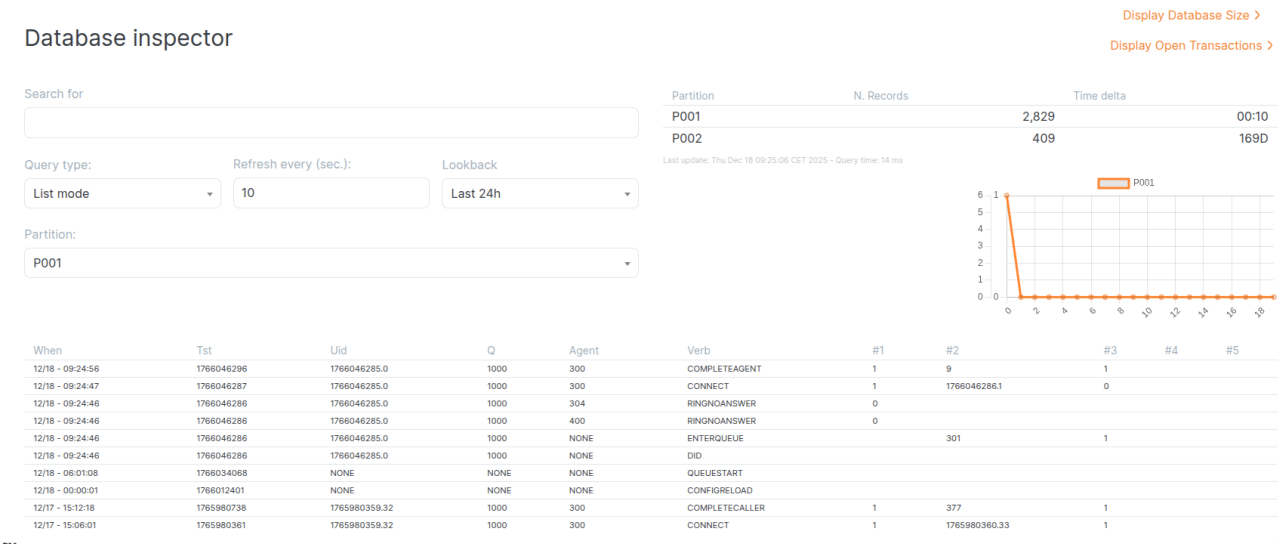
It is normal to have a Sent time of a few seconds when running Uniloader over the web.



The most common issues scenarios are that either commands stay forever in `WAIT` state, or you get `KO` errors. A long `WAIT` state means that either the Uniloader is not running, it cannot connect or it is waiting after a long set of errors before attempting the next call. Either case can be fixed by restarting Uniloader. If you get errors in the form of `KO` responses, there are usually two reasons: either the authentication credentials are wrong - which you can verify by running `uniloader test` - or, more often, the channel format for agents or calls is incorrect. If, say, a logged on agent appears on the PBX as "SIP/123", then a pause command to "Agent/123" or "PJSIP/123" won't work. By checking the actual dialog that was sent to the PBX and comparing it to the actual queue on the PBX, it is easy to spot such mistakes.

Live inspector of the QueueMetrics database

This page displays the current contents of the `queue_log` table, to make it easy to see data as it is appended by the PBX, and also allows to inspect the size of database tables.



The last 100 lines of the queue_log table for the given partition are displayed. In "Split mode" the last 40 lines of the `queue_log` table about calls and the last 40 lines about agent status are displayed separately.

When the time period "last 24 hours" is selected, the display will reload automatically every 10

seconds. When using longer time periods, the page will only reload if you change any parameters.



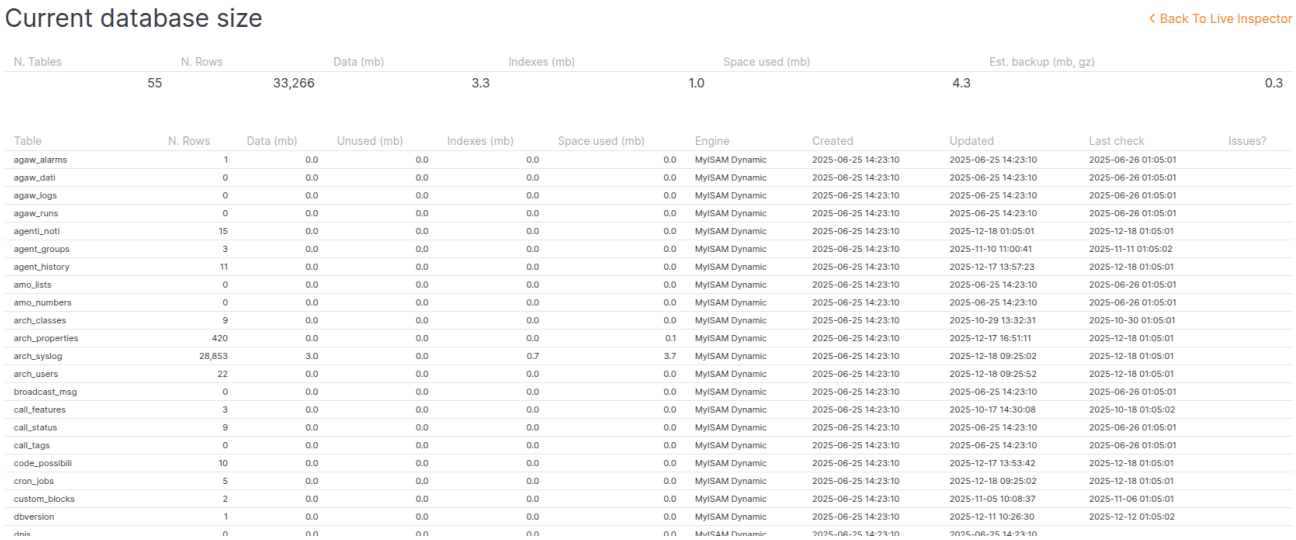
If you see "Partition null" in the graph, this means the `queue_log` table is empty.

It is possible also to search for a substring within a given partition, e.g. a uniqueid or an agent code. This may be very slow, and it usually requires a complete table scan. Do not do this repeatedly on a busy production box!

The Partition graph displays the number of events per minute in the last hour or so.

Database size

The inspector also lets you see the current state and size for all tables in the QueueMetrics database.



Current database size

‹ Back To Live Inspector

N. Tables	N. Rows	Data (mb)	Indexes (mb)	Space used (mb)	Est. backup (mb, gz)
55	33,266	3.3	1.0	4.3	0.3

Table	N. Rows	Data (mb)	Unused (mb)	Indexes (mb)	Space used (mb)	Engine	Created	Updated	Last check	Issues?
agaw_alarms	1	0.0	0.0	0.0	0.0	MyISAM Dynamic	2025-06-25 14:23:10	2025-06-25 14:23:10	2025-06-26 01:05:01	
agaw_dati	0	0.0	0.0	0.0	0.0	MyISAM Dynamic	2025-06-25 14:23:10	2025-06-25 14:23:10	2025-06-26 01:05:01	
agaw_logs	0	0.0	0.0	0.0	0.0	MyISAM Dynamic	2025-06-25 14:23:10	2025-06-25 14:23:10	2025-06-26 01:05:01	
agaw_runs	0	0.0	0.0	0.0	0.0	MyISAM Dynamic	2025-06-25 14:23:10	2025-06-25 14:23:10	2025-06-26 01:05:01	
agenti_noti	15	0.0	0.0	0.0	0.0	MyISAM Dynamic	2025-06-25 14:23:10	2025-12-18 01:05:01	2025-12-18 01:05:01	
agent_groups	3	0.0	0.0	0.0	0.0	MyISAM Dynamic	2025-06-25 14:23:10	2025-11-10 11:00:41	2025-11-11 01:05:02	
agent_history	11	0.0	0.0	0.0	0.0	MyISAM Dynamic	2025-06-25 14:23:10	2025-12-17 13:57:23	2025-12-18 01:05:01	
amo_lists	0	0.0	0.0	0.0	0.0	MyISAM Dynamic	2025-06-25 14:23:10	2025-06-25 14:23:10	2025-06-26 01:05:01	
amo_numbers	0	0.0	0.0	0.0	0.0	MyISAM Dynamic	2025-06-25 14:23:10	2025-06-25 14:23:10	2025-06-26 01:05:01	
arch_classes	9	0.0	0.0	0.0	0.0	MyISAM Dynamic	2025-06-25 14:23:10	2025-10-29 13:32:31	2025-10-30 01:05:01	
arch_properties	420	0.0	0.0	0.0	0.1	MyISAM Dynamic	2025-06-25 14:23:10	2025-12-17 16:51:11	2025-12-18 01:05:01	
arch_syslog	28,853	3.0	0.0	0.7	3.7	MyISAM Dynamic	2025-06-25 14:23:10	2025-12-18 09:25:02	2025-12-18 01:05:01	
arch_users	22	0.0	0.0	0.0	0.0	MyISAM Dynamic	2025-06-25 14:23:10	2025-12-18 09:25:52	2025-12-18 01:05:01	
broadcast_msg	0	0.0	0.0	0.0	0.0	MyISAM Dynamic	2025-06-25 14:23:10	2025-06-25 14:23:10	2025-06-26 01:05:01	
call_features	3	0.0	0.0	0.0	0.0	MyISAM Dynamic	2025-06-25 14:23:10	2025-10-17 14:30:08	2025-10-18 01:05:02	
call_status	9	0.0	0.0	0.0	0.0	MyISAM Dynamic	2025-06-25 14:23:10	2025-06-25 14:23:10	2025-06-26 01:05:01	
call_tags	0	0.0	0.0	0.0	0.0	MyISAM Dynamic	2025-06-25 14:23:10	2025-06-25 14:23:10	2025-06-26 01:05:01	
code_possibili	10	0.0	0.0	0.0	0.0	MyISAM Dynamic	2025-06-25 14:23:10	2025-12-17 13:53:42	2025-12-18 01:05:01	
cron_jobs	5	0.0	0.0	0.0	0.0	MyISAM Dynamic	2025-06-25 14:23:10	2025-12-18 09:25:02	2025-12-18 01:05:01	
custom_blocks	2	0.0	0.0	0.0	0.0	MyISAM Dynamic	2025-06-25 14:23:10	2025-11-05 10:08:37	2025-11-06 01:05:01	
dbversion	1	0.0	0.0	0.0	0.0	MyISAM Dynamic	2025-06-25 14:23:10	2025-12-11 10:26:30	2025-12-12 01:05:02	
dnis	0	0.0	0.0	0.0	0.0	MyISAM Dynamic	2025-06-25 14:23:10	2025-06-25 14:23:10		

The top table displays the number of database tables that make up the database, the number of distinct rows present, the amount of data and indexes and the expected disk space necessary to create a gzipped database backup.



The estimated backup size is meant as a rough estimate of the file size you could get by creating a compressed database backup as explained in Backing up. Make sure you have at least 2x the estimated disk space free before attempting a backup! Also, remember that for a complete backup, you need to save the configuration too.

The bottom table display, for every database table:

- The number of data rows, the data and indexes in use, and the total disk space used. Significant unused space may be reclaimed by optimizing the underlying table.
- The storage format (`myisam` or `innodb`)
- When the table was created, last updated and last checked
- Whether the table is marked as **crashed**, in which case you will have to manually perform a check to allow QueueMetrics to use it.



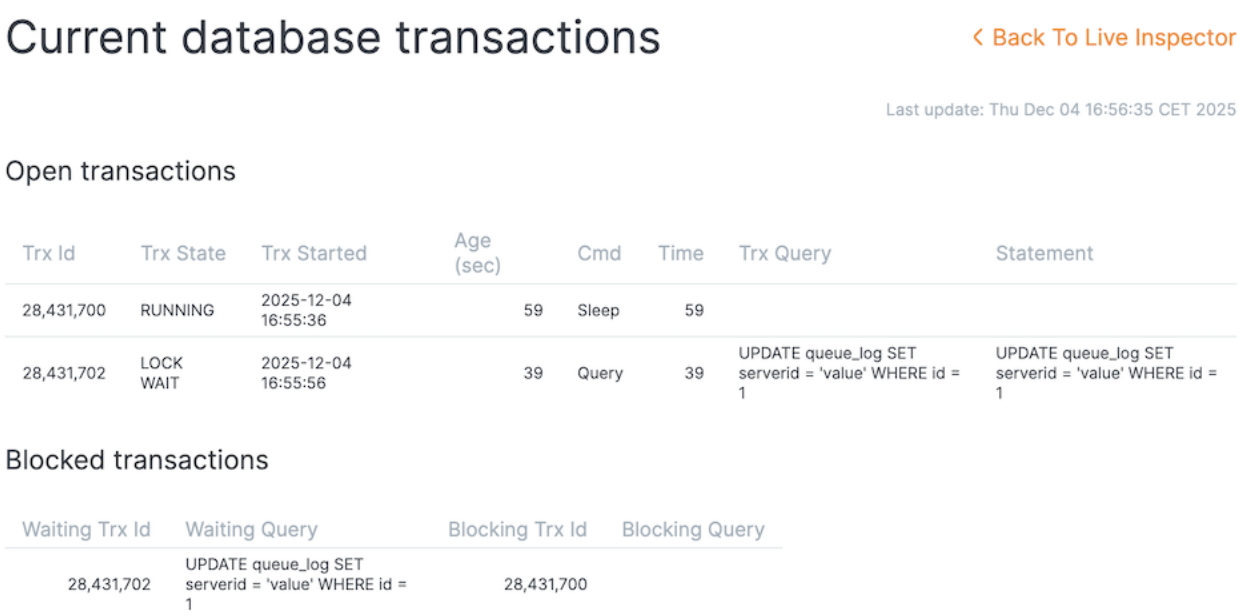
The largest tables are, usually, `arch_syslog` (the audit log, as described in The Audit log) and `queue_log` containing call data. You can have QueueMetrics take care of trimming the audit log for you by setting up a Scheduled job.

Transaction inspector

This page lets you see the current open transactions on the QueueMetrics database — which ones are active, how long they have been running, and their exact queries.

A database transaction is a sequence of one or more SQL operations that are executed as a single logical unit of work. When a transaction begins, it may acquire locks on the data it's modifying to prevent other transactions from making conflicting changes. If a transaction takes too long to complete or fails to commit, it can hold these locks and prevent other operations from proceeding.

The **Open transactions** table displays all currently active transactions, including their transaction ID, current state (such as RUNNING or LOCK WAIT), start time, age in seconds, and the query being executed. A transaction in LOCK WAIT state is waiting to acquire a lock held by another transaction.



In case any transaction cannot complete because it is being blocked by another transaction, both blocked and blocking transactions are shown in the **Blocked transactions** table at the bottom. This helps you identify which transaction is causing the blockage so you can take appropriate action, such as investigating why the blocking transaction has not completed or manually terminating it if necessary.

Transactions only apply to database tables of type `innodb`.



To grant the PROCESS privilege to the user queuemetrics, connect to MySQL as a user with sufficient privileges (like root) and run: MySQL

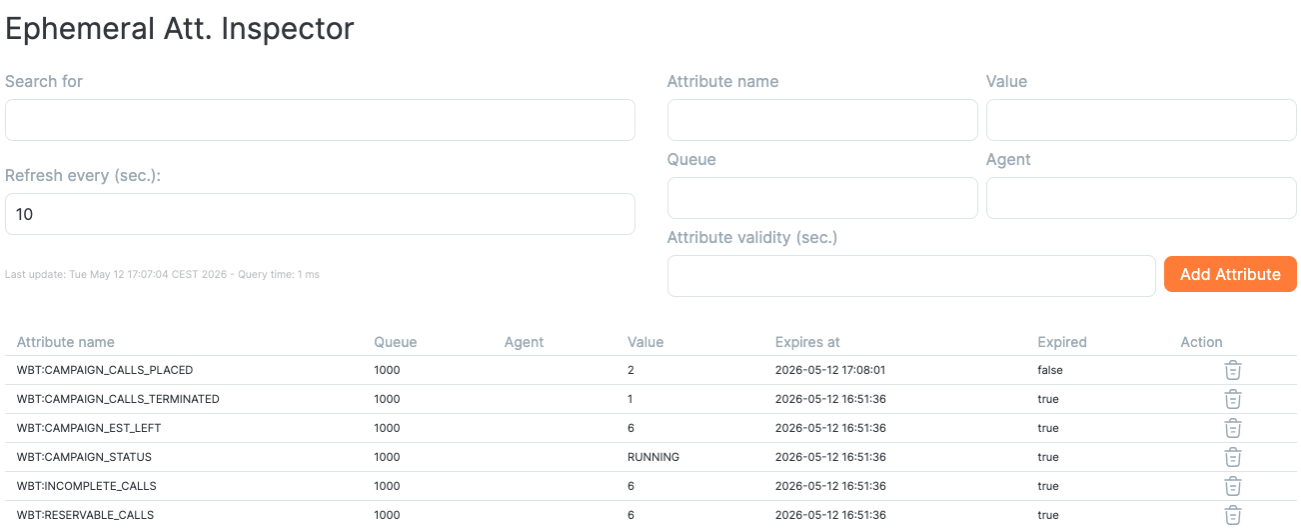
```
GRANT PROCESS ON *.* TO 'queuemetrics'@'localhost';
```

```
FLUSH PRIVILEGES;
```

```
Note:
- Replace 'localhost' with the appropriate host if the user connects from a different
host (e.g., '%' for any host, or a specific IP address).
- The PROCESS privilege is a global privilege, so it must be granted with ON *.* — it
cannot be granted on a specific database.
```

Live ephemeral attribute inspector of the QueueMetrics database

This page displays the current contents of the `ephemeral_ext_data` table, to make it easy to see data. To access this page, a user must be holding the `USR_DIAGNOSTICS` key.



The table displays the last 100 lines of the `ephemeral_ext_data` table. Each row identifies the attribute name, the associated queue, the specific agent, the expiration date, and the current expiration status. Clicking the trash icon allows for the immediate deletion of the selected row.

It is possible also to search for a substring, e.g. a attribute name or an agent code or a queue. This may be very slow, and it usually requires a complete table scan.

New ephemeral data can be added directly from this page. The attribute name and the attribute validity period are the minimum required inputs for a successful update.



Attribute name cannot contain the . character



In order to delete or add attribute, a user must be holding the `EPHEMERAL_WRITE` key.

RAM cache inspector

This page displays the current status of the different caching layers used within QueueMetrics.



The **log cache** is an object cache that stores, for each partition, event objects already processsed. It must be turned on by setting *realtime.useRowCache=true*. For each partition the number of cached objects it is displayed as "Rows". If you use hotdesking you should see a number of objects in the "Hotd" section as well. All of your partitions should appear here.



the **log cache** will be used only if the storage type is SQL-based.

The **SQL cache** caches SQL results to some common queries. Yo can see the current cache size (it is usually quite small), the total number of requests, how many of them were answered ("Cached fresh"), how many of them had the query repeated ("Cached stale") and the overall cache efficiency. In a busy call center this should be around 70-90%.

The **String cache** counts the number of string instances that are currently cached. Values in the 100,000 - 1 million element range are not uncommon.

All caches are designed to be emptied when working in order to reclaim memory, so you can safely click on "Clear caches" even while QM is running.

Testing email delivery

The DbTest contains a page where you can check the SMTP configuration by entering all parameters and trying to send an email. As this often requires some tweaking in the context of setting up a Scheduled job, it is better to start from a known working configuration.

The page prints the current configuration being used, and the dialog that happens with the SMTP server.

You have to enter an e-mail address to deliver to, and the page will initially load with the parameters as defined in the system preferences (see Configuring system preferences).

The page will leave the password field empty and won't display passwords; if you leave the password field empty, it will use the default password or the one you last used.

Each and every email attempt, be it successful or unsuccessful and no matter who generated it, will be logged to the Audit Log for further inspection.



The page is only meant to test parameters, and won't change them.

E-mail quotas

To avoid run-away processes sending thousands of emails per day, there is a hard limit in QueueMetrics of the number of e-mails that can be sent in 24h, both per-user and system-wide. See the properties `default.smtpDailyQuota` and `default.smtpDailyUserQuota` for a complete description. Any email sent after these quotas are reached will fail "as if" the upstream provider had rejected them.

When a quota limit is reached, the Audit log will show an error message like "User quota exceeded for user John Doe - 27 emails sent successfully in the last day" so that you can tell the reason for the failure. Any jobs running that triggered such an error will be marked as failed.

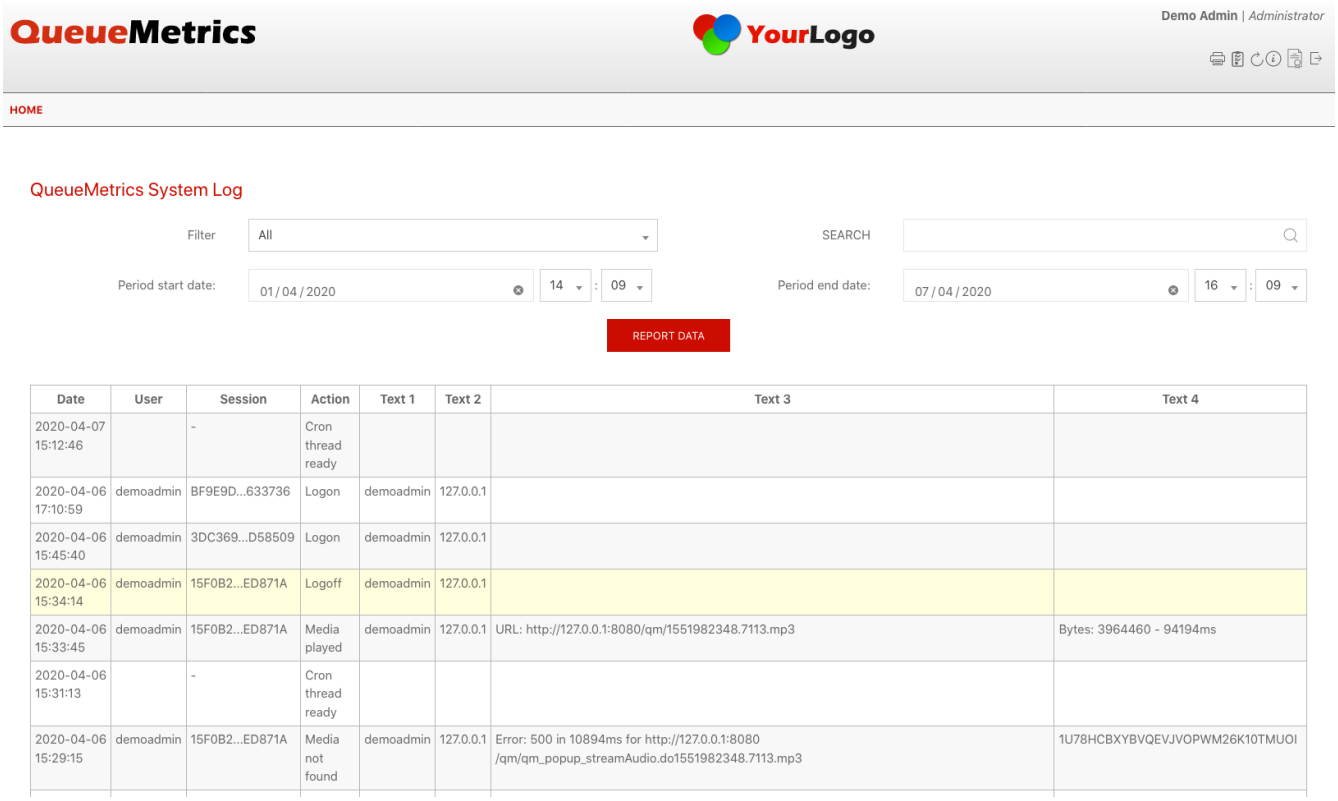


All e-mail sent, be it from manually triggered export jobs, from scheduled export jobs, or from the SMTP tester, are counted against quotas.

System audit log inspector

QueueMetrics keeps track of a number of activities that happen on the system; for example, every time a user logs on or off, this fact and the related IP address is added to the audit log. The same happens when the configuration is modified or large reports are run.

Administrators who hold the key USR_SYSLOG can access the audit log from *Home Page → View Audit Log*.



Date	User	Session	Action	Text 1	Text 2	Text 3	Text 4
2020-04-07 15:12:46		-	Cron thread ready				
2020-04-06 17:10:59	demoadmin	BF9E9D...633736	Logon	demoadmin	127.0.0.1		
2020-04-06 15:45:40	demoadmin	3DC369...D58509	Logon	demoadmin	127.0.0.1		
2020-04-06 15:34:14	demoadmin	15F0B2...ED871A	Logoff	demoadmin	127.0.0.1		
2020-04-06 15:33:45	demoadmin	15F0B2...ED871A	Media played	demoadmin	127.0.0.1	URL: http://127.0.0.1:8080/qm/1551982348.7113.mp3	Bytes: 3964460 - 94194ms
2020-04-06 15:31:13		-	Cron thread ready				
2020-04-06 15:29:15	demoadmin	15F0B2...ED871A	Media not found	demoadmin	127.0.0.1	Error: 500 in 10894ms for http://127.0.0.1:8080 /qm/qm_popup_streamAudio.do1551982348.7113.mp3	1U78HCBXYBVQEVJVOPWM26K10TMUOI

The page displayed is very simple; it allows searching in a given date range, by a substring of the values that appear on screen, and by "family" of events.

A number of different records are tracked throughout the system - see the Appendix for complete details.



On a busy system with 50+ agents, this log may get large fast; as it does not get deleted automatically, you should set up a scheduled job to trim it periodically. See Scheduled jobs for more details. You can check the current database size from the Live Inspector page.

Engine audits

It is possible to set default "size" limits, in terms of the time taken to run a report and the number of rows scanned to create it.

These limits apply to reports, webservices and real-time pages, and when exceeded will cause a log entry that details the user who was running that large report and the parameters used to create it.

You can control the duration through property `audit.maxReportTime` (defaults to 30 seconds) and the size through `audit.maxRowsFetched` (defaults to one million rows). Be careful that a low limit might affect each and every report run, so you may create a huge number of lines on the audit log.

In any case, any license violation (that is, a report with more agents than are available in your current license) and any Out Of Memory errors will be logged.

QueueMetrics Themes

Since QueueMetrics 20, the User Information page will allow each user to select his own theme, based on a selection of available themes.

System administrators are able to set a default theme for all users, by setting the following system property:

```
default.selectedTheme
```

This property decides the **default theme for ALL users**.

Example values for the property are:

```
default.selectedTheme=default
```

or

```
default.selectedTheme=dark
```

where default, dark are Theme IDs.

If the propery is empty for whatever reason, **it will always revert back to the value "default"**.

If the theme name is not available to the user (For example the name is spelled wrong, the "default" theme does not exist, or the selected theme was removed), **then a fallback theme will be provided** (See Below).

Currently selected theme

The currently selected theme is stored in the cookie "theme", **that is persisted indefinitely (99999 days)**. If the cookie is not present, **the default theme specified in the system property default.selectedTheme will be returned**.

If the theme specified in the default.selectedTheme property, **is not present in the list of themes available to the user**, a fallback theme will be determined, by taking **the first theme in alphabetical order** (of the description, not of the code) from the available themes list. This list is made up of all the themes that have a theme.properties file in the themes folder.

If for some reason, no theme is available, the fallback theme will be "default".

The fallback order is thus the following:

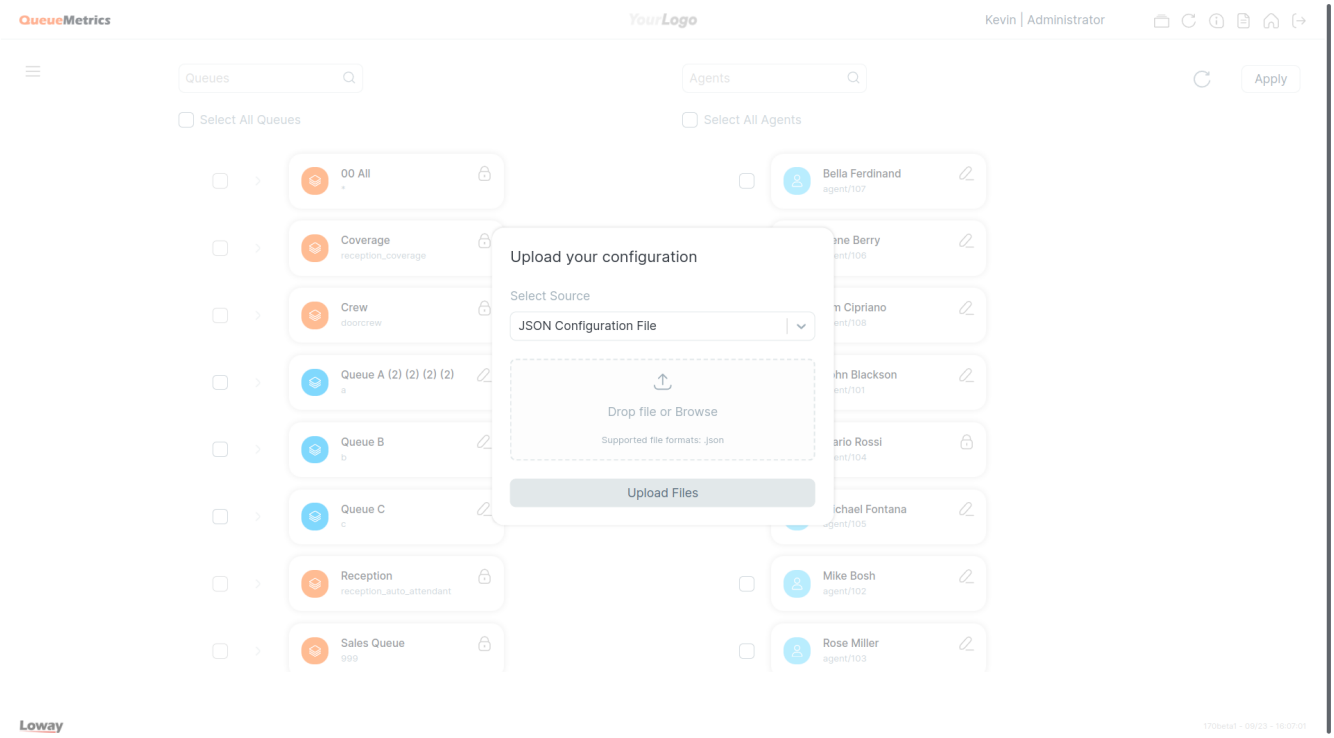
cookie ("theme") ⇒ default.selectedTheme (if available) ⇒ first available theme ⇒ "default"

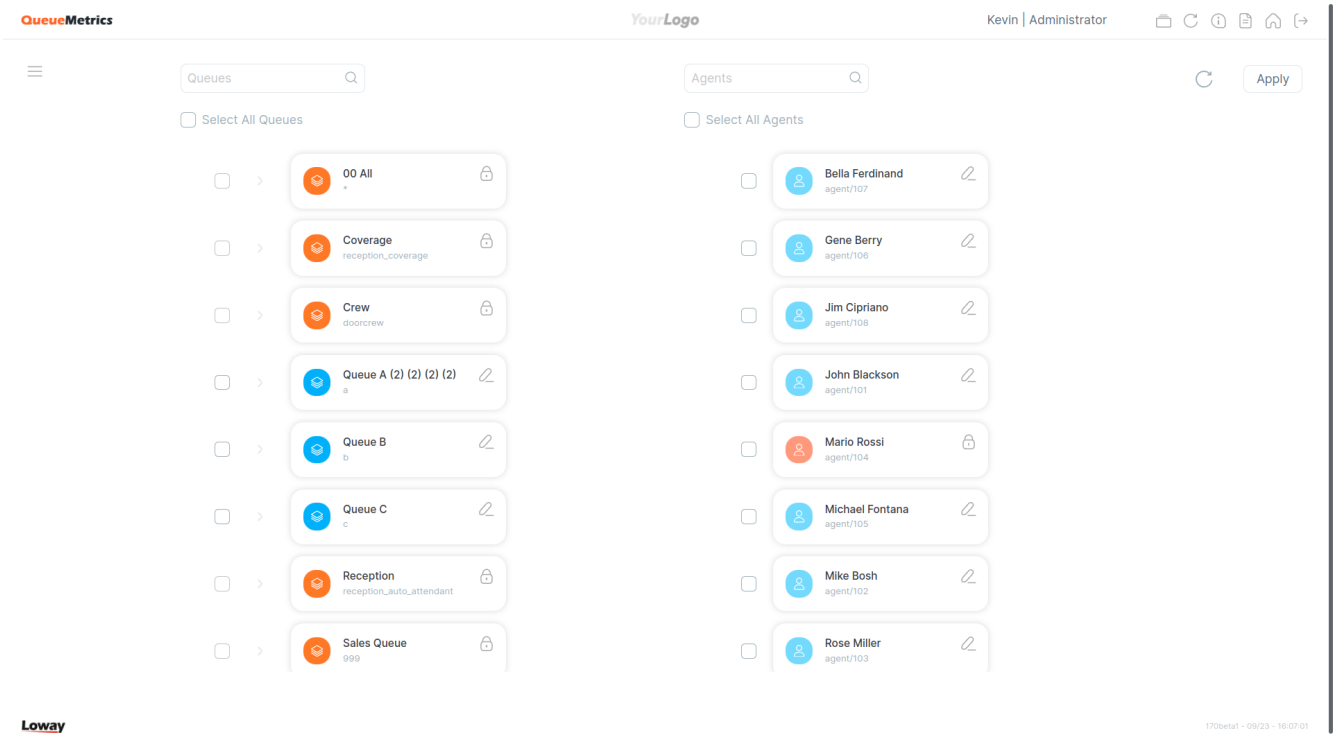
User Information Page

The list of available themes to choose from **will be located in the User Information Page**:

qm_adm/users_page.jsp == Synchronizer

QueueMetrics Configuration Synchronizer





The term Configuration refers to a QueueMetrics' set of queues, agents and agent-queue associations that QueueMetrics is able to assume.

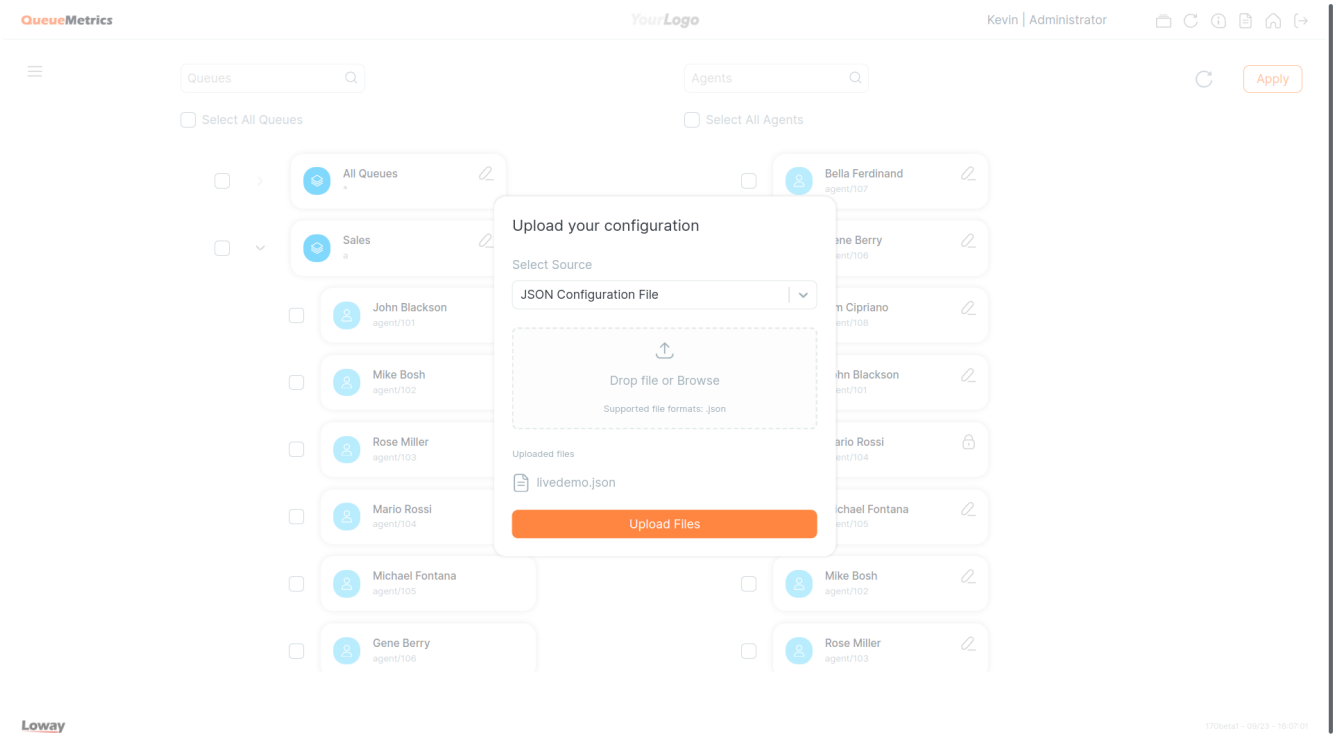
This module allows the user to quickly configure QueueMetrics by importing an existing QueueMetrics configuration and applying it to the current one. This configuration can be exported from another system (for example when migrating QueueMetrics from one system to another), or from the same system if you saved a previous configuration as a backup.

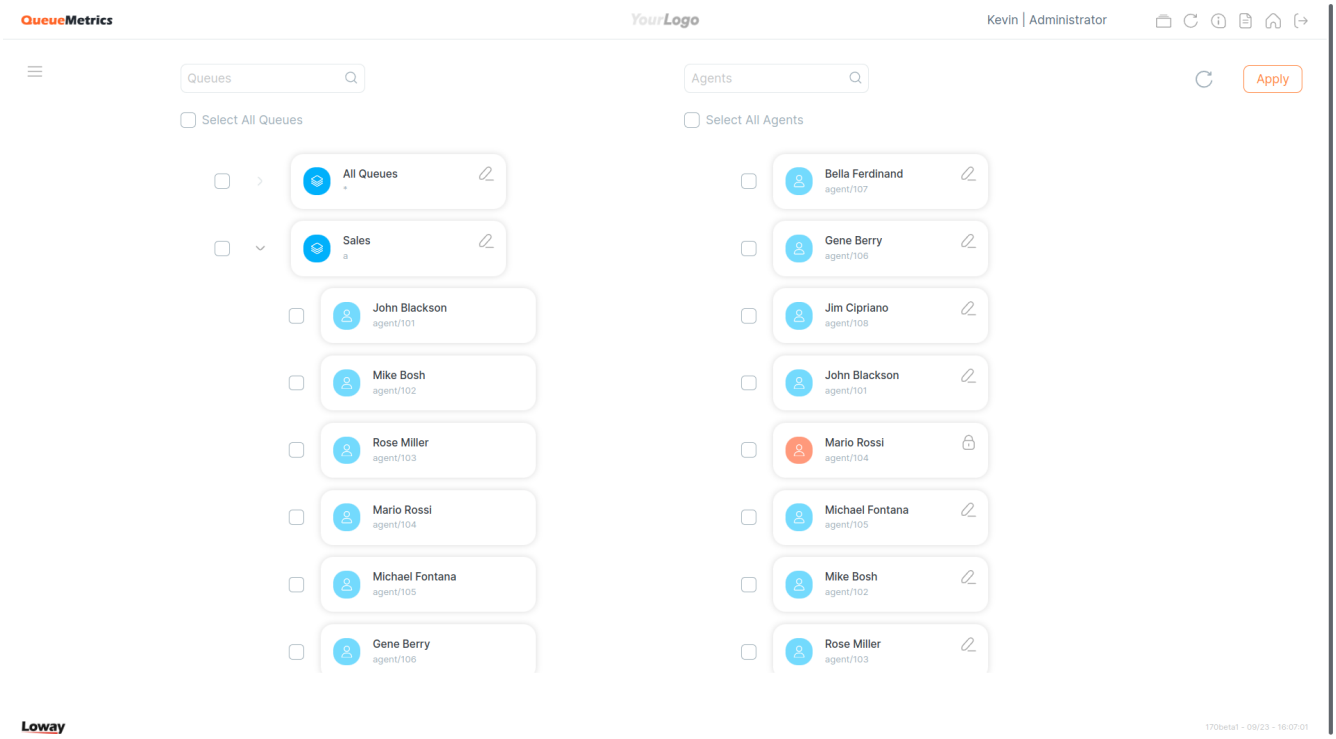
QueueMetrics configurations can be imported from multiple sources, listed below:

- JSON File.
- Queue Log Database Table
- URL
- Queue Log File

JSON File

QueueMetrics can export configuration files (see below) that can then be imported into any other QueueMetrics, including itself. By selecting JSON File, the user is instructing QueueMetrics to read the configuration from said file, after selecting it or dropping it into the drag area.



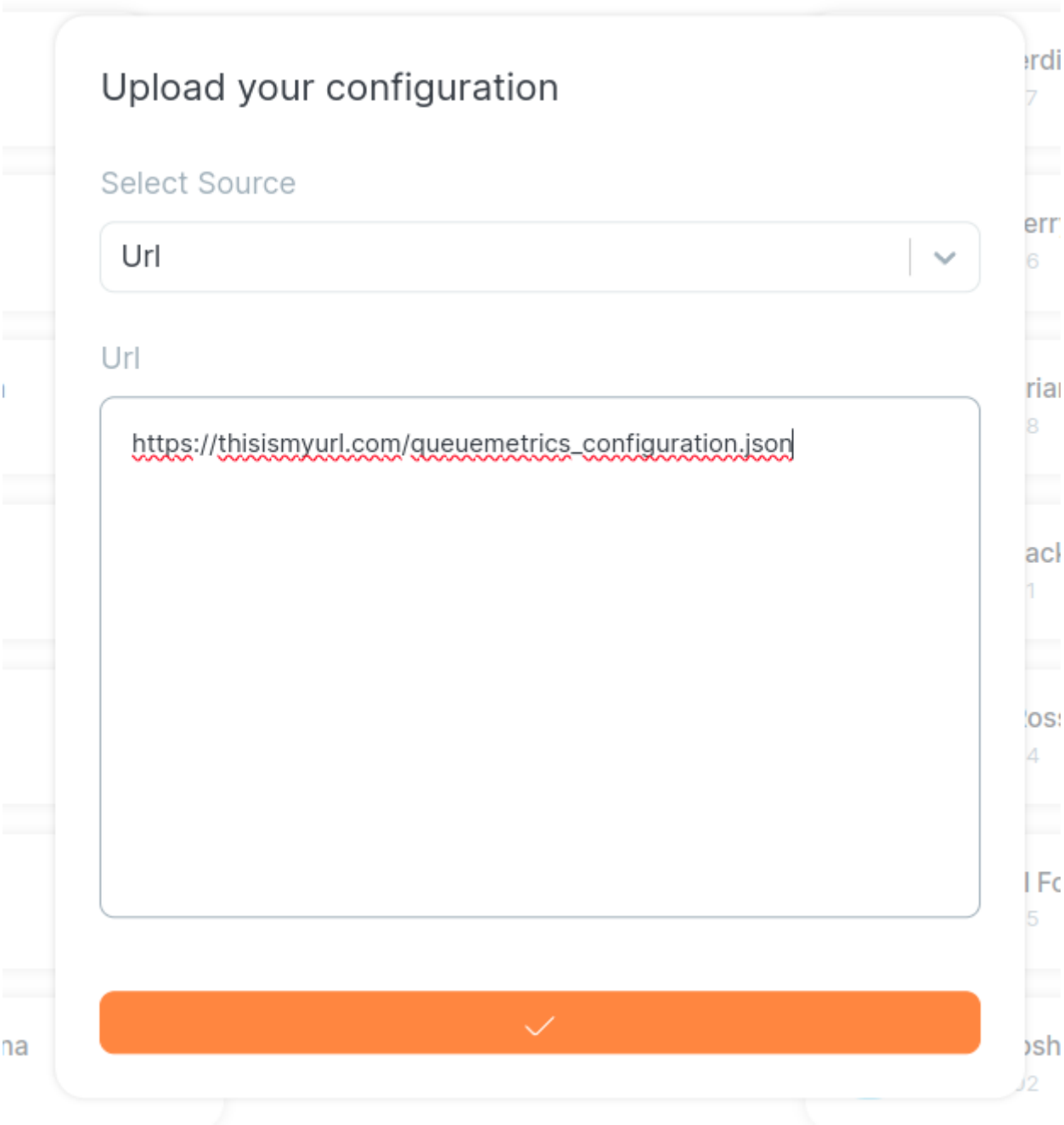


Once the file is imported correctly, the configuration shown in the Synchronizer will resemble the one described in the file.

If the imported configuration is different from the current configuration of the QueueMetrics system the user is running, the "Apply" button in the top right side of the screen will become enabled, allowing the user to finalize the configuration.

URL

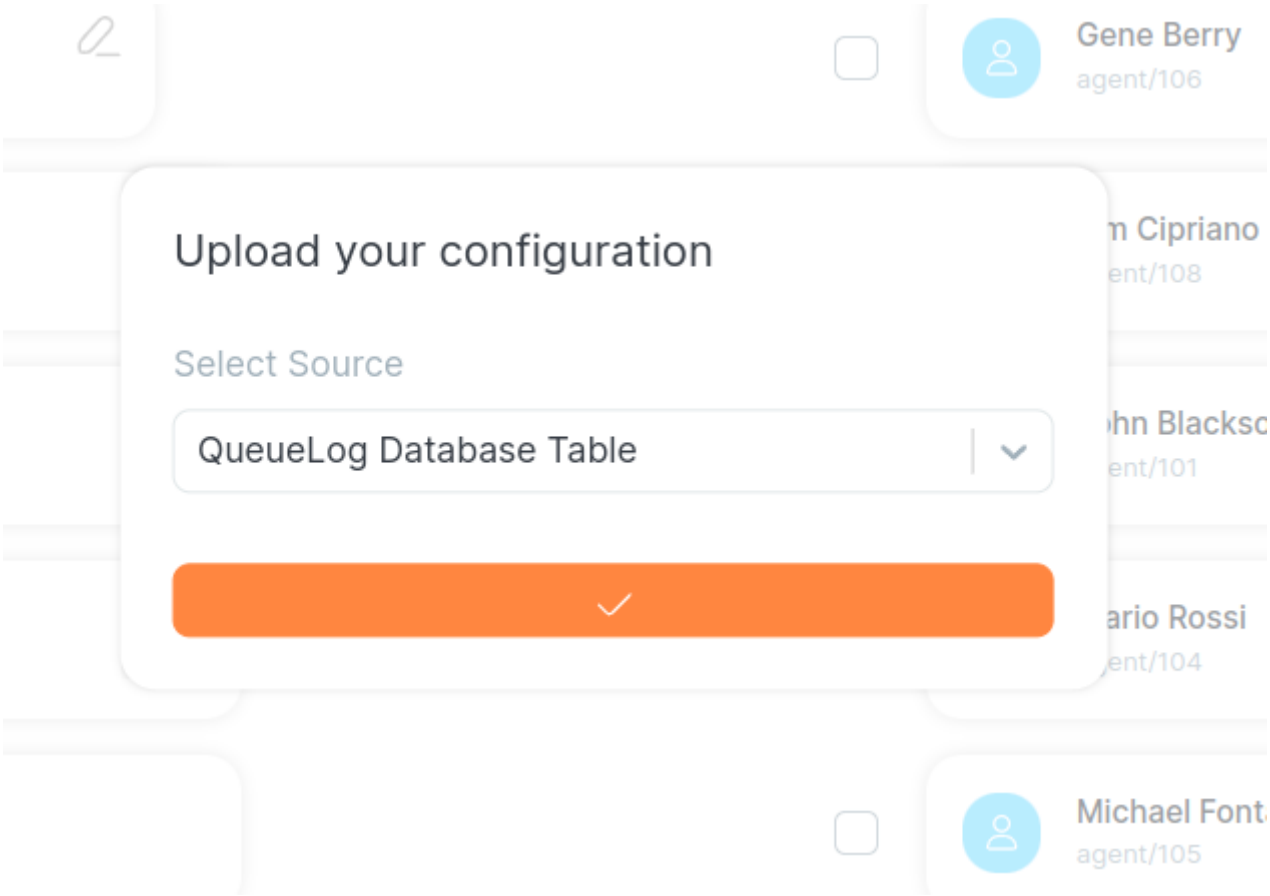
Similarly, a JSON configuration file can be imported from any URL it is hosted at.



Queue Log Database Table

Another useful feature in the QueueMetrics setup phase, is the Queue Log Database Table reader.

All the call and queue data from the PBX that is relevant for QueueMetrics is stored in an internal database table. By selecting this option, the user is telling QueueMetrics to analyze the data gathered in said table in order to produce a valid configuration. This is especially useful when setting up a new system, to let QueueMetrics automatically create the queue and agents needed for QueueMetrics to report on your PBX activity.

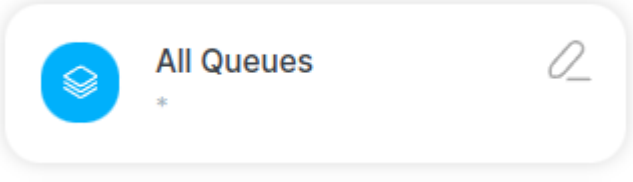


Queue Log File

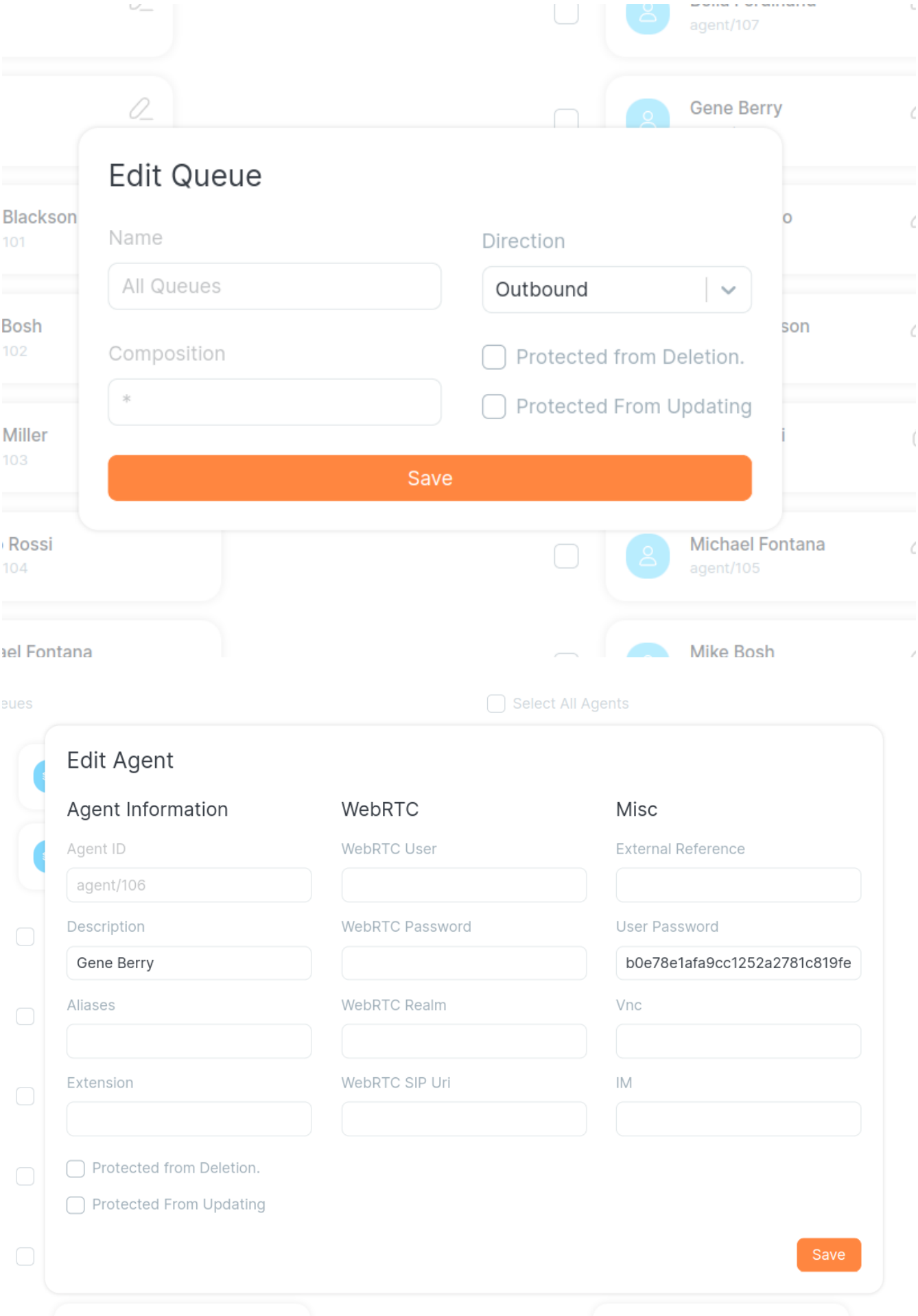
Similarly to the previous option, this one allows QueueMetrics to read the necessary information directly from a PBX-generated queue_log file.

Editing the configuration manually

Aside from importing an existing configuration, users can edit the current configuration manually, by clicking on the "edit" icon (Pencil) next to the queue and agent names.

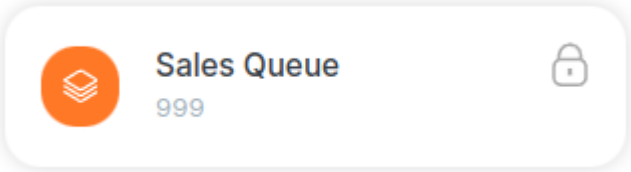


Doing this will open the relevant forms that allow the user to change a queue or agent information.



If a queue or an agent displays a "lock" icon (Padlock) instead of the "edit" icon (Pencil), it means

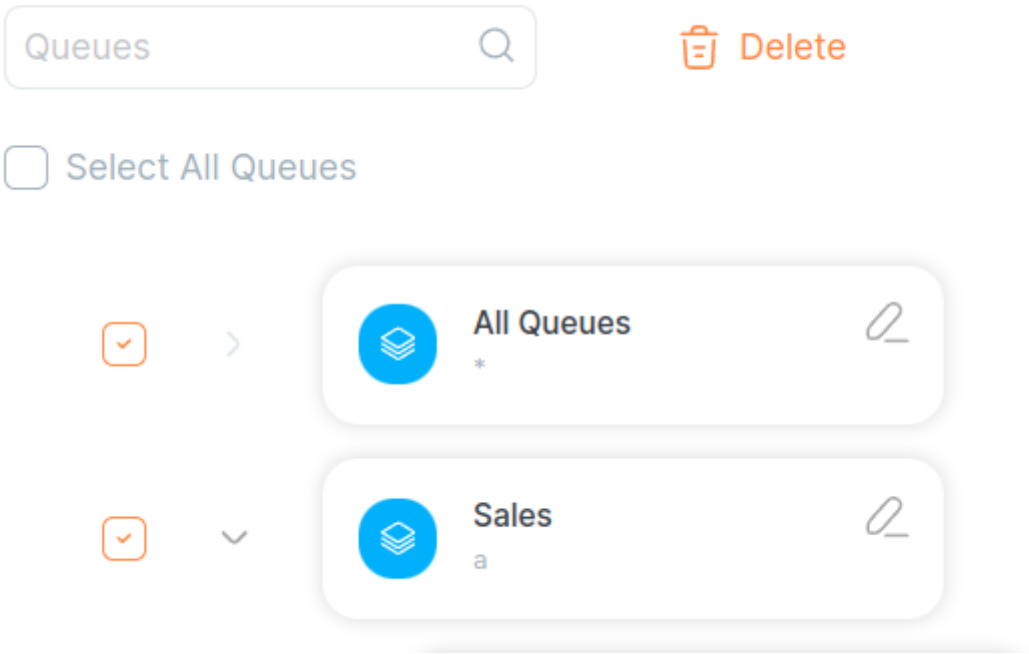
that said agent or queue are protected from editing or deleting. This means that said agents and queues can only be edited from the Queue and Agent editors.



Deleting agents and queues

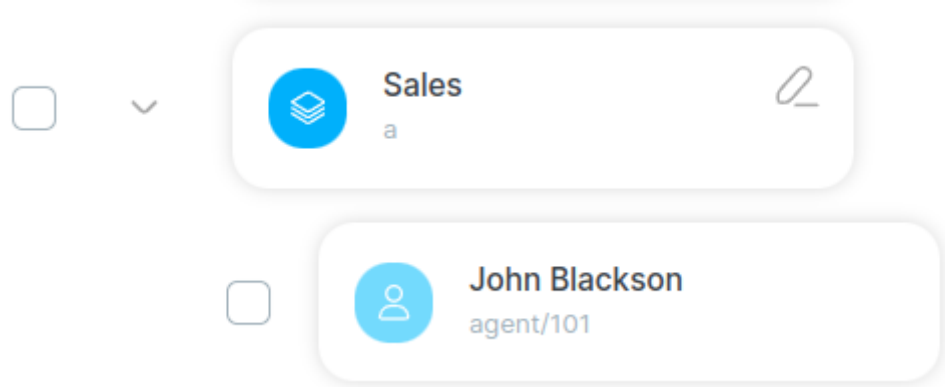
To delete agents and queues, the user has to click on the checkbox on the left of the entity they wish to eliminate.

Once one or more entities are selected, they can be eliminated by clicking on the delete button that appears at the top of the list. By clicking on the select all checkbox, the user can select all agents and queues at once.



Assigning Agents to Queues

Assigning an agent to a Queue is very simple. The user needs to drag an agent from the list on the right, and drop it on the queue they want to assign the agent to. This will create an "Agent on Queue" object that can be eliminated in the same way as a queue, by clicking on the checkbox on its left.



Deleting a queue will automatically delete all the agent/queue associations that were present on that particular queue.

Applying the configuration

If the "Apply" button is enabled, it means that the user can finalize the configuration change. This will prompt QueueMetrics to compare the current configuration present on the system, with the new one that the user just finalized.

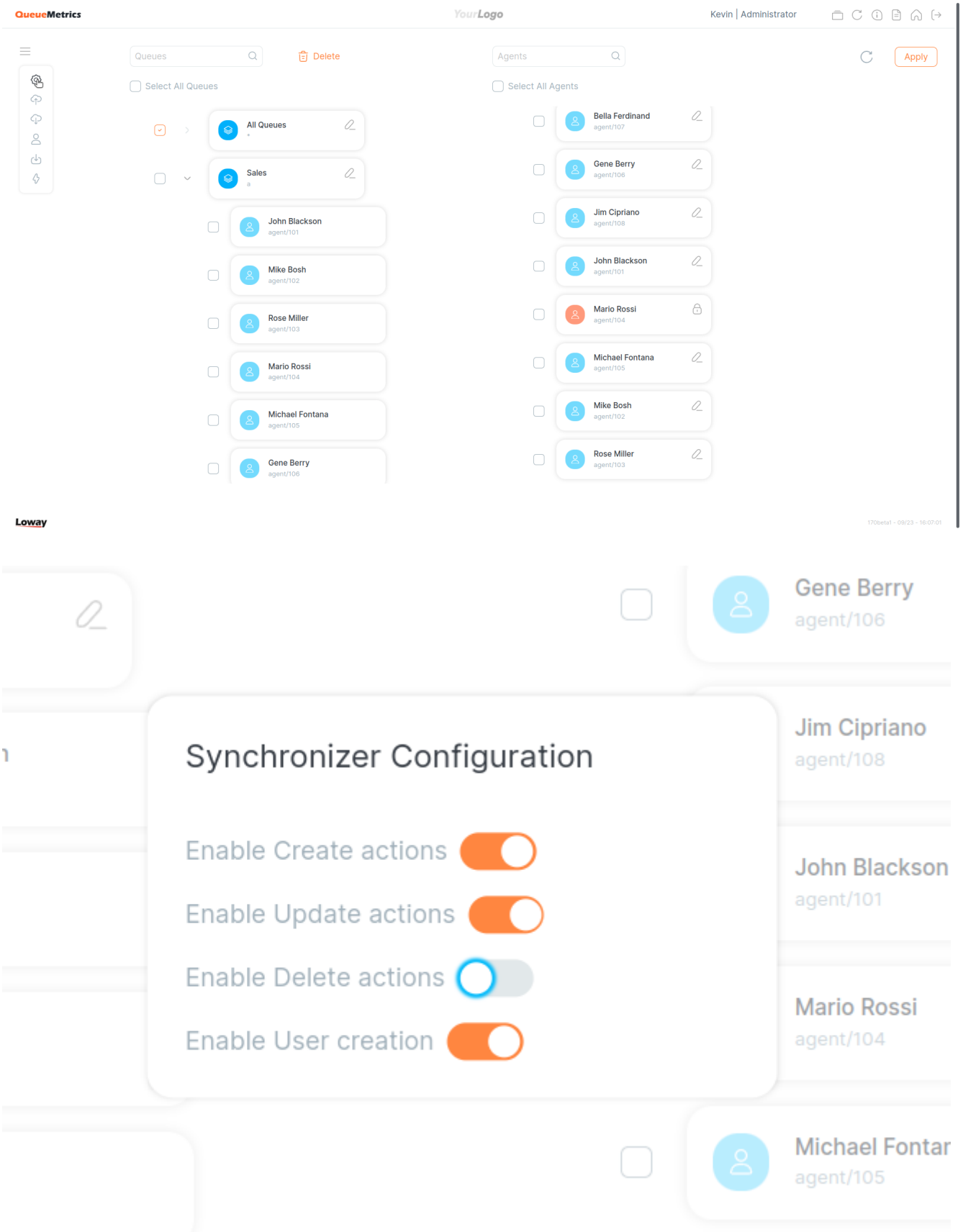
Once QueueMetrics is done comparing the two configurations, it will produce a list of actions that will be performed immediately to bring the current configuration to the new one.

These actions can be of four different types:

- Create Actions: can create a queue, an agent or an agent on queue association.
- Delete Actions: can delete a queue, an agent or an agent on queue association.
- Update Actions: can update a queue, an agent or an agent on queue association, by changing some of its information.
- Create User: can create a valid QueueMetrics User. This type of action only happens when a new Agent is created, and it will assign a random password to the linked user, that can be changed in the User Configuration page.

Disabling actions

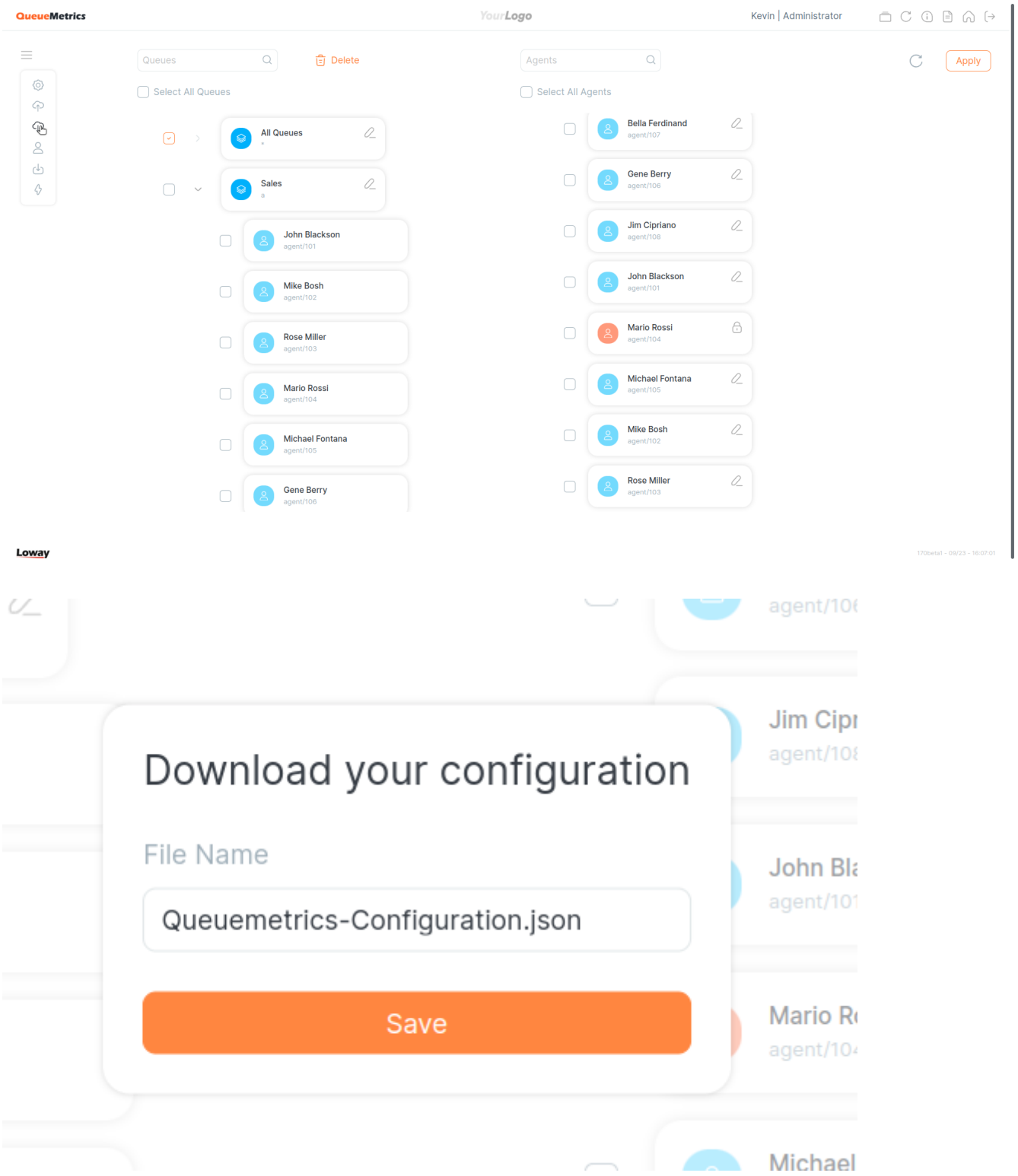
Actions in the synchronizer can be disabled selectively, by clicking on the "settings" icon (Gear) in the top left menu.



Disabling a specific type of action will not apply said type to the final step of the configuration. If, for example, the user disables delete actions, the final configuration will include all agents and queues from the old AND new configuration, since it will not be allowed to delete any queue or agents.

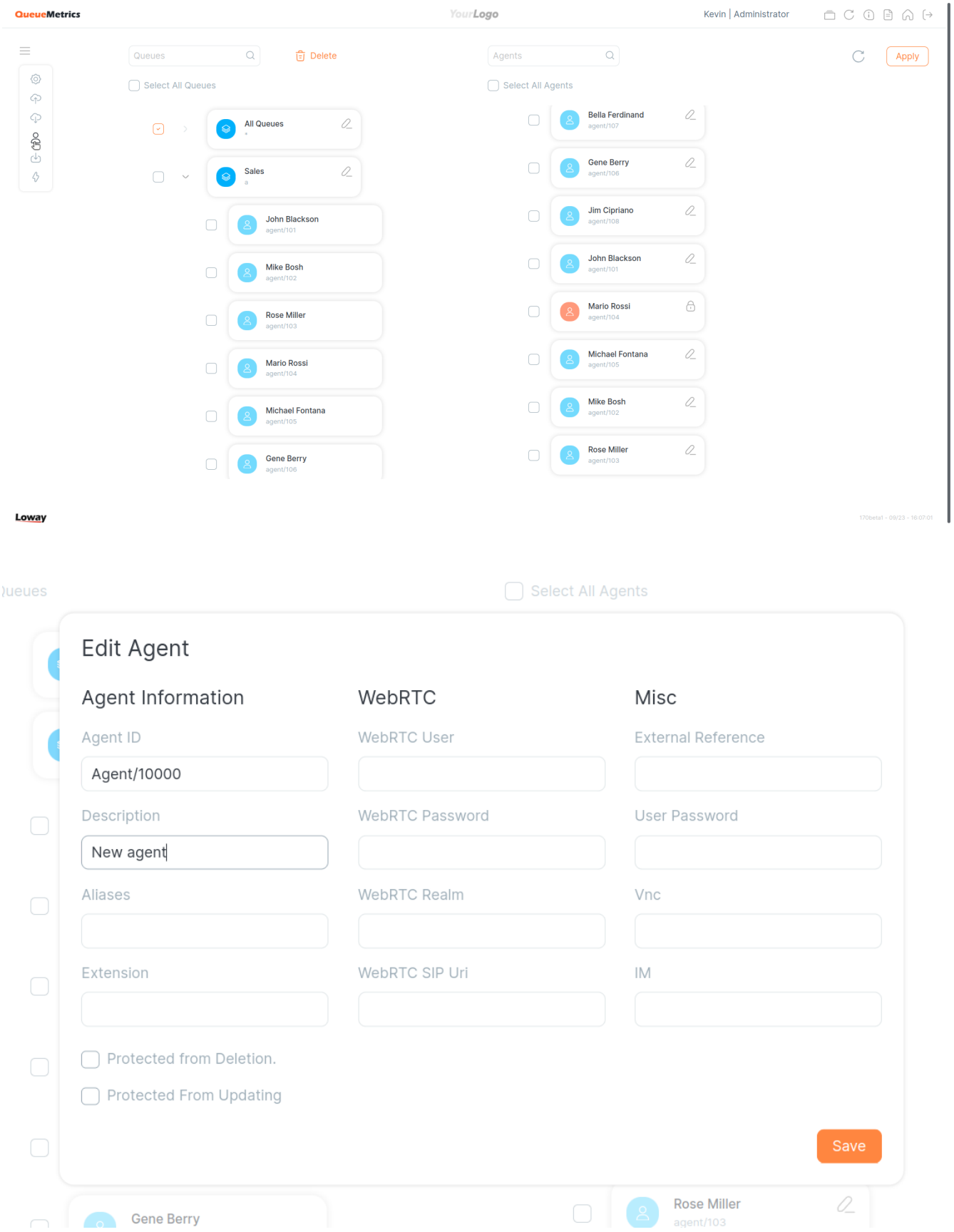
Exporting configurations

By clicking on the "export" icon (Download) in the top left menu, the user can export the current configuration in a JSON file, to be imported in the future or by another QueueMetrics system.



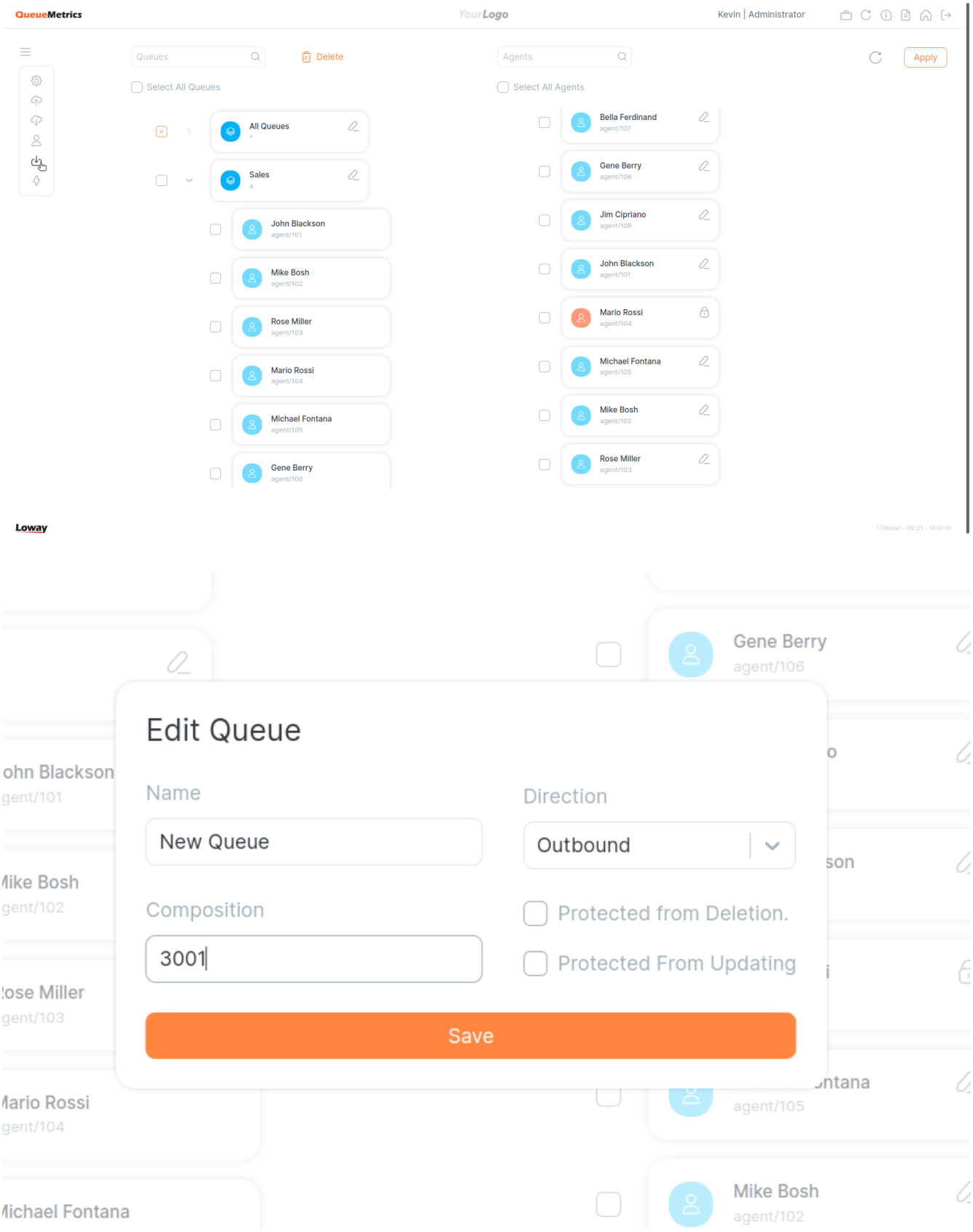
Creating an Agent

By clicking on the Agent icon the user can create a new agent to add to the configuration currently displayed by the synchronizer.



Creating a Queue

By clicking on the Queue icon the user can create a new queue to add to the configuration currently displayed by the synchronizer.



If a queue with the same name or composition already exists in QueueMetrics, no action will be performed. To apply changes via the Synchronizer, you must first delete and recreate the queue; alternatively, you can manually edit it within the Queue configuration page.

General Actions

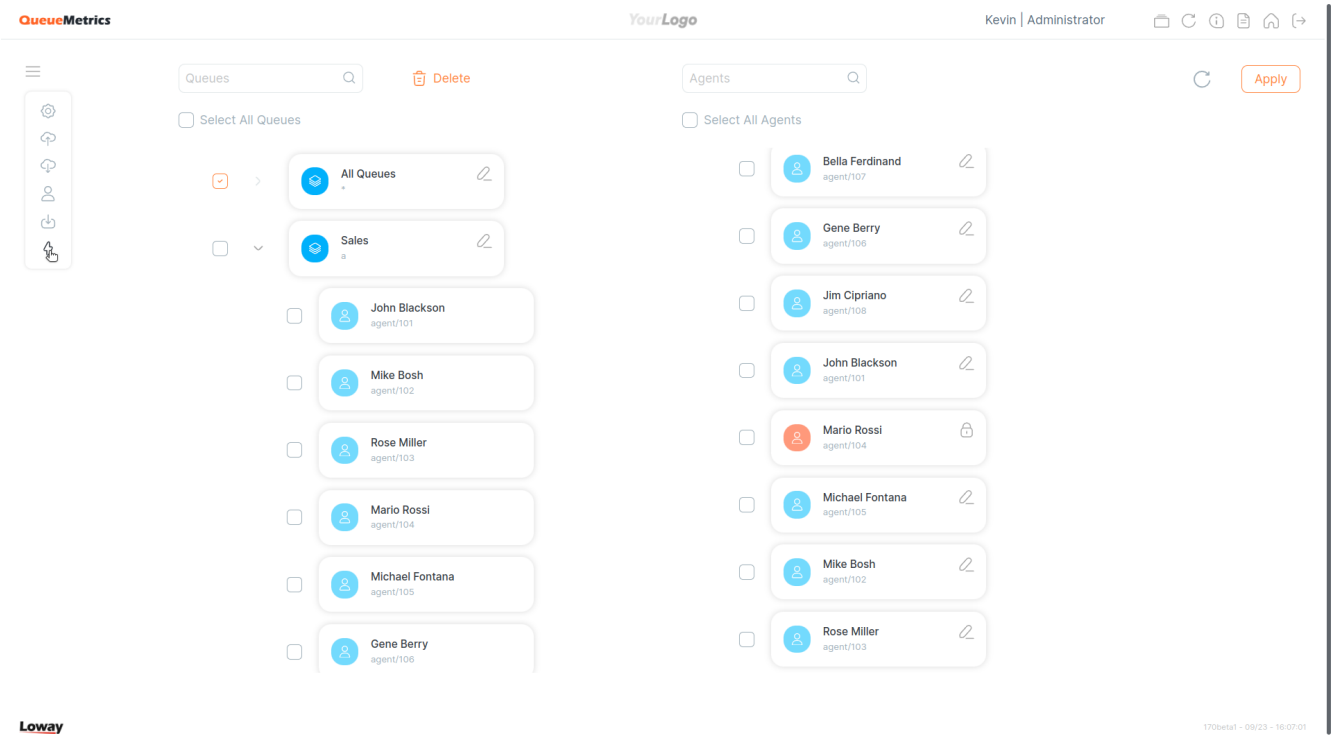
You can use the **General Actions** panel to execute various actions on a group of objects at the same time.

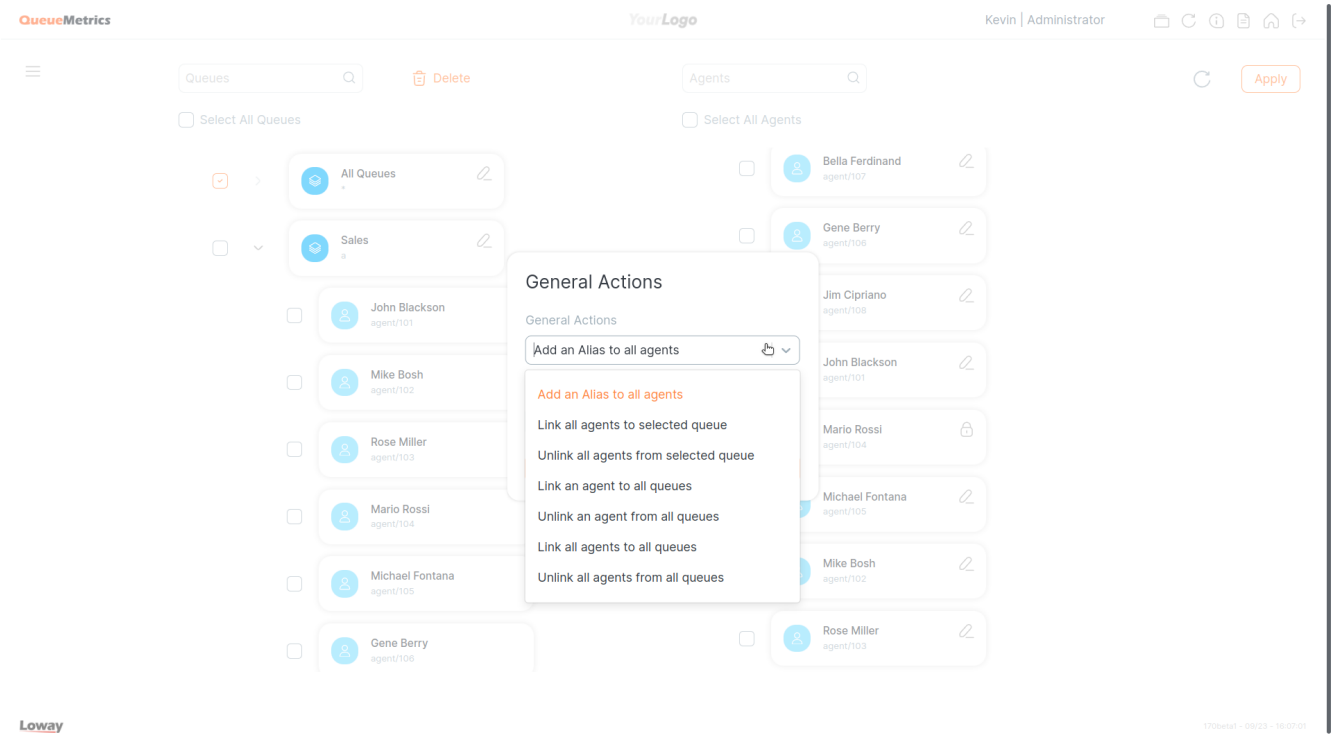
- Add an alias to all the agents.



you can use ${num} to add the agent code (eg 200 if the agent is agent/200) to your alias. You can use ${descr} to add the agent name to the alias.

- Link all agents to one queue
- Unlink all agents from a queue
- Link one agent to all queues
- Unlink one agent from all queues
- Link all agents to all queues
- Unlink all agents from all queues




QueueMetrics
YourLogo
Kevin | Administrator
Queues
Delete
Agents
Apply
Select All Queues
Select All Agents
All Queues
Sales
John Blackson
agent/101
Mike Bosh
agent/102
Rose Miller
agent/103
Mario Rossi
agent/104
Michael Fontana
agent/105
Gene Berry
agent/106
Bella Ferdinand
agent/107
Gene Berry
agent/106
Jim Cipriano
agent/108
John Blackson
agent/101
Mario Rossi
agent/104
Michael Fontana
agent/105
Mike Bosh
agent/102
Rose Miller
agent/103
General Actions
General Actions
Add an Alias to all agents
Add an Alias to all agents
Link all agents to selected queue
Unlink all agents from selected queue
Link an agent to all queues
Unlink an agent from all queues
Link all agents to all queues
Unlink all agents from all queues
Loway

The QueueMetrics Platform model

Platforms are the way QueueMetrics is able to connect to different models of PBXs; each platform encapsulates the logic required to exchange data with the underlying PBX.

Generic concepts

A Platform is composed of a number of swappable components that encapsulate all interaction with the underlying PBX.

Those components let you define the way QueueMetrics:

- pulls data from the PBX ("Ingestor")
- analyzes and stores data sent from the PBX ("Storage")
- interacts with the PBX ("PBX Actions")
- drives an outbound dialer ("Dialer")
- listens to stored recordings ("PM Recordings")
- listens to live calls ("PM Listen")

Each platform is defined through the system preferences (see Configuring system preferences) by setting the appropriate properties.

Platform feature	**Main controlling property**	**Notes**
Ingestor	-	Not yet implemented.
Storage	default.queue_log_file	
PBX Actions	platform.pbx	
Dialer	platform.dialer	
PM Recordings	audio.server	
PM Listen	audio.liveserver	

These components are meant to be "mix-and-match", so that you can e.g. use the same audio recordings mechanism with totally different storage engines.

Defining system properties for members of a cluster

Unless otherwise specified, in the case you are running a clustered system, you can have "default" settings and then separate settings for each machine in the cluster.

For example, if you are trying to read a configuration property called "my.prop" and you are on a cluster server called "alpha":

- first, a property called *cluster.alpha.my.prop* is read
- if not found, the property *my.prop* is read

- if not found, the default value is used.

Ingestors

No ingestors are currently used.

Storage models

At the moment, QueueMetrics only supports its classic storage model, either reading from the database or through a flat-file.

Outbound dialer

The outbound dialer is controlled through the `platform.dialer` configuration property.

Allowed values are:

- NONE - no dialer available
- WOMBATDIALER - Connects to an instance of WombatDialer. This parameter is inferred if the property `default.wombat.agent_recall_enabled is set to true.

None

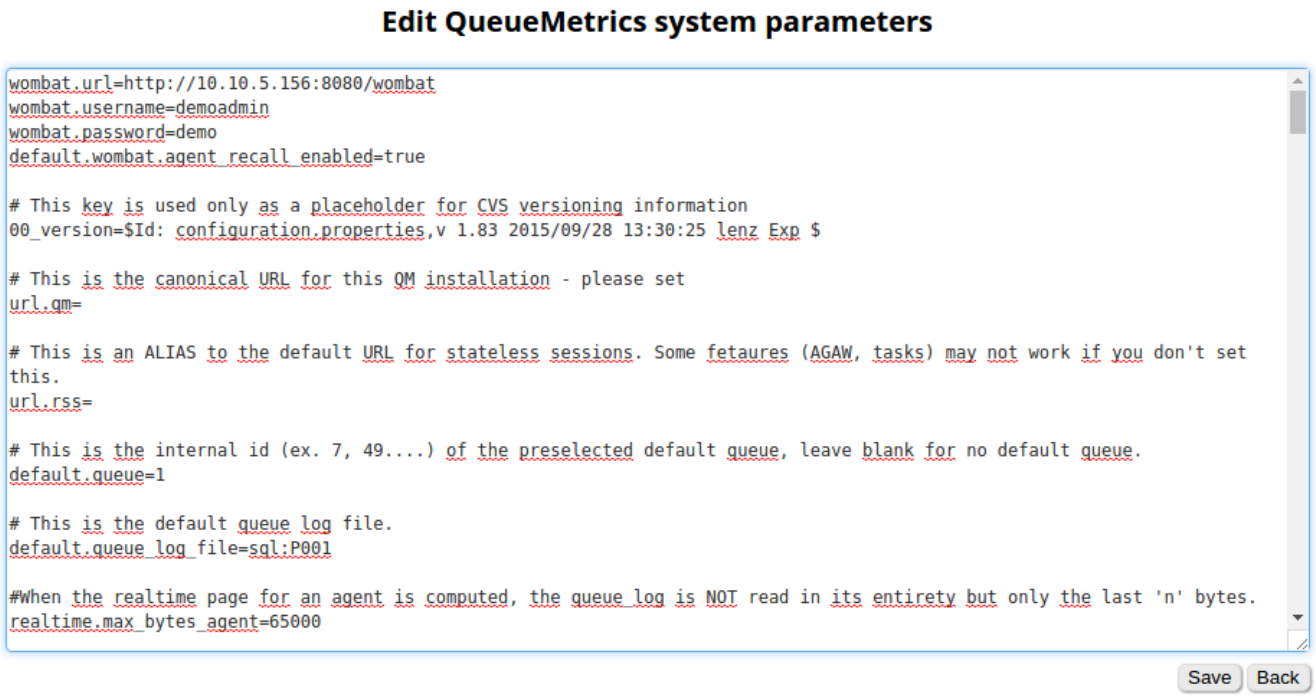
No dialer is available, all dialer actions are turned off.

WombatDialer

QueueMetrics is able to connect to a WombatDialer instance to schedule calls.

This must be provided in the form of four system parameters (that can be set in the Edit System Parameters page).

- `wombat.url`: The URL of our WombatDialer instance.
- `wombat.username`: A valid username associated with a user with the needed grants.
- `wombat.password`: The password for the user.





For more information of how to integrate WombatDialer within QueueMetrics, please see WombatDialer Recalls

Listening to calls using Pluggable Modules (PM)

Pluggable modules are used in two areas:

- Listening to recorded (closed) calls, i.e retrieving recordings
- Listening to live calls, i.e. setting up a channel "spy" feature.

In order to control which module is called, two configuration properties are set:

- *audio.server* controls the module to do find recorded calls
- *audio.liveserver* controls the module to set up live call listening

Each configuration property is set to the complete name of a Java class that implements the required server. *Such names must be set exactly as described, or an exception will be raised.* Each module can then have its own configuration properties to control its own behaviour.

Supported platforms

The following is a list of supported platforms.

Classic

This is the "classic" locally-hosted QueueMetrics platform. Data can be stored on a flat-file, uploaded to the QueueMetrics database or read out Asterisk's own tables. In order to perform actions on the PBX, the custom QueueMetrics dial-plan is required for all actions (Actions: Classic) or only for outbound dialing (Actions: DirectAMI).

Platform feature	Suggested modules	Notes
Ingestor	Not needed (upload via Uniloader)	
Storage	SQL, flat-file, Cluster	
PBX Actions	CLASSIC / DIRECT_AMI	Default
PM for recorded calls	LocalFilesByDay, Oreka, MultiListener	
PM for live calls	ClassicQMListenerRT, Oreka	

QueueMetrics-Live

This is the QueueMetrics-Live platform. Data is stored on the SQL database and is uploaded through Uniloader. Actions to be performed are sent back to Asterisk through the Uniloader and require custom dialplan added to the local PBX.

As there is no local audio storage, we suggest using a custom JSON audio recording platform for audio storage.

Platform feature	Suggested modules	Notes
Ingestor	Not needed (upload via uniloader)	
Storage	SQL/Cluster	
PBX Actions	CLASSIC_LIVE / DIRECTAMI_LIVE	Also enabled if CLASSIC and default.webloaderpbx=true
PM for recorded calls	JsonListener	
PM for live calls	ClassicQMListenerRT	

Freeswitch

The Freeswitch platform is able to receive rewritten data from a Freeswitch / FusionPBX system. Each QueueMetrics system analyzes one single tenant.

Platform feature	Suggested modules	Notes
Ingestor	Not needed (upload via `uniloader fsw`)	See Uniloader manual.
Storage	SQL	
PBX Actions	FREESWITCH / FREESWITCH_LIVE	
PM for recorded calls	JsonListener	
PM for live calls	ClassicQMListenerRT	

Enswitch

The Enswitch platform is able to receive data from an Integrics Enswitch 3.14+ system, perform actions on the system and listen to audio recordings.

Platform feature	Suggested modules	Notes
Ingestor	Not needed (upload via native loader)	
Storage	SQL	
PBX Actions	ENSWITCH	
PM for recorded calls	EnswitchPlatformListener	
PM for live calls	ClassicQMListenerRT	Will work.

MirtaPBX

The MirtaPBX platform is able to receive data from a MirtaPBX system, perform actions on the system and listen to audio recordings.

Platform feature	Suggested modules	Notes
Ingestor	Not needed (upload via native loader)	
Storage	SQL	
PBX Actions	MIRTAPBX	
PM for recorded calls	MirtaPbxListener	

JSON API

The JSON API platform is able to control an external platform through a unified API.

Platform feature	Suggested modules	Notes
Ingestor	Web	Usually Uniloader or custom integration
Storage	SQL	
PBX Actions	JSONAPI	
PM for recorded calls	ClassicXmlRpcRecordings	
PM for live calls	ClassicXmlRpcListenerRT	

Platform: PBX Actions

Each action driver allows a distinct set of features to be switched on, so not all actions are supported by all drivers.

Even when an action is supported an allowed by adriver, it is possible to disable it by setting the relevant property to `false`, like in:

```
platform.allows.addmember=false
```

If no property is set, or it is set to true, then the property will be shown if the underlying driver supports it.

A complete list of allowed actions that can be switched off on-demand can be found at Configuration properties for Platforms.

DirectAMI actions

In DirectAMI mode, actions performed on the PBX do not require a piece of dial-plan to be included; QueueMetrics will connect directly to the AMI port of your Asterisk system (as determined by *callfile.dir* and its *cluster.*.manager* equivalents in cluster mode) and will perform the actions you require.

This mode is slightly less flexible than the dialplan-based Classic mode, but:

- It is way easier to set up, as you just need to specify once the channel formats for your PBX
- It is more efficient when you need to join or leave multiple queues, as actions are batched into a single transaction

Tracking outbound

As it is not currently possible to originate outbound calls tracked by QueueMetrics without using the dialplan, if you need to do manual outbound you still need to include the custom *'extensions_queuemetrics.conf'* dialplan; QueueMetrics will then dial extension *28@queuemetrics* for manual outbound dialing.

Since Uniloader 0.4, it is possible to track outbound calls made by QueueMetrics without using the dialplan if you have outbound tracking enabled. This feature is not turned on by default and has to be set manually. This feature is at the moment experimental.



Outbound dialing is turned off by default and must be turned on manually if needed.

Configuration

To turn it on you need to set:

- *platform.pbx* to DIRECTAMI

The following configuration options are available:

- *platform.directami.verbose* whether to dump all AMI interactions to the system log (default: false)
- *platform.directami.agent* the way an agent channel is supposed to be (default: "Agent/${num}")
- *platform.directami.extension* the way an agent extension channel is supposed to be (default: "SIP/${num}")
- *platform.directami.localext* the channel used for chanspy (default: "Local/${num}@from-internal")
- *platform.directami.transfer* the transfer destination, in the form *ext@context* (default: "${num}@from-internal")
- *platform.directami.outbound.enabled* whether outbound dialing, though dialplan or Uniloader, is enabled (Default: false)
- *platform.directami.outbound.stanza* the point in the dialplan that should be called when doing outbound (Default: "28@queuemetrics")
- *platform.directami.outbound.usetracker* will use the Uniloader to track outbound call. (Deafult: false)
- *platform.directami.outbound.trackerdialout* the point in the dial-plan that will be used to generate outbound calls with the Uniloader (Default: "${num}@from-internal")
- *platform.directami.spyoptions* the additional options you want to use for the Asterisk application ChanSpy when listening to calls (Default: ""). By changing these options, you can have extra parameters added. The main "kind" of option (*b* for spy, *w* for whisper or *B* for barge) is added automatically depending on the kind of spying required.
- *platform.directami.cmdseparator* the separator used for parameters in your Asterisk version. Defaults to comma (",") and should be changed to pipe ("|") only for very old Asterisk systems (pre-1.8)
- *platform.directami.optimizedagentchannel* is the format of the agent channel when it is live and you want to spy on it. If blank, uses the same physical agent channel, based on the agent code or the extension code depending on whether you are in hotdesking mode or not. Defaults to blank. Please note that the `${num}` variable may contain the full agent name including its technology (e.g. "SIP/1234"), so you may want to use `${num:s}` to include only the numeric part.
- *platform.directami.stateinterface* is the interface to get the agent state from. This is useful when adding Local channels, where Asterisk is not aware whether they are in conversation or not. By specifying a SIP device, Asterisk can always know its correct state and act accordingly. If not specified, it is simply not sent to the queue.



In all channel configuration, the string `${num}` is replaced with the local agent code or extension, as appropriate. When adding or removing members from a queue, the string `${queue}` is expanded to the identifier of the current queue. Both placeholders accept a suffix syntax like `${num:3}` to include the string without the first three characters, `${num:s}` to capture what comes after the slash (if present) and `${num:p}` to capture what comes before the slash.

When running in cluster mode, you can have configuration that is specific for one single cluster member and it will override the general one - see Defining system properties for members of a cluster.

Examples

Example for a FreePBX system running without hotdesking enabled. The agent is logged on to the queue as "Local/XXX@from-internal", that is rewritten by QueueMetrics as "Agent/XXX", and matches a SIP extension with the same code. We specify a StateInterface so the queue is always aware of the state of the agent's phone.

```
default.hotdesking=0

platform.pbx=DIRECTAMI
platform.directami.agent=Local/${num}@from-internal
platform.directami.transfer=${num}@from-internal
platform.directami.outbound.enabled=true
platform.directami.localext=Local/${num}@from-internal
platform.directami.optimizedagentchannel=SIP/${num:s}
platform.directami.stateinterface=SIP/${num}
platform.directami.verbose=false
```

This is an example for a FreePBX system running with hotdesking enabled. The agent is logged on to the queue as "SIP/YYY" but appears in QM as "Agent/XXX" and a proper hotdesking record is written to match the two entities.

```
default.hotdesking=86400

platform.pbx=DIRECTAMI
platform.directami.agent=Agent/${num}
platform.directami.extension=SIP/${num}
platform.directami.transfer=${num}@from-internal
platform.directami.outbound.enabled=true
platform.directami.localext=SIP/${num:s}
platform.directami.verbose=false
```

This is an example for a FreePBX system running with hotdesking enabled but without dialplan. Uniloader must be running in "tracker" mode with "outbound" enabled.

```
default.hotdesking=86400

platform.pbx=DIRECTAMI
platform.directami.agent=Agent/${num}
platform.directami.extension=SIP/${num}
platform.directami.transfer=${num}@from-internal
platform.directami.outbound.enabled=true
platform.directami.outbound.usetracker=true
platform.directami.outbound.trackerdialout=${num}@from-internal
```

```
platform.directami.localext=SIP/${num}
platform.directami.verbose=false
```

Some Notes on Spying

When spying, QueueMetrics always tries to connect the supervisor to the agent's channel. If the live agent channel appears under a different name from the one used to create it (e.g. if the agent channel was a `Local/XXX@ctxt` that has been optimized to `SIP/XXX`), then you need to supply the parameter *platform.directami.optimizedagentchannel* in order to let QueueMetrics find the correct agent channel to spy on.

When spying, you can try different settings of *platform.directami.spyoptions* to configure additional behavior. The following options are added automatically:

- *b* (spy): Supervisor is able to listen to the on-going call, but cannot talk. This is the default.
- *w* (whisper): Supervisor is able to listen to the on-going call and can talk to the agent; caller does not hear supervisor.
- *B* (barge): Supervisor is able to listen and speak to all parties on the on-going call

DirectAMI Live actions

In DirectAMI Live, DirectAMI calls are not performed immediately but stored on the AMI queue and then performed by Uniloader on the receiving server. This means you can have the advantages of DirectAMI (no dialplan included for most use cases) on remote systems where actions are proxied by Uniloader.



To inspect the status of commands being sent to the remote system, see Checking the AMI Queue.

Configuration

To turn it on you need to set:

- *platform.pbx* to DIRECTAMI_LIVE

All other configuration options are the same as DIRECTAMI.

Classic PBX Actions

This is the default platform for QueueMetrics and it is able to perform all QueueMetrics interactions to the PBX. Actions are executing by calling the *extensions_queuemetrics.conf* dial-plan, that must be available on the target machine.

Depending on the target PBX, you may need to edit the supplied dialplan to use it.

Configuration

To turn it on you need to set:

- *platform.pbx* to CLASSIC (or just leave blank).

Available actions are controlled in the system preferences (see Configuring system preferences) by setting:

- *default.hotdesking* to a number > 0 to enable hotdesking

Each action is controlled by a stanza in the system preferences, for example to set call outcomes:

```
callfile.calloutcome.enabled=true
callfile.calloutcome.channel=Local/12@queuemetrics/n
callfile.calloutcome.extension=10
callfile.calloutcome.context=queuemetrics
```

More details are availble in Enabling Agent's page actions.

QMLive PBX Actions

In this mode, actions are performed by queueing them and waiting for *uniloader* to fetch and perform them. Actions are expressed as calls to the *queuemetrics* dialplan, that needs to be available on the target PBX.

This platform is used in QueueMetrics Live to allow remote actions.



To inspect the status of commands being sent to the remote system, see Checking the AMI Queue.

Configuration

To turn it on you need to set:

- *platform.pbx* to CLASSIC_LIVE
- or set *default.webloaderpbx* to *true* in Classic mode

Freeswitch actions

With the Freeswitch actions module, commands are sent to Freeswitch's *mod_callcenter* though the ESL API.

As in FusionPBX the id of *mod_callcenter* agents is a UUID, it is important to configure all agents so that their UUID is set in the *External reference* field. A list of agents with their UUIDs can be obtained by running `uniloader pbxinfo fusionpbx`.



The Freeswitch module will not enable any interactions if hotdesking is enabled.

Configuration

To turn it on you need to set:

- *platform.pbx* to `FREESWITCH`
- and make sure that:
  - your *callfile.dir* is set to `fsw:ClueCon@127.0.0.1:8021`, or whatever credentials you need for the Freeswitch ESL.
  - *default.hotdesking=0*
- *platform.freeswitch.addmember=true*
- *platform.freeswitch.tenant* is the domain name of the current tenant connected to this QM instance
- *platform.freeswitch.use_external_ref* whether to use the agent's own id or the "external reference" configured on the agent (default: true - use external reference)
- *platform.freeswitch.verbose* whether to dump all interactions to the system log (default: false)

If you want to have agents perform manual outbound:

- *platform.freeswitch.agentChannel* is the channel template to reach your agent, as specified in variable `num`. The id of the outbound queue is available as code `q` - for example to display it as the caller number when the agent phone rings. Make sure you set the FreeSwitch channel variable `qm_queue` to the ID of the outbound queue to be used for this call, otherwise it won't work. If this attribute is left empty, manual outbund is disabled.
- *platform.freeswitch.destinationNumber* is the number to be dialed, as specified in variable `num`. It is dialed as-if it was composed on the main dial-plan within the specified tenant.

The settings above only control manual outbound, that is the agent selecting an outbound queue and then entering a number manually (or using AMO to do the same thing). Automatic outbound, if needed, is performed automatically by Uniloader.

If you want to perform spy/whisper/barge:

- *platform.freeswitch.spychannel* is the format of the channel to reach the supervisor who wants to attach to a given call. Variable `num` is the extension of the supervisor, `q` the queue on which the call being spied on currently is, and `a` the agent code.
- *platform.freeswitch.spycmd* is the command to be executed when *spychannel* connects. The variable `spymode` stands for the correct code for application eavesdrop, while `callid` is the uniqueid of the call that is to be spied.



For this to work, you must make sure that you run Uniloader 24.11.3 or newer. If you run an earlier version, use the *Earlier mode* described below.

Examples

Example for a FusionPBX system running at address 192.168.7.5:

```
default.hotdesking=0
callfile.dir=fsw:ClueCon@192.168.7.5:8021
```

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

- On manual dial out, the agent's phone will display "QM-123" if the call is being made on outbound campaign "123"
- On chanspy, the agent's phone will initially ring with caller-id "Spy_q123_a321", where 123 is the queue that holds the call being spied, and agent 321 the current connected agent.
- To dial extension 1234, in both cases it will be dialled channel `user/1234@mytenant.myserver.com`

Configuration - earlier mode



At the moment, only actions for agent login, logoff and pause are supported. Also, as agent presence is unique and not related to specific queues, on the agent page only a "ALL" queue will be shown.

To turn it on you need to set:

- *platform.pbx* to `FREESWITCH`
- and make sure that your *callfile.dir* is set to `fsw:ClueCon@127.0.0.1:8021`, or whatever credentials you need for the Freeswitch ESL.

The following configuration options are available:

- *platform.freeswitch.use_external_ref* whether to use the agent's own id or the "external reference" configured on the agent (default: true - use external reference)
- *platform.freeswitch.verbose* whether to dump all interactions to the system log (default: false)



The Freeswitch module will not enable any interactions if hotdesking is enabled.

Examples

Example for a FusionPBX system running at address 192.168.7.5:

```
default.hotdesking=0
callfile.dir=fsw:ClueCon@192.168.7.5:8021

platform.pbx=FREESWITCH
```

```
platform.freeswitch.use_external_ref=true
platform.freeswitch.verbose=true
```

Freeswitch Live actions

In Freeswitch Live, Freeswitch API calls are not performed immediately but stored on the AMI queue and then performed by Uniloader on the receiving server.



To inspect the status of commands being sent to the remote system, see Checking the AMI Queue.

Configuration

To turn it on you need to set:

- *platform.pbx* to FREESWITCH_LIVE

All other configuration options are the same as FREESWITCH.

Examples

This example is for a system where Uniloader and Freeswitch are running on the same machine. Please note that the ESL address is relative to Uniloader.

```
default.hotdesking=0
callfile.dir=fsw:ClueCon@127.0.0.1:8021

platform.pbx=FREESWITCH
platform.freeswitch.use_external_ref=true
platform.freeswitch.verbose=true
```

JSON API Actions

This platform lets you control an arbitrary external system that implements a very simple JSON API. This way, QM can control any PBX system, and it is often used to interact with cloud providers that "abstract away" the specific PBX API.

For each interaction request, a simple JSON stanza like the one below is sent to an HTTP/s end-point you define as a parameter called `cmd`, using either HTTP POST or GET.

```
{
    "requestor": "Loway QueueMetrics 19.04.3",
    "token": "1234",
    "action": "ADDMEMBER",
    "login": "demoadmin",
    "serverId": "aleph",
    "agent": {
```

```
        "agentId": "101",
        "externalReference": null
    },
    "extension": "999",
    "targetExtension": null,
    "queues": [{
        "queueId": "111",
        "penalty": "UNKNOWN",
        "externalReference": null
    }, {
        "queueId": "112",
        "penalty": "UNKNOWN",
        "externalReference": null
    }],
    "callId": null,
    "message": null
}
```

QueueMetrics expects no specific answer to the call, as long as it succeeds with an HTTP status 200, as any requested action will be reflected in the `queue_log` data being uploaded.



For example, imagine an agent that wants to join a queue. They send a call for action ADDMEMBER with the correct details. As soon as the PBX processes the call, an ADDMEMBER event for the agent appears on the `queue_log`; at this point QueueMetrics updates the agent status to show that they joined the queue.

Configuration

An example configuration might be like the one below:

```
platform.pbx=JSONAPI
platform.jsonapi.url=https://api.your.pbx.com/yourendpoint/
platform.jsonapi.method=POST
platform.jsonapi.token=1234
platform.jsonapi.enabled=ADDMEMBER,REMOVEMEMBER,PAUSE,UNPAUSE
platform.jsonapi.verbose=true
```

The meaning is:

- `platform.jsonapi.url`: is the URL end-point that will be called. Must be specified.
- `platform.jsonapi.method`: either `GET` or `POST` (default: POST)
- `platform.jsonapi.token`: a token used to identify the specific QueueMetrics instance. Default: empty.
- `platform.jsonapi.enabled`: a comma-separated list of which actions are possible through the API. You do not need to implement them all, just the ones you specify. Default: none.
- `platform.jsonapi.verbose`: whether the request an payload will be written on QM's system logs. Default: false.

Actions

- AGENTLOGIN - Agents starts working. Queues are controlled externally.
- AGENTLOGOUT - Agents stops working.
- ADDMEMBER - Agents joins one or more queues.
- REMOVEMEMBER - Agent leaves one or more queues.
- PAUSE - Agents starts a pause. The pause code is passed as `message`.
- UNPAUSE - Agent stops the pause.
- CALLOUTCOME - Agents sets a call outcome for a specific call. The outtcome code is passed as `message`.
- ADDFEATURECODE - Agent adds a feature code to a call. The feature code is passed as `message`.
- REMOVEFEATURECODE - Agent removes a feature code.
- CUSTOMDIAL - Agent dials out a specific number on a given queue.
- SENDTEXT - Send SIP SMS to an extension.
- SOFTHANGUP - Supervisor wants to close a call.
- TRANSFER - Supervisor transfers a call to a new extension/queue
- SPY_INBOUND - Supervisor wants to start a spy session on an incoming call.
- SPY_OUTBOUND - Supervisor wants to start a spy session on an outbound call.
- WRITE_INFO_VAR - Agent wants to write an arbitrary variable to a call.

You will usually implement either AGENTLOGIN/AGENTLOGOUT or ADDMEMBER/REMOVEMEMBER-style presence for your agents.

JSON request details

In the JSON request, fields have the following meaning:

- `requestor`: the specific version of QueueMetrics sending this request
- `token`: an auth/identification token you use to distinguish from which QM system the request is coming
- `action`: one of the actions above
- `login`: the login of the user starting this action
- `serverId`: on clustered QM system, to which machine in the cluster this request belongs to. You can set up the `cluster.XXX.platform.jsonapi.token` property to override the default token for server XXX.
- `agent`: the agent to which this action belongs. Contains fields `agentId` and `externalReference`
- `extension`: the extension we are referring to
- `targetExtension`: an extension that is the target of this action (e.g. spying) or a variable name for WRITE_INFO_VAR
- `queues`: an array of queues this action refers to. Each queue defines a `queueId`, a `penalty` (if

known to QM), and an `externalReference`

- `callId`: the Unique-ID of the call this action refers to
- `message`: a free text message, e.g. the feature code or the pause code this action needs. For features codes, if the feature code contains a text entry as well, our message field appears in the format `ftrcode|message`, otherwise if the text entry is blank or missing, just the feature code. For variables, the contents of the variable.

When parsing the JSON request, you can expect all the fields above to be present, and must accept any other fields that may be added to the request in the future. The JSON block has always the same shape, and unused parameters for the current action are set to `null`.



External reference ids are not currently returned.



When testing specific action, it is often useful to point QueueMetrics to a service like Requestbin, so you can easily track outgoing requests and see which fields are set and how.

Enswitch actions

The Enswitch actions module will connect over HTTP/HTTPS to an existing Enswitch 3.14+ system and perform actions.

The following actions are supported:

- Setting outcomes and call features
- Hanging up and transfering calls from the RT page
- Spying inbound calls from the RT page
- Logging-on and logging-off agents to queues
- Pause and unpause agents

Configuration

To turn it on you need to set:

- *platform.pbx* to `ENSWITCH`
- *default.hotdesking* to 0 (very important - if not so all actions will be disabled and an error will be logged)

The following configuration options are available:

- *platform.enswitch.http.baseurl* is the main URL for the Enswitch system
- *platform.enswitch.http.login* an API user log-in
- *platform.enswitch.http.password* an API user password
- *platform.enswitch.guiaddmember* whether ADDMEMBER/REMOVEMEMBER actions are available (default: false - only chanspy is enabled and actions must be done through the agent's

phones)

- *platform.enswitch.verbose* whether all Enswitch interactions should be logged in full to the standard log (default: false)
- *platform.enswitch.lookbackClosedCalls* for how long a recently-closed call should be searched on CDRs (default: 7200 seconds)
- *platform.enswitch.spyPrefix* the default prefix to use to initiate a chanspy on an inbound call (Default: *91)
- *platform.enswitch.bargePrefix* the default prefix to use to initiate a barge-in on an inbound call (Default: *92)
- *platform.enswitch.qpriority.main* the queue priority for an agent being logged in with priority MAIN (default: 8)
- *platform.enswitch.qpriority.wrap* the queue priority for an agent being logged in with priority WRAP (default: 6)
- *platform.enswitch.qpriority.spill* the queue priority for an agent being logged in with priority SPILL (default: 4)
- *platform.enswitch.openLogons* whether agents have an "open" logon or must already be present on the queue. Default: false
- *platform.enswitch.dialoutPrefix* the prefix used to implement dial-out. Use placeholder `$Q` to add a queue-id (see below)
- *platform.enswitch.dialoutSpyPrefix* the default prefix to use to initiate a chanspy on an outbound call (Default: *91)
- *platform.enswitch.dialoutBargePrefix* the default prefix to use to initiate a barge-in on an outbound call (Default: *92)
- *platform.enswitch.dialoutClid* a caller-id to be used as presentation for outbound calls. If none is specified, the default one is used. Note that if an incorrect one is specified, all calls will fail.

To listen to audio recordings, you also need to turn on the EnswitchPlatformListener (see Pluggable Listener for the Enswitch platform).

Selecting chanspy vs barge-in

As QueueMetrics has only one "spy" action available, the choice of whether it is a "normal" spy or a barge-in depends on the number you enter as your local extension.

If the extension is prefixed with a "!", then a barge is attempted using the default prefix defined in *platform.enswitch.bargePrefix*; otherwise the prefix specified in *platform.enswitch.spyPrefix* is used.

Prefixes are set as in the `Features -> Feature prefixes` menu of your Enswitch system.

Open and Closed log-on modes

The default integration mode for Enswitch uses a "closed" log-on mode; this means that QueueMetrics cannot log an arbitrary agent on to Enswitch, but will basically toggle the pause status of an existing agent, albeit logging the event "as if" it was a real log-on/log-off or a pause/unpause event.

Weird as this sounds, this means that an administrator can configure with certainty which agents are supposed to work on a queue, and end-users cannot change them - they can just toggle them on and off.

If you set `platform.enswitch.openLogons=true`, a log-in will instead mean that an extension with the same agent code and `dtype=phone` will be added to the queue, with a priority as defined on the QM queue definition. A log-off will remove such extension from a queue.

Outbound mode

In order to use outbound, you need to create a special extension prefix that will take a few digits for the queue id and then the number to be dialled, for example let's say our prefix is `*787`, then tree digits for our queue, then the number to be dialled - e.g. eg `*787 146 123456` will dial number `123456` on queue `146`.

Then queue `146` must be a real Enswitch (inbound) queue, that we will use as a placeholder to make sure that only active agents (currently logged on, not puased), can actually dial out.

Agents will then:

- select an outbound queue from the GUI. Only outbound queues that they are currently logged on can be selected. This means that you can control the visibility of queues to agents though each queue's visibility key
- enter a number of at least two digits and click on the phone icon to initiate a call

When this happens, a call is sent to the agent's phone; as soon as they pick it up, a second leg to the callee is dialled and all queue log events are to be recorded in real-time as the call progresses.

The outbound call is displayed on the agent's panel as it happens, and also on the real-time screen. As the agent need to be logged on to an outbound queue to make an outbound call, there is no risk of misalignment between their log-in status and calls made; and the queue will be existing in Enswitch (though not actually used to process incoming calls).

Also, from the real-time page and the wallboard, it is possible to start a **chanspy** of such an outbound call.

A possible configuration could be:

```
platform.pbx=ENSWITCH
default.hotdesking=0
platform.enswitch.guiaddmember=true
platform.enswitch.verbose=true
platform.enswitch.http.baseurl=https://my.pbx....
platform.enswitch.http.login=user
platform.enswitch.http.password=password
platform.enswitch.dialoutPrefix=*787$Q
platform.enswitch.dialoutSpyPrefix=*91
platform.enswitch.dialoutBargePrefix=*92
platform.enswitch.dialoutClid=
```

```
platform.enswitch.audioSearchMargin=500
```

Note how the `$Q` placeholder allows adding the queue id within the outbound prefix. This also allows having only one default outbound queue, by not adding the placeholder in the generated number.

For example, if you are dialling number 1234 on queue 101 with `platform.enswitch.dialoutPrefix` set to...

- `*787$Q` would dial `*787 101 1234` - normal node (spaces added for clarity)
- `*787` would dial `*787 1234` - no queue id, all outbound logged on a queue you define
- `*787$Q999` would dial `*787 101 999 1234`

Generated queue_log

When QueueMetrics works with *Agent/1234* on Enswitch:

- It creates artificial logs for *Agent/1234*
- It adds/removes/toggles presence for an Enswitch extension *1234* having `dtype=phone`
- The extension itself will most likely log its calls under *SIP/1234@enswitch* or something similar

Therefore it is important in the agent definition that your agent be called *Agent/1234* with an alias of *SIP/1234@enswitch* so that it will work properly.



Enswitch extensions are often long numeric codes, and agents may have a hard time remembering that their login is *Agent/770102319*. To work around this issue, make sure you configure their email address in the agent's user profile - it can then be used in place of their login and it's easier to remember.

Another difference in generated queue_log is that the PAUSEALL verb will appear not just once but for all queues, that might in turn emit a local pause event. This causes no problem to QueueMetrics, and is caused by the fact that any log entry on Enswitch must belong to an existing queue. So you may see duplicated records at the same time-stamp.

MirtaPBX actions

The MirtaPBX actions module will connect over HTTP/HTTPS to an existing MirtaPBX system and perform actions.

The following actions are supported:

- Setting outcomes and call features
- Hanging up and transfering calls from the RT page
- Logging-on and logging-off agents to queues
- Pausing and Unpausing agents on queues
- Dialing out on specific queues

MiRTA also supports out-of-the-box outbound call tracking to QueueMetrics on a default queue.

INFO: Remember that each queue you want tracked on QM must be set up manually, and queues that are not won't be sending data to QM.

Configuration

To turn it on you need to set:

- *platform.pbx* to MIRTAPBX
- *default.hotdesking* to 0
- *platform.mirtapbx.baseurl* to your MirtaPBX url (e.g. https://my-mirta-pbx.com/)
- *platform.mirtapbx.key* to your MirtaPBX tenant key (e.g. a234Dadf4dfsdg4)
- *platform.mirtapbx.tenant* to your MirtaPBX tenant (e.g. TENANT1)

So it would look like:

```
platform.pbx=MIRTAPBX
default.hotdesking=0

platform.mirtapbx.baseurl=https://my.mirta-server.pbx/
platform.mirtapbx.key=a234Dadf4dfsdg4
platform.mirtapbx.tenant=TENANT1

platform.mirtapbx.verbose=true

audio.server=it.loway.app.queuemetrics.callListen.listeners.MirtaPbxListener
```

This same configuration is shared by the audio driver that can easily access recordings with no extra steps - see here.

Manual Outbound Calling in MirtaPBX

Agents can make manual outbound calls directly from the agent page by clicking the "Dial" button and selecting from available outbound campaigns in the dropdown menu. This also means that both AMO campaigns and recall tasks work as expected.

For QueueMetrics to properly track an outbound call, all of the following conditions must be met:

- The outbound campaign must exist as a queue in MiRTA
- The agent must be a logged-on member of that queue
- The queue must be configured with type `outbound` in QueueMetrics' queue configuration
- The destination number must be dialed through a trunk (not a local extension)

If conditions 1 or 2 are not met, QueueMetrics will display an error message when the agent attempts to dial.



Best Practice: Separate Inbound and Outbound Queues

While MiRTA allows queues to handle both inbound and outbound calls, we recommend keeping them separate for better reporting and management in QueueMetrics. Instead of creating a single queue like "Sales," create two separate queues: "Sales In" and "Sales Out."

This is because you get:

- Improved Agent Visibility: Since agents must join queues, supervisors can instantly see which team members are handling inbound versus outbound calls.
- Accurate Performance Metrics: Separate queues allow you to track statistics independently, which is critical because e.g. inbound wait time is a key performance indicator for call centers, while outbound wait time is not applicable or controllable. Combining these metrics produces meaningless averages.

You can still view combined data by creating an aggregate queue called "Sales" that merges "Sales In" and "Sales Out" for comprehensive reporting. You could also have multiple outbound campaigns related to the same inbound queue, or vice-versa.

Platform: PMs to match Recorded Calls

These PMs are used to find audio recordings.



Most of these PMs (e.g. *LocalFiles*, *LocalFilesByDay*) read recordings directly from disk. QueueMetrics runs as its own dedicated user, which typically does not have permission to read files owned by the PBX — see Granting access to local recordings for how to fix this.

Using multiple PMs at once: MultiListener

Module name:	**MultiListener**
Full Java Path:	it.loway.app.queuemetrics.callListen.listeners.MultiListener
Properties used:	*audio.multi* lets you define a set of PMs to be queried for files (enter their names, separated by pipe) *audio.multi.** specifies the properties of each listener.
Available since:	1.7.0

This PM lets you query multiple PMs in the order you specify to look for the call you are looking for. A common scenario may be the following one:

- All calls are recorded to a local volume, e.g. */queues/audio*. This is where files just recorded are held.
- A nightly process compresses the files to MP3 and moves them to a large NAS device mounted under */mnt/nas*, where they are stored separated by day.

In order to retrieve calls, we want QM to first check in */queues/audio*; if nothing is found, then we will look under */mnt/nas/2010-11-23*. This can be implemented with the following configuration:

```
# define the PM and the search order
audio.server=it.loway.app.queuemetrics.callListen.listeners.MultiListener
audio.multi=loc|nas

# first PM: local calls
audio.multi.loc=it.loway.app.queuemetrics.callListen.listeners.LocalFiles
audio.multi.loc.default.monitored_calls=/queues/audio

# second PM: NAS storage
audio.multi.nas=it.loway.app.queuemetrics.callListen.listeners.LocalFilesByDay
audio.multi.nas.default.monitored_calls=/mnt/nas/%YY-%MM-%DD
```

What we do here is the following:

- We first define a MultiListener and tell it via the *audio.multi* property to actually query a PM called "loc" first and one called "nas" if nothing is found. You can have as many PMs as you need and you canset their names as you best see fit.
- We specify the PM to be used for "loc" in the *audio.multi.loc* property. Properties to be set for it are appended to the *audio.multi.PMNAME.* hierarchy, as we do in this example to set the *default.monitored_calls* property.
- As you can see, you can have multiple PMs of the same type as well as different, and ecah can have their own configuration properties.

If you use MultiListener within a cluster, and you the cluster has members `s1` and `s2`, you should use a configuration like the following one:

```
# define the PM and the search order
audio.server=it.loway.app.queuemetrics.callListen.listeners.MultiListener
audio.multi=loc|nas

# first PM: local calls for either server
audio.multi.loc=it.loway.app.queuemetrics.callListen.listeners.LocalFiles
audio.multi.loc.cluster.s1.monitored_calls=/mnt/server_1/current_calls
audio.multi.loc.cluster.s2.monitored_calls=/mnt/server_2/current_calls

# second PM: NAS storage for either server
audio.multi.nas=it.loway.app.queuemetrics.callListen.listeners.LocalFilesByDay
audio.multi.nas.cluster.s1.monitored_calls=/mnt/nas/s1/%YY-%MM-%DD
audio.multi.nas.cluster.s2.monitored_calls=/mnt/nas/s2/%YY-%MM-%DD
```

If you want, you could use the same folder for recordings of both servers (though it is better to keep them separate to speed-up access); still, they have to be specified for all servers in the cluster.

Plain old recordings: LocalFiles

Module name:	**LocalFiles**
Full Java Path:	it.loway.app.queuemetrics.callListen.listeners.LocalFiles
Properties used:	*default.monitored_calls* in a single-server environment, or *cluster.SERVER.monitored_calls* in a cluster
Available since:	1.4.7

This is the standard search method that comes with QueueMetrics. Basically, all directories under *default.monitored_calls* are explored recursively, and all audio files matching the Asterisk ID of the main call that was queued are retrieved. Therefore the call files found can be zero or more.

This PM is sub-optimal for very large call centres, where the cost of scanning through all recordings (maybe on remotely mounted disks) could take a significant time. If you are in such an environment, see the *LocalFilesByDay* entry.

This PM is used by default if no other server is specified in the system preferences (see Configuring system preferences).

Large storage with recordings: LocalFilesByDay

Module name:	**LocalFilesByDay**
Full Java Path:	it.loway.app.queuemetrics.callListen.listeners.LocalFilesByDay
Properties used:	*default.monitored_calls* in a single-server environment, or *cluster.SERVER.monitored_calls* in a cluster *audio.lookBack* for how many hours before or after midnight is a call considered a "borderline" case (default 4).
Available since:	1.4.7

This PM works exactly like the *LocalFiles* one, but allows using placeholders in the file path; this way, you can set the default recordings directory to handle only a subset of all recordings.

For example, if you set *default.monitored_calls* to */var/myrecordings/%YY-%MM/* when trying to listen to a call that was made on Jan 9, 2007 will expand to */var/myrecordings/2007-01/* therefore making the directory scanning much more manageable.

Valid placeholders include:

- %YY → the 4-digit year when the call was made
- %MM → the 2-digit month when the call was made
- %DD → the 2-digit day of month when the call was made
- %SE → in a clustered environment, the server name (all lower case)
- %QU → the queue name (all lower case)

Though this is unlikely, it is possible that a call gets recorded on a given day and then gets queued on a different day, e.g. for calls that happen around midnight. QM handles this case by double-checking all calls within a boundary of *n* hours from the midnight in both the days that are divided by that midnight. This behaviour can be set using the *audio.lookBack* property.

Asterisk can easily adapt to recording files in a way that is compatible with this storage model, like e.g.:

```
. . . .
exten => 999,n,Set(MONITOR_FILENAME=/audio-nas/${STRFTIME(${EPOCH},,%Y-%m/%d)}/call-
${UNIQUEID}.wav)
exten => 999,n,Queue(778,t,,)
. . . .
```

Will store audio files as:

```
/audio-nas/2011-03/10/call-123456.7890.wav
```

The nice part is that Asterisk will automatically create missing directories, as needed.

Pluggable Listener for the Enswitch platform

Module name:	**EnswitchPlatformListener**
Full Java Path:	it.loway.app.queuemetrics.callListen.listeners.EnswitchPlatformListener
Properties used:	* `platform.enswitch.audioSearchMargin` The search margin, in seconds (default: 3600). * Plus, the same properties used for the Enswitch PBX platform to connect - see Enswitch actions
Available since:	16.10.9

This listener is able to retrieve and download calls through the Enswich 3.13+ API. Shares its configuration with the Enswitch Actions module.

To find a call, it goes looking for all the calls recorded that are "around" the time of beginning of the call to be retrieved, so e.g. with a margin of one hour, it starts searching one hour before the beginning of such call and ends two hours (two times the margin) after the beginning.

The width of this "margin" is controlled by `platform.enswitch.audioSearchMargin` - on busy systems, it might be advisable to set it to a lower value (e.g. 1000 seconds) as to avoid retrieving hundreds or thousands of calls.

Pluggable Listener for the MiRTA PBX platform

Module name:	**MirtaPbxListener**
Full Java Path:	it.loway.app.queuemetrics.callListen.listeners.MirtaPbxListener

Module name:	MirtaPbxListener
Properties used:	The same properties used for the MiRTA PBX PBX platform - see MiRTA PBX actions
Available since:	20.xx

This listener is able to retrieve and download calls through the MiRTA PBX API.

Shares its configuration with the MiRTA PBX Actions module.

Query an external system: JsonListener

Module name:	JsonListener
Full Java Path:	it.loway.app.queuemetrics.callListen.listeners.JsonListener
Properties used:	- *audio.jsonlistener.url*: the URL that will be queried - *audio.jsonlistener.method*: the HTTP method to use (GET or POST; default: POST) - *audio.jsonlistener.searchtoken*: a token that authorizes/identifies the system. - *audio.jsonlistener.verbose*: whether HTTP connections must be logged. Default: false.
Available since:	20.xx

This listener will query an external server, sending in a JSON payload that identifies the call and expecting a result for each media file that is attached to it. If there are no recordings, the server is expected to return an empty list of results.

What this module does is:

- Connects to the *url* you specified, using the *method* you specified.
- Sends a JSON payload (see below) that specifies a call to be checked for recordings.
- Expects zero or more results of recordings (audio, video or other) found for that call

The payload that is sent us similar to the one below - it is sent as parameter `query` of a web form - this way the same semantics is used for both GET and POST forms (though POST has a content-type of `application/x-www-form-urlencoded`):

```
{"searchToken": "TOK1234",
 "callStartTst": 1546765179,
 "direction": "inbound",
 "agent": "agent/6039",
 "callId": "1544818293.8579",
 "callServer": "",
 "queue": "2004",
 "requestor": "demoadmin",
 "clientIp": "10.10.3.123"}
```

That is interpreted as:

- `searchToken`: is the token you specified in the configuration. This can be used to grant access to your server, and/or to identify separate QM servers
- `callStartTst`: the time-stamp when a call started in QM. This will be close but not necessarily exactly the same time when recordings started
- `direction`: inbound or outbound. Might not be accurate on all calls.
- `agent`: the agent code
- `callId`: the unique identifier of the queue leg of the call
- `callServer`: if QM is monitoring a cluster, the PBX on which the call was processed
- `queue`: the queue identifier
- `requestor`: the log-in of the user requesting this operation
- `clientIp`: the remote IP address of the requestor

And the results must be a list of zero or more JSON objects, like the below:

```
[
 {"name": "My call 1544818293.8579.mp3",
  "url": "https://audiostore:9999/audio/a_mp3.mp3?unqid=1544818293.8579",
  "type": "MP3",
  "size": "1234"},
 {"name": "Suopevisor Chat for 1544818293.8579",
  "url": "https://audiostore:9999/chat/?unqid=1544818293.8579",
  "type": "OTHER",
  "size": ""}
]
```

For each object:

- `name`: is a human-readable name that will be shown to the user
- `url`: is a URL where the audio/media can be retrieved from. You may want to include security features in it, e.g. time or ip-based limits
- `type`: a tag specifying the contents of this recording (see below)

- `size`: a string describing the file (currently not used).

Allowed `type` tags are:

- `MP3`: MP3 - will use the Audio player if enabled
- `WAV`: Wav file - will use the Audio player if enabled
- `GSMWAV`: A GSM file in a WAV container
- `OGG`: Ogg Vorbis - will use the Audio player if enabled
- `M4A`: AAC audio - will use the Audio player if enabled
- `AUDIO`: Other audio
- `VIDEO`: Any video
- `OTHER`: Any other attachment to be opened in the browser



Whe developing a new service, note that if if the service fails to run, it will report no error to the user, as if there were no audio files. Still, detail on the errors will appear on QM's log.

External audio recorder: OrekaWeb

Module name:	**OrekaWeb**
Full Java Path:	it.loway.app.queuemetrics.callListen.listeners.OrekaWeb
Properties used:	* *oreka.jdbcUrl* points to the server where the OrekaWeb database is stored. Firewalls and MySQL user setup must allow a JDBC connection coming from the QueueMetrics server. * *oreka.sipHeader* is the name of the tag to be tracked in the Oreka system. If missing, it's *X-Unique-ID*. * *oreka.web* is the URL of an OrekaWeb application - QM uses Oreka's applets for video playback. * *oreka.playersize* lets you set the size of the player, e.g. "1024x780" * *oreka.mode*: set to "1" if running Oreka up to minor build number 2494, to "2" if newer.
Available since:	1.5.1

This PM lets you offline all the audio recording to an Oreka system - see http://oreka.sourceforge.net/

This PM lets you playback audio (and optionally video) of recorder calls stored in Oreka. In order to listen to live calls, it is possible to use either some Asterisk-based method, e.g. *ClassicQMListenerRT* below, or an Oreka-based methods like *OrekaWebRT* below.

It needs the JDBC URI to point to the Oreka database; the database must contain the following tables: *orktag*, *orksegment*, *orktape*, *orkservice*, *orktagtype*.



In order to have QueueMetrics associate the Asterisk call-ids correctly, you must configure Asterisk and Oreka to store the call-id of the main leg of the call, the one upon which the Queue() command is called.

Propagating the SIP header

As Oreka is a passive recording solution based on SIP, and the call's UniqueId is used to match a call in QueueMetrics, it is necessary for you to add the UniqueId information to the SIP headers.

If/how this can be done depends on the kind of channels you have as members of the queue.

If you have static or dynamic SIP phones as members of the queue, e.g.

```
[myQueue]
....
member => SIP/1234
member => SIP/1235
```

you can simply use the following piece of dialplan:

```
....
exten => s,n,SIPAddHeader(X-Unique-ID: ${UNIQUEID})
exten => s,n,Queue(myQueue|t|30)
....
```

If instead you have other types of channels as members of the queue, e.g.

```
[myQueue]
....
member => Agent/101
member => Local/102@agents
```

then you need to store the UniqueID in an inherited variable, e.g.

```
...
exten => 411,2,Set(__MASTERID=${UNIQUEID})
```

```
exten => 411,3,Queue(myQueue|t|30)

[agents]
exten => _XXX,1,SipAddHeader(X-Unique-ID: ${MASTERID})
exten => _XXX,2,Dial(SIP/${EXTEN}|300)
```

This makes it possible to use Oreka in all common usage scenarios.

Configuring event capture in Oreka

You need to modify OrkAudio's *config.xml*, under the <*VoIpPlugin*> section:

```
<SipExtractFields>X-Unique-ID</SipExtractFields>
```

And restart OrkAudio.

Which version of Oreka do I need?

The minimal software you can use seems to be the commercial version (Orecx TR). This includes G729 Codec and Live Monitoring.

Video playback



As the original Java applet does not work anymore in modern browsers, video playback is not supported any longer. Documentation remains for historical purpouses.

Orecx is able to capture and store along with the audio recording of the call a screen capture of the agent's workstation while the call was made. The importance of such a feature is obvious.

If a video recording is present for a given call, then the audio file will be followed by the string "[vid]" to show that it's a joint audio and video recording.

In order to play it back, QM will not stream it through a browser but will open up the VNC player that ships with OrkWeb; therefore you must configure the *oreka.web* property. The applet is not used in case of audio-only recordings.

Advanced Oreka support: OrekaEncrypted

Module name:	**OrekaEncrypted**
Full Java Path:	it.loway.app.queuemetrics.callListen.listeners.OrekaEncrypted

Module name:	OrekaEncrypted
Properties used:	* *oreka.jdbcUrl* points to the server where the OrekaWeb database is stored. Firewalls and MySQL user setup must allow a JDBC connection coming from the QueueMetrics server. * *oreka.sipHeader* is the name of the tag to be tracked in the Oreka system. If missing, it's *X-Unique-ID*. * *oreka.web* is the URL of an OrekaWeb application - QM uses Oreka's applets for video playback. * *oreka.playersize* lets you set the size of the player, e.g. "1024x780" * *oreka.username* and *oreka.password*: the account used to access OrekaWeb * *oreka.mode*: set to "1" if running Oreka up to minor build number 2494, to "2" if newer.
Available since:	12.04
Requires:	OrkWeb 1.4-2178 or newer

This PM is an advanced version of the *OrekaWeb* module, and it offers the same functionalities plus a few additional ones:

- Support for encrypted Oreka calls: calls can be stored in an encrypted format and will be decrypyed dinamically by Oreka. The PM may handle encrypted and unencrypted contents at the same time.
- Support for audio-only playback through the Oreka player (a new link will let you open the player as well as download the file as was possible in earlier versions)
- Support for tags: call tags are passed to the Oreka player, and you can use the player to move back and forth between them
- QM acts as a proxy for all OrekaWeb contents

The same set-up instruction apply as per the OrekaWeb PM.

Secure access

The OrekaEncrypted PM has QM act as a secure proxy for all Oreka contents:

- The OrekaWeb server can be invisible to the user (e.g on a private network);

- There are no more limitation for cross-domain downloading
- There is a double security check; first, when a file is requested, the proxy checks that this file belongs to the list of audio files that the current user just searched; then, QM will authenticate to OrkWeb and, if successful, will try and stream the file back to the client.
- For additional security, any audio/video file is streamed through a small content buffer that is constantly overwritten and that is immediately cleaned after usage; it is never written to disk on the QM server.

If you turn on encryption and authenticated downloads on the Oreka system, and use HTTPS to connect to QM, the result is a very secure audio server for your Asterisk system.

(Obsolete) Using an external server: ClassicXmlRpcRecordings

Module name:	**ClassicXmlRpcRecordings**
Full Java Path:	it.loway.app.queuemetrics.callListen.listeners.ClassicXmlRpcRecordings
Properties used:	*default.audioRpcServer* (non-clustered) or *cluster.SERVER.audioRpcServer*: The address of the XML-RPC server implementing the QMAudio.findStoredFile interface.
Available since:	1.4.7

This is the standard XML-RPC implementation and makes it easy to create a completely custom scheme to handle recordings. The output of this function must be a single URL that can either stream the audio file or launch a player to stream that call. This is completely user-configurable.

The details of how to write an XML-RPC server for the *QMAudio.findStoredFile* interface can be found on the XML-RPC guide for QueueMetrics. We ship a sample implementation of such a server in the *xmlrpc_audio_server.php* server that comes with QueueMetrics.

See also section Enabling XML-RPC call listening and streaming.

(Obsolete) Pluggable Listener for Enswitch Platform Integration



This module is obsolete. Please use Pluggable Listener for the Enswitch platform instead.

Module name:	**Enswitch**
Full Java Path:	it.loway.app.queuemetrics.callListen.listeners.thirdparty.Enswitch

Module name:	Enswitch
Properties used:	*enswitch.serverclass* is the full qualified name of an external class code implementing the interface specified on next paragraph. We expect to have an external implementation resulting in a .class files (IntegrationExample.java in this example) stored in a suitable jar and/or classpath available to QueueMetrics. *enswitch.server* used to provide authentication on the remote server. *enswitch.user* used to provide authentication on the remote server. *enswitch.password* used to provide authentication on the remote server.
Available since:	12.5.4-418

Exposed Interface

QueueMetrics expects to have in his classpath an implementation of the following interface:

```
public interface EnswitchRecording {
        public String[][] getUrls(String host, String username, String password,
String callID);
}
```

The getUrls should return a matrix containing the list of URLs associated to the specified callID. Each row in the matrix specifies:

- The URL at column 0
- The name (usually shown as clickable link by QueueMetrics) at column 1

(Obsolete) Pluggable Listener for CallCabinet Platform Integration



This listener was removed from QM 20 onwards.

Module name:	CallCabinetForQm
Full Java Path:	it.loway.app.queuemetrics.callListen.listeners.CallCabinetForQmListener

Module name:	CallCabinetForQm
Properties used:	* `default.callcabinet.customer_id` used to provide authentication on the remote server. * `default.callcabinet.site_id` used to provide authentication on the remote server. * `default.callcabinet.api_key` used to provide authentication on the remote server.
Available since:	18.04.1 up to 20.xx

To make sure QueueMetrics looks for your recordings on your CallCabinet online repository, five parameters must be set.

- Your CallCabinetForQmForQM Customer ID.
- Your CallCabinetForQm Site ID.
- The Valid CallCabinetForQm API Key.
- The CallCabinetForQm Listener
- The HTML5 Audio Player has to be enabled (optional)

And you must make sure that the recordings are saved on CallCabinet with the CustomerInternalRef field set to the Asterisk Unique ID of the call used by QueueMetrics.



Please note that CallCabinet for QueueMetrics requires a special key, and won't work with normal API keys.

Go to Edit System Parameters and edit the following parameters:

```
audio.server=it.loway.app.queuemetrics.callListen.listeners.CallCabinetForQmListener
default.callcabinet.customer_id=xxxx
default.callcabinet.site_id=xxxx
default.callcabinet.api_key=xxxx
audio.html5player=true
```

Make sure you replace the "xxxx" characters with your Customer ID, Site ID and API Key. The `audio.html5player=true` parameter lets us to listen to the recordings directly on our browser.

Go back to the HomePage and take a look at any report containing some of the calls that are recorded.

Queue details

Date	Caller	Queue	IVR	Wait	Duration	Pos.	Disconnection	Handled by	Attempts	Code	Stints	Srv
10/24 - 11:13:14	201	300	0:00	0:02	0:06	1	Caller	200	1			
10/24 - 12:03:58	201	300	0:00	0:01	0:04	1	Agent	200	1			
10/24 - 12:04:42	201	300	0:00	0:01	0:06	1	Caller	200	1			
10/24 - 14:08:20	201	300	0:00	0:01	0:02	1	Agent	200	1			

Export as... Current page: 1 / 1

Clicking on the Call Detail Icon (the magnifying glass icon on the right), will show at the bottom that QueueMetrics retrieves the recordings related to that call's Call ID (Asterisk Unique Call ID).

Call Detail - Google Chrome

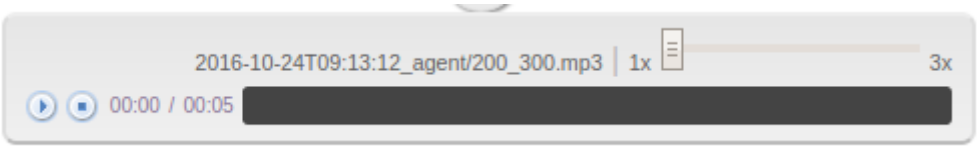
localhost:8080/QueueMetrics/qm/popup_call.jsp

Caller ID:	201
Handled by:	agent/200
Duration:	6 sec.
Time in IVR before queueing:	0 sec.
Waiting time:	2 sec.
Original position	# 1
Disconnection cause:	Caller disconnected
Transferred to:	
Attempts:	1
Last Failed Attempt:	-
Bridged Channel:	1477300394.9
Stints:	1
URL:	
Status code:	
Tag:	
Srv	
DNIS	
IVR selection	
- 2016-10-24T09:13:12_agent/200_300.mp3	

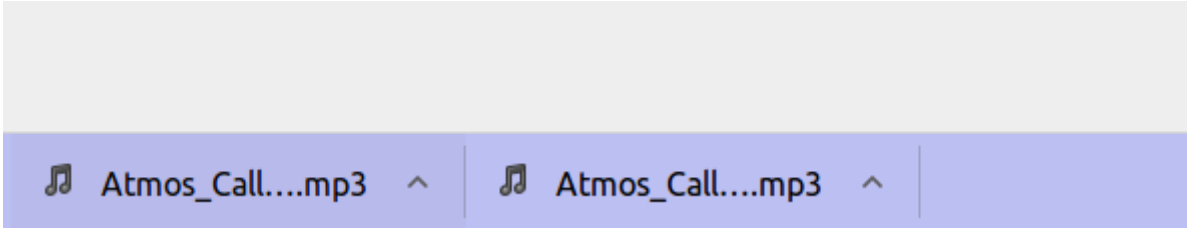
The name format QueueMetrics uses to represent a recording is the following:

```
DateTime_agent_queue.mp3
```

Clicking on the Play icon just right of the recording name will stream the recording directly without downloading it.



To download it instead, one must click on the recording name.



Cluster Mode

If QueueMetrics is working in Cluster Mode, it can manage recordings stored on different CallCabinet Sites.

This is because Cluster Mode allows the user to define different System Parameters for different PBXs.

we can define properties specific to each PBX Server, by using the following syntax:

- cluster.servername.callcabinet.customer_id=xxxx
- cluster.servername.callcabinet.site_id=xxxx
- cluster.servername.callcabinet.api_key=xxxx

So, by using the prefix “cluster.ServerName.SystemParameter”, where ServerName is the name of the PBX (as defined in the cluster.servers system parameter) and SystemParameter is the name of the System Property to be set (e.g. “default.callcabinet.site_id”), without the “default.” prefix, we can allow the same properties to have different values depending on the server we are currently monitoring.

Platform: PMs to match Live Calls

These PMs are used to listen to live calls.

Live calls through QueueMetrics: ClassicQMListenerRT

Module name:	**ClassicQMListenerRT**
Full Java Path:	it.loway.app.queuemetrics.callListen.RTlisteners.ClassicQMListenerRT
Properties used:	**For listening to inbound calls:** callfile.monitoring.channel, callfile.monitoring.extension, callfile.monitoring.context; **For listening to outbound calls:** callfile.outmonitoring.channel, callfile.outmonitoring.extension, callfile.outmonitoring.context; **In a single-server environment:** callfile.dir (points to a local call-file directory or a manager interface port); **In a clustered environment:** cluster.SERVER.manager (points to each Asterisk server's manager interface port)
Available since:	1.4.7

This is the standard QM behaviour: when listening to inbound or outbound calls, a popup appears and asks for a local extension. That local extension is connected to the live channel so that the local user can listen to the ongoing call.

In order for this to work, the dial-plan on each Asterisk server must implement the correct logic - an example is given in the *[queuemetrics]* context that comes with QueueMetrics.

This PM is used by default if no other server is specified in the system preferences (see Configuring system preferences).

(Obsolete) Live calls through an external module: ClassicXmlRpcListenerRT

Module name:	ClassicXmlRpcListenerRT
Full Java Path:	it.loway.app.queuemetrics.callListen.RTlisteners.ClassicXmlRpcListenerRT
Properties used:	*default.audioRpcServer* (non-clustered) or *cluster.SERVER.audioRpcServer*: The address of the XML-RPC server implementing the QMAudio.listenOngoingCalls interface.
Available since:	1.4.7

This is the standard XML-RPC implementation and makes it easy to create a completely custom scheme to handle live monitoring. The output of this function must be a single URL that will launch a player to stream that call. This is completely user-configurable.

The details of how to write an XML-RPC server for the *QMAudio. listenOngoingCalls* interface can be found on the XML-RPC guide for QueueMetrics. We ship a sample implementation of such a server in the *xmlrpc_audio_server.php* server that comes with QueueMetrics.

See also section Enabling XML-RPC call listening and streaming

(Obsolete) Live calls through Oreka: OrekaWebRT

Module name:	OrekaWebRT
Full Java Path:	it.loway.app.queuemetrics.callListen.RTlisteners.OrekaWebRT
Properties used:	*oreka.rtserver* is the master property that tells QM if Oreka is clustered or not *oreka.web* is the URL of an OrekaWeb application - QM uses Oreka's applets for playback. *oreka.rtserver.xxx* is used for clustered configurations.
Available since:	1.5.2

This PM lets your supervisors monitor agents using a web-based interface provided by Oreka. The supervisors will simply click on a live call and it will be streamed to them through their browser (note: a window will open and will close immediately before the popup opens. This is expected behaviour).

In order for this PM to work, your system configuration must matche these criteria:

- You should be using Callback agents, where the agent extension is correctly filled in at logon time

- Pop-up windows should be openable by QM - this feature is disabled by default in most modern browsers.
- You should also select a way for this PM to choose on which Oreka server the call must be listened on.

Do not forget to set the *oreka.web* property in any case in order to download the playback applet.

Using only one Oreka server

If you are deploying only one Oreka server, you should set the address of the live listening port by setting *oreka.rtserver* to *fixed* and then entering the live streaming port as follows:

```
oreka.rtserver=fixed
oreka.rtserver.address=http://hostname:59120/?type=stream&localparty=#AGENTEXT#
```

This will work even on a clustered system, as long as there is only one Oreka server. Note how the agent extension is expanded in the string (see below for the full list of expansion tokens).

Using a cluster of Oreka servers

If you have a set of Oreka servers (likely because you have a cluster of Asterisk servers), you can associate a separate Oreka server to each box in the cluster. You do so by setting *oreka.rtserver* to *cluster* and then entering the live streaming port for each member of the cluster, as follows:

```
oreka.rtserver=cluster
oreka.rtserver.aleph=http://ork_aleph:59120/?type=stream&localparty=#AGENTEXT#
oreka.rtserver.beth=http://ork_beth:59120/?type=stream&localparty=#AGENTEXT#
```

In this example, all calls processed on server "aleph" will be searched on server "ork_aleph", while all calls processed on server "beth" will be processed on server "ork_beth".

Using multiple Oreka servers with UniqueID

If you have a set of Oreka servers that are not linked one-by-one to a set of Asterisk boxes, you can associate a separate Oreka server to each call in the cluster, by prepending a digit to the call's UniqueID that will be used to know on which server each call is being handled. You do so by setting *oreka.rtserver* to *chandigit* and then entering the live streaming port for each member of the cluster, as follows:

```
oreka.rtserver=chandigit
oreka.rtserver.1=http://ork_aleph:59120/?type=stream&localparty=#AGENTEXT#
oreka.rtserver.2=http://ork_beth:59120/?type=stream&localparty=#AGENTEXT#
```

In this example, all calls which UniqueID starts wilth "1" will be handled by the "ork_aleph" server, and all calls which UniqueID start with "2" will be queries on the "ork_beth" server.

Expanded properties

The following properties are expanded in the Oreka live listening URL:

- AGENTEXT is the numeric extension an agent is logged on from
- AGENTCODE is the code of the agent

Quality Assessment in QueueMetrics

QueueMetrics includes a Quality Assessment (QA) module that lets you:

- Define a set of metrics to be used for call grading
- Have the QA team grade calls while they're being processed or from historical recordings
- Run complete reports by queue and by agent

Enabling QA monitoring

In order to use QA monitoring, you should have the following security keys assigned:

- QA_TRACK: this key means that the person can input QA data. If this person has the keys to access historical calls or real-time calls, he will be able to fill-in QA forms. Individual forms can be further restricted by key-protecting them
- QA_REPORT: this keys means that the holder can access QA reports. Individual forms can be further restricted by protecting them with a reporting key as well.
- USR_QAEDIT: this key means that the holder can modify and create QA reporting forms.

Understanding Quality Assessment

The QA module in QueueMetrics was built in order for a specific QA supervisor to track the performance of agents on a given set of **metrics**. Each metric is expressed as a long description and has an unique **engagement code** (a short acronym up to 5 letters).

Metrics are user-definable and are clustered together in **forms**; a form can hold up to 130 metrics divided in up to 10 **metric groups**.

A single reporter can grade a call only once for each defined form; any attempt to grade a the same form for the same call multiple times will not be accepted.

For security reasons, **call grading data cannot be modified once input**, and forms with live data associated to them cannot be deleted from the system. In order to have a reduced set of metrics available if you use successive versions of a form over the period, it is possible to **close a form**, i.e. to avoid further input. Deletions, if any, will be performed at the database level by the system administrator.

Grading data is expressed as integer numbers between 0 and 100; grading all fields is mandatory, except for fields marked as "optional" in the form definition. The QA team can also input free text comments linked to a specific call.

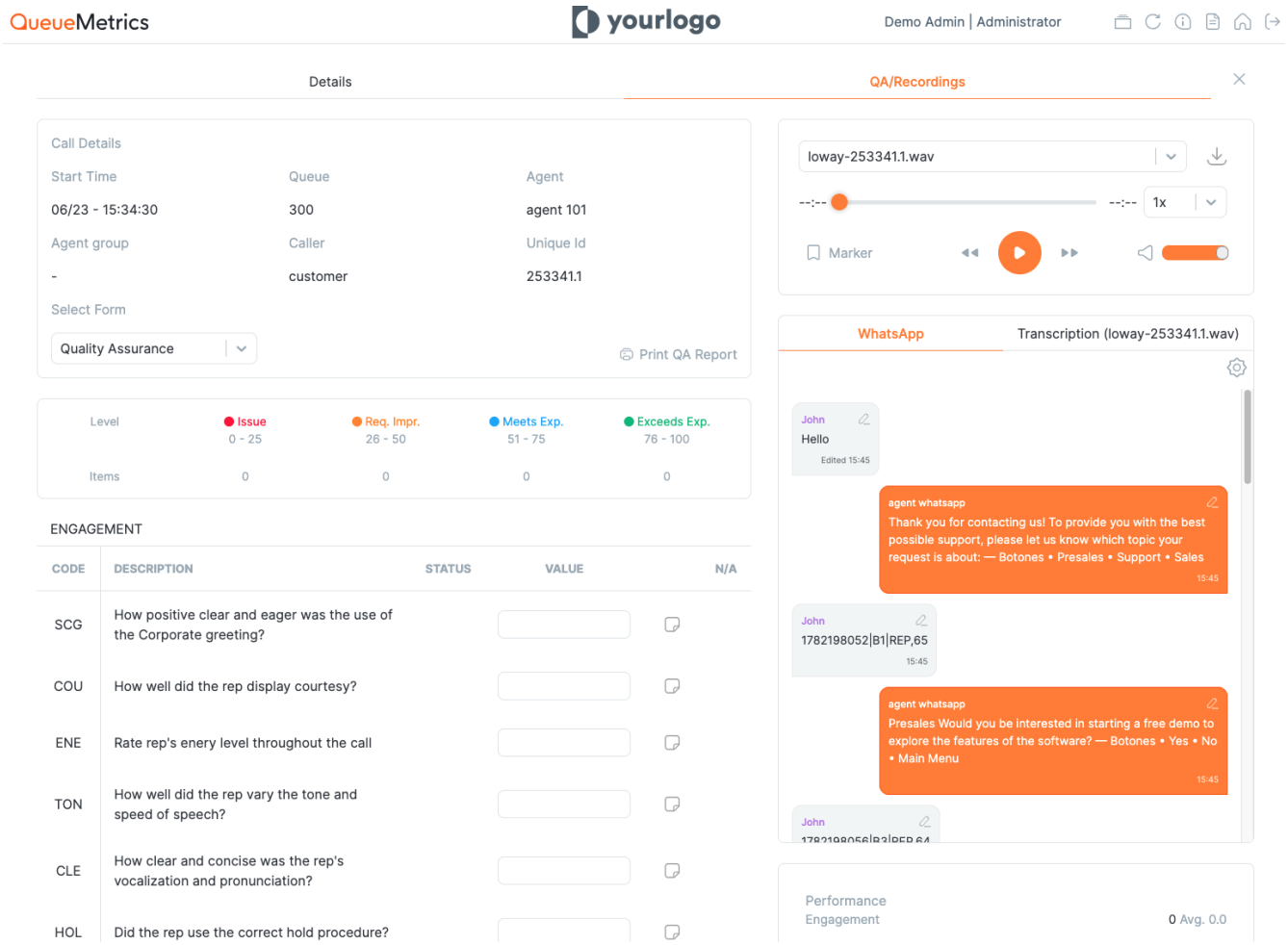
It is possible to edit **thresholds** for different **levels** of QA grades, e.g. 0-25: *Issue*, 26-50: *Improvement required*, 51-75: *Meets expectations*, 76-100 *Exceeds expectations*. These values can be defined on a form-by-form basis, and make it possible to count the number of items that belong to each category and to use a colour code for immediate graphic representation.

Grading calls

Grading data can be input while listening to the live call (Unattended monitoring) or while looking at the historical call details or through a particular formatted URL string.

Grading historical calls

In order to do the grading of historical calls, you proceed as is the case for audio recordings. If QA grading is enabled, the QA Form will appear in the QA tab, as in the picture below.



The input form

If multiple QA forms are available for this QA person, they will be able to select the correct form by selecting the "Input form" field on the top right.

The top-left box shows the current threshold values for each levels (please note that the label associated to each level is form customizable from the form configuration page. See Configuring QA forms for further information). While you input data into the form, you can see that the number of items that fall into each category and the average and total scores are updated in real-time.



For each section is reported the overall section score and average. Zero values are marked in red. This simplifies the discovering of session shortcuts (See Configuring QA items for further details) for already scored forms.

At the bottom of the form is the button **Save**, that is used to exit the form saving changes. This button will appear only when data are in a consistent state that allows saving.

If you load a saved form, it will be shown in read-only mode.

On the bottom part of the form there are the different items to be graded, grouped into a set of categories. If a box contains invalid data (i.e. something that is not a number between 0 and 100 included) it will be displayed in red and the form won't let you save.

Following the form definition, items can be graded by:

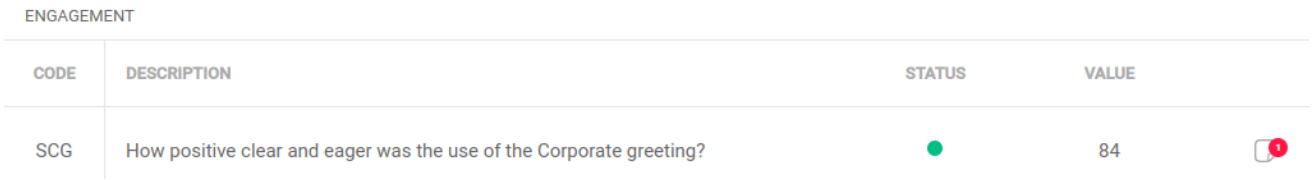
- inserting a score value
- selecting the appropriate value from the dropdown menu
- checking the proper Yes/No options.

Items that are not mandatory have an associated **N/A** checkbox; when checked, it disables the related score value and lets the user save a form without specifying any score for that item. If all fields within a given section are defined as **N/As**, then the Overall Performance will display the entire section with an N/A Average total value.

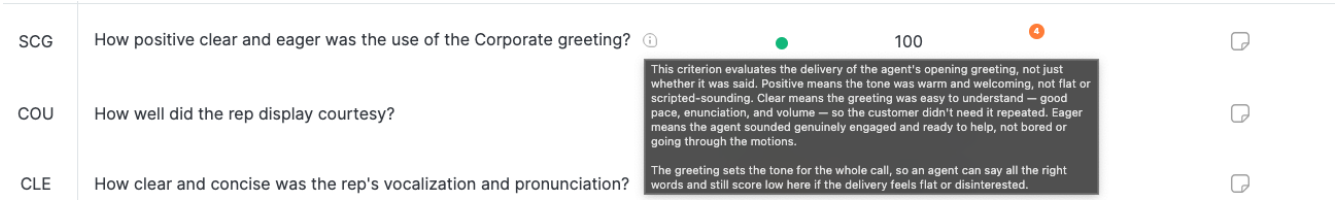
If an item is a failed **shortcut** item; that is, if a shortcut item totalizes a score that falls into the "Issue" category, the overall form score will be set to 0.

The value set in some items may control the set of items that are enabled for the current form; that's why the form is evaluated again after each user input.

Is possible to assign a free text note or a set of notes to each question in the form. We call this type of notes as "per-question" notes. Per-question notes could be added clicking on the icon on the right side of each question and typing the text in the shown popup. Questions with already associated per-question notes are marked with a different icon as per the following figure:

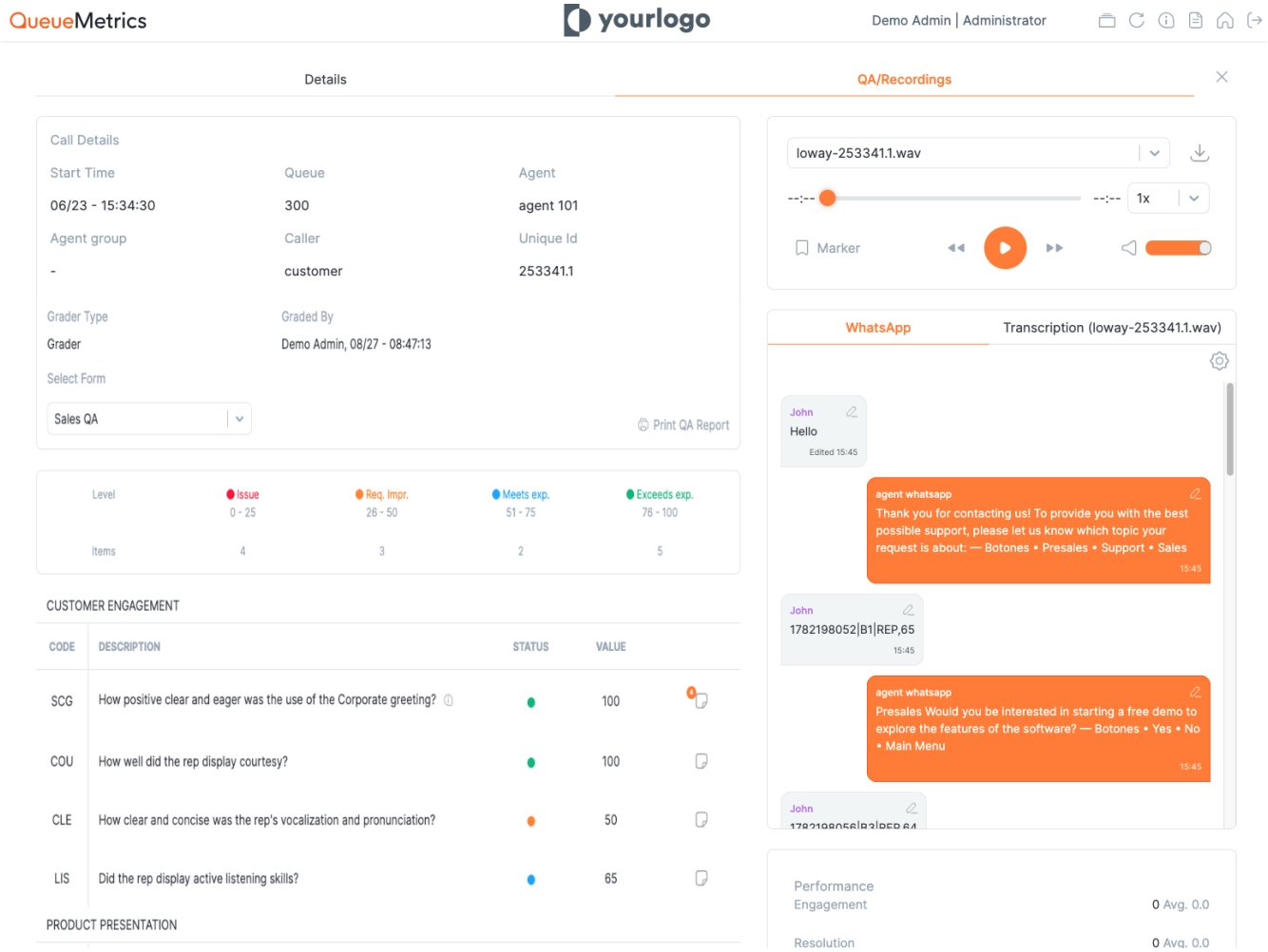


If a question has a deeper explanation, an info icon will be present at the end of the question. Hovering over the icon will show a tooltip where you can read that explanation.



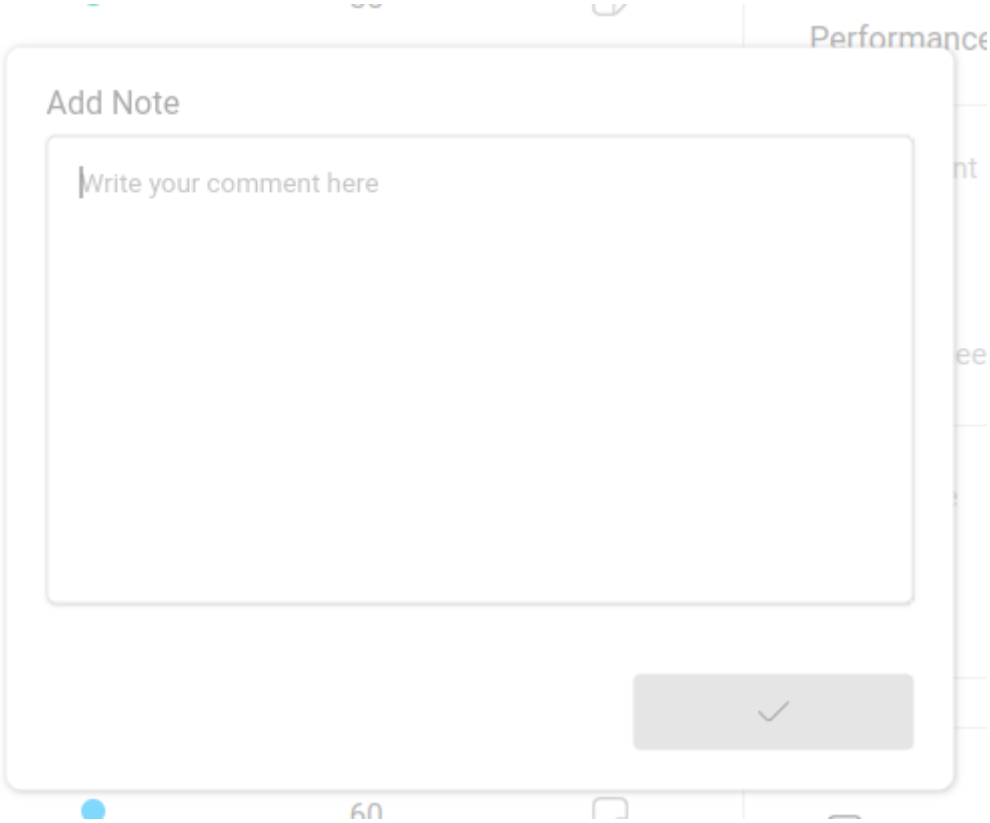
For a thorough description of how Forms and Items can be set up, please see the chapter Configuring QA forms .

When a form is saved, it appears as per the following figure:



It basically shows the same data that was input, but it cannot be changed anymore (except for users holding specific keys as defined by Removing or Editing QA forms) and the GraderType and who graded it and when is shown. If there are known audio recordings for this call, they are shown in the top-right.

Pressing the **Add Note** button, a modal will appear, letting you enter a note:



The user can insert a note that will be saved by pressing the "Add" button; all comments already added are listed in chronological order together with the per-question notes inserted for a set of specific form. Per-question notes associated to the selected form are shown at the beginning of the comments list.

For each call it is possible to add markers which can be created and deleted as required, in order to keep further details regarding that specific call. This works best in conjuction with the HTML5 audio player as described on the HTML5 Player.



Is not possible to submit partial forms.

Grading over HTTP access

Is possible to grade a particular call through an HTTP request to the QueueMetrics server with an URL specifically formatted for this purpose.

When an URL is typed in the browser, QueueMetrics redirects its output to the login page (if required) where the user could log in to continue.

QueueMetrics shows the grading input form in the browser window and the user can grade the call and/or add notes to it.

The URL to be used to trigger the grading procedure should follow the syntax below:

```
http://qmserver.corp:8080/queuemetrics/qm_qa_jumptogradepage.do?
   QAE_astclid=1286184814.122
   &QAE_queue=queuename
   &QAE_formName=FormToBeGraded
```

```
    &QAE_CallStartDate=2010-10-04.11:00:00
```

(of course the URL should appear all on one line).

In the example, we trigger a grading procedure on the host *qmserver.corp* on port 8080 The context is queuemetrics (but could change based on local install). Then there are some parameters following:

- **QAE_astclid** : specifies the Asterisk unique id for the call to be graded
- **QAE_queue** : specifies the queue name where the call has been taken
- **QAE_formName**: specifies the name of the form to be graded
- **QAE_CallStartDate**: specifies the day where the call has been taken. The value should be formatted as YYYY-MM-DD.hh:mm:ss and should represent the time before the call (it's not important to specify the exact time where the call has been taken but it's important to specify a time near the period before the call).

Removing or Editing QA forms

Users holding the key *QA_REMOVE* can delete a form.

When a form is deleted, their content is dumped on the Audit Log.

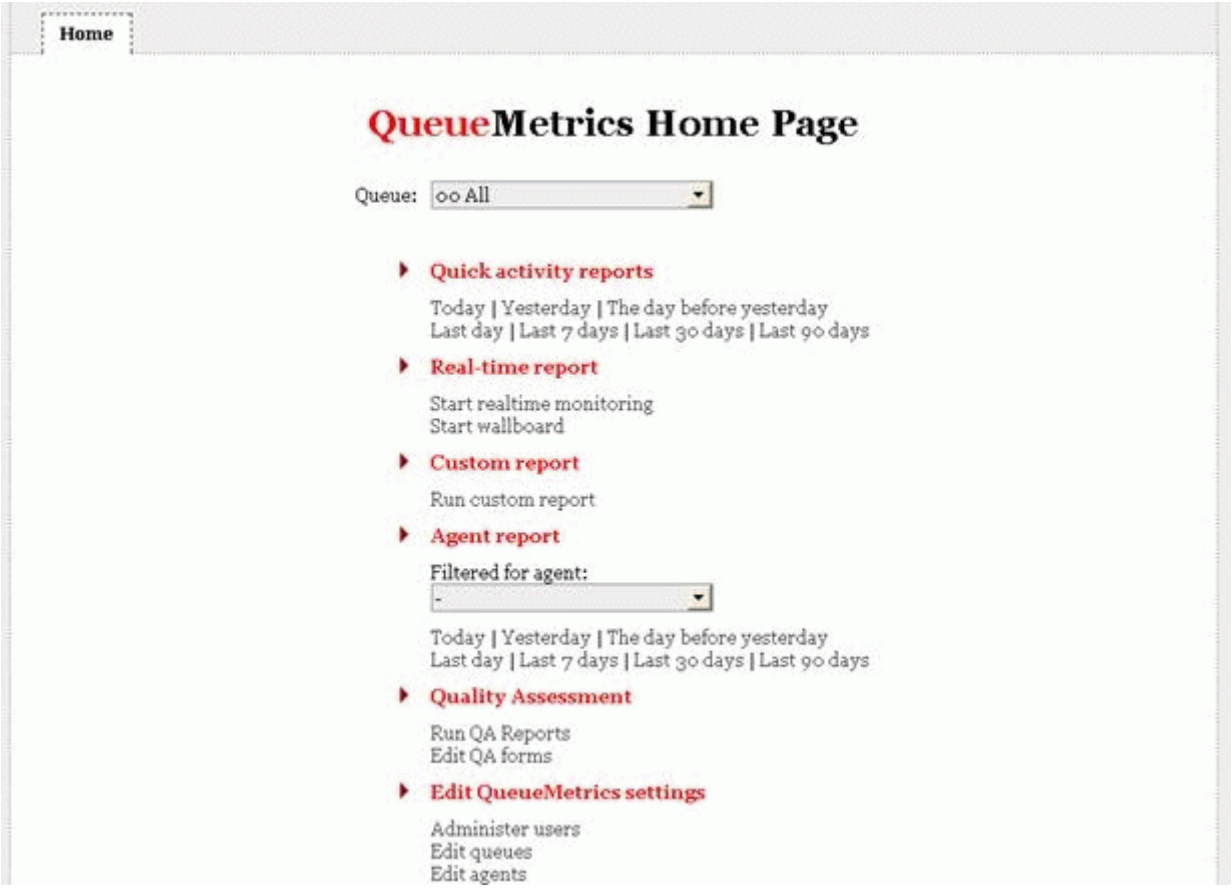
All accesses on deleted forms are highlighted by a special message shown in the form.

After deleting a form, it is again possible to grade a call as if it was never graded before.

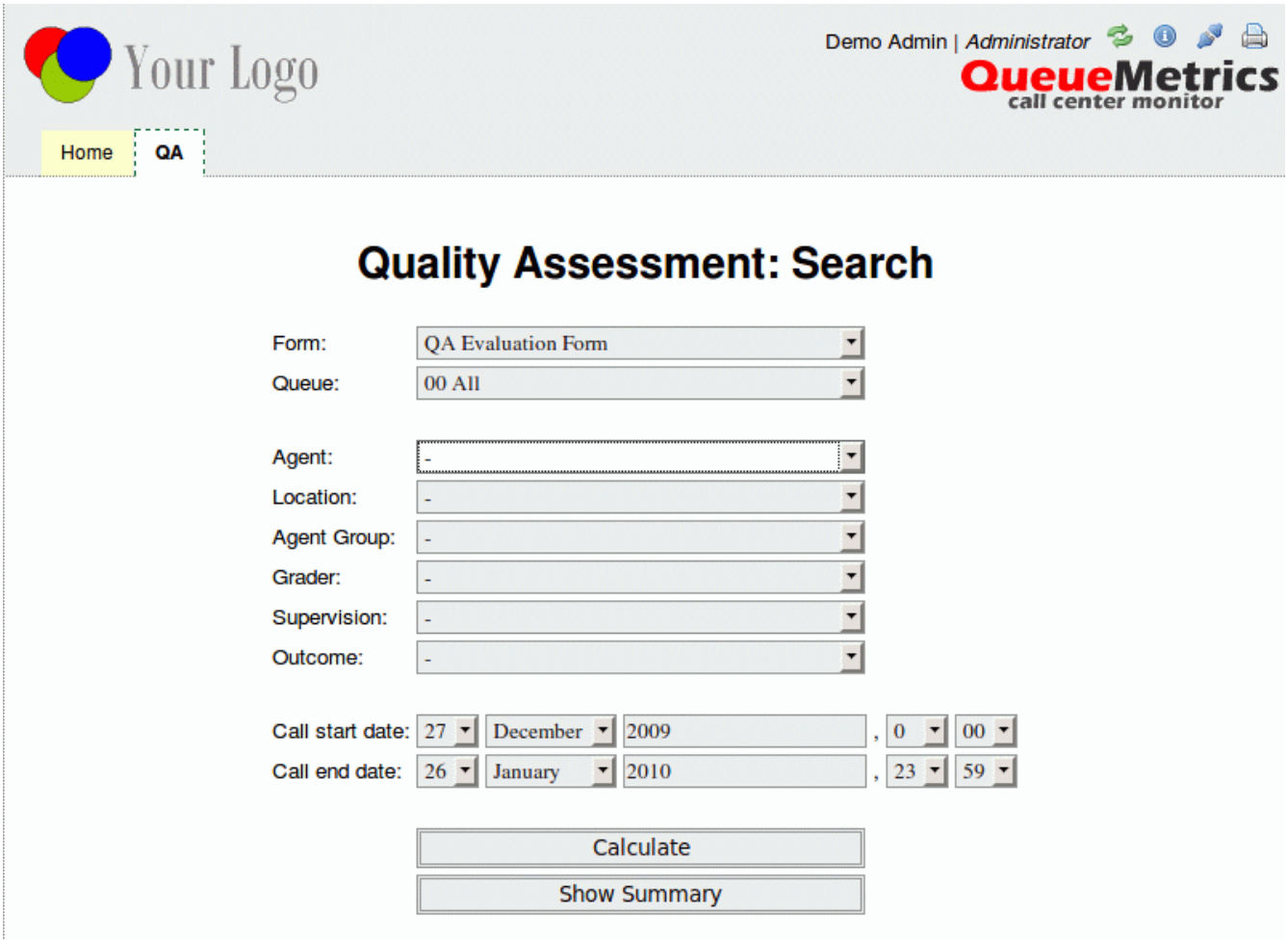
Users holding the keys *QA_REMOVE* and *QA_REPLACE* can edit an already submitted form. By editing an already submitted form, QueueMetrics performs a normal delete action, then shows to the user a new editable form with pre-filled values in each row. Data are processed by following the usual procedure.

Running QA reports

In order to run QA reports, you must go to the main page of QM and click on the "Run QA forms" label.



The system will show the following form:



The parameters have the following meanings:

- **Form** is the name of the form you want to run a report for
- **Queue** can be one or more queues. You can the run different reports for different queues, or use a catch-all queue
- **Agent** is an optional Agent filter
- **Location** is an optional Location filter
- **Agent Group** is an optional Agent Group filter
- **Grader** is an optional parameter that filters by the person who compiled the form
- **Supervision** is an optional Supervisor filter
- **Outcome** is an optional call outcome code filter
- **Asterisk call-id** is the unique-id of the call
- **Caller** is the caller's number (exactly as it appears on the caller-id, with no rewriting)
- **Start Date** and **End Date** are about the start time of the calls which QA forms that will be included in the report.

By clicking on "Calculate" or "Show Summary" the actual results are shown. If multiple search parameters are set, they must be all valid on the result set.

If you have used Extra Scores (see Chapter 20.9 - *Configuring QA forms*) within the QA form, the "Calculate" or "Show Summary" might return averages that are higher than 100.

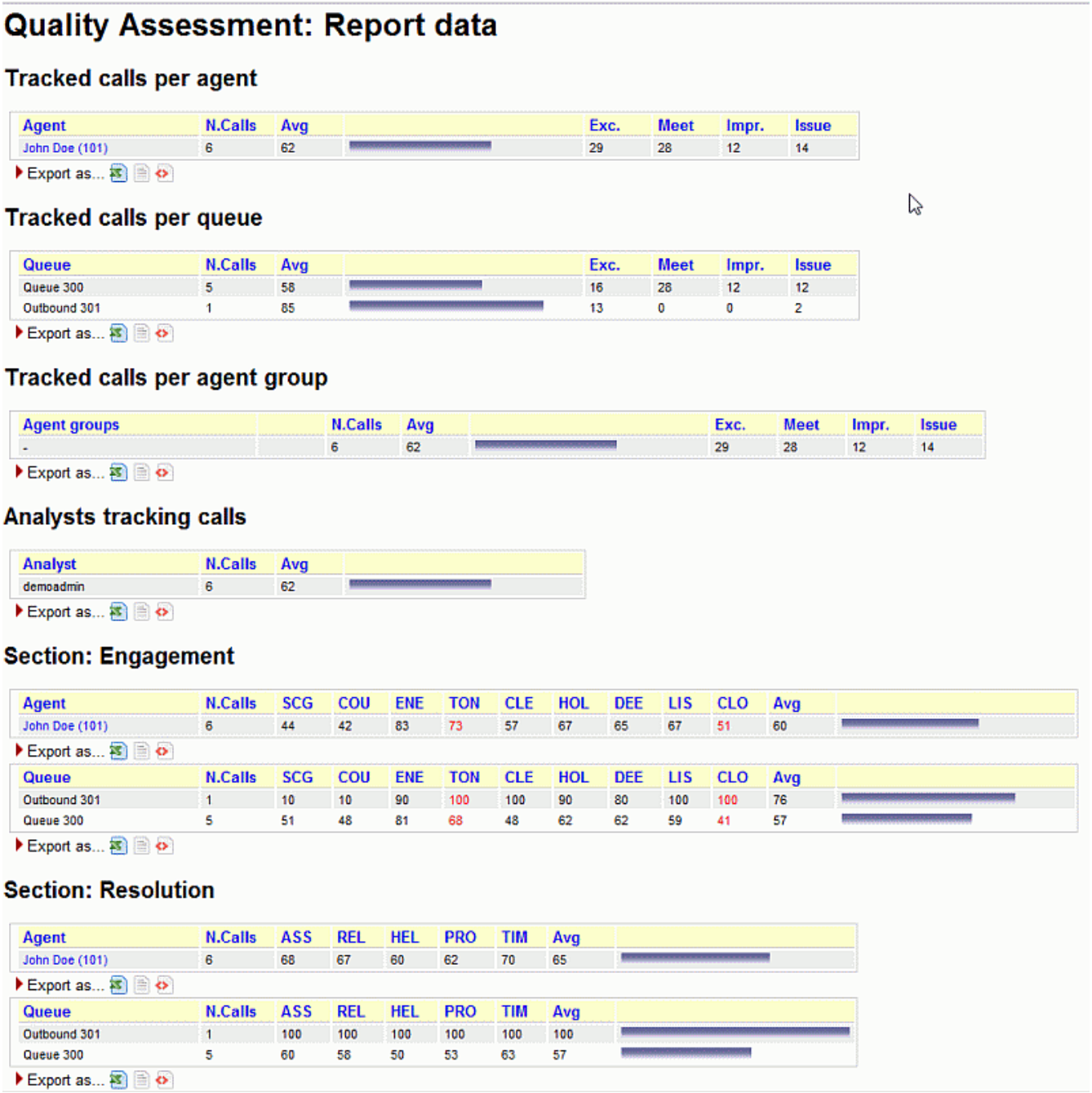
It is also possible to run a report that compares graders to each other - see Grader calibration reports.



The fields **Asterisk call-id** and **Caller** always consider your query as a case-insensitive substring to be matched, so e.g. if you have a call with a unique-id of *567890.1234*, entering *.1234* will be enough to match it. You can also use _ (underscore) to match any single unknown character or % (percentage) to match zero or more occurrences of any character.

The main QA report

The button "Calculate" shows a report like:



Quality Assessment: Report data

Tracked calls per agent

Agent	N.Calls	Avg		Exc.	Meet	Impr.	Issue
John Doe (101)	6	62		29	28	12	14

Export as...

Tracked calls per queue

Queue	N.Calls	Avg		Exc.	Meet	Impr.	Issue
Queue 300	5	58		16	28	12	12
Outbound 301	1	85		13	0	0	2

Export as...

Tracked calls per agent group

Agent groups		N.Calls	Avg		Exc.	Meet	Impr.	Issue
-		6	62		29	28	12	14

Export as...

Analysts tracking calls

Analyst	N.Calls	Avg	
demoadmin	6	62	

Export as...

Section: Engagement

Agent	N.Calls	SCG	COU	ENE	TON	CLE	HOL	DEE	LIS	CLO	Avg	
John Doe (101)	6	44	42	83	73	57	67	65	67	51	60	

Export as...

Queue	N.Calls	SCG	COU	ENE	TON	CLE	HOL	DEE	LIS	CLO	Avg	
Outbound 301	1	10	10	90	100	100	90	80	100	100	76	
Queue 300	5	51	48	81	68	48	62	62	59	41	57	

Export as...

Section: Resolution

Agent	N.Calls	ASS	REL	HEL	PRO	TIM	Avg	
John Doe (101)	6	68	67	60	62	70	65	

Export as...

Queue	N.Calls	ASS	REL	HEL	PRO	TIM	Avg	
Outbound 301	1	100	100	100	100	100	100	
Queue 300	5	60	58	50	53	63	57	

Export as...

The **Tracked calls per agent** report shows:

- The total number of calls that were tracked for each agent
- The average score for each agent
- The total number of items that fall into "Exceeds expectations", "Meets expectations", "Improvement required" and "Issue" for each agent.

As you can see, the names of each agent are clickable in order to obtain a detail of calls by agent.

All statistics that are computed per-agent are then recomputed per-queue and per-agent-group.

The **Analysts tracking calls** reports shows how many calls each supervisor graded and what was the average score that this supervisor gave.

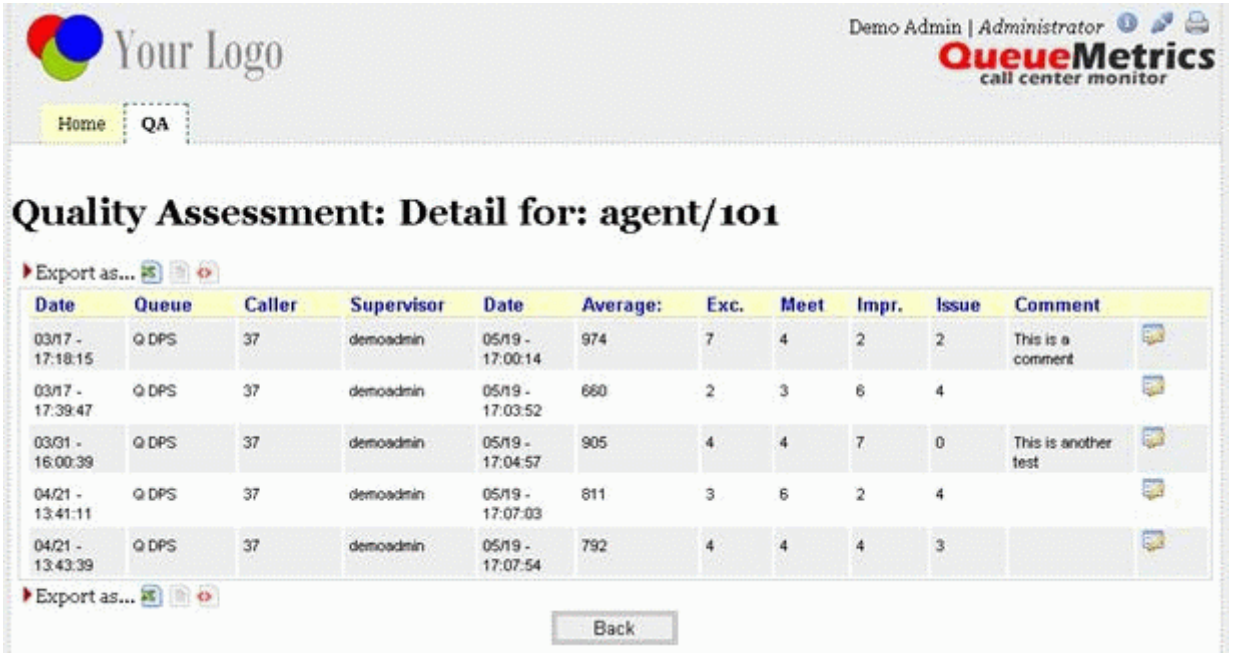
Then, for each Section defined in the QA form, you will get the average scores for each item, plus an

average of all average scores in order to point out problems.

If an item is shown in red, it means that such item has been assigned a zero-weight value. If an item is shown in gray, it means that such item has been set as non scoreable item. For further information on configuring items within the form, refer to the paragraph Configuring QA items.

All columns can be sorted by clicking on the item name and all data can be downloaded in Excel, CSV or XML format.

In order to have a better understanding of what is going on, you can click on an agent's name and get the details, as below:

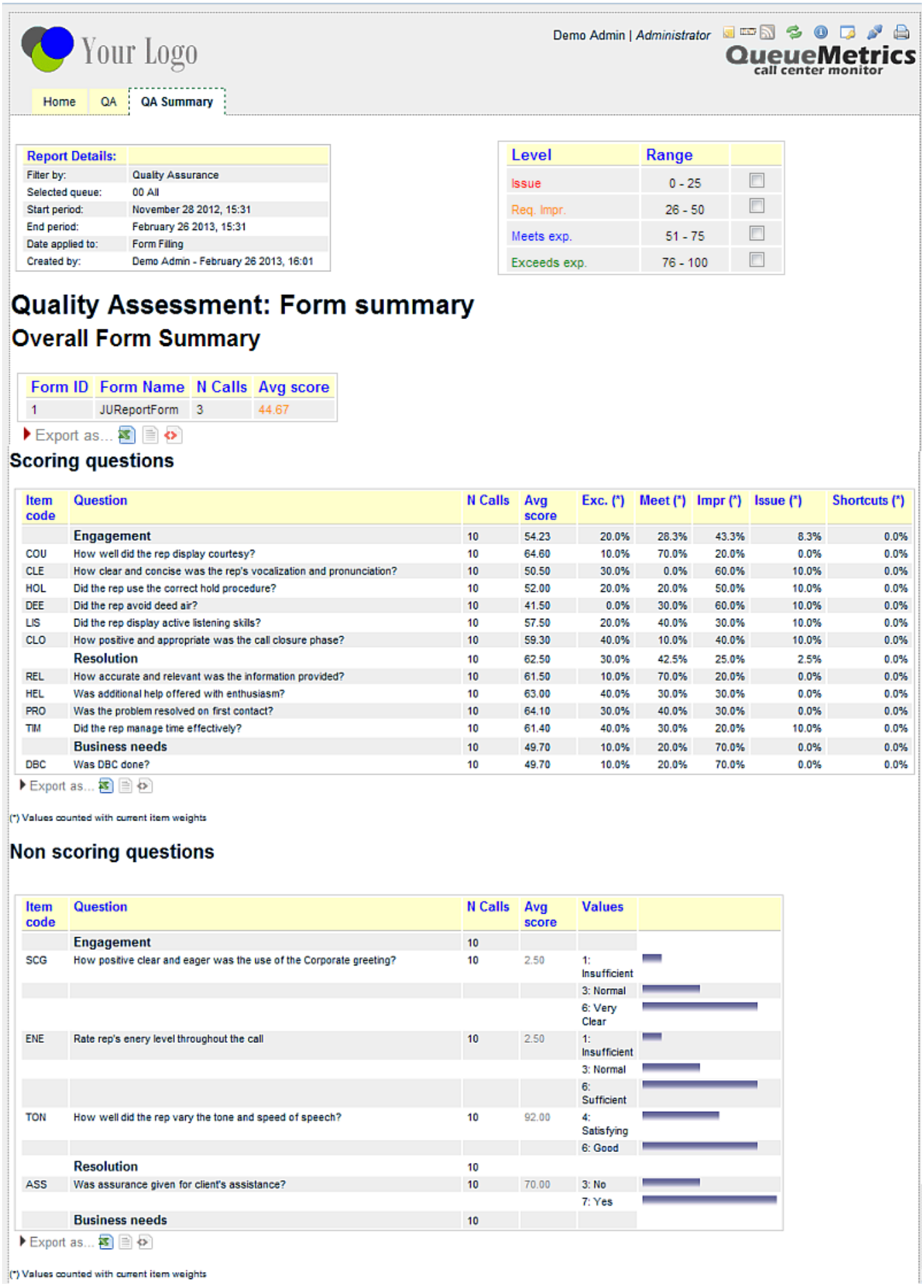


This shows the details of all calls stored, the number of items for each call that fall into each grading category, the average rating for each call and the comment.

By clicking on the form icon on the right, you can access the QA form that was graded for this calls, so you can access individual scores and listen to audio recordings that are related to this form.

The QA Summary report

The "Show Summary" button shows a report like:



This report calculates, for each item and for each section in the form:

- average for the whole form

- number of calls graded
- average score and cumulated percentage for each item and section that "Exceeds expectations" (the column marked "Best") or "Meet expectations" (Good) or "Improvement required" (Ok) or "Issue" (Req.Imp.) (only for scoreable item)
- average score and number of time each value is found in the reports (only for non scoring questions)
- cumulated percentage for each item marked as "shortcut"

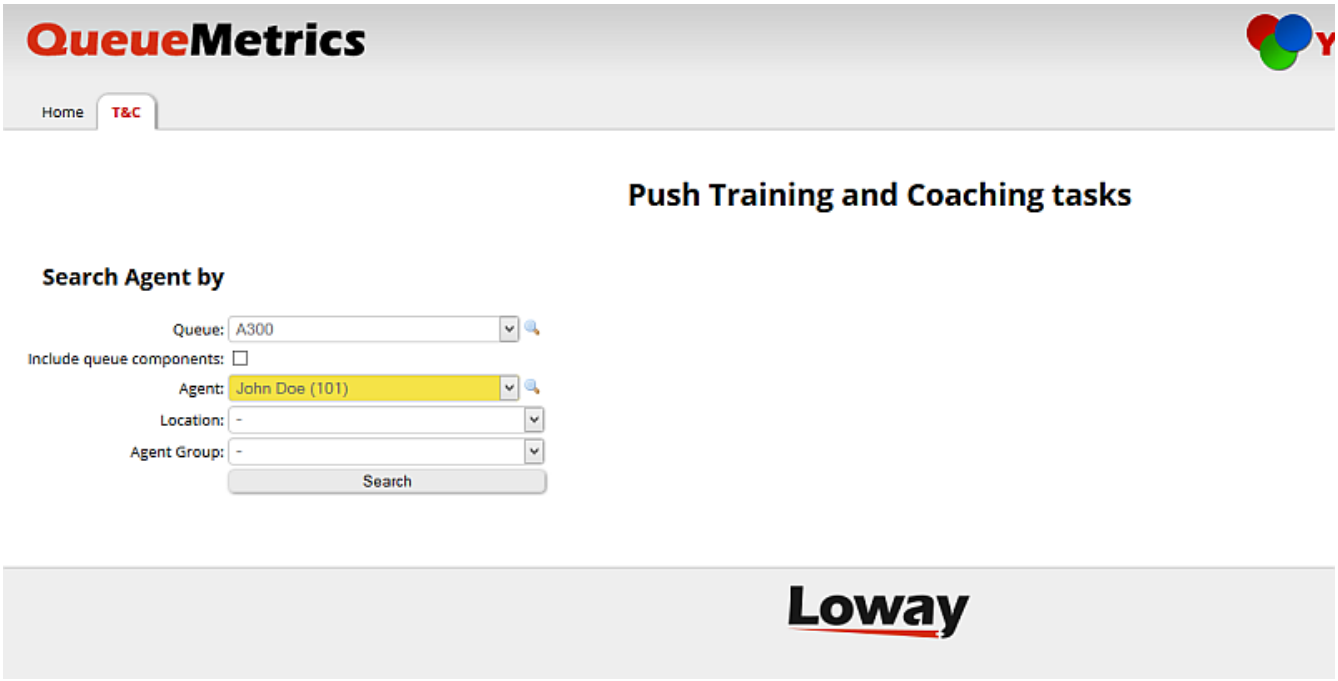
All values are computed accorded to the current item weights (only for scoreable items), in case you use weighted items.

The data can be exported to Excel, CSV and XML formats.

On the top right of the report is a box containing the list of levels with related checkboxes. Values shown in the result table are coloured following the checkboxes status. This is useful to hilite questions where average values fall in a specific set of levels. The status of checkboxes are persisted through sessions.

The Training And Coaching Page

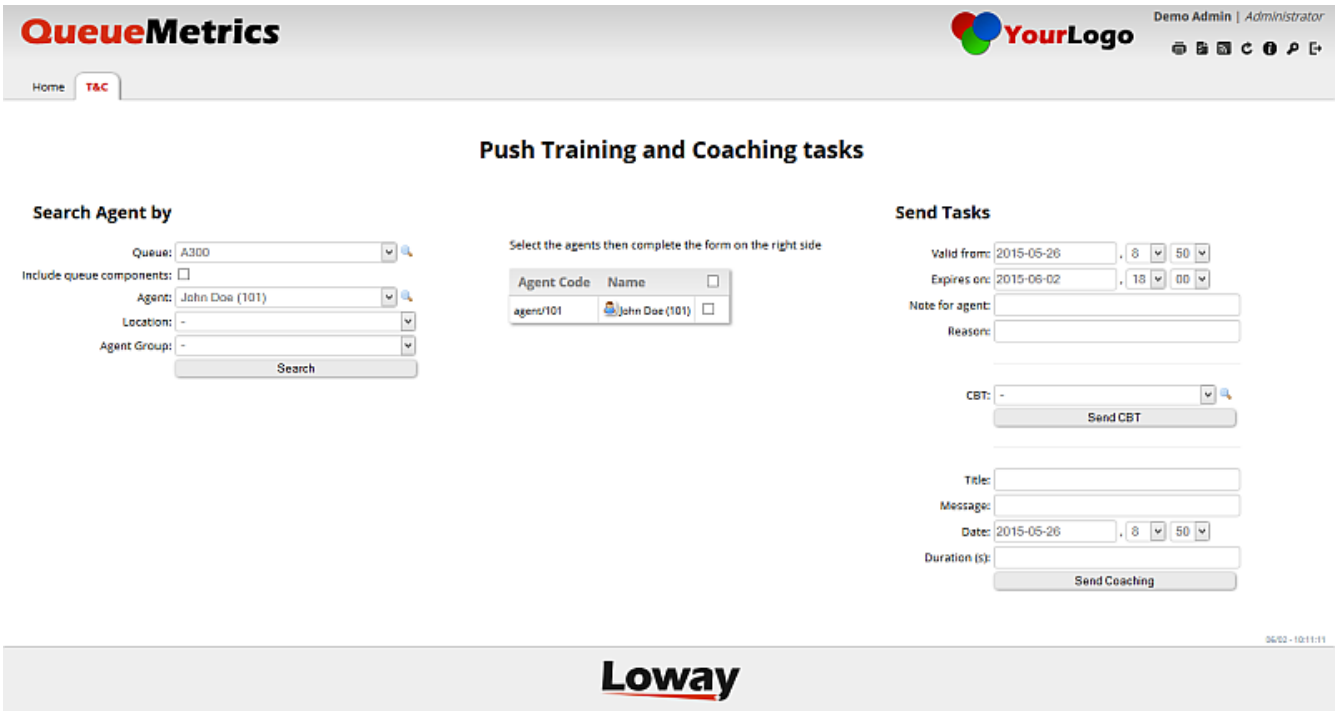
Users holding the security key TASK_PUSH_TC will be able to access to the Training and Coaching Page. This page is the main access point for sending CBTs and Meeting (also named Coaching) tasks to an agent or a set of agents. It's based on a three steps wizard where the list of users are defined in the first two steps and the proper CBT or Meeting tasks are sent in the third step. Agents selection starts by several finding criterias, as depicted below:



- The queue dropdown
- If the finder should take care about queues components (this is useful when a composed queue is selected in the previous dropdown)

- An agent dropdown
- A location dropdown
- An agent group dropdown

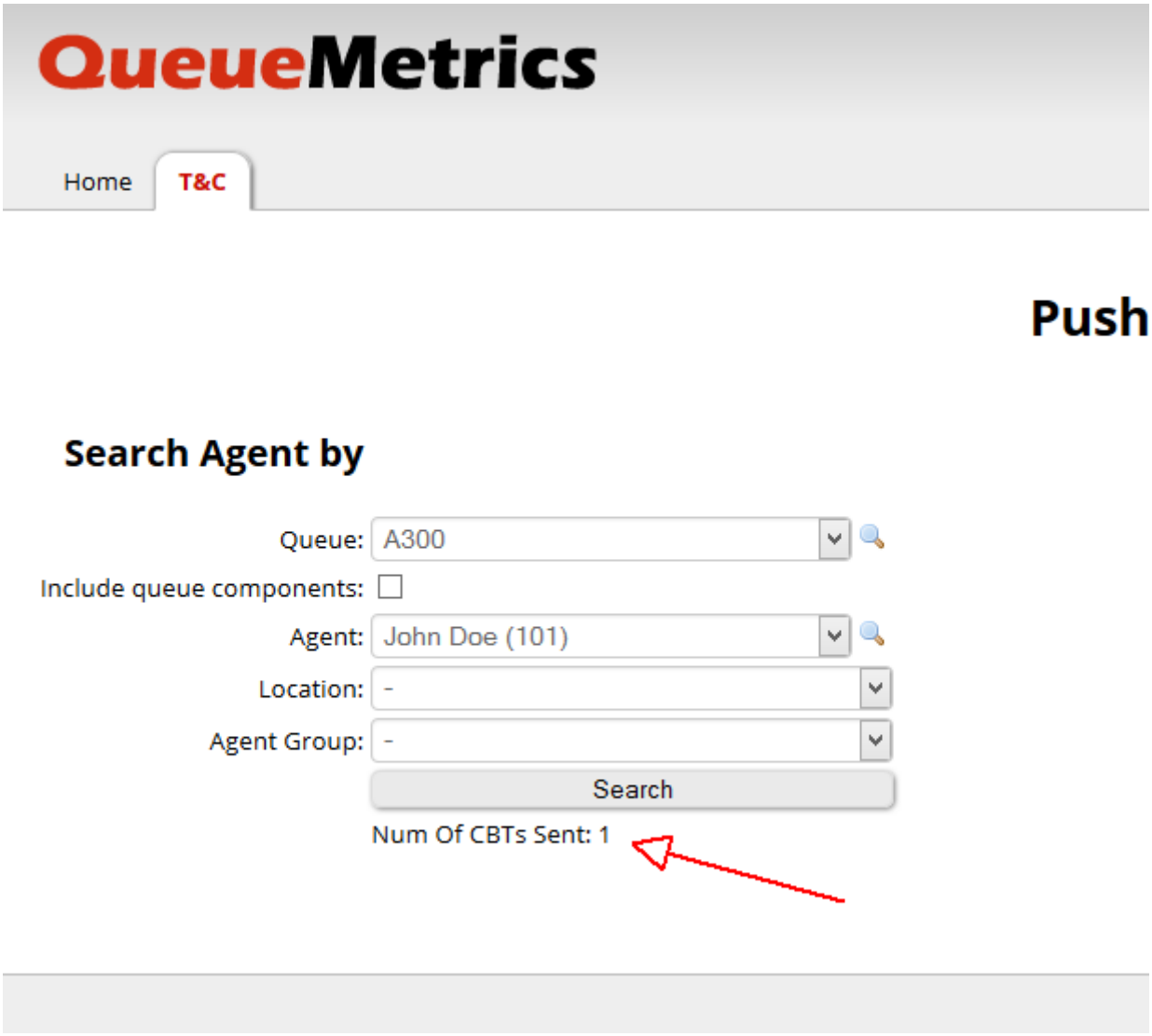
Each agents matching the criterias are shown as soon as the user presses the "Search" button, as reported in the picture below.



Looking at the list of agents found in the center of the page, a single agent can be selected by clicking on their related checkbox whether all agents in the list can be selected by clicking on the checkbox placed on the header of the list.

As soon as agents are selected, the user can send a CBT or a Coaching task by properly filling the fields related to time and date validity and, optionally, a note that will be shown to the agents and a reason (hidden to the agents). Specific fields for CBTs and for Coaching tasks can be found on the most right-bottom side of the page. Tasks can be sent by pressing "Send CBT" or "Send Coaching" buttons.

At the end of process, QueueMetrics shows the number of CBT/Coaching tasks sent as reported by the following picture.


QueueMetrics
Home
T&C
Push
Search Agent by
Queue:
A300
Include queue components:
Agent:
John Doe (101)
Location:
-
Agent Group:
-
Search
Num Of CBTs Sent: 1

Advanced tracking of agent and grader performance

Advanced tracking of agent performance lets you pinpoint those agents whose behavior can and should be improved. In order to do this, a **score** is created out of multiple items related to:

- their way each call is handled (e.g call duration)
- their call performance (e.g. number of sales)
- QA scores that were given to their calls

As the scoring is in itself quite complex and made up of multiple factors, scoring is based on a **rule set** that represent a business-specific set of targets that should be met. For each rule, you have two possible levels of non-compliance, that is a yellow and a red threshold. Each threshold can, in turn, have a peculiar score associated.



For example, you could say that the expected call duration is 100 seconds; calls that are between 100 and 150 seconds are "yellow" and worth 1 review point, calls that are over 150 seconds are "red" and worth 4 review points. The higher your review score, the more prominently the agent will be displayed.

When applying a rule set to a set of calls, you get a **score** expressed in **review points** for each agent selected that represents the sum of all anomalies as detected by the chosen ruleset.

The system then displays the agents involved in reverse score order, prompting the grader to investigate further by accessing the set of calls and the set of QA records and the relevant audio recordings.

The result of this activity is:

- In-depth knowledge of agent performance
- **Agent life cycle management**: the grader can move agents between agent groups, so that you can manage a process where an agent belongs to multiple skill groups during their lifetime
- Continuous improvement of agent performance through **agent tasks**, e.g. coaching sessions, or completing Computer Based Training to improve the agent's skills.

For example, an agent could start her life as member of the group *New Hires*. When reviewed after a while, she could be moved to *New Hires Probation* when she is found lacking in some subject. After a while she could be checked again and moved back to *New Hires*.

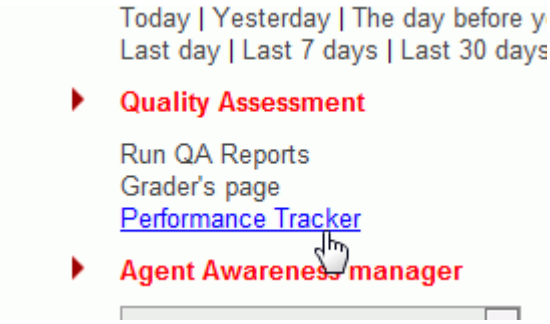
As collateral features, the system also offers facilities to:

- Create rule sets based on the average properties of a set of calls. This makes it easy to have reference points that can then be manually edited.
- Track the lifecycle of agents. This is done by tracking the different agents groups each agent has been a member of and the time period they have been there.

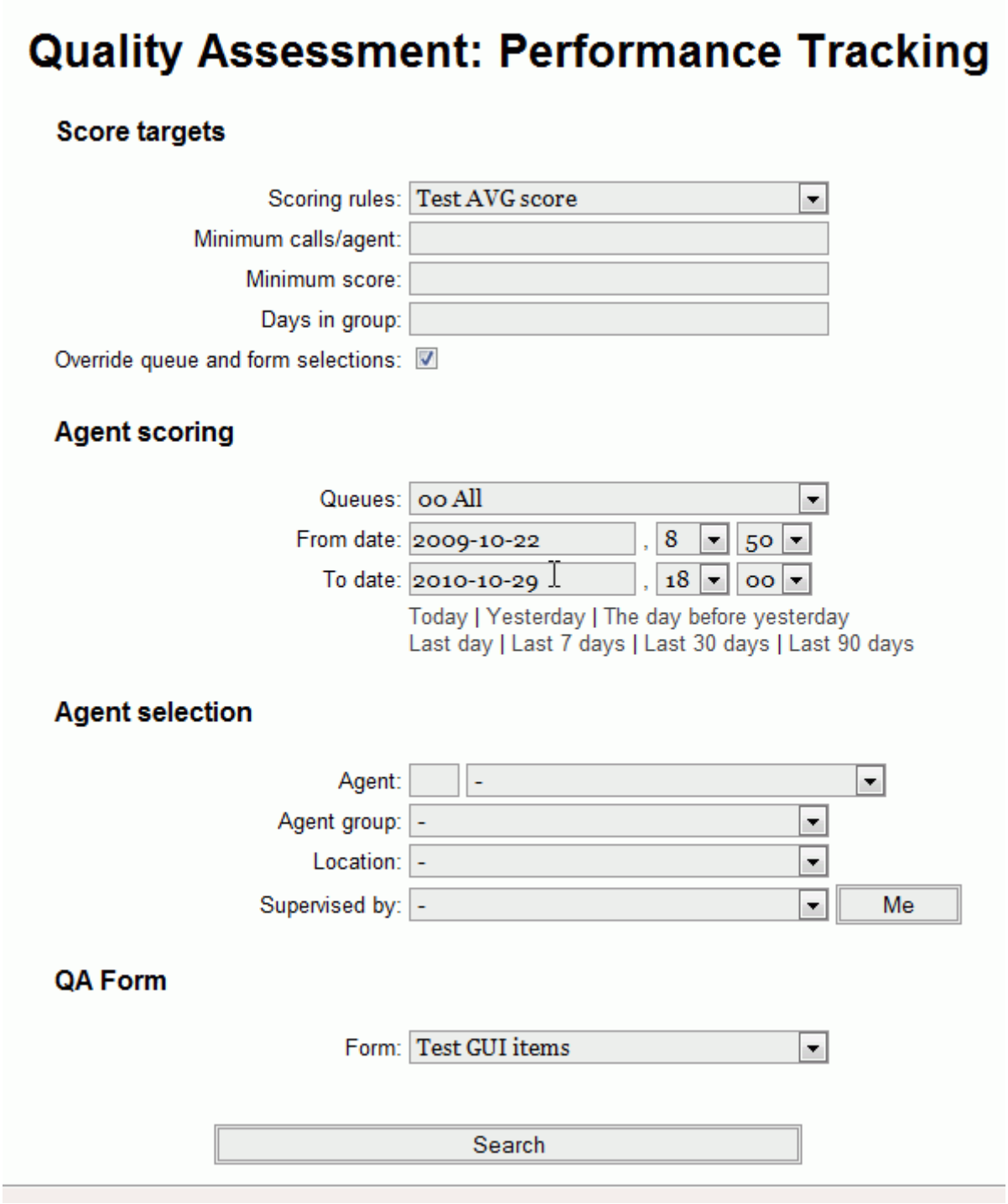
Just like for agents, there is also the problem of comparing graders to each other, in order to have a "fair" view of what is going on and to make sure that grading happens under the company's guidelines and not each grader's own preferences. **Grader calibration reports** fulfill this purpose by comparing graders to each other.

Tracking agent performance

For users holding the key "QA_PERF_TRACK" a new link appears in the QueueMetrics home page, as shown below:



When clicking on it, you are lead to the main search page:



This page lets the grader search for a set of agents to be reviewed. This requires setting three search dimensions:

- A queue (or set of queues) and a time period
- A way to search for a set of agents (a specific one, or a group, or a location, or all agents that have the same supervisor).
- A QA form to be graded
- A **rule set** that applies to the above search and defines scoring. You should define your own before you start this activity (see Defining agent performance rules).

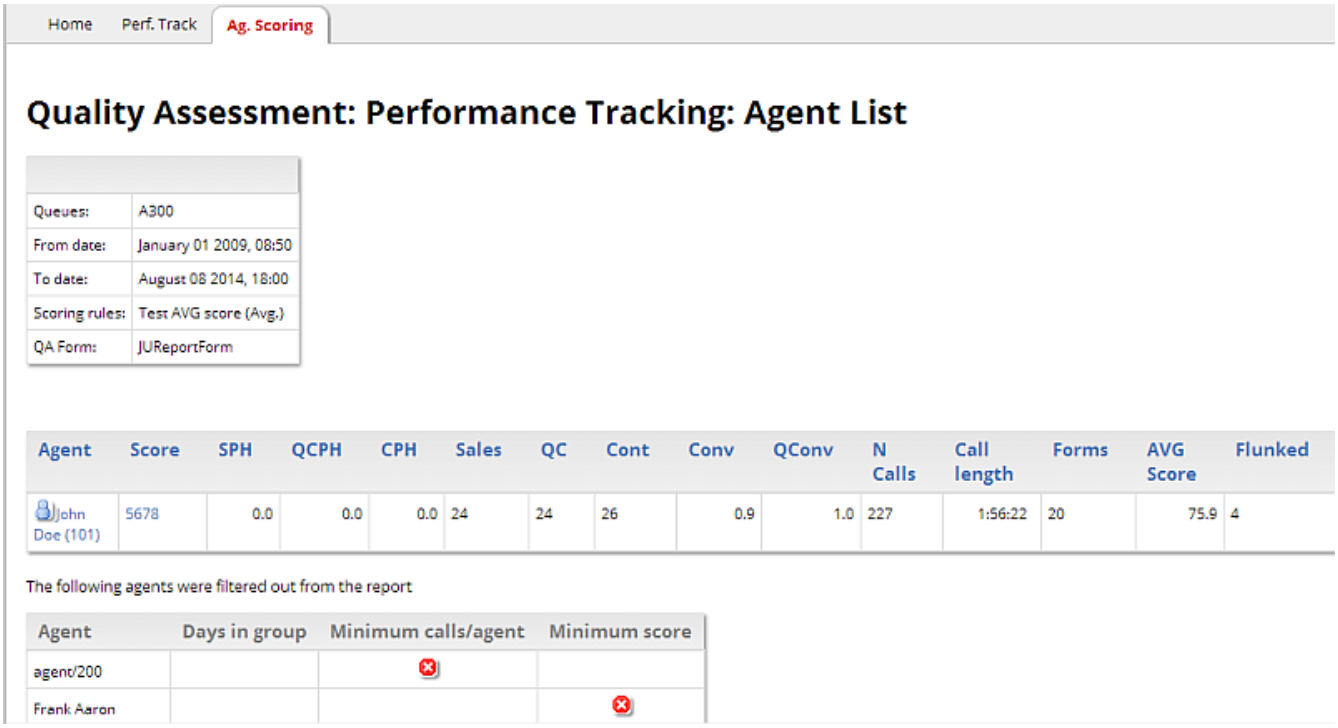
The scoring rule is usually associated to a particular queue and form but the user can override this selection by checking the option "Override queue and form selections" and by specifying other

parameters that affect the calculation, like:

- the minimum expected score,
- the minimum number of calls that should be analyzed,
- the minimum days the agent had to be in the group at the run period specified for the analysis.

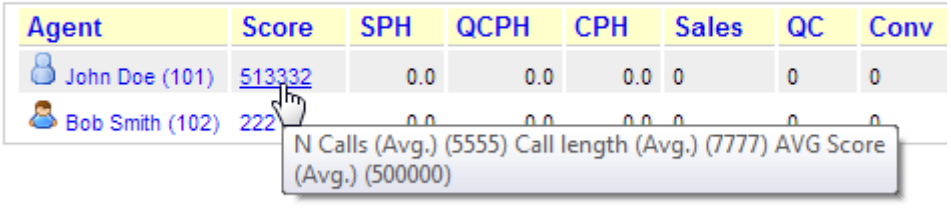
These minimums are to avoid considering agents that are undersampled (e.g. if an agent has been scored only once, we can expect this score to be less meaningful compared to an agent whose score is based on 10 elements).

The button "Search" starts the calculation process and a new page will be displayed:



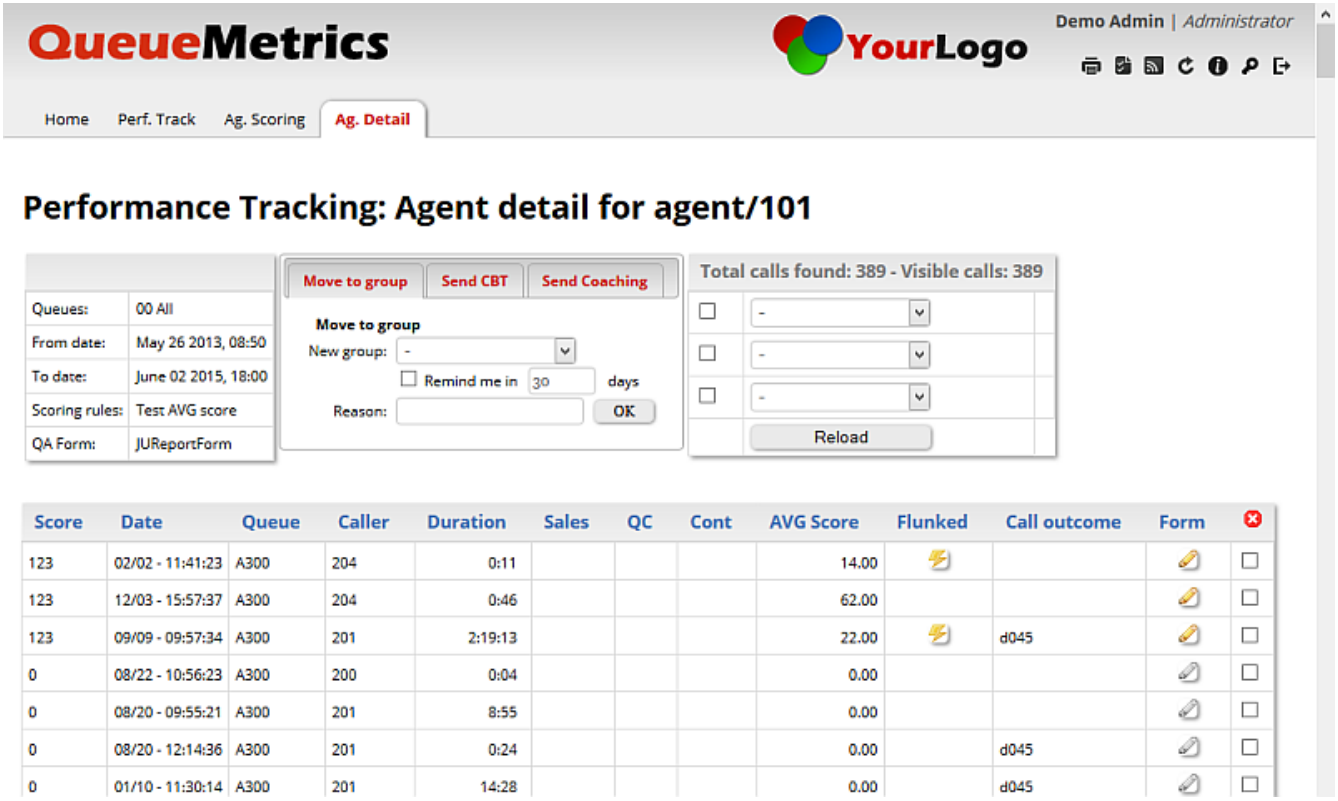
The items shown here are **averages** on all the calls that were found in the current set. The selected score rule is used to compute the overall Score value, and agents are shown sorted by their score in descending order.

Flying with the mouse over the score value a detail rules contributing to the overall score are shown, as reported on the picture below:

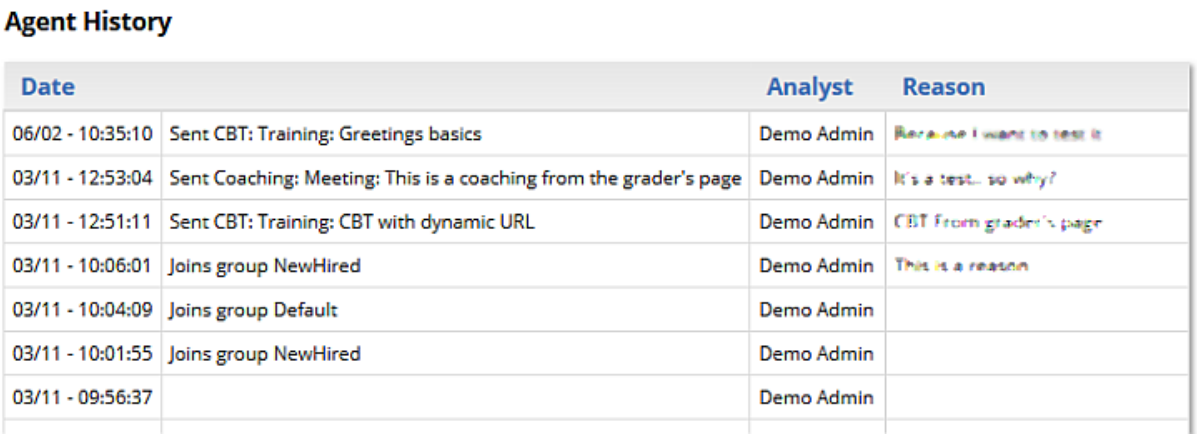


At the bottom of the main table result, a second table shows the agents (if any) that were not included in the report and the threshold that was not met.

Clicking on the agent name or the associated score value the user gives access to details. They are reported in a different page, like the one shown below.



The details page is split in two parts. The top part reports the score details for each call the agent answered. The bottom one shows the detailed history associated to that particular agent.



Each line in the top table reports the score calculated by the rule selected in the search page **without being averaged**, and other relevant information for each call.

An icon representing a pencil is shown if the call has a QA form associated with it; by clicking it, the associated QA form will be shown in a separate pop-up dialog. Users allowed to grade calls could find a set of grayed out pencil icons to be used to score new calls from this page. After scoring each call they should press "Refresh" to have the page updated.

A special icon is assigned to *Flunked* calls. A call is defined as *Flunked* if the related QA Form has been graded but reached an average value below the Issue level.

Dynamic drill down

To make the grader's life easier in case the set of calls to be analyzed is large, it is possible to use a set of dynamic criteria to reduce the data that is currently displayed.

The table on the top-right lets you add up to three drill-down rules being active at the same time. Rules can be quickly disabled by clicking on the "Turn off" checkbox on the left of each rule.



The title of the section shows the total number of calls found for the agent and the number that is actually displayed. The ones that are not are the ones that are filtered out. In order to see them all again, just clear or turn off all the filters.

The following rules are defined:

- Call length
- A form's average score
- The performance tracker score
- The call outcome
- Whether the call is a Flunked one, or a Sale, or a Qualified Contact, or a Contact

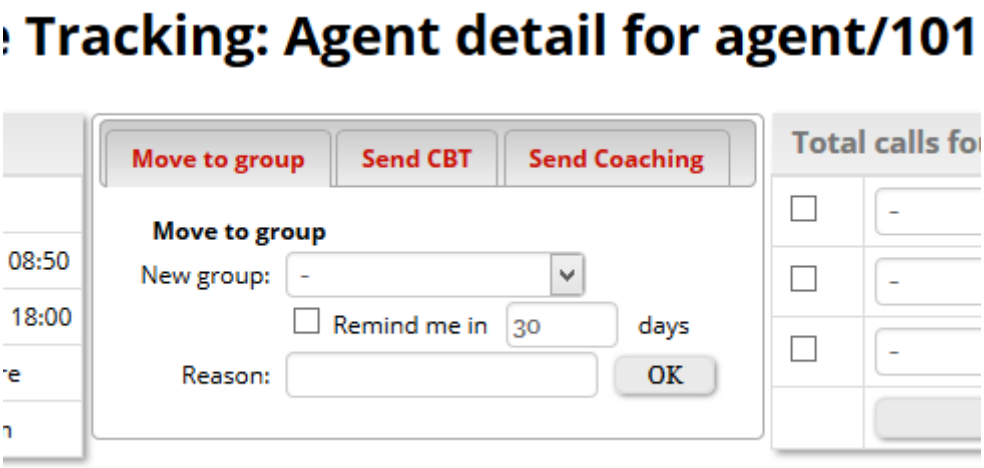
Special notes about rules:

- If you enter multiple rules at once, they are all active
- Rules expecting a time duration (e.g. call length) will accept input as H:MM:SS (e.g. 0:23) or an integer number of seconds.
- Rules working on a float value have the float value converted to the nearest integer and then the rule is applied.

It is always possible to sort data in the call list table by clicking on the title.

Taking remedial actions

By the top of the page, the grader can take remedial actions using the form displayed below.



Move the agent to a different group

In order to move the agent to a different group, the grader has to select the new group through the dropdown; he can specify a reason in the lower text box then press the OK button on the right side of the dropdown group.

If the user checks the "Remind me" checkbox before pressing the OK button, QueueMetrics will send a reminder task to the grader himself that will be displayed after a specified number of days. (This can be used as a reminder and is optional).

A new row with the operation details will be inserted in the agent's history table after completion.

Send a CBT to the selected agent

In order to send a CBT (Computer Based Training) to the specific agent the report is referring to, the grader should select an already known CBT from the dropdown list or manually specify a URL. An optional note (visible to the recipient agent) and an optional reason (hidden to the agent) can be added. Grader can optionally link a specific call with the CBT to be sent, so, the agent will be able to refer to that particular call in the task details. To link a specific call to the CBT, the grader should click on the checkbox he can find each row in the call list. Unlinking the call is done by clicking on the red icon in the list header.

A new *Teaching* task will be sent to the agent with the title and the inserted URL.

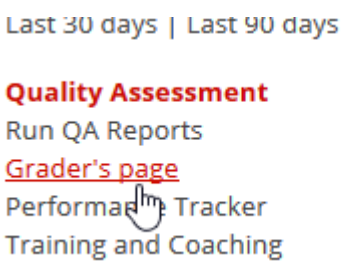
A new row with the operation details will be placed in the history table after completion.

Send a Coaching (Meeting) task to the selected agent

A grader may want to send a Coaching/Meeting task to the specific agent the report is referring to. This can be done through the *Send Coaching* tab. A Coaching task is defined by a *Title*, a *Message* and a task date. The grader can optionally specify a coaching duration (in seconds), a note (visible to the agent) and a reason text (hidden to the agent). Pressing OK generates a new coaching task with the information provided. The task will be placed in the history table after completion.

The Grader's Page - Finding calls to be graded

This is a separate page accessed by users holding the QA_GRADER security key.



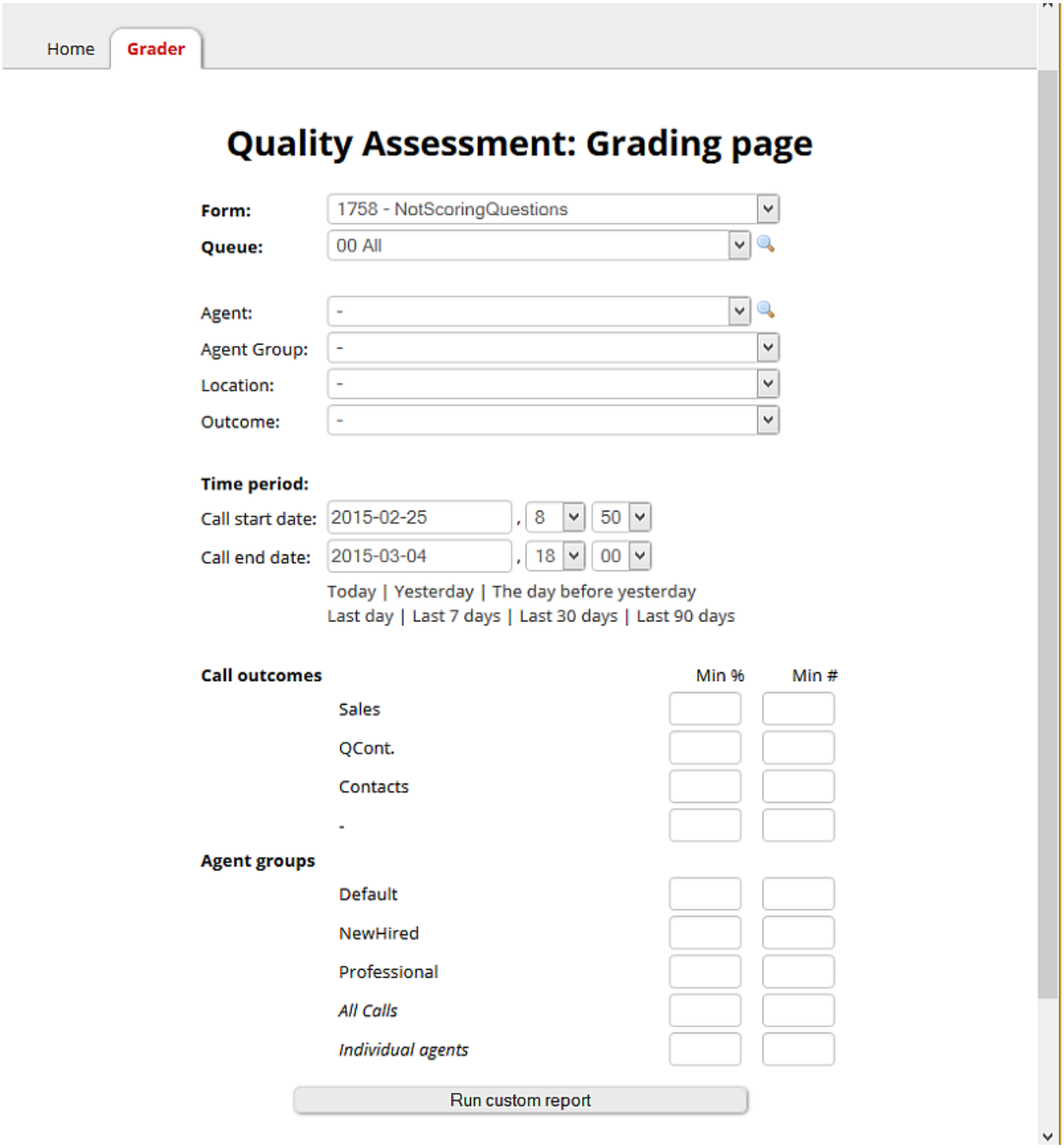
QA graders page is to provide graders with a tool that will help them in finding which calls are to be graded. Calls can be searched by specifying several targets: per client, per queue, per agent and per agent group. Each target can be expressed as an absolute minimum, a percentage or both.

Default targets are expressed:

- Per queue or set of queues
- Per location
- Per individual agents (that is, the minimum required for each agent)
- Per call outcome (see Configuring call outcomes)
- Per agent or agent groups (see Configuring agent groups)

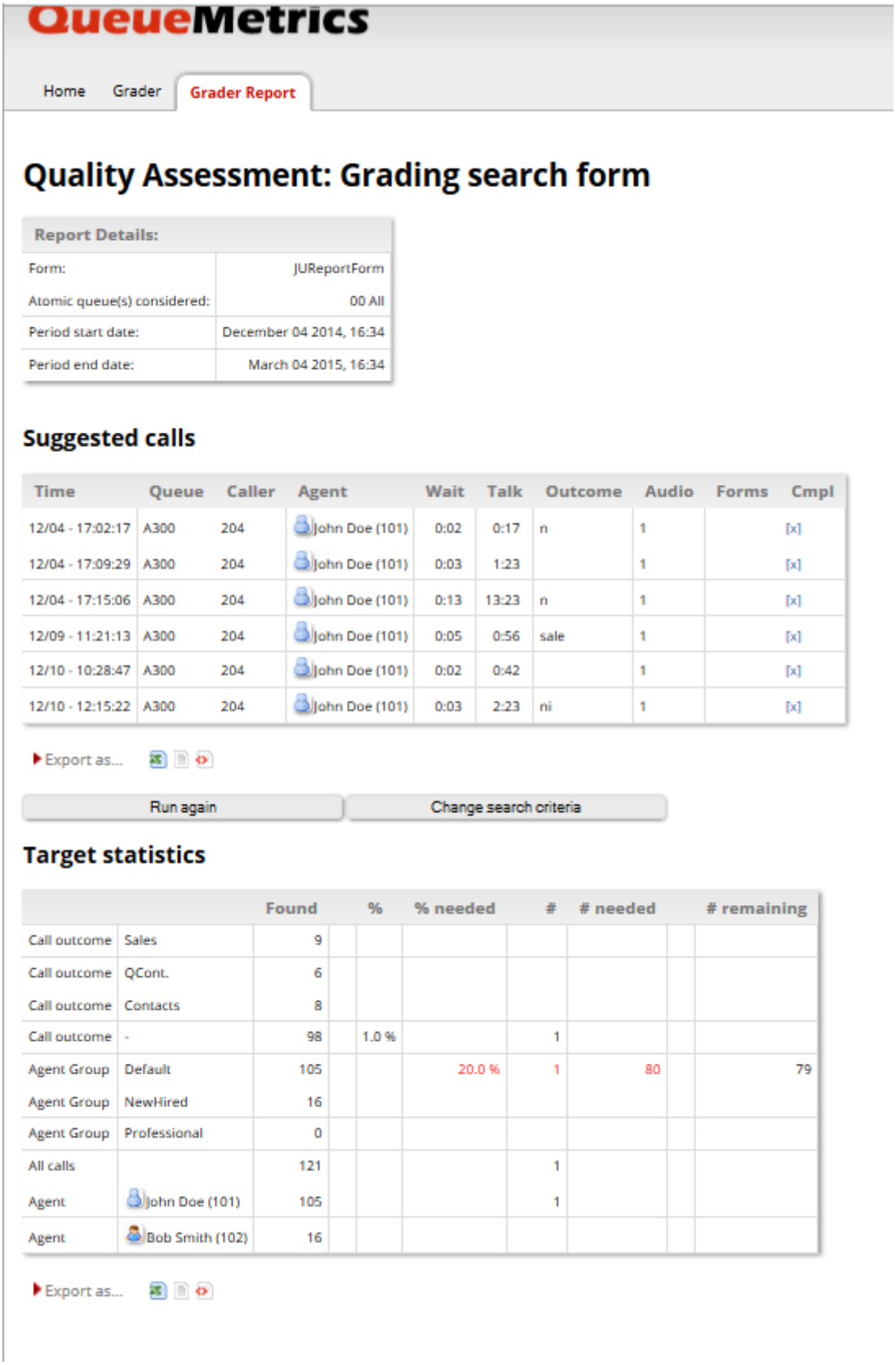
Targets can be disabled as well, by leaving them empty.

All grading activity is done per form, that is, related to one specific form.



The QA personnel is supposed to input the right targets given the goal at hand (eg. daily or weekly, per client or per queue, etc).

When they do, QueueMetrics will run a report that displays the absolute numbers involved and the number of targets that are currently being hit. At the top of the page, QueueMetrics will then display a random selection of calls to be graded, weighted on the number of targets each call hits (i.e. the more targets a call hits, the more likely it is to appear in the list).



The grader may do grading on some or all of the suggested calls, or may decide to do differently. When done, he clicks on "Run again"; all the form is computed again, targets are recomputed and so

is the set of possible calls.



The Grader's page also allows automatic grading of transcribed calls through an AI model - see Automating QA.

Selecting which calls to grade

Each call included in the search criteria per time period or queue is first listed. The set of distinct agents involved in the queue is obtained.

Calls are then filtered off if they match the following rules:

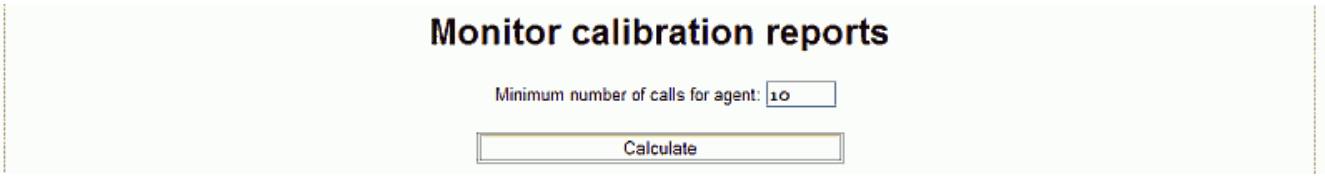
- They have already been graded for this form
- They have no audio recording
- They will help no target

For each of the remaining calls, a score is computed by assigning one point to each of the criteria that are currently unmet absolute number and percentage are treated as distinct criteria). Calls are then weighted based on the square of their score (that is: if a call will help 10 targets, its relative score will be 100, while if it helps in 5 targets, the relative score will be 25), and extracted randomly, so that it's more likely for QueueMetrics to draw calls that match multiple criteria, but even a call matching one single criteria may appear. As the computations involved are quite heavy (mostly when deciding if a call has associated audio and QA scoring, as this might be repeated tens of thousands of times) each grader will have a small set to use before reloading.

Grader calibration reports

This is a separate, one-page report that is only accessed by supervisors earmarked by the key "QA_CALREP" only (in addition to "QA_REPORT").

To access the page, you go to *Quality Assessment → Run QA Reports* and fill-in the form by the bottom of the page :



You will also use the form by the top of the page you usually use for QA reports.

On the input page you select:

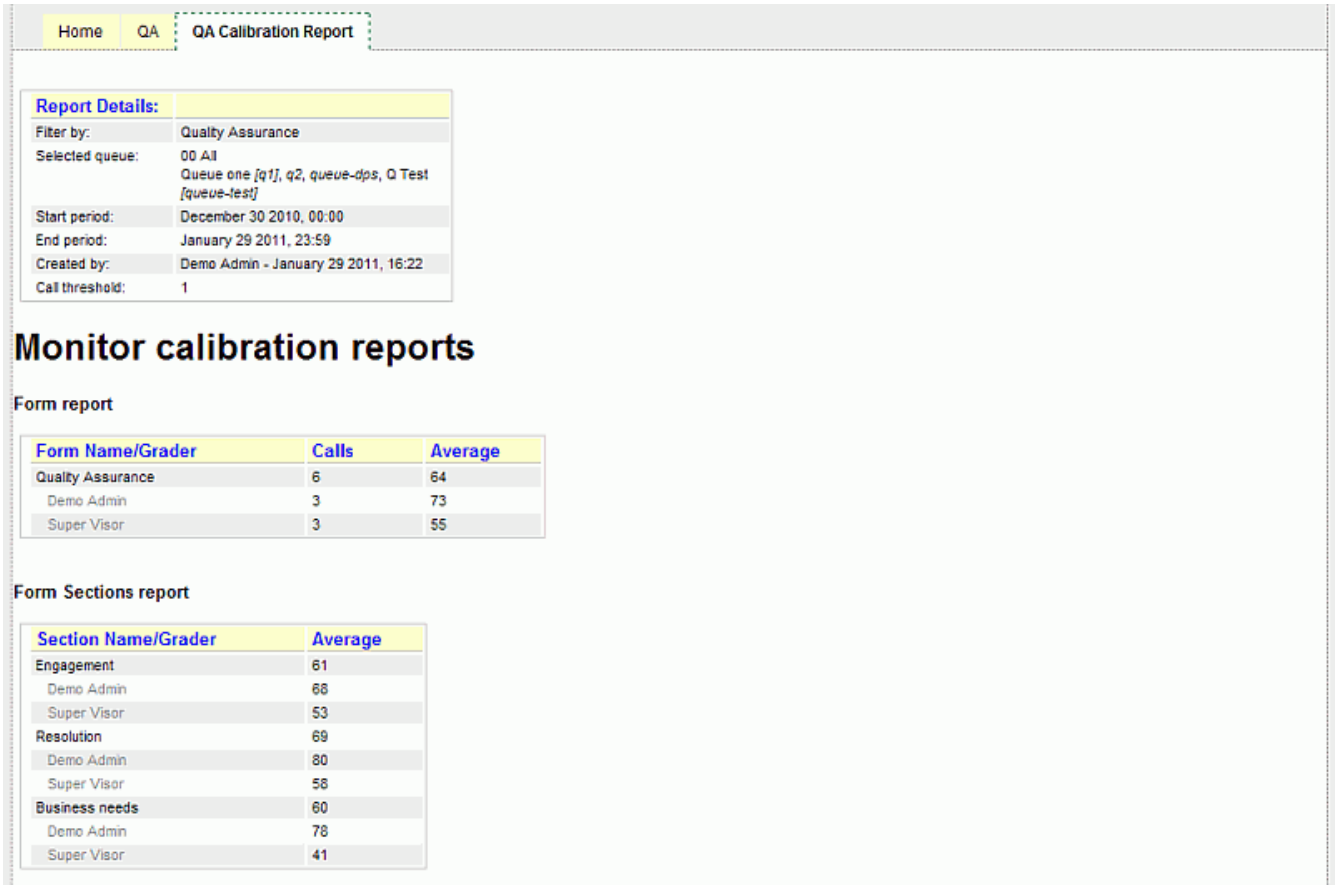
- A date range (dual selector plus predefined periods)
- One Form (if we had an option for All, we'd get way too many)
- A queue or composite queue
- A call outcome, or none for all calls
- It is possible to select an agent group as an additional filter.

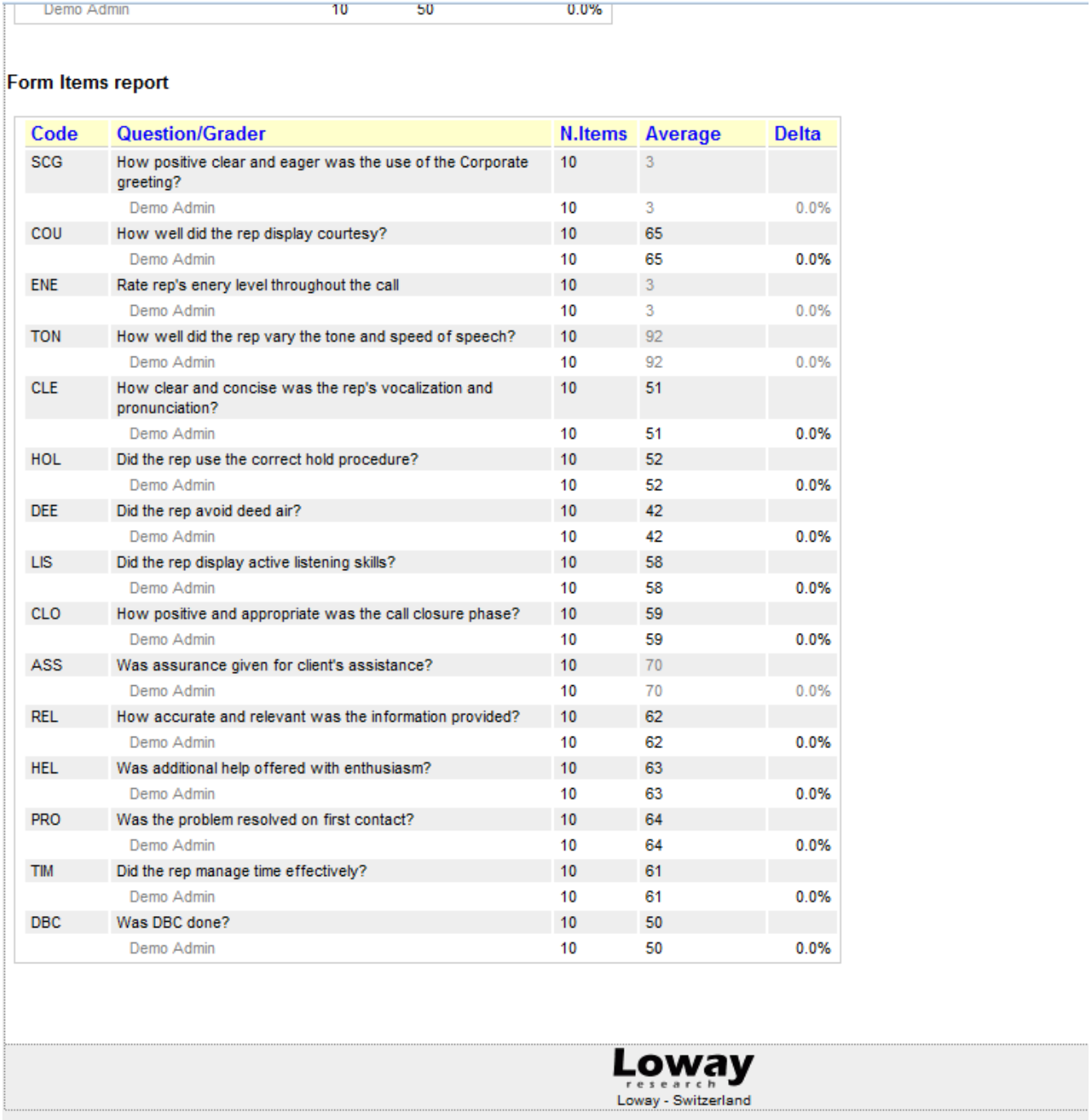
- It is possible to add a minimum threshold of graded calls per grader to be included

The analysis happens at three levels:

- The whole form
- The section level
- The question level

For each form/section/question, a table is computed for the general and for each agent that has graded at least X items:





Demo Admin 10 50 0.0%

Form Items report

Code	Question/Grader	N.Items	Average	Delta
SCG	How positive clear and eager was the use of the Corporate greeting?	10	3	
	Demo Admin	10	3	0.0%
COU	How well did the rep display courtesy?	10	65	
	Demo Admin	10	65	0.0%
ENE	Rate rep's enery level throughout the call	10	3	
	Demo Admin	10	3	0.0%
TON	How well did the rep vary the tone and speed of speech?	10	92	
	Demo Admin	10	92	0.0%
CLE	How clear and concise was the rep's vocalization and pronunciation?	10	51	
	Demo Admin	10	51	0.0%
HOL	Did the rep use the correct hold procedure?	10	52	
	Demo Admin	10	52	0.0%
DEE	Did the rep avoid deed air?	10	42	
	Demo Admin	10	42	0.0%
LIS	Did the rep display active listening skills?	10	58	
	Demo Admin	10	58	0.0%
CLO	How positive and appropriate was the call closure phase?	10	59	
	Demo Admin	10	59	0.0%
ASS	Was assurance given for client's assistance?	10	70	
	Demo Admin	10	70	0.0%
REL	How accurate and relevant was the information provided?	10	62	
	Demo Admin	10	62	0.0%
HEL	Was additional help offered with enthusiasm?	10	63	
	Demo Admin	10	63	0.0%
PRO	Was the problem resolved on first contact?	10	64	
	Demo Admin	10	64	0.0%
TIM	Did the rep manage time effectively?	10	61	
	Demo Admin	10	61	0.0%
DBC	Was DBC done?	10	50	
	Demo Admin	10	50	0.0%

Loway research
Loway - Switzerland

For each form/section/question, an average is computed and compared to the one of all graders who graded at least X calls in the specific area. This way it is easy to spot trends and anomalies on grading behavior. Values shown in gray refers to non scorable questions. Average for this type of questions is counted only for reference purposes.

Call transcriptions and Automated QA

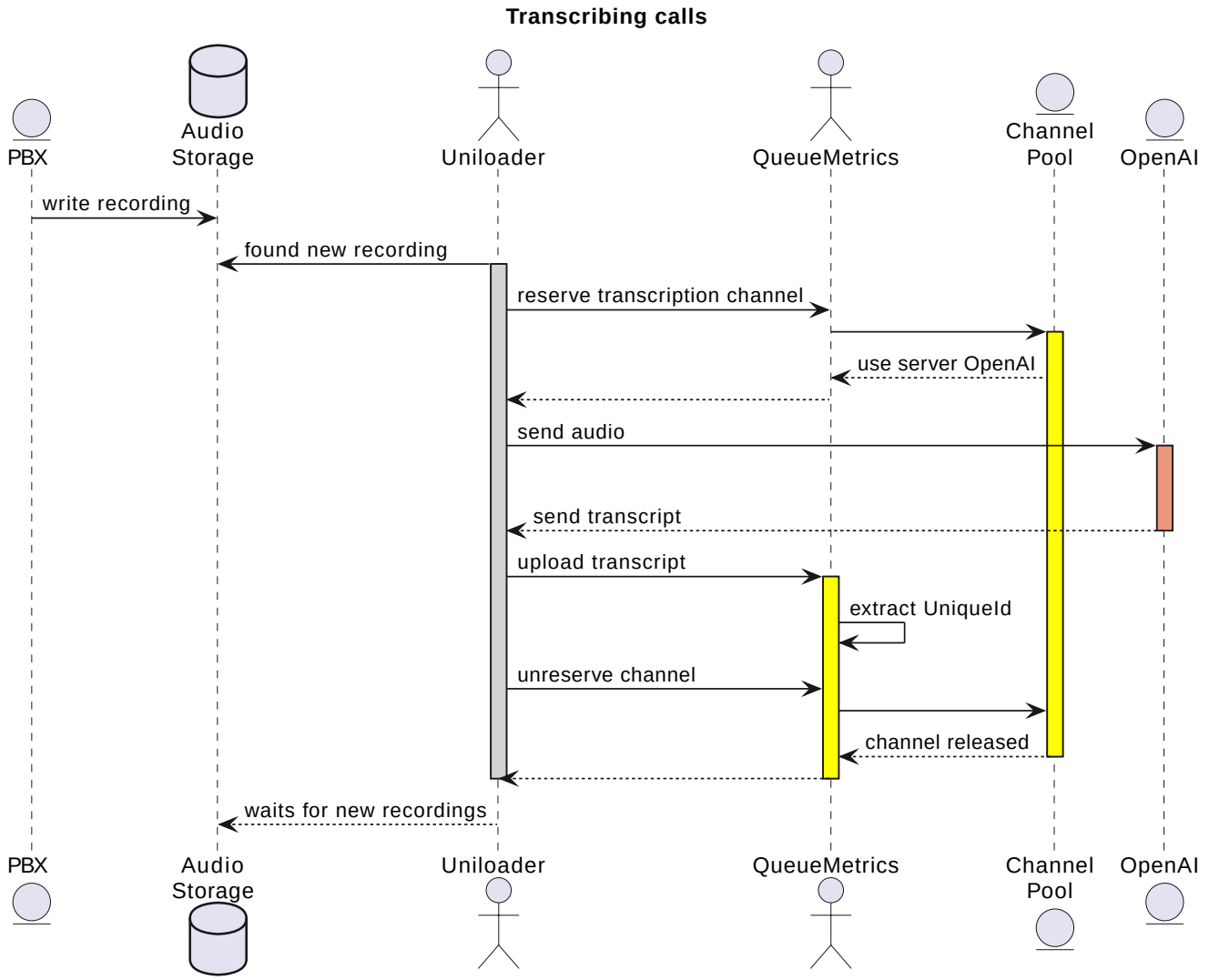
Concepts

QueueMetrics can automatically transcribe the phone calls handled by your Contact Center and, optionally, automatically evaluate them for quality. When an interaction is already text-based to begin with — for example a chat or a WhatsApp conversation — there is nothing to transcribe: the conversation itself can be shown in the interaction detail and automatically evaluated in the same way.

To transcribe calls, QueueMetrics relies on transcription servers, which can be either remote (a third-party software-as-a-service) or local to your own infrastructure. This lets you handle cases where sending call data outside of your organization is not acceptable — for example for banks or healthcare providers — by using a locally-installed transcription server instead of a remote service.

The life-cycle of a call

A call starts its life as a recording on disk. Uniloader constantly monitors the recording files that appear on disk and, as soon as the PBX stops writing to one of them, queues it for transcription.



To have a call transcribed, Uniloader asks the QueueMetrics server whether a transcription server is currently available. QueueMetrics keeps track of how many transcriptions are running in parallel on each configured server, and looks for one that still has a free channel. As soon as one is found, Uniloader sends the audio file to that server, downloads the resulting transcription, converts it to the format QueueMetrics expects and uploads it.

Setting up the server pool

Both third-party transcription services and, even more so, locally-installed transcription servers share the same limitation: they can usually only process a limited number of calls at the same time. This is why QueueMetrics manages transcription capacity centrally, instead of leaving every single Uniloader instance to guess which server is actually available. The QueueMetrics server manages a **pool** of transcription servers, each with its own capacity, that can be used according to different rules. Servers can be added to or removed from the pool at any time, without interrupting the transcription process.

This means QueueMetrics can work not only with the best-known transcription services (which are convenient, as you only need to buy some credits and enter an access token) but also with locally-installed servers, for customers who cannot or do not want to send the content of their calls outside of their own infrastructure.

Each server, whether a SaaS contract or a local installation, only offers a limited number of **channels**: the maximum number of calls it can process in parallel. Whenever QueueMetrics needs to perform a transcription, or an AI-assisted quality evaluation (see Automating Quality Assessment), it borrows a free channel from the pool and returns it once the operation is complete. The same pool of servers and channels is shared by both transcription and AI grading. If no channel is free at that moment, the request simply waits until one becomes available.

For each transcription server in the pool, you can configure:

- **Name**: a name to identify the server
- **Driver**: which kind of transcription engine/service this server is
- **URL**: the address QueueMetrics should contact
- **Token**: the access credential for that service, if required
- **Priority**: used to decide which server to prefer when more than one has a free channel
- **Main Tag**: which kind of work this server is willing to accept
- **Extra Tags**: Any additional tags that Uniloader uses to tag this server, comma-separated.
- **Max Channels**: how many operations this server can run in parallel
- **Description**: a free-text description
- **Parameters**: further, driver-specific configuration

Once saved, a server also shows its current **Status**, whether it is currently seen as **Healthy**, and the time of its **Last Health Check** — all reported automatically by QueueMetrics and not editable.

Tags decide which kind of work can be routed to a server. The tags currently known to

QueueMetrics are:

- **TD**: transcription with diarization, i.e. also recognizing who is speaking
- **TR**: plain transcription of a recording. If a diarization model is not available (e.g. for some local models) or is too expensive to run, you can get the same result cheaply by using a stereo recording of the call. In this case, Uniloader will first separate the two channels, transcribe each separately, and then reconstruct the call flow based on timings.
- **QA**: AI-assisted QA evaluation of a call, using an LLM

All of a server's settings, including its status, priority and tags, can be changed at any time while the server is in use, without interrupting the transcriptions or evaluations already running on it - the new settings simply apply to the next operation that needs a server. This also means you can have several servers configured with different priorities for the same kind of work. For example, you could prefer your local server for plain transcriptions and only fall back to a SaaS service when there's a problem or too many calls to handle locally: to do so, you would pause the local server and resume the SaaS one, or the other way around, as needed.

When QueueMetrics looks for a server to handle a given operation, it considers every active, healthy server that carries the required tag and still has a free channel, and picks the one with the lowest priority number first; if there is more than one server at that same priority, it prefers the one that currently has the most free channels.

Monitoring

From the server pool configuration page, "Go to live view" opens a live status page listing every configured transcription server, showing its name, driver, current status (active or paused) and how many of its channels are currently in use out of the maximum. From here, a server can be paused or resumed - for example to take it offline for maintenance - and its recent activity can be reviewed.

For each server, the list of transcription channels currently in use shows the agent whose call is being transcribed, the media file involved, the channel's state, the number of tokens consumed so far, and when the channel was reserved and last seen active.

Transcribing calls, in practice

Once a call has been transcribed, the conversation is immediately visible in the call detail, right below the audio recording. Clicking on a transcribed sentence jumps the audio player to that point of the call, so you can listen to it. If a transcription is not perfect, it can be edited.

Extracting a call's Unique-Id from the recording name

When a transcription is uploaded, QueueMetrics needs to know which call it belongs to. To do this, it derives the call's Unique-Id from the name of the media file being transcribed; this unique-id is what is later used to match the transcription back to its call, which in turn is what makes it possible to run automated QA evaluation on it (see Automating Quality Assessment) - without it, a transcription cannot be linked to a call for automated grading purposes.

This is controlled by the system property `llm.captureUniqueId`, which holds a regular expression applied to the media file name: whatever text matches the (first populated) capture group becomes the call's unique-id. The regular expression can contain more than one capture group - for example to match several different file naming conventions with a single pattern - whichever group actually matches for a given file is the one used. If this property is left empty (the default), no unique-id is captured from the file name.

The exact filename format depends on which PBX generated the recording, so the regular expression needs to be adapted accordingly. A few examples:

PBX	Sample filename	Regexp	Notes
Freepbx	`q-1000-304-20260807-110757-1786093677.0.wav`	`++^.*-(.+)\.[^.]+$++`	Captures everything from the last dash to the last dot (`1786093677.0`), regardless of the file extension.
FusionPBX	`cd4ca96e-1ab0-4ac0-b4f6-50e1c4650b62.wav`	`++^(.+)\.[^.]+$++`	The whole file name is the unique-id (a UUID); captures everything up to the last dot, regardless of the file extension.



If the regexp contains multiple capture groups, the first that matches will contain the Unique-Id.

I forgot to capture the Unique-Id. What now?

If you forgot to capture the Unique-Id for a set of transcribed calls, or the regular expression you used to capture it was incorrect, those calls will not be available for automated QA analysis.

If you do not use automated QA, there is nothing else you need to do.

If you want those calls to be available for automated QA, as described in Automating Quality Assessment, you need to transcribe them again. Once they are re-transcribed, the calls are uploaded to QueueMetrics again and their Unique-Id is captured correctly. To do so:

- Correct the `llm.captureUniqueId` property, if needed, so that it captures the Unique-Id properly.
- Delete the existing transcription of each affected call, by visualizing it and selecting **Delete Chat** from the chat settings (gear icon), as described in Deleting a Chat.
- Restart the transcription service with `service uniloader-transcribe restart`. The check for missing transcriptions is performed only at startup, so the calls will not be queued for transcription until the service is restarted.



If this happens often, we suggest scheduling a daily restart of the service, so that any calls whose transcriptions were deleted during the previous day are picked up and transcribed again.

Setting up QueueMetrics for transcriptions

First, make sure that you have the latest Uniloader installed, and that the `uniloader-transcriber` service is running on the PBX server (or wherever the audio recordings are).

Log in to QueueMetrics, go to Users, and find the user called "transcriber"; enable it and set a password. This account will be used to send transcriptions.



If you are starting from an older QueueMetrics install, just create a user called "transcriber", in class `ROBOTS`, with the custom key `CHAT_WRITE`.

Edit the transcriber's configuration file at `/etc/defaults/uniloader-transcribe`, and set:

```
# Base URL of the QueueMetrics instance
QM_URL=http://127.0.0.1:8080/queuemetrics/

# QueueMetrics login
QM_LOGIN=transcriber

# QueueMetrics password
QM_PASSWORD=.....

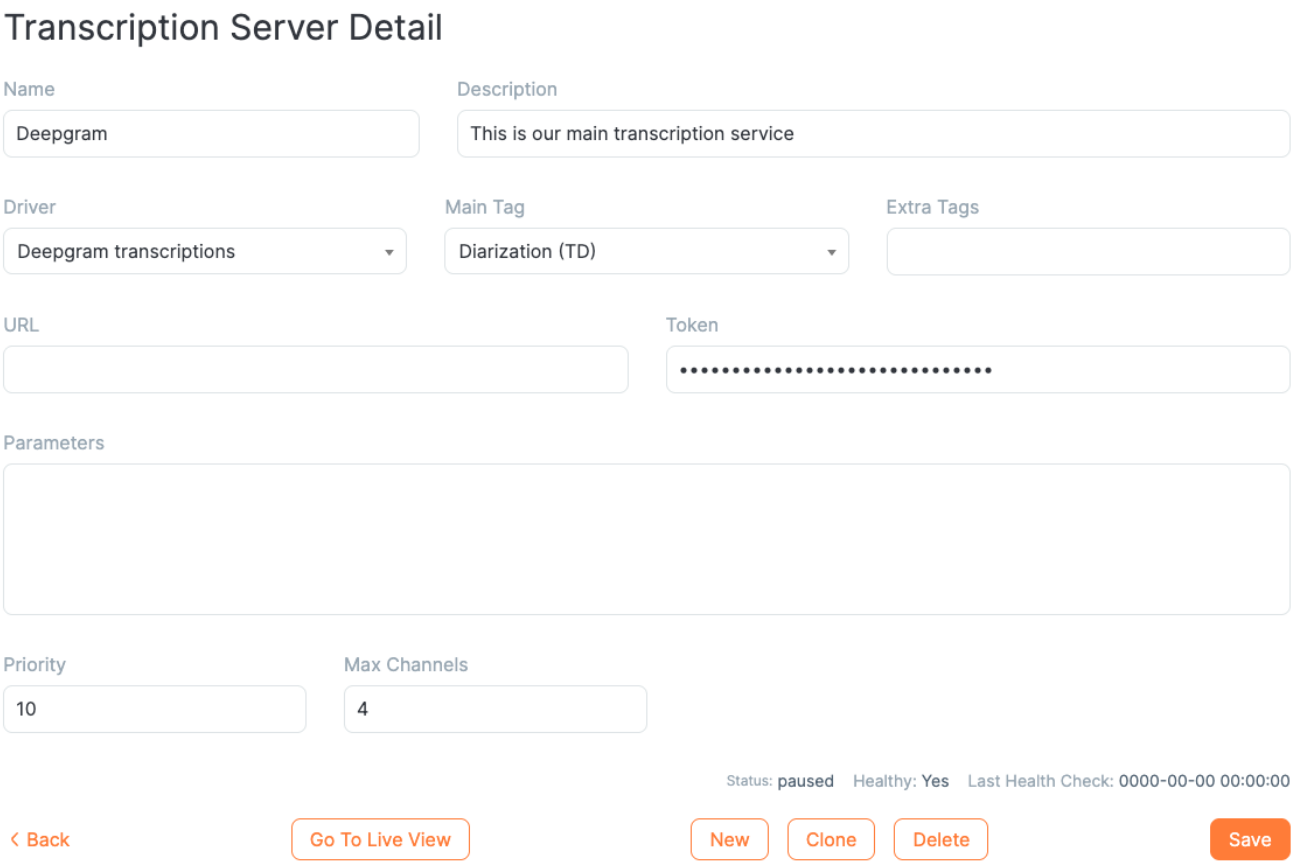
# Directory to watch for audio files
WATCH_DIR=/var/spool/asterisk/monitor/
```

then restart the service with `service uniloader-transcribe restart`.

Take a look at how the recordings are named in the `WATCH_DIR`: you need to be able to extract the call's unique-id from the rest of the file name. For example, if your recordings are named `audio-134.567.wav`, you would keep everything between the last dash and the extension. To do this, set these properties in QM (Configuration → Edit system parameters):

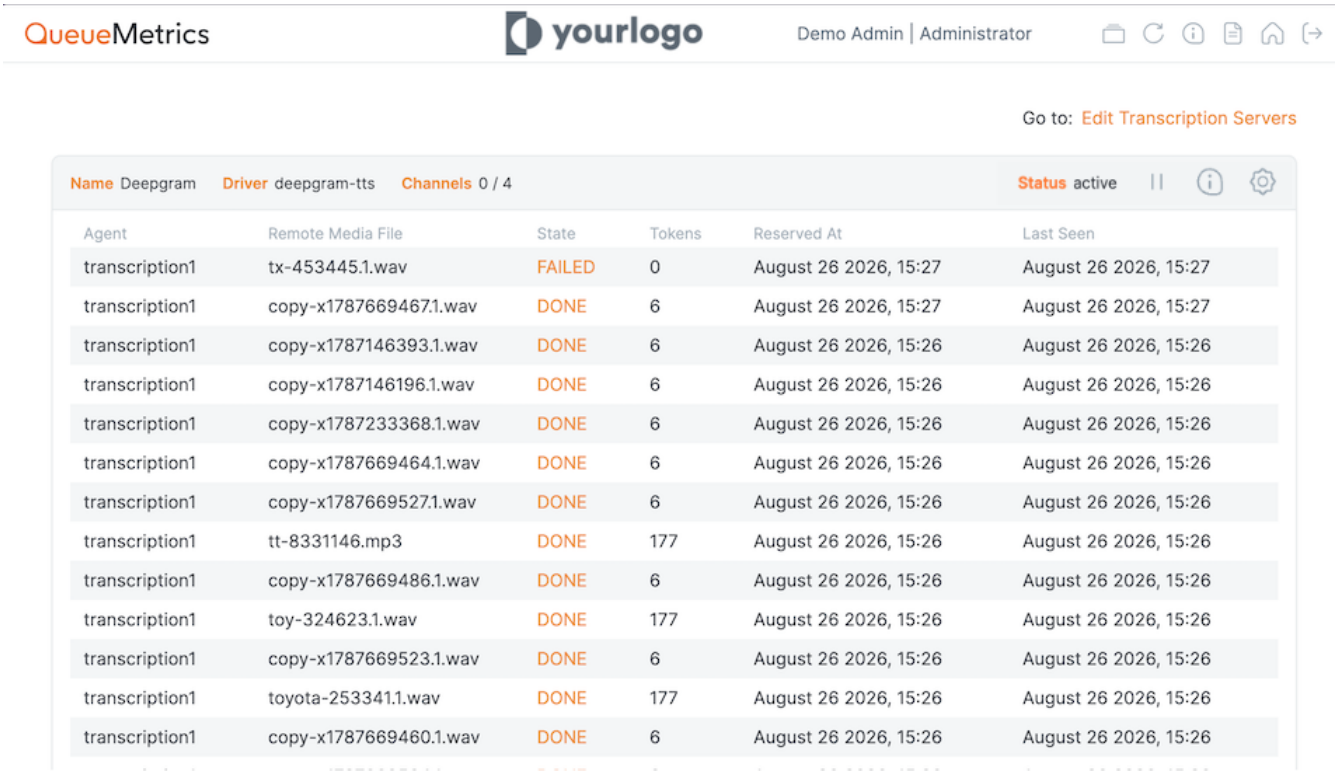
```
llm.captureUniqueId=^.+?-(.+).wav
llm.verbose=true
```

Next, go to Configuration → Transcription Servers and create a transcription server. Here we use the common Deepgram service as an example:

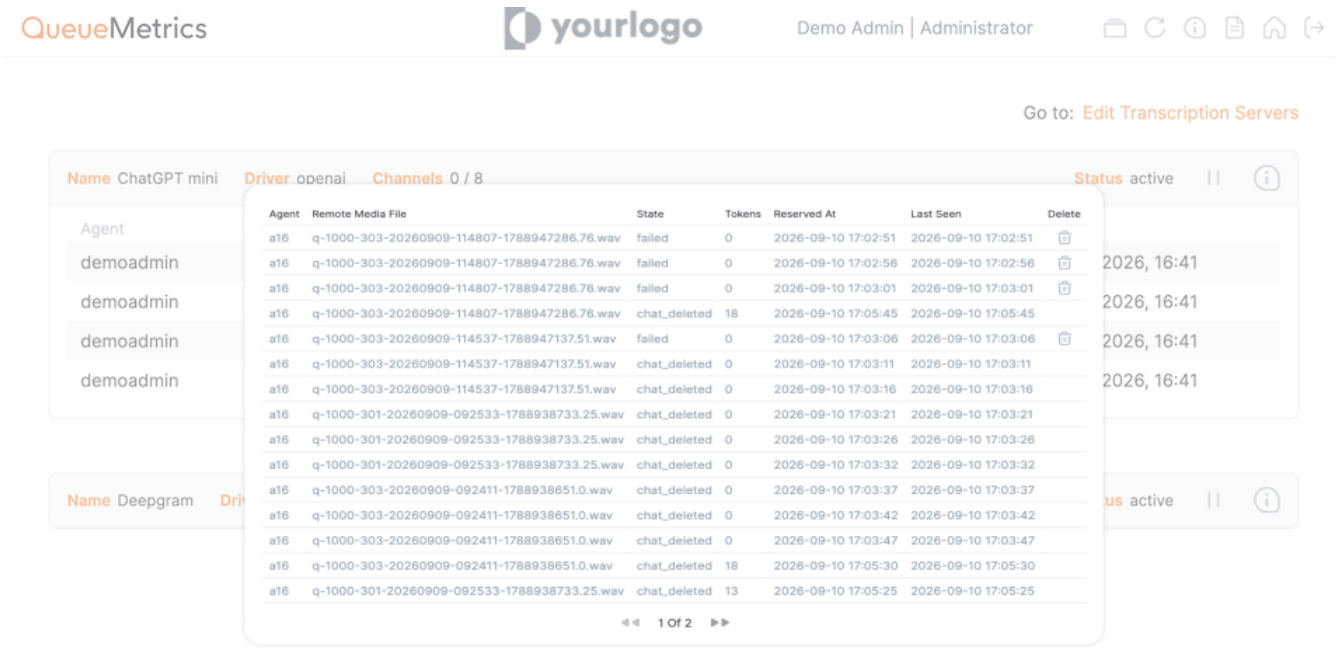


Enter any name you like, and select the driver "Deepgram transcriptions". Enter your API key, and set the main tag for this server to "TD" (Transcription and Diarization) - this is essential, as it is how the transcriber finds the server. You can also choose the number of parallel channels, depending on your contract.

Within a few seconds, the transcription server should start pushing transcriptions to QM. To check that it is working, click "Go to Live view" (or go to the home page and add the "Live transcriptions" box):



This shows all transcriptions currently in progress or just completed (in the last 5 minutes). You can freely stop and restart servers, and change their parameters and relative weight by clicking its Cog icon; the system reacts immediately. You can also see the detail of every transcription attempt handled by a server by clicking its Info icon:



From here you will also be able to delete a transcription channel if its status is in a terminal failed state (SKIPPED, FAILED, TIMEOUT), by clicking the trash icon if present. This will set the status to CHAT_DELETED for all attempted transcriptions of that specific recording, which tells Uniloader to retry the transcription. If the deletion is successful, the deleted channels will appear in the recent transcriptions job.

At this point, if you look up one of the calls that was just transcribed in a report and open its call detail → QA form, you will see the call transcribed and diarized. Clicking on a transcribed phrase plays back the matching audio at roughly the right moment. If a transcription is incorrect and you have the `CHAT_EDIT` security key, you can correct it.



By editing the settings in Uniloader, you can have finer control on how calls are transcribed; for example, you can control how many worker threads will be used for transcribing calls (flag `workers`), whether you want implicit diarization using a stereo recording (`stereo-agent`, remember to enable it in Asterisk though the option `x` of MixMonitor) and even the percentage of calls that you don't want to transcribe (`skip-prc`).

Automating Quality Assessment

Once a call has been transcribed - or is a chat/WhatsApp-type interaction that was naturally text-based to begin with - it can be automatically evaluated.

This automated evaluation uses the very same quality assessment tools already available in QueueMetrics, so you can see evaluations made by human graders alongside evaluations made by the AI.

The idea is that the two approaches work together: automated evaluation is used to flag potentially problematic situations, which are then reviewed by a human grader. This way, review effort is concentrated only where it actually brings a benefit, avoiding unnecessary work.

Just as with human grading, QueueMetrics lets you avoid evaluating every single call, and instead pick calls for evaluation based on who answered them and what their outcome was. For example, you might want to review all calls marked as a sale, plus a further 20% of the remaining ones. At the same time, you might want to evaluate 50% of the calls answered by newly-hired agents, but only 20% of those answered by more experienced agents. QueueMetrics finds the smallest possible set of calls that satisfies all of these criteria at the same time, so that you get the best possible result for the lowest cost.



A complete description of how the page criteria work can be found at the Grader's Page.

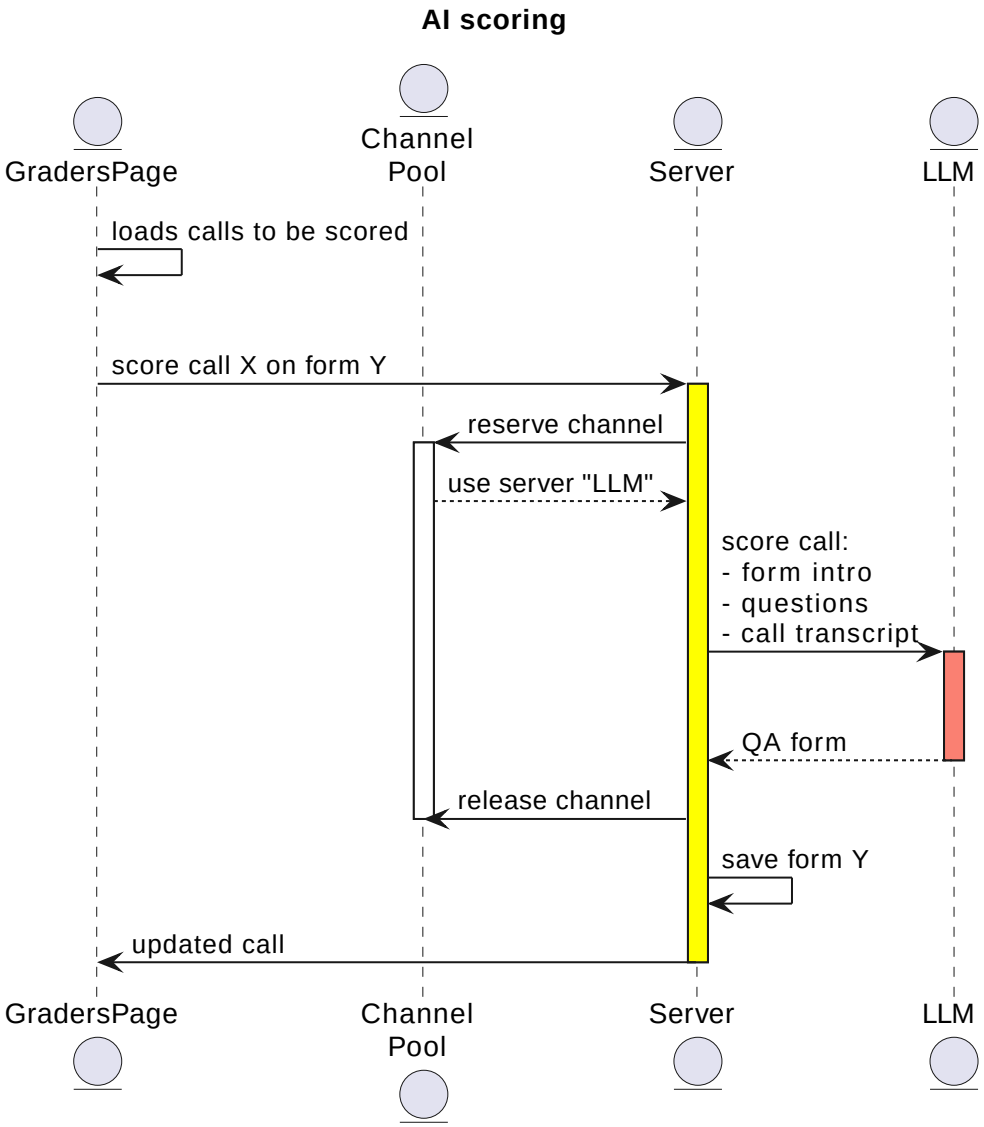
Automated evaluation works much like manual grading: a supervisor goes to the Grader page, sets the evaluation criteria and which calls should be selected, and presses the "AI Grading" button. Calls are then loaded, one after another, onto the page in front of them, and evaluated in real time by the configured AI models.

Evaluating calls requires a QA form to be filled in; this form can carry general instructions describing the overall evaluation criteria, and each individual question in the form can in turn carry its own, more specific instructions on what that particular question should look for.



The QA form used for AI Grading must have its "AI prompt" configured (in the form's own configuration page); AI Grading cannot run on a form that has no "AI prompt" set, or that is locked. See Configuring QA forms.

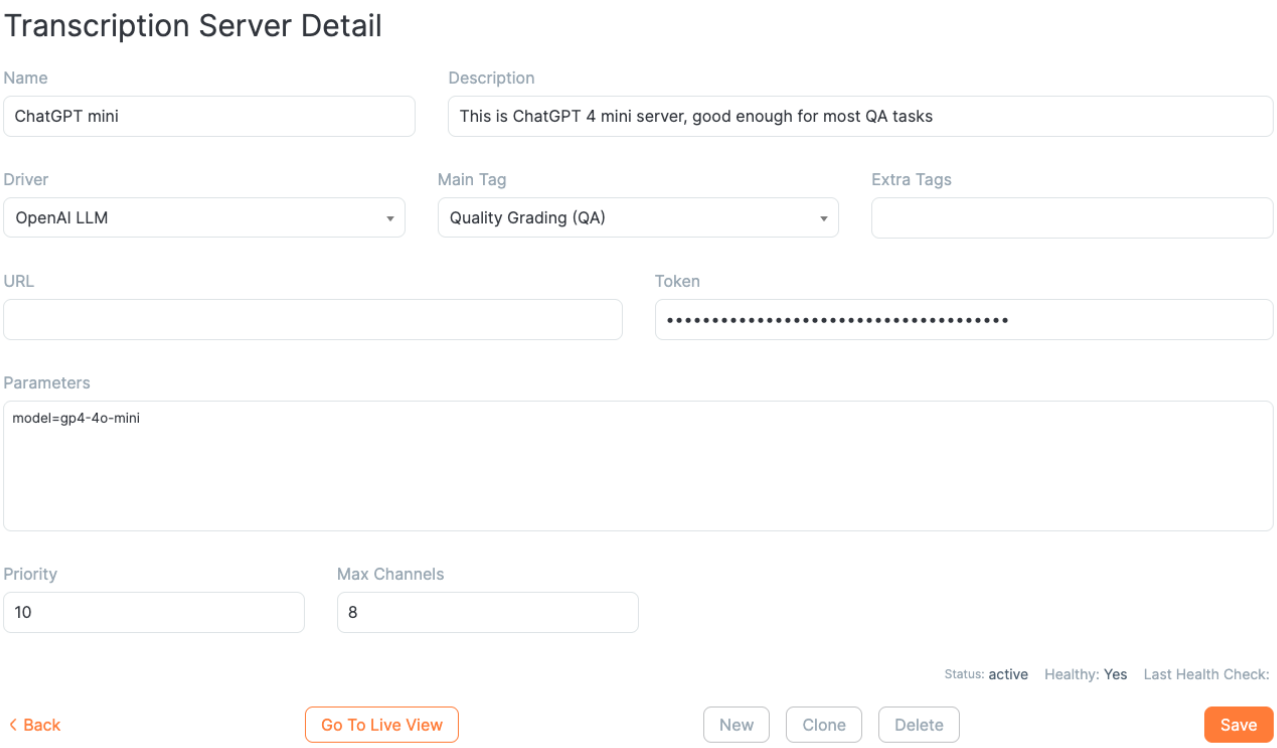
As this process moves forward, forms are filled in and become available in the call detail, just like manually graded ones. The AI can leave comments tied to each call and to each question of the form, explaining why it gave a particular evaluation. Forms filled in by the AI are always clearly distinguishable from the others in quality reporting.



It is possible to simply leave the AI Grading page open: it will keep loading and evaluating calls on its own until there are no more calls left that match the requested criteria. If, for any reason, you want to interrupt the activity - to run a different search, or simply because enough data has been gathered - this can be done at any time, by navigating back or to a different page.

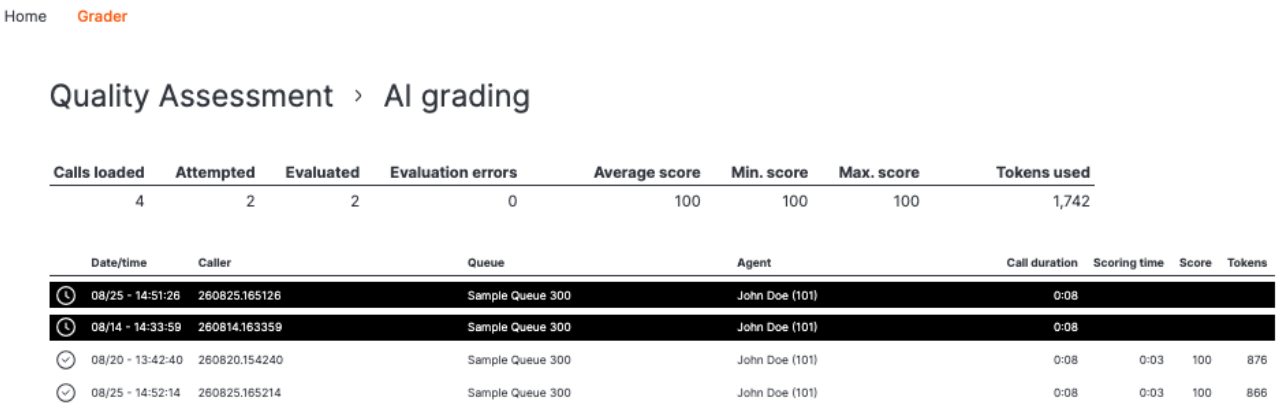
Automated QA in practice

To start, you need to create at least one LLM server to do the evaluation. This is very similar to what you did above when setting up transcriptions. Go to Configuration → Transcription Servers, and create a transcription server for ChatGPT:



Select the driver "OpenAI LLM", and make sure you enter your API key and the model you want to use. Most LLMs expose a large set of configurable parameters, but this should be enough to get you started.

Now go to the Home page and add the "QA reports" block; from there, click on the Grader's Page. For Automated QA to work, the QA form must include a prompt telling the LLM what you expect, and it helps if every QA item also has an explanation of what that question means. We supply a "Sample AI inbound" QA form to get you started: select it and click "AI Grading".



The page starts empty and progressively loads calls; as each call is scored, it changes colour and the statistics are updated. This continues until you navigate away from the page, or until there are no more calls matching the requested criteria, at which point a "No more calls" line is displayed.

Calls change colour and icon depending on their state:

Call status	Background
Still to be scored	White
Being checked	Brown
QA ongoing	Black
In error, retrying	Light red
In error, failed 3 times	Red

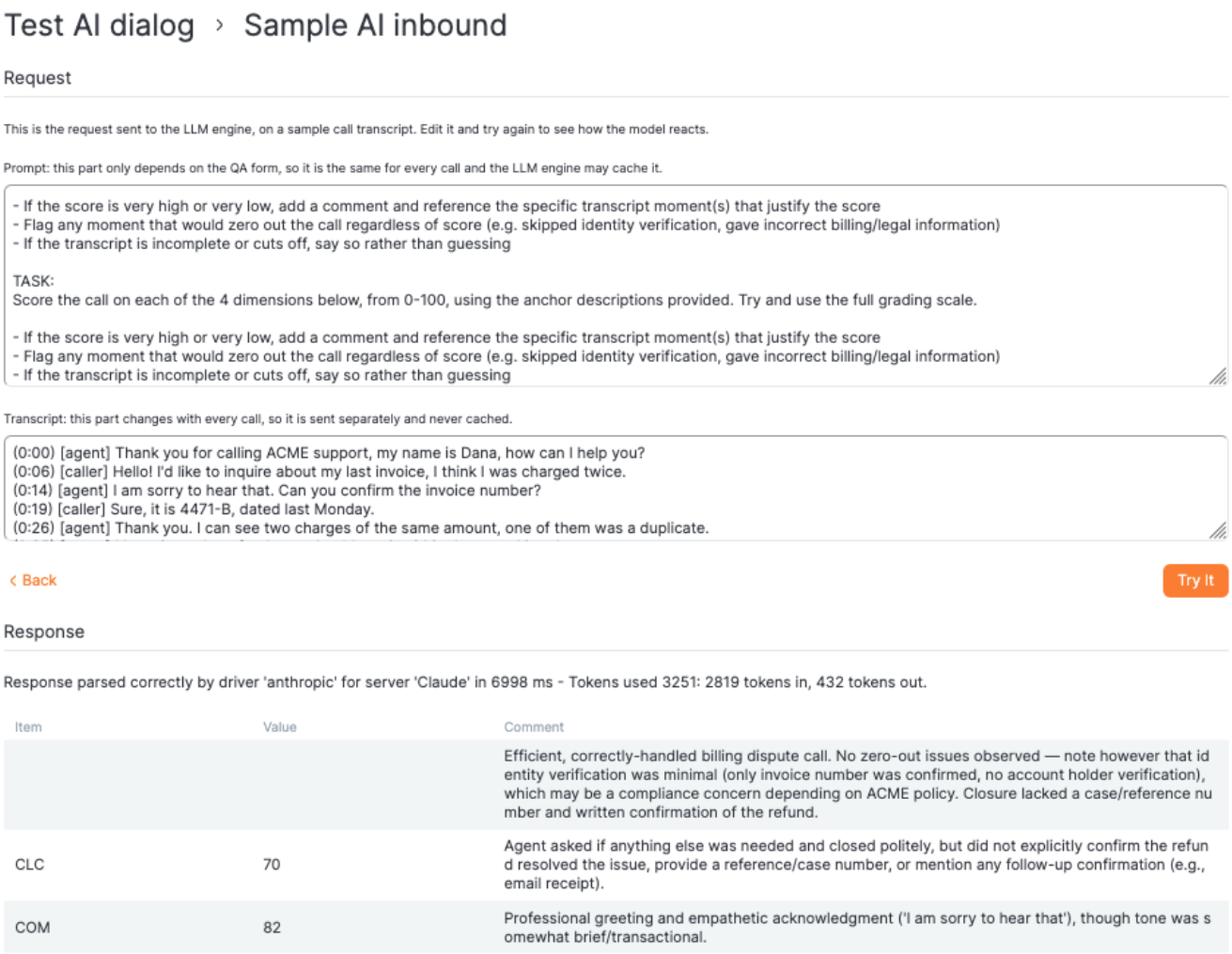
If you then look up one of these calls, you will find the QA evaluation loaded on its QA form. Note that the "Grader" section of the QA form shows "AI" to distinguish these forms from the ones evaluated by humans.



Once a call is scored, the generated QA form can be reviewed immediately by clicking on the pencil icon that appears on the right.

Troubleshooting automated QA

How do you make sure the script and the descriptions you provide work well with the LLM model of your choice? From the QA form editor, just below the script, click "Test AI dialog".



This opens a page showing the complete prompt that will be sent to the LLM, made up of:

- Your AI prompt
- The list of questions, their meaning, and their allowed values
- The different quality levels
- A sample call

You can edit this prompt until you are satisfied with it; clicking Run sends it to the currently configured LLM (tag "QA") and prints the result, both as interpreted by QM and, below it, as a raw JSON response.



You should replace the sample call with one or more real transcripts, to test how they are actually evaluated. For your convenience, the call transcription itself is input in a separate text area.

Once you are happy with the updated prompt, note down the changes you made and apply them to the script and the questions in their respective editors.

Transcription drivers

For a transcription driver to work, it must be supported both by QueueMetrics and by Uniloader: QueueMetrics needs to know the driver in order to let you configure a server that uses it (see Setting up the server pool), while Uniloader is the component that actually sends the audio and retrieves the transcription from that service. A driver that is only known to one of the two will not work.

Transcription servers support two kinds of work (see Setting up the server pool for the **TR** and **TD** tags): plain transcription (**TR**), and transcription with diarization (**TD**), i.e. also recognizing who is speaking.

OpenAI

- **Driver**: `openai-tts` (OpenAI transcriptions)
- **Token**: your OpenAI API key (required)
- **URL**: fixed - this driver always talks to the official OpenAI API, so the **URL** field can be left empty
- Supports both **TR** and **TD**
- **Parameters**: defaults to `model=base`, `response_format=verbose_json`, `timestamp_granularities[]=segment`. Any other key can be added to override these or supply further ones - for example, to force a specific model (`model=gpt-4o-transcribe`, `model=gpt-4o-transcribe-diarize`, `model=whisper-1`, ...) or to switch to the newer, flat `response_format=diarized_json` format.
- Since this is the official OpenAI API, QueueMetrics also reports the number of tokens used by each transcription.
- **Max Channels**: OpenAI does not publish a simple "concurrent calls" number for its transcription models - instead, it enforces a requests-per-minute limit that scales with a usage

tier based on your account's cumulative spending, going from a handful of requests per minute on entry-level tiers up to several hundred (or more) on higher ones. As a rule of thumb, set **Max Channels** comfortably below the requests-per-minute limit shown for your account, and re-check it from time to time, as both your tier and OpenAI's own limits can change.

Example parameters for diarization (tag `TD`):

```
model=gpt-4o-transcribe-diarize
response_format=diarized_json
chunking_strategy=auto
```

Example parameters for transcription only (tag `TR`):

```
model=whisper-1
response_format=verbose_json
timestamp_granularities[]=segment
```

OpenAI-compatible API

- **Driver**: `openai-api-tts` (OpenAI-compatible transcriptions)
- **URL**: required - point it at your own OpenAI-compatible transcription endpoint (for example a self-hosted Whisper server, or a third-party service exposing an OpenAI-compatible API)
- **Token**: as required by that particular service
- Behaves exactly like the OpenAI driver above - same **Parameters**, same support for **TR**/**TD** - just against whatever endpoint you configure.
- Since this is not the official OpenAI API, no token usage is reported for this driver.

A number of open-source, self-hostable projects expose an API compatible with OpenAI's transcription endpoint and can be used with this driver, for example:

- WhisperX API server: https://github.com/Nyralei/whisperx-api-server
- Speaches (formerly faster-whisper-server): https://github.com/speaches-ai/speaches
- openedai-whisper: https://github.com/matatonic/openedai-whisper
- docker-whisper: https://github.com/hwdsl2/docker-whisper



This list is not an endorsement of any specific project; verify licensing, support and diarization capabilities before choosing one for production use.

Max Channels here is not bound by a vendor's contract, but by the actual capacity of the server you deployed: how many transcriptions it can run at once depends on its own hardware (CPU/GPU) and configuration, so set **Max Channels** to match what that specific installation can sustain, not an arbitrary number.

Example parameters for diarization for WhisperX Api Server (tag `TD`):

```
model=base
response_format=verbose_json
timestamp_granularities[]=segment
language=en
align=true
diarize=true
min_speakers=2
max_speakers=2
```

Example parameters for stereo transcription for WhisperX Api Server (tag `TR`) - make sure that your Uniloader has parameters `--task-tags TR --stereo-agent R` (transcription only, agent speaks on the right channel):

```
model=base
response_format=verbose_json
timestamp_granularities[]=segment
language=en
align=true
```

Deepgram

- **Driver**: `deepgram-tts` (Deepgram transcriptions)
- **Token**: your Deepgram API key (required)
- **URL**: fixed - this driver always talks to the official Deepgram API, so the **URL** field can be left empty
- Supports both **TR** and **TD** - diarization is enabled by selecting a diarization model, rather than a simple on/off switch
- **Parameters**: defaults to `language=en`, `paragraphs=true`. Any other key can be added to override these or supply further ones - for example to pick a different `model` (e.g. `nova-2`), a different `language`, or a specific diarization model.
- Deepgram has no concept of tokens: QueueMetrics instead reports the (rounded) duration of the audio, in seconds, in place of a token count, for billing/telemetry purposes.
- **Max Channels**: Deepgram publishes explicit concurrency limits per plan - at the time of writing, its Pay-As-You-Go plan allows up to 100 concurrent transcription requests over its REST API, with Growth and Enterprise plans allowing several times that. Pay-As-You-Go and Growth limits cannot be raised without moving to a different plan, so **Max Channels** should not be set above what your plan actually allows.

Example parameters for diarization (tag `TD`):

```
language=en
paragraphs=true
diarize_model=latest
```

Example parameters for transcription (tag `TR`):

```
language=en
paragraphs=true
smart_format=true
```

KrokoAI

- **Driver**: `krokoai-tts` (KrokoAI transcriptions)
- **URL**: required - point it at your KrokoAI transcription endpoint. The URL is a websocket, so it could be something like `ws://my.kroko.server:8080`
- **Token**: not needed
- **Parameters**: not needed
- Supports **TR** only - no diarization
- Duration-based: QueueMetrics reports the (rounded) duration of the audio, in seconds, in place of a token count, for billing/telemetry purposes.

LLM drivers

The LLM models used for Automating Quality Assessment are configured the same way as transcription servers (see Setting up the server pool): add a server tagged **QA**, using the driver for the LLM provider you want to use, an API access token, and the model-specific parameters described below. If **URL** is left empty, QueueMetrics uses the provider's default endpoint.

Unlike the transcription drivers described above, LLM drivers do not depend on Uniloader at all: the AI grading evaluation is performed entirely by the QueueMetrics server, which talks to the LLM provider directly. This means a new LLM driver can be made available to QueueMetrics without needing a matching Uniloader update.



AI models and providers evolve very quickly. The information below (default models, endpoints and parameters) may not be up to date by the time you read it - please double check with the provider's own current documentation, or with QueueMetrics support, before relying on it.

OpenAI

- **Token**: your OpenAI API key
- Default **URL** (used when left empty): `https://api.openai.com/v1/chat/completions`
- **Parameters**:
  - `model`: the OpenAI model to use. This is mandatory - there is no default. The model must be from the "4o" family or later, as older models cannot guarantee the structured output this integration relies on. As a starting point, `gpt-4o-mini` is a cheap and fast choice; if scores turn out not to be discriminating enough, `gpt-4o` can be used instead.

- `temperature`: optional, defaults to `0`.

Example parameters (tag `QA`):

```
model=gpt-4o-mini
```

Anthropic

- **Token**: your Anthropic API key
- Default **URL** (used when left empty): `https://api.anthropic.com/v1/messages`
- **Parameters**:
  - `model`: the Anthropic (Claude) model to use. This is mandatory - there is no default. A recent, mid-tier Claude model is recommended as a starting point. At the time of writing it could be `claude-haiku-4-5-20251001`.
  - `max_tokens`: optional, defaults to `1024`.
  - `anthropic-version`: optional, defaults to `2023-06-01`.
  - `cache_control`: optional, no default - when absent, prompt caching is not used at all. See the note below.

Example parameters (tag `QA`):

```
model=claude-haiku-4-5-20251001
max_tokens=1024
```

Prompt caching. Every grading request QueueMetrics sends is made of two parts: the common part - the task specification, the list of items and the rating scale, all of which come from the QA form and are therefore identical for every call graded with that form - followed by the transcript of the call being graded, which is obviously different every time.

Anthropic can cache the common part, so that from the second call onwards it is not re-processed (and not charged) at full price. To turn this on, add a `cache_control` parameter: its value is a JSON object that is passed to Anthropic exactly as you write it.

```
model=claude-haiku-4-5-20251001
max_tokens=1024
cache_control={"type": "ephemeral", "ttl": "1h"}
```

The `ttl` field selects how long a cached entry survives: `1h` as above, or the default of five minutes if you leave it out (`cache_control={"type": "ephemeral"}`). Pick the one-hour lifetime when calls of the same form are graded minutes apart, and the default when grading runs in bursts. Writing an entry costs slightly more than a normal request, and the longer lifetime costs more still, so it only pays off if the entry is actually read back by a later call.

If present, `cache_control` is applied only to the common part. The transcript is always sent as a separate uncached block.

Note that Anthropic only caches a part that is long enough - the threshold depends on the model and is a few thousand tokens. A QA form whose common part is shorter than that will simply not be cached, with no error and no visible difference: the request goes through normally.

QueueMetrics Tasks

Tasks are a general mechanism to send some QM users a set of actions to perform. This could be, e.g. an agent which call has been reviewed, or a grader knowing that an agent has accepted a task.

Tasks are handled on a separate page that works as a "mailbox" for incoming tasks and lets you check the tasks you generated.

Differently from the way QueueMetrics usually works, tasks also allow for "direct URL linking", so that you can have a short URL that is published over RSS or email and allows the user access to the task. Before access is granted, anyway, user authentication is performed.

Tasks are strictly typed, that is, you can have only a given set of them. In order to handle a basic one-to-one communication, a generic *Note* task is provided.

Tasks are shown on the top of each page generated by QueueMetrics - when you have new tasks, you see a flasing NEW icon. A user's tasks are exported over RSS - a RSS icon allows easy subscription by clicking on it.

Access to the Task subsystem is controlled by the key TASKS.

The task page

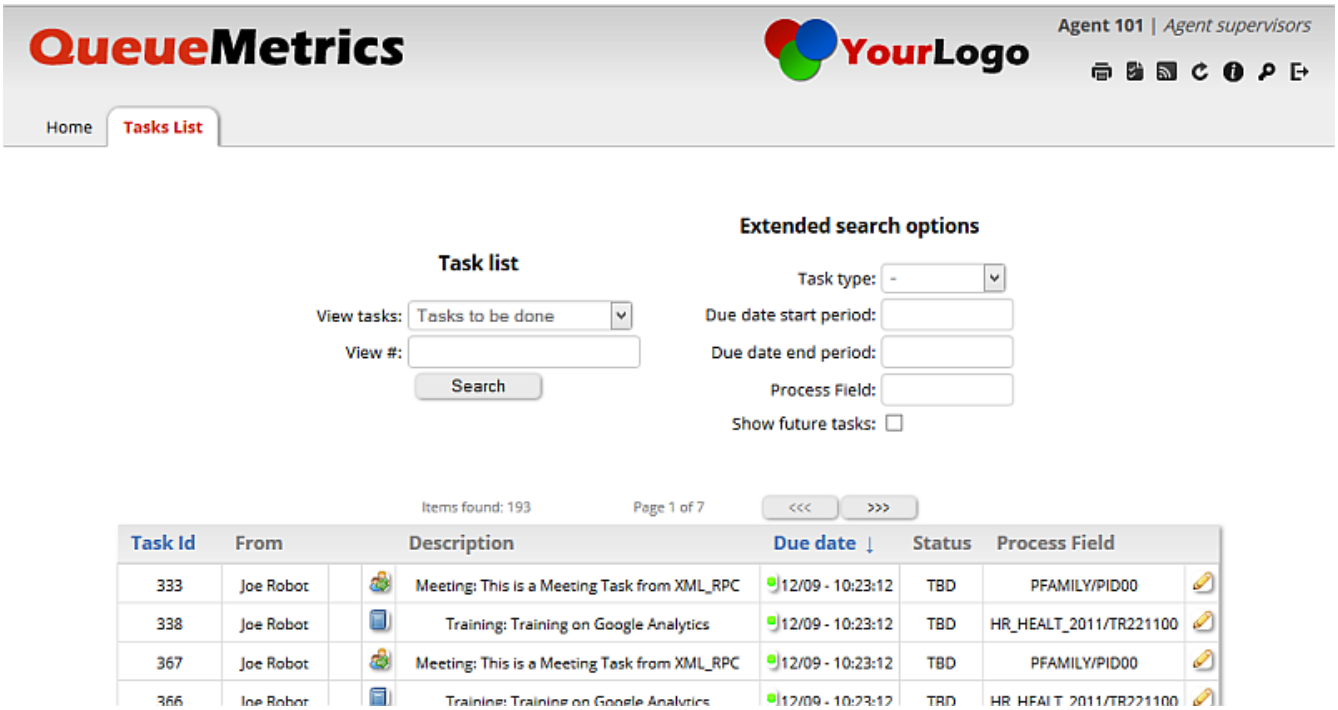
If the user is enabled for tasks, they will see a couple of new icons on the top every page:



The yellow icon leads to the tasks page; if you have new tasks, it is displayed together with a flashing NEW icon.

The orange icons lets you access a RSS feed for all the tasks for the current user (see below on how to set this up).

When you click on the yellow icon, you are lead to the tasks editor page:



On top of the page, you see a selector with:

- **Tasks to be done**: tasks addressed to me that I have to complete
- **My group**: Tasks for my class that I can address
- **Tasks I completed**: the set of tasks that I completed (and their completion codes)
- **Task I sent**: The tasks I sent somebody else (if any)
- A box to access directly a task by number (it will be found only if the current user can see it: that is, he is the assignee, or a member of the group it is assigned to, or the assigner).

You can optionally make the list shorter by selecting:

- A task type
- The period the task was supposed to be due
- An option to display future tasks as well
- The Process field which is formed by the Process Family / Process ID and can be inserted only via XML-RPC.

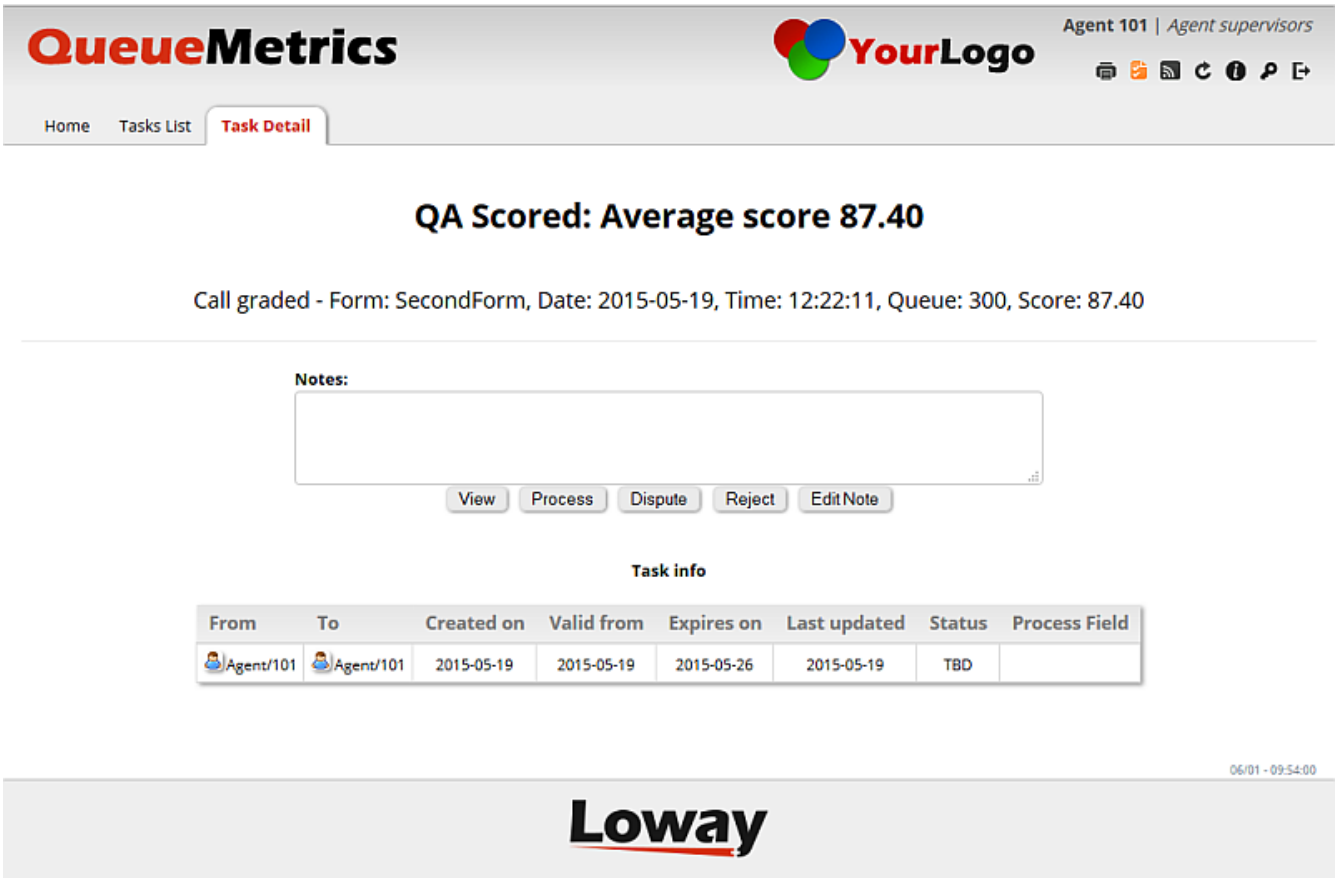
For each task, the following information is displayed:

- An icon displays whether the task is overdue (red), about to be due (yellow) or yet to be due (green)
- The task sender
- The task ID, that is an unique number that lets you access a specific task directly
- An icon for the type of task involved
- The task description
- The due date (if any).

- The status: it can be **TBD** (to be done) or **Completed** or **Disputed**.

The list is paged, so you only see the first 10 elements or so.

When you click on a the pencil icon of a task, it is displayed in a panel like the one below:



This is a QA task; tasks of other kinds may differ slightly in their appearance and behavior.

The top box contains a description of the task; while the bottom box contains a note that can be edited freely.

The following buttons are typical:

- **View**: in case the task is related to something else (like in this case, a QA form) this button lets you access a form with the related information.
- **Process**: the user acknowledges having seen the task
- **Dispute**: the user acknowledges having seen the task, and disputes the given score
- **Edit Note**: lets you save the edited Note field.

By the bottom of the page, general information on the task is displayed:

- **From**: the originator
- **To**: the receiver
- **Created on**: the day it was created
- **Valid from**: the day the task started being addressable

- **Expires on**: the deadline for completing the task
- **Last updated**: the date this task was last processed
- **Status**: the current task status

If the task is related to other tasks, a task history table is shown. In this table user could find, for each related task, information about:

- **Task Id**: the unique identifier of the related task
- **From**: the originator
- **To**: the receiver
- **Created on**: the day it was created
- **Valid from**: the day the task started being addressable
- **Expires on**: the deadline for completing the task
- **Last updated**: the date this task was last processed
- **Status**: the current task status
- **Abstract**: the task title
- **Notes**: notes associated to the task (if any)

When a user processes or disputes a task, it is possible to ping a specific URL if defined within the property default.tasks.pingURL. Further details about this property can be found within the System Preferences chapter.

Extended task reports

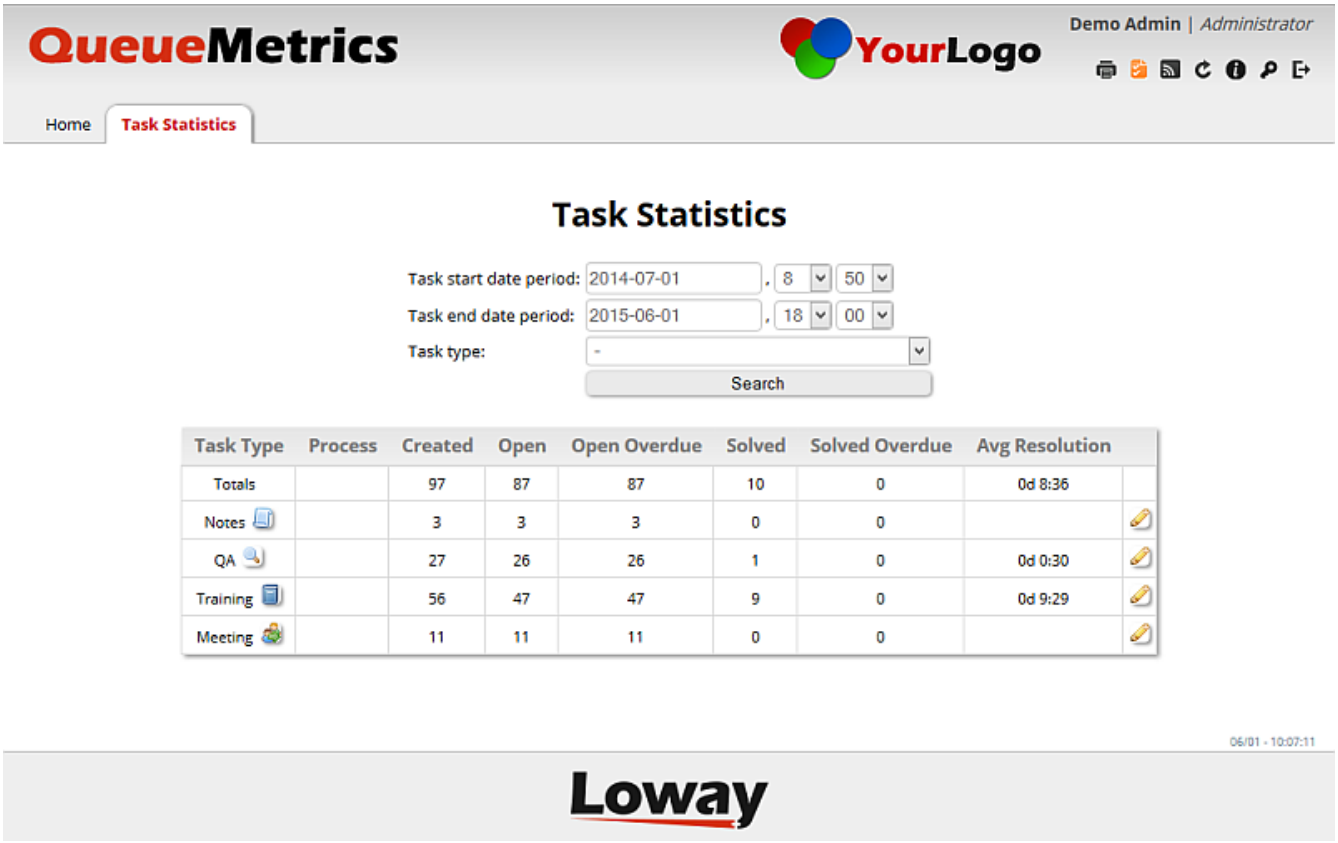
Users holding the special key TASKS_VIEWALL can monitor all tasks on the system - they will have three new choices in the drop-down menu:

- All tasks
- All tasks to be done
- All tasks done

Combined with the advanced search criteria, they can be used by the superuser to monitor the well-being of tasks being handled on the system.

Task Statistics

Users holding the special key TASKS_REP can access the tab showing task statistics where it is possible to Search by Tasks start/end date and by Task type.



The result of such Search shows, for each Task Type:

- The Process field
- Created/existing tasks for each task Type
- How many tasks are Open
- How many are Overdue
- How many have been Solved within the due date
- How many were Solved late
- The Average Resolution Time

The "View Details" icon leads to Task Details where one can see the Status of each Task, the users involved, the Task Creation date/time, Validity From and Expiry date/time.

Tasks for groups: Class tasks

Tasks can be addressed to user Classes as well as users. This is useful e.g. for supervisors, when you mean "Any supervisor", not a specific one.

When you act on such a task, you get the ownership, that is the task is changed as to signify it was originally sent to the acting agent.

You do not see outstanding class tasks with the "new" icon, as it is for personal tasks.

When a task is processed by a person, the acting person is displayed as "PersonsName on behalf of AgentClass".

Task validity and expiration

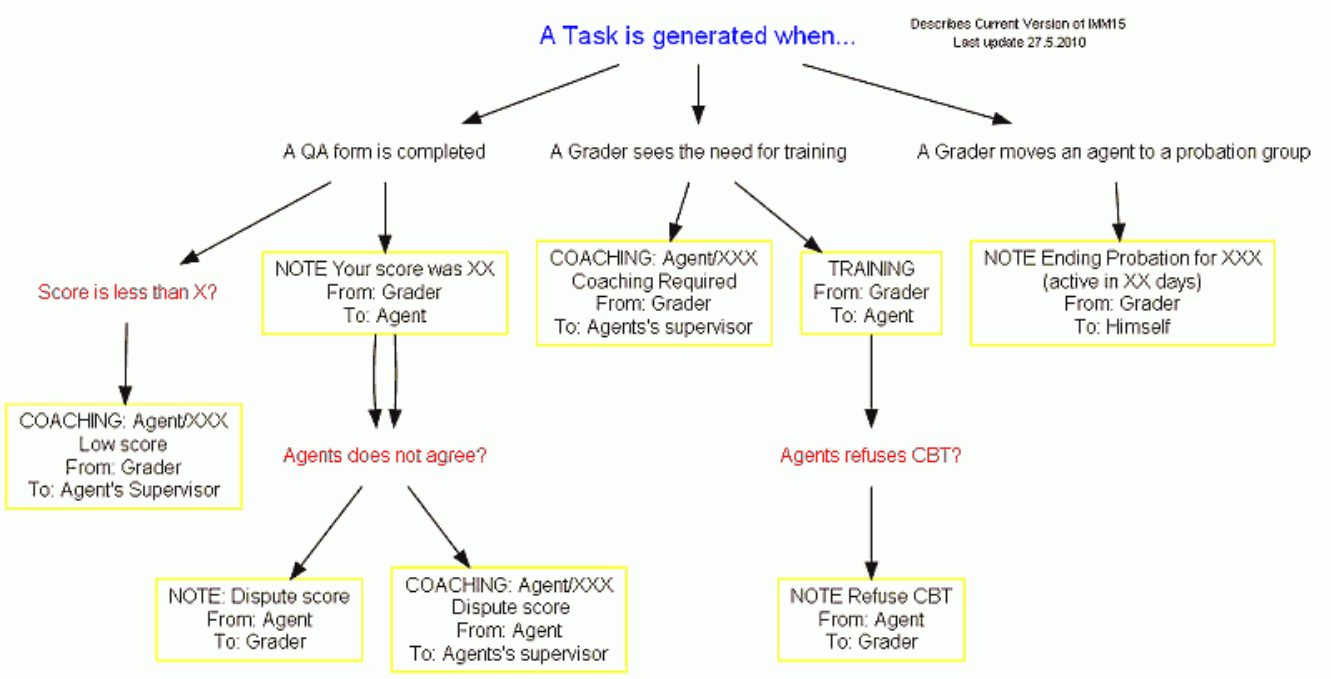
Each task has a "Valid from" and "Due by" dates. This makes it possible to create tasks that:

- Show up in the future (so you can add yourself a task to complete some future action)
- Be notified of the expiration dates.

All tasks that do not have a specified expiration date are to expire by 7 days after they are added to the system.

Types of tasks handled by QueueMetrics

The following task types are currently defined, as explained in the graph below:



Call QA Graded Task

This task is generated when someone grades a call processed by an agent, and it is sent to the agent that processed the call.

The title is. "Call graded on queue XXXX processed on XXX".

When you click on the task, you are lead to the correct QA form for that call. You can enter comments on that call and ACCEPT/DISPUTE the grading. When you process that task, a new task of type NOTE is sent to the grader.

If you DISPUTE the grading, a NOTE task is sent out to the person who graded the call and a separate one to the supervisor of the agent who disputed (if any).

Note Task

This task simply displays a text and can open up a URL when clicking on the Accept button. This is used to notify on an ACCEPT/DISPUTE. The task note is sent when some events happen in

QueueMetrics like, for example, when someone grades an agent call, or an administrator/supervisor changes group for a particular agent. The task note could also be sent through an external XML-RPC call.

Training Task

This is a special task that contains a URL and is sent to the agent by the grader to improve their skills. This is used to send links to CBTs where URLs can be defined through the Configuring CBTs (Computer Based Training) pages. This task could be sent through the Performance Tracker Result Page, the Grader Report page and the Training & Coaching Page.

An agent can accept (open up the URL) or dispute this task. (This task basically behaves like a Note but has a Dispute button as well).

Meeting Task

Meeting Tasks (also called Coaching Tasks) can be sent though the Performance Tracker Result Page, the Grader Report page and the Training & Coaching Page. This task includes a message and a date and time of the meeting event. This is a task that can also be sent to an agent via XML-RPC.

An agent can accept or dispute/refuse/reject this task.

RSS data export for tasks

Any QueueMetrics user can subscribe to their own "task inbox" as RSS feed. You can access the RSS feed and subscribe to it in most browsers just by clicking on the orange RSS icon that is displayed next to the Tasks icon.

In order to display the RSS icon, you need to set the properties as described in the chapter below: "Setting computed URLs"



This access method offers no security at all, so any user who has access to any other user's workstation can access the other person's RSS feed.

The RSS feed contains only the the titles of current tasks to be done and a link; when the link is clicked, the user is authenticated (if necessary) and lead to the tasks page, so basically clicking on the RSS link is the same thing as going to the Tasks page and selecting a task in order to view its details. If the user that logs on is not the intended recipient of the task, the task is not displayed.



It is advisable that the RSS polling speed be set to a minimum on the RSS reader (like once every 20 minutes or so) in order to minimize the load on the QM server.

Setting computed URLs

In the configuration files, you should set two properties before using RSS tasks:

```
url.qm=http://qmserver.my.corp:8080/queuemetrics
```

```
url.rss=http://qmserver-rss.my.corp:8080/queuemetrics
```

In general, the servers *qmserver* and *qmserver-rss* should be the same machine - this is necessary because if your RSS reader is embedded in your browser, every time it requests a RSS feed, it also logs you off from the current session.

Payroll data in QueueMetrics

QueueMetrics is able to produce extensive payroll information suitable to be exported to a third party application.

QueueMetrics is able to extract most of the data required for payroll generation from the agent sessions information reported by QM. Through some special pages a supervisor can edit and correct that information before exporting.

How it works

Payroll is based on agent sessions and is displayed a as separate "micro-application" within QM, available only to agents holding a special key. Each agent has a specific payroll code associated through the agents configuration page. This *payroll code* identifies the agent in the payroll generated data file.

To Supervisors that can access to payroll subsection will be shown all "punch times", for a given period of time, with the opportunity to zoom in, display incomplete sessions and display sessions for one specific agent. Supervisors holding a second special key are allowed to correct - enlarge or restrict - agent sessions, given a set of rules that avoids overlaps with other agent activities, to be further specified. A third key allow "enlarging" agent sessions, i.e. causing the cost to be more for the call center. Payroll extraction is just manual: the user should inspect payroll data and download payroll trace files, to be loaded into external WR Timetracker for further processing. Supervisors are able to see all punches and/or to filter out some of them by agent group and/or location. All activity is logged into the QueueMetrics log subsystem.

Payroll information generated by QueueMetrics is based on a specific output file format:

- **ADP** "punch files", that basically handle the time an agent has been available under ACD

but the underlying structure let us able to expand the file format selection by implementing a specific file writer.

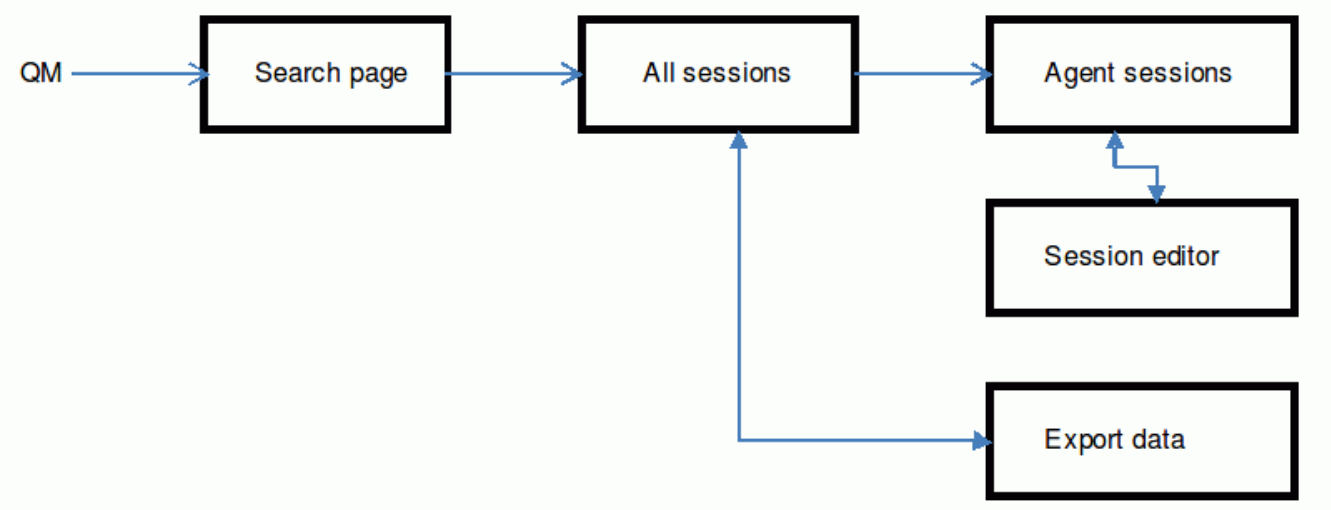
To summarize, here are the keys that limit user's permissions when working with Payroll:

- Users allowed to check the payroll page will be marked by the key **PAYROLL**
- Users allowed to edit the queue_log records will be marked as **QLOG_EDIT**
- Users allowed to edit session data by making it longer will be marked by **QLOG_LNGR** (they must hold QLOG_EDIT as well).

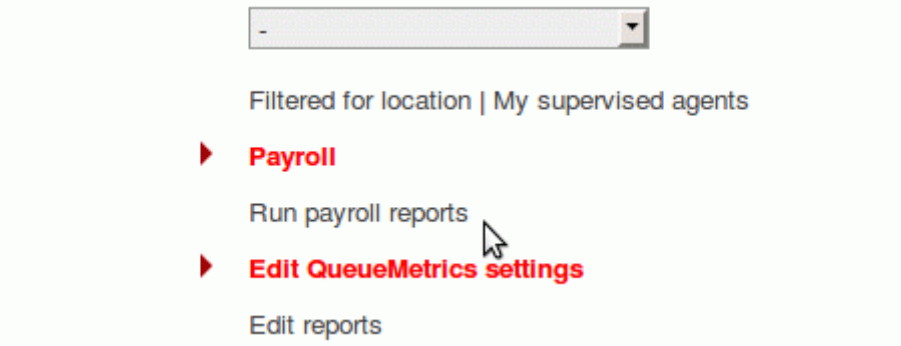
All queue_log editing is tracked and logged.

Payroll web pages

The payroll process happens on the following web pages:

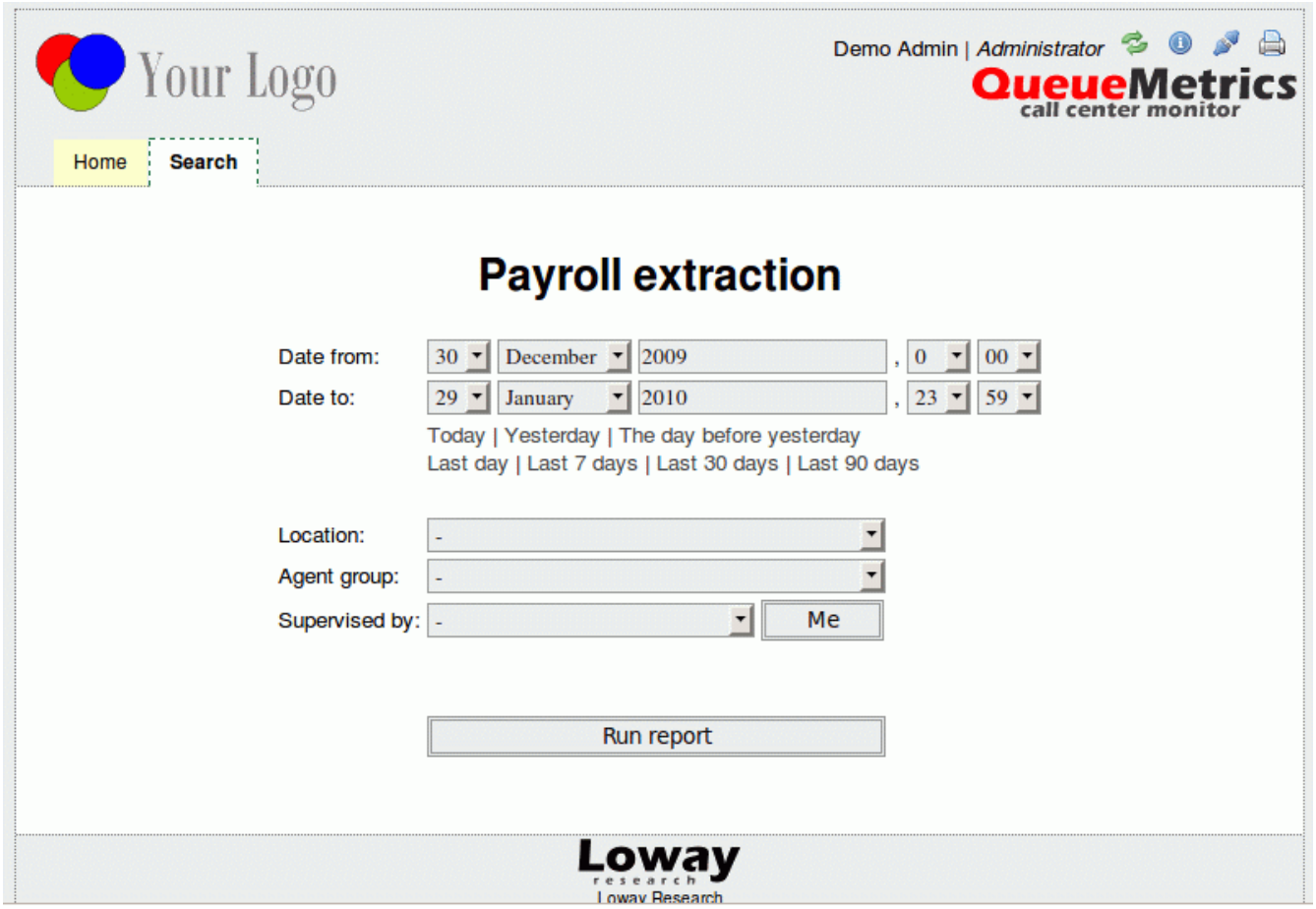


The Search page is linked from the main QM page.



The search page

The search page lets a supervisor search for session activity.

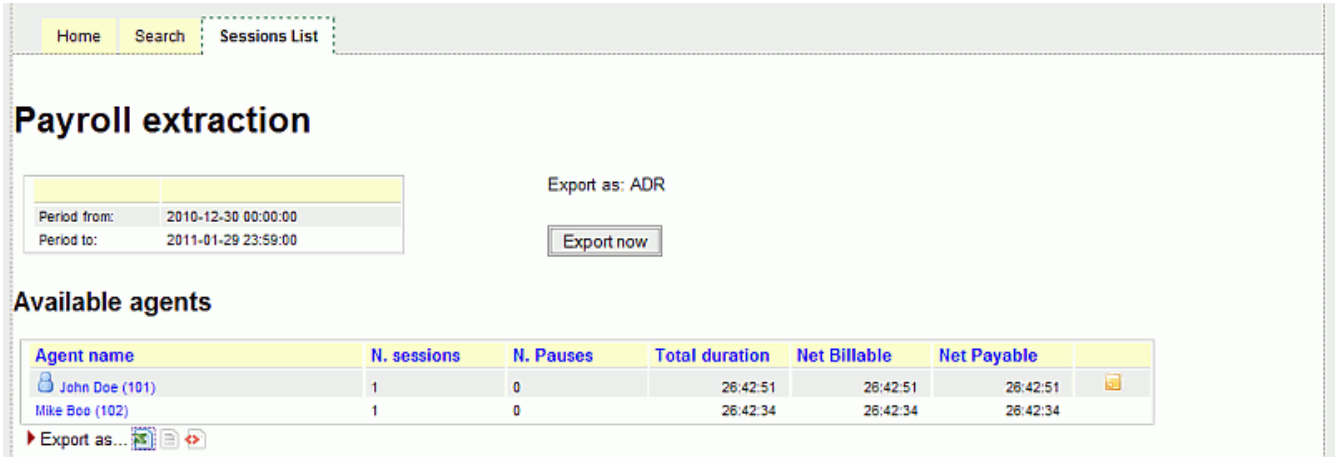


The page lets you select a time interval, or choose one of the pre-selected time intervals. It also lets you filter by one criteria of the following ones:

- Location
- Agent group
- Supervisor (a button "me" is available if the user is a knows supervisor, and will pre-select the current supervisor)

The Sessions page

The payroll extraction page lets you preview the data that will be downloaded in the "punch" format. If you click on an agent's name, then you will be lead to a page where all sessions for that agent will be shown for the specified time interval.



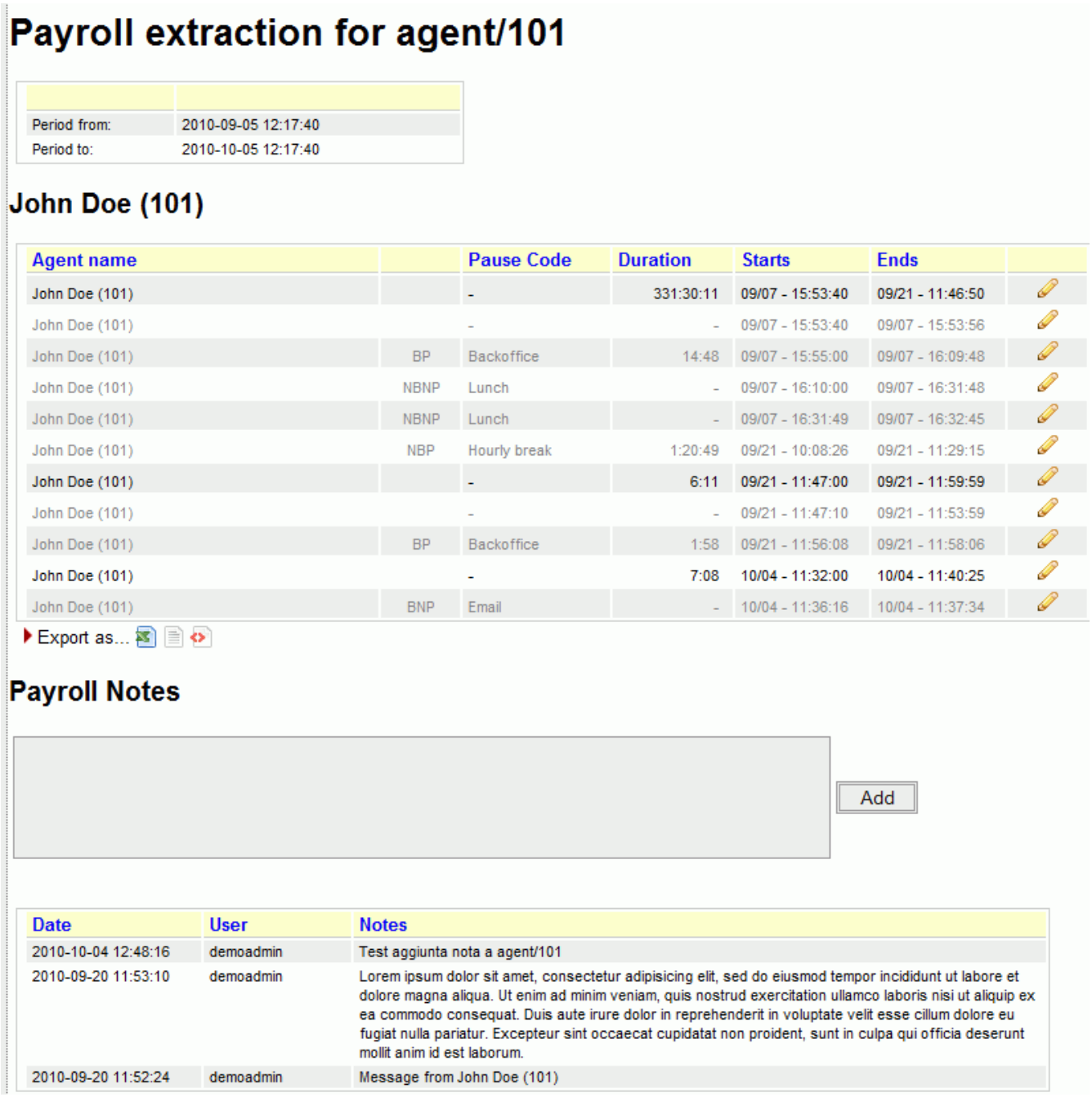
The table can be exported as Excel/CSV/XML, as all other QM tables.

Next to the agent name, if present, is an icon that displays the current Agent group. If there is a payroll note for the agent, a yellow icon is displayed by the end of the row.

The button "Export now" lets you download the "punch" data file, in the format specified on screen.

The Agent detail page

On the Agent detail page all sessions for that agent will be shown for the specified time interval. Next to the agent name, if present, is an icon that displays the current Agent group.



Pause types and codes are displayed, according to the following table:

- **BP**: Pause is Billable and Payable
- **BNP**: Pause is Billable but not payable (be careful!)

- **NBP**: Pause is not billable but Payable
- **NBNP**: Pause is neither billable nor payable

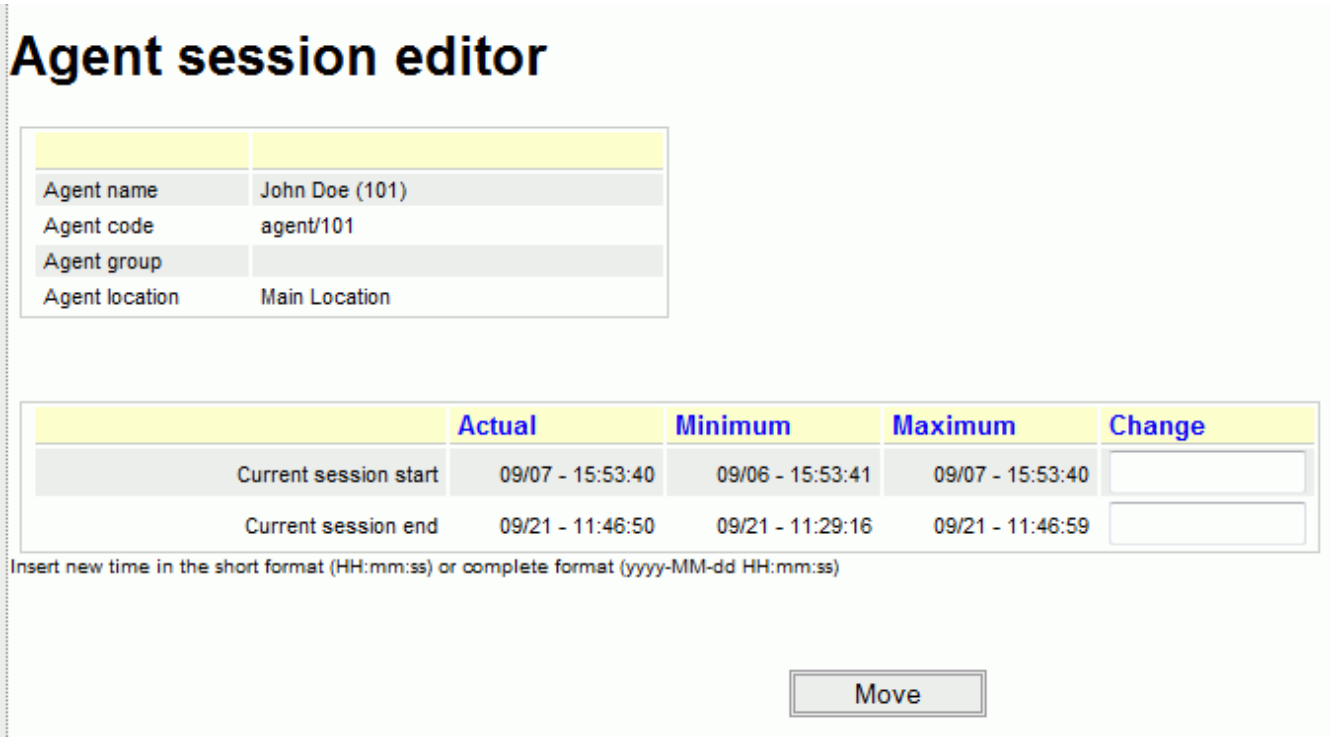
A set of payroll notes can be added by the user and read. Notes could also be deleted by a user holding the proper security key (See Appendix A: Security keys for further information).

If you have the grants to edit a session, by clicking on the icon placed next to the duration field, you will be able to edit that session.

The table can be exported as Excel/CSV/XML, as all other QM tables.

The Session editor page

The page displays the information about the selected agent session, and it searches the maximum/minimum values allowed for starting and ending the session.



It will be able to change the agent session by entering the new start and end times, either one or both at once.



If you need to make a session extend over the midnight, you must enter the full resulting date in the format like "2010-11-07 11:03:40"

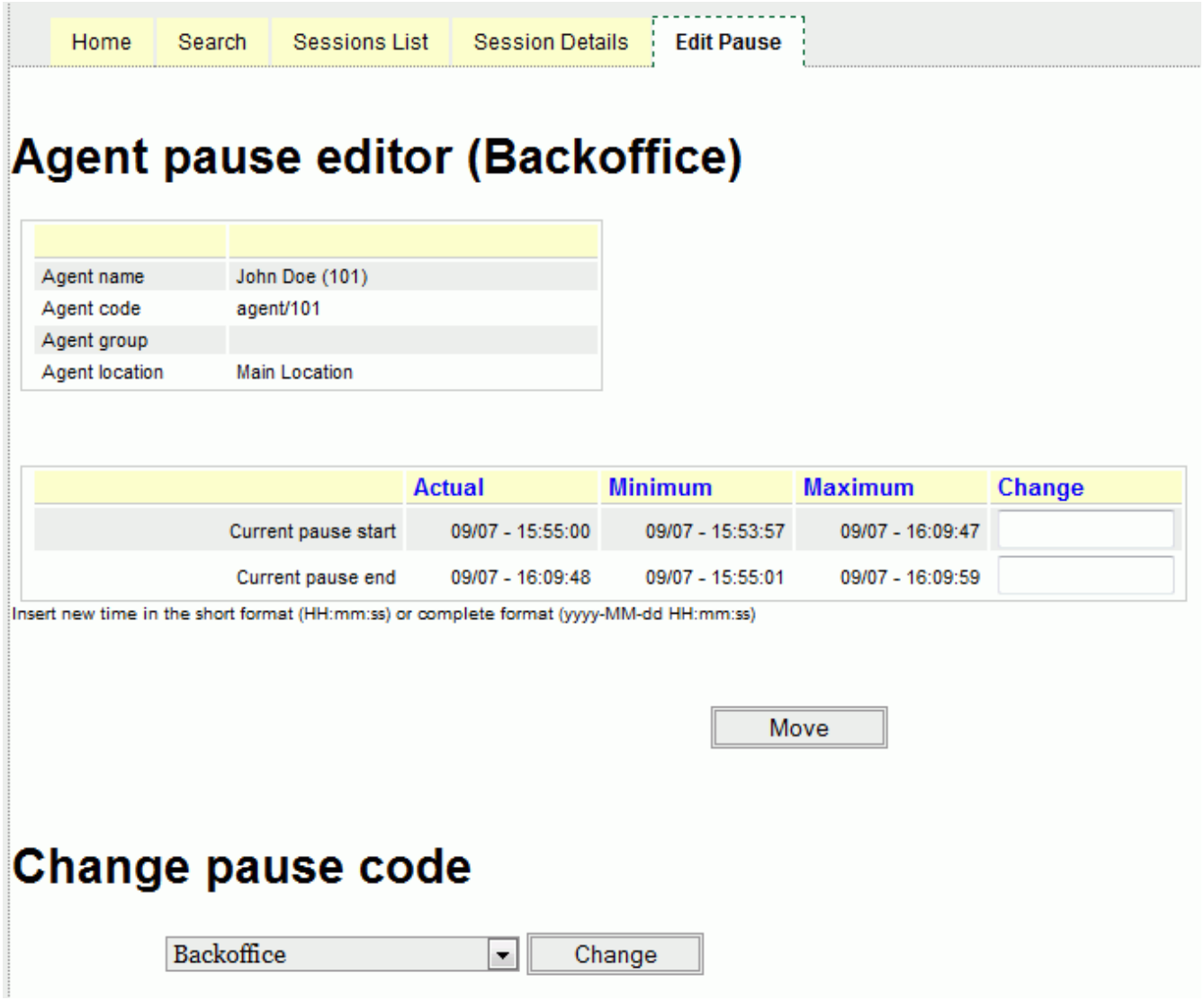
An error is raised if this makes the session longer and you do not hold the key QLOG_LNGR).

If the period is invalid, an error will be displayed. When the changes are applied, the page will reload with the new data

The pause editor page

The page displays the information about the selected agent pause, and it searches the

maximum/minimum values allowed for starting and ending the session.



This page works exactly like the Session editor, but lets you set/change the pause code as well.

An error is raised if this makes the session longer or you change a pause from non-payable to payable without holding the key QLOG_LNGR.

If the period is invalid, an error will be displayed. When the changes are applied, the page will reload with the new data



Each time the user asks for a pause modification (either the start/end time, either the pause code) the system modifies the database information in order to reorganize the pause in a *standard* format. The standard format is characterized to have a PAUSEALL, PAUSEREASON, UNPAUSEALL sequence where the PAUSEREASON is placed one second after the PAUSEALL event. This prevents problems with possible malformed pauses present in the database. All modifications are logged in the syslog pages with corresponding rollback SQL statements.

Editing the system queue_log file

When the user asks for changes on a specific agent session statement, QM will try and see if it can find that session and it is "well-formed". This means that some sessions might not be updateable though they work fine in QM.

Not well-formed sections are, for example, sections where a log-on event is not present, or overlapped with other events. This could be caused by a not corrected queue log, following some pbx unavailability or something other.

If the target session is "well-formed", QueueMetrics will try to detect whether the change causes some havoc with other calls/sessions. If it does, the change is rejected.

If the target section is well-formed but causes a growing cost for the call center, and so requires the key QLOG_LNGR, that key is checked; if not found, the change is rejected.

Well-formed agent sessions

An agent session is considered well-formed if:

- At least one line with one of the Agent logon verbs is present at the time stamp that is given as the start of the session
- A line with one of the logoff verbs is present at the time stamp that is given at the end of the session, with same partition ID as at least one starting line found

The editing log

All activity details are logged to the master QM log, where they will be available for inspection though direct SQL access.

The trace log contains:

- A description of the changes, the time and the user who requested them
- A sequence of SQL that generates the new session timing
- A sequence of SQL that is able to restore the records as they were before the change

Configuring Asterisk for QueueMetrics

QueueMetrics is designed to analyze queue_log data provided by any Asterisk installation; the following guidelines will help you to 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 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

Please note that, for Asterisk version 1.4 and greater, the ${CALLERID} variable has been replaced by ${CALLERID(num)} or ${CALLERID(all)}.

The URL will appear on a clickable link on the Agent's page.

If you set the property *realtime.agent_autoopenurl* to *true*, whenever the Agent's page is reloaded, the most recent unopened URL is launched automatically.

Listening to recorded 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 configuration properties 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. The search behaviour can be customized -see Listening to calls using Pluggable Modules (PM).

- *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 led 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*.

Creating stereo recordings

By default, Asterisk mixes the agent and the caller into a single audio channel, so a recording sounds like a normal phone call with both parties on the same track. If you plan to transcribe your calls, it is worth recording them in stereo instead, with the agent on one channel and the caller on the other.

To do so, add something like this to *extensions.conf*:

```
; record the call in stereo
exten => 551,n,MixMonitor(stereo_recording_${UNIQUEID}.raw,Db)
```

The two flags passed to `MixMonitor()` are what make this work:

- `D` — records the two legs of the call on separate channels of a stereo file, with the agent on channel 0 and the caller on channel 1, instead of mixing them into a single mono track.
- `b` — only records audio while the channel is bridged to another channel, so you don't end up with silence at the start or end of the file while the call is ringing or on hold.

Proper **diarization** (tag **TD**, see Transcriptions) — telling apart who is speaking from the audio alone — is not always available: some local models don't support it, and on SaaS services it can be significantly more expensive than plain transcription. Recording the call in stereo, with the agent and the caller on separate channels, gives you the same result without needing a diarization model at all: each channel can simply be attributed to whoever is on it. A plain mono recording can still be transcribed (tag **TR**), but without diarization or a stereo recording the transcript has no reliable way to say who said what.



Stereo files created this way are **not** meant to be played back: most browsers cannot play a two-channel raw/stereo recording as a normal phone call, and QueueMetrics does not offer playback for these files from the browser. Use them for transcription only; if you also want agents or supervisors to listen back to calls, keep a normal mono recording as well.

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,Set(DB(dynlogin/log_Agent-${EXTEN:3})=${EPOCH})
exten => _422XX,5,Hangup

; Remove Member - 423
exten => _423XX,1,Answer
exten => _423XX,2,RemoveQueueMember(my-queue,SIP/${EXTEN:3})
exten => _423XX,3,Set(ORGEPOCH=${DB(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,Set(ORGEPOCH=${DB_DELETE(dynlogin/log_Agent-${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 logon/logoff, in order to have such data available for reporting.

Defining outbound queues (campaigns)

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 added the concept of a "campaign" or "outbound queue", that is a set of calls made by different agents that are working for the same purpose. Of course there is no such thing as an outbound queue in Asterisk, so we have to run a special piece of dialplan or 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, launch them from the Agent's page or maybe use a predictive dialler for the task. QueueMetrics does not care, as long as the correct events are logged.

Placing outbound calls

If you run Asterisk 1.4 or newer and want to place outbound calls, you use an example script supplied within the *extensions_queuemetrics.conf*; it should be imported by the main Asterisk configuration.

After this, if you place a call directed to *Local/XXXYYYYYYY@queuedial*, where *XXX* is the code for the campaign and *YYYYYY...* the number to be dialled, a call will be created and logged as *Agent/ZZZ*, where *ZZZ* is the caller-id of the extension placing the call.

You may want to tweak the following supplied piece of dialplan to adapt it to your needs:

```
[queuedial]
exten => _XXX.,1,Set(QDIALER_QUEUE=q-${EXTEN:0:3})
exten => _XXX.,n,Set(QDIALER_NUMBER=${EXTEN:3})
exten => _XXX.,n,Set(QDIALER_AGENT=Agent/${CALLERID(num)})
exten => _XXX.,n,Set(QDIALER_CHANNEL=SIP/${QDIALER_NUMBER})
exten => _XXX.,n,Set(QueueName=${QDIALER_QUEUE})
exten => _XXX.,n,MixMonitor(Q-${QDIALER_QUEUE}-${UNIQUEID}.WAV|b|)
;exten => _XXX.,n,Set(CALLERID(all)="1234567890" <1234567890>) ; Uncomment and change
this if you need to set your own caller ID
```

```
exten => _XXX.,n,Goto(qm-queuedial,s,1)
```

You can/should modify the following variable definitions:

- *QDIALER_QUEUE* is taken from the first three digits. If you have only one campaign system-wide, you may want to hardcode this value so the user needs not input it.
- *QDIALER_AGENT* is the Agent code that the call will be logged under. The simplest approach is just to use the extension's caller-id, under the hypothesis that Agent/123 works at SIP/123. You may also look up under Asterisk who is the agent working at a given extension - an example is given in the *[queuedial-loggedon]* context in the same file.
- *QDIALER_CHANNEL* is the channel that you have to dial to call out. Will likely be something like *Zap/g1/${QDIALER_NUMBER}* or *SIP/myprovider/${QDIALER_NUMBER}* in a production system.
- You can comment out the MixMonitor line if you don't need call recordings.

Please note that:

- 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, while the Agent field is the caller.
- It is possible to do live listening of outgoing calls (see Enabling ACD call attempts recording on Asterisk 1.0 and 1.2).
- It's possible to specify your caller ID uncommenting the line where the Set function is called and, obviously, changing the caller ID information to properly set it as required. The same modification is needed for the extensions 28 definition present in the same file.

Placing outbound calls through the AGI script

This section applies only if you run a version of Asterisk 1.0 or 1.2; for 1.4 or newer, please use the dialplan logic supplied in the file *extensions_queuemetrics.conf*.

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 parameters 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
- It is possible to do live listening of outgoing calls (see Enabling ACD call attempts recording on Asterisk 1.0 and 1.2).

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.

QueueMetrics starts to analyze AGENTATTEMPT verbs when the configuration key *default.ignoreRingNoAnswer* is set to true.

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. In this case QueueMetrics reports in the realtime page the last agent that had not picked up the phone when ringing. QueueMetrics starts to analyze RINGNOANSWER verbs when the configuration key *default.ignoreRingNoAnswer* is set to false.

It is possible to have the AGENTATTEMPT information in a not patched Asterisk 1.4 with some modifications in the dialplan. This option is limited to people not using the hotdesking feature. For more information on that, please refer to the QueueMetrics advanced configuration manual.

Listening to live calls: Unattended Call Monitoring

In order to implement this feature, QueueMetrics follows the following steps:

- It will try to dial the channel defined in the property *callfile.monitoring.channel* by passing the local extension. This should make your local phone ring.
- Once the call is picked-up, it will try to dial *11@queuemetrics* (if the call is inbound) or *14@queuemetrics* (if the call is outbound) in order to start the ChanSpy() monitoring and will pass along all required variables to match the requested call.

To enable unattended audio monitoring for inbound calls, you'll have to edit the Asterisk dial-plan in order to include the *[queuemetrics]* context.

- Make sure that the *queuemetrics* context exists and that the extensions 10, 11 and 14 are defined for it. See Appendix B: The [queuemetrics context]
- Make sure that the channel defined in the property *callfile.monitoring.channel* is set to *Local/$EM@from-internal/n* (in this example, your telephone would be known by Asterisk as something like *105@from-internal*).
- 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 a (now preferred) 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
- **Important**: make sure that each agent will have their local extension set in QueueMetrics; usually entering "-" will be enough. If this is not set, the icon will not appear.
- Now, when you click on the icon, a callfile will be generated and call snooping will start.

To enable unattended call monitoring for outgoing calls as well, you'll have to set the piece of dial-plan referenced by the *callfile.outmonitoring...* properties.

Outgoing calls placed through *queueDial.agi* will usually be listened to by attaching to the local

SIP/XXX or *Local/XXX* channel of the calling agent and not to the standard *Agent/XXX* channel used for inbound, so a different piece of dial-plan will be used. Note that in order for QueueMetrics to reference the outgoing calls, you must tell it that queue direction is Outgoing.

See also Appendix B: The [queuemetrics context] for an example of implementing Asterisk code for inbound and outbound call monitoring.

It is possible to use different PMs to handle different live audio - see Listening to calls using Pluggable Modules (PM).



if you pass an empty variable to the ChanSpy() command, it will let the user listen to any channels on the system. *This may be a major security issue*. So if you edit the supplied dialplan to match your configuration, make sure that you add a check in case a computed channel to listen on might be empty!

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.

It is possible to set input validation for both the agent code and the extension. As an example, by setting the key *realtime.agentRegexp=1\|d\|d* you are defining that the agent code must start with a

one and cannot be more than 3 digits, while by setting the key *realtime.extensionRegexp=\d\d\d0* you define that the agent extension must be four digits and must end with a zero.

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.

In order to enable this, we first tell QueueMetrics to use the XML-RPC Pluggable Modules for both call listening and streaming:

```
audio.server=it.loway.app.queuemetrics.callListen.listeners.ClassicXmlRpcRecordings
audio.liveserver=it.loway.app.queuemetrics.callListen.RTlisteners.ClassicXmlRpcListene
rRT
```

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_audio_server.php
```

The server must implement three 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 (displayed as returned)
  - $FILE_ENCODING : encoding of the audio file (eg mp3)
  - $FILE_DURATION : duration of the audio file

  In case multiple values are to be returned (because e.g. the call is split into multiple recordings) then the following format must be used:

- $FILE_FOUND : true
- $FILE_LISTEN_URL : "MULTI:http://url1 http://url2 http://url3"
- $FILE_LENGTH : "100k 50k 120k"
- $FILE_ENCODING : "mp3 mp3 mp3"
- $FILE_DURATION : "1:00 0:30 1:20"

As you can see, the listen URL starts with the string "MULTI:" and has multiple values separated with space. The other parameters also hold multiple values separated by space.

- **QMAudio.listenOngoingCall**
  This function is used to query for an ongoing inbound 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.
- **QMAudio.listenOngoingCallOutgoing**
  This function is used to query for an ongoing outgoing call. If found, QM will launch a new popup to open the player which URL is returned. The parameters are the same as for *QMAudio.listenOngoingCall*.

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 just terminated (see Updating past calls).

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 set shortly after the end of a call (see Updating past calls).
- 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 full "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.

Keeping the UNIQUEID of the call when setting status code

One of our clients has successfully implemented Call Outcomes by using AEL.

In the Hangup-Extension, use:

```
if("${MEMBERINTERFACE}" != "" && "SIP/${CALLERID(ani)}" != "${MEMBERINTERFACE}" ) {
   // to be able to record QueueMetrics call outcome later
   Set(GLOBAL(queue_last_call_${MEMBERINTERFACE:4})=${UNIQUEID});
}
```

The outcome is recorded like this:

```
_#XX = > {
        // ${user_name} contains the extension number of the agent
        Answer();
        if ("${queue_last_call_${user_name}}" != "") {

QueueLog(NONE,${queue_last_call_${user_name}},SIP/${user_name},CALLSTATUS,${EXTEN:1});
              Set(GLOBAL(queue_last_call_${user_name})=);
              Playback(beep);
        } else {
              Playback(beeperr);
        }
        Hangup();
     }
```

Enabling call features

A call feature code is a code to be input by a user telling the sub-status of a call, be it inbound or outbound. This feature code is a string and may be input either when the call is ongoing or after a short while from its end. Several call feature codes can be associated to a single call; QueueMetrics will count all the associated feature codes. A feature code may have an optional free text associated with it.

The queue_log entry looks like the following one:

```
1234|1231.1|NONE|Agent/1234|INFO|FTR|36
```

This will set the feature code to "36" for the call which Call-ID is "1231.1" it may be an open call or it may be just terminated.

A second version of the verb is available:

```
1234|1231.1|NONE|Agent/1234|INFO|FTR|41|Hello there
```

This has exactly the same meaning but the feature code contains a free text associated with it.

Feature codes can be removed from a call by mean of a trace added in the queue log:

```
1234|1231.1|NONE|Agent/1234|INFO|NOFTR|36
```

This will remove the feature code "36" to the call with Call-ID is "1231.1". If no feature code "36" have been already associated with the call, QueueMetrics simply discards the trace.

The following rules apply:

- Features should be entered while the call is ongoing or within a few minutes from its closure

(see Updating past calls). The loopback period is the same as for setting the outcome.

- Features have a code and an optional textual value. The text should be printable ASCII7 and cannot contain pipe or special, non-printable characters. Though we enforce no limit on this, we suggest using no more than 40 characters.
- If multiple features with the same code are applied for the same call, the last one wins (sets the text of it)
- In order to remove a feature, a special verb NOFTR is passed
- If successive lines set the feature after a NOFTR, the feature is set and so on.

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 previous codes.
- The PAUSEREASON must be passed within 30 minutes from the end of a pause; otherwise it will be silently discarded (see Updating past calls).
- 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 full "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.



Since Asterisk 1.6, it is possible to pass a pause reason code to the native Pause application. QueueMetrics will handle this correctly, and allows mixing the two methods as you best see fit.

Tracking required skills

Skills can be specified at any time before of after the call enters the queue but the suggested way is to have them immediately after the ENTERQUEUE verb as in:

```
1167335958|1167335958.530054|cpc|NONE|ENTERQUEUE||01909723335
1167335958|1167335958.530054|cpc|NONE|INFO|REQSKILL|DE|90
1167335958|1167335958.530054|cpc|NONE|INFO|REQSKILL|NET|70
1167335958|1167335958.530054|cpc|Agent/101|CONNECT|41|Agent/101
1167335958|1167335958.530054|cpc|Agent/101|COMPLETECALLER|41|23
```

In the example above, the call is queued on a queue called *cpc* with required skills *DE>=90* and *NET>70*, so it could be a request for someone servicing internet support in German. Basically you just add an event for each required skill, and the level at which it is required. We do not need this information on connected/failed calls, as we assume that the routing is always correct.

The following rules apply:

- Rules are read as a label (the rule) and a number. If the number is invalid, it is read as zero and no exception is raised
- The label can be any valid ASCII7 string, punctuation and spaces
- For a skill to be added, it must have a non-empty label and a skill level > 0
- If a skill is repeated multiple times, the last value wins. In the call details, the full sequence of events is displayed
- It is not possible to remove a skill from a call

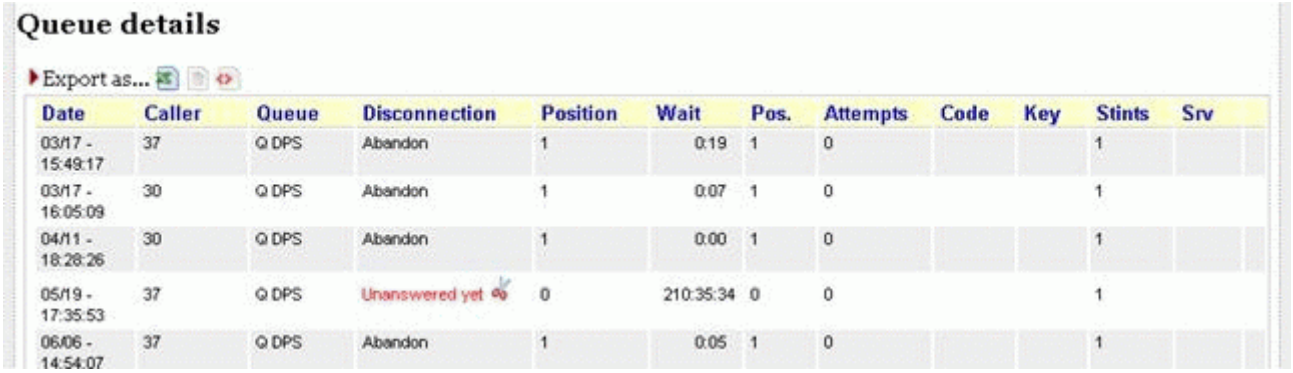
Closing ongoing calls

It sometimes happens that Asterisk will not log the call termination records for a call; as QM is based on the logged events, a call missing the call closure log will linger on forever in the realtime screen (or at least the maximum time allowed by the ...) and will appear as Ongoing or Not answered yet in the historical reports.

It is possible to manually close a call from either the historical reports or the real-time screen. In order for this to work:

- You must be running with MySQL storage or clustered storage
- Your user must own key CLOSECALLS

When this is done, open calls on the reports will show a red scissor icon:



And the same will happen for the real-time screen:



By clicking on that icon, a popup will appear that will ask for the length the call should be closed to. This length refers to the wait duration if the call is not answered and the conversation time if the call is answered. It is possible to change that from the default 5 seconds by setting a configuration property.

If the call has already been closed in the meantime, or you're doing this operation twice, QM will report that the call has already been closed.



if you do this on calls that are still ongoing, you will risk having duplicate data on the report. So don't use this feature unless you know what you are doing. The required security key must be manually assigned only to trusted users.

Tracking DNIS and IVR information



In order to track IVR information, it would be advisable to follow the newer logging format defined in Implementing IVR tracking.

In order to keep track of DNIS and IVR information that relates to each call, you have to write special records on the *queue_log* file that QueueMetrics parses.

This is very easy to do, e.g. imagine you have a piece of dialplan where you are going to call queue *q-sample* and you have the DNIS code in the *MYDNIS* dialplan variable, and the sequence of keys pressed as *MYIVR*:

```
exten => s,n,........
exten => s,n,QueueLog(q-sample,${UNIQUEID},NONE,INFO,DID|${MYDNIS})
exten => s,n,QueueLog(q-sample,${UNIQUEID},NONE,INFO,IVR|${MYIVR})
exten => s,n,Queue(q-sample|nt|||60)
exten => s,n,........
```

There is no predefined format for DNIS and IVR information; QueueMetrics just handles it as free-form text strings. It can be optionally decoded by creating values in the IVR and DNIS configuration pages.

You can output only one record, or both, or none, depending on what you need.

Adding TAG information to calls

Some call centers require to report call set at the list level below campaign. This is frequently used for outbound campaings.

QueueMetrics searches TAG information on the queue log file. To enable Asterisk to place a TAG signature in the queue log a dialplan modification is needed. Below is an example on how this could be done, assuming you have the TAG code in the *MYTAG* dialplan variable:

```
exten => s,n,........
exten => s,n,QueueLog(q-sample,${UNIQUEID},NONE,INFO,TAG|${MYTAG})
exten => s,n,Queue(q-sample|nt|||60)
exten => s,n,........
```

You can assign only one TAG for each call. If more than one TAG are assigned to a single call, QueueMetrics will discard all TAGs except the latest found in the queue log file.

Enabling Hotdesking in the agent page

Since the demise of AgentCallBackLogin, it has been hard to do "hotdesking" in Asterisk - that is, having agents that work on queues because of their competences and not because they are sitting at a given extension.

In QueueMetrics, hotdesking is very easy to implement and it has no downsides, because:

- it is completely transparent to Asterisk
- you can emulate the single-sign-on behavior of AgentCallBack and still have the flexibility of adding/removing members as needed on a queue by queue basis.
- call recordings, agent monitoring and all other functionalities are unaffected

Requirements:

- MySQL storage model
- Asterisk 1.4 or 1.6

How it works

Set the following properties within the configuration properties, as follows:

```
default.queue_log_file=sql:P001         <-- change as needed
```

```
callfile.dir=tcp:admin:amp111@127.0.0.1 <-- change as needed
default.rewriteLocalChannels=true
callfile.agentlogin.enabled=false
callfile.agentlogoff.enabled=false
default.hotdesking=86400
```

Make sure that *extensions_queuemetrics.conf* is loaded in the Asterisk dialplan (you need to use the extensions_queuemetrics file that comes with QueueMetrics).

This setup means that we access the queue_log file through the database, connect to Asterisk over AMI to send commands, rewrite agent codes, do not use Agentcallback-style agents and enable hotdesking.

Now we use a piece of dialplan like this one when we associate an agent to a queue:

Imagine we have AGENTCODE set to 200 (the agent's login code) and AGENT_EXT set to 123 (this is the SIP extension code):

```
....
exten => 35,3,QueueLog(NONE,${UNIQUEID},Agent/${AGENTCODE},HOTDESK,SIP/${AGENT_EXT})
exten => 35,4,AddQueueMember(myqueue,SIP/${AGENT_EXT})
....
```

This logs on Agent/200 to queue "myqueue", tracking him as SIP/123. Note that from the point of view of Asterisk, we only see that extension 123 is made a member of the queue.

When you logoff, pause, unpause agents, you always work at the SIP level (the actual extension that is linked to the queue) so there is no need to change anything.

If you use the QueueMetrics Agent's page, you can do logon/logoffs/pauses from the buttons by the top of the page; this lets you add an agent to all queues at once, like you used to do with AgentCallBackLogins, and still retain the flexibility to change that at runtime.

Example hotdesking configuration

In the following sections, we summarize the changes that have to be made to an existing system to enable hotdesking.

Changes to configuration properties

Add/change the *default.hotdesking* property to 86400. This property enables hotdesking and lets the parser "look back" up to 1 day (change as needed).

```
default.hotdesking=86400
```

Add/change the sections below:

```
callfile.agentpause_ht.enabled=true
callfile.agentpause_ht.channel=Local/32@queuemetrics/n
callfile.agentpause_ht.extension=10
callfile.agentpause_ht.context=queuemetrics

callfile.agentunpause_ht.enabled=true
callfile.agentunpause_ht.channel=Local/33@queuemetrics/n
callfile.agentunpause_ht.extension=10
callfile.agentunpause_ht.context=queuemetrics

callfile.agentaddmember_ht.enabled=true
callfile.agentaddmember_ht.channel=Local/35@queuemetrics/n
callfile.agentaddmember_ht.extension=10
callfile.agentaddmember_ht.context=queuemetrics

callfile.agentremovemember_ht.enabled=true
callfile.agentremovemember_ht.channel=Local/37@queuemetrics/n
callfile.agentremovemember_ht.extension=10
callfile.agentremovemember_ht.context=queuemetrics
```

This code specifies the Asterisk extensions that QueueMetrics will call for each button present in the agent live page when hotdesking is enabled.

Change the *realtime.agent_button_x.channel* key to the value *Local/[EM]@from-internal*

This last option is needed only if you use custom agents buttons to dial out extensions and should be repeated for each dial-enabled button. In the code below, a valid example for the button 4 is reported:

```
realtime.agent_button_4.enabled=true
realtime.agent_button_4.caption=Secretary
realtime.agent_button_4.url=
realtime.agent_button_4.channel=Local/[EM]@from-internal
realtime.agent_button_4.ext=200@queuedial
```



if you use a channel like Local/123@from-internal as the hotdesking extension, remember to turn off local channel rewriting first, or it will not work.

Changes to extensions_queuemetrics.conf

Here should be defined the Asterisk extensions used by QueueMetrics to perform actions triggered from the agent live page.

Add to this file the code reported below:

```
; extension 32: agent pause with hotdesking (with pause code)
exten => 32,1,Answer
```

```
exten => 32,2,NoOp( "QM: Pausing Agent/${AGENTCODE} at extension
SIP/${QM_AGENT_LOGEXT} \
    with pause reason '${PAUSEREASON}' made by '${QM_LOGIN}' " )
exten => 32,3,PauseQueueMember(,SIP/${QM_AGENT_LOGEXT})
exten => 32,4,System( echo
"${EPOCH}|${UNIQUEID}|NONE|Agent/${AGENTCODE}|PAUSEREASON|${PAUSEREASON}" \
    >> /var/log/asterisk/queue_log )
exten => 32,5,Hangup

; extension 33: agent unpause with hotdesking
exten => 33,1,Answer
exten => 33,2,NoOp( "QM: Unpausing Agent/${AGENTCODE} at extension
SIP/${QM_AGENT_LOGEXT} \
    made by '${QM_LOGIN}' " )
exten => 33,3,UnpauseQueueMember(,SIP/${QM_AGENT_LOGEXT})
exten => 33,4,Hangup

; extension 35: agent addqueuemember with hotdesking (for asterisk v1.4+)
exten => 35,1,Answer
exten => 35,2,NoOp( "QM: AddQueueMember (asterisk v1.4+) Agent/${AGENTCODE} at
extension \
    SIP/${QM_AGENT_LOGEXT} on queue ${QUEUENAME} made by '${QM_LOGIN}' with
prioritylabel \
    '${QM_AGENT_PRIOLBL}' and prioritynum '${QM_AGENT_PRIONUM}'" )
exten => 35,3,Macro(queuelog,${EPOCH},${UNIQUEID},NONE,Agent/${AGENTCODE},\
    HOTDESK,SIP/${QM_AGENT_LOGEXT})
exten => 35,4,AddQueueMember(${QUEUENAME},SIP/${QM_AGENT_LOGEXT})
exten => 35,5,Hangup

; extension 37: agent removequeuemember with hotdesking (for asterisk v1.4+)
exten => 37,1,Answer
exten => 37,2,NoOp( "QM: RemoveQueueMember (asterisk v1.4+) Agent/${AGENTCODE} at
extension \
    SIP/${QM_AGENT_LOGEXT} on queue ${QUEUENAME} made by '${QM_LOGIN}'" )
exten => 37,3,RemoveQueueMember(${QUEUENAME},SIP/${QM_AGENT_LOGEXT})
exten => 37,4,Hangup
```

Please note that the *extensions_queuemetrics.conf* file that ships with QueueMetrics already has these changes embedded.

In order to have the hotdesking working a complete QueueMetrics restart and Asterisk reload should be performed.

Running Asterisk 1.8 with QueueMetrics

QueueMetrics is compatible with Asterisk 1.8 but you need to properly set it. The first requirement is related to a strange behavior found in Asterisk 1.8.0 and 1.8.1 that prevents Asterisk to properly log all queue activity until a reload command is issued from the CLI. To fix this problem we had to change the code in the logger.c file found in the main subfolder present in the asterisk sources,

near the line 396, in order to have something similar to what is listed below:

```
        if (qlog)
                fclose(qlog);

        {
                char tmp[4096];
                snprintf(tmp, sizeof(tmp), "%s/%s", ast_config_AST_LOG_DIR,
queue_log_name);
                qlog = fopen(tmp, "a");
        }
```

Then we had to rebuild asterisk and to reinstall it.

The next step is to replace the extensions_queuemetrics.conf file with the specific version for Asterisk 1.8. To do this, you need to copy the extensions_queuemetrics_18.conf replacing the one present into the asterisk configuration folder then reload the dialplan from the CLI. The extensions_queuemetrics_18.conf is targeted to Asterisk 1.8 with hotdesking enabled.



This applies ONLY to some earlier versions of Asterisk 1.8

Handling Agents priorities on queues

It is possible to define priorities when logging agents in a specific queue. The priority associated to each agent is dependent on how the agent was configured in the queue (main, spill or wrap). This feature populates two new channel variables ${QM_AGENT_PRIOLBL} and ${QM_AGENT_PRIONUM} when AddQueueMember is called by extensions 25 and 35 in the queuemetrics context.

- QM_AGENT_PRIOLBL is set to "U" when the queue is not assigned to the agent, "M" when the queue is a normal working queue for the agent (Main), "W" when the agent is set as Wrap for the queue, "S" when the agent is set as Spill for the queue.
- QM_AGENT_PRIONUM is set to 0 when the queue is not assigned to the agent or the queue is a normal working queue for the agent; 1 when the agent is set as Wrap for the queue, 2 when the agent is set as Spill for the queue.

Using known numbers

In QueueMetrics, it is possible to keep track of a set of "known numbers" through a specific editor as described in Configuring known numbers (VIP callers / blacklists)

For each number, we track:

- An action to be done
- A name and description
- An optional expiry date

- The number of times the rule was hit, and the last time it was hit
- An optional "agent affinity" attribute

A webservice - built to be easy to query from the Asterisk dialplan or through JSON - lets the user query whether the number exists and what to do with it.

Possible actions include:

- *Blacklist*: the number is to be hung up, or moved to a low priority queue
- *VIP*: the number is to be moved to a VIP queue, or its queue priority is to be made higher
- *Agent affinity*: if a number has "affinity" with some agent, then the agent should be tried first (by using a personal queue, or reaching them through a direct channel and "faking" agent records so that they appear as if the call was processed on a queue)
- *Caller name* / CRM ID: the caller-id is rewritten so that a proper name can be associated with the call.

Those actions are to be implemented at the Asterisk dialplan level.

The webservice is meant to be easy to interoperate with a standard Asterisk CURL call.

```
exten =>
s,1,Set(res=${CURL(http://my.qm:8080/queuemetrics/numberLookup.do?number=$(CALLERID)&m
ode=action&user=user:pass)})
```

The page returns a response as text/plain that will be embedded in the dialplan variable *res*. The response is based on the following table:

Table 3. Possible API modes

Mode	Sample response	Notes
action	UNKNOWN BLACKLIST VIP <blank>	The response UNKNOWN means that the number is not listed; the response blank means that it is listed but no action is required. The number of hits and last hit counters are updated.
affinity	Agent/106 '' <blank>	The number has a known affinity for Agent/106. If no known affinity, returns blank.
name	John Doe <blank>	An URL-encoded name for this number, or blank if no name set / unknown number. The name will be stripped of non-Ascii7 characters.

Mode	Sample response	Notes
json	{ 'action\': 'vip\', 'name\': 'John Doe\', 'affinity\': 'Agent/101\' }	In this mode, a JSON object is returned with the information shown. This way you can query all the parameters at once, or you can use a JSON library for querying. The response is returned as application/json.
hash	action=vip&name=John Doe&affinity=	In this mode, the returned string is encoded as an Asterisk hash (see below). This way, all the results can be read at once.

Notes

- It is important to make sure that the HTTP error case is handled at the dialplan level, in case QM was unavailable at the time of the call.
- It is also important to set a quick connection timeout through CURLOPT(<option>) to make sure that if QM does not answer quickly, the default case is handled.

The trivial format used in the response makes it possible to use the service using the command-line curl, or by using a trivial scripting language.

The "user" parameter is mandatory and requires a valid QueueMetrics username and password, separated by a ":". This is needed because the API will be publicly exposed. The user must hold the security key *PBXAPI*

In order to avoid security risks, the user should be custom-made and assigned to a class that only contains the key PBXAPI. This way the user will be able to log on to QM but will have no actual grants but logging off.

Asterisk hashes

When querying an Asterisk hash, the result is built as a string so that you can do:

```
Set(HASH(numinfo)=${CURL(myurl)})
```

to load the results from QM, and then parameter e.g. *action* can be read as:

```
${HASH(numinfo,action)}
```

In order to avoid issues when returning data from QueueMetrics in the HASH format:

- Data is returned as UTF8
- All special characters in the returned strings will be replaced by an underscore. Those being:

- non printable (CR, LF, etc.)
- having a possible meaning in the Asterisk dialplan or in the querystring-like hash format being non ASCII7

When querying each response separately, the response is instead UTF8.

Configuring the AMI connection

QueueMetrics bases its reports on data generated from the Asterisk queue_log file; still, it sometimes needs to send commands to Asterisk in order to perform some actions, e.g. log on agents, or listen to live calls.

In order to perform such commands, two things are required:

- A working AMI connection should be present
- The *extensions_queuemetrics.conf* file should be included in the PBX's dialplan

For historical reasons, the default way QueueMetrics used to send commands was to generate Asterisk call files; now this method is obsolete and the correct one is to set-up an AMI connection.

In order to set up an AMI connection, you have to set the following property like e.g.:

```
callfile.dir=tcp:admin:amp111@127.0.0.1
```

The AMI URL is in the following format: *tcp:username:password@server:port*

- username: This is the AMI username
- password: This is the chosen "secret"
- server: This is the IP address of the server, or 127.0.0.1 if the same server.
- port: This part is optional; if not present will default to 5038.

All three fields are mandatory. The password is sent over a clear-text TCP connection, so make sure to protect it using e.g. a VPN tunnel if it is to traverse public networks.



Username and password should be made only of letters and digits; no other character should be used.

The configuration above should be matched by the configuration in Asterisk's own *manager.conf* file, that should look like the following one:

```
[general]
 enabled = yes
 port = 5038
 bindaddr = 0.0.0.0
 webenabled = no
```

```
[admin]
 secret = amp111
 deny = 0.0.0.0/0.0.0.0
 permit = 127.0.0.1/255.255.255.0
 read = system,call,log,verbose,command,agent,user,originate
 write = system,call,log,verbose,command,agent,user,originate
```

In order to make testing easier, QueueMetrics includes a test tool that checks whether the current connection is working or not; see Checking an Asterisk Manager connection for details.

Listening to encrypted recordings

QueueMetrics allows to listen to recordings that are stored in an encrypted format. This works by invoking a custom-supplied filter that will decrypt the recording on-the-fly before QM streams it back to the user.



This is possible only for recordings that are read from disk and streamed by QM; it does not work for recordings that are streamed by a third-party player (e.g. Oreka), which will usually implement its own encryption scheme.

What is a filter

In order to decrypt a call, QueueMetrics uses a filter, i.e. a program (usually a script) that, given the filename that it needs to decrypt, will output the decrypted file to STDOUT. This way the decrypted file is never saved on disk.



Encrypting and decrypting recordings on-the-fly can impose a severe load on your QueueMetrics server, as encryption is usually CPU-intensive.

A sample filter may look like the following script:

```
PASSW=myPassword
echo $PASSW | gpg --passphrase-fd 0 --batch --decrypt $1
```

As the filter is not dependent on any specific encryption technology (public key, symmetric keys, etc) QueueMetrics is able to adapt to whatever technology suits you best.

Please note that the called script does not receive a password - it must be able to run the decryption internally. Most encryption technologies have the concept of "secure password stores", so that you can avoid storing the password in a plain-text format.

Setting up a filter

In order for QM to decrypt a file, it must match two conditions:

- It must end in .crypt, as appended to the natural extension of the file (e.g. the encrypted version of a file named *'audio.mp3*' must be called *'audio.mp3.crypt*')

- The configuration property *'audio.decrypt'* must point to the decryption filter, as in the example below.

The script to be run must be readable and executable by the QueueMetrics process, as in:

```
audio.decrypt=/encryptionTools/decryptGPG.sh
```

When an encrypted file is found by QueueMetrics, it is displayed with a "lock" icon. By clicking on it, the file is decrypted on the server and streamed back in an unencrypted format.

If a file is not encrypted, QueueMetrics will stream it back without attempting any decryption.



For security reasons, QueueMetrics can only call an external filter if the JVM property `-Dqm.execCmds=1` is set. Without it, the decryption script will not be executed. The property can be added to the JVM options (`QM_JAVA_OPTS`) - see Further configuration.

Encrypting calls

As Asterisk does not currently offer any facility for storing encrypted recordings, audio files must be encrypted on a periodical basis.

- Every so often, a process runs and checks for unencrypted recordings in the audio destination directories
- Every file found is first encrypted, and if the encrypted file was actually created, then its unencrypted version is removed.

We offer a sample encryption routine in the files *'encryptAllGPG.sh'* and *'encryptGPG.sh'* that can be used as an example to deploy your own script.



The sample encryption and decryption scripts are available under the *'WEB-INF/mysql-utils/audio-encryption'* folder in QM. They are meant as a reference blueprint only and may not be suitable for the required Corporate security standards.

Tracking IVR data

IVR information is tracked natively within QueueMetrics.

IVR information is different from queue information, as:

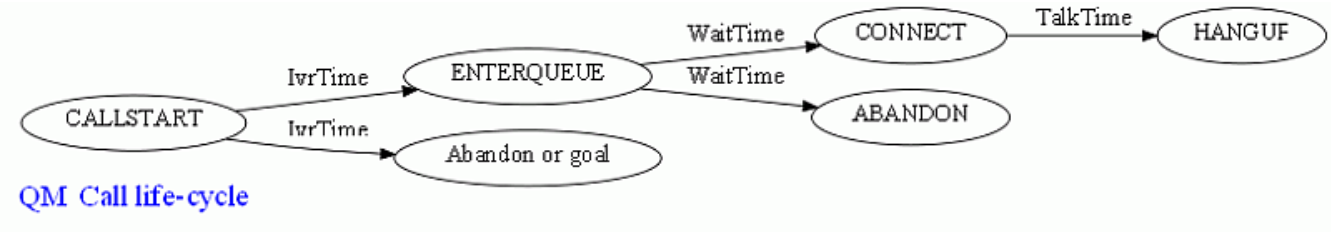
- IVR data is typically tracked before a call hits a queue, and bears no queue information. So IVR calls might be ancestors of any call in the system.
- If you happen to know from the start of the call that a particular call belongs to a more specific **area**, you may want to track this immediately so that this IVR call will not appear on "unassigned" calls (see below for more information).

- IVR data is made up of multiple **IVR menus**, each of which may have a **selection**, that is a digit pressed by the caller in order to progress forward.
- An **IVR Path** is a unique sequence of IVR menus that lead to a destination
- IVRs may define **goals**, that is activities that don't lead the caller directly to a queue but are tracked as they satisfy, in a fully automated manner, the needs of a caller. For example, when a caller reaches the TTS menu reading back their current account amount in your banking IVR, that is usually a goal you want to track.
- If a call is still in the IVR phase at the end of the analysis, without having reached any goal or having hung up, it is considered hung up just after the last known event.

IVR tracking requires changes to the current dial-plan of Asterisk in order to track the required information. Such changes - detailed in section Implementing IVR tracking - are not complex to implement, and are implemented natively in the QueueMetrics module of FreePBX. So if you use a common Asterisk distribution, chances are you already have IVR tracking available.

A call's life-cycle

When a call is tracked within QM, it may traverse the following tree:



QM Call life-cycle

As you can see, QM defines three time periods about a call:

- the IVR Time, that is between when a call is first tracked and when it hits a queue. For calls that do not have any events before hitting a queue, it is always set to zero.
- the Wait Time, that is how long a call has been waiting on a queue before being answered or hung up
- the Talk Time, that is the length of the conversation between the agent and the caller.

IVRs and QueueMetrics

QueueMetrics handles IVRs when running reports by filtering all calls by the set of queues specified in the report. As IVRs have no queues, all IVRs are processed for the time period requested. IVRs that end up on queues different from the ones being reported on are reported as "lost".

IVR activity usually takes place before the call is queued. Filters on IVRs, DNIS and caller-ids are correctly applied to the analysis if specified (so you can drill down an IVR analysis on a specific selection path or on a caller / called number).

What you get out of an IVR analysis:

- Tracing IVR paths: the report IV01 - IVR Traversals. This way you can see which paths were traversed, how many calls went through each path and what happened to them. This is very

useful as it is the key to understanding attrition - people hanging up without having reached a goal.

- Tracing IVR timings: the report IV02 - IVR Timing. By understanding the timing statistics of each IVR menu, you can improve the user experience by making often-accessed items higher on the menus and on top of the tree. This improves user satisfaction and saves circuit usage.
- Tracing IVR goals: the report IV03 - IVR Goals. For each goal, what path traversal tree was followed, how much it took for it to be traversed as average / min / max times.
- IVR call details: all calls that were not queued are available in the IVR call details, under the codes OD04 - IVR details (paged) and OD05 - IVR details (full list).

The detailed description of each data block is visible in the relevant manual section.

Using IVR areas

Considering an IVR call as a possible ancestor of any queued call on the system is often overkill. Very often you are able to tell - for example, based on the DNIS or the caller-id - the "area" that the call pertains to. If this does not happen, calls tracked at the IVR level do not belong to a queue but are possible ancestors of any queue – and therefore appear in the IVR and attrition reports for any queue.

Though this requirement is formally correct (as the main reason for having an IVR offered is routing the caller into the correct queue) it often happens that at some point during the call life-cycle, before the call is connected to a queue, you can determine an “area” of interest that may or may not be a specific queue.

For example, if your call center services multiple clients, you will likely have a separate DNIS for each. When that DNIS is called, maybe you do not yet know if the call will be connected to `client7-sales` or `client7-support`, but you are sure that it will be for `client7`, so it will be inappropriate to show that call as a general untagged ancestor to any call queued on the system.

Tagging calls by area in the QueueMetrics security system

The QueueMetrics security system is built so that only information pertaining to the queues a user has express permission on are accessible for reporting or monitoring.

As the unit of access is the queue, we had to find an access token that acts "as-if" it was a proper queue. We call this "area" and we will log this to the queue_log in the general queue field.

This makes it possible to decide on which areas the reports should run by adding the areas to the allowed queues for a specific report, as you would generally do when creating a composite queue.

For instance, to implement the example above, you might be reporting on a composite queue defined as `ivr-client7|client7-sales|client7-support`.

A call starts its life-cycle by belonging to area NONE, then starts belonging to an area as soon as one is specified, and starts belonging to a queue as soon as it hits one. Neither behavior is reversible.

The following visibility rules apply:

	IVR call - area = NONE (c.1)	IVR call - area set (c.2)	IVR call - queue reached (c.3)
Any report for any queue	Visible	-	-
Reports for destination queue(s) only	Visible	Not Visible	Visible
Reports for area(s) and destination queue(s)	Visible	Visible	Visible

The initial QueueMetrics IVR tracking only implemented columns #1 and #3 of the table above. By adding the "areas" as in column #2:

- Existing systems are backwards-compatible; the behavior does not change unless you specify an area for a call;
- You are free - if you want - to specify an area immediately, so that the behavior specified in column #1 does never happen on your system.



We discourage using `NONE` as an area - if you have no information, just call it `ivr`.

Using areas

The following general rules apply to areas:

- A call is displayed in QueueMetrics as belonging to the last area or queue it was showing at its termination or by the end of the requested time-frame.
- An area is set on the first call verb having an area different from NONE. This may or may not be the first tracking verb on the call.
- You cannot specify an area of NONE after a call had an area specified. This behavior is implemented in order to avoid logging the area on each IVR key-press on manually-created dial-plans. If you do, the area is not changed
- If you specify a different area for the call, the new code replaces the old one.
- When a call finally enters a queue, the queue code replaces the area code

The area name might be any valid name in Asterisk (lowercase ascii7 string, not containing spaces or pipes). Uppercase names are automatically lowercased upon reading. It is perfectly allowable to use the very same code for a queue and an area.



We suggest using a name similar to "ivr-abc" in order to make it clear it is not a normal queue.

In cases where the IVR tree is extremely complex, you might want to specify multiple areas for the same call. In this case, when you run a report, you must specify all the different areas that make up your IVR tree as the allowed queue(s) for the report. As always:

- Any call which final state was without an area will be visible in all reports.
- Any call which final state is in a certain area is only visible to users having that area defined as a possible queue

Implementing IVR tracking

In order to track IVR data, we need to tell QM a few bits of information:

- When a digit is pressed, the time-stamp when the event happens, the digit pressed and the name of the IVR menu. These are logged through the verb `INFO IVRAPPEND`
- When a call is started, you may want to log the moment when the call reaches the PBX; the calling number and optionally the calling DID. This is done through `INFO IVRSTART` and it is optional
- When a goal is reached, you may want to log the name of the goal. This is done through `INFO IVRGOAL` and it is optional
- When a call is hung up in the IVR, you may want to log this information in order to have detailed IVR timing. This is done through `INFO IVRHANGUP` and it is optional.

Dial-plan tracking

In order to implement dial-plan tracking, the unique-id logged must match the unique-id of the call leg that will reach the queue.

In order to track the **beginning of a call**, you should produce an INFO IVRSTART record like the following one:

```
1353461650|1353461627.33271|ivr|NONE|INFO|IVRSTART|1234|5556777
```

This way you track both the caller-id of the caller ("1234" in our case) and the DNIS the call came through ("5556777" in our case). You may leave either field blank if that information is not needed.

The record above is optional - if it is missing calls will be tracked from the first IVR menu, and the traversal time for that IVR menu will be set to zero.

In order to **track an IVR digit**, you would use the following format:

```
1353461660|1353461627.33271|ivr|NONE|INFO|IVRAPPEND|1|menu-2
```

In this case, we tell QM that the key 1 was pressed in IVR "menu-2". The IVR event duration (that is, how much it took the caller to make a selection) is automatically inferred from the end of the previous event. If this is not the case, you have an option to pass it explicitly to QM as the last parameter (6 seconds in the case shown):

```
1353461670|1353461627.33271|ivr|NONE|INFO|IVRAPPEND|1|menu-2|6
```

If a **goal** is reached, you should mark this by producing a record like:

```
1353461680|1353461627.33271|ivr|NONE|INFO|IVRGOAL|attendant3|menu-2|6
```

The goal in this case is called "attendant3". The IVR name is optional and not currently used. The explicit timing duration is optional.

A call can have only one goal. Any call hitting a queue is considered to have reached a goal.

In order to tell QM that a call **hung-up in an IVR**, you should produce a record like:

```
1353461660|1353461627.33271|ivr|NONE|INFO|IVRHANGUP|
```

Tracking areas

The logging is changed so that the INFO verbs allows the reading of this piece of information. So any sub-verb (e.g. INFO DNIS, INFO IVRSTART, INFO IVRAPPEND and more) will work correctly with this.

An example of logging might be:

```
1345678|1234.5678|IVR-1|NONE|INFO|IVRSTART|1234|12345
```

This logs the IVRSTART as belonging to area IVR-1.

It is perfectly valid to assign a call to an area from a specific point in time onwards (generally because they made a defining IVR decision)

```
1345668|1234.5678|NONE|NONE|INFO|IVRSTART|1234|12345
1345679|1234.5678|IVR-3|NONE|INFO|IVRAPPEND|3|MYMENU
```

As shown above.

Deprecated IVR tracking verbs

The following records are understood by QM but should not be used anymore:

```
1000000|214530.A|sk-fissi|NONE|INFO|IVRWAIT
1000000|214530.A|sk-fissi|NONE|INFO|IVR|345
```

The FreePBX module

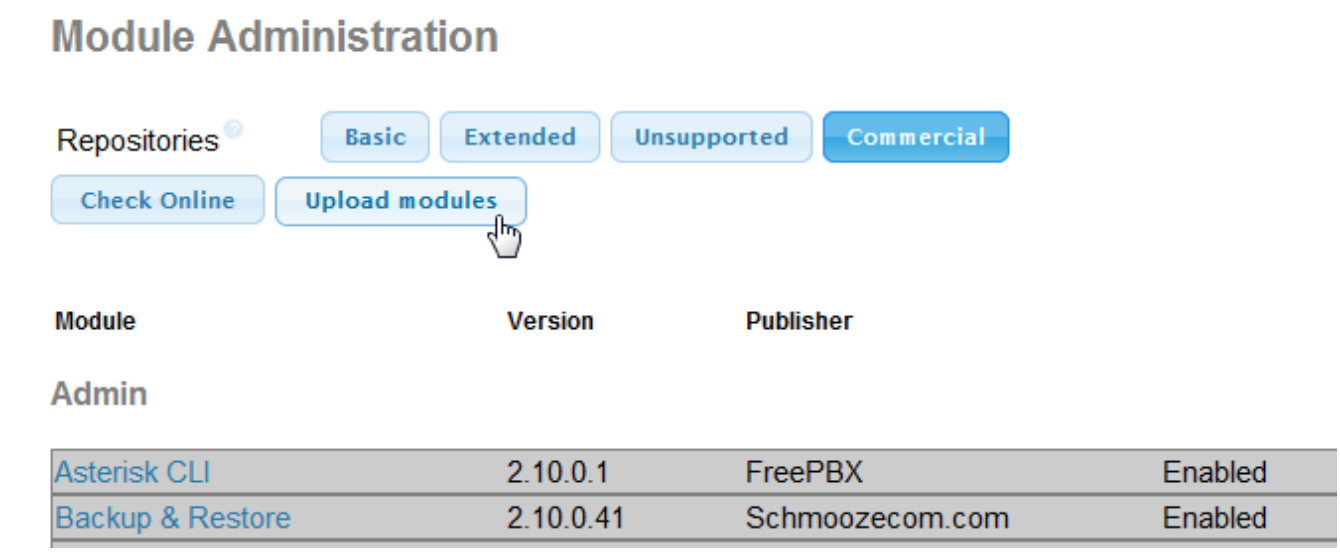
Since FreePBX 2.11 a new module is available on the FreePBX SVN repository. This module allows to easily integrate FreePBX IVR with QueueMetrics. The module will be published shortly and will be available through the standard FreePBX modules administration page. Till this date, you could

manually install the module following the below steps. In a command shell, type the code:

```
svn co http://svn.freepbx.org/modules/branches/2.11/queuemetrics
tar -cvf queuemetrics-2.11.0.1.tar queuemetrics/
gzip queuemetrics-2.11.0.1.tar
```

This generates a queuemetrics-2.11.0.1.tar.gz module you can install on your FreePBX box.

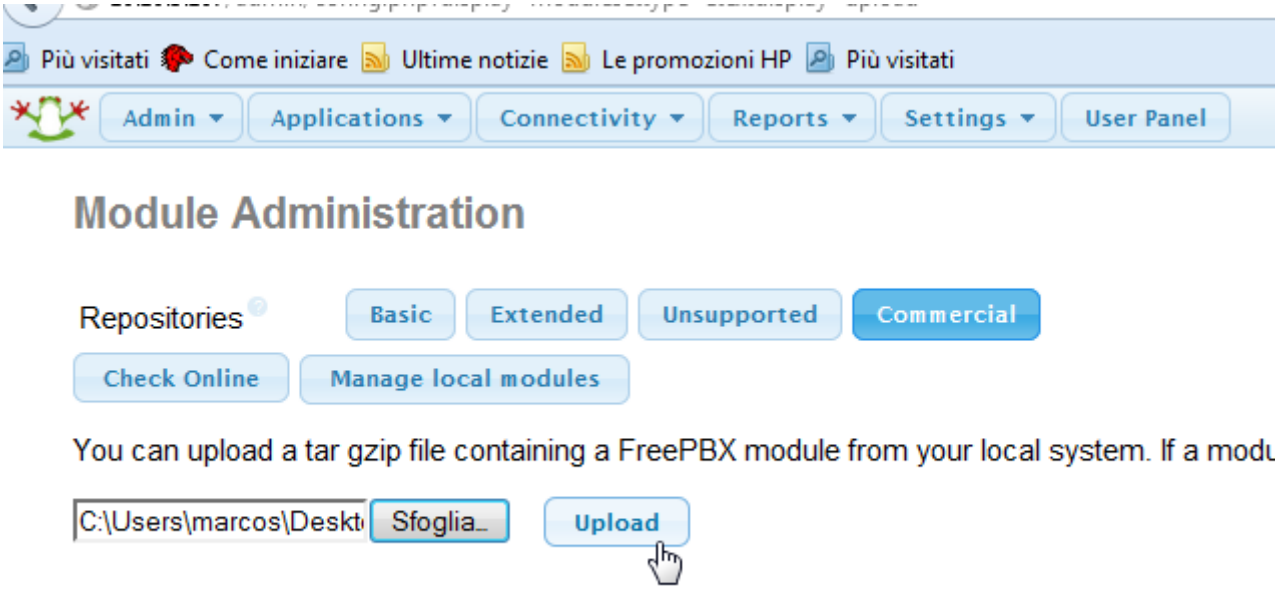
In the FreePBX modules administration page press the "Upload modules" button.



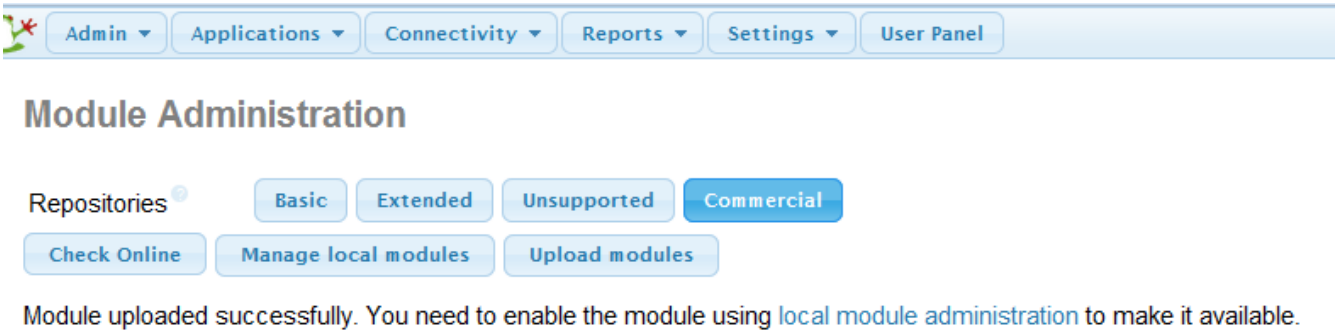
Browse to the queuemetrics-2.11.0.1.tar.gz file you downloaded from the FreePBX repository.



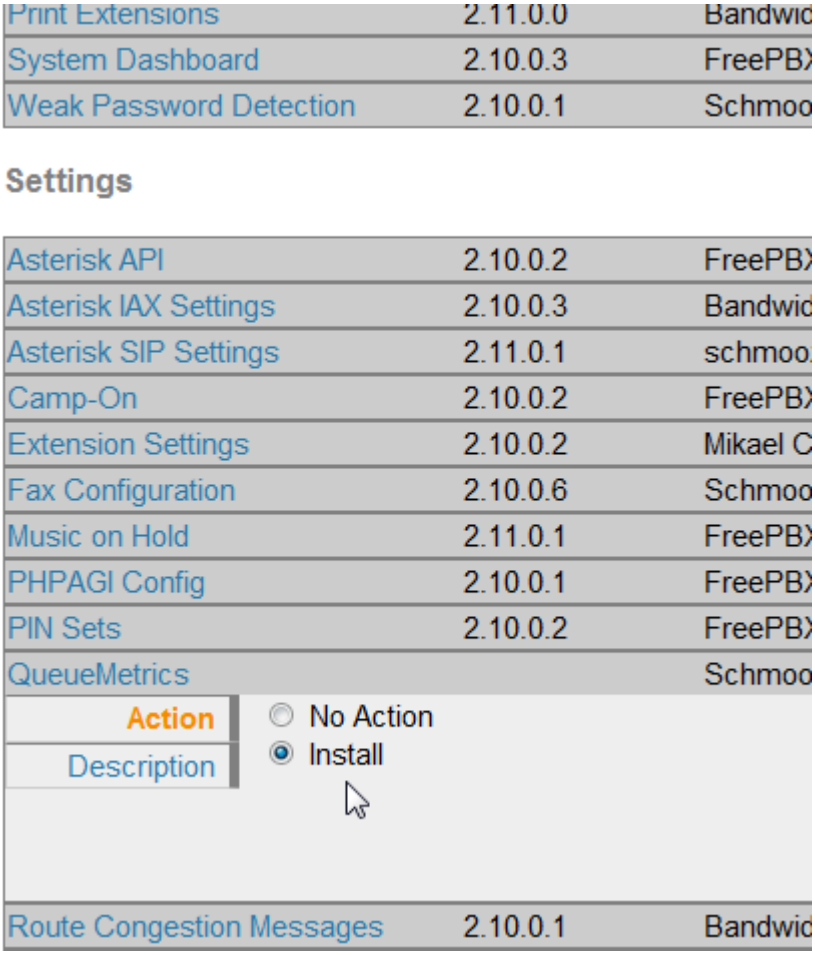
then press "Upload" to store the file on your FreePBX box.



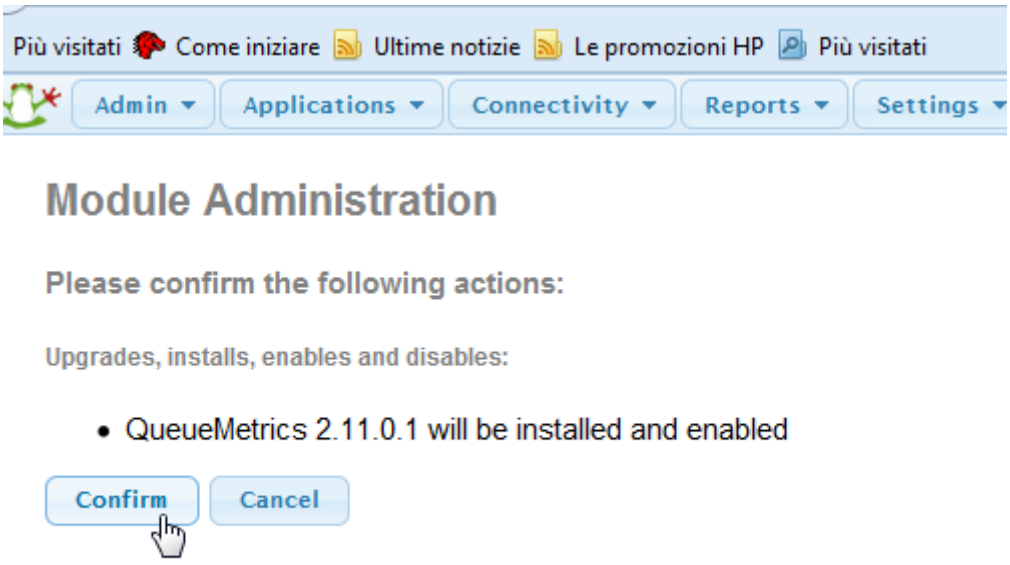
You will find the module in the modules administration page.



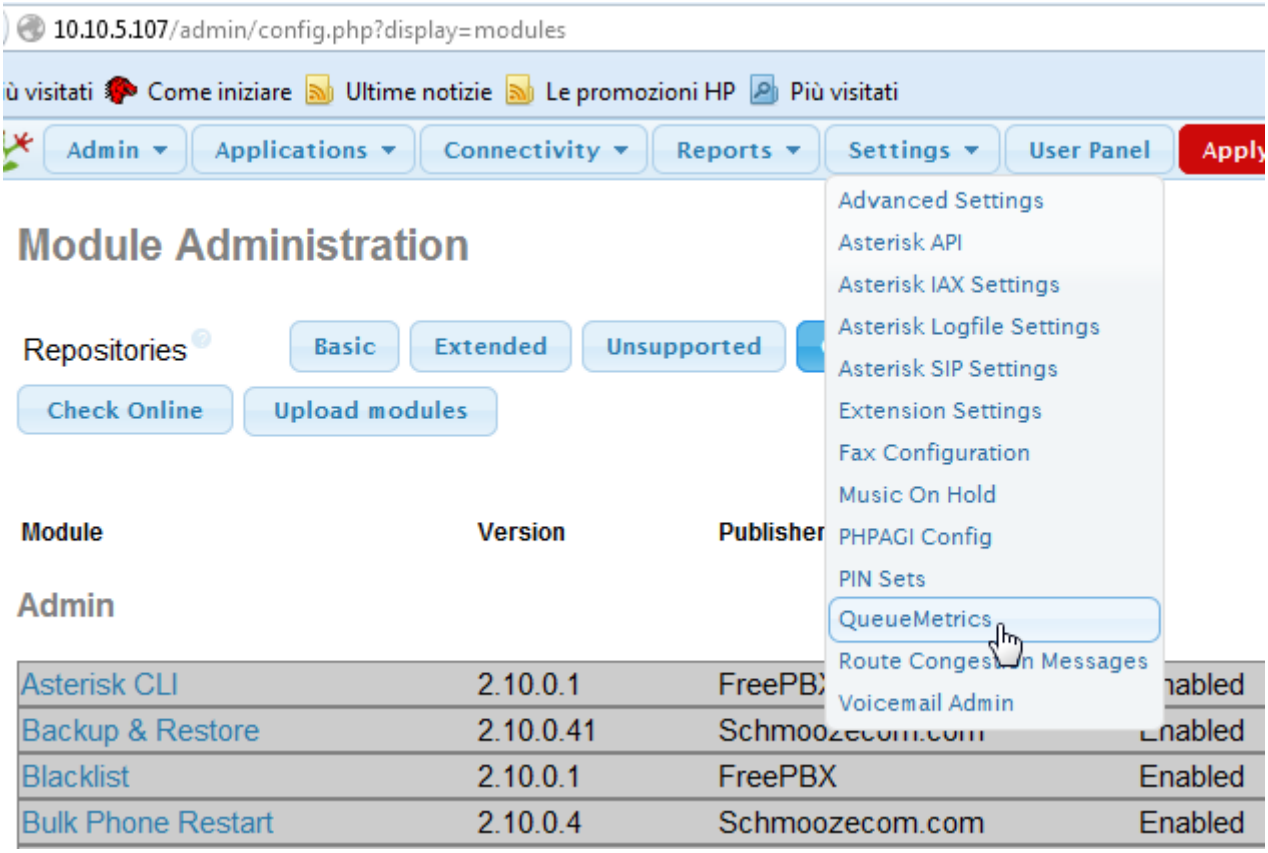
Click on it then select the "Install" option. Click on "Process" at the bottom of the page.



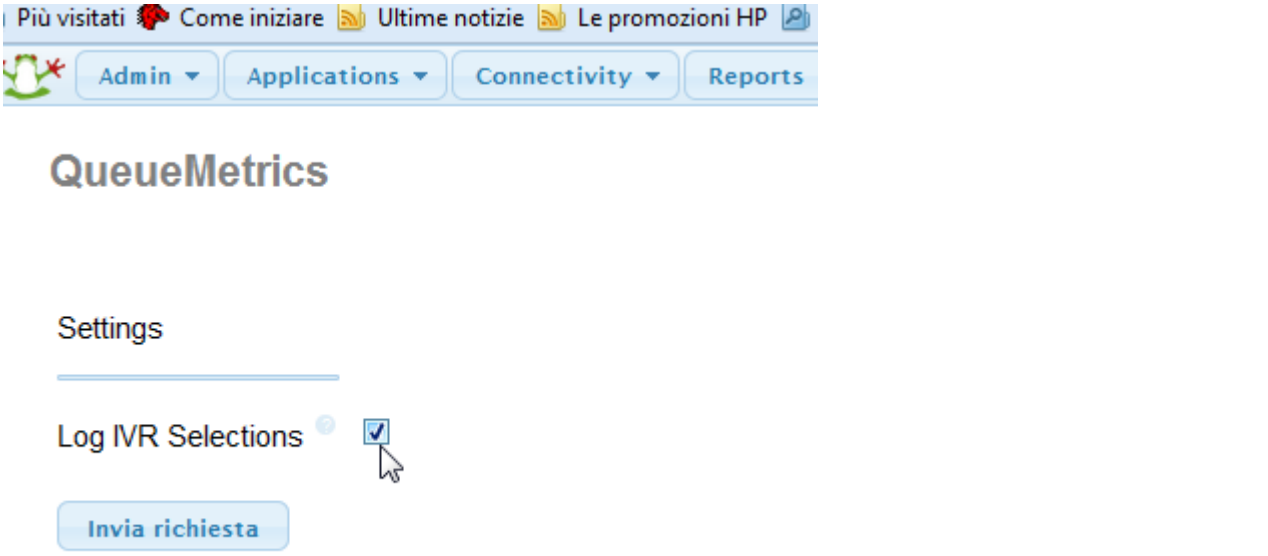
Then confirm.



At the end of the procedure you will find a new option on the Settings menu.



This opens a new page settings where you can enable the QueueMetrics IVR log routines.



Appendix A: 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
USRADMIN_RESTRICTED	User can edit other users, but only in selected classes. See Restricted user administrators
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 and call features
USR_CALLTAG	User can edit call tags
USR_PCODE	User can edit pause codes
USR_MYSQL	User can see the MySQL database page
USR_QAEDIT	User can edit the set of Quality Assessment metrics
USR_AGROUPS	User can edit agent groups
USR_IVR	User can edit the list of known IVR selections
USR_DNIS	User can edit the list of known DID/DNIS
USR_CBTS	User can edit the list of known CBTs
USR_SKILLS	User can edit agent skills
USR_CASES_EDIT	User can edit cases
USR_CLIENTS_EDIT	User can edit clients
REALTIME	User can see real-time stats
RTLIVE	User can access the Live 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
CALLMONITOR_ADDTAGS	The user can add markers to recorded calls

KEY	MEANING
CALLMONITOR_DELTAGS	The user can delete markers from recorded calls
MON_AUDIO	The user can monitor (spy) a real-time call
MON_WHISPER	The user can whisper to the agent on a real-time call
MON_BARGE	The user can barge-in on 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
QA_TRACK	User can enter Quality Assessment data
QA_REPORT	User can run Quality Assessment reports
QA_REMOVE	User can delete Quality Assessment data
QA_REPLACE	Together with QA_REMOVE, user can edit an already submitted Quality Assessment data
CLOSECALLS	This user can close ongoing calls from the Real-time or the historical stats page.
BRO_MSG	This user can send broadcast messages to agents
MON_IM	This user can start an IM chat to an agent
CONFIG	This user can start the auto configuration wizard (attended and unattended mode)
USR_AGROUPS	This user can edit custom agent groups
PAYROLL	This user is allowed to check the payroll page

KEY	MEANING
PAYROLL_REMOVENOTE	This user is allowed to remove a note in the payroll page
QLOG_EDIT	This user is allowed to edit the queue_log records
QLOG_LNGR	This user is allowed to edit session data by making it longer (they must hold QLOG_EDIT as well)
USR_REPORTS	Edit QueueMetrics reports
USR_REPORTS_EXPORT_EDIT	Edit PDF and XML export jobs
USR_REPORTS_EXPORT	This user is allowed to export whole reports in PDF and XLS
USR_KNUMBERS	User can view and edit known numbers
USR_DIAGNOSTICS	User can run diagnostics
TASKS	User can see/edit tasks he sent and he received
TASKS_VIEWALL	User can see all tasks present in the database
BATCH_ADM	Audio export - Creates and closes batches
BATCH_ADD	Audio export - This user can add calls to an open batch
BATCH_VIEW	Audio export - This user can see batches
BATCH_DEL	Audio export - This user can remove calls from a batch.
QA_PERF_TRACK	Can run Agent Performance Tracking
QA_PERF_RULES	Can define rulesets for Agent Performance Tracking
QA_CALREP	Access to Grader calibration reports
TASKS	User can see their tasks
TASKS_REP	User can access the tab showing Task Statistics
TASKS_VIEWALL	User can see other people’s tasks
TASK_PUSH_TC	User can access to the Training and Coaching page for sending CBTs and Coaching tasks
USR_SYSLOG	User can view the system’s audit log

KEY	MEANING
KEYUPDATE	User can install a new QueueMetrics activation key
QA_GRADER	Allows access to the Grader's page and related statistics
QUEUE_LST	Allows direct access to the call list (skipping the Reports page)
VISITOR	Grants access to a partial set of statistics and features such as the Remote Monitoring page. The VISITORS class holds this key (plus USER MON_VNC MON_AUDIO).
RT_ADDMEMBER	User can add agents to a queue from the realtime page
RT_REMOVEMEMBER	User can remove agents from a queue from the realtime page
RT_PAUSEAGENT	User can pause agents from the realtime page
RT_UNPAUSEAGENT	User can unpause agents from the realtime page
RT_SENDTEXTAGENT	User can send a SMS to the agent's phone from the realtime agent (Asterisk 10+ only)
RT_HANGUPCALL	User can hangup a live call from the realtime page
RT_TRANSFERCALL	User can transfer a call to a specific extension from the realtime page
EDIT_CFG	User can edit the local system preferences (see Configuring system preferences)
SSAREPORT	Enable Self Service reporting for an Agent
SQUERY_ADD	Users can save reports as homepage links
SQUERY_FORALL	User can save public reports
WALLBOARD_ADD	Users can save new wallboards
WALLBOARD_FORALL	User can save public wallboards
WALLBOARD_PHONE	User can use the wallboard softphone
PBXAPI	User can call PBX-level services

KEY	MEANING
SYNCHRONIZER	User can access the Configuration Synchronizer
HIDE_NUMBERS	Users holding this key will have telephone numbers masked. See Displaying numbers
NODIALOUT	The agent cannot select a queue, make AMO calls from the softphone, or make outgoing calls autonomously. See Disable outbound autonomous calls
USR_HID	The agent can select an HID device in the softphone to answer and hang up calls.
QUEUE_LOGIN_ASSIGNED_ONLY	Users can only login to their assigned queues.
CHAT_READ	User can view chats or transcriptions
CHAT_WRITE	User can insert new chats or transcriptions
CHAT_EDIT	User can modify chats or transcriptions
CHAT_DELETE	User can delete chats or transcriptions
EPHEMERAL_READ	User (or external system) can read ephemeral, time-limited key/value attributes coming from external integrations
EPHEMERAL_WRITE	User (or external system) can write ephemeral, time-limited key/value attributes coming from external integrations
TRANS_SERVER	User can view and update the pool of transcription servers - see Call transcriptions and Automated QA
USR_AMO	Administrative access to the AMO (Assisted Manual Outbound) pages
USR_CRONJOBS_EDIT	User can edit Scheduled Jobs
USR_CUSTOMBLOCKS_EDIT	User can edit Custom Data Blocks
AUTOANSWER_PHONE	User can toggle Auto Answer on the softphone
WQLOADER	User can upload logs over HTTP

Obsolete

KEY	MEANING
AGAW	This user can access AGAW facades (for agents).
AGAW_ADM	This user can access the AGAW administration screens
AGAW_REP	This user can access the AGAW supervisor screen

Appendix B: 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.



You can check the current dialplan of a working Asterisk system from the Using the DbTest Diagnostic Tools page.

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.

Whenever an action is invoked by a logged-on user, the following variables are set at the channel level:

- QM_LOGIN is the login of the current Qm user asking for the action to be performed
- QM_CLASS is the current class the requesting user is in.

This makes it possible to perform addirtional security checks or auditing at the Asterisk level, but is not used by the supplied dialplan.

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

- **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 of inbound traffic possible through the command ChanSpy(). The variables QM_AGENT_CODE, QM_EXT_MONITOR, QM_AGENT_EXT are set, as well as QM_CALLERID, QM_QUEUE and QM_QUEUE_URL. 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,NoOp( "QM_LOGIN: ${QM_LOGIN}" )
exten => 11,6,ChanSpy(${QM_AGENT_CODE})
exten => 11,7,Hangup
```



this may have important security implications - see Listening to live calls: Unattended Call Monitoring.

- **12**: *Call status code*
  This extension logs a calls status code. The variables CALLSTATUS, CALLID, QM_LOGIN 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 '${AGENTCODE}' made by '${QM_LOGIN}'" )
exten => 12,3,System( echo "${EPOCH}|${CALLID}|NONE|Agent/${AGENTCODE} \
             |CALLSTATUS|${CALLSTATUS}" >> /var/log/asterisk/queue_log )
exten => 12,4,Hangup
```

- **14**: *Remote monitoring of outgoing calls*
  This extension makes unattended monitoring of outbound traffic possible through the command ChanSpy(). The variables QM_AGENT_CODE, QM_EXT_MONITOR and QM_AGENT_EXT are set, as well as QM_CALLERID, QM_QUEUE and QM_QUEUE_URL. The following example explains how the feature works:

```
; 14: Remote monitoring of outgoing calls - like SIP/callednumber
exten => 14,1,Answer
exten => 14,2,NoOp( "QM_AGENT_CODE: ${QM_AGENT_CODE}" )
exten => 14,3,NoOp( "QM_EXT_MONITOR: ${QM_EXT_MONITOR}" )
exten => 14,4,NoOp( "QM_AGENT_EXT: ${QM_AGENT_EXT}" )
exten => 14,5,NoOp( "QM_CALLERID: ${QM_CALLERID}" )
exten => 14,6,ChanSpy(SIP/${QM_CALLERID}|q)
exten => 14,7,Hangup
```

  Please note that you should set the channels SIP/xxxx to the names of your local outgoing channel or the name of the local SIP leg of the call.



this may have important security implications - see Listening to live calls: Unattended Call Monitoring.

- **16 and 17**: *Add/Remove call feature codes*
  This extension pair adds or removes a call feature code. The variables FEATCODE, CALLID, AGENTCODE, QM_LOGIN and FEATTEXT are defined. The following example explains how the feature works:

```
; extension 16: set call status code
exten => 16,1,Answer
exten => 16,2,NoOp( "QM: Setting call feature '${FEATCODE}' on call '${CALLID}' for
agent '${AGENTCODE}' made by '${QM_LOGIN}' with freetext ${FEATTEXT}" )
exten => 16,3,System( echo
"${EPOCH}|${CALLID}|NONE|Agent/${AGENTCODE}|INFO|FTR|${FEATCODE}|${FEATTEXT}" >>
```

```
/var/log/asterisk/queue_log )
exten => 16,4,Hangup

; extension 17: unset call status code
exten => 17,1,Answer
exten => 17,2,NoOp( "QM: Removing call feature '${FEATCODE}' on call '${CALLID}'
for agent '${AGENTCODE}' made by '${QM_LOGIN}' with freetext ${FEATTEXT}" )
exten => 17,3,System( echo
"${EPOCH}|${CALLID}|NONE|Agent/${AGENTCODE}|INFO|NOFTR|${FEATCODE}|${FEATTEXT}" >>
/var/log/asterisk/queue_log )
exten => 17,4,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 ${AGENT_EXT}@sip made by '${QM_LOGIN}'" )
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/${AGENTCODE} \
                    made by '${QM_LOGIN}'")
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}' made by '${QM_LOGIN}'" )
exten => 22,3,PauseQueueMember(|Agent/${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 AGENTCODEis defined. The following example explains how the feature works:

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

- **24** and **25**: *Agent AddQueueMember*
  These extensions (targeted to asterisk 1.2 the first, for asterisk 1.4 the second) dynamically add an agent to the specified queue. The variable AGENTCODE and QUEUENAME is defined. Only for the extension 25, the variables QM_AGENT_PRIOLBL and QM_AGENT_PRIONUM are set with the information related to agent priority in the queue: QM_AGENT_PRIOLBL could have the values U, M, W, S, respectively for agents not assigned in the queue, assigned as main, assigned as wrap, assigned as spill in the queue. The variable QM_AGENT_PRIONUM has the value 0 for agent not assigned in the queue or assigned as main in the queue, 1 for agents assigned as wrap, 2 for agents assigned as spill in the queue. The following example explains how the feature works:

```
exten => 24,1,Answer
exten => 24,2,NoOp( "QM: AddQueueMember (v1.2) Agent/${AGENTCODE} \
  on queue ${QUEUENAME} made by '${QM_LOGIN}'" )
exten => 24,3,System( echo "${EPOCH}|${UNIQUEID}|${QUEUENAME} \
  |Local/${AGENTCODE}@from-internal|ADDMEMBER|" >> /var/log/asterisk/queue_log )
exten => 24,4,Hangup

exten => 25,1,Answer
exten => 25,2,NoOp( "QM: AddQueueMember (v1.4+) Agent/${AGENTCODE} \
  on queue ${QUEUENAME} made by '${QM_LOGIN}' \
  with prioritylabel '${QM_AGENT_PRIOLBL}' and prioritynum '${QM_AGENT_PRIONUM}'" )
exten => 25,3,AddQueueMember(${QUEUENAME}|Local/${AGENTCODE}@from-internal)
exten => 25,4,Hangup
```

- **26** and **27**: *Agent RemoveMember*
  These extensions (targeted to asterisk 1.2 the first, for asterisk 1.4 the second) dynamically remove an agent to the specified queue. The variable AGENTCODE and QUEUENAME is defined. The following example explains how the feature works:

```
exten => 26,1,Answer
exten => 26,2,NoOp( "QM: RemoveQueueMember (v1.2) Agent/${AGENTCODE} \
  on queue ${QUEUENAME} made by '${QM_LOGIN}'" )
exten => 26,3,System( echo "${EPOCH}|${UNIQUEID}|${QUEUENAME} \
  |Local/${AGENTCODE}@from-internal|REMOVEMEMBER|"  \
    >> /var/log/asterisk/queue_log )
exten => 26,4,Hangup

exten => 27,1,Answer
```

```
exten => 27,2,NoOp( "QM: RemoveQueueMember (v1.4+) Agent/${AGENTCODE} \
  on queue ${QUEUENAME} made by '${QM_LOGIN}'" )
exten => 27,3,RemoveQueueMember(${QUEUENAME}|Local/${AGENTCODE}@from-internal)
exten => 27,4,Hangup
```

- **28**: *Agent custom dial*
  This extension lets able the agent to dial extensions through outbound queues from the agent's live page. The variable AGENTCODE, EXTTODIAL, and OUTQUEUE is defined. Is possible to force a specific caller ID uncommenting the queue where the Set function is called and, obviously, changing the caller ID information to your needs. The following example explains how the feature works:

```
exten => 28,1,Answer
exten => 28,n,NoOp( "QM: Agent Custom Dial. Dialing ${EXTTODIAL} \
  on queue ${OUTQUEUE}  made by '${QM_LOGIN}'" )
exten => 28,n,Set(QDIALER_QUEUE=${OUTQUEUE})
exten => 28,n,Set(QDIALER_NUMBER=${EXTTODIAL})
exten => 28,n,Set(QDIALER_AGENT=Agent/${AGENTCODE})
exten => 28,n,Set(QDIALER_CHANNEL=SIP/${QDIALER_NUMBER})
exten => 28,n,Set(QueueName=${QDIALER_QUEUE})
exten => 28,n,MixMonitor(Q-${QDIALER_QUEUE}-${UNIQUEID}.WAV|b|)
;exten => 28,n,Set(CALLERID(all)="1234567890" <1234567890>) ; Uncomment and change
this if you need to set your own caller ID
exten => 28,n,Goto(qm-queuedial,s,1)
exten => 28,n,Hangup
```

- **29**: *Send SMS to agent's phones*
  This extension allows the users holding the proper key to send a short message to the agent's phone from the realtime page. This feature is supported by Asterisk revision 10 and later versions and is disabled by default. Please note that this feature should be supported by the agent's phone.

```
exten => 29,1,NoOp( "QM: Send Text from Live Page. Sending text to ${EXTTODIAL}
made by '${QM_LOGIN}' " )
exten => 29,n,Set(MESSAGE(body)=From: ${QM_LOGIN} - ${MESSAGEBODY})
exten => 29,n,MessageSend(sip:${EXTTODIAL})
exten => 29,n,Hangup
```

- **30**: *Hangup a live call*
  This extension allows the users holding the proper key to send an hangup message to the PBX in order to hangup a live call from the realtime page.

```
exten => 30,1,NoOp( "QM: Call Hangup made by ${QM_LOGIN} for callID: ${CALLID} with
agent code ${AGENTCODE} from extension ${QM_AGENT_LOGEXT}")
exten => 30,n,ChannelRedirect(${CALLID},queuemetrics,10,3)
exten => 30,n,Hangup
```

- **31**: *Redirect a live call*
  This extension allows the users holind the proper key to send a transfer event to the PBX from the realtime page. This forces the live call to be transferred to a specified extension.

```
exten => 31,1,NoOp( " QM: Call redirect ,ade by ${QM_LOGIN} for callID: ${CALLID}
to extension ${REDIR_EXT}")
exten => 31,n,ChannelRedirect(${CALLID},from-internal,${REDIR_EXT},1)
exten => 31,n,Hangup
```



When using AddQueueMember/RemoveQueueMember to dynamically login/out to a queue is mandatory to match the agent code with their extension; eg. Agent/303 must be sitting at extension 303.



When using AddQueueMember/RemoveQueueMember to dynamically login/out to a queue the agent pause/unpause dialplan given must be changed to fit the current agent channels; eg. if Agent/303 is added to the queue as Local/\303@agents, the pause code specified in the dialplan must be changed to PauseQueueMember(|Local/\${AGENTCODE}@agents)

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 C: System preferences

QM stores system-wide preferences on its database. They can be edited as described in Configuring system preferences.

All properties are case-sensitive.



You can check the current set of system preferences from the Using the DbTest Diagnostic Tools page.



If you want to see a complete file at once, see Example.

Defaults

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. Might be a file, or a partition, e.g `sql:P001`, or `cluster:*`. See `sqlPreset` below.
`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.webloaderpbx`	Enable or disable the web proxy for actions generation. Please refer to the Uniloader User Manual for further information.
`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.showQueueCompositio n`	If true, show the details of the queues composing the aggregate queue; if flase, 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.useRawAgentSessions`	If true, show all agent sessions. If false, show only agent sessions with at least one call handled. Defaults to false.

Property name	Description
default.closeDuration	The default duration of a call that is manually closed. This is the wait time for calls that have not been answered and the talk time for calls that have been answered.
default.crmapp	If present and not empty it will enable the CRM integration column on the answered/unanswered call details tables. The key could be populated with an URL where some tokens will be expanded by the QM engine. Find a list of valid tokens here: CRM Integration One example of valid URL with token expansion could be: http://server/app?agent=[A]&unique=[U]
default.crmlabel	If present, used by the Realtime agent page to identify the CRM page on the dropdown menu
default.showAstClid	If present and set to true it will enable the asterisk unique ID column on the answered/unanswered call details tables.
default.showSecondsOnTotalCalls	If present and set to true, the summary report call time figures will be shown in hhmmss format instead of hours format
default.disablebackhistory	If set to true, disable the history back navigation button in browsers
default.secondsServiceLevel	The default SLA that Traffic Distribution graphs will use (see DD08). Default: 20 seconds
default.jobmanifest_language	The language used when generating a manifest file for exported jobs. Tipically it affects the ID3 tags stored in mp3 recorded call files
default.pausecoderequired	If true, agents are required to provide a valid pause reason when entering a pause from the agent page. Default value is false
default.agentCanAmendPause	If set to "Yes", the agent can amend the current pause code retroactively changing the pause code for the current pause in reports. If set to "No", agents that try to change pause code will have their current pause ended an a new pause with the new pause code will be initiated
default.noncontig.days	Which days to include in Custom Reports non-contiguous time - (1: Sun 2:Mon) e.g. 23456 means MON to FRI
default.noncontig.period1.start	Start and end times (as HH:MM:SS or HH:MM) for non-contiguous time reports
default.noncontig.period1.end	See above
default.noncontig.period2.start	See above
default.noncontig.period2.end	See above

Property name	Description
default.tasks.pingURL	If present and enabled, this property allows to specify a URL that is to be queried by the QueueMetrics server every time a task is completed/disputed by a person. All task information is sent to this URL. One example could be: http://server/index.html
default.timeZoneOffset	Defines The default time zone offset. Valid values are between -24 and 24 hours (default = 0)
default.ssarMaxReportPeriod	Defines the maximum reporting period for a Self Service agent report. Default set to 15 days
default.export.orientation	Default page orientation for PDF whole report export buttons (value: portrait or landscape)
default.displayHomePageNews	Enable/Disable the QueueMetrics news block present on the home page
default.disable_directlogin_to_agentpage	If set to true, users holding AGENT keys are not directed to the Agent page when logging on QueueMetrics
default.wombat.agent_recall_enabled	Enable/Disable the access to the Recall Scheduler panel in the Icon agent page. (since QM 15.02.4)
default.groupingMargin	Controls the margin used for FCR call clustering. Default 168hrs (7 days)
default.rewriteHotdeskingChannels	Enables rewriting of hotdesking information in data1 field of queue_log table. Default true.
default.rememberLastExtension	Controls the behaviour of the agent login panel in the agent's page. Default is set to true, which remembers the last extension the agent used
default.show_agent_priority	Default value is **true**, if set to **false** will hide the agent level in the agent page.
default.show_agent_priority_reports	Default value is **true**, if set to **false** will hide the agent level column in reports.
default.effectiveCallThreshold	The minimum length in seconds of a call to be counted as Effective; default is 150 seconds. See AG22.
default.maxRowsXls	The maximum number of rows per table that will be exported in XLS format. Default is 1000.

Short calls

Property name	Description
default.shortCallWait	Exclude calls shorter than X seconds that ended with a caller abandon. Default: -1, i.e. *off*.
default.shortCallTalk	Exclude taken calls shorter than X seconds that ended with a caller abandon. Default: -1, i.e. *off*.

Property name	Description
default.shortAttempt	Exclude attempts shorter than X seconds from attempt count. Default: -1, i.e. *off*.
default.shortcall_treshold	Controls the threshold in seconds for short/long calls for reports, default is 30
default.shortCallsLimit	The default Short Call limit that Traffic Distribution graphs will use (see DD08). Default: 5 seconds

Maximum reporting period

Property name	Description
report.maxAllowedDays	Reports that are longer than these many days cannot be run. Defaults to 0, meaning "No limit". The number of days in the current reporting period is approximated to the lowest value (that is, a report for a 36 hours counts as 1 day).
report.maxAllowedDaysExceptions	A set of security keys and maximum number of days, e.g "KEY1:18,KEY2:23". If maxAllowedDays is set, then the most favourable value for keys that the user holds is then used. Keep in mind that the most favourable value you can set is 0, meaning no limit. By setting these exceptions, you can control on a class-by-class or even user-by-user basis the actual maximum period to be enforced. You can use either system keys or your user-defined keys.
report.maxAllowedDaysHint	A hint message that will be added to the error if there is a maxAllowedDays violation. Might be used e.g. to tell the user to run large reports on a different instance.

Any violations will be logged on the syslog.

SMTP settings

All the settings for your SMTP host. Used by QueueMetrics to send reports by e-mail. If you encounter issues with e-mail delivery, please refer to the SMTP debug page at SMTP test.

Property name	Description
default.smtphost	SMTP server host name or IP address
default.smtpport	SMTP server host port
default.smtpfrom	Sender E-mail address used by QueueMetrics
default.smtpuser	Username to authenticate to the SMTP host (Mandatory)
default.smtpassword	Password to authenticate to the SMTP host (Mandatory)
default.smtpssl	Use SSL when sending mail (value: true or false)
default.smtpDebug	SMTP transaction are reported on (value: true or false, default false)

Property name	Description
default.smtpDailyQuota	Maximum number of emails that can be sent in a day to avoid flooding your SMTP provider. Defaults to 200. If 0, no limit.
default.smtpDailyUserQuota	Maximum number of emails that can be sent in a day by a specific user. Defaults to 50. If 0, no limit.

Call SLA

It is possible to have a different definition for the inital part of the SLA, having e.g. SLA computed every 5 seconds up to 30 seconds and every 10 seconds up to 60.

Property name	Description
sla.max_initial_delay	The max initial delay and interval that will be shown in the SLA graphs
sla.initial_interval	
sla.max_monitored_delay	The max delay and interval that will be shown in the SLA graph graphs
sla.interval	

Platforms

The following properties control the general settings for all platforms. Each platform driver then has its own set of properties, see Platforms.

Property name	Description
platform.allows.addfeaturecode	If set to false, disables this action even when the driver allows it. Default true.
platform.allows.addmember	If set to false, disables this action even when the driver allows it. Default true.
platform.allows.agentlogin	If set to false, disables this action even when the driver allows it. Default true.
platform.allows.agentlogout	If set to false, disables this action even when the driver allows it. Default true.
platform.allows.calloutcome	If set to false, disables this action even when the driver allows it. Default true.
platform.allows.customdial	If set to false, disables this action even when the driver allows it. Default true.
platform.allows.pause	If set to false, disables this action even when the driver allows it. Default true.
platform.allows.removefeaturecode	If set to false, disables this action even when the driver allows it. Default true.

Property name	Description
platform.allows.removemember	If set to false, disables this action even when the driver allows it. Default true.
platform.allows.sendtext	If set to false, disables this action even when the driver allows it. Default true.
platform.allows.softhangup	If set to false, disables this action even when the driver allows it. Default true.
platform.allows.spy_inbound	If set to false, disables this action even when the driver allows it. Default true.
platform.allows.spy_outboud	If set to false, disables this action even when the driver allows it. Default true.
platform.allows.transfer	If set to false, disables this action even when the driver allows it. Default true.
platform.allows.unpause	If set to false, disables this action even when the driver allows it. Default true.
platform.allows.write_info_var	If set to false, disables this action even when the driver allows it. Default true.

Parser

The following properties control how QueueMetrics interprets data read from Asterisk. Depending on the version of Asterisk you are using, they may have to be set differently from the system defaults in order to get a correct picture of what is going on.



All these parameters only influence the way the queue_log is parsed, but make no change in the way QM handles interactions to the PBX.

Property name	Description
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.considerIncompletetEntities	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 *Local/xxx@context* to *Agent/xxx* . Default: false.
default.rewriteLocalWithQueue	If true, extension *Local/123@ext* on queue *Q1* is read as *Agent/Q1-123* . Default false.
default.joinMultiStintCalls	If true, multi-stint calls in the current analysis set are joined by default

Property name	Description
default.useEndingChannelName	If true, the last reference to an agent is used as its name (in case they are different)
default.stripChannelNames	If true, anything after the "-" sign is deleted (ie. SIP/203-abcd is read as SIP/203). If false, the agent channel name is loaded as in the queue_log file. Default: true.
default.ignoreQueueStarts	If false, agents are logged on and calls closed on QUEUESTART records. Otherwise they will be ignored. Default false.
default.alwaysLogonUnpaused	If set to true, when an agent logs on, he will always be considered unpaused. This means: when an agent logs on, are they always unpaused, or should we keep their last pause status?. Defaults to false (keep last status).
default.ignoreRingNoAnswer	If true, the analyzer will ignore the RINGNOANSWER verbs in the queue log in favour of AGENTATTEMPT verbs.
default.exitOnAgentDumpSysCompat	If true, AGENTDUMP and SYSCOMPAT verbs considered call closure records. If false, they are counted as failed attempts.
default.maxOngoingWaitTime	If set > 0, calls having more than the number of seconds of wait time are skipped. Defaults to 0 (all calls counted).
default.maxOngoingTalkTime	If set > 0, calls having more than the number of seconds of talk time are skipped. Defaults to 0 (all calls counted).
default.removeDoubleClosures	If set to true, a second closure record for a call just closed will be skipped. If the second record is a TRANSFER or similar, the call is closed as TRANSFER. This works around a long-standing Asterisk bug where a second call would be displayed with no telephone number. Default: true.
audit.maxReportTime	Any report (or real-time page run) that is not shorter than this value will be logged on the Audit log. Default: 30000 (milliseconds).
audit.maxRowsFetched	Any report (or real-time page run) that requres the scanning of more than this number of queue_log rows will be logged on the Audit log. Default: 1000000.
default.decodeKnownNumbers	If calling/callee numbers should be decoded when the number matches a Known number. Defaults to true.
default.hiddenNumberPolicy	The policy used to hide numbers. Defaults to hash. See Displaying numbers for a list of options.
default.skipAgentChannels	A comma-separated list of agent channels that must be ignored on reading. The supplied agent channels are in the normalized form, as controlled by default.stripChannelNames. Whenever one of those appears in the agent position of a queue_log record, the whole row is skipped as if it did not exist on the database. Note that it's only useful for outbound calls - for inbound queues, any calls taken by one of those agents will remain indefinitely open. Ditto for agent attempts - call routing data will be incomplete. Default: blank.

System administration

The following parameters affect how QueueMetrics interacts with the host system it is running on.

Property name	Description
admin.classesWhenRestricted	Space-separated names of the user classes that a restricted user administrator (key USRADMIN_RESTRICTED) can manage; names are not case-sensitive. Defaults to AGENTS. See Restricted user administrators.
pwd.defaultLevel	Default level for saved passwords. See Secure passwords.
pwd.minAllowedLevel	Minimum allowed level for passwords. See Secure passwords.

Layout and behavior

Property name	Description
layout.logo	Your company logo (full or relative path) - shall be resized to be an image with maximum size of 200 x 40 px, preferred size is 150 x 40 px. [More details]. The variable $WEBAPP refers to the local webapp, as an alternative use the full http://.. URL.
layout.splash	Welcome string displayed on the login page. HTML not allowed to avoid XSS risks.
default.noLicenseWarning	Set to *true* to disable license expiration notifications on the Home Page.
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
default.forceDefaultLanguage	Whether to always offer the default system language as defined above (default: false)
default.viewTechInfo	Is it possible to see Tech Info on the licence page and run DBTest?
url.qm	The URL of the webapp QM is running under, if not detected correctly. See Secure Mode
url.rss	The URL of the webapp QM is running under - used for RSS access. Like http://1.2.3.4:8080/qm
default.displayHomePageNews	Whether the news block on the home page should appear or not
layout.secureMode	Whether Secure Mode is enabled. Default: false.
default.decimalDigits	Number of digits to display for floating-point numbers - defaults to 1. See warning below.
format.date	Overrides the default full date format for all languages. Usually similar to yyyy-MM-dd. See warning below.

Property name	Description
format.dateTime	Overrides the default full date and time format for all languages. Usually similar to MMMM dd yyyy, HH:mm. See warning below.
format.dateTimeShort	Overrides the default "short" date and time format for all languages. Usually similar to MM/dd - HH:mm:ss. See warning below.
format.time	Overrides the default full time format for all languages. Usually similar to HH:mm:ss. See warning below.
format.timeShort	Overrides the default "short" time format for all languages. Usually similar to HH:mm. See warning below.



Please note that date, time and number formatting MUST follow the rules specified in https://docs.oracle.com/javase/8/docs/api/java/time/format/DateTimeFormatter.html - failure to provide a valid format string may break reporting and/or render reports unreadable. Also, changing these properties requires a full restart of the webapp and possibly cleaning Tomcat caches.

Database Access

The following properties define the fields used by the table in MySQL storage. See Monitoring clusters with QueueMetrics for complete information.

Used with sql:P001|i or cluster:*|i, where i is the preset code. The common format sql:P001 with no pipe implies using preset 1.



While it is possible to use different table formats, it is not advisable to do so unless you can also add the correct access indexes. It is usually easier to simply upload all data to the QM database and access data from there.

Property name	Description
sqlPreset._i_.table	Sets the table name for preset *i*
sqlPreset._i_.f_time_id	The time columns.
sqlPreset._i_.use_timestamp	True: time is a Unix timestamp; False: time is an SQL date-time
sqlPreset._i_.f_call_id	
sqlPreset._i_.f_queue	
sqlPreset._i_.f_agent	
sqlPreset._i_.f_verb	
sqlPreset._i_.f_partition	May be left blank for partition-less schemas
sqlPreset._i_.f_data1	
sqlPreset._i_.f_data2	
sqlPreset._i_.f_data3	
sqlPreset._i_.f_data4	
sqlPreset._i_.f_data5	

Property name	Description
sqlPreset._i_.f_incr	The order-preserving index column. May be left blank, but this may lead to incorrect results.

Realtime Page

Property name	Description
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.aggr_by_queue_invisible	Is the aggregated by queue panel in the realtime page invisible by default? 0 false, 1 true
realtime.aggr_by_tag_invisible	Is the aggregated by tag panel in the realtime page invisible by default? 0 false, 1 true
realtime.agent_and_outcome_invisible	Is the aggregated by agent and outcome 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.
realtime.startHour	The starting hour of the day, in order to compute the realtime report. It can be either a fixed hour (e.g. 3: from the last 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.waitAlarmOnLiveCalls	Decide whether to check for alarms on the wait time of ongoing conversations.
realtime.hideExportButtons	If true, hide export buttons on the Real-time page. Defaults to false.
realtime.absolutePauseTimes	If true, the start of the current pause is shown as an absolute hour; if false, it is shown as the time passed since.
realtime.calls_invisible.buttonEnabled	Decide which buttons o the real-time page can be toggled by the user. Buttons not enabled are set to their default value.
realtime.agents_invisible.buttonEnabled	
realtime.members_only.buttonEnabled	
realtime.aggr_by_queue_invisible.buttonEnabled	

Property name	Description
realtime.aggr_by_tag_invisible.buttonEnabled	
realtime.agent_and_outcome_invisible.buttonEnabled	
realtime.assignedLocationsOnly	If true, the user will not be able to monitor without a given location. See page Using Locations for more details.
realtime.useRowCache	Cache result objects for the Real-time and Agents page when using SQL or cluster storage. Set to true to enable. Defaults to false.
realtime.show_incorrect_queue_sets	If true, queues the agent is working on and he's a member of are displayed in black; queues the agent is working on but he's not a member of are displayed in brown; any queues the agent is a known member of but he's not currently logged on to are displayed as a tooltip to the gray "Down Arrow" symbol. Default false.
realtime.preventLoggedAgentInfoChanges	If true, or missing, the agent cannot change their code and extension if logged on at least one queue
default.callStatusTimeout	How much time (in seconds) an agent have to set an outcome code to the call after the call has finished. Defaults to 1800.
realtime.relativePauseTimes	Adds a new column (Paused For) to the "Agents currently logged in" table in the real-time page. This column shows how long an agent has been on pause, only if the pause is still ongoing.
realtime.ignoredQueues	One or more queues, separated by commas, that will be ignored by Presentialist. Default is _q-outbound, that is the "ignored calls" queue created by Uniloader in outbound tracking mode. See Real-time Performance.

Wallboard

Property name	Description
realtime.useNewWallboard	Whether to use the new user-configurable wallboard. Defaults to true.
default.wallboard_kiosk_user	Default user for the Wallboard Kiosk URL.

Agent's Realtime Page

Property name	Description
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.

Property name	Description
realtime.agent_autoopenurl	When the real-time page for an agent shows a new call in the call list, and if the call detail contains an URL, this URL will be open in a new browser window.
realtime.agent_button_X.enabled	Enable or disable a custom button in the realtime page. X shall be an integer between 1 and 4.
realtime.agent_button_X.caption	This is the label associated to a button.
realtime.agent_button_X.url	Defines the URL that will be opened when the button is pressed. The tokens [A] and [U] are expanded by QueueMetrics with, respectively, the Agent's ID and the most recent call Asterisk Call Unique ID as displayed in the call list. If no calls are present, *Unspecified* will be used instead.
realtime.agent_button_X.channel	Defines the first leg to be used in a dial command issued to the Asterisk server when the agent presses the button. E.g. *Local/\104@from-internal* .
realtime.agent_button_X.ext	Defines the second leg to be used in a dial command issued to the Asterisk server when the agent presses the button. E.g. *200@ext-queue* .

Property name	Description
realtime.dynamicLoginQueues	In the the new Icon panel based agent's page, defines how an agent is allowed to log in to their own queues: - *all*: The login panel shows all the queues visible by the agent in the available queue list and the agent is able to login on any of the queue - *registered*: The login panel shows only the queues assigned to the agent and the agent could login/out on a single queue or on all queues - *assigned*: The login panel shows only the queues assigned to the agent and the agent could login/out only on all queues only (in this situation the login panel behave like the "old" Log on and Log off pushbuttons) In the old Agent Realtime page defines what queues should be listed in the dropdown when agents log-in/out through the Add Member/Remove Member button. - *all*: The dropdown will show the "All assigned" option followed by the queues assigned to the agent (in QueueMetrics queues configuration) and queues where the agent was not assigned but it's free to log in dynamically - *registered*: The dropdown will show the "All assigned" option followed by the queues assigned to the agent (in QueueMetrics queues configuration) - *assigned*: The dropdown will show only the "All assigned" option. In this situation the Add Member/Remove Member buttons behave like the "old" Log on and Log off pushbuttons.
realtime.agentRegexp	Sets allowed agent codes (e.g. set as "2\d\d" to allow agents like "2XX" but not "3XX" or "2X")
realtime.extensionRegexp	Sets allowed extension codes. Any other code will be rejected. (e.g. set as "3\d\d" to allow extensions like "3XX")
default.lockedAgentPopupCode	If true, the agent cannot change their code in the login/logoff/pause pop-ups. Defaults to false.
realtime.useActivePolling	Whether to have the client poll the server "behind the scenes" and reload the page when a call is detected.
realtime.ajaxPollingDelay	The interval between each successive client poll. The expected delay for calls is one half of this value.

Property name	Description
realtime.all_includes_current_logins	If true, when logging off from ALL queues, any queue the agent is currently logged on will be disconnected. Default false.
realtime.agent_background_url	Default background web page for the realtime agent page. Normally used if no CRM and other background are used. If empty, the standard QueueMetrics background applies.
realtime.agent_web1_url	An URL used as first backgound in the Realtime Agent page. The key could be populated with an URL where some tokens will be expanded by the QM engine. Valid tokens are [A] (expanded with agent code). See also default.crmapp and default.crmlabel.
realtime.agent_web1_label	If present, used by the Realtime Agent page to identify this specific background on the dropdown menu
realtime.agent_web2_url	An URL used as second backgound in the Realtime Agent page. The key could be populated with an URL where some tokens will be expanded by the QM engine. Valid tokens are [A] (expanded with agent code). See also default.crmapp and default.crmlabel.
realtime.agent_web2_label	If present, used by the Realtime Agent page to identify this specific background on the dropdown menu
realtime.agent_webpanel1_url	An URL to be presented into the custom web panel in the Realtime Agent page. This URL does not support token expansion.
realtime.agent_webpanel1_label	If present, used by the Realtime Agent page to enable and identify the custom web panel on the dropdown menu
realtime.assertExtensionNotInUse	Before logging an agent in (in hotdesking mode) makes sure the extension is not in use. This option runs an analysis behind the scenes, so use it sparingly as will increase server load.
realtime.agentPausedOnLogin	When an agent logs in, if she is currently not working on any queues, pause them with the "autopauseCodeLogin" code. If the agent is currently working on some queues and paused, and adds new queues to their set, a pause is forced to make sure the Asterisk state is consistent on all queues. This option runs an analysis behind the scenes, so use it sparingly as will increase server load.
realtime.autopauseCodeLogin	The pause code to use for autopauses on login
agent.openUrlInPopup	If true, URLs in Icon will open in a separate window. Defaults to false.
agent.outcome_size	Maximum number of outcomes shown in the outcome menu before having to scroll. Defaults to 15, minimum value is 1 and maximum value is 50.
agent.pause_dialout	If true, stops agents from dialing out when on pause.
	QAP Agent Page

Property name	Description
agent.max_qap_calls	How many calls are supposed to be shown to each agent on their QAP agent call list. Defaults to 100. A higher number may be necessary, but it will have a performance impact.
agent.qap_cached_for_ms	How long is a cached real-time image valid to be used by the QAP agent page? default to 1500 ms. By making this value slightly higher or lower, you can adjust the amount of full recomputes that are needed to generate status data for the QAP agent page, and therefore a major amount of load on busy servers.
agent.leaveAllQueuesOnLogoff	If true, agents will be automatically logged off from all queues at logout

Asterisk Interaction

Property name	Description
callfile.dir	The call-file directory Asterisk uses to generate calls based on .call files. Must be writable by the Java process. Default _/var/spool/asterisk/outgoing As an alternative, you may enter a Manager interface URI here, in the format _tcp:user:password@server If you do, QM will not generate call-files but will use the manger interface to generate calls. The same field is used by the asterisk configuration wizard when "Single Machine AMI" was selected as source.
callfile.monitoring.enabled	If unattended audio monitoring is enabled on this system. Default true.
callfile.monitoring.channel	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. A number of variables act as placeholders to be substituted by the actual data Asterisk is using: **$AG**: the current agent; **$AE**: the agent's extension; **$EM**: the monitoring extension; See Listening to live calls: Unattended Call Monitoring for further information.
callfile.monitoring.extension	
callfile.monitoring.context	
callfile.agentpause.enabled	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.agentpause.channel	
callfile.agentpause.extension	
callfile.agentpause.context	
callfile.agentunpause.enabled	This function is used to end a pause from the Agent's page - see the sample dial plan provided.

Property name	Description
callfile.agentunpause.channel	
callfile.agentunpause.extension	
callfile.agentunpause.context	
callfile.agentlogin.enabled	This function is used to log in an agent from the Agent's page - see the sample dial plan provided.
callfile.agentlogin.channel	
callfile.agentlogin.extension	
callfile.agentlogin.context	
callfile.agentlogoff.enabled	This function is used to log off an agent from the Agent's page - see the sample dial plan provided.
callfile.agentlogoff.channel	
callfile.agentlogoff.extension	
callfile.agentlogoff.context	
callfile.calloutcome.enabled	This function is used to set the call outcome code from the Agent's page - see the sample dial plan provided.
callfile.calloutcome.channel	
callfile.calloutcome.extension	
callfile.calloutcome.context	
callfile.addfeature.enabled	This function, together with callfile.removefeature.enabled, is used to add a call feature from the Agent's page - see the sample dial plan provided.
callfile.addfeature.channel	
callfile.addfeature.extension	
callfile.addfeature.extension	
callfile.removefeature.enabled	This function, together with callfile.addfeature.enabled, is used to remove a call feature from the Agent's page - see the sample dial plan provided.
callfile.removefeature.channel	
callfile.removefeature.extension	

Property name	Description
callfile.removefeature.context	
callfile.agentdial.enabled	This function is not implemented yet.
callfile.agentdial.channel	This function is not implemented yet.
callfile.agentdial.extension	
callfile.agentdial.context	
callfile.outmonitoring.enabled	This function lets you monitor outgoing calls using a different piece of dial-plan, as outgoing channel names might be different from incoming ones.
callfile.outmonitoring.channel	
callfile.outmonitoring.extension	
callfile.outmonitoring.context	
callfile.agentaddmember.enabled	This function is used to dynamically add an agent to a specific queue from the Agent's page - see the sample dial plan provided.
callfile.agentaddmember.channel	
callfile.agentaddmember.extension	
callfile.agentaddmember.context	
callfile.agentremovemember.enabled	This function is used to dynamically remove an agent from a specific queue from the Agent's page - see the sample dial plan provided.
callfile.agentremovemember.channel	
callfile.agentremovemember.extension	
callfile.agentremovemember.context	
callfile.customdial.enabled	This function allows an agent to dial through one outbound specific queue - see the sample dial plan provided.
callfile.customdial.channel	
callfile.customdial.extension	
callfile.customdial.context	
callfile.writevar.enabled	This function lets an agent set a variable on a specific call - see the sample dial plan provided.
callfile.writevar.channel	
callfile.writevar.extension	

Property name	Description
callfile.writevar.context	

WombatDialer integration

Property name	Description
wombat.url	URL where the Wombat Dialer can be reached
wombat.username	Username needed to access to some WombatDialer services
wombat.password	Password for accessing some WombatDialer services
default.wombat.agent_recall_enabled	Enable/Disable the access to the Recall Scheduler panel in the Icon agent page.

Real-Time Sounds

Property name	Description
sound.yellowAlarm	Sound to be played if a yellow alarm is triggered. Can be either an absolute URL or a relative path
sound.redAlarm	Sound to be played if a red alarm is triggered. Can be either an absolute URL or a relative path

Cluster configuration

Property name	Description
cluster.servers	A set of servers, which names must be used for subsequent properties
cluster._servername_.manager	The manager API for this server, in the format tcp:user:pass@server. This field is also used by the asterisk autoconfiguration wizard when "Cluster AMI" source was selected.
cluster._servername_.queuelog	The queue log partition to use, in the format sql:P001
cluster._servername_.monitored_calls	The directory where monitored calls for this server can be found. If it starts with "http", an XML-RPC server to query this information
cluster._servername_.callfilesdir	The directory in which callfiles must be generated for this sever. Usually leave blank.
cluster._servername_.audioRpcServer	The URL of an XML-RPC server to be used for audio monitoring
cluster._servername_.agentSecurityKey	The key with which this cluster entry must be protected on the Agent's page

Property name	Description
`cluster._servername_.websocketurl`	The websocket URL used by the Realtime Agent page softphone for the selected server. It shold be in the followin form: ws://IPServer:Port/path
`cluster._servername_.sipaddress`	The SIP server name or IP, used by the Realtime Agent page softphone for the selected server
`cluster._servername_.rtcWebBreaker`	rtcWebBreaker flag used by the Realtime Agent page softphone for the selected server. The RTCWeb Breaker is used to enable audio and video transcoding when the endpoints do not support the same codecs or the remote server is not RTCWeb-compliant. Valid values are "true" or "false". If not present, the default value is "true".
`tokenfor._user_`	Forces a specific token for a HTTP-based data upload to a cluster, if the uploading end-point does not support the `token` field (that is, it passes it as an empty string). This way you can create a separate data upload user for each PBX. Eg. `tokenfor.loader1=aleph`.

Audio Monitoring

Property name	Description
`audio.server`	The PM to use for listening to recorded calls.
`audio.liveserver`	The PM to use for listening to live calls.
`default.audioRpcServer`	The URL of an external XML-RPC server for both listening of recorded calls and live call monitoring.
`audio.lookBack`	How many hours to check for midnight crossing. Used by the *LocalFilesByDay* PM.
`audio.decrypt`	The streaming decryption filter for encrypted recordings. Requires the JVM property `-Dqm.execCmds=1`.
`audio.html5player`	Plays audio in the browser using an HTML5 audio player. Defaults to false.
`mediaproxy.enabled`	Whether QM acts as a proxy for recordings - see MediaProxy. Default: false
`mediaproxy.linkvalidity`	How long a link generated by MediaProxy remains usable (in seconds).
`mediaproxy.header`	The name of a custom header for the security token, or `NONE`
`mediaproxy.token`	The value of the securty token sent in the header.

LLM

See Call transcriptions and Automated QA for background on these properties.

Property name	Description
`llm.captureUniqueId`	A regular expression, applied to the name of an uploaded media file to derive the call's unique-id (needed to match a transcription back to its call for automated QA): the first populated capture group is used as the unique-id. The regexp can have more than one capture group (e.g. to match different file naming conventions with a single pattern) - whichever group actually matches is used. If blank (the default), no unique-id is captured from the file name.
`llm.verbose`	If true, logs the raw request and response exchanged with the LLM model used for automated QA grading, at WARN level. Default: false.

QA

Property name	Description
`default.searchQA_byCallDate`	Defines if the QA reports should be calculated by call date or by filling form date
`default.skip_task_on_qagrading`	If not present or set to false, a new task will be sent to the graded agent each time a new qa form will be completed. If present and set to true, no tasks will be sent.
`qa.minPrcAllCalls`	The minimum percentage of calls to be graded for the "All calls" target, proposed on the Grader page when the user has not entered one yet. Range 0-100, default 100 (propose grading all calls). Set to 0 to disable the proposal, leaving the field blank as before this property existed.

WebRTC settings

Property name	Description
`default.sipaddress`	The SIP server name or IP address used by the Realtime Agent page softphone.
`default.websocketurl`	The WebSocket URL used by the Realtime Agent page softphone. Should be in the form: `ws://IPServer:Port/path`
`default.rtcWebBreaker`	Enables audio and video transcoding when endpoints do not support the same codecs or the remote server is not RTCWeb-compliant. Valid values: `true` or `false`. Default: `true`.
`default.iceservers`	List of ICE servers to use. For TURN servers with authentication, the format is: `turn:username:password@host`. Username and password should be URL-encoded if they contain special characters. Separate multiple servers with a comma. E.g. `stun:stun.example.com,turn:user:pass@turn.example.com`
`default.connection_recovery_max_interval`	Maximum interval in seconds between WebSocket reconnection attempts. Default: 30.

Property name	Description
default.connection_recovery_min_interval	Minimum interval in seconds between WebSocket reconnection attempts. Default: 2.
default.no_answer_timeout	Time in seconds after which an incoming call is rejected if not answered. Default: 60.
default.wallboardphone.server	(Wallboard softphone) The PBX server the phone connects to.
default.wallboardphone.websocketurl	(Wallboard softphone) WebSocket URL for the connection.
default.wallboardphone.username	(Wallboard softphone) SIP authentication username.
default.wallboardphone.password	(Wallboard softphone) SIP authentication password.
default.wallboardphone.display_name	(Wallboard softphone) Descriptive name shown to the called party when calling.
default.wallboardphone.iceservers	(Wallboard softphone) List of ICE servers to use. Same format as default.iceservers.
default.wallboardphone.register	(Wallboard softphone) Whether the softphone should register to the PBX.
default.wallboardphone.autoanswer	(Wallboard softphone) Whether the phone should automatically answer incoming calls. Only applies when no other session is currently active.
default.wallboardphone.connection_recovery_max_interval	(Wallboard softphone) Maximum interval in seconds between WebSocket reconnection attempts. Default: 30.
default.wallboardphone.connection_recovery_min_interval	(Wallboard softphone) Minimum interval in seconds between WebSocket reconnection attempts. Default: 2.
default.wallboardphone.no_answer_timeout	(Wallboard softphone) Time in seconds after which an incoming call is rejected if not answered. Default: 60.
qap.answer_src	HTTP address of a custom answer tone audio file.
qap.hangup_src	HTTP address of a custom hangup tone audio file.
qap.dialtone_src	HTTP address of a custom dialtone audio file.
qap.ringtone_src	HTTP address of a custom ringtone audio file.



In a clustered environment, sipaddress, websocketurl and rtcWebBreaker can be overridden individually per cluster node. See the *Cluster configuration* section for details.

Misc

Property name	Description
manager.dump	By setting this property to true, the dialog between Asterisk and QM used to show the Live! Page is dumped to the Catalina.out log file. This makes it possible to send it over to Loway for debugging purpouses.
export.conversionCommand	If present, this specify the batch script (full) name to be called by the MP3 HTTP Transfer implementor for export tasks. Requires the JVM property -Dqm.execCmds=1.
-Dqm.execCmds=1	JVM property (not a preference): allows calling external commands, e.g. for audio.decrypt and export.conversionCommand.

Autoconfiguration Wizard

Property name	Description
default.autoconf.source	Defines the default source that will be selected in the dropdown list on the wizard configuration page. It could assume the following values: **file** for File sources; **ami** for Single Machine AMI; **amic** for Clustered Machines AMI; **rtdb** for Asterisk Realtime Database; **quef** for Asterisk Queue Log file
default.autoconf.fileagents	Defines the default agents file definition will be shown in the configuration wizard page and will be read by the configuration wizard unattended mode
default.autoconf.filequeues	Defines the default queues file definition will be shown in the configuration wizard page and will be read by the configuration wizard unattended mode
default.autoconf.fileusers	Defines the default users file definition will be shown in the configuration wizard page and will be read by the configuration wizard unattended mode
default.autoconf.filequeuelog	Defines the default queue log file will be shown in the configuration wizard page and will be read by the configuration wizard unattended mode
default.autoconf.realtimedrv	Defines the database technology used by asterisk to read/write the realtime database (as Java Driver package) For MySQL the default value is com.mysql.jdbc.Driver
default.autoconf.realtimeuri	Defines the realtime database location and authentication parameters in jdbc format.

Synchronizer

Property name	Description
default.synchronizer_mode	Controls the default mode for the synchronizer page and for the QmPushCfgService Json service, default is "CR_UPD_DEL_USR" (Create Update Delete and Create Users)
default.synchronizer_reader	Controls the default source reader for the synchronizer page, default is "SOURCETYPE_FILE"
synchronizer.default_file_queuelog	Controls the default location for the queue_log file for the Synchronizer menu
synchronizer.default_file_agents	Controls the default location for the agents.conf file for the Synchronizer menu
synchronizer.default_file_queues	Controls the default location for the queues.conf file for the Synchronizer menu
synchronizer.default_file_users	Controls the default location for the users.conf file for the Synchronizer menu
synchronizer.url_source	Controls the URL for the URL source reader in the Synchronizer
synchronizer.default_password	This is the password that will be set as the default passwords for all users created through the synchronizer. If not specified all new passwords will be generated randomly. It is not specified by default.

Valid synchronizer modes are:

- CR_UPD_DEL_USR
- CR_UPD_DEL
- CR_DEL_USR
- CR_UPD_USR
- CR_UPD
- CREATEONLY_USR
- CR_DEL
- CREATEONLY
- DEL_UPD_USR
- DEL_UPD
- DELETEONLY_USR
- DELETEONLY
- UPDATEONLY_USR
- UPDATEONLY
- NONE

External auth sources

See External auth for more details.

Property name	Description
auth.externalSource	Leave empty if none. Valid values are json, ldap, xmlrpc.
auth.verboseLog	Whether to keep a verbose log of external auth
default.authSystemId	The system-ID for this QM license. Can be currently any user-chosen name for the system.
auth.jsonServerUrl	A server URL to query for JSON auth
auth.jsonPost	Whether to use POST (true) or GET (false). Default: false
auth.jsonIgnoreUsers	A set of users not to be checked on JSON. Comma-separated list of logins.
auth.ldapServerUrl	The LDAP URL, e.g. ldap://ldap.mycompany.com:389
auth.ldapBind	A LDAP binding used for log-on. Usually the DN or e-mail, but depends on your server.
auth.ldapLoginAttr	If the actual login is to be rewritten to a LDAP attribute,
auth.ldapBase	The base for searching LDAP users. Only used if `ldapLoginAttr`is set
auth.ldapQuery	A LDAP query to match the user logged on. Only used if `ldapLoginAttr`is set
auth.ldapIgnoreUsers	A set of users not to be checked on LDAP. Comma-separated list of logins.
auth.ldapFailureDelegates	Whether a failure in LDAP auth is delegated or forbidden.
default.authRpcServerUrl	If set, XML-RPC auth is used. Points to the URL of the auth server.

Single Sign-On (SSO)

See Using SSO for more details.

Property name	Description
sso.driver	One or more comma-separated SSO drivers
sso.placeholderClass	The class where new users will be created. Default: empty - no users are created
sso.verbose	SSO web calls are logged
sso.microsoft.loginMessage	Default: "Sign in with Microsoft"
sso.microsoft.loginIcon	The URL of custom icon
sso.microsoft.applicationId	The appication (client) ID
sso.microsoft.tenantId	The tenant (directory) ID

Property name	Description
sso.microsoft.password	The secret (password)
sso.microsoft.authUri	If the app is not multi-tenant, enter the OAuth 2.0 authorization endpoint. Default: "https://login.microsoftonline.com/common/oauth2/v2.0/authorize"
sso.google.loginMessage	Default: "Sign in with Google"
sso.google.loginIcon	The URL of custom icon
sso.google.clientId	The client ID
sso.google.clientSecret	The secret
sso.google.authUri	Default: "https://accounts.google.com/o/oauth2/auth"
sso.google.tokenUri	Default: "https://oauth2.googleapis.com/token"

Assisted manual outbound (AMO)

Property name	Description
amo.placedCallTimeout	The minimum time after a placed call will be evalued for recycling

Obsolete parameters

Property name	Description
default.showLostCallsWhenFiltering	If true, lost calls are shown when running a report in filter mode. This is usually *false*, as all lost calls would be shown even if you run a report for a single agent.
	Removed in version 1.5.1
dbmaint.agaw_oldestRun	Oldest obsolete run to keep when running a database optimization, in minutes
dbmaint.agaw_oldestLog	Oldest obsolete log to keep when running an optimization, in minutes
dbmaint.agaw_oldestBroadcast	Oldest obsolete broadcast entries to keep when running an optimization, in minutes
realtime.agent.show_agaw	Enable or disable the AGAW subset information table present in the agent's realtime page.
default.subqueueModeEnabled	If set to true, all activities on subqueues are reported in the parent queue. Default: false
realtime.all_subqueues	Enable default showing of all subqueues if set to 1
realtime.all_subqueues.buttonEnabled	
script.reboot	The command to restart Tomcat. Must be set if this is wanted.

An example configuration

This is a commented example configuration that contains most properties so you can see how they are supposed to be used in context.

```
# This is the canonical URL for this QM installation - please set
# If unset, it will be detected.
url.qm=

# This is an ALIAS to the default URL for stateless sessions. Some features (AGAW,
tasks) may not work if you don't set this.
url.rss=

# This is the internal id (ex. 7, 49....) of the preselected default queue, leave
blank for no default queue.
default.queue=1

# This is the default queue log file.
default.queue_log_file=sql:P001

# When the realtime page for an agent is computed, the queue_log is NOT read in its
entirety but only the last 'n' bytes.
realtime.max_bytes_agent=65000

# When set to true, the live agent page will open a window each time the URL
associated to the first row of call detail table changes.
realtime.agent_autoopenurl=true

# Defines how many queues the agent will select when logging on through
AgentCallBackLogin
# Valid values are: all, registered, assigned
realtime.dynamicLoginQueues=all

# The top level directory where monitored calls are held.
# Do NOT forget to add the ending slash.
default.monitored_calls=/var/spool/asterisk/

# How many digits to consider as a default area code.
default.areacode_digits=3

# Preset start and end hours and number of days for the custom report.
default.start_hour=8

default.end_hour=18

default.days=7

# If liveclock is enabled, the system clock is synchronized with asterisk server
system clock.
liveclock.enable=true
```

```
# Show summary report call time in hhmmss format instead of hours.
default.showSecondsOnTotalCalls=false

# The max delay that will be shown in the TOS graph.
sla.max_monitored_delay=120

# How big are time intervals for the TOS graph.
sla.interval=10

# It is possible to have a different definition, for the initial part of the SLA
sla.max_initial_delay=20

# The interval for the initial part of the SLA
sla.initial_interval=5

# Time margin (expressed in hours) to include different attempts in the same cluster
default.groupingMargin=168

# Your company logo (full/relative path, or https URL) - Recommended size 90x20px
layout.logo=$WEBAPP/img/sm_your_logo.png

# HTML string displayed on the login page.
layout.splash=Please ask your system administrator for the correct credentials to
access this instance of QueueMetrics.

# 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_data5=data5
sqlPreset.1.f_incr=unique_row_count

sqlPreset.2.table=queue_log
sqlPreset.2.f_time_id=time_id
sqlPreset.2.f_call_id=call_id
sqlPreset.2.f_queue=queue
sqlPreset.2.f_agent=agent
sqlPreset.2.f_verb=verb
sqlPreset.2.f_partition=
sqlPreset.2.f_data1=data1
sqlPreset.2.f_data2=data2
sqlPreset.2.f_data3=data3
```

```
sqlPreset.2.f_data4=data4
sqlPreset.2.f_data5=data5
sqlPreset.2.f_incr=unique_row_count

#
# Asterisk-realtime for Asterisk 1.8
#
sqlPreset.a18.table=asteriskdb.queue_log
sqlPreset.a18.f_time_id=time
sqlPreset.a18.use_timestamp=false
sqlPreset.a18.f_call_id=callid
sqlPreset.a18.f_queue=queuename
sqlPreset.a18.f_agent=agent
sqlPreset.a18.f_verb=event
sqlPreset.a18.f_partition=
sqlPreset.a18.f_data1=data1
sqlPreset.a18.f_data2=data2
sqlPreset.a18.f_data3=data3
sqlPreset.a18.f_data4=data4
sqlPreset.a18.f_data5=data5
sqlPreset.a18.f_incr=id

#
# Asterisk-realtime for Asterisk 1.6
#
sqlPreset.a16.table=asteriskdb.queue_log
sqlPreset.a16.f_time_id=time
sqlPreset.a16.use_timestamp=true
sqlPreset.a16.f_call_id=callid
sqlPreset.a16.f_queue=queuename
sqlPreset.a16.f_agent=agent
sqlPreset.a16.f_verb=event
sqlPreset.a16.f_partition=
sqlPreset.a16.f_data1=data
sqlPreset.a16.f_data2=
sqlPreset.a16.f_data3=
sqlPreset.a16.f_data4=
sqlPreset.a16.f_data5=
sqlPreset.a16.f_incr=id


# Is the calls panel in the Realtime page invisible by default?
realtime.calls_invisible=false

# Is the agents panel in the realtime page invisible by default?
realtime.agents_invisible=false

# Show on the Realtime page, only the Agents who are "known" for the queue.
realtime.members_only=false

# How frequently the Realtime page will be refreshed (in seconds).
```

```
realtime.refresh_time=5

# If call-back agents should be considered still logged on after a system reload.
default.permanentCallBackAgents=false

# If incomplete entities (calls or agent sessions, that are in progress at the moment
the analysis is being run) should be counted in the reports or not. (default: true)
default.considerIncompletetEntities=true

# The call-file directory Asterisk uses to generate calls based on .call files, set to
blank for no call files at all or use TCP AMI connection
callfile.dir=tcp:admin:amp123@127.0.0.1

# Softphone parameters
default.sipaddress=127.0.0.1
default.websocketurl=ws://127.0.0.1:8088/ws
default.rtcWebBreaker=true

callfile.monitoring.enabled=true
callfile.monitoring.channel=Local/$EM@from-internal/n
callfile.monitoring.extension=11
callfile.monitoring.context=queuemetrics

callfile.outmonitoring.enabled=true
callfile.outmonitoring.channel=Local/$EM@from-internal/n
callfile.outmonitoring.extension=14
callfile.outmonitoring.context=queuemetrics

callfile.calloutcome.enabled=true
callfile.calloutcome.channel=Local/12@queuemetrics/n
callfile.calloutcome.extension=10
callfile.calloutcome.context=queuemetrics

callfile.addfeature.enabled=true
callfile.addfeature.channel=Local/16@queuemetrics/n
callfile.addfeature.extension=10
callfile.addfeature.context=queuemetrics

callfile.removefeature.enabled=true
callfile.removefeature.channel=Local/17@queuemetrics/n
callfile.removefeature.extension=10
callfile.removefeature.context=queuemetrics

callfile.agentlogin.enabled=true
callfile.agentlogin.channel=Local/20@queuemetrics/n
callfile.agentlogin.extension=10
callfile.agentlogin.context=queuemetrics

callfile.agentlogoff.enabled=true
callfile.agentlogoff.channel=Local/21@queuemetrics/n
callfile.agentlogoff.extension=10
```

```
callfile.agentlogoff.context=queuemetrics

callfile.agentpause.enabled=true
callfile.agentpause.channel=Local/22@queuemetrics/n
callfile.agentpause.extension=10
callfile.agentpause.context=queuemetrics

callfile.agentpause_ht.enabled=true
callfile.agentpause_ht.channel=Local/32@queuemetrics/n
callfile.agentpause_ht.extension=10
callfile.agentpause_ht.context=queuemetrics

callfile.agentunpause.enabled=true
callfile.agentunpause.channel=Local/23@queuemetrics/n
callfile.agentunpause.extension=10
callfile.agentunpause.context=queuemetrics

callfile.agentunpause_ht.enabled=true
callfile.agentunpause_ht.channel=Local/33@queuemetrics/n
callfile.agentunpause_ht.extension=10
callfile.agentunpause_ht.context=queuemetrics

callfile.agentaddmember.enabled=true
callfile.agentaddmember.channel=Local/25@queuemetrics/n
callfile.agentaddmember.extension=10
callfile.agentaddmember.context=queuemetrics

callfile.agentaddmember_ht.enabled=true
callfile.agentaddmember_ht.channel=Local/35@queuemetrics/n
callfile.agentaddmember_ht.extension=10
callfile.agentaddmember_ht.context=queuemetrics

callfile.agentremovemember.enabled=true
callfile.agentremovemember.channel=Local/27@queuemetrics/n
callfile.agentremovemember.extension=10
callfile.agentremovemember.context=queuemetrics

callfile.agentremovemember_ht.enabled=true
callfile.agentremovemember_ht.channel=Local/37@queuemetrics/n
callfile.agentremovemember_ht.extension=10
callfile.agentremovemember_ht.context=queuemetrics

callfile.writevar.enabled=true
callfile.writevar.channel=Local/38@queuemetrics/n
callfile.writevar.extension=10
callfile.writevar.context=queuemetrics

callfile.customdial.enabled=true
callfile.customdial.channel=Local/$EM@from-internal/n
callfile.customdial.extension=28
callfile.customdial.context=queuemetrics
```

```
callfile.smssend.enabled=false
callfile.smssend.channel=Local/10@queuemetrics/n
callfile.smssend.extension=29
callfile.smssend.context=queuemetrics

callfile.hangupcall.enabled=true
callfile.hangupcall.channel=Local/10@queuemetrics/n
callfile.hangupcall.extension=30
callfile.hangupcall.context=queuemetrics

callfile.transfercall.enabled=true
callfile.transfercall.channel=Local/10@queuemetrics/n
callfile.transfercall.extension=31
callfile.transfercall.context=queuemetrics

# Rewrites Local/XXX@ext channels in the format Agent/XXX
default.rewriteLocalChannels=true

# DO NOT CHANGE
realtime.use_sql_now=false

# If an ongoing call has a wait time that exceeds this value (in seconds), drop it.
(0: ignore)
default.maxOngoingWaitTime=7200

# If an ongoing call has a talk time that exceeds this value (in seconds), drop it.
(0: ignore)
default.maxOngoingTalkTime=7200

# Ignore QUEUESTART statements.
default.ignoreQueueStarts=false

# The starting hour of the day, in order to compute the realtime report. It can be
either a fixed hour (e.g. 3: from the last 3:00 AM) or a sliding window if prefixed
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.startHour=0

# The default locale for QueueMetrics. DO NOT CHANGE unless you know what you're
doing.
default.language=en
default.country=US

# If QueueMetrics should always show the default system language (default: false).
default.forceDefaultLanguage=false

# Disable the history back navigation button in browsers.
default.disablebackhistory=true

# Which channel name to show for a call: True: last one seen - False: first one found.
```

```
default.useEndingChannelName=false

# How to create Excel files: True: Excel 2003 XML - False: CSV.
default.useXmlExcel=true

# If true, show the queue composition; if fals, show just the main queue name.
default.showQueueComposition=true

# How long is an hour for hourly breakdown; default 60 minutes.
default.hourly_slot=15

# If true, anything after the "-" sign is deleted (ie. SIP/203-abcd is read as
SIP/203). If false, the agent channel name is loaded as in the queue_log file.
(default: true)
default.stripChannelNames=true

# Asterisk Unique ID visibility and related callback URL associated to the icon.
default.crmapp=https://www.queuemetrics.com/sample_screen_pop.jsp?agent=[A]&unique=[U]

# The name of the CRM application in the Icon Agent page.
default.crmlabel=CRM

# If true opens the url specified in default.crmapp in a pop-up instead of the
background Iframe.
agent.openUrlInPopup = false

# How much time (in seconds) an agent has to set an outcome code, to the call, after
the call has finished.
default.callStatusTimeout = 1800

# Show agent priority on queues in agent logon panel.
default.show_agent_priority=true

# Custom web pages for the realtime agent page.
realtime.agent_web1_url=https://www.queuemetrics.com/sample_screen_pop.jsp?agent=[A]
realtime.agent_web1_label=
realtime.agent_web2_url=https://www.queuemetrics.com/sample_screen_pop.jsp?agent=[A]
realtime.agent_web2_label=

# If true, it will enable the Asterisk unique ID column on the answered/unanswered
call details tables.
default.showAstClid=false

# Join multi-stint calls globally.
default.joinMultiStintCalls=false

# Decide whether to check for alarms on the wait time of ongoing conversations.
realtime.waitAlarmOnLiveCalls=false

# Sound for yellow alarm on real-time page - If you want this played, you could put
../img/sounds/dingdong.wav
```

```
sound.yellowAlarm=../img/sounds/silence.wav

# Sound for red alarm on real-time page - If you want this played, you could put
../img/sounds/ringer.wav
sound.redAlarm=../img/sounds/silence.wav

# Cluster to poll for Realtime data.
# List of pipe-separated members of the cluster.
# If callfilesdir = "", the commands will be sent through the manager API.
cluster.servers=aleph|trix

cluster.aleph.manager=tcp:dial:12345@10.10.3.5
cluster.aleph.queuelog=sql:P001
cluster.aleph.monitored_calls=/mnt/qm_streamcall/server_aleph
cluster.aleph.callfilesdir=
cluster.aleph.audioRpcServer=
cluster.aleph.agentSecurityKey=
cluster.aleph.websocketurl=ws://10.10.3.5:8088/ws
cluster.aleph.rtcWebBreaker=false
cluster.aleph.sipaddress=

cluster.trix.manager=tcp:admin:amp111@127.0.0.1
cluster.trix.queuelog=sql:P002
cluster.trix.monitored_calls=/mnt/qm_streamcall/server_trix
cluster.trix.callfilesdir=
cluster.trix.audioRpcServer=
cluster.trix.agentSecurityKey=
cluster.trix.websocketurl=ws://127.0.0.1:8088/ws
cluster.trix.rtcWebBreaker=false
cluster.trix.sipaddress=

# If true, hide export buttons on the Realtime page.
realtime.hideExportButtons=true

# If true, show all agent sessions. If false, show only agent sessions with at least
one call.
default.useRawAgentSessions=false

# If true, the agent cannot change their code in the login/logoff/pause popups.
default.lockedAgentPopupCode=true

# If true, or missing, the agent cannot change their code and extension if logged on
at least one queue.
realtime.preventLoggedAgentInfoChanges=true

# If true, the user will not be able to monitor without a given location.
realtime.assignedLocationsOnly=false

# If true, the start of the current pause is shown as an absolute hour; if false, it
is shown as the time passed since.
realtime.absolutePauseTimes=true
```

```
# Which buttons on the realtime page are enabled?
realtime.calls_invisible.buttonEnabled=true
realtime.agents_invisible.buttonEnabled=true
realtime.members_only.buttonEnabled=true
realtime.aggr_by_queue_invisible.buttonEnabled=true
realtime.aggr_by_tag_invisible.buttonEnabled=true
realtime.agent_and_outcome_invisible.buttonEnabled=true

# Create a text dump of the manager conversation for the Live page (#326)
manager.dump=false

# If a filter is on, no "lost" calls should be shown.
default.showLostCallsWhenFiltering=false

# The default length of a manually-closed call.
default.closeDuration=5

# This is the name of the class that finds recordings. See documentation for more
details.
audio.server=it.loway.app.queuemetrics.callListen.listeners.LocalFilesByDay

# This is the name of the class that starts live listening. See documentation for more
details.
audio.liveserver=it.loway.app.queuemetrics.callListen.RTlisteners.ClassicQMListenerRT

# Is it possible to see Tech Info on the licence page and run DBTest?
default.viewTechInfo=true

# Defines some default values for the autoconfiguration tool.
default.autoconf.source=file
default.autoconf.fileagents=/etc/asterisk/agents.conf
default.autoconf.filequeues=/etc/asterisk/queues.conf
default.autoconf.fileusers=/etc/asterisk/users.conf
default.autoconf.filequeuelog=/var/log/asterisk/queue_log
default.autoconf.realtimedrv=com.mysql.jdbc.Driver
default.autoconf.realtimeuri=jdbc:mysql://127.0.0.1/asteriskrealtime?zeroDateTimeBehav
ior=convertToNull&jdbcCompliantTruncation=false&user=admin&password=amp111

# If set to true, when an agent logs on, he will always be unpaused.
default.alwaysLogonUnpaused=true

# If set to something different than 0, enables hotdesking mode.
# The value is interval time (in seconds) used by the analyzer to look back searching
HOTDESK verbs in the queue log
default.hotdesking=86400

# If set to true, Agents are required to provide a valid pause reason when inserting a
pause from their Agent Page.
default.pausecoderequired=false
```

```
# The script to use in order to restart Tomcat.
script.reboot=/usr/local/queuemetrics/qm-current/WEB-INF/mysql-
utils/restart/restart.sh

# Which days to include (1: Sun 2:Mon; e.g. 23456 means MON to FRI).
default.noncontig.days=

# Start and end times (as HH:mm:ss) for noncontiguous reports
default.noncontig.period1.start=
default.noncontig.period1.end=
default.noncontig.period2.start=
default.noncontig.period2.end=

# Number of digits to display for floating-point numbers.
default.decimalDigits=1

# Task Ping URL for any approved/disputed operation.
default.tasks.pingURL=

# Defines if the QA reports should be calculated by call date or by filling form date.
default.searchQA_byCallDate=true

# A regex to validate agent codes (e.g. "1\\d\\d" to allow all three-digit numbers
starting with 1).
realtime.agentRegexp=.+

# A regex to validate extension codes (e.g. "\\d\\d" to allow all two-digit numbers).
realtime.extensionRegexp=.+

# Maximum period allowed for a Self Service Agent Report.
default.ssarMaxReportPeriod=15

# QueueMetrics mailer configuration (leave smtphost key empty to disable the mailer)
default.smtphost=
default.smtpport=587
default.smtpfrom=from.mail@example.com
default.smtpuser=from.user@example.com
default.smtpassword=verysecret
default.smtpssl=false
default.smtpDebug=false
# SMTP usage quotas
default.smtpDailyQuota=200
default.smtpDailyUserQuota=50


# PDF direct exported reports will use this orientation as default.
default.export.orientation=portrait

# If true, the queues where the agent is working and is a member of, are displayed in
black; queues where the agent is working on but is not a member, of are displayed in
brown; any queues the agent is a known member of, but is not currently logged on to,
```

```
are displayed as a tooltip to the gray "Down Arrow'' symbol. (default false).
realtime.show_incorrect_queue_sets=false

# If true, when logging off from ALL queues, any queue the agent is currently logged
on will be disconnected. (default false)
realtime.all_includes_current_logins=false

# The Realtime page uses Active Polling.
realtime.useActivePolling=true

# The Agent page polls every X seconds (DO NOT set too low or server load will go up).
realtime.ajaxPollingDelay=5

# Use RowCache to avoid hitting the DB on each request.
realtime.useRowCache=true

# Before logging an agent in (in hotdesking mode) makes sure the extension is not in
use.
realtime.assertExtensionNotInUse=true

# When an agent logs in, pause them with the "autopauseCodeLogin" code.
realtime.agentPausedOnLogin=false

# The code to use for autopauses on login.
realtime.autopauseCodeLogin=

# Do we want to use the HTML5 audio player?
audio.html5player=true

#
# DIRECT AMI configuration
#

platform.pbx=DIRECTAMI
platform.directami.agent=Agent/${num}
platform.directami.extension=PJSIP/${num}
platform.directami.transfer=${num}@from-internal
platform.directami.outbound.enabled=true
platform.directami.localext=PJSIP/${num}
platform.directami.verbose=true

# Encrypted password are (1) not required for log in (2) required for log-in.
pwd.minAllowedLevel=1

# When  updated, passwords are saved (1) plain-text (2) encrypted.
pwd.defaultLevel=1

# Default user for the Wallboard Kiosk URL.
default.wallboard_kiosk_user=robot

# Short calls; which calls are excluded when unanswered and wait time is less than X
```

```
seconds.
default.shortcalls_wait=-1

# Short calls;  which calls are excluded when answered and talk time is less than X
seconds.
default.shortcalls_talk=-1

# Short calls - which attempts are excluded if less than X seconds.
default.shortcalls_attempts=-1

# Default theme.
default.selectedTheme=default

# Reports that take longer end up on the Audit log.
audit.maxReportTime=30000

# Reports that scan more lines end up on the Audit log.
audit.maxRowsFetched=1000000

# reports that are longer than these many days cannot be run. 0: No limit.
report.maxAllowedDays=0

# a set of exceptions to maxAllowedDays with the format "KEY:N_DAYS,KEYS:N_DAYS"
report.maxAllowedDaysExceptions=

# a hint message that will be added to the error if there is a maxAllowedDays
violation
report.maxAllowedDaysHint=

# Whether known numbers (eg VIP) should be used to decode caller numbers
default.decodeKnownNumbers=true

# The number hiding policy for agents having the key HIDE_NUMBERS
default.hiddenNumberPolicy=hash

# The time-out after ist beginning that an AMo call will be recycled.
amo.placedCallTimeout=3600

# When sending an email, a new subeject instead of [QueueMetrics]
default.smtpSubjectPrefix=

# Additional text to be added to the email containing exported reports
default.smtpExtraBody=
```

Additional queue_log logs for QueueMetrics

In addition to the standard **queue_log** logs generated by Asterisk, QueueMetrics is able to parse a number of other *verbs* in order to enable advanced tracking features. Some of these additional logs are already generated by QueueMetrics itself or Uniloader although you can freely implement them in your own dialplans.

Basic queue_log structure

The **queue_log** file is normally located under */var/log/asterisk/*. Each line in this file is to be considered as an event that happened in the queues. The fields are separated by a pipe sign; here an example line:

```
10000005|214530.123|queue|Agent/123|CONNECT|4
```

The fields are, in order: *timestamp* of the event, *unique id* of the call, *queue name*, *agent code*, *verb name*, and up to 5 additional data fields whose usage depends on the verb.

You can add custom lines in **queue_log** by calling Asterisk's `QueueLog()` function.

Updating past calls

QueueMetrics usually allows modifying calls with custom values, e.g. setting feature codes, outcomes or variables - only:

- when the call is active, or
- up to a specified point in time after the call completes.

This value can be changed. The default is 1800 seconds (30 minutes) after the call is ended, but it can be edited by changing the value of `default.callStatusTimeout` in the configuration properties through the "Edit system parameters" page in QueueMetrics.

Having a longer period has two important issues:

- it is more likely that such data falls outside the given analysis window, and therefore may not appear correctly in some reports, and
- it may add a significant amount of load to the ongoing analysis.

Music-On-Hold Tracking

The MOH events must be tracked during the call (between the CONNECT and the COMPLETEAGENT or COMPLETECALLER events); the CALLERONHOLD event is written when the call is put on hold, CALLEROFFHOLD when the call is resumed. These events are currently tracked by Uniloader, enabling the Unitracker feature.

```
10000001|214530.A|queue|NONE|ENTERQUEUE||123456
```

```
10000005|214530.A|queue|Agent/302|CONNECT|4
10000015|214530.A|queue|NONE|CALLERONHOLD
10000025|214530.A|queue|NONE|CALLEROFFHOLD
10000042|214530.A|queue|Agent/302|COMPLETEAGENT|4|37
```

Tracking IVR

IVR events must be tracked before the beginning of the call; here's an example of IVR tracking with the IVRSTART and IVRAPPEND.

```
10000000|214530.A|ivr|NONE|INFO|IVRSTART|caller|dnis
10000010|214530.A|ivr|NONE|INFO|IVRAPPEND|X|menu-1
10000020|214530.A|queue|NONE|ENTERQUEUE||123456
10000030|214530.A|queue|Agent/302|CONNECT|10
10000040|214530.A|queue|Agent/302|COMPLETEAGENT|10|10
```

As you usually cannot know on which queue the IVR will end (otherwise you would not need an IVR...) we suggest logging on a figurative queue called `ivr`. This way, when you run a report only for your queue, you do not get IVR events; if you run a report for both `queue` and `ivr` at once, IVR events are added to the queue calls.



You could log everything under `ivr`, and use the menus to distinguish them, or you could have multiple distinct "queues", e.g. one per DID.

Available verbs are:

- `INFO IVRSTART`: logs the caller and DNIS. If you just need to log the DNIS for a non-IVR call, see Adding DID/DNIS information. You could also do both.
- `INFO IVRAPPEND`: user presses a key on the IVR tree
- `INFO IVRHANGUP`: call terminated during IVR
- `INFO IVRGOAL`: goal reached during IVR

Whenever possible, it is advisable to specify a menu for the IVR and avoid using `NONE` in the queue field.



You can find a detailed explanation of all verbs at Implementing IVR.

Adding DID/DNIS information

To track the number that the caller dialled (usually named DID or DNIS), you add a record like:

```
1020000|2.1|NONE|NONE|INFO|DID|1234
```

If you know that the DID leads to one fixed destination queue, you can add it so searching will be

more efficient:

```
1020000|2.1|queue|NONE|INFO|DID|1234
```

Adding Tags to a call

You can add a tag to a call (see Adding TAG information to calls), before, during and after the call itself, keeping in mind that if you run a report for a certain period of time only events that happened in the report's time window are shown. See Updating past calls.

A call can have only one tag, if multiple tags are added to the logs only the last one will be used.

Tags are added using the verb *INFO*. Here's an example of tag added to the call after a few IVR choices:

```
1020000|2.1|NONE|NONE|INFO|DID|1234
1020000|2.1|NONE|NONE|INFO|IVRAPPEND|1|i1
1020000|2.1|NONE|NONE|INFO|IVRAPPEND|3|i1
1020001|2.1|NONE|NONE|INFO|TAG|T2
```

A tag can be used in your CRM integration URL as follows: `http://mycrm/?tag=[TAG]`

Refer to this page for additional details about this feature: CRM Integration

Adding Feature codes to a call

You can add a feature code to a call (Adding TAG information to calls), before, during and after the call itself, keeping in mind that if you run a report for a certain period of time only events that happened in the report's time window are shown. See also Updating past calls.

Feature codes are added using the verb *INFO*. Here's an example of feature code added to the call after a few IVR choices; you can also see how a feature code can accept an additional free text field:

```
1020000|2.1|NONE|NONE|INFO|DID|1234
1020000|2.1|NONE|NONE|INFO|IVRAPPEND|1|i1
1020000|2.1|NONE|NONE|INFO|IVRAPPEND|3|i1
1020001|2.1|NONE|NONE|INFO|FTR|X1
1020001|2.1|NONE|NONE|INFO|FTR|X2|Free text field
```

Existing feature codes can also be removed. QueueMetrics will report the complete history of added and removed feature codes in the call's detail:

```
1080003|8.1|NONE|NONE|INFO|NOFTR|X1
1080003|8.1|NONE|NONE|INFO|NOFTR|X2
```

A feature code can be used in your CRM integration URL as follows: `http://mycrm/?x1=[F:X1]`

Or you can just check if a feature code exists (outputs `1`) or not (outputs `0`): `http://mycrm/?x1exist=[FX:X1]`

Refer to this page for additional details about this feature: CRM Integration



You can find a detailed explanation of all verbs at Enable call features.

Adding custom variables to a call

You can add custom variables to a call, before, during and after the call itself, keeping in mind that if you run a report for a certain period of time only events that happened in the report's time window are shown. See Updating past calls.

Variables are added using the verb *INFO*. Here's an example of variable added to the call:

```
1030000|3.1|q1|NONE|ENTERQUEUE||01202579573|1
1030001|3.1|q1|NONE|INFO|VAR|MY_VAR_NAME|MY_VAR_VALUE
1030002|3.1|q1|SIP/5435|CONNECT|2|1411721579.4098058|4
```

A variable can be used in your CRM integration URL as follows: `http://mycrm/?myvar=[V:MY_VAR_NAME]`. Refer to this page for additional details about this feature: CRM Integration

If the same variable is present multiple times with different attributes, the last one wins.



Both variable names and values are always lowercased by QueueMetrics. If you need to have an uppercase value, we suggest URL-encoding it. E.g. "Loway" → "%4coway"

Known variables

While in general you can use any variable and set it to any value, there are some variables that have a "magic" meaning to QueueMetrics, as detailed below.

Variable	Description
`QM_CASEID`	A reference to the "case id" of a case
`QM_CUSTID`	A reference to a specific customer

Tracking Outbound calls

If you want to track outbound calls you should create fake `queue_log` records as if the call was actually made in a queue. An outbound call is introduced by the CALLOUTBOUND verb. The called number will be tracked in the second additional field after the verb name (the first additional field refers to the call's URL). These events are currently tracked by Uniloader, enabling the Unitracker

feature.

This is a valid outbound call:

```
1092000000|1092214530.A|sk-fissi|Agent/302|CALLOUTBOUND||12345679
1092000005|1092214530.A|sk-fissi|Agent/302|CONNECT|5
1092000042|1092214530.A|sk-fissi|Agent/302|COMPLETEAGENT|5|37
```

Tracking pause reasons

QueueMetrics lets you discern between the different kinds of pause of your agents, in order to track also payable or billable activities. Pause codes are defined in the "Edit Pause codes" page (Configuring pause codes).

Using the PAUSEREASON verb immediately after the PAUSEALL verb you can add a pause code. In case of multiple PAUSEREASON verbs for the same pause only the most recent will be taken into consideration:

```
1400156431|x1|9998|SIP/99881|ADDMEMBER|
1400156432|NONE|NONE|SIP/99881|PAUSEALL|
1400156432|1400156430.782045|NONE|SIP/99881|PAUSEREASON|99
1400156432|NONE|NONE|SIP/99881|UNPAUSEALL|
1400156432|NONE|9988|SIP/99881|UNPAUSE|
1400156443|x2|9998|SIP/99881|REMOVEMEMBER|
```



You can find a detailed explanation at Enabling Pause Codes.

Adding the call status (outcome)

The call status will help you track the results of your calls. A call status can be added after the beginning of the call, and after the end of the call keeping in mind that if you run a report for a certain period of time only events that happened in the report's time window are shown. See Updating past calls.

In case of multiple call statuses only the last one is taken into consideration.

The call statuses are defined in the "Edit call outcomes" page (Configuring call outcomes)

```
10000000|1142264431.21|queue-dps|NONE|ENTERQUEUE||34
10000030|1142264431.21|queue-dps|Agent/102|CONNECT|30
10000040|1142264431.21|queue-dps|Agent/102|COMPLETEAGENT|30|10
10000050|1142264431.21|NONE|Agent/102|CALLSTATUS|30
```



You can find a detailed explanation at Enabling call outcomes.

Tracking Hotdesking events

Tracking the hotdesking events (Enabling Hotdesking in the agent page) just before the queue login events is very useful if you are using dynamic extensions in your call center. This will help QueueMetrics tracking the association between the agent code and the extension number used by that agent at any given moment:

```
100000|10101010.11|NONE|Agent/102|HOTDESK|SIP/1235
```

Tracking Skills

In skill-based routing systems you can track the requested skills with the REQSKILL verb. (Configuring skills). These events are valid before the call is connected to the agent.

```
100000|1167335958.530054|cpc|NONE|ENTERQUEUE||01909723335
100001|1167335958.530054|cpc|NONE|INFO|REQSKILL|DE|90
100002|1167335958.530054|cpc|NONE|INFO|REQSKILL|NET|70
100041|1167335958.530054|cpc|Agent/101|CONNECT|41|Agent/101
100064|1167335958.530054|cpc|Agent/101|COMPLETECALLER|41|23
```



You can find a detailed explanation of all verbs at Tracking required skills.

Appendix E: Audit log records

The following details are logged for all events:

- *Date* and time of the event
- *User-id* that is requesting/causing the event; if this is not applicable, a *0* may be logged instead.
- *Container session ID* - useful for tracking multiple activities done on the same user session and for further cross-matching with system logs.
- An *action* and zero or more parameters, as detailed below.

The audit table should be secured as needed by the system administrator by revoking the DELETE grants from it by the QueueMetrics database users.

Action class: User lifecycle (10XX)

Action: user logon - successful

- *Action-id*: 1001
- *Text1*: The full login, as a string, of the user logging on
- *Text2*: The IP address (dotted quad) of the user's workstation
- *Text3*: The login that was actually entered by the user, if different from current login. This may differ e.g. if a user logs on with their e-mail, or if a LDAP/JSON driver rewrites the source log-in to a QM login.

Action: user logoff

- *Action-id*: 1002
- *Text1*: The full login, as a string, of the user logging off
- *Text2*: The IP address (dotted quad) of the user's workstation



This event tracks only manual logoffs. Other causes of disconnection (e.g.. the user closes his browser, session timeouts, etc) are not tracked. Therefore you cannot count on having a logoff event for each logon event.

Action: user logon - unsuccessful

- *Action-id*: 1003
- *Text1*: The full login, as a string, of the user that tried to log on
- *Text2*: The IP address (dotted quad) of the user's workstation
- *Text3*: The error message displayed

Action: password change

- *Action-id*: 1004
- *Text1*: The full login, as a string, of the user logging on
- *Text2*: The IP address (dotted quad) of the user's workstation



A password change that you find logged may either have been initiated by a user on their own, or might have been forced by the administrator through a special link - see section Password reset.

SSO: OAuth consent granted

- *Action-id*: 1005
- *Text1*: The email of OAuth user
- *Text2*: The IP address (dotted quad) of the user's workstation
- *Text3*: The SSO driver providing authentication

This record is usually followed by a proper Login record.

SSO: OAuth consent failed

- *Action-id*: 1006
- *Text1*: The full text of the error (if available)
- *Text2*: The IP address (dotted quad) of the user's workstation
- *Text3*: The SSO driver providing authentication

SSO: OAuth profile failed

- *Action-id*: 1007
- *Text1*: The full text of the error (if available)
- *Text2*: The IP address (dotted quad) of the user's workstation
- *Text3*: The SSO driver providing authentication

SSO: User locked

- *Action-id*: 1008
- *Text1*: The email of OAuth user
- *Text2*: The IP address (dotted quad) of the user's workstation
- *Text3*: The SSO driver providing authentication

Current user appears to be locked, so no login is possible.

SSO: User not enabled

- *Action-id*: 1009
- *Text1*: The email of OAuth user
- *Text2*: The IP address (dotted quad) of the user's workstation
- *Text3*: The SSO driver providing authentication

Current user does not have a token matching the returned SSO profile.

SSO: New user created

- *Action-id*: 1010
- *Text1*: The email of OAuth user
- *Text2*: The IP address (dotted quad) of the user's workstation
- *Text3*: The SSO driver providing authentication
- *Text3*: The remote ID on the SSO system, to be used for enabling the user.

Action class: Key management (11XX)

Action: key changed

- *Action-id*: 1101
- *Text1*: The full login, as a string, of the user logging on
- *Text2*: The IP address (dotted quad) of the user's workstation
- *Text3*: The new key that was installed

Action: key accessed via XML-RPC

- *Action-id*: 1102
- *Text1*: The full login, as a string, of the user logging on
- *Text2*: The IP address (dotted quad) of the user's workstation
- *Text3*: The key that was passed (it may be blank if it was just a query)

Action: AGAW key changed

- *Action-id*: 1103
- *Text1*: The full login, as a string, of the user logged on
- *Text2*: The IP address (dotted quad) of the user's workstation
- *Text3*: The new key that was installed

Action: AGAW key accessed via XML-RPC

- *Action-id*: 1104
- *Text1*: The full login, as a string, of the user logging on
- *Text2*: The IP address (dotted quad) of the user's workstation
- *Text3*: The key that was passed (it may be blank if it was just a query)

Action: AGAW restarted

This action is logged only when the AGAW runner is restarted from the web GUI.

- *Action-id*: 1105
- *Text1*: The full login, as a string, of the user logging on
- *Text2*: The IP address (dotted quad) of the user's workstation

Action class: QueueLog editing (20XX)

Action: QueueLog edited

- *Action-id*: 2001
- *Text1*: The full login, as a string, of the user logging on
- *Text2*: The IP address (dotted quad) of the user's workstation
- *Text3*: The new statement
- *Text4*: The SQL rollback statement.

This event is triggered by a change to the queue_log made by the Payroll module. A rollback SQL statement is supplied in case it is needed to revert the changes.

Action class: QA editing (21XX)

Action: QA form deleted

- *Action-id*: 2101
- *Text1*: The full login, as a string, of the user logging on
- *Text2*: The IP address (dotted quad) of the user's workstation
- *Text3*: The rollback SQL statement

Action: Deletion of a comment

- *Action-id*: 2102
- *Text1*: The full login, as a string, of the user logging on
- *Text2*: The IP address (dotted quad) of the user's workstation

- *Text3*: Which comment was deleted

Action: Deletion of all comments

- *Action-id*: 2103
- *Text1*: The full login, as a string, of the user logging on
- *Text2*: The IP address (dotted quad) of the user's workstation
- *Text3*: Which call was involved

Action class: Recordings (22XX)

Action: Media tag removed

- *Action-id*: 2201
- *Text1*: The tag id

Action: Media tag added

- *Action-id*: 2202
- *Text1*: Tag ID and call ID

Action: MediaProxy - Media played

- *Action-id*: 2203
- *Text1*: The URL played
- *Text2*: How many bytes were transferred and how long it took. Note that even if the file was trasferred in its entirety, we do not know whether it was listened to or not.

Action: MediaProxy - Media failed

- *Action-id*: 2204
- *Text1*: HTTP error code, request duration and URL
- *Text2*: The recording's randomId

Action: MediaProxy - Media denied

- *Action-id*: 2205
- *Text1*: Reason for denial
- *Text2*: The recording's randomId

Action class: Realtime agent management (23XX)

See also: Platform actions.

Action: Realtime Agent Logon

- *Action-id*: 2301
- *Text1*: The full login, as a string, of the user logging on
- *Text2*: The IP address (dotted quad) of the user's workstation
- *Text3*: AgentCode: XXX AgentExtension: XXX

Action: Realtime Agent Logoff

- *Action-id*: 2302
- *Text1*: The full login, as a string, of the user logging on
- *Text2*: The IP address (dotted quad) of the user's workstation
- *Text3*: AgentCode: XXX AgentExtension: XXX

Action: Realtime Agent Pause

- *Action-id*: 2303
- *Text1*: The full login, as a string, of the user logging on
- *Text2*: The IP address (dotted quad) of the user's workstation
- *Text3*: AgentCode: XXX AgentExtension: XXX

Action: Realtime Agent Unpause

- *Action-id*: 2304
- *Text1*: The full login, as a string, of the user logging on
- *Text2*: The IP address (dotted quad) of the user's workstation
- *Text3*: AgentCode: XXX AgentExtension: XXX

Action: Realtime Agent SMS

- *Action-id*: 2305
- *Text1*: The full login, as a string, of the user logging on
- *Text2*: The IP address (dotted quad) of the user's workstation
- *Text3*: AgentCode: XXX AgentExtension: XXX

Action class: Realtime call management (24XX)

Action: Call soft hangup

- *Action-id*: 2401
- *Text1*: The full login, as a string, of the user logging on

- *Text2*: The IP address (dotted quad) of the user's workstation
- *Text3*: AgentCode: XXX AgentExtension: XXX UniqueID: XXXXXXXXXX

Action: Call transfer

- *Action-id*: 2402
- *Text1*: The full login, as a string, of the user logging on
- *Text2*: The IP address (dotted quad) of the user's workstation
- *Text3*: AgentCode: XXX AgentExtension: XXX UniqueID: XXXXXXXXXX

Action: Call closure

- *Action-id*: 2403
- *Text1*: The full login, as a string, of the user logging on
- *Text2*: The IP address (dotted quad) of the user's workstation
- *Text3*: UniqueID: XXXXXXXXXX

Action class: Engine audits (25XX)

For all engine audits, there is a "description" field that details:

- The type of report requested: Historical report or Real-time page. JSON webservices run either of these.
- A description of the filtering parameters in JSON format.

Action: Too many rows scanned

- *Action-id*: 2501
- *Text1*: The number of queue_log rows scanned
- *Text2*: A description of the report (see above)

Action: Report taking too long

- *Action-id*: 2502
- *Text1*: The actual duration of computing this report
- *Text2*: A description of the report (see above)

Action: License error

- *Action-id*: 2503
- *Text1*: The number of agents you'd need to run tshis report
- *Text2*: A description of the report (see above)

Action: Out of memory error

- *Action-id*: 2504
- *Text1*: Blank
- *Text2*: A description of the report (see above)

Action: Exceeded maximum reporting period

- *Action-id*: 2509
- *Text1*: Requested duration, in days
- *Text2*: Maximum possible duration, in days

See Maximum reporting period for further details.

Action: Report created

- *Action-id*: 2505
- *Text1*: Id report
- *Text2*: Name

Action: Report processed

- *Action-id*: 2506
- *Text1*: Id report
- *Text2*: Name

Action: Report Saved

- *Action-id*: 2507
- *Text1*: Id report
- *Text2*: Name

Action: Report Deleted

- *Action-id*: 2508
- *Text1*: Id report
- *Text2*: Name

Action class: Platform actions (26XX)

See also: Realtime agent management.

Action: LOGIN

- *Action-id*: 2601
- *Text1*: agent
- *Text2*: extension

Action: LOGOFF

- *Action-id*: 2602
- *Text1*: agent

Action: ADDMEMBER

- *Action-id*: 2603
- *Text1*: agent
- *Text2*: extension
- *Text3*: queue(s)

Action: REMOVEMEMBER

- *Action-id*: 2604
- *Text1*: agent
- *Text2*: extension
- *Text3*: queue(s)

Action: PAUSE

- *Action-id*: 2605
- *Text1*: agent
- *Text2*: extension

Action: UNPAUSE

- *Action-id*: 2606
- *Text1*: agent
- *Text2*: extension

Action: OUTCOME

- *Action-id*: 2607
- *Text1*: call-id
- *Text2*: outcome

Action: ADD FEATURE CODE

- *Action-id*: 2608
- *Text1*: callid
- *Text2*: -
- *Text3*: feature

Action: REMOVE FEATURE CODE

- *Action-id*: 2609
- *Text1*: callid
- *Text2*: -
- *Text3*: feature

Action: CUSTOM DIAL

- *Action-id*: 2610
- *Text1*: extension
- *Text2*: target extension
- *Text3*: queue-id (campaign)

Action: SEND SMS

- *Action-id*: 2611
- *Text1*: extension
- *Text2*: target extension
- *Text3*: message

Action: HANGUP

- *Action-id*: 2612
- *Text1*: call-id
- *Text2*: agent

Action: TRANSFER

- *Action-id*: 2613
- *Text1*: call-id
- *Text2*: destination extension
- *Text3*: agent

Action: MONITOR INBOUND CALL

- *Action-id*: 2614
- *Text1*: call-id
- *Text2*: extension to monitor
- *Text3*: Mode and queue

Action: MONITOR OUTBOUND CALL

- *Action-id*: 2615
- *Text1*: call-id
- *Text2*: extension to monitor
- *Text3*: Mode and queue

Action: SET VARIABLE

- *Action-id*: 2616
- *Text1*: call-id
- *Text2*: variable = value

Action class: Cron jobs (30XX)

Action: Cronjob dispatcher started

- *Action-id*: 3001

This happens each and every time QM restarts.

Action: Cronjob run successfully

- *Action-id*: 3002
- *Text1*: Job ID and name
- *Text2*: Job status
- *Text3*: Duration of the job

Action: Cronjob error

- *Action-id*: 3003
- *Text1*: Job ID and name
- *Text2*: Job status
- *Text3*: Duration of the job

Action: Export failed

- *Action-id*: 3004
- *Text1*: Report ID and type of export
- *Text2*: Error description



This will be logged every time an export fails, be it manually trigged or by a cron job. If the export that failed was run by a cron job, you will also find the Cronjob error (code 3003).

Action class: E-mail (31XX)

If you encounter issues with e-mail delivery, please refer to the SMTP debug page at SMTP test.

Action: E-mail sent successfully

- *Action-id*: 3101
- *Text1*: Recipent(s)
- *Text2*: Subject
- *Text3*: Size (if available)

Action: E-mail sending failed

- *Action-id*: 3102
- *Text1*: Recipent(s)
- *Text2*: Subject
- *Text3*: Size (if available)
- *Text4*: Error description. A complete stack trace may be available on the logs.

Action class: AMO events (32xx)

Action: AMO list completed

- *Action-id*: 3201
- *Text1*: The campaign that the list belongs to
- *Text2*: The list that was completed
- *Text3*: Whether the list is currently reservable
- *Text4*: -

Action: AMO list reopened

- *Action-id*: 3202

- *Text1*: The campaign that the list belongs to
- *Text2*: The list that was completed
- *Text3*: Whether the list is currently reservable
- *Text4*: -

Action: AMO campaign completed

- *Action-id*: 3203
- *Text1*: The campaign
- *Text2*: -
- *Text3*: -
- *Text4*: -

Action: AMO campaign reopened

- *Action-id*: 3204
- *Text1*: The campaign
- *Text2*: -
- *Text3*: -
- *Text4*: -

Action: AMO number batch loaded

This event is triggered by an API insert or a batch uploiad of numbers.

- *Action-id*: 3205
- *Text1*: The campaign that the list belongs to
- *Text2*: The list that was completed
- *Text3*: How many numbers were added
- *Text4*: -

Action: AMO list updated via API

- *Action-id*: 3206
- *Text1*: The campaign that the list belongs to
- *Text2*: The list that was completed
- *Text3*: The current list weight after the update
- *Text4*: Whether the campaign is paused

Action: AMO numbers recycled for list

- *Action-id*: 3207
- *Text1*: The campaign that the list belongs to
- *Text2*: The list that was completed
- *Text3*: How many numbers were recycled in this run
- *Text4*: -

Action class: Configuration changes (4XXX)

Configuration changes are logged in the same way for most configuration editors.

This works whether the user started the change from the GUI, or a webservice is performing the change, or QM itself through the Synchronizer.

When the detail of a specific element is recalled, the action is logged as "Displayed", while when a change is made, the action is logged as "Changed".

On Changed actions, you can have:

- New items
- Items cloned
- Items saved (from scratch or from a previous version)
- Items deleted



Note that any time you press Save, the item is logged as changed, even if you change no items, because at least the last change date is updated.

Whenever possible, the current "title" for the editor is logged along the record's id.

In general, the remote IP and session IDs are noot traced, but you will likely have a close logon/logoff record for the same user that you can match.

The following codes are logged:

- `QUEUES_VIEW`: 4001
- `QUEUES_UPDATE`: 4002
- `AGENTGROUPS_UPDATE`: 4012
- `AGENTGROUPS_VIEW`: 4011
- `AGENTS_UPDATE`: 4022
- `AGENTS_VIEW`: 4021
- `BROADCASTMSG_UPDATE`: 4032
- `BROADCASTMSG_VIEW`: 4031

- CALLTAGS_UPDATE: 4042
- CALLTAGS_VIEW: 4041
- CBTS_UPDATE: 4052
- CBTS_VIEW: 4051
- CLASSES_UPDATE: 4062
- CLASSES_VIEW: 4061
- CRONJOBS_UPDATE: 4072
- CRONJOBS_VIEW: 4071
- DNIS_UPDATE: 4082
- DNIS_VIEW: 4081
- EXPORTCALLS_UPDATE: 4092
- EXPORTCALLS_VIEW: 4091
- EXPORTJOBS_UPDATE: 4102
- EXPORTJOBS_VIEW: 4101
- FEATURES_UPDATE: 4112
- FEATURES_VIEW: 4111
- IVR_UPDATE: 4122
- IVR_VIEW: 4121
- KNOWNNUMBERS_UPDATE: 4132
- KNOWNNUMBERS_VIEW: 4131
- LOCATIONS_UPDATE: 4142
- LOCATIONS_VIEW: 4141
- OUTCOMES_UPDATE: 4152
- OUTCOMES_VIEW: 4151
- PCODES_UPDATE: 4162
- PCODES_VIEW: 4161
- QAFORMS_UPDATE: 4172
- QAFORMS_VIEW: 4171
- QAITEMS_UPDATE: 4182
- QAITEMS_VIEW: 4181
- QAPERFTRACKRULEGROUPS_UPDATE: 4192
- QAPERFTRACKRULEGROUPS_VIEW: 4191
- REPORTSEXPORT_UPDATE: 4202
- REPORTSEXPORT_VIEW: 4201
- SKILLS_UPDATE: 4212

- SKILLS_VIEW: 4211
- TASKLIST_UPDATE: 4222
- TASKLIST_VIEW: 4221
- USERS_UPDATE: 4232
- USERS_VIEW: 4231
- TRANSCRIPTION_SERVERS_UPDATE: 4234
- TRANSCRIPTION_SERVERS_VIEW: 4233
- CFGPROPS_UPDATE: 4242 (showing which property was changed when using the Explore System Parameters ttransaction)
- CFGPROPS_VIEW: 4241
- CUSTOMBLOCKS_UPDATE: 4252
- CUSTOMBLOCKS_VIEW: 4251
- CLIENTS_UPDATE: 4262
- CLIENTS_VIEW: 4261
- CASES_UPDATE: 4272
- CASES_VIEW: 4271
- AMO_LISTS_UPDATE: 4282
- AMO_LISTS_VIEW: 4281
- AMO_NUMBERS_UPDATE: 4292
- AMO_NUMBERS_VIEW: 4291

Action class: Wallboard events (50XX)

Wallboards are the Realtime 3 pages. In all these records, the user is the one who performed the action.

Action: Wallboard opened

- *Action-id*: 5001
- *Text1*: The wallboard ID
- *Text2*: The wallboard name

Action: Wallboard saved

- *Action-id*: 5002
- *Text1*: The wallboard ID
- *Text2*: The wallboard name

Action: Wallboard deleted

- *Action-id*: 5003
- *Text1*: The wallboard ID
- *Text2*: The wallboard name

Action class: Chat events (51XX)

Action: Chat deleted

- *Action-id*: 5101
- *Text1*: The call's UniqueID
- *Text2*: The id of the deleted row in the chat table
- *Text3*: If the chat was a transcription, the state of the transcription channel that generated it (e.g. `done`); blank otherwise

Action class: Testers (52XX)

These events are logged when a diagnostic tester is about to connect to an external server. The tester is free to connect to any host, so this log is the audit trail of what was tried and by whom.

To limit abuse, each user can run each tester at most 5 times in 5 minutes. When the limit is reached, the tester refuses to run until enough time has passed. Only runs that were actually started are logged here, so refused attempts do not extend the wait.



These records are counted to enforce the limit. If you exclude this action class from the audit log, the limit will not be enforced for that user.

Action: AMI tester run

Logged when the AMI tester (Asterisk Manager connectivity and dialplan) is run.

- *Action-id*: 5201
- *Text1*: The Asterisk server that the tester connects to
- *Text2*: The AMI port
- *Text3*: The AMI login used for the test
- *Text4*: The IP address (dotted quad) of the user's workstation

Action: SMTP tester run

Logged when an e-mail is sent with the SMTP tester, see SMTP test. Just opening the page is not logged.

- *Action-id*: 5202

- *Text1*: The SMTP relay host that the tester connects to
- *Text2*: The SMTP port
- *Text3*: The SMTP login used for the test
- *Text4*: The IP address (dotted quad) of the user's workstation

Appendix F: The QueueMetrics watchdog page

QueueMetrics has a watchdog page - that is, if you request a page called: *'/sysup.jsp*', you get an answer that is a valid JSON object of the format:

```
{
  "state" : "QMUP",
  "ramFreeMb" : 118,
  "ramTotalMb" : 152,
  "ramMaxMb" : 1818,
  "generatedOn" : "Tue Nov 05 16:44:30 CET 2013",
  "version" : "13.12.01 build 607"
}
```

If the page contains the string QMUP, this means that QueueMetrics is working and able to talk to the database.

The other information on memory usage can be used to be stored in a long-term tracking system (e.g. Nagios, Zabbix, Pandora....) in order to make it possible to understand if there are memory issues.

We offer a sample watchdog script that checks whether QM is running or not and restarts it - you can find it at *'WEB-INF/mysql-utils/restart/watchdogQm.pl*'.

Accessing JMX counters

You can extract JVM metrics in JSON format without needing a dedicated JMX client by adding the `JMXTOKEN` parameter to the `sysup.jsp` endpoint call:

```
curl "http://server.my:8080/queuemetrics/sysup.jsp?JMXTOKEN=mysecret"

{
  "state" : "QMUP",
  "generatedOn" : "Sun Feb 08 09:52:47 CET 2026",
  "version" : "24.11.12 / 376 - 2025.11.20-16:01 - 7bebec488b9@rel_24_11_m",
  "jmxCollectionTime" : 0,
  "ramFreeMb" : 64,
  "ramTotalMb" : 170,
  "ramMaxMb" : 170,
  "jmx" : {
    "gc_g1_old_num" : "1",
    "gc_g1_old_time" : "77",
    "gc_g1_yng_num" : "2818",
    "gc_g1_yng_time" : "10668",
    "jvm_code_cache" : "NOT_FOUND",
    "jvm_cputime_ns" : "14704680000000",
```

```
    "jvm_heap_used" : "110732592",
    "jvm_loaded_classes" : "7799",
    "jvm_nonheap_used" : "111011960",
    "jvm_thread_count" : "27",
    "jvm_total_compilation_time" : "68169",
    "jvm_unloaded_classes" : "4172",
    "pool_max_idle" : "5",
    "pool_max_total" : "1000",
    "pool_num_active" : "0",
    "pool_num_destroyed" : "111",
    "pool_num_idle" : "1",
    "pool_num_waiters" : "0",
    "tomcat_active_sessions" : "84",
    "tomcat_bytes_received" : "67704793",
    "tomcat_bytes_sent" : "139333040",
    "tomcat_error_count" : "5",
    "tomcat_request_count" : "540907",
    "tomcat_threads_acceptors" : "2",
    "tomcat_threads_busy" : "1",
    "tomcat_threads_current" : "10",
    "tomcat_total_processing_time" : "2151784"
  }
}
```

This approach is particularly useful for integrating with monitoring systems like Grafana, Prometheus, or other observability platforms. Instead of configuring a full JMX client connection, you can periodically poll this HTTP endpoint and parse the JSON response to extract metrics for visualization and alerting.

Configuration Requirements

- You must set the JVM property `qm.jmx.token` with your chosen secret token for JMX counter generation to be enabled
- The set of metrics returned is defined in the `WEB-INF/jmx.properties` configuration file - the default ones are a good starting point to diagnose issues.

Appendix G: Keyboard shortcuts

The following keyboard shortcuts are available in the QueueMetrics web interface.

They have been tested on default installations of Google Chrome and Mozilla Firefox, on Windows and Ubuntu.

Different operating systems or browsers might not work correctly.

Agent Page Shortcuts

Navigation Shortcuts

- **Ctrl** + **alt** + **o** → Navigates to the Overview page
- **Ctrl** + **alt** + **i** → Navigates to the Interactions page
- **Ctrl** + **alt** + **c** → Navigates to the Control page
- **Ctrl** + **alt** + **t** → Navigates to the Clients page

Topbar Shortcuts

- **Ctrl** + **alt** + **p** → Toggles the Softphone panel
- **Ctrl** + **alt** + **n** → Toggles the Notifications panel
- **Ctrl** + **alt** + **a** → Toggles the Agent Status panel

Control Shortcuts

- **Ctrl** + **alt** + **w** → Toggles the Start/End work button on the Overview page
- **Ctrl** + **alt** + **b** → Controls the Pause/Unpause function on the Control page
- **Ctrl** + **alt** + **m** → Controls the Change/Amend pause function on the Control page
- **Ctrl** + **alt** + **j** → Joins the selected queues on the Control page
- **Ctrl** + **alt** + **u** → Leaves the selected queues on the Control page

Softphone Keyboard Shortcuts

- **Enter** (while typing in the input box): Dial a number
- **Ctrl** + **e** → Hang up the current call
- **Ctrl** + + → Add a new session
- **Ctrl** + **a** → Answer an incoming call
- **Ctrl** + **b** → Access the Transfer panel

Wallboard Shortcuts

- **Shift** + **q** → QueueMetrics Home

- **Shift** + **p** → Open SoftPhone
- **Shift** + **l** → List Wallboards
- **Shift** + **m** → List Views
- **Shift** + **a** → Add Widget

Appendix H: Glossary

ARA: The Asterisk Realtime Architecture.

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.

Aggregate queue: see → *Composite queue*.

Atomic queue: a queue that matches one-to-one to an underlying Asterisk queue.

Call analyst: a person whose job is to grade agent's calls through the QA system. This may be a specific job or an agent.

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

Call-file: a function in Asterisk, where by writing a specially-crafted file, it is possible to interact with the dial-plan. With a modern version of Asterisk, it is generally better to use the *Manager interface*.

Campaign: a set of outbound calls placed for a given purpose.

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

DNIS (Dialed Number Identification Service) is a service that tracks which telephone number was dialed by a customer (e.g in case of multiple incoming numbers).

Engagement code: an acronym that represents a grading items for the QA forms.

Grader: see → *Call Analyst*.

Invisible queue: a queue that is defined in QueueMetrics but cannot be chosen from the front page. Useful for queue → *wildcard matching*.

IVR (Interactive voice response) is a dialog system that allows Asterisk to detect keypad inputs and address the caller to the correct queue or department.

Jabber: see → *XMPP*.

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.

Outbound queue: see → *Campaign*.

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.

Wildcard matching: a technique to group together all queues that have a name sharing similar characteristics. See Using wildcards in queue names for details.

XMPP (eXtensible Messaging and Presence Protocol): An open instant messaging protocol. Used for off-band communication in the AGAW subsystem.

Appendix I: Removed and deprecated features



Deprecated features are usually retained for at least one year before actually being removed from the product, as to allow a smooth migration path.

Deprecated features

- **Direct access** to the queue_log file is deprecated. You should upload data using Uniloader to the HTTP interface.
- **Live calls** are deprecated (see Live calls).
- **Payroll** where queue_log data is modified (see Payroll).

Removed features

- **Sub-queues**. If you still use sub-queues to handle different logical queues on a shared agent pool (e.g. for reservations of multiple hotels, events or movies, where you have different numbers for each item but have one large agent queue), you should migrate to call tags or DNIS (removed in 26.10).
- The **Icon page** (removed in 26.01).
- All **AGAW** functions.
- The old **agent page** (pre-Icon). Removing the key `NEWAGENTPAGE` now leads to the new QAP page (23.09).
- The **XML-RPC** interface for webservices was removed in 22.02. JSON webservices implement all the functionality and more, and are easier to consume from modern development environments.
- The Oreka video player is not supported any longer. Only audio playback is supported though Oreka.
- The paged data blocks `UD02`, `OD03` and `OD04` are automatically replaced in 20.11 with their non-paged equivalents, as paging is now handled by the new reports page.
- The data blocks `OD01`, `XS01` and `AC01` were removed in 20.11, because their functionality is not needed with the new reports page.
- The CallCabinet driver was removed in 20.11.
- The Qloader script was superceded by Uniloader, that is flexible and stable. It was removed in 20.11.
- The Visitor's page was removed in 20.11.
- The AGAW client for Firefox is not supported anymore in 19.10.
- Switching time-zones was removed in 19.10
- Exporting audio in batches was removed in 19.10.

Appendix J: Softphone Troubleshooting

Common Issues

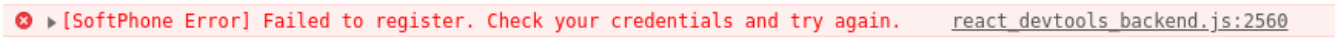
The SoftPhone does not work with an extension configured for Zulu

Unfortunately, extensions created for FreePBX's Zulu won't work with the embedded SoftPhone. They will also cause other compatibility issues, e.g. when being monitored (e.g. `chanspy`).

On the other hand, if you already have Zulu installed, you probably don't need a different softphone as well.

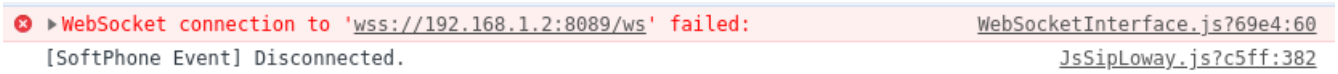
The SoftPhone does not register

Usually, if the SoftPhone does not register correctly, it's due to the fact that the extension credentials are not correct. If this is the case, you should see a message in the Browser console.



If you have this error, check your extension's username and password and try again.

Alternatively, the softphone could be unable to connect because the WebSocket address is wrong, and the phone cannot connect to the PBX. In that case you should see an error like this:



To fix this, make sure that you are connecting to the correct WebSocket using WSS. A good WSS connection string should look like this:

```
wss://IPADDRESS:PORT/ws
```

example

```
wss://192.168.1.1:8089/ws
```

If both the extension credentials and the WebSocket address are correct, then you might need to accept the SSL Certificate of the PBX (If it's a self-signed certificate). In that case you should see an error in the console with this code:

```
NET::ERR_CERT_AUTHORITY_INVALID
```

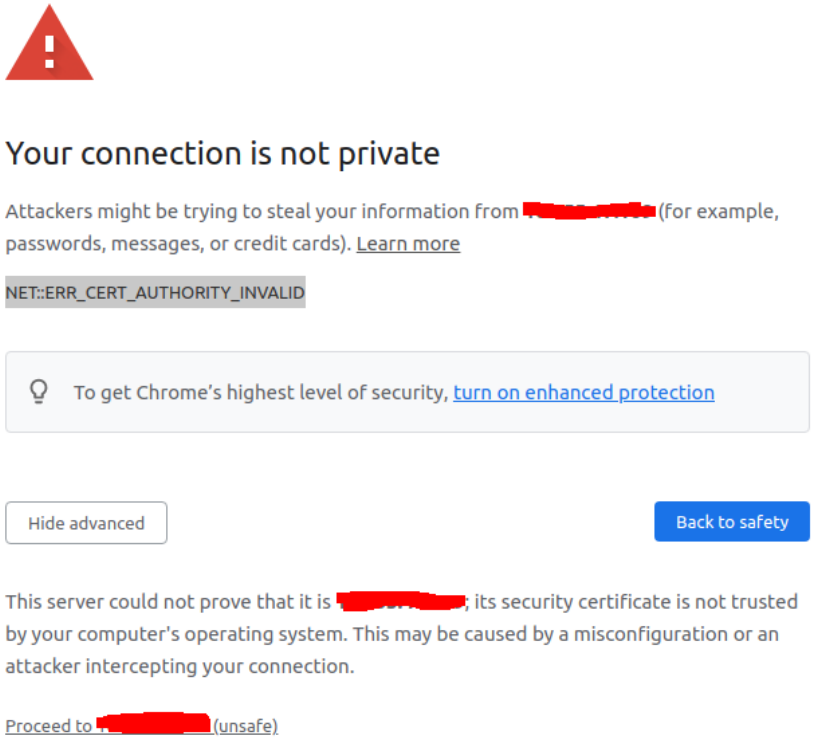
To fix this, navigate to this address, substituting your WebSocket IP address and Port.

```
https://IPADDRESS:PORT
```

example

```
https://192.168.1.1:8089
```

And create an exception for this address by clicking on "Proceed to ..." under "Advanced" in Chrome (Other browsers should offer a similar interface).



A common issue when setting up WebRTC is related to an incorrect WebSocket URL (wss://) or invalid/misconfigured SSL/TLS certificates. To verify that the WebSocket URL is correct and reachable, you can test it using curl.

How to Test

Take the exact same WebSocket URL (wss://) that you configure in your softphone, and simply replace wss:// with https://

Testing the WebRTC Connection

To verify that the WebRTC connection is working correctly, you can use the following `curl` command:

```
curl -i 'https://<YOUR_SERVER_ADDRESS>:8089/ws' \
-H 'Accept: */*' \
-H 'Accept-Language: en-US,en;q=0.5' \
-H 'Accept-Encoding: gzip, deflate, br, zstd' \
```

```
-H 'Sec-WebSocket-Version: 13' \
-H 'Sec-WebSocket-Protocol: sip' \
-H 'Sec-WebSocket-Extensions: permessage-deflate' \
-H 'Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==' \
-H 'Connection: keep-alive, Upgrade' \
-H 'Sec-Fetch-Dest: empty' \
-H 'Sec-Fetch-Mode: websocket' \
-H 'Sec-Fetch-Site: same-site' \
-H 'Pragma: no-cache' \
-H 'Cache-Control: no-cache' \
-H 'Upgrade: websocket'
```

Remember to replace <YOUR_SERVER_ADDRESS> with the correct domain name or IP of your server.

Expected Response

If the connection is successful, the command will not terminate but you should see a response like this:

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: sip
```

This indicates that the WebSocket handshake was successful and the WebRTC connection is established.

Troubleshooting

If you do not see the expected response, possible causes include:

- Firewall restrictions blocking the WebSocket port
- Invalid or misconfigured SSL/TLS certificates
- Asterisk WebSocket service not running or unreachable

Outgoing Calls Hang up as soon as I click the Dial button

If calls hang up immediately check your browser's console. If you see this error:



This means that your browser does not have permission to use your microphone or headphones/speakers. Make sure to enable the appropriate permissions to fix this error.

Incompatible SDP

Enable AVPF must be set to yes

Commmon issues when not using Nat (WebRTC clients are on same local network)

Since they are on the same network the phones do not need ICE Servers to work.

If calls do not reach the other client, or you get `invalid status 5 when answering, you probably have ICE Servers setup when they should be empty.

Common issues when using NAT (WebRTC clients are NOT on same local network)

SIP Failure code 503

PBX is unavailable or you haven't set the ICE Servers correctly.

Here is a list of Google's STUN servers, that are free to use. You can try if these work for you by setting one or more as the ICE servers for your WebRTC SoftPhone client, under ICE Servers in the phone settings.

```
stun:stun1.l.google.com:19302
stun:stun2.l.google.com:19302
stun:stun3.l.google.com:19302
stun:stun4.l.google.com:19302
```

One way audio

If one side does not have the ICE servers setup correctly, it will not receive any audio streams.

Signs of a good ICE Connection

Check in the browser console for the following string. If it's there it means that the ICE connection has been completed correctly.

```
[SoftPhone Event] Ice State Changed. Connection State: connecting Ice Connection State: connected Ice Gathering State: complete
```

Signs of a bad ICE Connection

If the Ice Connection State is "disconnected", then the ICE connection is not active and will not work correctly.

```
[SoftPhone Event] Ice State Changed. Connection State: connecting Ice Connection State: disconnected Ice Gathering State: complete
```