Puzzel Contact Centre Solution.
Raw Data Detailed Description.

February 13th 2020.
Database v1.6

puzzel. Solving Customer Interactions

puzzel. Solving Customer Interactions

Contents

1 Introduction

1.1 Revision log

Version	Major changes	Author
13.02.2020 / Db v1.6	New version of the documentation. No changes in the db! The most important changes/updates in the document: - Added info about time format/time zone… (1.5.4) - new chapter for **Scheduled task** (2.12) - More info for **Silent monitoring** (2.15) - Updated info in chapter 3.4 Offered to agent	Paal Kongshaug
14.11.2018 / Db v1.6	*This document describes the db version 1.6.* The most important changes from db v1.5 are: New columns in table **`call_events`**: - **sla** (For queue events. The queue's SLA if any) - **alt_sla** (For queue events. The queue's Alternative SLA if any) - **dte_scheduled_callback** (For queue events. The scheduled time if any) - **result_response** (Extra information for conversation events) New columns in table **`enqreg_header`**: - **marked_unansw** (set to true if agent clicked "Marked as unanswered" for a Dialler call) - **reserved** (set to true if agent rescheduled a Dialler call "to myself") New columns in table **`call_event_extras`** - **agent2agent** (a record will be created here with agent2agent=1 for an agent-to-agent call) - **from_cache** (a record will be created here with from_cache = 1 when a conversation event's agent is "drawn from cache") New columns in view **`vw_enqreg_total`**: - marked_unansw - reserved	Paal Kongshaug
27.07.2018	New version of the documentation. No changes in the db!. The most important changes in the document: - Added Consult to queue in chapter 2.3 Call transfer/consult call - New chapter 2.7.2: Scheduled calls - Added more details in chapter 2.8 Dialler (e.g. silent call callback) - New chapter 2.10.2 Automated agent and chatbot - New chapter 2.11.7: Email/Etask using the Puzzel "script engine"	Paal Kongshaug

1.2 About this document

This document has two main parts:

I. **General information and details about what data records that are generated for different scenarios** (incoming call, call transfer/consult, callback in queue, outgoing calls, email, chat, log on/off, wrap-up etc). Chapters 1, 2 and 3.

II. **Technical part** – chapter 4 - database tables, fields and values and chapter 5 – Functions and stored procedures and chapter 6 – Guidelines for downloading

It's very important that part I (chapters 1-3) is read and understood before making queries and analysing the results.

The intention of this document is to give the customer a good understanding of what the data means so that the data can be used wisely and hopefully without causing misunderstandings and wrong conclusions.

1.3 Statistics vs Archive vs Raw data

In Puzzel Contact Centre there are several different statistical reports available in the Administration Portal. The smallest time period for statistics reports is 15 minutes, and reports are normally available 2 hours after each quarter's end. Example: Reports for the time period 0900-0915 are available shortly after 11:15, and reports for day 1 are available shortly after 02:00 on day 2.

In the Administration Portal's Archive, you can find details about each call (including recording), chat (including the written chat text) and email handled in Puzzel. The Archive gives a simplified view of each call's/request's "journey", that is, menus, queues and agents involved.

For customers that need more details than what is available in Puzzel real-time, the Archive and in the historical statistics reports (standard and customised), we recommend Raw data.

1.4 Call and Agent Detailed Records – an Overview

Puzzel establishes a database for the customer. This database contains the Call and Agent Detailed Records for the customer's Puzzel solution. Records for calls (sessions) and agent activities arrive in the database continuously, normally within a delay of maximum 5 minutes after the session (activity) ended. The database is located in Puzzel's hosting environment and is operated by Puzzel, and the customer can access the database over a secure connection over internet. The database is based on Microsoft SQL Server 2012 (or newer), which the customer is given read access to.

1.5 Technical set-up

1.5.1 SQL database, IP-addresses, access and security

Each customer's Raw data DB is hosted in a **dedicated instance of SQL Server**, so that each customer's data is completely isolated.

Communication to the instance is over a **dedicated port** (per customer), and **IP address whitelisting** is enforced, ensuring communication is only allowed from the IP addresses specified by the customer.

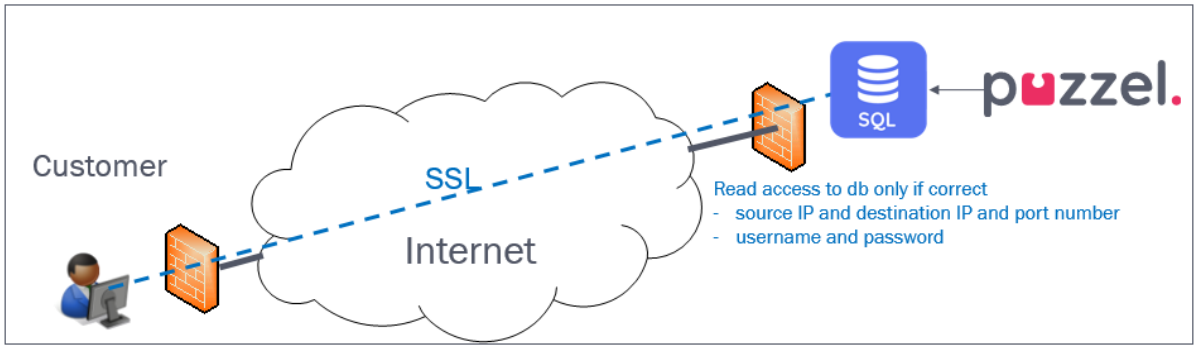
The **read-only user accounts** Puzzel creates for each customer are dedicated for the customer's DB and defined inside the SQL instance.

The SQL server is configured to **enforce encryption on all client connections, so that the** connection between server and client will always be encrypted by SSL regardless of whether 'Force Encryption' is set on the Client. The Certificate is issued by third party provider (DigiCert), with Key length 2048 bits and Hash algorithm sha256.

TLS 1.2 will be used for all clients supporting this. Older TLS and SSL protocols are no longer supported.
https://support.microsoft.com/en-us/help/3135244/tls-1-2-support-for-microsoft-sql-server.

A user can access the database only if

- *traffic comes from agreed source IP-addresses*
- *traffic is to the correct IP-address and the correct port number is used*
- *correct username and password are entered*



1.5.2 How long to save data?

Data records will be stored in the database for as many months as is agreed between the customer and Puzzel. Data records older than the agreed storage time will be deleted every night.

1.5.3 Normal usage

If the customer uses the database in a way that results in a very high bandwidth consumption and very heavy load on Puzzel's infrastructure, Puzzel has the right to limit the available bandwidth on Puzzel's internet access towards the database.

1.5.4 Time format / time zone

Records (events) in tables call_events, agent_events etc have `dte_start/ dte_updated` on format YYYY-MM-DD hh:mm:ss.nnn, e.g. 2019-05-20 12:17:43.440. The data type is datetime.

A Puzzel customer solution is linked to a "statistics country" and a time zone (e.g. Norway and W. Europe Standard Time)

The values in `dte_start/ dte_updated` is in **local time**, that is, in the local time of the Puzzel solution's defined "statistics" country.

1.6 Changes in database

Puzzel may change the database schema when needed, e.g. add new fields or introduce new values in a field, or add new tables, when new Puzzel functionality is added or functionality is changed, and this change requires the database to be updated. Puzzel will notify customers before the database schema is changed.

1.7 The responsibility to handle the data

It is the customer's responsibility to apply to the relevant laws and regulations for storing and processing the call and agent detailed records. The customer must fill out a data processing agreement and define therein the scope of Puzzel as a data processor on behalf of the customer. The agreement must be signed by both parties.

1.8 How to utilise the data

The customer can log on to the database with a SQL tool and run his own queries or use the Puzzel specific database-functions and stored procedures that are available. See details in chapter Functions and Stored Procedures. The customer can also connect to the database using e.g. Microsoft Excel.

What can the detailed records be used for?

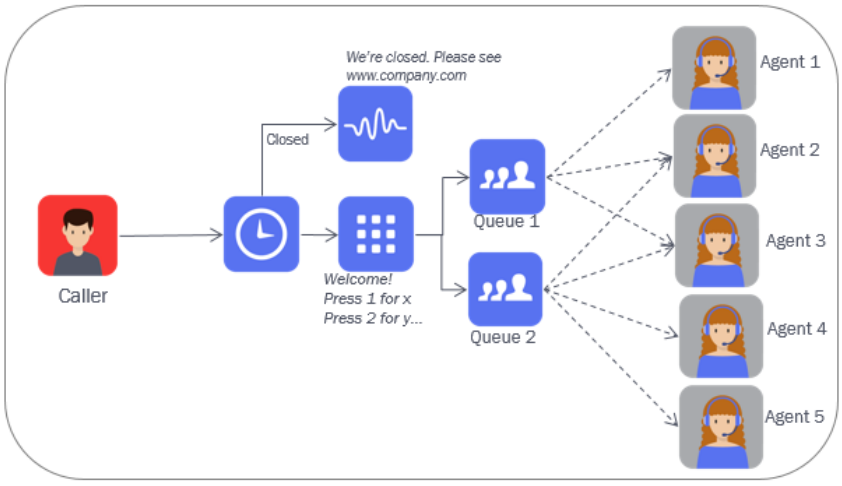
Examples:

- *Find a specific call based on known caller's number, agent or time*
- *Show all details for one agent for a given time period*
- *Generate your own report for a very specific time period (hh:mm:ss-hh:mm:ss)*
- *Generate your own report at fixed intervals (every hour, day, week or month)*
- *Look for repeat callers (callers calling back within a day or two) / analyse first call resolution*
- *Select necessary data for the last x minutes and calculate needed input to your workforce management system*

You do not need to download all the data to your own local database to analyse the data and generate reports, but if you want to store all the data in your local database/data warehouse, you may do this. One way to do it is to download all *new/updated* records every night. Remember that records may be updated in the database after initially created, e.g. for wrap-up, and remember that callback in queue results in at least 2 sessions. See chapter 6 Guideline for downloading to own local database(s)/system(s).

1.9 Caller, Request, Queue and Agent

When a caller dials a phone number to a Puzzel solution (an access number), the call is routed through the different modules in the Puzzel service logic, e.g. a menu and a queue.



The queue finds an available agent and calls to the agent. If this agent does not answer, the queue finds another agent to call. If the agent answers, the caller and the agent are joined. Some agents can answer one queue only, while other agents can answer several queues.

In addition to *phone* calls, the media types *email*, *social media* and *chat* can be handled in Puzzel. Any written request with an URI can be queued as email in Puzzel. *Request* is often used as a term for both calls and written requests (chat, email…) in queue.

1.10 Initial training and support

An introductory Raw data **training session** is compulsory, and we recommend that this session is arranged before or shortly after anyone starts writing SQL queries.

This is to be sure that both the ones that want reports from Raw data and the ones that will write the SQL queries actually understand the Raw data basics (what events that are generated for different call/email/chat scenarios) and Puzzel statistics basics.

Please note that Puzzel does not include support for how to write SQL in the Raw data product, and Puzzel's Support department cannot assist with customer-created SQL queries.

If you need additional consulting/training after the initial training, please contact your Puzzel Account Manager to discuss and arrange an extra consulting/training session.

2 Call and Agent Detailed Records

2.1 Call events and Agent events

The 2 main tables are call_events and agent_events.

- The **call_events** table contains all events related to routing of the calls and written requests, whether they are related to agents or not. Examples are Conversation events, Menu events and Queue events.
- The **agent_events** table shows all agent activities (log on/logoff/pause/return). In addition, it shows the Conversation events related to agents. This means that you will find Conversation events related to agents both in the `call_events` table and in the `agent_events` table.

2.2 Basic call flow and different call events

For each incoming request (call/chat/email etc.) to Puzzel, the Puzzel platform generates one or more **events** wrapped in one **session** (internal_iq_session_ id).

In the **call_events table** there are different **types** of Call events, e.g. Initiation event, Menu event, Queue event and Conversation event. For most of the contact centre routing modules the incoming call goes through, an event is created. One call may go through none or several audio/caller input/menu modules (menu events), none or several queues (queue events) and there may be generated none or several calls to agents or other destination numbers (conversation events).

When the **incoming call ends** (caller hangs up) the session ends, and the different events for this session are processed and inserted into the database.

Example:
A caller calls in at 12:00:00 and spends 1 minute in menus, 4 minutes in queue a, 3 minutes speaking with agent x and then 1 minute in queue b, and finally 4 minutes speaking with agent y, so the call ends at 12:13:00. All events for this incoming call are inserted in the database a few minutes after 12:13:00.

All events for one session will have the same call_id.

In special cases (e.g. Callback in queue , outbound calls and email in queue) there might be 2 or more sessions with the same `call_id`.

For incoming phone calls, the **Initiation event** represents the time the incoming call is connected to the Puzzel platform, but this event is also used for other requests like emails/chats and for outgoing calls initiated by the agent or the Dialler.

Example events for a call_id, with some of the columns in `call_events`:

rec_id	call_id	call_sequence	media_type_id	dte_start	duration_tot_sec	duration_speak_sec	source	destination	queue_key	menue_key	menue_choice	agent_id	event_type	result_code	answered	wrap_up_sec
6093331	74985249	1	1	27.09.2016 15:06:14	72		21490xxx	81511569					i		1	
6093332	74985249	2	1	27.09.2016 15:06:14	5					info_welcome			m	k		
6093333	74985249	3	1	27.09.2016 15:06:19	7					menu_main	1		m	k		
6093334	74985249	4	1	27.09.2016 15:06:26	29				q_sales				q	k		
6093335	74985249	5	1	27.09.2016 15:06:26	23	0		98214xxx	q_sales			220770	c	t		15
6093336	74985249	6	1	27.09.2016 15:06:49	37	**31**		21490xxx	q_sales			150674	c	k		39
6093337	74985249	7	1	27.09.2016 15:06:55	31								r	k		

Since some calls last very long (e.g. long time in queue and a long conversation with an agent, who may transfer the caller to a new queue or agent…), it may take a long time from a call starts until the events for the call are available in the database.

All events have a `call_id`, a start, a duration, a result (and other details), and there may be events that are overlapping in time, e.g. a queue event and conversation event(s) or 2 conversation events (consultation call).

In very special cases, menu events for audio files played to the caller while in queue are created, and such menu event's duration will be 'within' the queue event's duration.

2.2.1 About the caller's number and the Initiation event's source

For an incoming call, the caller's number is usually put in the Initiation event's `source`. Please note the number may be stored with or without +/00 and country code.

If the caller (A) calls to number B, which is forwarded to a Puzzel access number (C), the caller's number ***might*** end up in the Initiation event's `additional_source` or `redirect_source`, and the called number (B) might end up in `source`. (Puzzel reports what's received as calling party number, additional number and redirecting number from the network.)

For calls from callers that have a **secret number** or that have used CLIR (Calling Line Identity Restricted), the caller's number is not presented (`source` is empty).

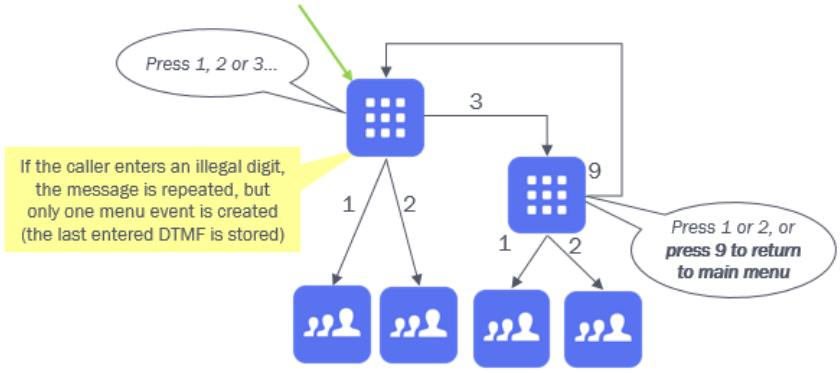
For outbound calls from agents, the called number is usually reported as `source`.

For agent-to-agent calls, the reported `source` is "xxxxxxxx".

2.2.2 About Menu events

One call_id can have 0 or more menu events, and for one call_id there might even be more than 1 menu events for the same `menue_key`. In each menu, one can configure what should happen if the caller dials an illegal digit or dials nothing within a time limit.

- *The caller can stay in the menu module and hear the message again. If a caller first enters 4, then 7 and finally 1 in a menu module asking for digits 1, 2 or 3, this will result in only one menu event with menue_key =1.*
- *The caller can be routed directly to an exit if he presses an illegal digit. The next module can be a queue, but if the next module is a menu saying 'wrong digit' and the call then is routed back to the first menu, there will be 2 menu events from the first menu.*



And, there can be menus with option 'press x to return to the previous/main menu'. If this exists, then one call_id can have 2 (or more) menu events from the same menu. The descriptions/names for the different menus can be found in table `menues`.

To compare number of menu events per `menue_choice` with the corresponding queue events (queue arrivals) is tricky due to the above, and since there might be several ways to enter a queue (from different menus, from different access numbers, from other queues and call transfer from agent to queue).

2.3 Call transfer and consult call

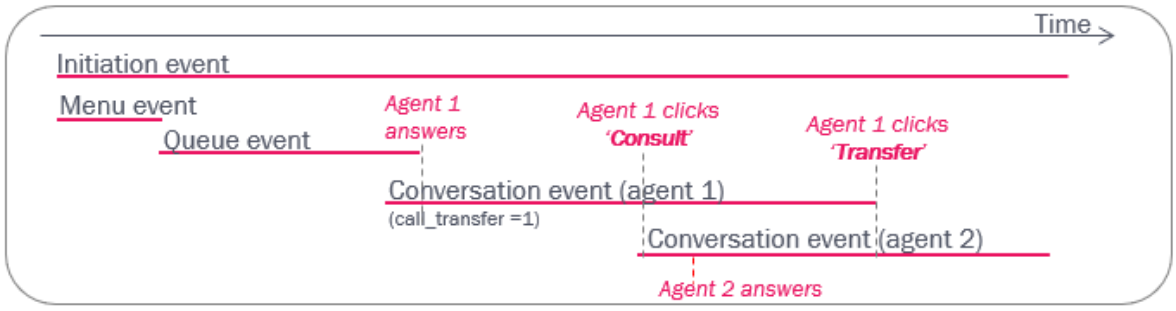
When agent 1 has answered a call, he or she may do a consult call and/or a call transfer.

2.3.1 Blind transfer to agent or a phone number

If agent 1 transfers the caller to agent 2 (or a phone number), a new Conversation event is created, and you can see in agent 1's Conversation event that the call was transferred (`call_transfer`=1). The Conversation event for the call to agent 2 (or the phone number transferred to) will not contain a queue_key.

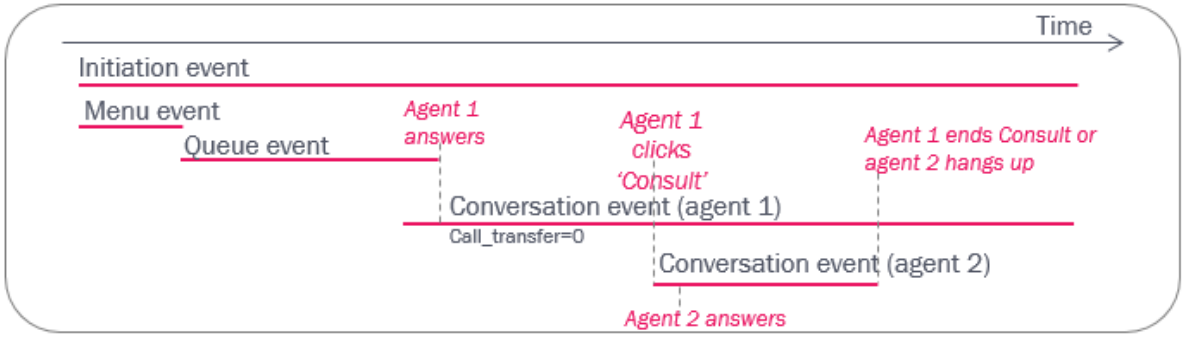
2.3.2 Consult + transfer

If agent 1 makes a consult call to an agent or a phone number and then transfers, the Conversation event for agent 2 will (not have a queue_key and) be **partly overlapping** in time with the Conversation event for agent 1.



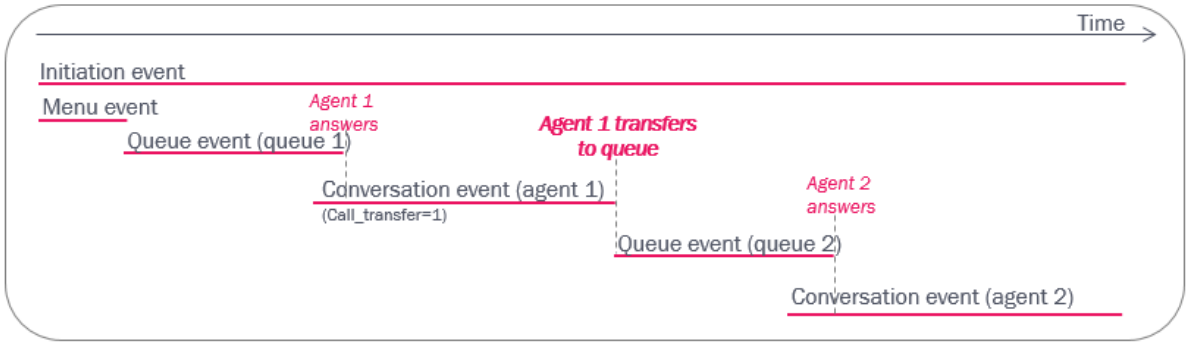
2.3.3 Consult only

If agent 1 does a consult call to agent 2 without transferring, the Conversation event for agent 1 will have `call_transfer`=0, and the conversation event for agent 2 will not have a queue_key.



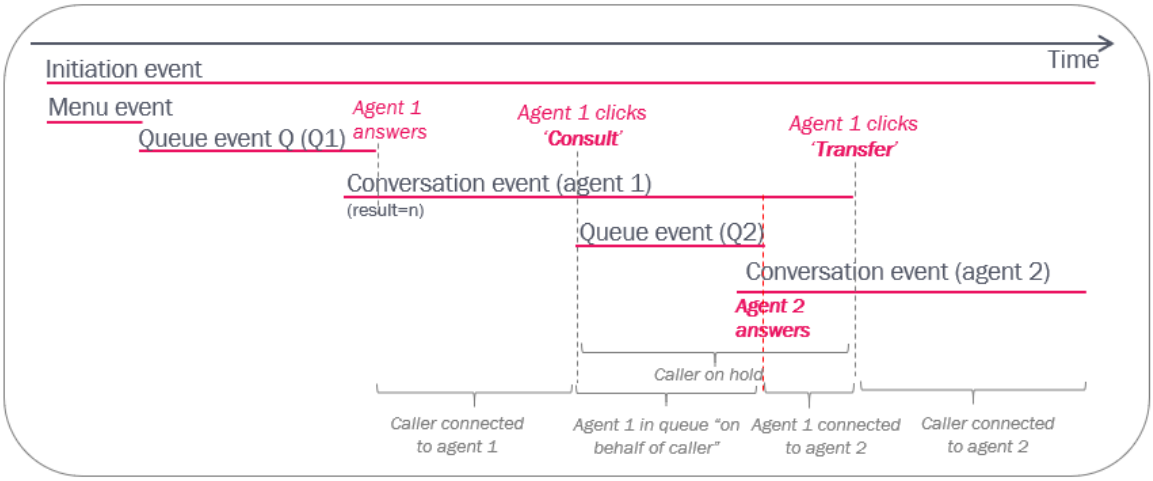
2.3.4 Transfer to queue

If agent 1 **transfers the caller to a queue**, there will be a new Queue event and possibly one or more new Conversation events later.

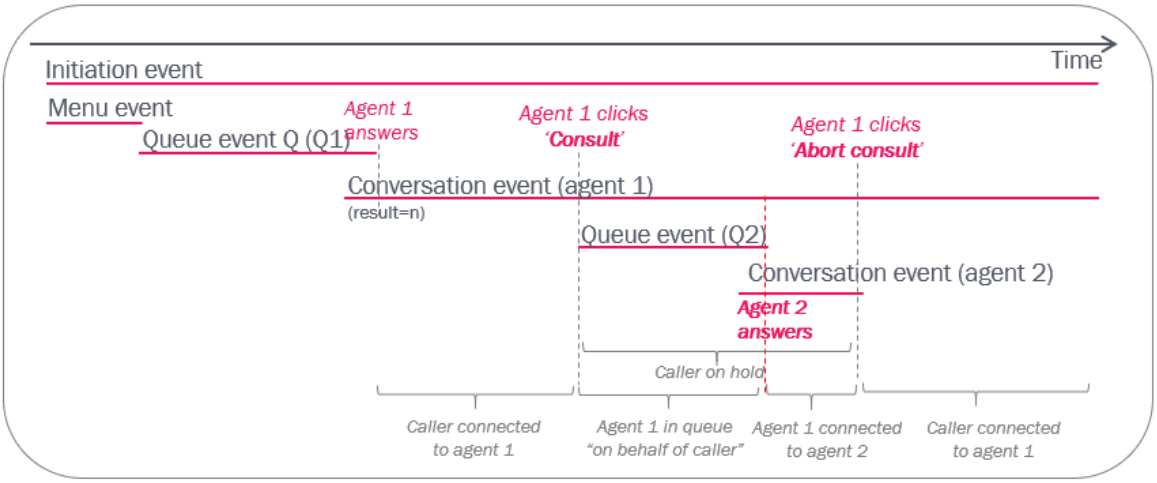


2.3.5 Consult to queue

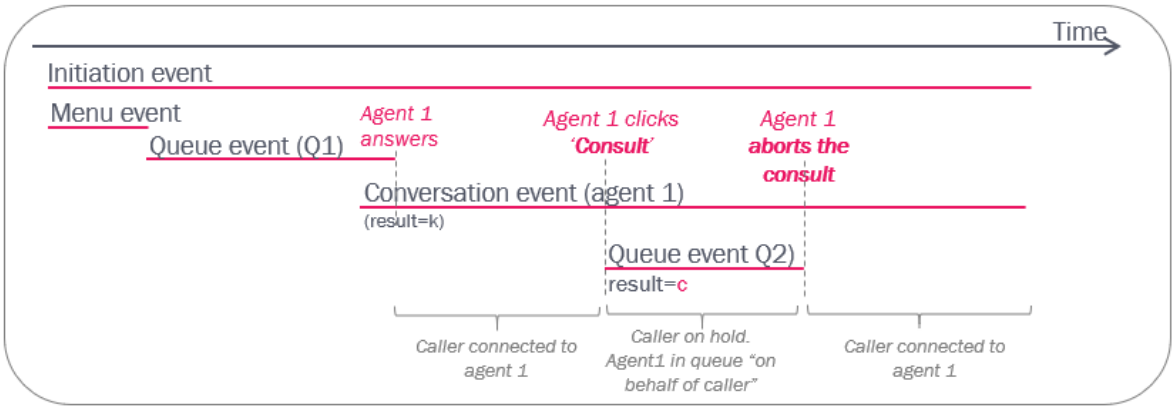
If agent 1 makes a consult call to a queue (to consult with *any* agent on that queue), agent 1 can **transfer** the caller to agent 2 after agent 2 on the new queue has answered:



If agent 1 makes a consult call to a queue, agent 1 can **end the consult** (and continue speaking with the caller) after having spoken with the other agent:

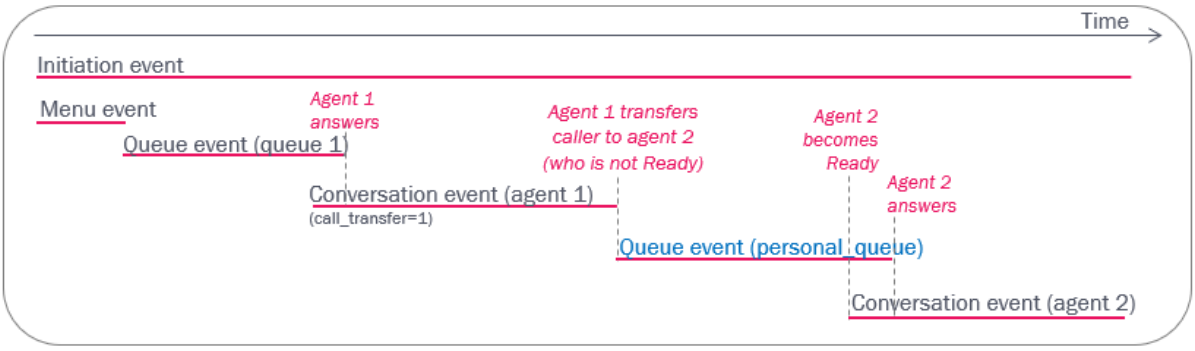


If agent 1 makes a consult call to a queue, but no agent answers within a short time, the agent can **abort the consult** and continue speaking with the caller:



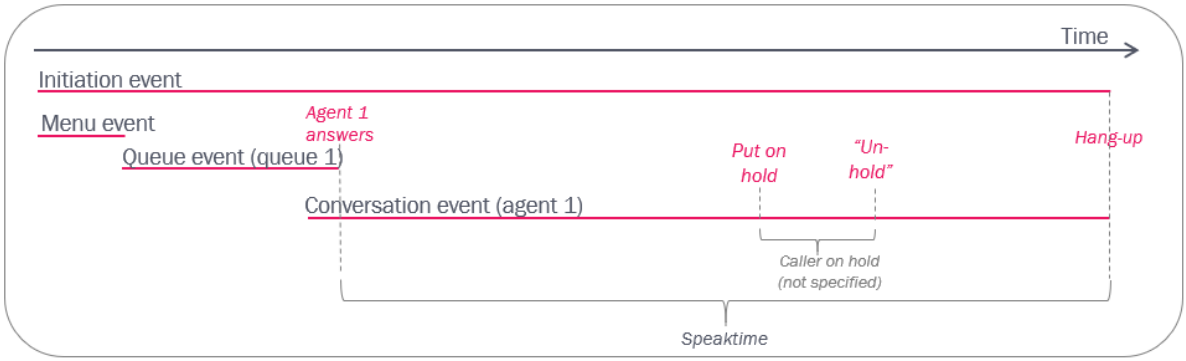
2.3.6 Transfer to not ready agent / Personal queue

If a Personal queue is configured in the Puzzel solution, an agent can transfer a caller to another agent that is logged on but not ready. If a caller is transferred to a not ready agent, the caller is put in the Personal queue waiting for this agent. When the agent becomes Ready, the call is sent to the agent (unless the caller has hung-up while waiting in the Personal queue).



2.3.7 Put on hold

The agent may **put the caller on hold** (during the reported Speaktime). This on-hold time is not specified in the Conversation event.



2.4 Agent status and agent events

2.4.1 Log on/Log off/Pause/Return (ContactCentreStatus)

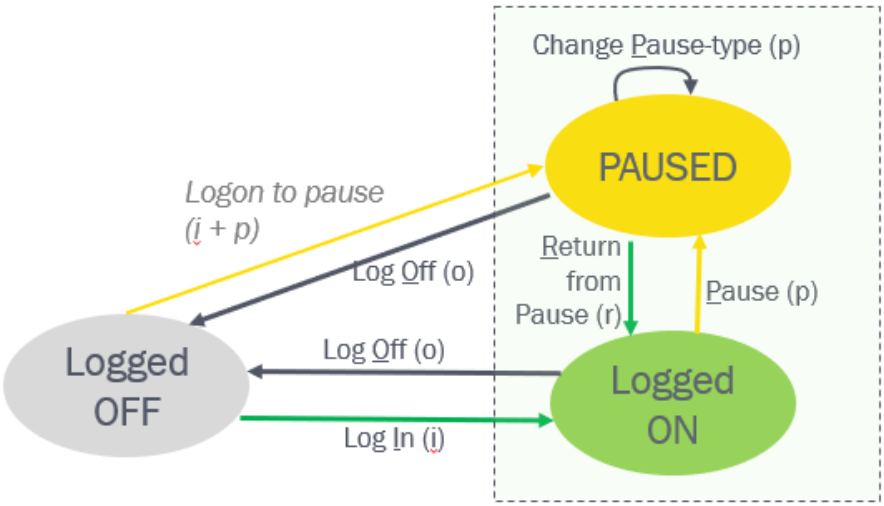
We do not create a raw data record when a user is signing in to Puzzel. Such sign-ins are shown in the Admin Portal's Access log.

The Puzzel agent (when signed in) can log on to queue, logon to pause, go to pause (from logged on), return from pause and log off. There are 3 different statuses:

- **Logged on** (and not paused)
- (Logged on and) **Paused**
- **Logged off**.

Please note that in the Puzzel user interfaces the term 'Logged on' sometimes includes agents in Pause.

For each time an agent logs on/logs off/starts pause/returns from pause, an **Agent event** with the correct timestamp and event_type (i/o/p/r) is created. These agent events do not have any duration, so one must calculate to find the time spent in each status.



Web services - **ContactCentreStatus**: LoggedOff, LoggedOn, Pause

Pause can be used for any reason for not being ready. It is possible to define different ***Pause types*** like Lunch, Meeting, Admin etc, and if this is defined, the actual Pause type is included in the agent event for entering Pause. An agent can change status from one pause type to another pause type.

The agent logs on to queue with a profile. The agent may have different profiles, each representing one or more queues that the agent can answer. The agent's profile name,

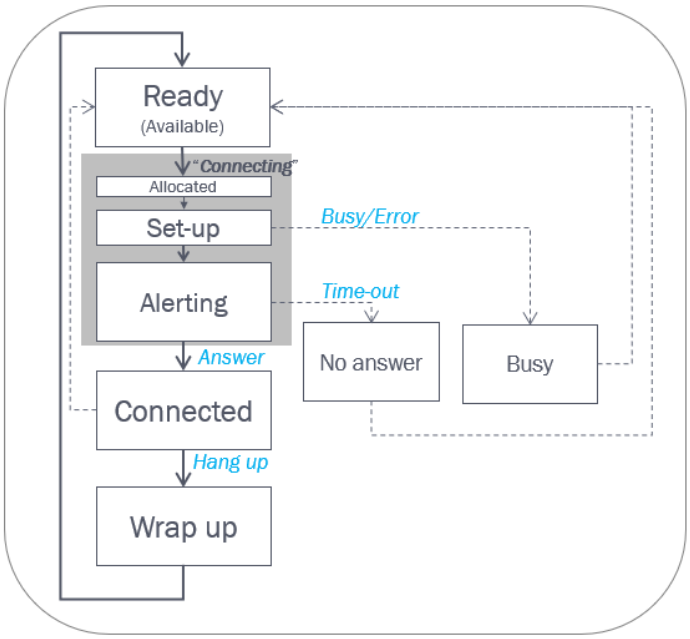
phone number and user group are shown in the agent event for logons (event_type=i), but not the actual queues the profile represents.

Please note that if you have a **pause type** called *Wrap-up* or *Extra wrap-up* (not recommended!), the time in this pause type is reported as time in status *Pause*, and not as time in status Wrap-up, so it will not be included in the reported Wrap-up and AHT in Puzzel real-time views and Puzzel statistics!

In agent.puzzel.com it is possible for agents to log on directly to pause (from January 2019). To avoid challenges with this new status transition (Off -> Paused) in any raw data queries or calculations, we decided to create both a login event and a pause event with dte_start some milliseconds apart when agent logs on directly to pause.

2.4.2 Agent status in the queue "engine"

When an agent logs on to the queue(s) she is set to status *Ready* (sometimes called *Available)*. When a Puzzel queue allocates this agent and then calls (or sends a written request) to the agent, the agent's status is first (*Allocated* and then) *Set-up* and then *Alerting*, but these statuses are presented as *Connecting* in the Puzzel user interface.



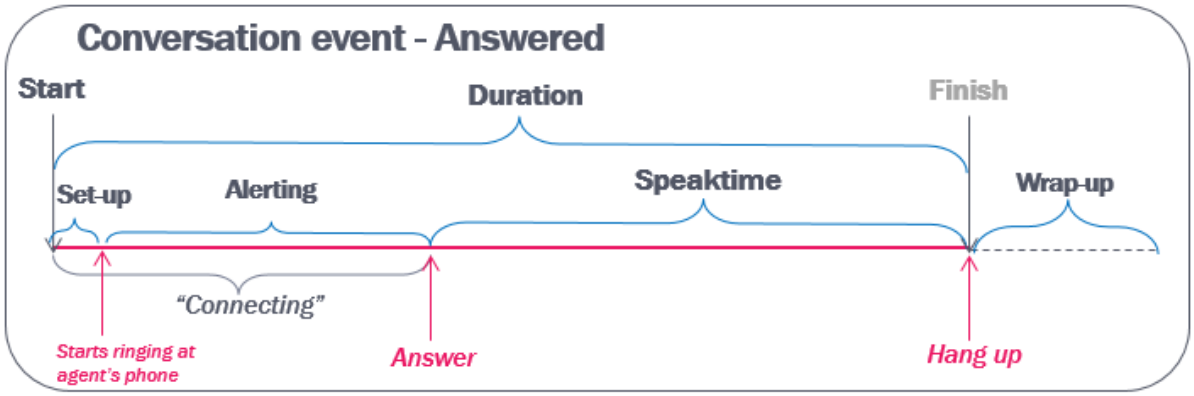
- If the agent answers, the agent status is set to Connected. At hang-up, the status changes to Wrap up (if > 0 sec wrap-up is configured) and then Ready.
- If the call to the agent number results in busy (or error), the agent status is set to Busy for the configured number of seconds (default 1) and then to Ready.
- If the call is not answered by the agent after x seconds ringing (configurable), the agent is set in status *No answer* for the configured number of seconds, and then back to Ready.

In the **Conversation event** we specify

- **Start** time (`dte_start`). At call set up (when written request was sent to agent)
- **Duration** (`duration_tot_sec`). From start until hang-up (until agent closes the written request tab),
- **Speaktime** (`duration_speak_sec`). From agent answers call (accepts the written request) until call hang-up (agent closes the written request tab),
- `dte_speak_start` (when speaktime starts)
- **Wrap-up** time (`wrap_up_sec`).
- And other details

Finish is shown in most illustrations and mentioned in the text, but Finish is not a field in the database. Finish is equal to Start (`dte_start`) + Duration (`duration_tot_sec`).

Please note that the Wrap-up happens after the Conversation event's Finish.



2.4.3 "Connecting" vs Set-up and Alerting

The connecting phase (=`duration_tot_sec` minus `duration_speak_sec`) consists of two phases; call set-up and alerting.

The call set-up phase is usually <0,5 sec when Puzzel softphone is used, 1-2 seconds when a landline phone is used, and maybe 3-8 seconds when calling to a mobile number.

The alerting phase ("ringing time") is usually very short (>0,7 sec) if the **softphone auto-answers**, and without auto-answer, the typical alerting time is 10-25 sec.

The lengths of the *Set-up* and *Alerting* phases are specified in the Conversation event (`setup_ms` and `alert_ms`).

Please note that some calls from the Puzzel platform do not receive the correct signalling events from other telecom operators or the local phone/SIP system, and this will affect the reported setup and alerting times. If Puzzel receives "answer" without a

"ringing/alerting" signal first, this will usually result in a too high value for `setup_ms` (the reported set-up will be the sum of actual set-up and alerting time), and the value in `alert_ms` might be 0). If the call events from your Puzzel solution contains 'wrong' values for *Set-up* and *Alerting*, please use the connecting phase (=conversation event duration minus speaktime) as a proxy for ringing time when calculating ringing time at agent level.

2.4.4 Details about Wrap-up

Wrap-up is initially a pre-defined number of seconds, but the agent may shorten wrap-up by clicking *Ready/Log off/Pause* during wrap-up, or click to extend the wrap-up.

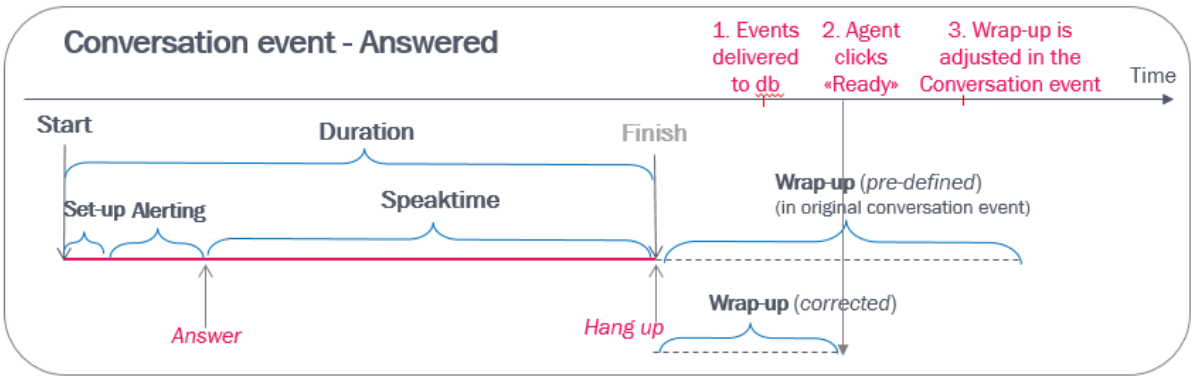
If the agent clicks ***Ready*** while in status wrap-up, the initial reported Wrap-up time (`wrap_up_sec`) in the Conversation event will be corrected, normally after no more than 5 minutes (given that the whole session has ended).

Example:
Shortly after call end, all records for the call_id including the conversation event with the long pre-defined wrap-up (e.g. 600 sec) can be found in table call_events.

rec_id	call_id	equence	dte_start	duration_	queue_key	agent_id	event_type	result_code	wrap_up_	alert_ms	dte_updated
15638962	7951737170097557339	1	20.01.2020 10:48	46			i				20.01.2020 10:50
15638963	7951737170097557339	2	20.01.2020 10:48	5			m	k			20.01.2020 10:50
15638964	7951737170097557339	3	20.01.2020 10:48	8			m	k			20.01.2020 10:50
15638965	7951737170097557339	4	20.01.2020 10:48	10	q_support		q	k			20.01.2020 10:50
15638966	7951737170097557339	5	20.01.2020 10:48	33	q_support	244731	c	k	600	9513	20.01.2020 10:50
15638967	7951737170097557339	6	20.01.2020 10:49	24			r	k			20.01.2020 10:50

After a while, when the agent has (extended and/or) ended his wrap-up, all events for this call_id are **replaced** with new events with **new rec_ids** and a new dte_updated, and now the conversation event has the updated value for wrap-up (here 322 sec).

rec_id	call_id	equence	dte_start	duration_	queue_key	agent_id	event_type	result_code	wrap_up_	alert_ms	dte_updated
15639025	7951737170097557339	1	20.01.2020 10:48	46			i				20.01.2020 10:56
15639026	7951737170097557339	2	20.01.2020 10:48	5			m	k			20.01.2020 10:56
15639027	7951737170097557339	3	20.01.2020 10:48	8			m	k			20.01.2020 10:56
15639028	7951737170097557339	4	20.01.2020 10:48	10	q_support		q	k			20.01.2020 10:56
15639029	7951737170097557339	5	20.01.2020 10:48	33	q_support	244731	c	k	322	9513	20.01.2020 10:56
15639030	7951737170097557339	6	20.01.2020 10:49	24			r	k			20.01.2020 10:56



If the agent is allowed to **extend wrap**, you will sometimes see that the initial reported wrap-up in a conversation event will be extended. One agent can click 'extend wrap-up' one or more times for one call.

If the agent clicks ***Log off/Pause*** while in **wrap-up**, the initial reported wrap-up in the Conversation event will be corrected, normally after no more than 5 minutes after the event happened (given that the session has ended).

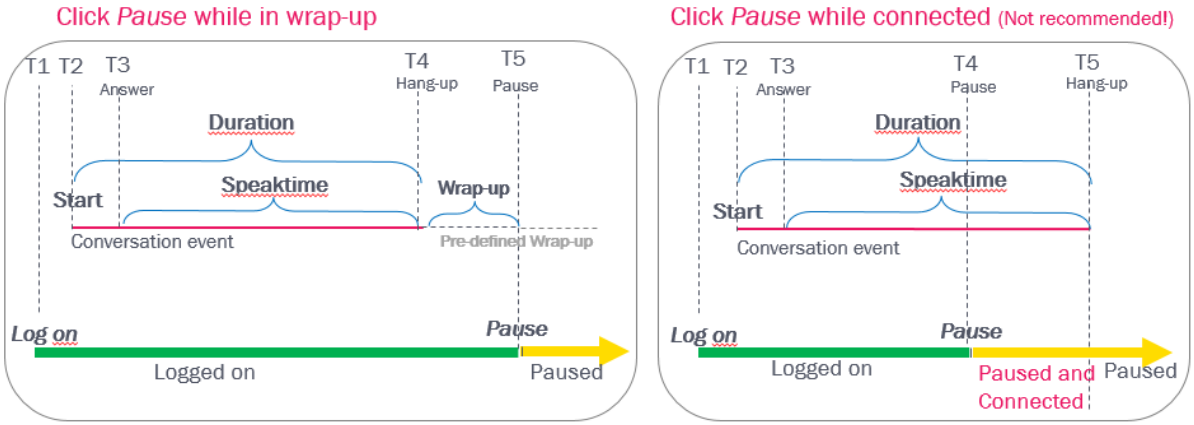
Please note that conversation events with other results than answered (k) also have a value for wrap-up (the pre-defined value that would have been used if the agent answered the call), so when you calculate (average) wrap-up time for calls for agents, **make sure you only include wrap-up for answered conversation events** (result_code=k).

If you have a pause type called *Wrap-up* or *Extra wrap-up (Not recommended!)*, remember that the time in this pause type is reported as time in status Pause (`event_type`='p') , and not as time in status Wrap-up in conversation events!

2.4.5 Connected while in status Pause or Logged off

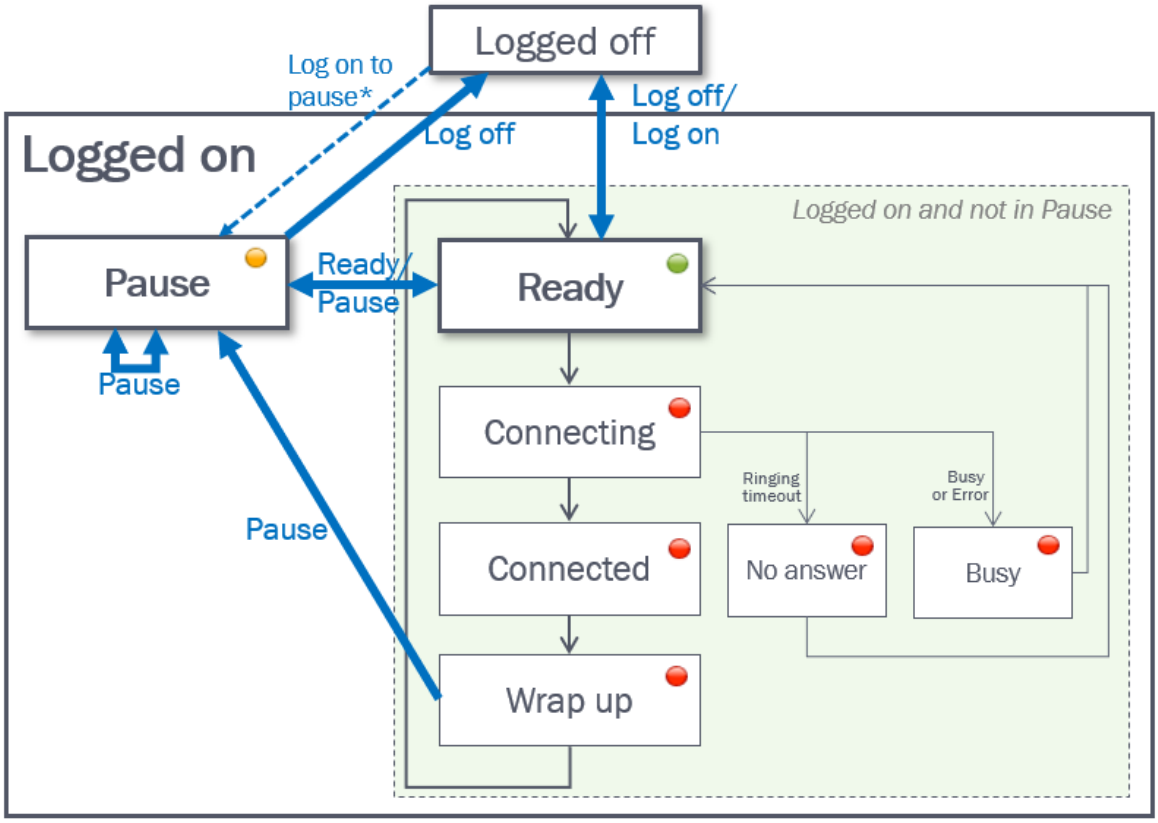
If the agent clicks ***Pause/Log off*** while *Connected*, the agent will be in status Pause/Logged off and in status Connected. The reported wrap-up time will here be 0.

The reason some agents click Pause/Log off while connected might be to avoid having a new call from the queue just after having hung-up since they plan to have a pause or to leave just after call end, but as long as a predefined wrap-up is configured, a new call will not arrive while in status wrap-up!



To avoid situations where an agent is in status [Connected and Pause] or [Connected and Logged off], we recommend that agents do not click Pause/Log off while connected, but **wait until wrap-up starts**, and then **click *Pause/Log off* while in status wrap-up**.

Normal status transitions:

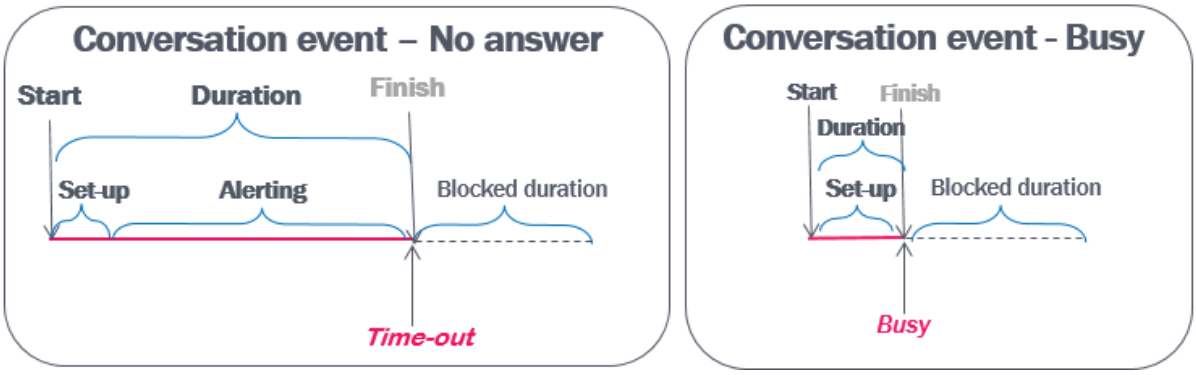


* If the agent clicks *Logon* with a pause type selected, the agent will be logged on and immediately put into pause (both logon and pause records are generated).

2.4.6 Busy and No-answer

If a call (from a queue) to an agent results in **busy or no-answer**, the agent is set to status Busy/No-answer for a configurable number of seconds. This pre-defined Busy/No-answer timeout or 'blocked' time is from v.1.5.0.0 recorded in `block_duration_sec` in Conversation events for media_type_id=1 (phone) with result busy (b) and timeout (t).

Please note that if the agent clicks Ready/Log off/Pause while in status Busy/No-answer, the agent status will change, but the `block_duration_sec` value will not be updated in the Conversation event (as we do with shortend/extended wrap-up).



2.4.7 'Audit log' – see all events for an agent (fnc_agent_events_window)

In table `agent_events` we have agent login (i)/logoff (o)/pause (p)/return from pause (r) events (with no duration) and conversation events with duration that includes information about connecting, connected/speaktime and wrap-up time.

If you want to see the "timeline" for an agent, you can use the function `fnc_agent_events_window` (see Functions and Stored Procedures) to list agent events for an agent, with start/duration and adjusted start/duration, chronologically.

Please note that conversation events for so-called 'slave agents', that is, **chat, social** and **email** requests**, are not included**!

If you use this function with **mode=1**;

- *Login* events are given adjusted duration until next log off (or period end)
- *Logoff* events are given adjusted duration until next logon (or period end if that happens first)
- *Pause* events are given adjusted duration until first *Return* from pause.
- *Return* from pause events are removed
- Answered (result k) and unanswered (result t, h, b…) *Conversation* events are listed
- Answered Conversation events with wrap-up time > 0 are divided into two:
  - one Conversation event
  - one **Wrap-up event** (event_type=w)*
- For the (ready) time between Conversation events we are generating ***Available events*** (event_type=a)*
- You can remove the Login events (which is implicit expressed by Available) from the result by adding 'where event_type not in ('i') to your query.

* Events with type Available (a) and Wrap-up (w) do not appear directly in the agent_events table.

Please note that we do not create (blocked) events for the (usually few and short) time periods the agent is in status Busy/No-answer.

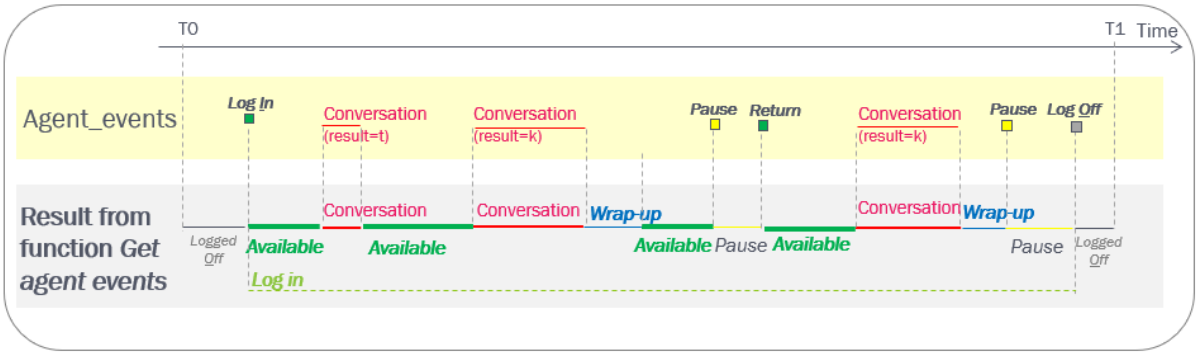
Example query and result:

```
select * from [dbo].[fnc_agent_events_window]
(150674,'27-jun-2016 10:00','27-jun-2016 10:20',1, , )
order by dte_start
```

	agent_id	dte_start	adj_dte_start	service_num	event_type	result_code	duration_sec	adj_duration_sec	duration_speak_sec	queue_key	pause_type_name	pause_type_id	internal_odr_id	Comment
1	150674	23.06.2016 15:19:25	***27.06.2016 10:00:00***	81511569	o	k	0	220						Most recent logon/log off before time period start, with adj_dte_start = time period start and adj_duration_sec = time until first login
2	150674	27.06.2016 10:03:39	27.06.2016 10:03:39	81511569	i	k	0	*814*						**Login event with adjusted duration until next logoff**
3	150674	27.06.2016 10:03:39	27.06.2016 10:03:39		a	k	41	41						Generated ***Available*** event
4	150674	27.06.2016 10:04:20	27.06.2016 10:04:20	81511569	c	*t*	20	20	0	q_support			863542049	Conversation event with result=t (offered but not answered)
5	150674	27.06.2016 10:04:40	27.06.2016 10:04:40		a	k	113	113						Generated ***Available*** event
6	150674	27.06.2016 10:06:33	27.06.2016 10:06:33	81511569	c	k	130	130	120	q_support			863543925	Conversation event with result=k (130 sec duration incl 120 sec speaktime)
7	150674	27.06.2016 10:08:43	27.06.2016 10:08:43	81511569	w	k	15	15		q_support				Generated Wrap-up event
8	150674	27.06.2016 10:08:58	27.06.2016 10:08:58		a	k	111	111						Generated ***Available*** event
9	150674	27.06.2016 10:10:48	27.06.2016 10:10:48	81511569	p	k	0	*86*			Break2	3039		Pause with adjusted duration equal to time in pause
10	150674	27.06.2016 10:12:15	27.06.2016 10:12:15		a	k	72	72						Generated ***Available*** event
11	150674	27.06.2016 10:13:26	27.06.2016 10:13:26	81511569	c	k	97	97	90	q_support			863546877	Conversation event with result=k (97 sec duration incl 90 sec speaktime)
12	150674	27.06.2016 10:15:03	27.06.2016 10:15:03	81511569	w	k	15	15		q_support				Generated Wrap-up event
13	150674	27.06.2016 10:15:18	27.06.2016 10:15:18		a	k	57	57						Generated ***Available*** event
14	150674	27.06.2016 10:16:15	27.06.2016 10:16:15	81511569	p	k	0	*58*			Break2	3039		Pause with adjusted duration equal to time in pause
15	150674	27.06.2016 10:17:13	27.06.2016 10:17:13	81511569	o	k	0	*166*						Logoff with adjusted duration until period end

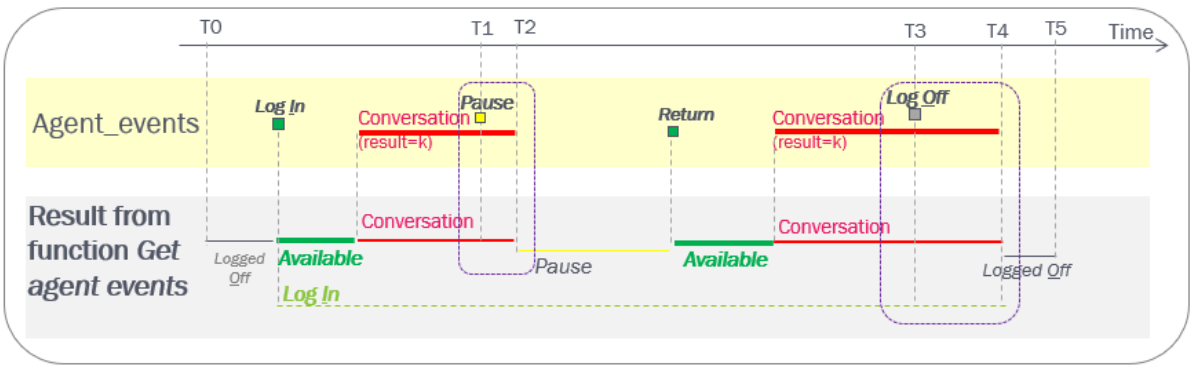
If you add "`where event_type not in ('i')`" to the query, row number 2 in the result (the login event with adjusted duration) will not be included.

The values in columns `adj_dte_start` and `adj_duration_sec` are used to make the next 2 illustrations, but the actual length of each event in the graph is not correct:



What if the agent clicks *Pause* while connected or *Log off* while connected?

- If the agent clicks *Pause* while connected (T1), the agent has status [Paused AND Connected] until call end (T2), and there will be no wrap-up.
- If the agent clicks *Log Off* while connected (T3), the agent has status [Logged off AND Connected] until call end (T4), and there will be no wrap-up.



New from DB v1.5 in these 2 cases:

- *The Pause (Log Off) event's* `adj_dte_start` *will be moved from T1 to T2 (from T3 to T4) and its* `adj_duration_sec` *will be reduced*
- *The Log In event's* `adj_duration_sec` *will be increased (so that it ends at T4)*

So, how do I find the time an agent has been "idle"?

Idle time might be defined as the time an agent is logged on and not in pause and waiting for a request from a queue.

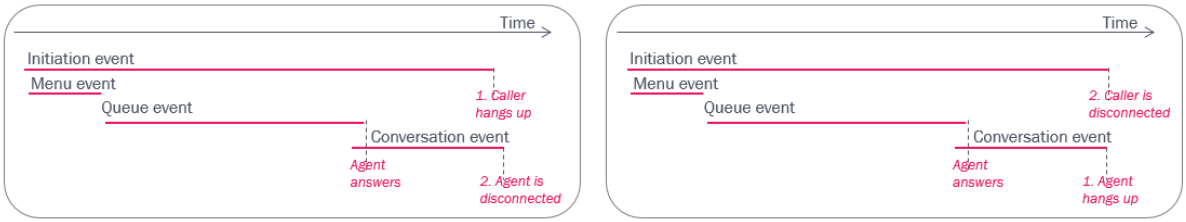
- For agents that only handle phone (and email if *Allow multiple emails* is OFF) in Puzzel, one could say that the agent's idle time is equal to the time the agent is in status Ready. This will be the time reported as *Available* plus the duration for Conversation events not answered when using the function *Get agent events.*
- For agents that handle **Chat, Social and/or email** (if *Allow multiple emails* = ON) in Puzzel, this is more complex. Please see chapter about Chat. The short version is that you have to include the conversation events for chat, social and email that the master agent's slave agents receive, in order to calculate when an agent is idle. One agent might have overlapping chats/social/emails, so the idle time is not equal to total time minus sum speaktime/connected time for the agent.

2.4.8 Puzzel agent status change based on Skype status

If a Puzzel agent uses the Puzzel **desktop** agent application and the Search property '*Update presence on user from Skype for Business through Agent Application*' is ON, this agent's Puzzel status will be changed from *Ready* to *Busy* if the user becomes engaged in a **Skype for Business call**. This feature is also supported in the new agent application (agent.puzzel.com) from June 2019. When the user's Skype status is back to Available, the Puzzel status is changed back to Ready. Please note that we don't create raw data events for such status changes.

2.4.9 Who hung up first?

When a caller is connected to an agent, the standard behaviour for a Puzzel solution is that when the caller hangs up first, the agent is disconnected, and when the agent hangs up first, the caller is disconnected.



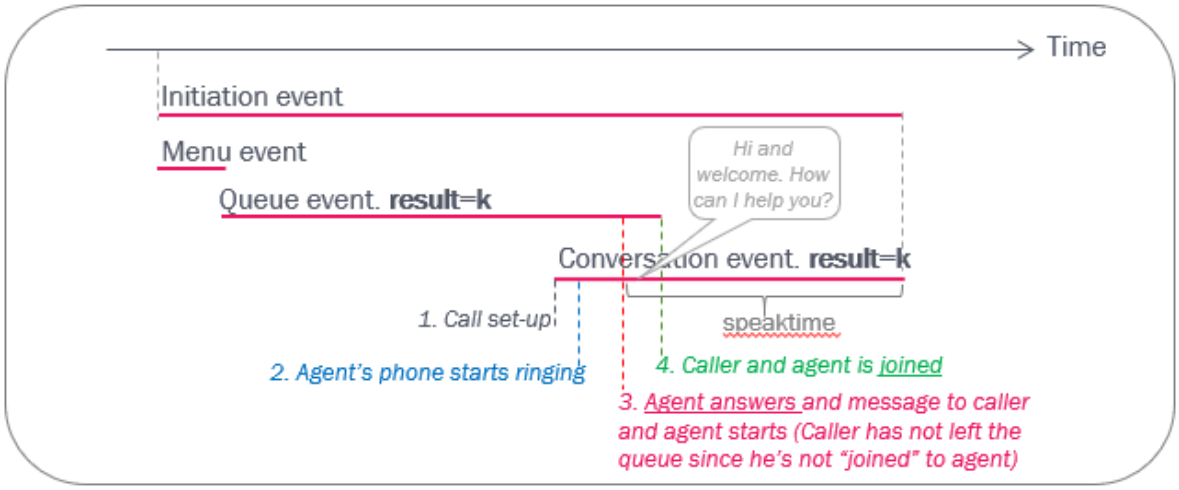
The actual time the caller was disconnected is reflected in the finish time in the **initiation event**, and the agent's disconnect time is found in the **conversation event's** finish.

Please note that if the caller and the agent were finished speaking and said goodbye, it's not unusual that the agent hangs up first. And, remember that the connection to the caller's phone or to the agent's phone may be lost due to network problems without the caller or the agent hung up on purpose!

2.4.10 Announcement to caller and agent before join

The default solution is that when an agent answers a call from a queue, the caller and the agent are joined immediately, and both the queue and the conversation event get result=k. The queue event ends when the agent answers (= when the speaktime starts).

If it is configured that a message is played for the caller and the agent before joining caller and agent, the queue event does not end when the agent answers the call, but when the parties are joined after the message is played.

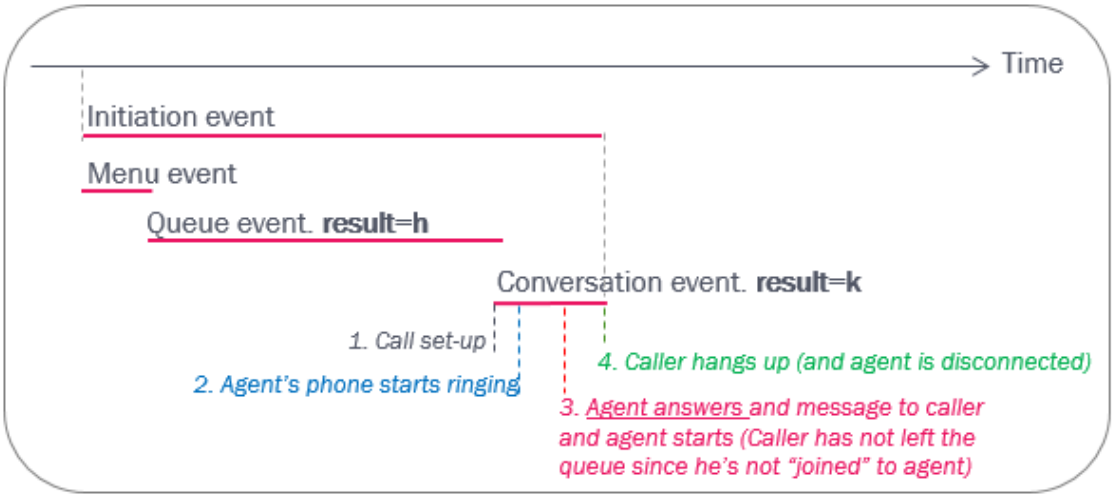


2.4.11 Hang-up in queue after agent answer (while message is played)

If it is configured that a message is played for the caller and the agent before joining the two parties, the caller or the agent can hang up while the message is playing!

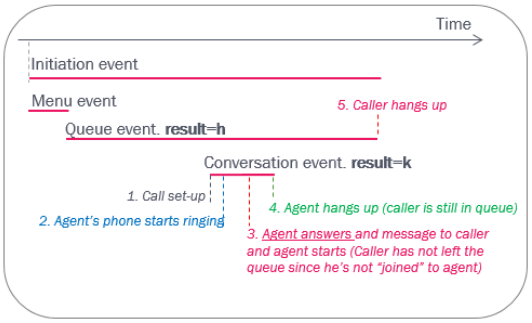
If the caller hangs up while a message is played for agent and caller:

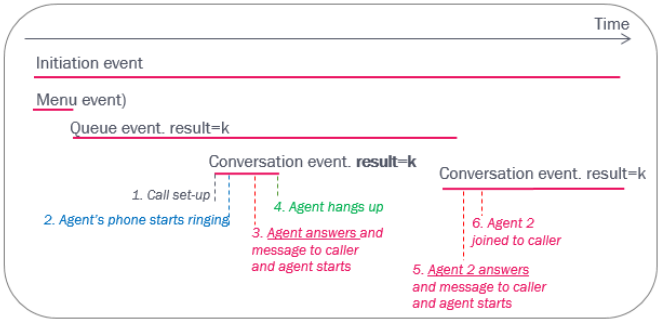
- *the **queue** event is given **result=h** (since caller was not joined to agent)*
- *the **conversation** event for the agent is given **result=k**.*



If the agent hangs up while a message is played for agent and caller:

- *the **conversation** event for the agent that answered gets **result=k**, and the caller is returned to queue to wait for a new agent (since he was not joined to agent)*
- *the queue event's result depends on how the caller later leaves the queue*





Special case

If a caller waiting in queue hangs up just after the allocated agent answered, the agent conversation event will have result=k (answered) and speaktime=0, and the queue event will have result=h (hang-up) since the caller was not joined to the agent.

2.5 Call event result codes

The different values for `result_code` for call events can be found in table `call result codes`

2.5.1 Results for Queue events

Different values for `result_code` used for **Queue events** (`event_type`=q)**:**

Result	Description
a	The caller left the queue after having dialled the defined **interrupt** digit (**abort**).
b	**Fallback**-exit used.
c	**Cancel**. Admin removed the call from queue (or consult to queue aborted by agent)
d	**Deleted**. The (email/social/phone callback) request was deleted from queue by admin. Or an agent has cancelled a callout before he/she answers the phone.
e	**Error**. The call (request) exited the queue due to an error.
f	**Full**. The caller tried to enter the queue but was denied access since the queue was full. (The max number of callers in queue is defined by the customer)
h	The caller **hung up** while waiting in queue, or the chatter closed the chat before agent accepted.
k	**OK**. The call (chat/email/social) waited in queue and was **answered** (accepted) by an agent
q	The caller ordered **Callback** in Queue.
s	**Reject**/Empty. The caller was denied access to the queue since the number of logged on agents were 0 (or too low)
t	The caller exited the queue after having waited the max allowed time in this queue (**time-out**).
?	**Unknown** (Something probably went wrong)

2.5.2 Results for Menu events

Different values for `result_code` used for **Menu events** (`event_type`=m)

Result	Description
d	The caller (dialled illegal digit(s) and) used the **default** exit
e	**Error**. Something went wrong.
h	The caller **hung up** in the menu.
k	**OK**. The caller dialled legal digit(s) in the menu
m	Max number of tries exceeded, and the call used the defined **Max-tries exit** (if defined).
t	The caller did not make any selection within the defined time limit (**time-out**), and used the defined **Time-out exit**

2.5.3 Results for Conversation events

Different values for `result_code` used for **Conversation events** (`event_type=c`)

Result	Description
b	For phone: The call resulted is **busy** before alerting (usually due to SIP response=486 or 600 or ISUP release code=17), or the agent clicked on the red Puzzel softphone ***hang up/reject*** button in Puzzel agent application while the call was ringing. For email/chat/social: The agent clicked "**Reject**" when a chat/email/social was offered in the Puzzel Agent application.
c	**Set-up timeout**. - The call **timed out** during call set-up - The chat/email/social request timed out while trying to find the agent's Puzzel client
e	**The call failed/error** from network, e.g. Invalid number format (ISUP 28), Unallocated number (ISUP 1), congestion/error in network
h*	The **caller hung up** while this call to agent was in **alerting** phase. (or, an agent using an external phone "Declined" an offered call)
k	**OK**. The call was answered / The written request was accepted by an agent.
q	**The caller hung up** while this call to the agent was in **set-up** phase (before ringing on agent's phone)
t	**Time-out (No answer)**. - The call was **not answered** within the predefined max ringing time (in Puzzel). - The chat/email/social request timed out while waiting for the agent to accept it
?	**Unknown**. Something went wrong

* See Agent rejecting a call

Agent rejecting an offered call

- If agent is using Puzzel softphone and is rejecting an offered (ringing) call by clicking the red hang-up/reject button in agent.puzzel.com, the conversation event will get result = b (busy).
- If agent is logged on to queue with an **external phone number** (instead of Puzzel softphone), and the agent is rejecting an offered call by clicking the black ***Decline*** button in agent.puzzel.com, the conversation event will get **result = h (hang-up)**. In this case, the call is not ended, so the queue event continues, and the next agent that is offered the call might answer.
- If agent is logged on to queue with an external phone number and is rejecting an offered call by clicking No/Reject on his external phone, the conversation event will usually get result = b (busy).

About Conversation event result *Busy* in statistics

If a chat/email/social request is sent to agent and the agent clicks *Reject*, the conversation event will have result_code = b. In Puzzel agent statistics prior to April 2019, conversation events with result=b for phone was not counted as "offered to agent", but result=b was counted as offered for chat/email/social. After April 2019 conversation events for phone with result_code=b is counted as "offered to agent" in Puzzel standard statistics. This change was communicated in the Puzzel release notes for the April 2019 release.

2.5.4 Extra information for calls (result_response)

In addition to the result_code, we also have a column called **`result_response`** (from db v1.6) showing the signalling response code Puzzel received from the network for a phone call. Please note that the value might be an ISUP cause code (usually 1-2 digits) or a SIP response code (3 digits). In the future, there will only be SIP responses in this column.

The main purpose with this column is to give more information for calls with result_code=e (error). The 2 main types of error are:

- phone number not in use (ISUP cause 1 or SIP response 404)
- error in network

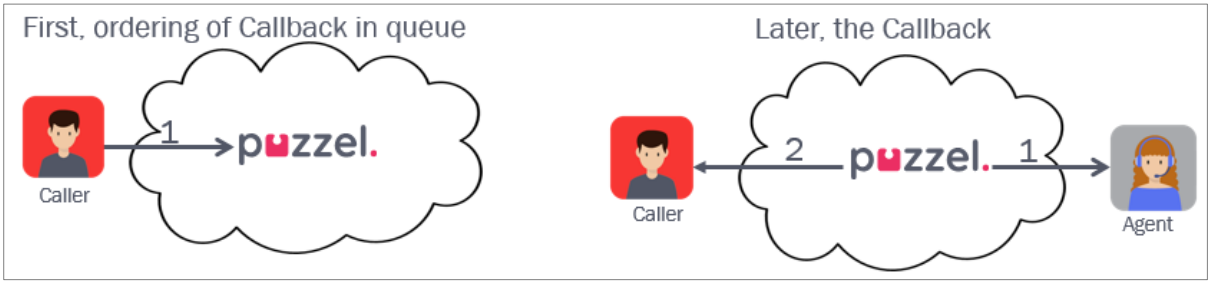
It might be interesting to look for Dialler calls (conversation events) with result_code = e and result_response = 1 or 404. If you find several conversation events with the same destination number and result_code=e and result_response 1 or 404, this phone number should be corrected in the source system the Dialler lists are generated from.

Some useful links:

- https://en.wikipedia.org/wiki/ISDN_User_Part
- https://en.wikipedia.org/wiki/List_of_SIP_response_codes
- https://www.dialogic.com/webhelp/IMG2020/2.3.0/WebHelp/cause_code_map_ss7_sip.htm

2.6 Callback in queue (aka 'CiQ')

When a caller orders Callback in Queue (often referred to as "CiQ"), this results in 2 or more sessions (with the same call_id).



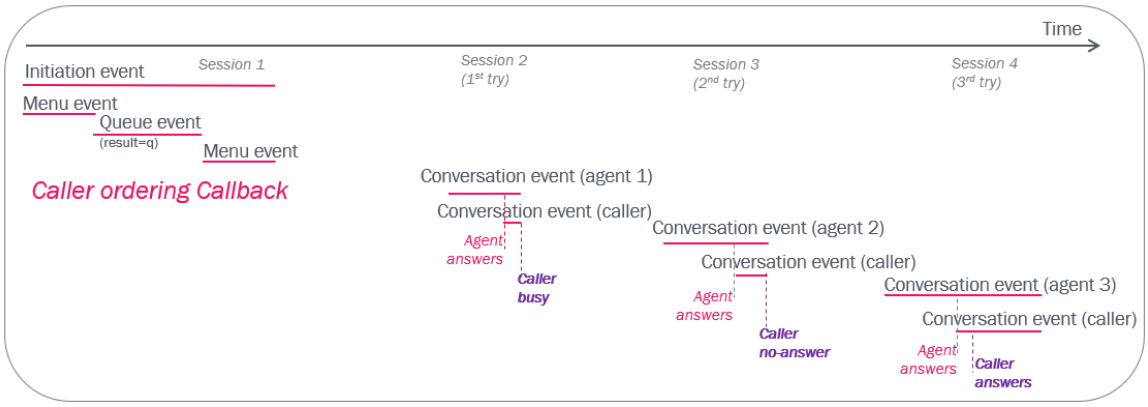
First, a session for ordering the CiQ, which lasts until the caller hangs up. Then a new session when Puzzel calls an agent and the one that ordered CiQ.

Please note that when events for a new session for an existing call_id are ready to be inserted into the database, all existing events for this call_id will be deleted, and then all old and new events for this `call_id` is inserted with new `rec_id` and new value in `dte_updated`.

It is possible to configure that Puzzel calls the one that ordered CiQ first, and then call an agent. Since very few customers use this, the examples in this document are based on calling agent first.

Example with callback answered on the first attempt:

If the customer does not answer, Puzzel might do 1 or 2 more tries (configurable).



All events for all sessions related to one CiQ will have the same `call_id`.

The Queue event will have result=q for a caller that ordered Callback.

Only Conversation events following a callback (or Callout or Dialler) will have a value in field **ciq**:

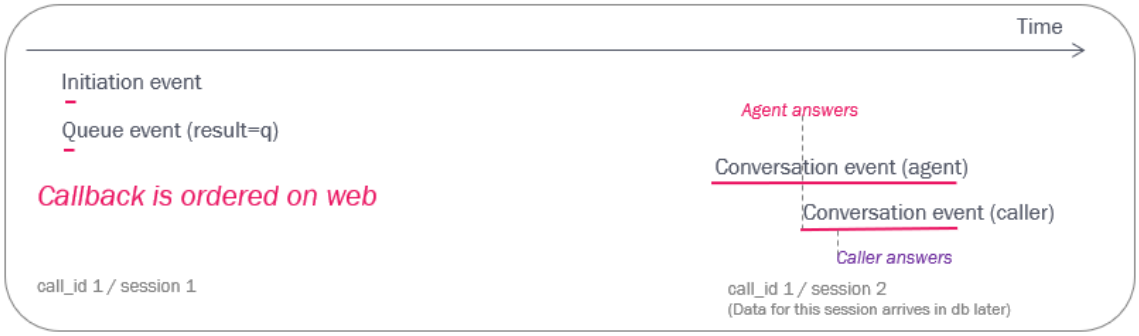
- *ciq=**a** for call to the agent*
- *ciq=**c** for call to the number callback is ordered to (destination or contact)*

When a call to the one that ordered callback results in busy, the answered call to the agent in this session will have a very short speaktime. When a call to the one that ordered callback is not answered, the call to the agent in this session will typically have 15-30 seconds speaktime.

If a caller starts to order Callback but aborts before the Callback ordering is finished, the caller may return to the queue (or hang up). This will result in one Queue event with result a (not q) and another Queue event if the caller actually returned to queue.

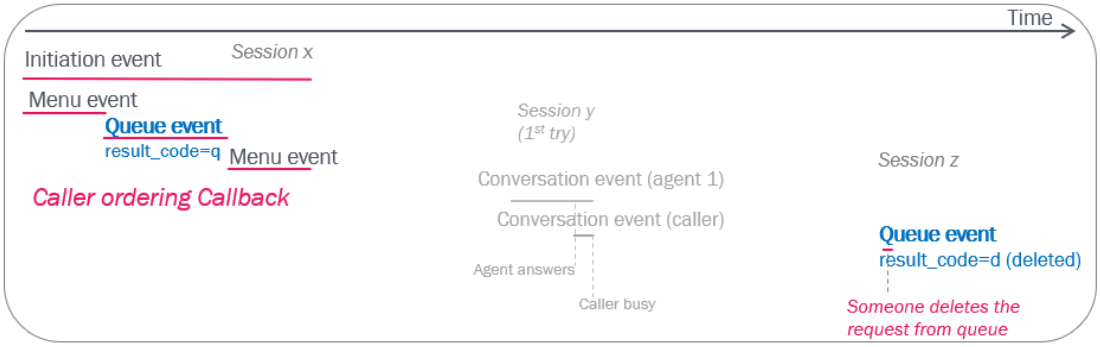
2.6.1 Callback ordered on web

If Callback is ordered by filling in a form on a web-page, this results in an Initiation- and a Queue event, both with duration 0 (or 1 sec). As for callback ordered by phone, the following Conversation events will arrive in the db later, that is, after the calls are done:



2.6.2 Delete Callback request from Queue

If a Puzzel administrator (with special access) deletes a Callback request from queue, this will result in a new Queue event with result_code=**d** (deleted) in addition to the already generated queue event with result_code=**q**. If you want to report total number of requests to a queue, you should count queue events with result_code ***unlike*** d.

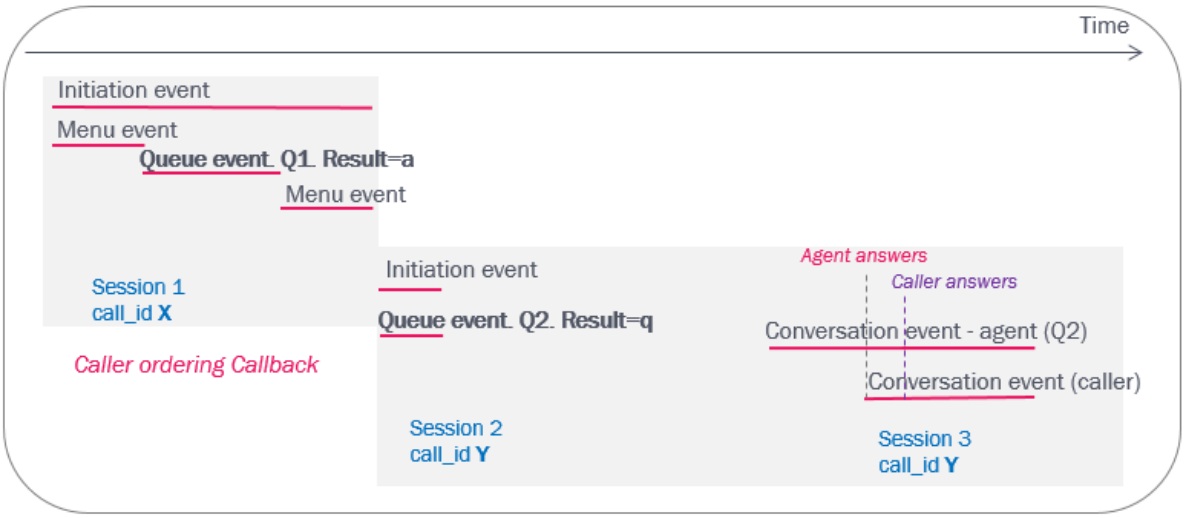


Please note that result_code=**d** means deleted for Queue events and default for Menu events. See also Delete email from queue.

2.6.3 Callback to another queue (non-standard)

The standard callback solution is that the caller orders Callback in queue to the queue he/she was waiting in. However, some customers require that callbacks are placed in a separate queue. When this is configured, this results in different events than for the standard solution:

- *Firstly, the initial session will have a queue event with result a instead of q, since callback is not ordered to this queue.*
- *Secondly, the following sessions will have a new call_id since this is a "new" ordered (web) callback to queue 2. Example:*



In Puzzel statistics this will be reported like this:

Total overview:

2 Incoming calls and 1 (or 0) answered

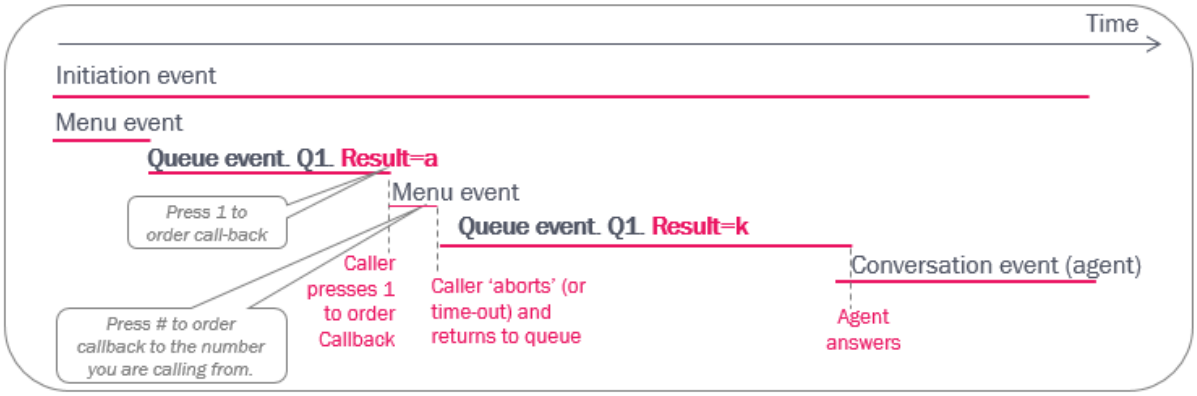
Details per queue:

Queue	Incoming calls	Total calls	Callbacks ordered	Exiting queue	Answered (excl callbacks)	Answered callbacks
Queue 1	1	1	0	1	0	0
Queue 2	1	1	1	0	0	1 (or 0)

2.6.4 Callback order not completed

If a caller (temporarily) leaves the queue to start to order Callback, but does not finish the order callback process, the call is usually sent back to queue.

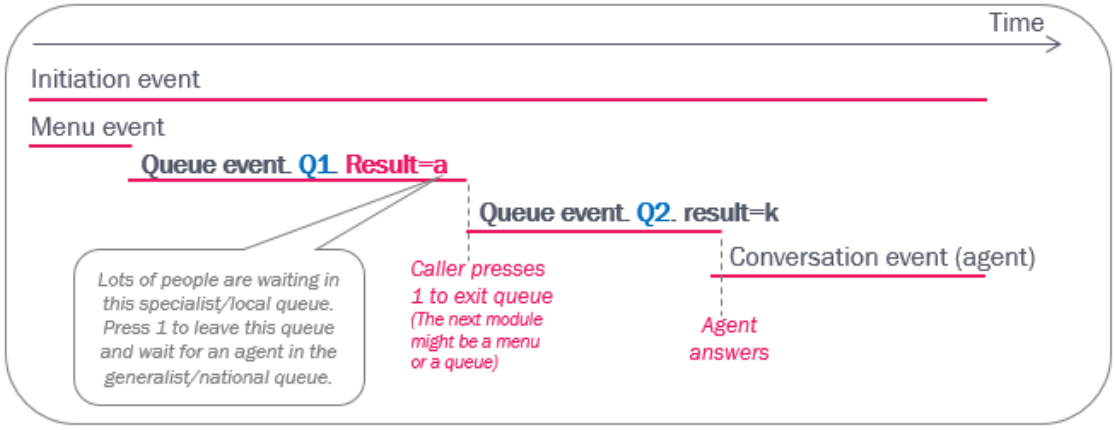
This results in a queue event with result=a, one or more menu events and **a new queue event**, which will have result=k if the call is later answered (or h if caller hangs up in queue).



If a caller starts ordering Callback but hangs up in the menu before the order process is finished, the queue event gets result=a (abort) and the menu event gets result=h (hang-up). If the caller does not press the needed digit(s) to confirm callback, the call might also be terminated.

2.6.5 Interrupt from queue

If the Puzzel solution offers callers in queue to leave queue (not order Callback) by pressing a digit, and the caller presses the digit, the call leaves the queue the queue event will have result=a.



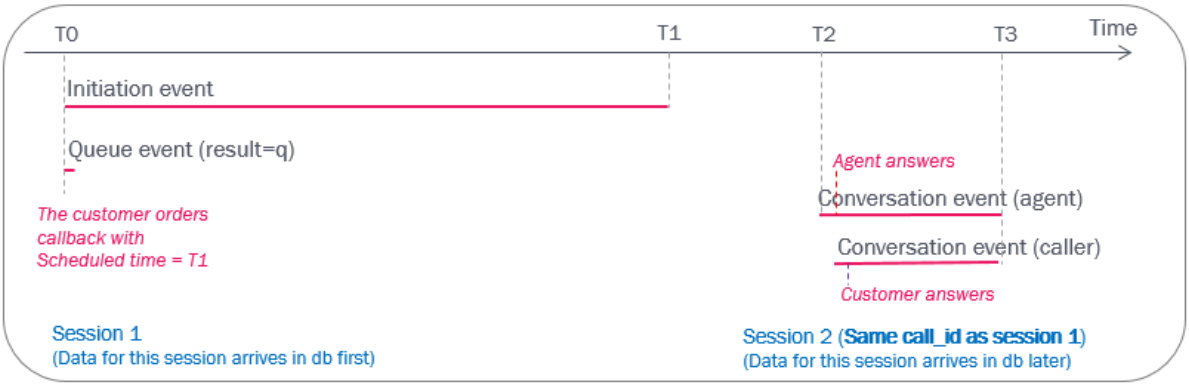
The call is routed to the defined module (e.g. another queue) following the interrupt exit.

2.6.6 Scheduled Callback ordered on web

If the customer that orders callback on a web page can specify a desired time to be called (the "scheduled" time), the Initiation event will have duration until the scheduled time!

Example (see illustration after the bullet list):

- *At T0, the end-customer orders callback («call me») with a specified time (T1)*
- *The request is put in the queue's «waiting room» until the scheduled time, and then it's moved into the queue with high priority*
- *The queue will then (at T2) call a ready agent. This may take some time if no agents are available at T1*
- *First, the events for the ordering of scheduled callback arrives in the Raw data db*
    - *Initiation event with duration T0 (order time) to T1 (scheduled time)*
    - *Queue event with duration 0*
- *After the calls to agent and customer has ended (T3, might be hours or days later than T0), the conversation events with the same call_id arrives in the db.*



2.6.7 Scheduled Callback ordered by phone

If a caller already in a queue can order a scheduled callback, the Initiation event will last until the ordering process ends.

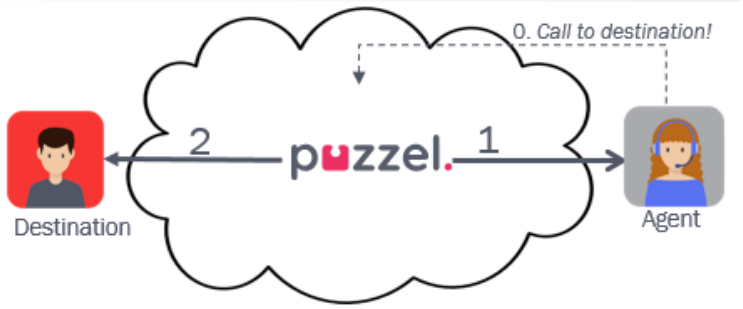
- *Between T0 and T1 the caller (in queue) orders callback with a specified time (T2)*

- *The request is put in the queue's «waiting room» until the scheduled time (T2), and then it's moved into the queue with high priority*
- *The queue will then call a ready agent (this may take some time if no agents are available at T2)*
- *First, the events (type i, m and q) for the ordering of Scheduled callback arrives in the Raw data db*
- *After the calls to the agent and the customer is ended (T3) the conversation events with the same call_id arrives in the Raw data db*

2.7 Outgoing calls

2.7.1 Outgoing calls from Puzzel (aka "Callout")

If a Puzzel agent wants to make an outgoing call through Puzzel, the agent can enter a phone number or select a catalog entry in the Puzzel agent application and click "*Call*" (or agent can click-to-call from another system that uses Puzzel web-services). This usually results in one answered call from the Puzzel platform to the agent (a conversation event with `ciq`=a and destination = agent's phone number) and then one call to the destination number (a conversation event with `ciq`=c and destination = the called number), answered or not.

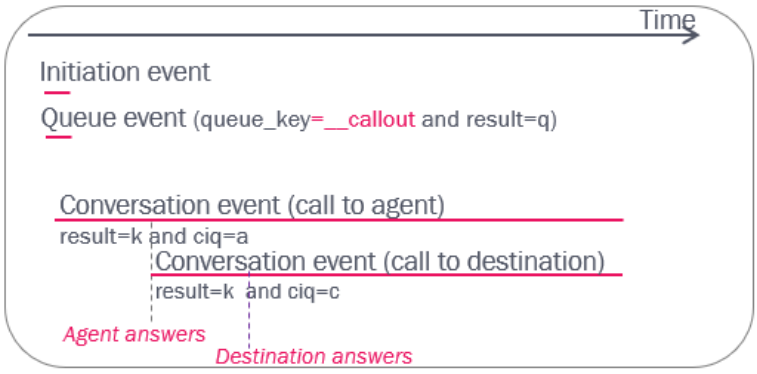


In addition to the 2 conversation events, this results in

- an Initiation event, with the called number as `source`
- a Queue event (in the outgoing calls queue, which usually has `queue_key` prefix '__Callout_') with result=q.

This is similar to the events for Callback in queue ordered on web. The conversation events (with another internal_iq_session_id but the same call_id) may appear in the database a bit later than the Initiation event and the Queue event.

An answered outbound call:



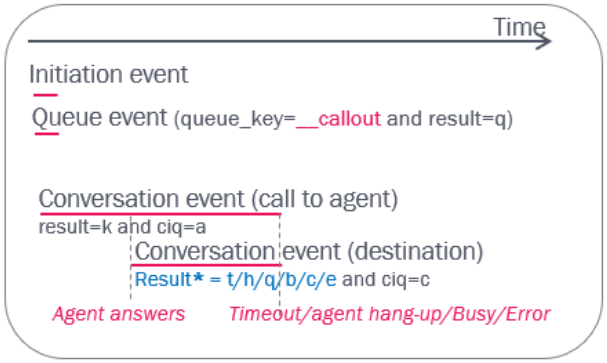
The call_events for an answered outgoing call (some columns are hidden here):

call_sequence	dte_start	duration_tot_sec	duration_speak_sec	dte_speak_start	source	destination	queue_key	agent_id	event_type	result_code	answered	ciq	wrap_up_sec	alert_ms	setup_ms
1	28.09.2016 11:39:01	1			21490xxx	81511569			i		1				
2	28.09.2016 11:39:01	0					__CALLOUT_		q	q		q			
3	28.09.2016 11:39:02	24	19	28.09.2016 11:39:07		98214xxx	__CALLOUT_	150674	c	k		a	60	1774	2838
4	28.09.2016 11:39:06	19	12	28.09.2016 11:39:13		21490xxx			c	k		c		6525	558

If the call to the destination number is **not answered**, the conversation event to the destination will usually have result ***h*** (=agent clicks *Hang up* while it's ringing) or ***t*** (ringing timeout for call to destination), and the speaktime in the Conversation event for the agent will usually be 15-30 seconds.

Please note that if the Puzzel agent clicks ***"Cancel"*** instead of **Softphone hang up** after having clicked *Call*, this results in an **extra Queue event with result=d (deleted)** in addition to the first queue event with result q (and conversation event(s)).

An unanswered outbound call:



* Different results in conversation event for call to destination
t if alerting timeout when calling contact
h if agent hangs up while the call to contact is in alerting phase
q if agent hangs up while call to contact is in set-up phase
b if the contact's number is busy (or contact clicks Reject)
c if call to contact times out during set-up phase
e if error in network or unallocated number

If the call to the destination number gives result=b (busy) or result=e (error), the Conversation event to the agent will have a very short speaktime.

About Display number

The initiation event's source is the Puzzel "access number" used for callout. If the agent selected a display number or if it is configured a display number unlike the access number, the used display number is not shown in Raw data.

2.7.1.1 Callout while in Pause

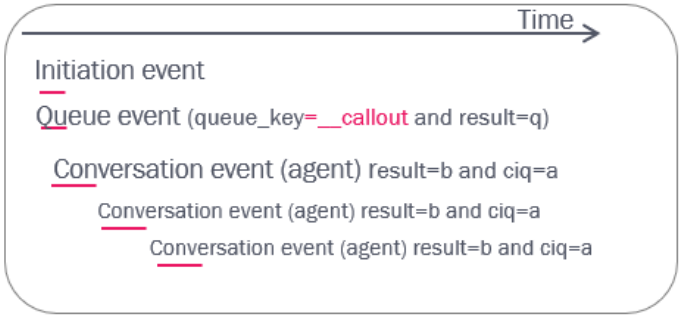
If the agent Calls out while in status Pause, the pause will be aborted (a *Return from pause* event is created) and the call is set up. If the agent does not want to receive an incoming call after the outgoing call is finished, the agent should click *Pause* when in status wrap-up. If *Pause* is clicked while connected, the agent will be in status [*Connected and Pause*] at the same time.

If the agent does not want to receive incoming calls but only do outgoing calls, the agent can log on to queue with a profile not containing incoming queues.

2.7.1.2 Error when calling agent that ordered callout

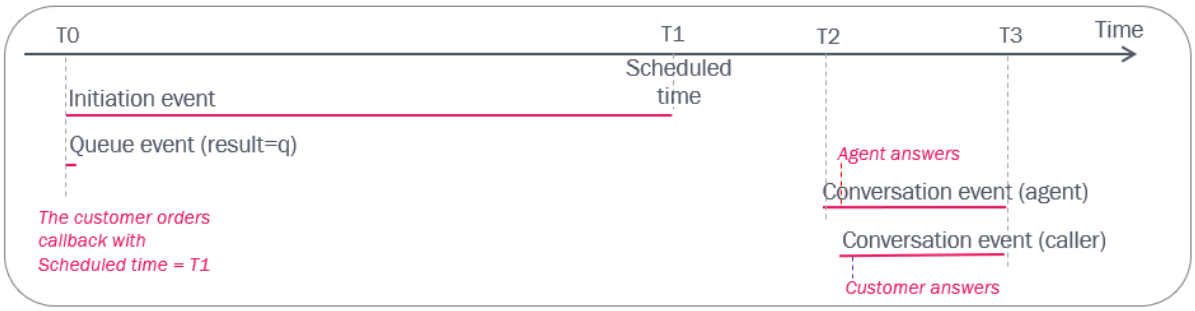
Please note that the call to the agent (that ordered callout) may result in busy, or in error/timeout if the agent logged on with wrong number or in cases with network congestion/error. If this happens, you might find a call_id with several conversation events to the agent (ciq=a) all with result b, c and/or e. If a call to the agent finally got answered (result=k), there will also be a conversation event to the called number (ciq=c).

When the agent clicks *Call* to make an outgoing call to a destination, Puzzel first calls the agent, and then the destination. If the call to the agent results in busy or error, there might be several conversation events for this agent (and possibly no conversation event for the destination) for this call_id.



2.7.2 Scheduled calls

The agent can (if given access in the Puzzel agent application) order a scheduled call (a future outbound call). The agent enters the destination phone number and the date+time the call should happen, selects “to me” or “to any agent” and a queue. At the scheduled time, the relevant queue calls the agent (when ready) and then the destination number. If “to any agent” was chosen, the first ready agent on the relevant queue is selected. A scheduled call generates these raw data:



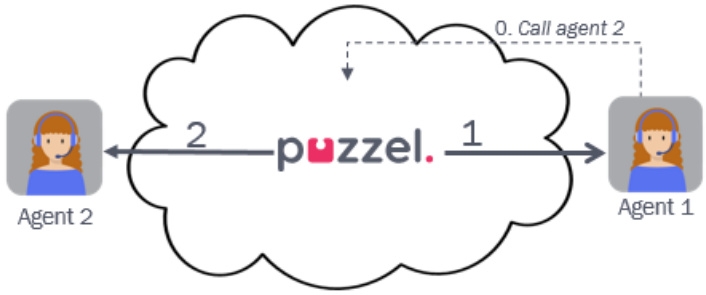
- At T0, the agent orders a Scheduled call with a scheduled time (T1) in queue X. An initiation event (duration from T0 to T1) and a queue event (duration = 0) is created and these records appear in the Raw data db shortly after T0.
  - The request is put in the queue's «waiting room» until the scheduled time, and then it's moved into the queue with high priority
  - If the agent is logged on at T1, the request is sent to the agent when he becomes Ready. If the agent is logged off at T1 or logs off before getting the call, the request is sent to another agent
  - The queue will (at T2) call the reserved agent when he's ready (or another agent if the reserved agent is not logged on or if the scheduled call was «to any agent»).
- After the calls to the agent and the customer has ended (T3, might be hours or days later than T0) these conversation events arrive in the Raw data db.

call_sequence	media_type_id	dte_start	duration_tot_sec	duration_speak_sec	dte_speak_start	source	destination	queue_key	agent_id	event_type	result_code	answered	ciq
1	1	22.11.2017 09:43	4595			98214836	81511569			i		1	
2	1	22.11.2017 09:43	0					q_sales		q	q		q
3	1	22.11.2017 11:27	52	48	22.11.2017 11:27		19500244731	q_sales	244731	c	k		a
4	1	22.11.2017 11:27	48	38	22.11.2017 11:27		98214836			c	k		c
5	1	22.11.2017 11:27	38							r	k		

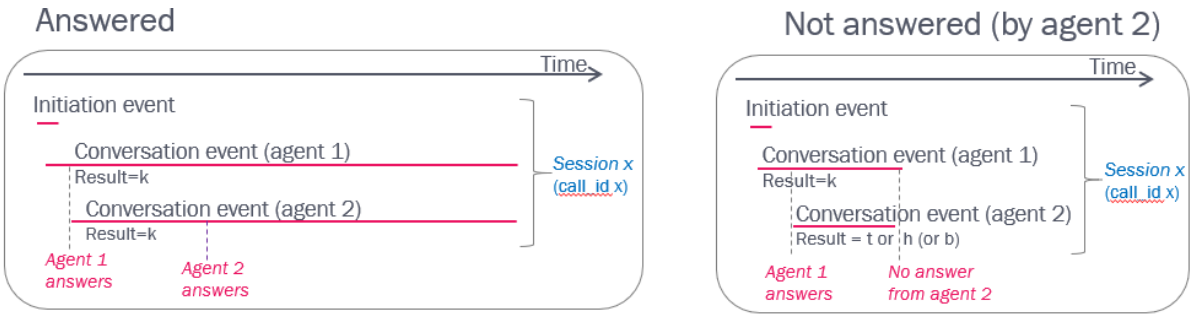
If an agent or an administrator deletes an ordered scheduled call before the scheduled time, a new queue event with result=d (deleted) with the same call_id is created.

2.7.3 Agent-to-agent calls

If agent 1 calls to agent 2, this will result in one initiation event, and one conversation event for each of the agents.



Please not that there will not be a queue event for an agent-to-agent call (only for ordinary outgoing calls), the conversation events will not contain any queue key, nor any value for ciq, and the initiation event's source will be "xxxxxxxx".



call_ sequence	duration _tot_sec	duration_ speak_sec	dte_speak_start	source	destination	service_ num	**queue_key**	agent_id	event_ type	result_ code	answered	**ciq**
1	31			xxxxxxxx	81511569	81511569			**i**		1	
2	31	30	12.07.2017 15:26:49		19500244736	81511569		244736	**c**	k		
3	30	25	12.07.2017 15:26:54		19500244731	81511569		244731	**c**	k		

For each agent-to-agent call, there will (from db version 1.6) be one record in the `call_event_extras` table with agent2agent=1 and a "link" to the agent-to-agent call's initiation event, so that you easily can identify these calls.

2.8 Dialler

2.8.1 General Dialler information

With Puzzel Dialler, the Puzzel administrator can upload a file with numbers to a Dialler campaign queue, and when this campaign is active and agents are logged on, the queue starts calling.

The Dialler Campaign queue(s) usually have `queue_key` starting with '**__Dial**', and Dialler calls usually have a dedicated service "access number", which is shown as the initiation event's `destination`.

Since several phone numbers (contacts) are put into the Dialler queue each minute, and a contact may wait in queue for minutes or hours before being called (depending of AHT, answer rate and number of agents), **it may take quite some time from the Initiation and Queue event (session 1) is delivered to the db until the conversation events (session 2) for the same call_id arrive in the db.**

In addition, **when a campaign is** (temporarily) **disabled, this will result in a queue event with result=d (deleted) in addition to the queue event with result=q for the contacts (call_ids)** that were in queue and now were removed from queue.

If a call to a contact's number results in busy or no answer, the Dialler usually calls this contact later. The number of call attempts per contact is configurable. For each new call to a contact, a new Puzzel request with a new `call_id` is created (with queue and conversation events).

It is possible to configure that one or more of the variables from the Dialler list (e.g. a contact_id and campaign_id stored in var1-var30) appear as call_variables in the Raw data. If this is done, you will find the same variable value (e.g. contact_id = 123456) as a call_variable record for all the calls made to the same contact.

Enquiry registration and Rescheduling done by agent

If the agent makes an Enquiry registration in the Puzzel agent application for a Dialler call, the selected topic(s) will appear in the Enquiry registration records.

If an agent **reschedules** a Dialler call (in the Enquiry registration part), this results in a new call to the same contact later. The rescheduling date/time and comment can be found in the enqreg_header record "belonging" to the call.

If the agent that rescheduled selected "to me", the Enqreg header record with the rescheduled time will have reserved = true (from db v1.6).

A rescheduled call is always "call agent first", also for a predictive mode (call contact first) campaign queue.

Please note that

- if an agent deletes his own rescheduled dialler call (in the Scheduled call tab), the contact will be "disabled" and the rescheduled call will not be done.
- if an admin disables a contact in the Dialler list (or if a contact is deleted/disabled using the API), the contact will not be called.
- we do not create any Raw data record for such contact disable/delete actions.

"Mark as unanswered" done by agent

If the agent is connected to a contact's answering machine, the call to this contact is technically answered and the conversation event will have result=k. If the agent (for whatever reason) clicks "*Mark as unanswered*" in the Enquiry registration part (possible from June 2018), the conversation event's result in raw data is not changed (still

result=k), but this makes the Dialler call this contact later (if more call attempts are left for the contact).

From db v 1.6 the new column `marked_unansw` in table `enqreg_header` will have value 1 if the agent clicked "Marked as unanswered".

Dialler list look-up for incoming calls to mark a contact as answered

If a dedicated display number is used for Dialler, it is possible to configure that for incoming calls to this number, Puzzel does a list look-up to see if the caller's number is in one (or more) Dialler list(s), and if so, this contact can be marked as "answered ok" in the Dialler list so that Dialler does not call this contact again. If this is done, it is possible to create **call variables** to register that this happened, e.g. dialler_marked_answered (containing phone number) and dialler_list_name. If you use this list look-up functionality and is interested in generating call variables for this, please contact Puzzel so that we can discuss a possible solution.

Error and Number not in use

The column **result_response** (from db v. 1.6) contains extra information for calls. Some of the conversation events for calls to contacts with result_code = e (error) might have result_response = 1 (ISUP - unallocated number) or 404 (SIP - Not found). If you find such calls, you should probably update these contacts phone numbers in your main/master source system for Dialler contacts.

See also Extra information for calls (result_response)

Mobile phone switched off or out of range

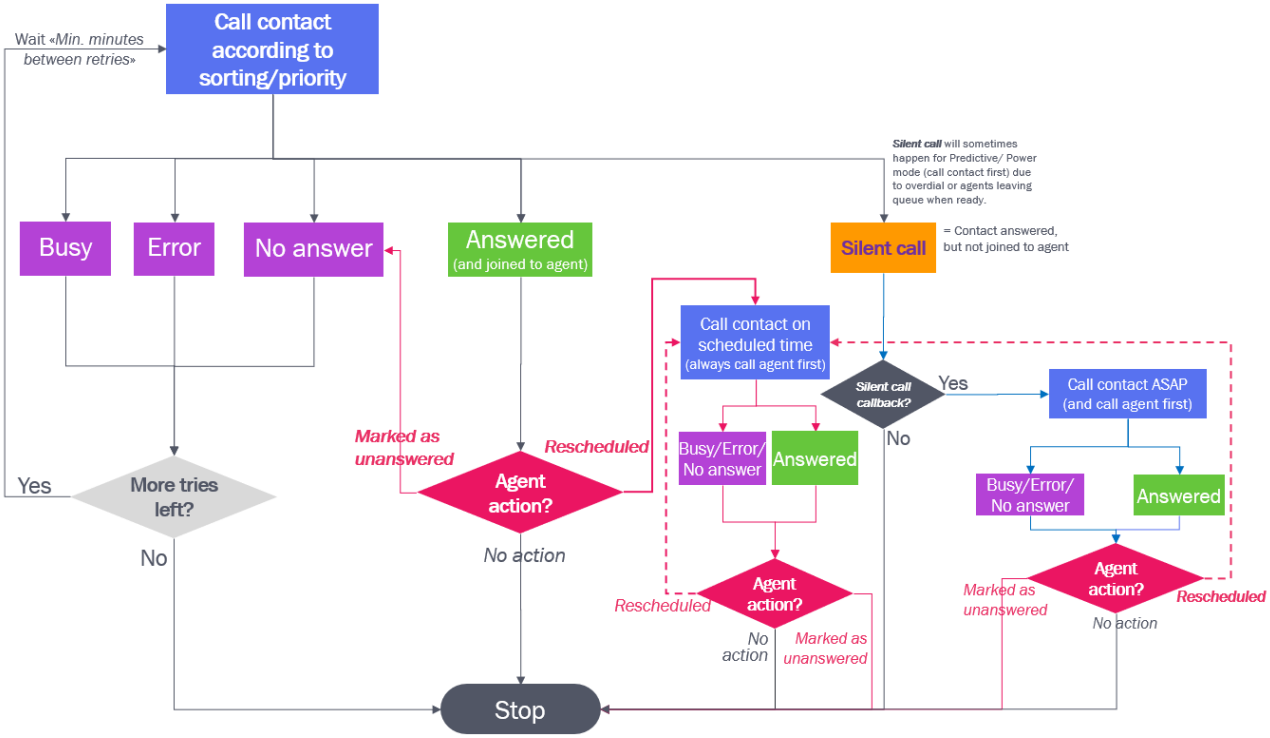
If the Dialler calls to a contact's mobile number and the mobile operator plays announcement "Mobile switched off…" without sending an answer signal, the conversation event for the call to this number will usually get result_code=t (timeout).

- On a predictive/power/progressive mode (call contact first) campaign, an agent will not be connected in such cases, since the Dialler does not receive an answer signal.
- On a preview mode (call agent first) campaign, the agent is on the line before calling the contact, but the message played might not be heard by the agent. Since the result for the call will be timeout, the contact will be called later (if more tries are left for this contact)

Call variables

It is possible to store info about a Dialler call in a call_variable, e.g. contact's external_id, Name or call attempt number. See Call variables

Dialler "flowchart":



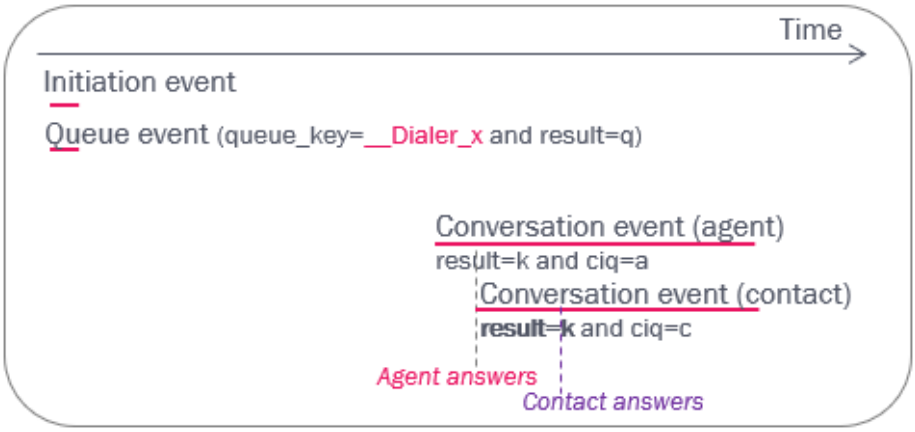
There are different Dialler modes that generates different raw data:

- *Preview mode (Call agent first)*
- *Predictive/Power/Progressive mode (Call contact first)*

2.8.2 Preview mode (Call agent first)

The queue first calls an agent (that gets screen-pop with information about the contact), and when the agent has answered, the queue calls the contact's number. This results in the same events as with ordinary outgoing calls.

An answered call:

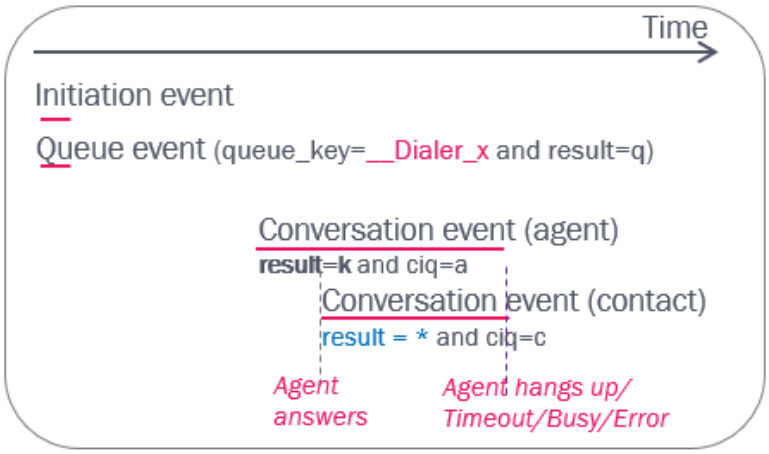


The queue event's result will be q, and the queue_key will usually start with "__Dialer".

Please note that the conversation events will appear in the db (much) later than the initiation event and the queue event (and with another internal_iq_session_id but the same call_id), since it may take several minutes (or even hours) from a list contact is put into queue until it is called.

An unanswered call:

When the agent answers, the contact is called, but the contact does not answer.

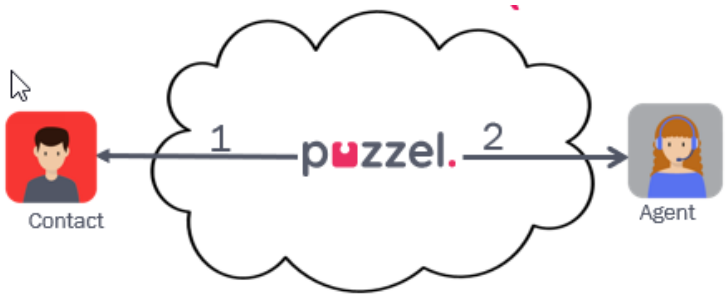


* The different no-answer scenarios (result in conversation event for call to contact) are:

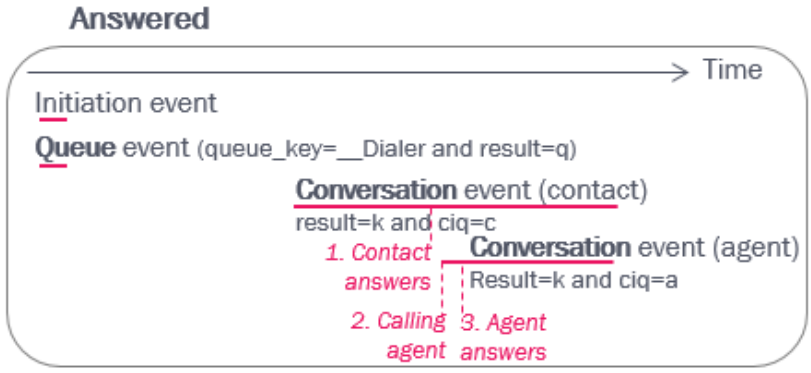
result_code	Description
t	alerting (ringing) timeout when calling contact
h	agent hangs up while the call to contact is in alerting phase
q	agent hangs up while call to contact is in set-up phase
b	the contact's number is busy (or called contact clicks Busy/Reject)
c	call to contact times out during set-up phase
e	error in network or number not in use. (result_response = 1 or 404 usually means number not in use)

2.8.3 Predictive/Power/Progressive mode (Call contact first)

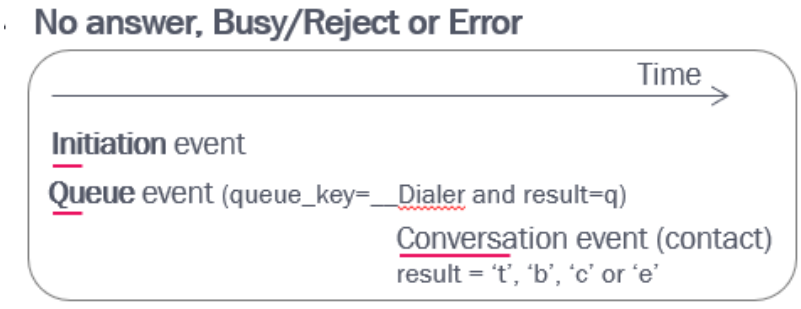
With these modes, the Dialler first calls to a contact's number, and only when a contact has answered, the queue calls a ready agent. For Predictive and Power mode, the Dialler sometimes calls more contacts than ready agents (aka overdial), and if more contacts than expected answer, there might be some silent calls (Puzzel disconnects the contact or contact hangs up before agent is connected). Silent calls might also happen without doing overdial, e.g. if a *Ready* agent clicks pause/log off, initiates an outbound call or if the agent's Softphone is unavailable.



The call to the agent results in auto-answer within ~1 second (given that the agent uses Puzzel Softphone).



If a call to a contact does not result in answer, the Dialler will not call an agent, but instead call the contact later if more tries are left.

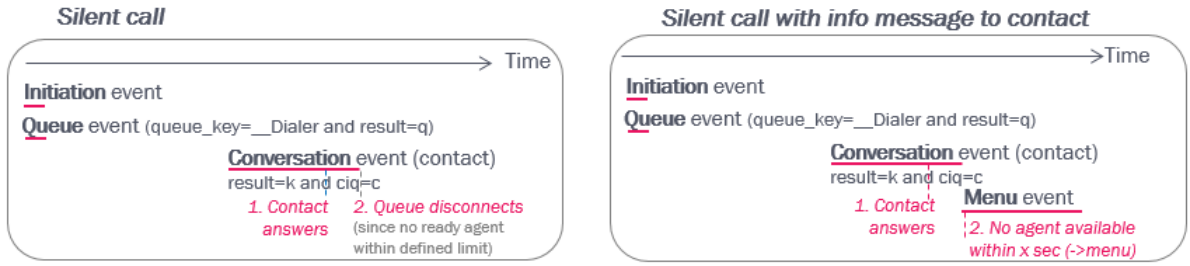


The typical result codes for **calls to contacts** (conversation events) for Progressive/Power/ Predictive mode Dialler queues:

result_code	Description
k	Call to contact is **answered** (by human or answering machine)
t	Alerting (ringing) **timeout** when calling contact
b	The contact's number is **busy** (or contact clicks Reject)
c	The call to the contact times out during set-up phase
E	Error in network or number not in use. (result_response = 1 or 404 usually means number not in use)

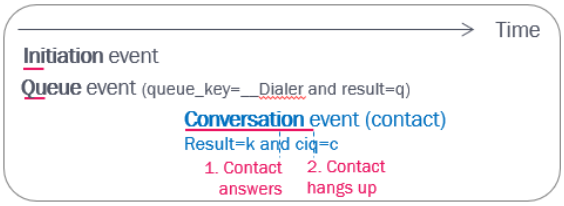
Silent calls

If a contact answers, and no agent is ready or answers within the defined thresholds, the Dialler will (optionally play a message for the contact and) disconnect the call. This is defined as a **"silent call".**



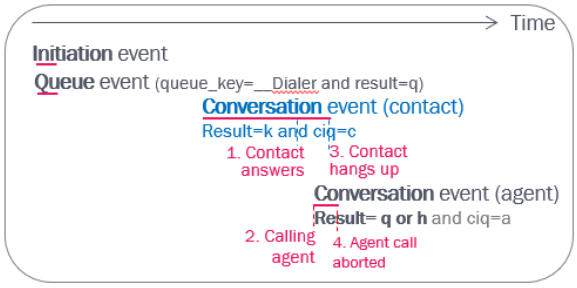
More Silent calls:

The contact answers and then **hangs up before agent is allocated**



Will only happen if *Abandon if no agent within sec* is set very high. Default is 2 sec and values > 4 sec is not recommend!

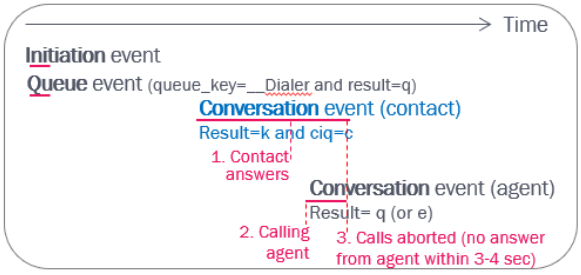
The contact answers and then **hangs up** while call to agent is in set-up (q) or alerting (h) phase



Since it takes ~1 sec from Puzzel allocates an agent until answer is received from the agent's Softphone, some callers might hang-up during this second.

- If *Abnadon if no agent within sec* is ≤3 sec, this will happen very seldom
- If *Abnadon if no agent within sec* is >4 sec (not recommend), this might happen more often.

The contact answers and an agent is allocated, but the **call is terminated due to no answer from the agent** within 3-4 sec



Might happen if an agent by mistake has logged on with an external phone number instead of Puzzel Softphone, or if an agent's Softphone is unavailable (e.g. If browser with Puzzel is closed)

A silent call is a call_id with a queue event with a Dialler queue_key and a conversation event with ciq=c (call to contact) and result=k (answered contact), but not containing a conversation event with ciq=a and result=k. (It might be a conversation event with ciq=a and result unlike k in this call_id).

A "call agent first" call_id on a "call contact first" queue?

- **Silent call callback:** It is possible to configure that "Silent call callback" should be done. If configured, a contact that experienced a silent call will be called shortly after a silent call, but this new call will be "call agent first" to make sure a live agent is on the line before calling the contact this time.
- **Rescheduling:** If an agent reschedules a call (on a predictive (call contact first) queue), the future call to this contact will be "call agent first"!

2.9 Group number / Unblockable agent

An *ordinary* Puzzel agent represents one person/one answering position, and one request is offered to the agent when the agent is ready.

An *Unblockable agent* (called *Group number* in the Puzzle Admin Portal) is used to represent a phone number the Puzzel solution calls to, and behind this (group) number there will be more than one person answering and possibly several queue positions in a local PBX. In Puzzel, the customer defines how many "lines" an unblockable agent/Group number represents, and Puzzel sets up calls to an unblockable agent as long as not all "lines" are in use.

An unblockable agent is represented in the Puzzel internal database by one *master agent* and several *slave agents.* There will be one slave agent per "line" defined for this unblockable agent. The master agent is used to log on/off, and this results in logon/log off of all the master's slaves.

Calls to an unblockable agent (Conversation events) will have a slave agent's `agent_id` in the tables `agent_events` and `call_events`. The relationship between slave agents and the master agent is found in the table `agents`.

2.10 Chat

2.10.1 Chat with live agents

If your Puzzel solution handles chat in addition to phone calls, things are more complex than phone only solutions. A Puzzel agent can handle only one phone call at a time, but several chats/written requests. Each connected/active written request is shown as a Dialog tab in the Puzzel agent application.

For each customer/user group or per agent, two important properties are defined:

a) **Maximum** total number of concurrent written requests: **X**
b) **Block phone** if number of written requests is greater than: **Y**

In addition, both *Max concurrent chat/social requests* and *Max concurrent email/task requests* can be defined (from January 2020), to e.g. allow total max 5 written requests, but only max 1 email but max 5 chats.

In the Puzzel internal database we create one "slave agent" per possible written request for the (master) agent. This is similar to Group number/Unblockable agent. If the agent can handle 4 written requests (X=4) there will be 4 slave agents for this (master) agent.

Examples:

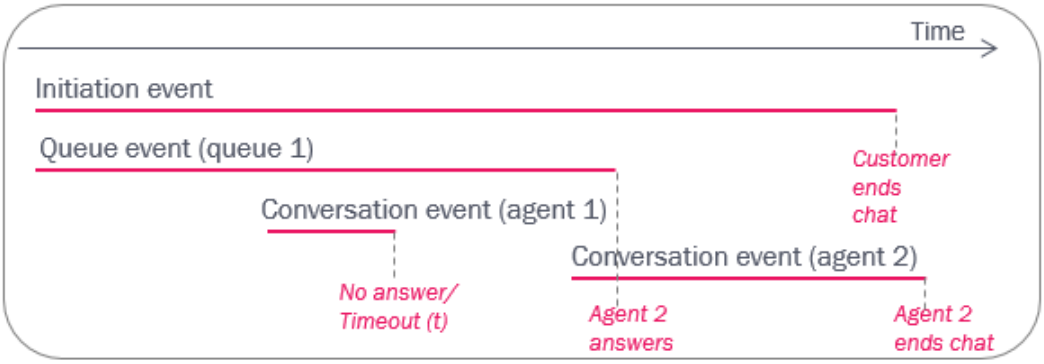
a) If Y is 0, the agent will not receive a phone call if he already has a written request.
b) If Y is 2, the agent can receive a call if the agent has maximum 2 written requests.
c) If X is 4, the agent can receive maximum 4 written requests, but if phone already is connected, new written requests are not offered

The number of active written requests is shown in brackets next to the (master) status, e.g. *Ready (0), Ready (1), Busy (2)* or *Connected (1).*

If several chats wait in queue and an agent who can handle e.g. 3 chats becomes Ready, the queue first sends one chat to the agent, and if this is accepted, the queue sends the next chat to the agent.

An incoming chat request is put in queue and offered to an agent. When the agent is offered a chat, this results in a Conversation event with one of the slave agent's agent-id (not the same id as is used when the agent is offered a phone or email request). To find the slave agent's corresponding master agent id (`chat_master_user_id`) you need to look in the table ***agents***.

If the agent that first receives a chat in Puzzel does not answer (result_code=t), the queue sends the chat to the next agent.

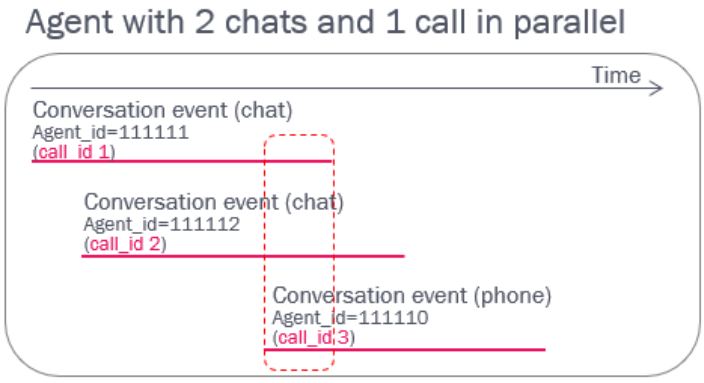


If the queue tries to send a chat (or any written request) to an agent that is **logged on** to queue but that has **closed his/her Puzzel agent application**, the queue will after a while send the chat (written request) to the next ready agent. The conversation event for the logged on agent whose Puzzel agent application was unavailable will have `result_code`=c.

If the person that initiated the chat ends the chat before the agent closes the chat tab, the Initiation event will have a finish earlier in time than the Conversation event finish.

If the agent finishes the chat before the chatter, the Initiation event will have a finish later than the Conversation event finish.

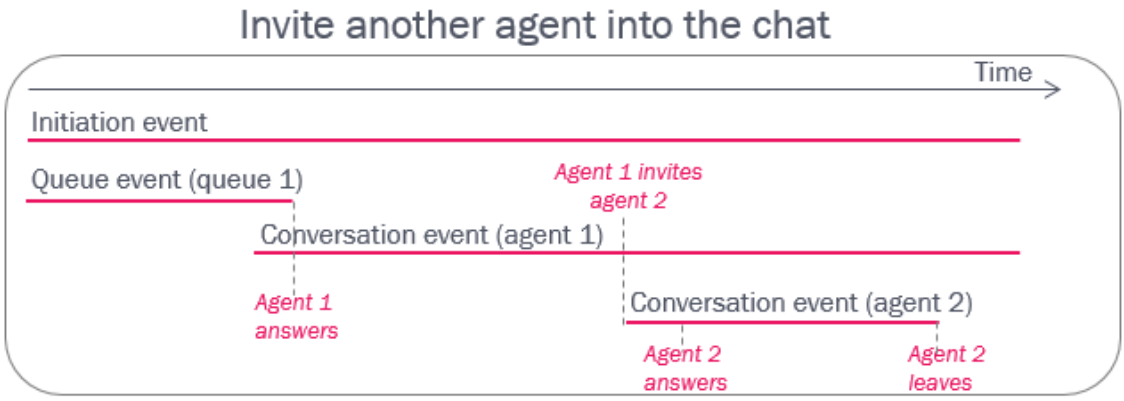
There might be more than one chat Conversation event for the same agent for the same time interval (in different `call_ids`), and there might be a phone Conversation event covering the same time period as one (or more) chat Conversation event(s).



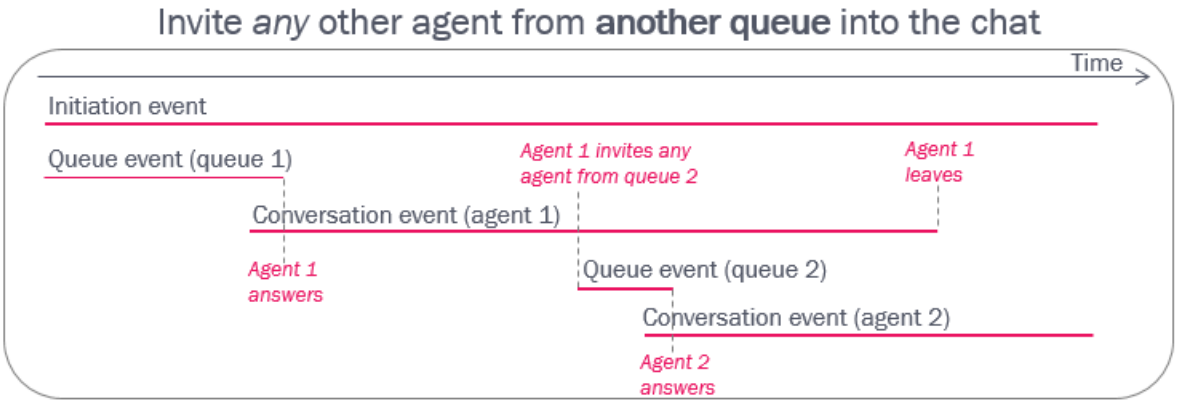
By looking in the table `agents`, you'll find that `agent_id` 11 and 12 both are «slaves» with the same master agent id, here 1.

Agent 1 can invite agent 2 into an ongoing chat with the end-customer. This will result in a new Conversation event for agent 2. (Today, this event for agent 2 contains the same

queue_key as the event for agent 1, but the queue_key in agent 2's conversation event might be removed in a future release of Puzzel). Agent 1 or agent 2 may leave the chat first. If agent 1 leaves first, this is similar to a (consult and a) transfer, so the conversation event for agent 1 will have `call_transfer`=1. If agent 2 leaves first (agent 1 continues the chat), the conversation event for agent 1 will have `call_transfer`=0 (no transfer, this was only a consult).



The agent may also invite any agent from another queue (the first ready agent will be offered the chat). This will result in a new Queue event and a Conversation event for the agent that gets the chat from the queue.

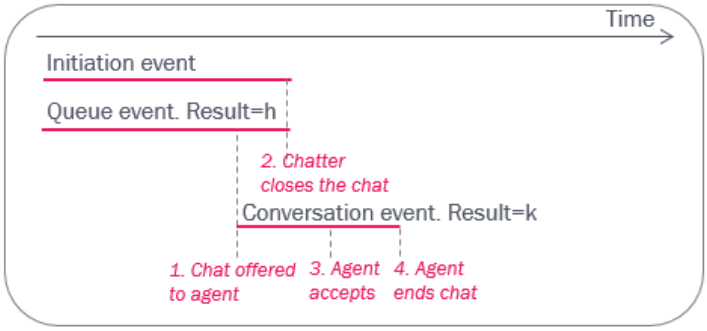


Please note that if you summarise the time an agent has spent answering phone (Speaktime + wrap-up) and answering chats in for example a 15 minute time period, you might end up with more than 15 minutes! This is because the agent may have answered several chats (and possibly phone) at the same time.

A very special case

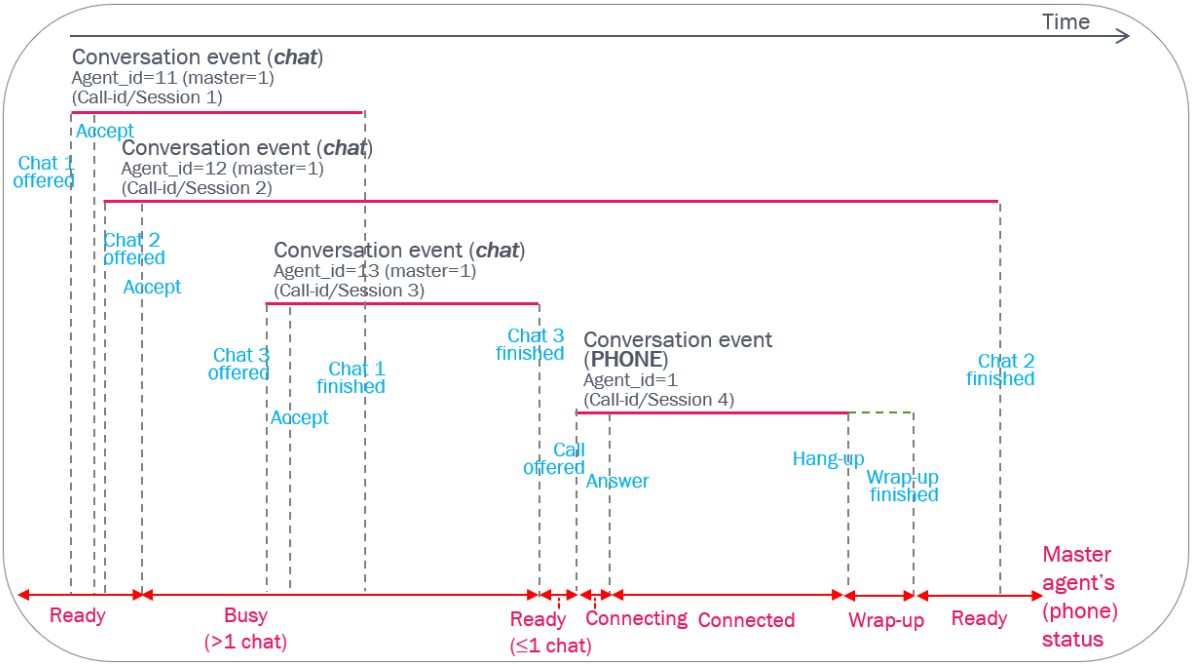
If the chatter closes/ends the chat after the chat is offered to an agent but before the agent has clicked 'accept', and the agent then accepts the chat, the agent will see that the chatter has disconnected, so the agent will quickly end the chat. This results in a

queue event with result h and a conversation event with result k, and a very short speaktime. This `call_id`'s initiation event will have `answered`=1.



Example: Several requests sent to one agent with this configuration:

- *Block phone if number of chats (written requests) is greater than: 1*
- *Max parallel chats (written requests): 3*



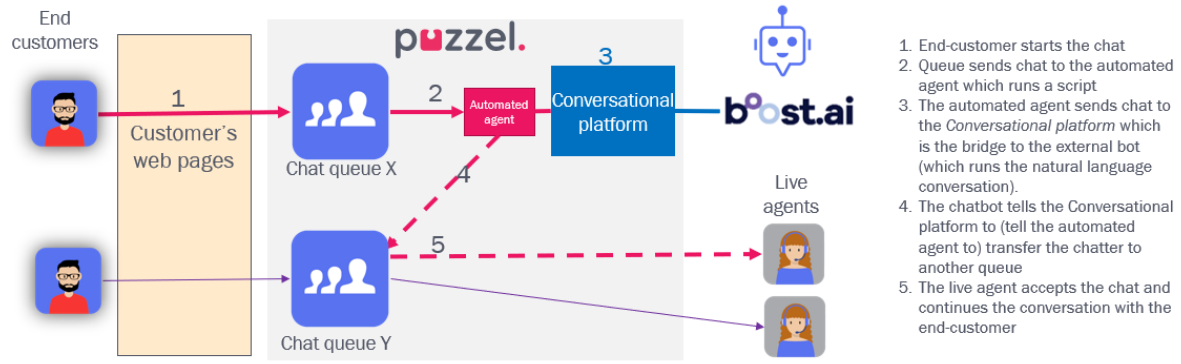
2.10.2 Automated agent and chat-bot

A chat queue can be answered by an automated agent (running a script) and possibly also a chatbot (e.g. boost.ai) instead of live agents. An end-customer that starts a chat pointing to a queue answered by automated agent(s) (and a chatbot) is sent to a Puzzel agent that runs a script. If a chatbot is used, the communication between Puzzel and the

chatbot-platform goes through the Conversational platform. Each Puzzel agent can handle max 8 concurrent chats, so you may need more than 1 Puzzel agent.

If the end-customer wants to chat with a live agent after having chatted with the automated agent/chatbot, he is transferred to another queue answered by live agents.

The queue and conversation events for chatting with an automated agent (powered by a chatbot) will be the same as chatting with live agents.

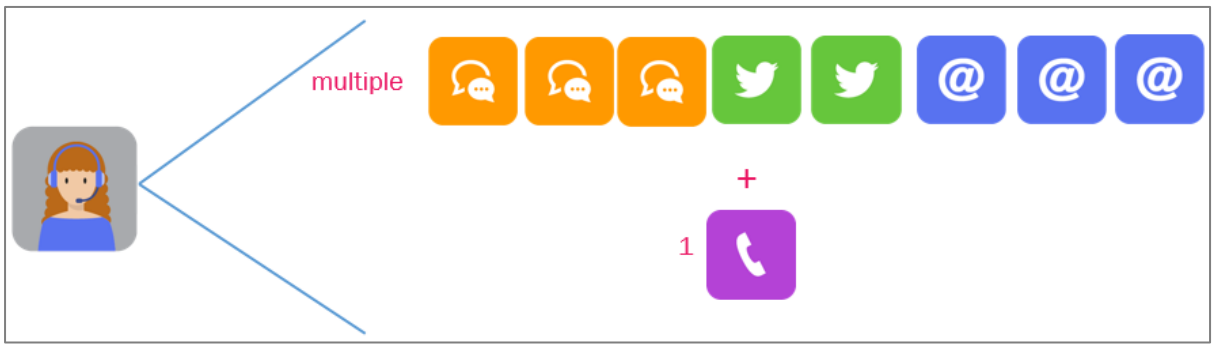


2.11 Email in queue

2.11.1 Email on 'master' agent vs email on 'slave' agents

If you have a Puzzel solution with requests using media type email (used for real email, e-tasks (any written request identified with a URL) or Scheduled tasks), the conversation events for email requests will include an `agent_id` that is different from the phone (master) `agent_id`.

Each phone master `agent_id` will have one or more 'slave' agents used for chat, email and social media requests. Ref the chapter about Chat.



The "old" Puzzel email solution was that email was treated like phone, that is, one agent could handle one phone OR one email at the same time (email on 'master' agent). If you

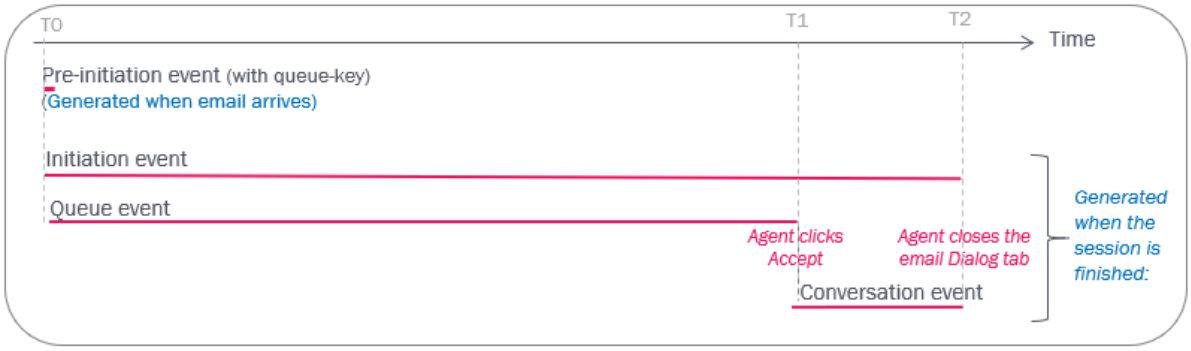
have a Puzzel solution like this (*Allow multiple emails* is OFF), the agent_id in the conversation event for email will be the agent_id of the agent 'master', that is, the same agent_id as used for phone. The plan is to decommission this old email solution in 2020.

2.11.2 Pre-initiation event

An incoming email to a Puzzel queue first results in one ***Pre-Initiation event*** (event_type=p), which contains timestamp, source, destination and `queue_key` for the email request, so that it is possible to count arrived emails before they are answered[1].

Then, when the email is answered by agent (or deleted from queue), Puzzel generates the ordinary records; Initiation event, Queue event and one or more Conversation events. The events in the second session will have the same call_id as the first session with the Pre-Initiation event.

The Conversation events for email may result in answer/accepted (result k), busy/reject (b), timeout (t) or setup timeout (c). Wrap-up is not possible for conversation events for email (requests handled on 'slave' agents). The agent closes the Dialog tab for the email in the Puzzel agent application when he/she has finished handling this email request.

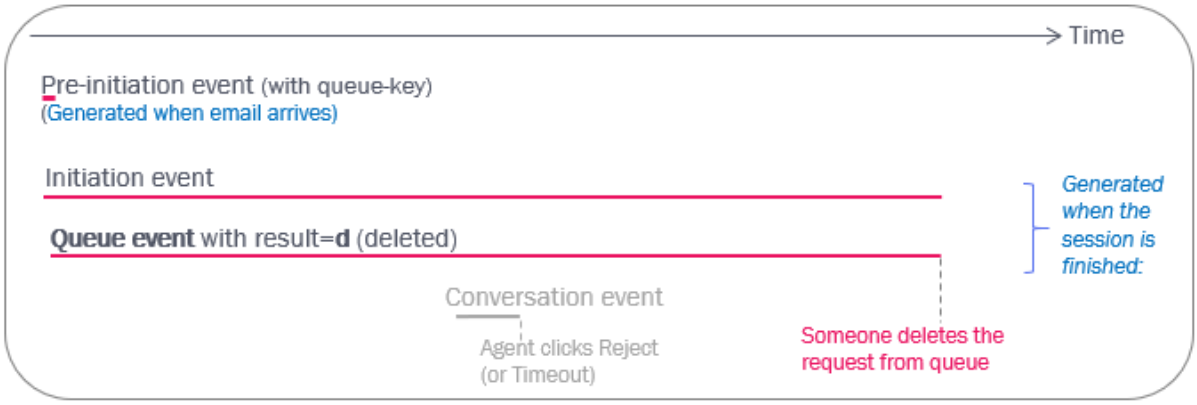


A special case:
If an email stays in a Puzzel queue longer than the allowed max time, the request (and the session) will be deleted. Today we don't generate more records than the already delivered *pre-Initiation event* in this case. This might be changed in the future.

[1] Exception: If a request with media type email is sent to Puzzel but is not put in queue because the queue is full (too many in queue already), a pre-initiation event will not be generated.

2.11.3 Delete email from queue and delete email connected to agent

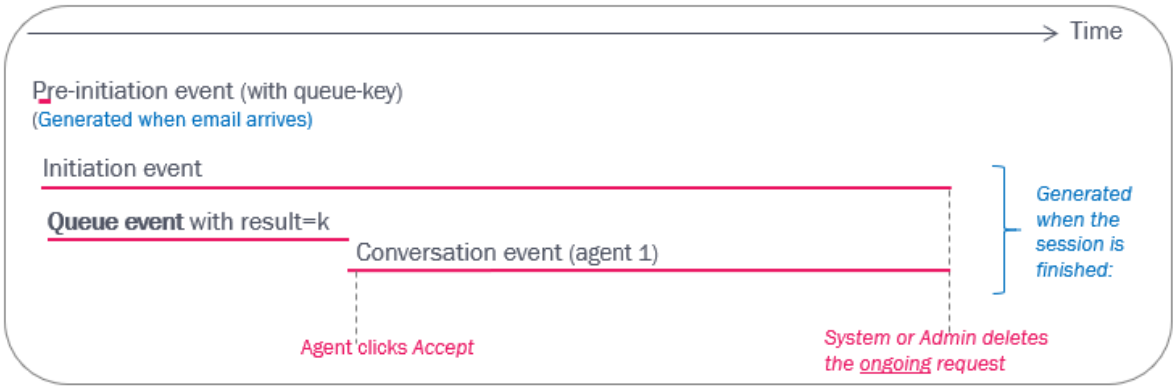
If a Puzzel admin **deletes an email from queue** (that is, before it's accepted by an agent), this results in an Initiation event and in a Queue event with result_code=d.



Please note that result_code=d means *deleted* for *Queue* events and *default* for *Menu* events. See also Delete Callback request from Queue.

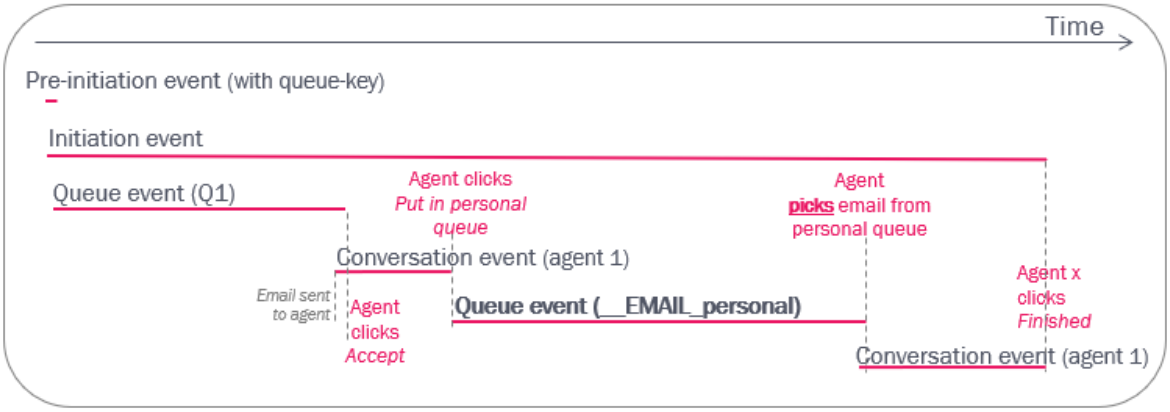
Delete "Ongoing" email request

- *If an email request is offered to an agent that accepts it, but the agent does not close the email tab in the Puzzel agent application within 72 hours, Puzzel will delete the email request, set the (slave) agent status back to available, and generate raw data records (initiation event, queue event and conversation event (with duration 72h))*
- *If an admin deletes an "ongoing email request", that is, an email connected to an agent (e.g. when an agent left for the day/week-end with an active email tab), the system will generate raw data records (initiation event, queue event and conversation event (with duration until the delete happened))*



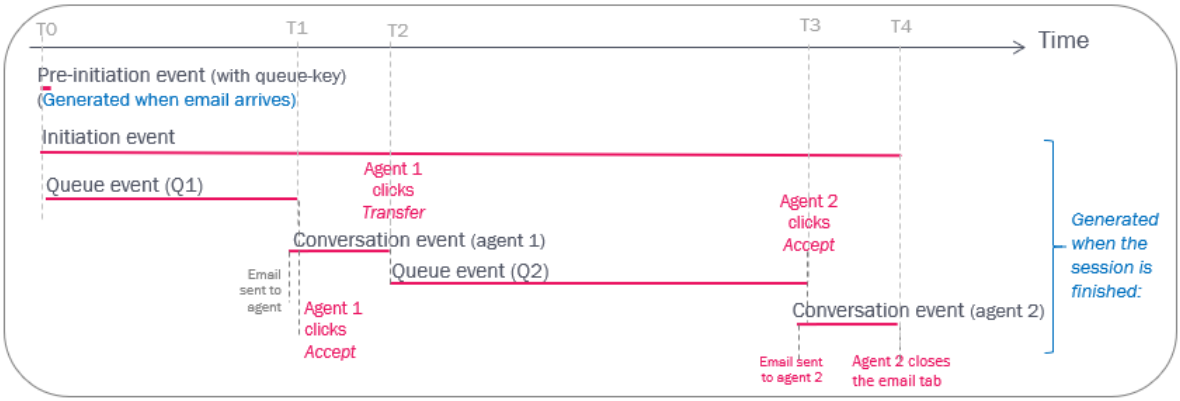
2.11.4 Put email in Personal queue

If configured, the agent can put an accepted email in Personal queue and later pick it up and finish it. This results in 2 conversation events for the agent separated by a queue event with `queue_key` «__EMAIL_personal».



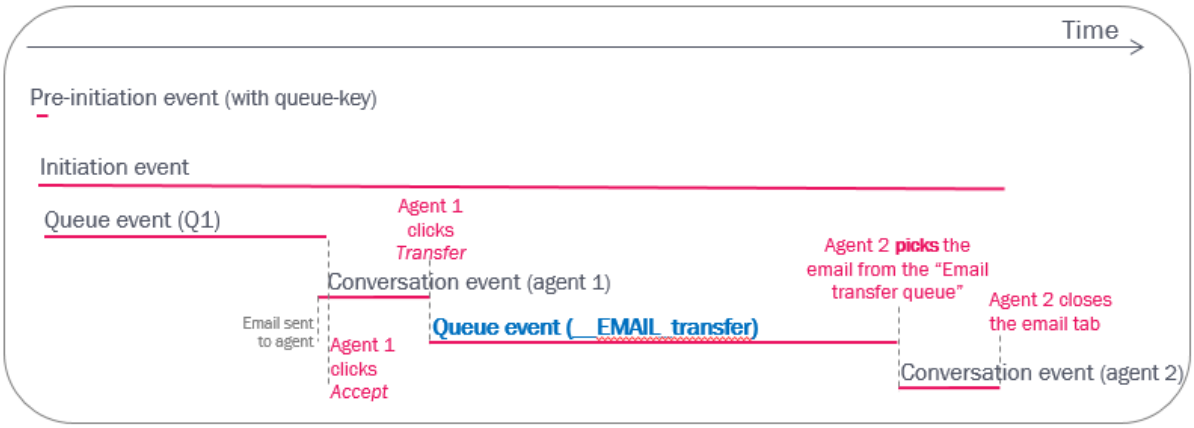
2.11.5 Transfer email to queue

If the agent accepts an email and then transfers it to another queue, this will result in a new queue event and then a new conversation event, similar to call transfer to queue.



2.11.6 Transfer email to agent

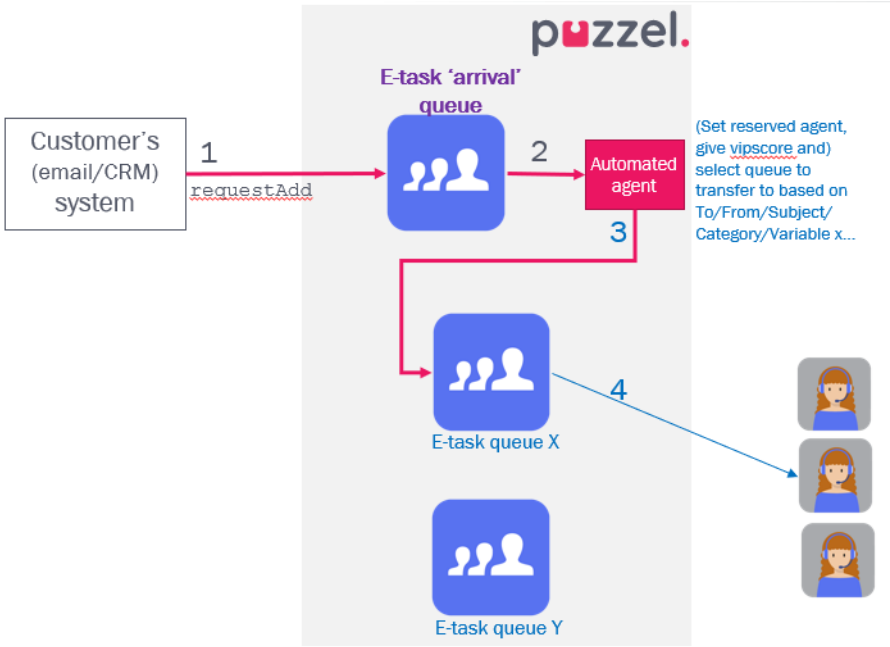
If configured, the agent can transfer an accepted email to another agent. Such a transfer will result in a queue event (`queue_key` = «__EMAIL_transfer») and then a conversation event for the agent that picked the email from EMAIL_transfer queue.



2.11.7 Email/Etask using the Puzzel "script engine"

A typical solution:

1. An etask/email request is sent to Puzzel, and Puzzel sends it to the "arrival" queue
2. The E-task arrival queue immediately sends the request to an *automated* agent, which runs a script (analysing information).
3. The *automated* agent transfers the request to another queue
4. When the request is «first» in this queue, it's sent to a live agent, who accepts it and does the etask/answers the email



The raw data events will be similar to the events for emails handled by live agents.

2.12 Scheduled task

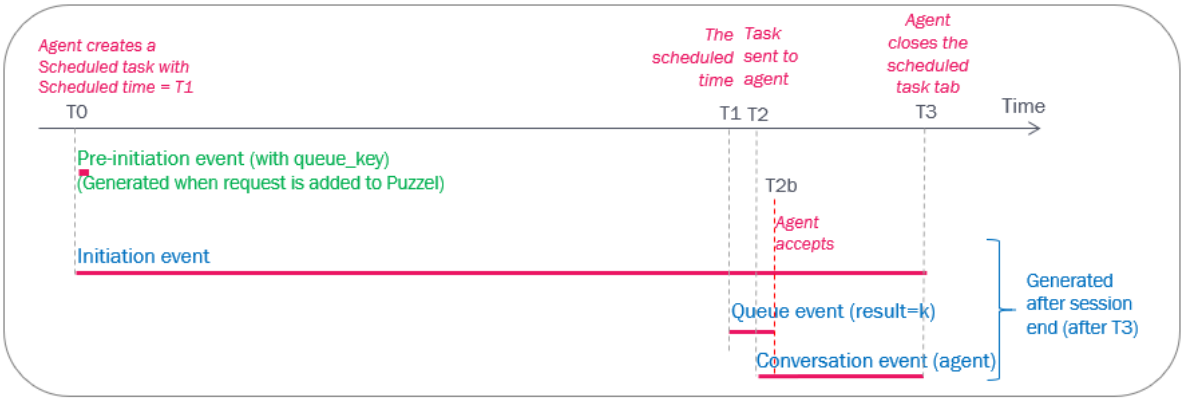
Agents can add scheduled tasks into Puzzel from January 29th 2020 (if given access). A scheduled task is like an email with a scheduled time, and the described task can be to do something or call someone.

The Raw data events created are similar to the ones generated for ordinary emails, that is, first a pre-initiation event, and then when request is finished, an initiation event, a queue event and conversation event(s).

In addition, we create (call_variable) records for scheduled_time, subject/title, Contact name, Phone number, contact_id. These “call_variables” for Scheduled tasks (and email and chat) are from January 29th 2020 shown in the Puzzel Admin Portal’s Archive, but not yet in Raw data. The plan is to include these variables for email/chat in the call_variables table in Raw data in the next raw data release.

The agent can create a Scheduled task when logged off, logged on, when in pause or when connected. The Scheduled task can be to himself, to another named agent or to any agent on queue x. The task is sent as a request to agent at the scheduled time (if agent is ready), and when agent closes the task request tab in the agent application, the raw data is created.

The queue event’s Start is at the Scheduled time, and duration is until agent accepted the request in the agent application.

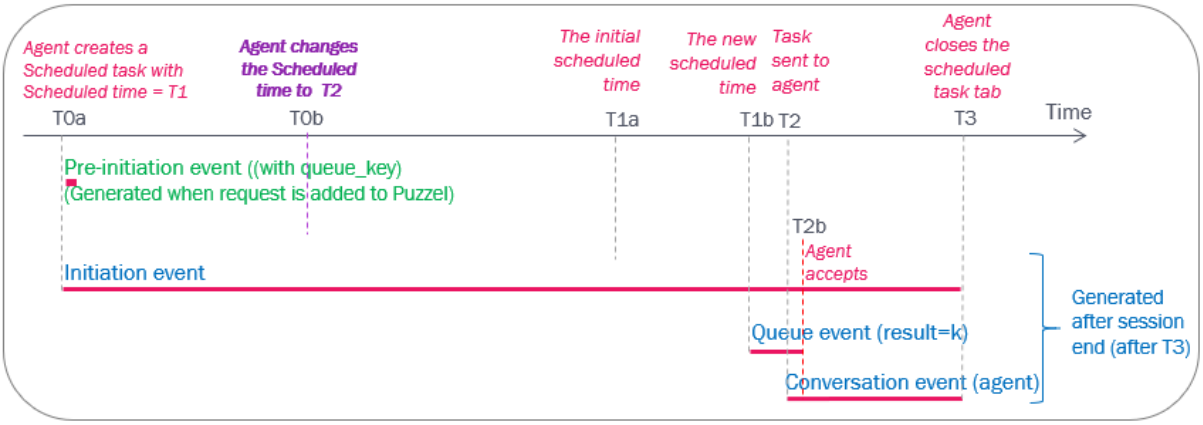


The (email) task request's conversation event ends when the agent closes the task request tab in the agent application, as for any written request.

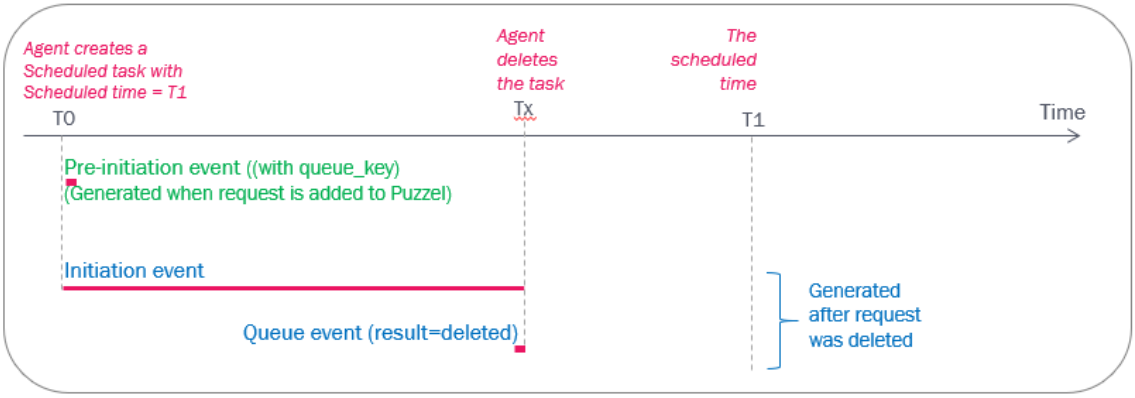
If the agent **changes the scheduled time** and/or other variables in a Scheduled task, the data delivered when the task is done will be correct.

Example:

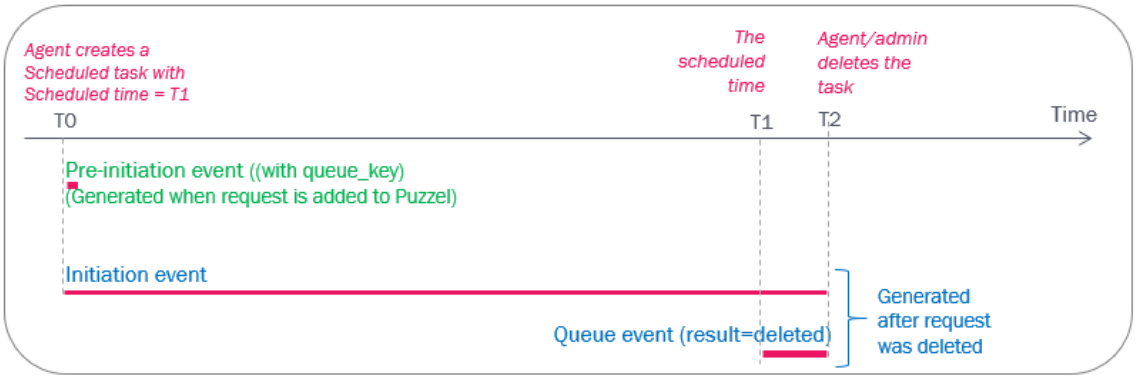
- At T0a, the agent creates a scheduled task with a scheduled time=T1a, but at T0b the agent changes scheduled time to T1b
- At T1b the request is put into queue, and then at T2 it's sent to agent
- First (at T0a), a pre-initiation event is created
- When the agent has closed the task request tab (T3), Puzzel creates the Initiation event (length until T3), Queue event (start at T1b) and Conversation event (and in a future release also the call_variables)



If a scheduled task is **deleted before the scheduled time** (when it is shown as "scheduled"), a queue event with no duration and result=d is created, in addition to the initiation event and some call_variables.

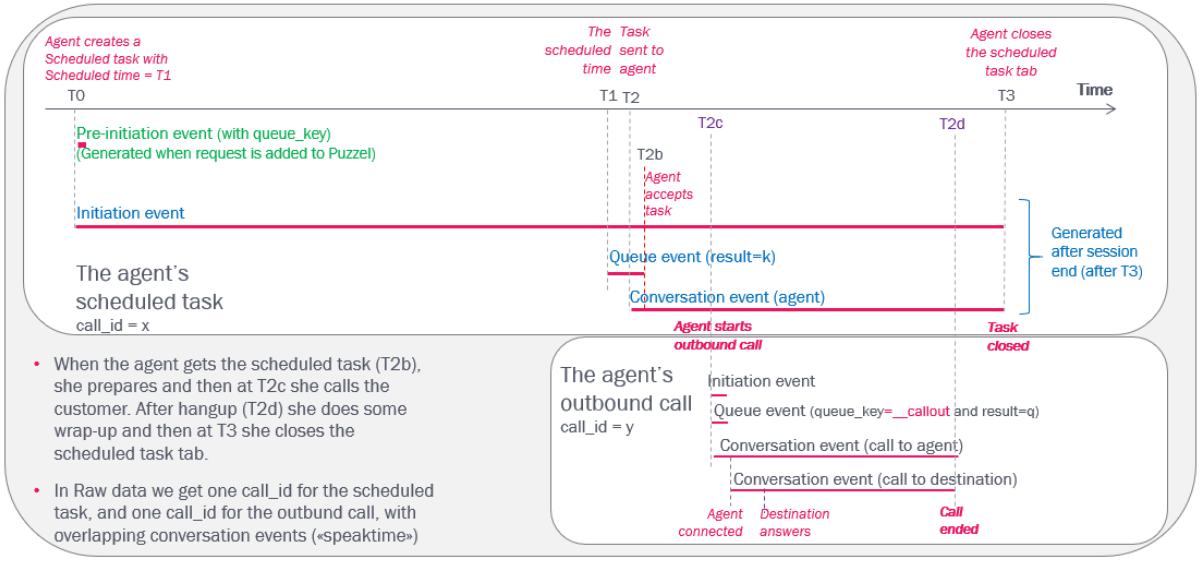


If a scheduled task is **deleted after the scheduled time**, that is, when it is shown as in queue but not yet sent to agent (e.g. because the reserved agent is not available), a queue event with start at scheduled time, duration until it was deleted and result=d is created, in addition to the initiation event.



If the task is to *call* someone

If the task is to call someone, the agent can make the outbound call while having the scheduled task request open, and after the call has ended, the task should be closed.



Planned functionality in agent.puzzel.com and next raw data db version.

- If the agent from an ongoing call's tab creates a scheduled task, the scheduled task will contain the call's session_id as related session.
- If the agent from an open scheduled task request tab clicks "reschedule", a new scheduled task can easily be created. Such a new scheduled task will contain the old scheduled task's session_id as related session.

2.13 Social media requests

Social media requests were handled as media type email until June 2016. From July 2016, a new media type for social media requests was introduced. All customers with Social media requests in Puzzel was migrated to the new solution using media type Social during summer of 2016.

From db v1.5.0.0, new records in call_events related to media type social started to use the new media_type_id = 7 (social).

Requests with media type Social is treated like chat (on 'slave' agents), so that the agent can have multiple social media requests and possibly one phone call at the same time.

A social media request connected to an agent can be deleted by admin (if agent is logged off), and a social media request connected to agent is automatically deleted if it's not finished after 10 hours connected to the agent.

A social media request will have a pre-initiation event (generated when request arrives in Puzzel platform) and an initiation event, queue event and a conversation event (or more), similar to email and chat.

The different types of social media requests covered by media type Social are:

- Facebook public posts
- Facebook private messages
- Twitter
- TrustPilot
- Whatsapp
- SMS*

* Incoming SMS to an access number/code word that is sent to a queue and answered by agent, so that customer and agent can "chat"/communicate using SMS.

There might be one or more queues with traffic for the different channels, but the data structure is the same:

- Pre-initiation event
- Initiation event
- Queue event
- Conversation event

2.14 Enquiry registration

If configured, the agent can do an Enquiry registration when he/she answers/has answered a call/request. The agent chooses a Topic (or more) within a Category (or more), and possibly a *Comment* and/or possibly a *Reschedule time* (for Dialler calls).

One Enquiry registration contains the following information

- *Timestamp*
- *Agent id*
- *Category (one or more)*
- *Topic (one or more)*
- *Mediatype*
- *Comment (if entered)*
- *Dialler reschedule time (if entered)*
- *Queue_key**
- *related_iq_session_id**
- *Marked as unanswered (option for Dialler only)*
- *Reserved to me (for Dialler rescheduling)*

* If configured that Enquiry registrations should be linked to the requests.

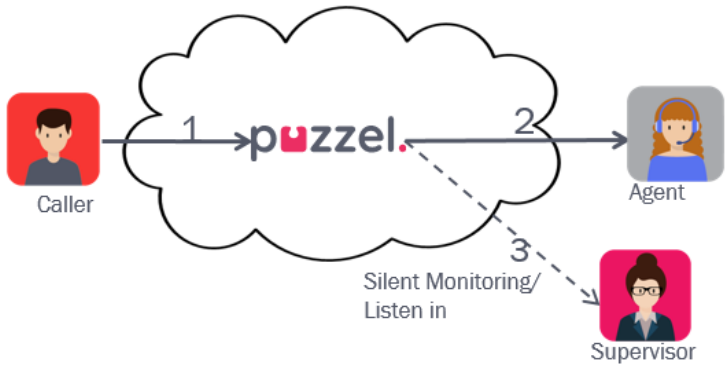
If the Enquiry registration record contains a `related_iq_session_id`, this id can be used to find the corresponding request's speak-time, wrap-up time etc. Since one incoming call can be answered by more than one agent (due to call transfer), there may be more than one Enquiry registration per call.

The Enquiry registration records are stored in the tables `enqreg_header`, `enqreg_category` and `enqreg_topic`, and there is also a view called `vw_enqreg_total` that "links" the information.

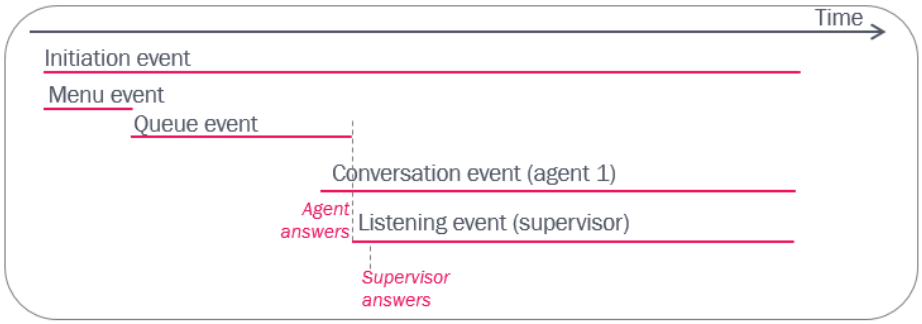
2.15 Silent Monitoring / Listen in

If configured in the Puzzel solution, supervisors can order Silent monitoring/Listen in on agent calls. The supervisor "orders" Silent Monitoring through the Puzzel Admin Portal, and the agent is notified in the agent application (and a biiip on the phone) when a monitoring call is connected, if notification is configured.

When the agent answers the call, the Puzzel platform calls the supervisor's/listener's phone number. The call to the listener may result in answer, timeout or busy.



A call to the listener has event_type=L (Listen in), so that they easily can be separated from ordinary calls to agents (with event_type=c (conversation)).



The supervisor (X) will usually enter his own phone number as the listener's number, but he may order listen in on an agent on behalf of another person (Y), and therefore enter this person's number as listener's number.

Until January 22nd 2019, the call_events with event_type = L only included the listener's phone number as **destination,** but no value in column `agent_id` (see row with call_sequence 7 below). From January 23rd 2019, the call_events with event_type = L also contains **the ordering supervisor's user_id** (xxxxx below) **in column `agent_id`.**

call_ sequence	dte_start	duration_ tot_sec	duration_ speak_sec	source	destination	service_ num	queue_key	menue_key	menue_ choice	agent_id	**event_ type**	result_ code	wrap_up _sec	alert_ ms	setup_ ms
1	2018-12-21 12:28:43.450	56		48142907	21496002	81511569					**i**				
2	2018-12-21 12:28:44.170	5				81511569		Welcome Demo			**m**	k			
3	2018-12-21 12:28:48.760	11				81511569		Demo Main Menu	2		**m**	k			
4	2018-12-21 12:28:59.780	9				81511569	q_support				**q**	k			
5	2018-12-21 12:28:59.870	40	31		19500244731	81511569	q_support			244731	**c**	k	28	8780	211
6	2018-12-21 12:29:08.960	31				81511569					**r**	k			
7	2018-12-21 12:29:09.540	30	25		04798214836	81511569				xxxxx	**l**	k		2169	2617

2.16 Recording events

In Puzzel there are 2 ways of recording phone calls:

1. ***Forced*** (aka *automatic*)
    - Forced on queue: All calls from a queue to agents are recorded
    - Forced on agent/user group: All calls to an agent or all agents in a user group are recorded
2. ***Manually by agent*** (agent starts and stops recording for ongoing call).
   The agent can only start a recording if this option is enabled, if the agent is the "active part" in the conversation (not a received consult call) and if a forced recording is not already running.

When a call is recorded, we record what the active agent says and hears, including any consult call(s). The new default from January 29th 2020 is that consult calls to agents are recorded, but for consult calls to numbers we only record the agent leg (and not the leg to the consulted party).
See more details here: https://help.puzzel.com/hc/en-us/articles/208234285-Call-recording

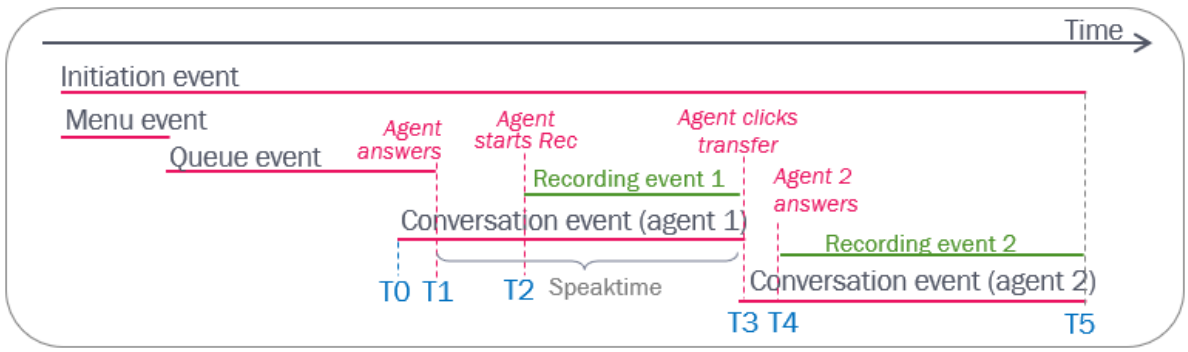
Consult call vs call transfer

- *When agent 1 makes a **consult** call to agent 2, we do not start a (new) recording even if recording is forced on agent 2.*
- *When agent 1 **transfers** a call to agent 2, we start a new recording on agent 2 if recording is forced on agent 2 (or if agent 2 clicks Start Recording)*

Forced recording vs manually started

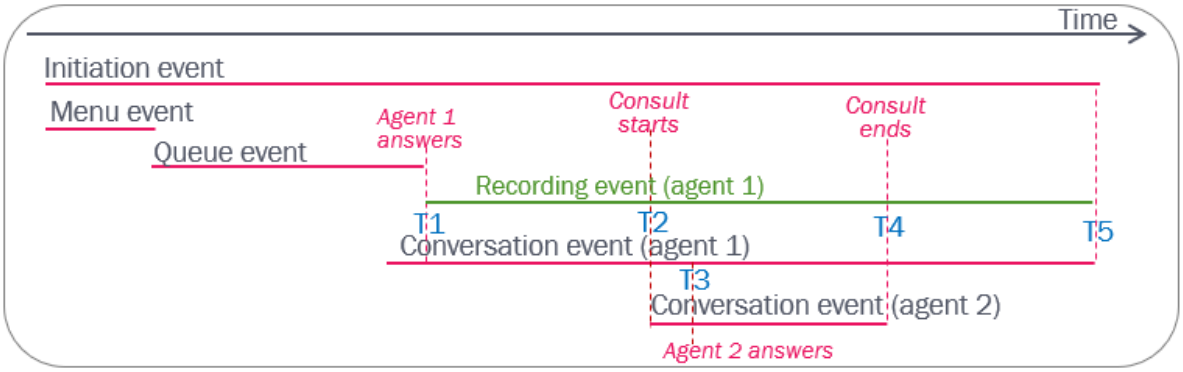
- *For **forced** recording, the Recording event starts when the speaktime in the conversation event starts (=Finish minus Speaktime)*
- *For Recordings started **manually**, the recording event starts after the speaktime starts, and it ends before or when the speaktime ends.*

Example 1: Transfer



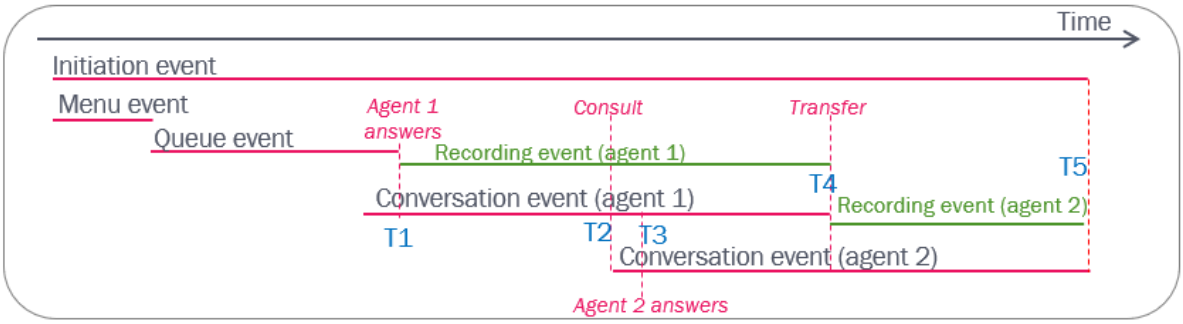
- *Agent 1 manually starts recording and then transfers:*
  Recording event 1 is from the agent clicks Start Recording until Transfer is clicked (T2-T3)
- *Recording is forced on agent2:*
  Recording event 2 is from where agent 2 answers until call end (=T4-T5)

Example 2: Consult



- *Recording is forced on queue or forced on agent 1: Puzzel records as long as agent 1 is connected and is the "active" part (T1-T5)*
- *Recording is forced on agent2: No new recording since agent 2 is never the "active" agent*

Example 3: Consult and transfer



- *Recording is forced on queue or agent 1: Recording event 1 is as long as agent 1 is the "active" agent (T1-T4)*
- *Recording is forced on agent 2: Recording event 2 is from where agent 2 is the "active" agent until call end (=T4-T5)*

Caller accepting/denying recording

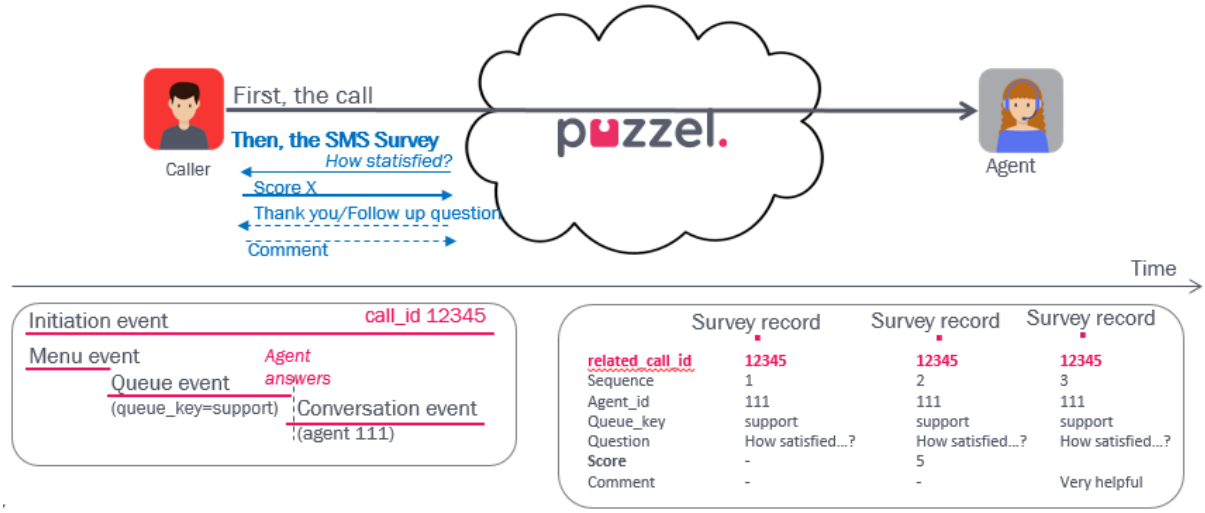
If your Puzzel solution has a menu asking the caller to dial 1 to accept recording or dial 1 to deny recording, you might experience that some calls are not recorded even if forced recording is configured on the queue/agent. To see how many callers that accepted or denied recording, look for call_ids with the relevant DTMF value in the relevant menu.

2.17 Survey records

2.17.1 SMS survey

For Puzzel customers with SMS Survey, an SMS with a question can be sent to the customer after a call ('How satisfied…?'), and the customer might answer by sending an SMS with a score (usually 1-6) and/or a comment. The SMS service can send a follow-up question /thank you message, and the customer may send a 2nd answer (also if follow-up/thank you was not sent).

SMS Survey records are stored in table `surveys`.



There might be 0, 1, 2 or 3 SMS survey records per call_id.

- *For the 1st SMS sent to a customer, a record with the question is created (sequence=1)*
- *For the 1st answer (with score and/or comment) received, a new record is created (sequence =2). If a new message is sent to the customer, the question/message is included in the Question field.*
- *For the 2nd answer (with score and/or comment) received, a new record is created (sequence =3)*

About *parent* and *fertile*

- *Records with sequence=1 has no parent, but records with sequence =2 or 3 has a record with sequence =1 as parent.*
- *Records with sequence =1 are always fertile (true), and records with sequence =3 is never fertile. Records with sequence =2 is fertile only if a follow-up question/thank you message is sent.*

The `destination` is the mobile phone number that the survey is sent to.

The survey records contain *agent_id* and *queue_key*, so that you easily can calculate e.g. average score per queue and/or per agent.

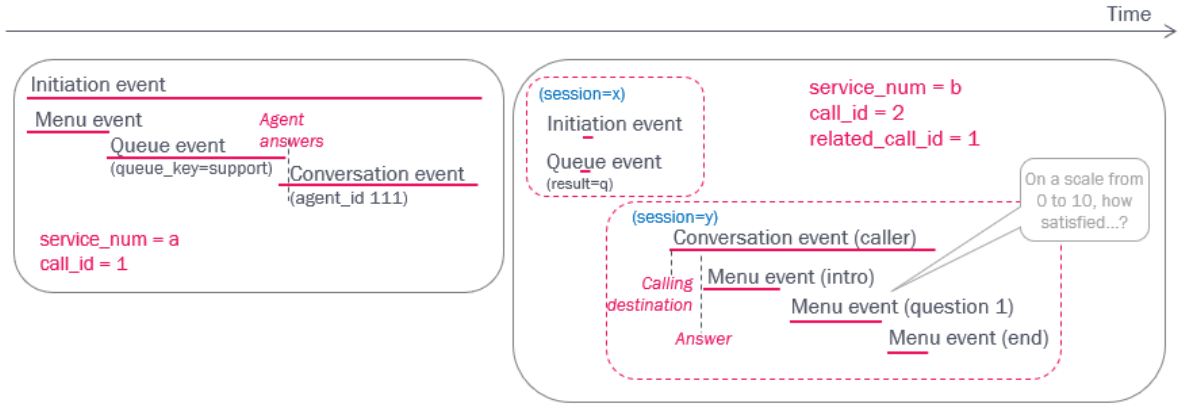
The SMS survey records also contain *related_call_id* (and related_iq_session_id), so that the survey results can be linked to the calls. With these survey records, you can calculate the average score per queue and/or per agent and relate it to e.g. wait time in queue and speak time to see how wait time and agent speak time affects the score. If the related call_id has more than one answered queue event or more than one answered conversation event, the survey record usually contains the id for the last queue and agent.

Please note that a customer may send one score first (in sequence 2) and then another score (in sequence 3). In statistics, we usually use the score from sequence 3 if it exist, and if not, we use score from sequence 2.

2.17.2 Callback Survey

A Survey (callback) call is a (web) callback from Puzzel IVR to the person that just finished the call with the agent. The Puzzel IVR might ask one or several questions (different menu events).

A callback Survey does not result in records in table surveys! The Survey call has its own `call_id` and usually another `service_num` than the main Puzzel solution, and the results will be found in menu events in the call_events table.

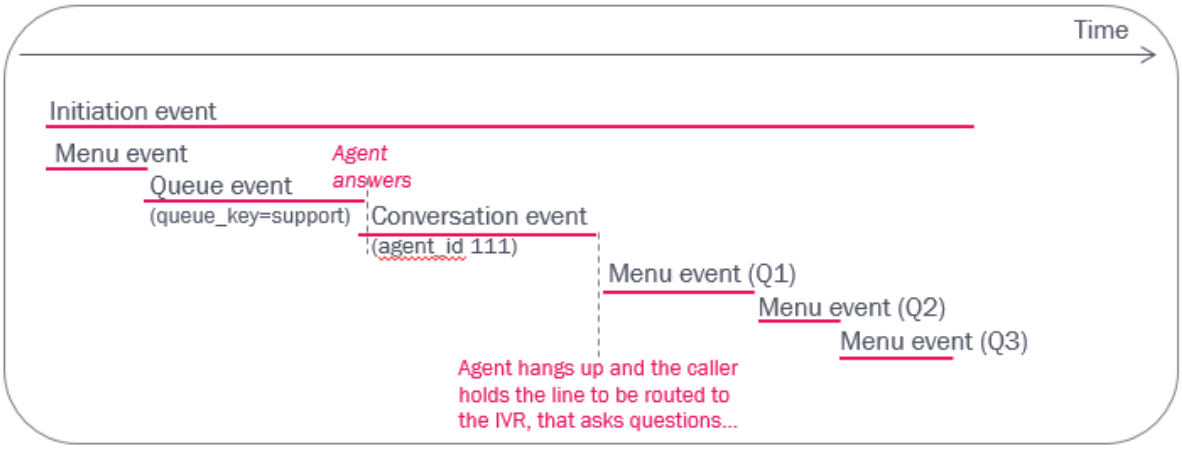


The link between the call to the service were the caller spoke with the agent and the Survey call can be found in table `call_relations` (if configured by Puzzel that the main service's session is sent to the survey service as system_related_session_id). For this example there will be a record in `call_relations` with call_id=2 and `related_call_id`=1.

2.17.3 Hold-the-line Survey

In this Survey type the caller is asked to hold the line after the agent hangs up. If he/she does, the Puzzel IVR will ask question(s) to the caller.

A Hold-the-line Survey does not result in records in table surveys! The Survey answers (DTMF values) to the Survey questions for a caller can be found in different menu events in table call_events in the same `call_id` as the incoming call.



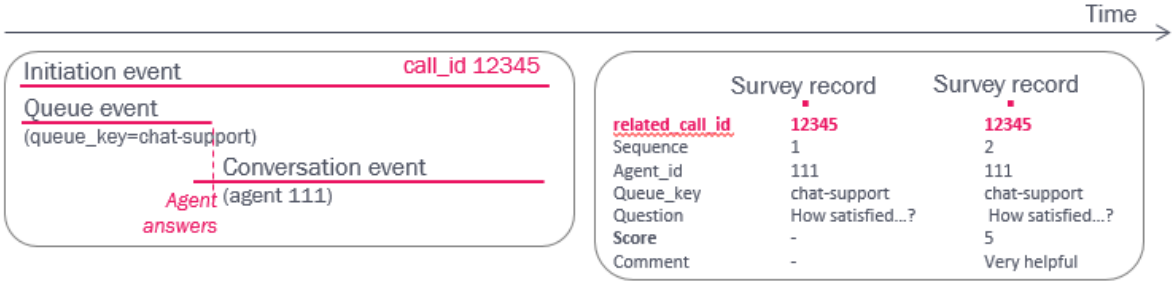
2.17.4 Chat Survey

After a Puzzel Chat, the chatter will be asked a Survey question, if configured. Chat Survey records are stored in table `surveys`. There might be 0, 1 or 2 Chat survey records per call_id for chat.

- *When a chatter is offered a survey, that is, to rate the chat he/she just finished, a chat survey record with the question is created (sequence=1)*
- *If the chatter answers the survey (enters a score and optionally a comment), a new chat survey record is created (sequence =2)*

The chat survey records contain `agent_id` and `queue_key`, so that you can calculate e.g. average score per queue or per agent (without linking the survey records to the previous chat records).

The chat survey records contain `related_call_id`, so that the survey results can be linked to the chats and their queue wait time, chat speak time etc.



2.18 Link between call_ids: call_relations

If you want to create a link or a relation between to call_ids in raw data, e.g. for web callback Surveys, this can be configured. See chapter Callback Survey. Such relations will be found in table **call_relations**. One record is stored for each such call relation. The call_id will be call_id from call #2, and the related_call_id will be the call_id for call #1.

If you want to create such relations between calls (in different services in) your Puzzel solution, please contact Puzzel support to discuss the details to figure out if and when this can be configured (system_related_session_id).

This table may be used for different 'linking' scenarios in the future, e.g. for Scheduled task, where these scenario are relevant: Linking a call to a Scheduled task, linking a Scheduled task to a call, or linking a Scheduled task to a new Scheduled task.

2.19 Call variables

It is possible to do external look-ups from Puzzel to a system (e.g. your CRM-system) on behalf of a caller, e.g. to find the customer number based on the caller's phone number or a value entered using DTMF. If a value retrieved from such a look-up is stored in a variable in Puzzel and copied to raw data ('xdrcopy'), such values can be stored in a separate table called call_variables in the raw data. It is also possible to store call/system variables not related to external look-up for raw data.

The call_variables table contains the call_id, timestamp, variable **name** and variable **value**. There might be 0, 1 or more records for one call_id, and there might be more than 1 record for one variable name for the same call_id.

Call variables can also be useful for Dialler calls. Examples of what can be included:

- values stored in one or more of the variables in the list (Var1-Var10)
- external_id (from the list)
- call attempt number (1, 2, 3)
- silent_call (e record with value true when silent call happens)
- Dialler mode (0=Preview, 1=Preditive, 2=Progressive, 3=Power)
- SMS text sent to customer after no-answer

If you want to store something as a call_variable and show it in your raw data sql db, please contact Puzzel support to discuss the details and to order the needed service configuration (variable definition and 'xdrcopy').

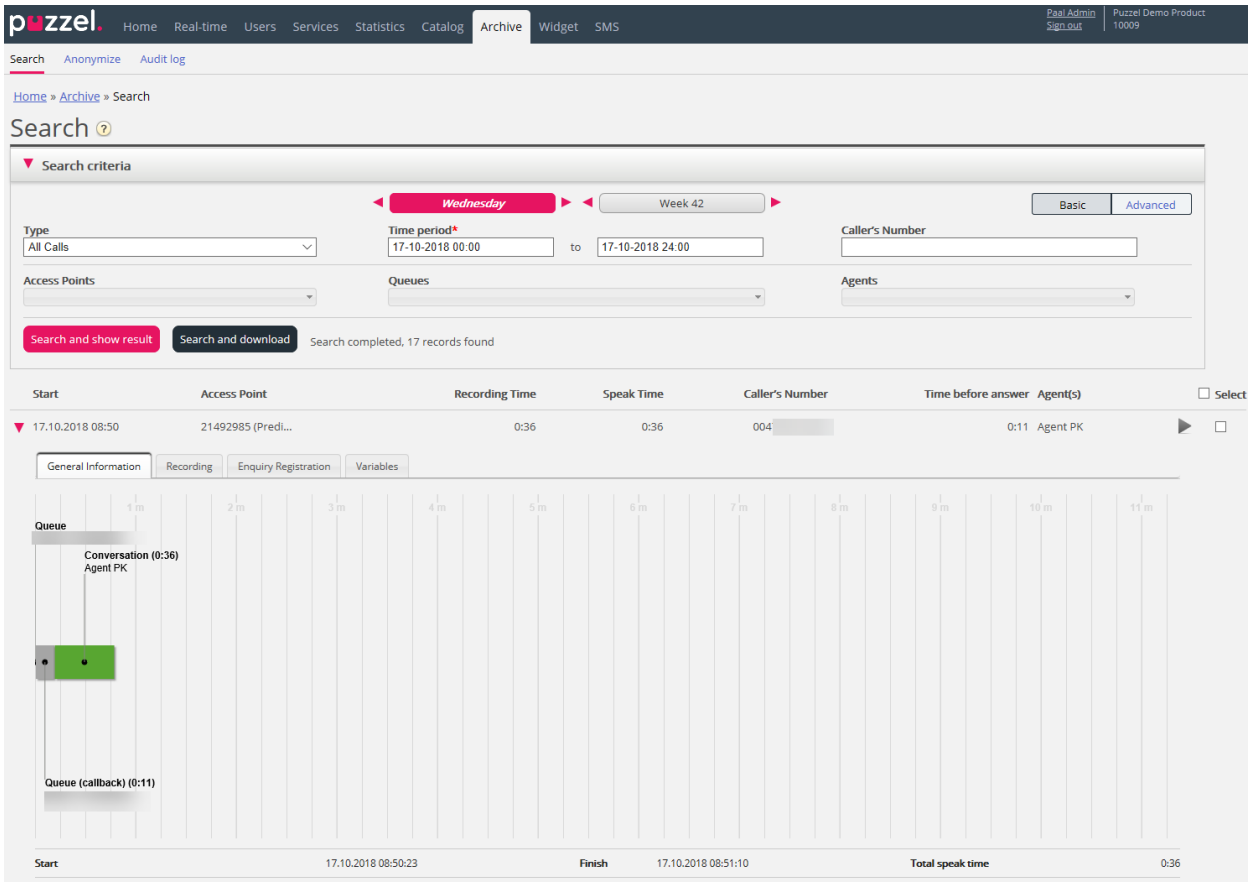
2.20 Call visualisation in Puzzel's Archive

The reason for this chapter is to explain possible differences between the detailed call information found in Raw Data and what you might see in the Archive in Puzzel's Admin Portal.

The Puzzel Archive gives a ***simplified illustration of the calls***.

The user can search for a call (email or chat) and see a simplified graphical illustration of what happened and listen to recording(s) in this call (if any).

In the Archive's call graph, we shown one «status» at a time for a call (=Call-id). A call goes through none or more Menus, none or more Queues and has none or more Conversation events and possibly a consult part (or more).

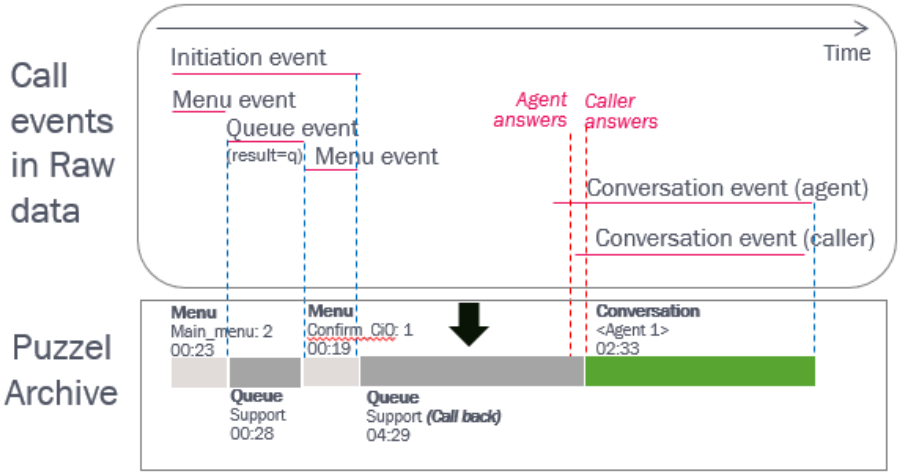


Please note that:

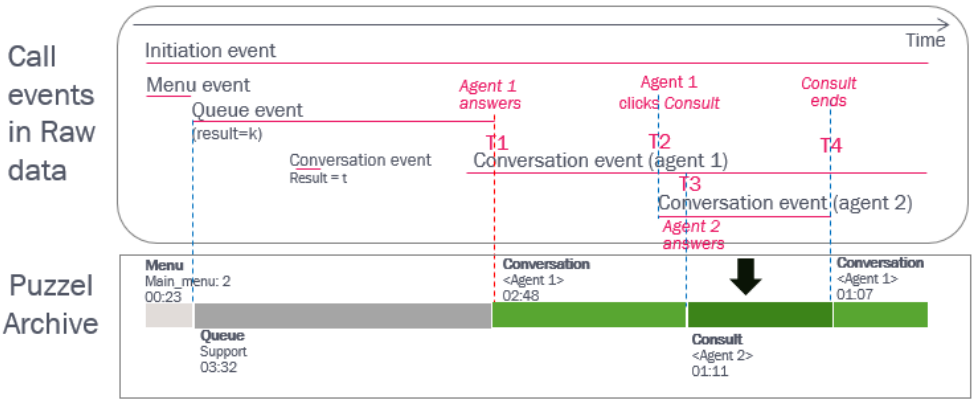
- *Conversation events with result busy/no answer/timeout are not shown in Archive*
- *Only speak-time for Conversation events are shown (not connecting and wrap-up)*
- *For consult calls (2 conversation events partly overlapping in time) we show the consult part separately in the Archive (see illustration)*

- *For Callback in queue…*
  - *the Archive only shows the time the agent and the caller were joined as Conversation time (see illustration)*
  - *the Archive shows queue time for the time the caller is not "on the line" (the queue event stops when caller exits the queue to order Callback)*
- *Conversation events for Silent monitoring/Listen in calls (type 'I') are not shown*

Callback in queue in Archive



Consult call in Archive:



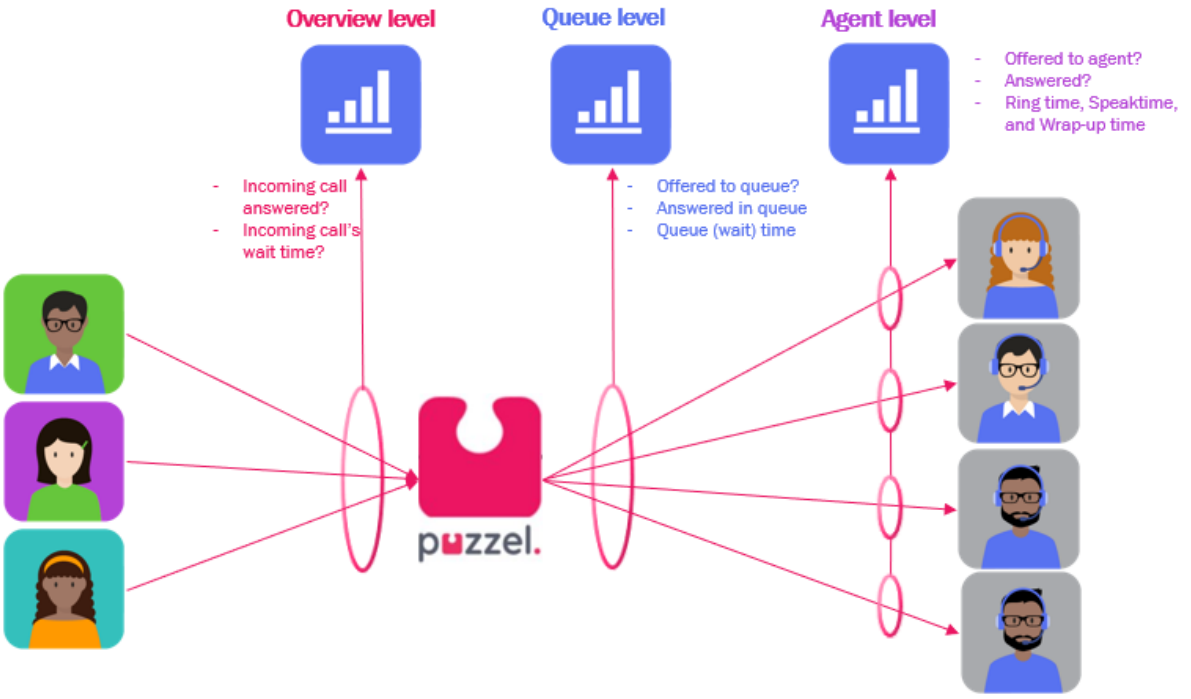
More details about the Archive can be found in the Admin Portal User Guide on help.puzzel.com.

3 A few words about statistics

Some of the issues discussed here is also included in the Statistics section on help.puzzel.com

3.1 Different measuring levels

There are many ways to summarise, and different focuses. You might think that the question “How many calls were answered?” is easy to answer, but the answer depends on what focus you have or at what “level” you are focusing.



Example:

One caller calls an access number, enters queue A, which first calls ready agent 1 (who rejects call) and then agent 2 who answers (after 40 sec in queue). Agent 2 then transfers the call to queue B. Queue B calls agent 3 who does not answer, and then agent 4, who answers (after 3 minutes in queue B). Finally, agent 4 tries a consult call to agent 5, but agent 5 does not answer, and the caller and agent hangs up.

- *On an **Overview level** we might say that this incoming call (one caller) is answered after 40 seconds wait time since (at least) one agent answered.*
- *On a **queue level**, we have one caller that entered 2 queues and was answered in both queues. (Different queue time in the 2 queues)*
- *On the **agent level**, 5 different agents were called, 5 had one call offered, but only 2 agents answered.*

The number of conversation events with a given queue_key is not the same as the number of queue events for the same queue_key, since there may be 0 or several conversation events (agents called) for each queue event for the same call_id, that is, for one caller in queue.

So, how many calls did we have, and how many were answered?

3.2 Callback in queue: Wait time and Answered or not?

What is the "correct" *wait time* for an incoming call on overview level and queue level?

On an overview level an incoming call's wait time could be defined as the time from it enters the first queue until the first agent answers. The reason is that any time in menu(s) before entering the first queue is not wait-time since the caller has not chosen what to wait for before having entered a queue. On a queue level, one could say that the wait time before answer is the time the caller spent in the queue before being answered by agent.

Callback in queue makes things more complex. The standard callback solution is that the queue calls the agent first, and then the one that ordered callback. When the agent answers and the call to the one that ordered callback is not answered, Puzzel will try again 1 or 2 more times if configured. See illustrations in chapter Callback in queue (CiQ).

In Puzzel Statistics, the rule is that an incoming call that ordered Callback is answered if the caller and the agent is joined (the call_id needs answered conversation events with ciq=a and ciq=c).

What is the "correct" wait time for an answered callback in queue?

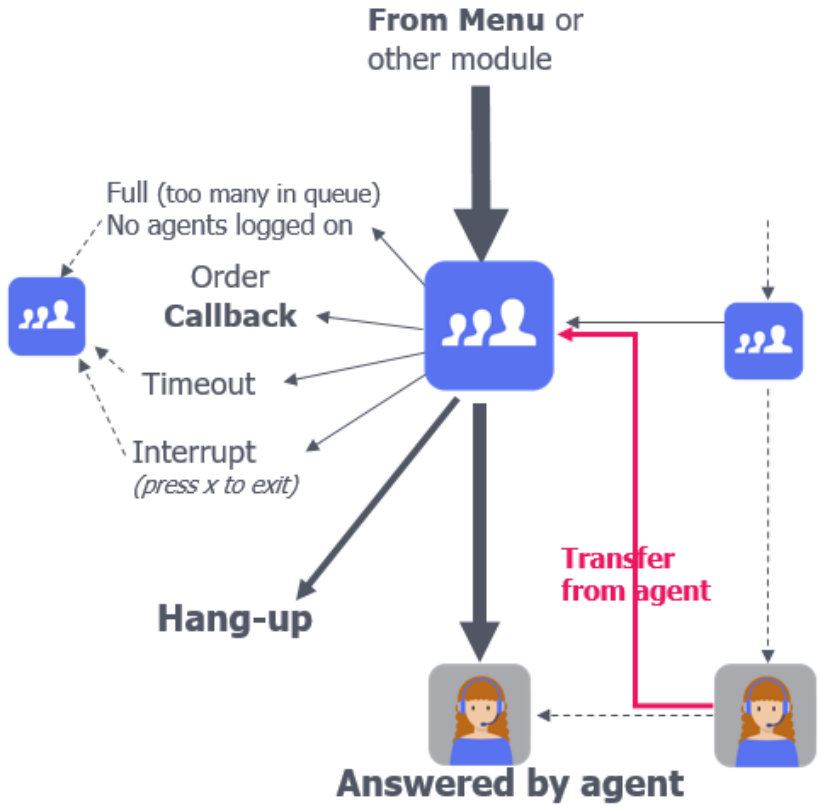
a) As seen from the caller, one could argue that the wait time for a Callback is the time from entering queue until an agent and the caller is connected (this might happen on the 1st, 2nd or 3rd callback attempt)

b) As seen from the contact centre, one could argue that the wait time is from entering queue until the first agent answers, even though the one that ordered callback not always answers on the 1st attempt.

The higher the share of callbacks in queue, and the higher the share of callbacks being answered on 2nd or 3rd attempt, the greater the difference between wait time for ***a*** and ***b*** will be. If all Callbacks are answered on the 1st attempt, the difference in wait time between ***a*** and ***b*** will be only a few seconds.

3.3 Several ways to enter and exit a queue

One incoming call may go through none, one or several queues. The number of calls offered to a queue is seldom equal to the number of answered calls plus the number of hang-ups in queue.



Fallback exit and "stand-alone"

When an incoming call arrives in a Traffic Module (TM) in the Puzzel platform, the TM asks the Puzzel databases which agent that should get the call, and then the TM calls the agent. If a TM cannot reach the Puzzel db, the TM enters 'stand-alone' mode. When in stand-alone, the TM chooses the Queue's Fallback exit if this is defined. If no Fallback is defined, the TM selects one of the agents that has answered a call lately (without knowing if this agent now is logged on). This may result in calls to agents that are in status pause or logged off, and these calls may be answered or not.

If Puzzel, on behalf of a caller in queue, calls to an agent that is "drawn" from the cached phone numbers of agents that recently answered, this results in a conversation event (as usual), and in a record in `call_event_extras` with from_cache = 1.

3.4 "Offered" to agent

The agent's answer rate is defined as requests answered divided by requests offered. Should all calls/requests set up to agents count as "offered", or only the calls/requests the agent had a fair chance to answer?

Which conversation event results are counted as Offered to agent in Puzzel standard agent statistics?

- Busy/reject (b) – included in "Offered"
    - For calls, the agent can click *No/Reject* when the call is ringing, or the call to the agent might get busy signal immediately. (If agent uses an external phone, the ready agent's phone might be busy in a non-Puzzel call.
    - For written requests, if the agent clicks *Reject* when offered a request
- Alerting timeout (t) - included in Offered
    - For calls, the agent did not answer within the defined max ringing time
    - For written requests offered to the agent's application, the request is not accepted within the defined "Max offer time"
- Caller hang-up (q or h) - included in Offered
    - The caller hung up while the call to the allocated agent was in set-up or ringing phase. See more details in chapter Results for Conversation events
    - Will not happen for written requests.
- Error (e) - not included in Offered
    - Error when calling agent
    - Error when sending a written request to the agent's Puzzel application.
- Set-up timeout (c) - not included in Offered)
    - When calling the agent, the call setup was not received correctly by the agent's phone within the max time, so the call did not "ring"
    - When sending a written request to the agent, the agent application did not respond "received and ringing"
- Answered (k) is of course included in Offered

3.5 In which time period does a call belong in Puzzel statistics

In Puzzel Statistics, a call_id with all its events is usually reported in the time period where the initiation event has its finish!

Puzzel generates all raw data for a session AFTER it is finished, so we know nothing about the request (in raw data) when it's ongoing. Two exceptions:

- *Callback in Queue: Here we get events for the order callback session first, and later we get events for the actual call session(s)*

- *E-mail in queue: Here we first get a pre-initiation event, and when email handling is finished we get all events for the session.*

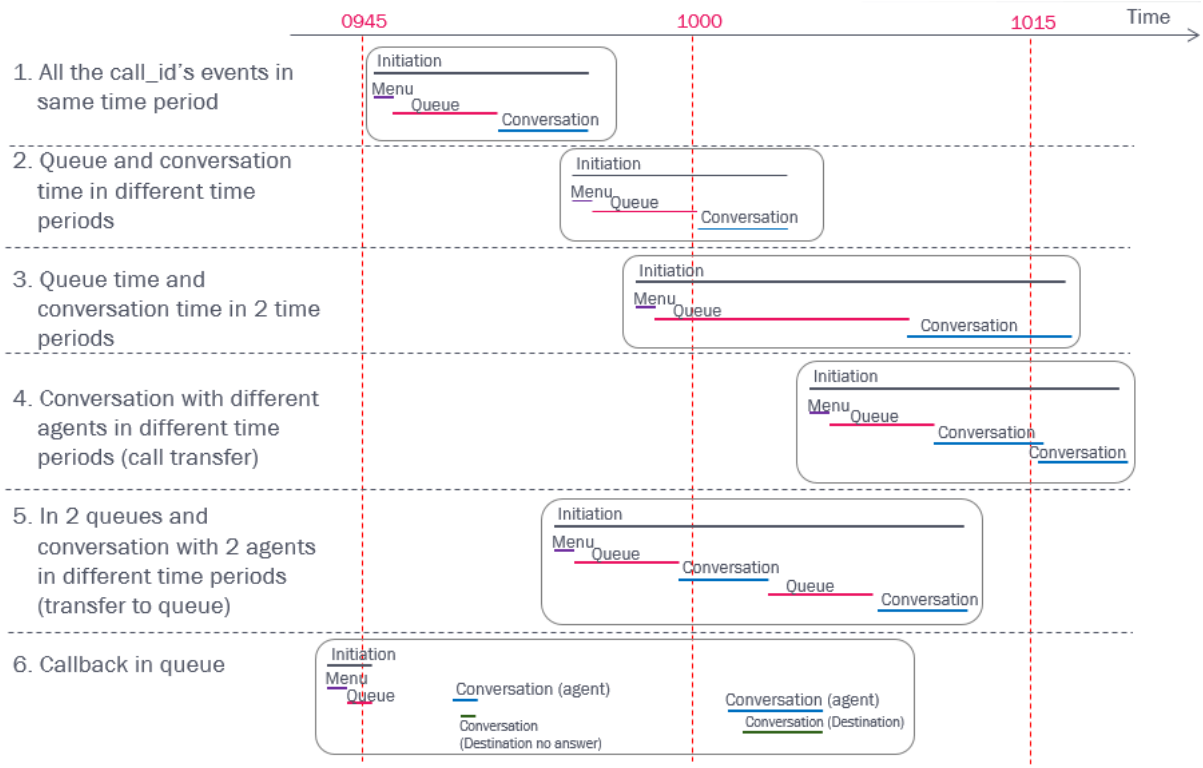
Every 15 minute all day long Puzzel generates all statistics for all customers for the 15-minute period that ended xx minutes ago (default 120 minutes, minimum 15). When a customer orders a report in Puzzel Admin, we summarise the needed 15-minute statistics on the fly. With this time categorisation, we are sure all call_ids are included.

Customers with Callback in queue and delay less than 120 minutes usually have their statistics updated every midnight in case some callbacks happened more than 120 minutes after it was ordered.

If a **callback** or **Dialler** call stays in queue over midnight (not recommend), this results in a first session with an initiation and queue event day 1, and a new session (with the same call_id) with conversation event(s) arriving day 2. Since statistics for day 1 is already generated, statistics for day 1 needs to be regenerated after the conversation events arrived day 2.

With Puzzel raw data, you can categorise calls based on their start time if you want, but be aware that if you ask for very fresh data, some calls/chats and emails might not yet be finished, so the raw data do not exist yet!

In which time periods do these calls belong in a queue report and in an agent report?



You can use different time categorisation for different queries/report types. One idea is to generate an Agent report where calls (Conversation events) are put in the time period the call to the agent started (or ended), and a Queue report where Queue events are put in the time period the call entered the queue (queue event start).

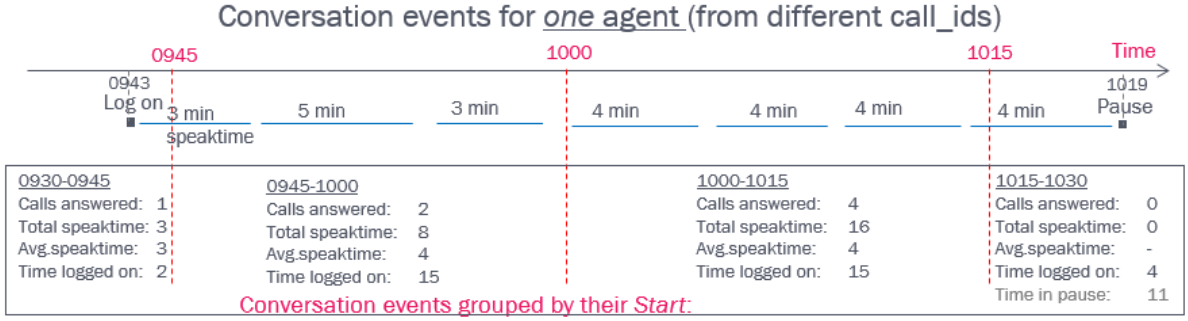
If you want to make a queue report for queue events that started in time period X, and you want to include agent speaktime (like in Details per queue), you have 2 options:

1) Select the conversation events that started in time period X.
   Some of these conversation events belong to call_ids that arrived in queue in the previous time period, and conversation events for some of time period X's queue events have start in the next time period and will not be included!
2) Select the conversation events that belong to the queue events that started in time period X (based on call_id).
   Some of these conversation events have a start in the next time period.

Or, you can make a Queue report *without* agent speaktime, since speaktime and wrap-up time per queue can be found in the sum rows in your Details per agent per queue report.

Split a single call's speaktime between two time periods?

If you report calls per 15, 30 or 60 minute interval, several conversation events will have speaktime starting in one time period and ending in the next. If you place a conversation event in a time period based on its start (or finish), the reported speaktime per time period will not be 100% «correct».



The longer the time period, the fewer calls will have their speaktime in 2 time periods. Since you probably want to report number of calls and average speaktime, we don't recommend to spilt the speaktime from one conversation event into 2 time periods.

3.6 First call resolution / repeat callers?

First call resolution can be defined as “Resolve the customer problems, questions or needs the first time they call, with no follow-up required”. Can you see this by looking at Raw data records only? How does the agent know that a “case” is closed when the call ends? Are you sure the customer will not call back later? Can Enquiry registration be used? E.g. let agent click “New case – closed now”, “New case – needs follow up”, Existing case – re-opened”…? Or is this information available in your CRM system where you log calls and link it to case? Or maybe the scores and comments in SMS Survey after a call (and chat survey after a chat) give some indications?

What about “repeat callers”?

You can look for calls from the same phone number that are answered on the same queue, but please note that there are some challenges with this approach:

- One person might call from different phone numbers (mobile, work, home)
- Is a new call about an existing “case” or is it a new “case”?
- What time period should you use? 1 week? If a call from a person was answered on Friday, how do you know if this person will call back (about the same case) next week?

4 The Database Structure (Data Model)

4.1 Introduction

The database holds information about all call and agent related activities which have relation to the Puzzel Contact Centre solution.

There are two "main" tables holding all the events, `agent_events` and `call_events`. In addition, there is a set of "basic tables" supporting the two event tables. These basic tables contain more descriptive information about codes, ids, and values used in the event tables.

The *call_events* table holds a set of records for every call/chat/email/social media request handled by Puzzel, and it also expresses the duration and outcome (result) of each of these events. **All events belonging to the same call are related to each other through the column *call_id*.**

The *agent_events* table holds information about every so-called "*agent driven contact centre events*", i.e. every time the agent is logging on or off, or is going to or from pause. In addition, every call/call attempt which agents are involved in will be stored here (as well). It's important to know that, except from the call/call attempt event, all events in this table will have a duration of 0 (i.e. the time it takes to active the action) – exactly as in Puzzel's central systems (which is the source for all the data stored in the tables). Though, a set of functions/procedures calculating the duration of all events in this table will be made available in the database (described separately). Those functions/procedures will also be used to calculate other useful values, as well as "generating" events for wrap up and availability.

In order to make the tables easy and faster to work with, some data are de-normalised (i.e. stored in both event tables). This is true for those *Conversation event*s in the *call_event*s table having agents involved – such events will have some of the same columns copied over to the *agent_event*s table.

On the next page a graphical layout of the database model is shown. On the pages there after the content of each table are described in detail.

4.2 Database diagram

Fig. Database diagram, part-1 (of 2)

* New in db v1.6

Fig. Database diagram, part-2 (of 2)

* New in db v1.6

4.3 Table descriptions

4.3.1 agent_event_sources

Table name:	agent_event_sources		
Contains / Role:	Holds the description for the different possible sources for an agent event. Table agent_events holds a foreign key against this table.		
Columns	Name	Datatype	Description:
	event_source	Char (1) Primary key.	The code for the source. Currently these codes are in use: a – Automatic (logoff at midnight or after x unanswered calls) c – Very old Connect agent desktop clients i – Internal (for event_type c) p – Phone (logon/off from Phone "back-door") w – Web (Agent application or Admin Portal*) ? – Unknown (should not happen)
	descript	Varchar (255)	Describes the code.

* If an agent is logged off/on by an admin using the Admin Portal, this is also shown in the Admin Portal's change log.

4.3.2 agent_event_types

Table name:	agent_event_types		
Contains / Role:	Holds the description for the different event types for an agent event. Table agent_events holds a foreign key against this table.		
Columns	Name	Datatype	Description:
	event_type	Char (1) Primary key.	The code for the type of event. Currently these codes are in use (could be extended in the future): ? Unknown a Available (i.e. logged in and ready for calls). **Note!** This is event does not appear directly in the *agent_event* table, but could be part of the result when built-in table value functions are used in queries against *agent_event* table c Conversation (i.e. in phone) i Log In (to queue) o Log Out (from queue) p Entering Pause r Return from pause w WrapUp. **Note!** This is event does not appear directly in the *agent_event* table, but could be part of the result when built-in table value functions are used in queries against *agent_events* table.
	descript	Varchar (255)	Describes the code.

4.3.3 agent_result codes

Table name:	agent_result_codes		
Contains / Role:	Holds the description for the different possible result codes (“outcomes”) of an agent event.		
Columns	Name	Datatype	Description:
	result_code	Char (2) Primary key.	The code for the result. Currently these codes are in use: ? Unknown b Busy (event type 'c') c Timeout in setup phase (event_type 'c' only) e Error h Hangup by caller while calling agent (event_type 'c') k OK q Caller hangup during setup (event_type 'c' only) t Timeout (=no answer, event type ‘c’ only)
	descript	Varchar (255)	Describes the code.

4.3.4 agent_events

Table name:	agent_events		
Contains / Role:	All the “call centre events”, and also the *Conversation event*s (i.e. events representing the time when the agent is involved in calls).		
Columns	Name	Datatype	Description:
	rec_id	Int Primary key.	Internal, auto-numbered. A unique id for each record.
	agent_id	int	A unique number identifying an agent. The table has a foreign key against table *agents* for this column.
	dte_start	datetime	When the start of the event occurred.
	profile	nvarchar(100)	Name of the profile the agent used at (last) login.
	service_num	varchar(100)	The service number the agent is in context of when logged in.

	phone_num	varchar(100)	The phone number to the agent (specified at last login).
	duration_sec	int	The total duration of the event (in secods). NOTE! So-called "call centre events", i.e. logging in and out, and moving into or returning from pause, will always have a duration of 0. Though, working against the table through built-in table functions will also return additional duration columns expressing the time the agent stayed in each of the "call centre events". Though, the *Conversation event*s (type 'c') will hold a duration > 0
	event_type	char(1)	Refers (foreign key) to table *agent_event_types*.
	event_source	char(1)	Refers (foreign key) to table *agent_event_sources*.
	result_code	char(2)	A result code telling something about the "outcome" of the event. Refers (foreign key) to table *agent_result_codes*.
	queue_key	nvarchar(100)	The *key* (short code) for the queue involved in the event. See table queues for a more descriptive name (NOTE! Use both the *queue_key* and the *service_num* as join criteria columns, the key itself is not necessarily unique if the database holds data for more than one customer!) Outer join is recommended, since there *might* exist keys without extended descriptions.
	pause_type_name	nvarchar (100)	Name of a pause type, only set for events of type 'p'.
	pause_type_id	int	A numeric id (also) identifying the pause reason.
	call_transfer	bit	Set to 0 or 1 for events of type 'c' (conversation), for all other events the value is NULL. If 1, the call has been transferred (to another agent or to just another number) by the agent just before he/she terminated his/her conversation.
	wrap_up_sec	int	Only set for events of type 'c', and only if a wrap up time is predefined for the agent. Wrap-up time is the time after a terminated call until the agent is ready for new incoming calls. Value is expressed in seconds.
	block_duration_sec	int	Only used for *Conversation event*s, and only if busy or no-answer is the result for the call. After such a result, the user may be set "passive" for a short period. The duration of such a period will be set here. Value is expressed in seconds.
	internal_adr_id	bigint	For internal use, for "connection" back to Puzzel's internal system. Only set for "call centre events".

	internal_odr_id	bigint	Only set for *Conversation events*. For internal use, and also the column linking the event to it's "origin" in the *call_event* table. NOTE! For emails and chats we do not have any *internal_odr_id* from the "base system", still a "link" to the corresponding record in table *call_events* is wanted. To separate this from phone calls (avoid value conflicts), all *internal_odr_id* values for email and chats conversation will hold negative values – as the corresponding record's column *rec_id* in table *call_events* is multiplied with -1
	internal_country_ src_db	Varchar(10)	Mostly for internal use. Indicates the country for the database (internally) where the original record is fetched from. International standard for country codes is used (for example 'NO' for Norway, 'SE' for Sweden, and so on).
	dte_updated	datetime	A timestamp indicating when this *agent_event* record is stored/last updated in this database.
	phone_type_id	int	To indicate if it's a Puzzel Softphone or external phone that is used by the agent. Value 1 or 2. See table 'phone_types' for details. (Introduced in version 1.4.0.0. NULL value for older records.)
	usergroup_id	int	See 'table' (actually a view) *user_groups* for description.

4.3.5 agents

Table name:	Agents		
Contains / Role:	Table holding information about each agent. "Supports" the table *agent_event*, which holds a foreign key reference to this table. This table is updated every night if there are agent changes in Puzzel (new or deleted agents, or agents having changed user group).		
Columns	Name	Datatype	Description:
	agent_id	int	A unique number identifying the agent. For internal db use only, the end-user does not know about this.
	customer_key	varchar(100)	The customer key (number) for the customer who "owns" this agent. For those customers only holding one customer key at Puzzel, all agents will have the same customer_key.
	user_num	varchar(50)	A (up to six digit) number that identifies the agent. Known for the end-users. Unique within same customer key. Optional field, not necessary set for all agents.
	full_name	nvarchar(250)	A combination of the agents first and last name.

	usergroup_id	int	A database internal unique id for the user group.
	usergroup_name	nvarchar(250)	The name of the user group.
	dte_updated	datetime	Datestamp for the last update of this agent record.
	chat_role	Tinyint	For agents having chat/social/email media type/skill in the profile, there will be a separate agent_id for each possible parallel request the agent can handle. (Max parallel written requests is defined by the admin). For such agents, there will always be one "normal" agent_id (the so-called master agent), and one or more (chat) slave agent_ids.) The link between the master agent and the (chat) slave agents are done through column *chat_master_user_id* (see below). *Chat_role* will be NULL for agents without chat/social/email, 0 for the "master" agent when chat/email/social is activated, and 1 for slave agents.
	chat_master_user_id	Int	The agent_id for the master agent. See also above.
	unblockable_role	tinyint	If "unblockable" is activated, there is a set of "agents", one for each line. One of the agents, the so-called *master agent* will hold the name and password, and is the one the end users will see and deal with. *Unblockable_role* is NULL for non-unblockable agents, 1 for the unblockable master agent, and 0 for all other lines (slave agents). The value 99 is used for those slvae lines not in use anymore (typically when the number of lines in parallel has been reduced later).
	unblockable_group	int	A unique id grouping all the unblockable agents within the same unblockable group together. (The number used is the number for the usergroup to whom the unblockable agent (group) belongs).
	Deleted	bit	The agent might be deleted in the source (active) system, though we still need it here in the database for historical reasons. Value of 1 means that the agent is deleted in Puzzel, 0 = the agent is (active) in Puzzel. The column *dte_updated* is also updated as well.
	user_name	nvarchar(50)	The agent's username (when login is done).
	email	nvarchar(256)	The user's registered email address in Puzzel.
	mobile	nvarchar(256)	The user's registered mobile phone number in Puzzel. (The phone number the agent uses when logging on to queue is found in table agent_events in field phone_num).

4.3.6 call_event_types

Table name:	call_event_types		
Contains / Role:	Holds the description for the different event types for a call event. Table *call_events* holds a foreign key against this table.		
Columns	Name	Datatype	Description:
	event_type	Char (1) Primary key.	Currently these codes are in use (may be extended later): c Conversation i Start/initiation l Listen in / silent monitoring m Menu p Pre-initiation event (for email only, see chapter 2.9 Email in queue) q Placed in queue r Recording started
	Descript	Varchar (255)	Describes the code.

4.3.7 call_events

Table name:	call_events		
Contains / Role:	All the events happening within all calls.		
Columns	Name	Datatype	Description:
	rec_id	Int Primary key.	Internal, auto-numbered. A unique id for each record.
	customer_key	Varchar (100)	The customer key (number) for the “owner” of the call.
	call_id	Numeric(19,0)	A number which is unique for each call (request).
	call_sequence	int	A sequence number (starting with 1) for each events within the same call. The order is determined by the time for start of the event (*dte_start*).

	media_type_id	int	Identifying which type of media, refers (foreign key) to table *media_types*.
	dte_start	datetime	The date/time for the start of the event.
	duration_tot_sec	int	The total duration of the event, in seconds.
	duration_speak_sec	int	The total speak time (if a Conversation event), in seconds.
	dte_speak_start	datetime	The date/time of when the speaktime started within the event (if a Conversation event). Expressed in seconds. The time gap between *duration_speak_sec* and *dte_speak_start* will be the time spent in allocate/setup/alerting.
	source	Varchar(500)	Identifying the caller. For an event of media type phone, this will be the caller's phone number (the number the Puzzel platform received as Calling Party Number) For **email**, it's the received 'From' email address For **chat**, it's the received Chat_Id which usually is the email address the chatter has entered in the chat form. For **social**, it usually contains the conversation id. This might be changed to be the 'author' in a future release.
	destination	varchar(500)	For ***initiation events***, the destination will be - the access number the caller called (or was forwarded to). - for **chat**, it's the Puzzel id of the chat queue. - for **email**, it's an email address. - for **social**, it usually contains the customer number and 'Facebook' or 'Twitter' For ***conversation events*** *for media type phone*, the destination will be the agent's phone number or the called destination's number. - client.puzzel.com's Softphone = 19500xxxxxx - agent.puzzel.com's Softphone = 19510xxxxxx
	additional_source	varchar(500)	Contains the so called "additional calling party number" for phone calls. For other media types we might add information here in a future release.
	redirect_source	varchar(500)	Contains the so called "redirecting number" for phone calls. For other media types we might add information here in a future release.
	service_num	varchar(100)	An access number "belongs" to a service number (which again belongs to a customer). This is the service number. A service number could have more than one access number.

	queue_key	nvarchar(100)	For events of type *queue* ('q') and *conversation* ('c'). The *key* (short code) for the queue involved in the event. See table queues for a more descriptive name (NOTE! Use both the *queue_key* and the *service_num* as join criteria columns, the key itself is not necessarily unique if the database holds data for more than one customer!) Outer join is recommended, since there *might* exist keys without extended descriptions.
	menue_key	nvarchar(100)	For events of type *menu* ('m'). The *key* (short code) for the menu involved in the event. See table queues for a more descriptive name (NOTE! Use both the *queue_key* and the *service_num* as join criteria columns, the key itself is not necessarily unique if the database holds data for more than one customer!). Outer join is recommended, since there *might* exist keys without extended descriptions.
	menue_choice	varchar(255)	For events of type *menue* ('m'). The DTMF codes (optionally) keyed in by the caller. Might be screened (set to empty) for "secret" information, like for example credit card numbers.
	agent_id	int	For events of type *conversation* ('c'). The unique id for the agent involved. If events of type 'c' does not have an *agent_id* (i.e. NULL), it indicates an outgoing call to "just" a number, or call to a caller in a call back in queue sequence.
	event_type	varchar	Refers (foreign key) to table *call_event_types*.
	result_code	varchar	Refers (foreign key) to table *call_result_codes*.
	answered	tinyint	Only set for the event of type 'i'. This is a kind of "summary" for the whole call/email/chat, telling if the request was connected to an agent. The main purpose is to make it easier to search for incoming calls/chats/emails that where answered or not. The value is 0 or 1, where 1 is set if at least one conversation event (with ciq=NULL) for the call_id is answered, or if one conversation event with ciq=a and one with ciq=c is answered. For all other events than 'i' the value is NULL.
	ciq	char	Only set when "call back in queue" happens. On the Conversation event, it might me either 'a' (agent) or 'c' (caller) to reflect to whom the call (leg) was made. On a 'q' (queue) event, it is set to 'q'. For all others, the value NULL is set.
	call_transfer	Bit	Set to 0 or 1 for events of type 'c' (conversation), for all other events the value is NULL. If 1, the call or email has been transferred (to another agent or to just another number) by the agent just before he/she terminated his/her conversation.

	wrap_up_sec	int	Only set for events of type 'c', and only if a wrap up time is defined/used for the agent. Wrap up time is the time after a terminated call until the agent will be available for new incoming calls. Value expressed in seconds.
	alert_ms	int	Only set for events of type 'c', and (so far) only for calls of type 'phone'. Express the *alert time*, also called *ringing time*. Value expressed in milliseconds.
	setup_ms	int	Only set for events of type 'c', and (so far) only for calls of type 'phone'. Express the so called *setup time* in the Public Switched Telephone Network (PSTN), i.e. the time from the number is dialed and up to the time when the alert (ringing) starts. Value expressed in milliseconds.
	block_duration_sec	int	Will only be used for *Conversation event*s, and only if busy or no-answer is the result for the call. After such a result, the user may be set "passive" for a short period. The duration of such a period will be set here. Value in seconds.
	internal_iq_session_id	Unique-identifier	The session id for the call/request. One call_id might consist of 1 or more sessions. (Primarily for internal use, for "connection" back to Puzzel's internal system.)
	internal_odr_id	bigint	Only set for *Conversation event*s. For internal use. Also used for linking the event to it's "twin" in the *agent_event*s table.
	dte_updated	datetime	A timestamp indicating when this *call_event* record is stored/last updated in this database.
	sla	int	The queue's predefined SLA, if any
	alt_sla	int	The queue's predefined Alternative SLA, if any
	dte_scheduled_callback	datetimeoffset	The queue event's scheduled time, if any
	result_response	int	Shows the signalling response code Puzzel received from the network for a phone call (conversation event with media_type id=1). Please note that the value might be an ISUP cause code (usually 1 or 2 digits) or a SIP response code (3 digits). Please see Extra information for calls (resulting in error) (result_response)

4.3.8 call_event_extras

Table name:	call_event_extras		
Contains / Role:	Holds additional information related to a specific call_event record. Currently, the only information stored here is *transfer_text*, but in the future more columns / information might be added. That's why the more generic name '*...extras*' is used for the table		
Columns	Name	Datatype	Description:
	rec_id	Int Primary key.	Internal, auto-numbered. A unique id for each record.
	call_events_rec_id	bigint	Refers to the *rec_id* column in table *call_events*, for which this record belongs to.
	transfer_text	nvarchar (MAX)	The transfer text typed in.
	agent2agent	int	An agent-to-agent call's initiation event will have a record in this table with agent2agent=1. See Agent-to-agent calls
	from_cache	Int	If the called agent was "drawn from cache" instead of being correctly allocated, the conversation event will have a record here with from_cache = 1. See Several ways to enter and exit a queue

4.3.9 call_relations

Table name:	call_relations		
Contains / Role:	Two separate "calls" might be related to each other, given correct setup of the service. This table holds such information.		
Columns	Name	Datatype	Description:
	rec_id	Int Primary key.	Internal, auto-numbered. A unique id for each record.
	call_id	Numeric(19,0)	The call_id for call #2, i.e. the call who comes with the relation information.
	related_call_id	Numeric(19,0)	The call_id for call #1, i.e. the call that call #2 relates to.
	internal_session_id	uniqueidentifier	Internal. The session_id used by 'back-end' for call #2.

4.3.10 call_variables

Table name:	call_variables		
Contains / Role:	Depending on the service, a call might deliver variable values. Such variables can be picked up values from external lookups, it can be calculation made “on the fly”, and so on. NOTE! The same variable *might* occur more than once within same call. If so, the timestamp in column *dte_time* will be unique.		
Columns	Name	Datatype	Description:
	rec_id	Int Primary key.	Internal, auto-numbered. A unique id for each record.
	internal_session_id	uniqueidentifier	The internal_iq_session_id for the call who “owns” the variable.
	call_id	numeric(19,0)	The call_id for the call who “owns” the variable.
	dte_time	datetime	When the variable value was “stored”/dumped.
	name	nvarchar(MAX)	The name of the variable.
	value	nvarchar(MAX)	The value assigned to the variable.

4.3.11 call_result_codes

Table name:	call_result_codes		
Contains / Role:	Holds the description for the different possible result codes (i.e. the “outcome”) of a call event. Table *call_events* holds a foreign key against this table.		
Columns	Name	Datatype	Description:
	result_code	Char (2) Primary key.	Currently these result codes are in use: ? Unknown a Interrupted in queue (event_type='q') b Two meanings: 1) For event type 'q': FallBack exit used. 2) For event type 'c': Busy c Two meanings: 1) For event type 'q': Admin removed the call from queue. 2) For event type 'c': Timeout in setup phase. d Two meanings: 1) For event type ‘m’: Default exit used 2) For event type ‘q’: Deleted (request was deleted from queue) e Error f The queue was full (event type = 'q') h Hangup - the call was terminated by caller before the event was completed (event type 'c', 'm' and 'q')

			k Ok m Max tries exceeded (for event type 'm' only) q Two meanings: 1) For event type 'q': Call back is ordered. 2) For event type 'c': Caller hangup during setup s The queue was closed (event_type 'q') t Timeout (event type 'c', 'q' and 'm'). For event type 'c' this means "no answer".
	descript	Varchar (255)	Describes the code.

4.3.12 media_types

Table name:	media_types		
Contains / Role:	Describes the name for the different media types a call/request might consist of. Table *call_events* holds a foreign key against this table.		
Columns	Name	Datatype	Description:
	media_type_id	Integer Primary key.	The code for the media type. Currently these ids are in use: 0 Undefined 1 Phone 3 Email 4 SMS 5 Chat 7 Social
	descript	nvarchar (255)	Describes the media type id.

4.3.13 menues

Table name:	menues		
Contains / Role:	This is a kind of "support table" for the event tables. Holds a more descriptive text for the menue_keys used in table *call_events*. If joined from the event tables, both *service_num* and *menue_key* should be used as join criteria, since databases holding data for more than one customer might have duplicates for the key alone. Outer join is recommended since all keys in use not necessary has a description defined in this table.		
Columns	Name	Datatype	Description:
	service_num	Varchar(100)	
	menue_key	nvarchar(100)	The key used in the *call_event* table.
	descript	nvarchar(255)	Describes the key.

4.3.14 queues

Table name:	Queues		
Contains / Role:	This is a kind of "support table" for the event tables. Holds a more descriptive text for the queue_keys used in table *call_events* and *agent_events*. If joined from the event tables, both *service_num* and *menue_key* should be used as join criteria, since databases holding data for more than one customer might have duplicates for the key alone. Outer join is recommended since all keys in use not necessarily has a description defined in this table.		
Columns	Name	Datatype	Description:
	service_num	varchar(100)	
	queue_key	nvarchar(100)	The key used in the *call_events* and *agent_events* tables..
	descript	nvarchar (255)	Describes the key.

4.3.15 phone_types

Table name:	phone_types		
Contains / Role:	Holds the description for the different phone_types (used/referred to in table agent_events).		
Columns	Name	Datatype	Description:
	phone_type	Int	Currently these codes are in use (may be extended later): 0 None/Unknown 1 Phone (external phone) 2 SoftPhone (Puzzel's Softphone)
	descript	Varchar (255)	Describes the type.

4.3.16 enqreg_header

Table name:	Enqreg_header		
Contains / Role:	Three tables are related together in order to store the so-called enquiry registration data (see database diagram). This table is the "master" of these tables, holding one record for each enquiry registration that has been made.		
Columns	Name	Datatype	Description:
	enqreg_header_recid	Int Primary key.	Internal, auto-numbered. A unique id for each record.

	internal_session_id	uniqueidentifier	For internal use, for "connection" back to Puzzel's internal system.
	customer_key	varchar (100)	The customer key (number) for the "owner" of the call.
	internal_country_src_db	varchar(10)	Mostly for internal use. Indicates the country for the database (internally) where the original record is fetched from. International standard for country codes is used (for example 'NO' for Norway, 'SE' for Sweden, etc).
	dte_time_stamp	datetime	A timestamp indicating when this enquiry registration was completed.
	agent_id	int	Optional. May contain the id for the agent who made this enquiry registration.
	queue_key	nvarchar(255)	Optional. May contain the queue for which this enquiry registration should be related to.
	comment	nvarchar(MAX)	Optional. Will contain the comment the agent entered (if any).
	reschedule_time	datetimeoffset	The time entered for the rescheduled (Dialler) call
	enquiry_media_type	nvarchar(255)	A text indicating on which type of media the enquiry registration was related to. Contains one of the following values; "Undefined", "Call", "Chat", "EMail" and "SocialNetworks"
	related_iq_session_id	uniqueidentifier	Optional. May contain a unique internal_iq_session_id for a call (the call may consist of more than one session) for which this enquiry registration should be related to.
	dte_updated	datetime	A timestamp indicating when this record is stored/last updated in this database.
	marked_unansw	varchar(255)	true if the Dialler agent selected Marked as unanswered, otherwise NULL. See General Dialler information
	reserved	varchar(255)	true if the agent that rescheduled the Dialler call selected "To myself", otherwise NULL. See General Dialler information

4.3.17 enqreg_category

Table name:	Enqreg_category		
Contains / Role:	Three tables are related together in order to store the so-called enquiry registration data (see database diagram). This table is the "child" of the table *enqreg_header*, holding one record for each category that has been registered within enquiry registrations.		
Columns	Name	Datatype	Description:
	enqreg_category_recid	int Primary key.	Internal, auto-numbered (in this database). A unique id for each record.
	enqreg_header_recid	int Foreign key	Refers to the "owner" record in table *enqreg_header.*
	category_id	int	A unique id (generated by back-end) for the category.
	category_name	nvarchar(255)	The name of the category

4.3.18 enqreg_topic

Table name:	Enqreg_topic		
Contains / Role:	Three tables are related together in order to store the so-called enquiry registration data (see database diagram). This table is the "child" of the table *enqreg_category*, holding one record for each topic that has been registered within each category within enquiry registrations.		
Columns	Name	Datatype	Description:
	enqreg_topic_recid	int Primary key.	Internal, auto-numbered (in this database). A unique id for each record.
	enqreg_category_recid	int Foreign key	Refers to the "owner" record in table *enqreg_category.*
	topic_id	int	A unique id (generated by back-end system) for the topic.
	topic_name	nvarchar(255)	The name of the topic.

4.3.19 surveys

Table name:	surveys		
Contains / Role:	When surveys are made after a call, and on other platforms, like for example SMS, the survey results are stored here.		
Columns	Name	Datatype	Description:
	rec_id	Int Primary key.	Internal, auto-numbered. A unique id for each record.
	internal_session_id	uniqueidentifier	For internal use, for "connection" back to Puzzel's internal system.
	customer_key	varchar (100)	The customer key (number) for the "owner" of the survey.
	country_code	varchar(10)	Mostly for internal use. Indicates the country for the database (internally) where the survey record is fetched from. International standard for country codes is used (for example 'NO' for Norway, 'SE' for Sweden, and so on).
	related_iq_session_id	uniqueidentifier	The internal_iq_session_id for the call (request) the survey results refers to. Primarily for internal use.
	related_call_id	numeric(19,0)	The call_id for the call (request) the survey results refers to.
	survey_type	datetime	The media of the survey. Currently, SMS and CHAT are values appearing here.
	sequence	int	The survey may consist of multiple steps, i.e. questions back and forth between the customer (caller) and "us". Sequence indicates the order of them. Starting with value of 1.
	dte_time	datetime	When this survey record was sent/received.
	destination	nvarchar(500)	To whom (i.e. the "address") we asked for the survey. For SMS surveys this will be the phone number. For chat survey, we usually get the chatter's name.
	agent_id	Int	The agent who was involved in the call which the survey is about.
	queue_key	nvarchar(100)	The queue that was involved in the call which the survey is about.
	fertile	tinyint	1 if this survey record *might* have more records coming later (see also the description for the *sequence* column)

	question	nvarchar(MAX)	The question sent to the caller/chatter/customer.
	score	nvarchar(100)	The score reported back from the caller/chatter/customer
	max_score	nvarchar(100)	The maximum value possible to give as *score*.
	min_score	nvarchar(100)	The defined minimum value for *score*.
	comment	nvarchar(MAX)	The caller/chatter/customer might report back a text comment (in addition to the score). If so, it's stored here.
	dte_updated	datetime	A timestamp indicating when this record is stored/last updated in this database.

4.3.20 user_groups

Table name:	user_groups
Contains / Role:	Currently this is implemented as a view. Might be changed to table later. Anyhow – view or table – you can use this to obtain more information of the usergroups, for example when referred to from column *usergroup_id* from table *agent_events*. Please see chapter 4.4 for description of the views in the database.

4.4 View descriptions

4.4.1 vw_enqreg_total

View name:	vw_enqreg_total		
Contains / Role:	This is a view over the 3 enquiry registration tables (*enqreg_header / enqreg_category / enqreg_topic*). Links them together to simplify queries where you want to see/deal with all information (header/topic/category) related to each of the enquiry registrations.		
Columns	Name	Datatype	Description:
	enqreg_header_recid	int	See *enqreg_header.**enqreg_header_recid***
	internal_session_id	uniqueidentifier	See *enqreg_header.**internal_session_id***
	customer_key	varchar(100)	See *enqreg_header.**customer_key***
	internal_country_src_db	varchar(10)	See *enqreg_header.**internal_country_src_db***
	dte_time_stamp	datetime	See *enqreg_header.**dte_time_stamp***

	agent_id	int	See *enqreg_header.**agent_id***
	queue_key	varchar(255)	See *enqreg_header.**queue_key***
	enquiry_media_type	varchar(255)	See *enqreg_header.**enquiry_media_type***
	related_iq_session_id	uniqueidentifier	See *enqreg_header.**related_iq_session_id***
	dte_updated	datetime	See *enqreg_header.**dte_updated***
	comment	nvarchar(MAX)	
	reschedule_time	datetimeoffset	
	enqreg_category_recid	int	See *enqreg_category.**enqreg_category_recid***
	category_id	int	See *enqreg_category.category_id*
	category_name	nvarchar(255)	See *enqreg_category.**category_name***
	enqreg_topic_recid	int	See *enqreg_topic.**enqreg_topic_recid***
	topic_id	int	See *enqreg_topic.**topic_id***
	topic_name	nvarchar(255)	See *enqreg_topic.**topic_name***
	marked_unansw	varchar(255)	See *enqreg_header.**marked_unansw***
	reserved	varchar(255)	See *enqreg_header.**reserved***

4.4.2 user_groups

View name:	user_groups		
Contains / Role:	Currently this is implemented as a view. Might be changed to table later.		
Columns	Name	Datatype	Description:
	usergroup_id	Int	Unique id for the usergroup.
	usergroup_name	Nvarchar(250)	The name of the usergroup. Over time a usergroup name might change. If so, the most recent name is returned.

5 Functions and Stored Procedures.

Some Puzzel specific db-functions and/or stored procedure are available in the database. Even if they are not needed to understand/work with the data in the tables, their purpose is to simplify many of the most common calculation/analyses to be made from a typical "call centre analytics" point of view. For example, it can help you find the duration of every event in the *agent_events* table (even if the column in table is 0), it can "produce" separate wrap-up and available events in the same table, and so on.

5.1 Get agent events for a single agent (fnc_agent_events_window)

Please see more information, explanations and examples in chapter 2.4.7.

Object name:	*fnc_agent_events_window*
Type:	Table value function
Description:	Delivers the so called *agent events* for a particular agent, and for a specified period window. The main source is the table *agent_events*, but in addition to what's found there some new events will be created (*available* and *wrap up events*). Also the start time, the end time, and the duration, of those events which are *ongoing* at the start and/or end of the specified period window will be adjusted to "fit" into these time frame "borders". It is also possible (but not required) to specify an additional time frame (not date, but 'HH:MM') for which the selected events should be further narrowed. For example you may ask for all events starting on January 1st 2013, and ending January 31st 2013, but limited only to those events appearing between '09:00' and '11:30'. As described above, also for this additional time frame all ongoing events at the "border" will be adjusted.
Parameters:	**Name** / **Datatype** / **Description** @agent_id / int / The (unique) id for the agent to search for. @dte_from / datetime / Date/time for the start of the period. @dte_to / datetime / Date/time for the end of the period. @mode / int / A "flag" used to decide the outcome of some of the new (or old) events: • 0 = Wrap up events (type 'w') will be created and returned (in addition to the 'normal' events).

			• 1 = Wrap up (type 'w') and available events (type'a') will be created (added). 'r'-events (return from pause) will be removed (even if they exist in table *agent_events)* – because 'r' is implicit expressed through the 'a' events. • 2 = As 1, but the 'r'-events are also returned. We believe that the value of 1 is the one that will be used in most cases.
	@time_from	char(5)	Optional (though, must be in parameter list, but can be NULL or empty string). If set, the @*time_from* and @*time_to* specifies a further (more "narrow") time limit (HH:MM) within *each day* the events should belong to. In most cases, these parameters is only relevant to specify in the @*dte_from* and @*dte_to* spans over more than one day. Some rules: • Valid format is HH:MM, all five digits must always be given. • @time_to should be greater than @time_from.
	@time_to	char(5)	See description for @*time_from* above.

Returns:

Returns a table with a subset of the columns from the table *agent_events*, and with two additional columns for the adjusted start and duration values (see *Description* above.

The table returned is like this (columns without description is described under description for table *agen_events).*

Column name	Datatype	Description
agent_id	int	The (unique) id for the agent to search for.
dte_start	datetime	The (original) start for the event.
adj_dte_start	datetime	The start for the event within the period window. Will differ from *dte_start* for those events being *ongoing* at the time specified by parameter @*dte_from* / @*time_from.*
service_num	varchar(100)	
event_type	varchar(10)	
result_code	varchar(10)	
duration_sec	int	The original duration of the event.
adj_duration_sec	int	The adjusted duration of the event. Will differ from *duration_sec* for those events being *ongoing* at the time specified by parameter @*dte_to* / @*time_to*.
duration_speak_sec	int	

	queue_key nvarchar(100) pause_type_name nvarchar(100) pause_type_id int internal_odr_id int

Samples:	• Get the events for agent 9999 between 08:30 and 16:00 on May 2nd 2013:
	`select *`
	`from [dbo].[fnc_agent_events_window](`
	`9999, '2-may-2016 08:30', '2-may-2016 16:00', 1, NULL, NULL)`
	`order by dte_start`
	• Get the events for agent 9999 between 08:30 and 16:00 on May 2nd 2016, but **exclude Login events** (since time logged in are 'included' in Available and other events)
	`select *`
	`from [dbo].[fnc_agent_events_window](`
	`9999, '2-may-2016 08:30', '2-may-2016 16:00', 1, NULL, NULL)`
	`where event_type not in ('i')`
	`order by dte_start`
	• For agent 9999, find the total sum per event (and per pause type) between 08:30 and 16:00 on May 2nd 2016:
	`select event_type, pause_type_name, count(*) Ant, sum(adj_duration_sec) AS sumAdjDurSec`
	`from [dbo].[fnc_agent_events_window](`
	`9999,'2-may-2016 08:30:00', '2-may-2016 16:00', 1, NULL, NULL)`
	`group by event_type, pause_type_name`

5.2 Get agent events for multiple agents (fnc_all_agents_events_window)

<table>
<tr><td>Object name:</td><td>fnc_all_agents_events_window</td></tr>
<tr><td>Type:</td><td>Table value function</td></tr>
<tr><td>Description:</td><td>Delivers the same information as described for function fnc_agent_events_window above, except that it's delivered for all agents for a specified customer. Except from the @customer_key parameter which replaces the @agent_id parameter, all other parameters are the same – as well as the returned table.</td></tr>
<tr><td>Parameters:</td><td>
<table>
<tr><th>Name</th><th>Datatype</th><th>Description</th></tr>
<tr><td>@customer_key</td><td>varchar(100)</td><td>The customer_key for whom we shall get events for all the agents.</td></tr>
<tr><td>@dte_from</td><td>datetime</td><td>See fnc_agent_events_window.</td></tr>
<tr><td>@dte_to</td><td>datetime</td><td>See fnc_agent_events_window.</td></tr>
<tr><td>@mode</td><td>int</td><td>See fnc_agent_events_window.</td></tr>
<tr><td>@time_from</td><td>char(5)</td><td>See fnc_agent_events_window.</td></tr>
<tr><td>@time_to</td><td>char(5)</td><td>See fnc_agent_events_window.</td></tr>
</table>
</td></tr>
<tr><td>Returns:</td><td>Returns a table is identical to the one described for fnc_agent_events_window.</td></tr>
<tr><td>Samples:</td><td>

Get the events for all agents for customer 12345 between 08:30 and 16:00 on May 2nd 2016:
<pre>
select *
from [dbo].[fnc_all_agents_events_window](
        '12345,
                '2-may-2016 08:30',
                '2-may-2016 16:00',
                 1, NULL, NULL)
order by dte_start
</pre>

This is a more advanced query, where we get the duration for any event, per agent and per day, for all agents for customer 12345 between June 3rd and (up to, not included) June 8th. But only between '08:00' and '16:00' for each of the days. Also the agents table is joined in order to get agent's name.
<pre>
select a.agent_id, full_name,
datepart(year,adj_dte_start) YR,
</pre>

</td></tr>
</table>

```
datepart(month,adj_dte_start) MNTH,
datepart(day,adj_dte_start) [DAY],
event_type,
sum(adj_duration_sec) duration_sec,
pause_type_name
 from [dbo].[fnc_all_agents_events_window]('12345',
'3-jun-2016', '08-jun-2016', 1,'08:00','16:00') AS events,
 agents a
 where a.agent_id=events.agent_id
 group by a.agent_id, a.full_name, datepart(year,adj_dte_start),
datepart(month,adj_dte_start),datepart(day,adj_dte_start),
        event_type, pause_type_name
order by 1,2,3,4,5, pause_type_name
```

5.3 Search for (incoming) calls (fnc_search_for_calls)

Object name:	*fnc_search_for_calls*
Type:	Table value function
Description:	A "helper" function to make is easier to search for calls – given a set of known criterias.
Parameters:	See table below.
Returns:	Returns a table containing one and only one column; *call_id* with datatype *numeric(19,0).*
Samples:	• Get the all call_events for answered calls coming from phone '87654321' between 09:00 and 11:00 on June 3rd 2016:

Name	Datatype	Description
@source_number	varchar(100)	The caller's number.
@destination_number	varchar(100)	The number called
@dte_from	datetime	The earliest start for the call(s)
@dte_to	datetime	The latest start for the call(s)
@answered	bit	Should the call have been answered or not? Set 1 for yes, 0 for no, and NULL if you don't have any opinion.

Note! The parameters might be empty/null. The more you specify as parameter values, the less will the result be.

```
select * from call_events
where call_id in (select call_id from dbo.fnc_search_for_calls
('42793704',
NULL,
'3-jun-2016 09:00',
'3-jun-2016 11:00',1))
```

5.4 SQL queries to generate reports

We have made a couple of SQL queries that generates simplified versions of the reports *Details per queue* and *Details per agent per queue*. These queries are available in the document “Raw data - SQL queries for reports” on help.puzzel.com under Knowledgebase - Developers.

6 Guideline for downloading to own local database(s)/system(s).

Let's start to divide the database tables into two types; "basic tables" and transactional tables.

Basic tables are:

- agent_event_source
- agent_event_types
- agent_result_codes
- agents
- call_event_types
- call_result_codes
- media_types
- menues
- queues
- phone_types

Transactional tables are:

- call_events
- agent_events
- enqreg_header
- enqreg_category
- enqreg_topic
- surveys
- call_variables
- call_relations
- call_event_extras

6.1 Transfer content of the basic tables

The basic tables are relatively small. If you need them locally, a complete download on a regular basis is recommended. There is no "date stamp" (or similar) to support easy selection of changes only.

6.2 Transfer content of the transaction tables

The tables *call_events, agent_events*, *enqreg_header* and *surveys* is normally bigger, and an incremental transfer of the data (i.e only new and changed record) is strongly recommended.

NOTE(!) that Puzzel, according to contract, have the right to strongly reduce, or even disconnect, connections forcing too heavy data transfer load on the connections.

Common to all four tables above is that they have a column named *dte_updated*. This is a date stamp holding the exact time for when the record was inserted or last updated. Vi suggest that you, per table, keep track of the highest value for this column after each transfer - and use this value as a starting point the next time you select from the tables.

The tables *call_events_extras*, *call_relations* and *call_variables* do not have an explicit *dte_updated* column. Though, they all relate to a *call_id* in table *call_events* (*call_relations* and *call_variables*), or to the *rec_id* in *call_events* (*call_events_extras*). Ensure loading of records from these three tables each time the "parent" (or "owner") in table *call_events* are loaded.

Important for *call_events* and *agent_events* records:
For different reasons, part of the data for a call (call_id) might be added on a later stage. This is caused either by later callback sessions (related to the original call), by changed wrap-up information caused by the agent after the (related) call ended, or due to a Dialler call happening (much) later then the contact was put into the queue. In such cases, **all data (events) for the call_id are removed and reloaded with a complete set of the new/modified data**. This will also include the *agent_event* table if agents are involved in the call (agent_events of type 'c'). The column *rec_id* in the tables could therefore change over time for a particular call. If you need to avoid double loading of *call_events/ agent_events* records in such cases, you will probably need a locally stored history log holding the *call_ids* loaded recently.

"Late update" of a call may only occur for a limited period of time. For the time being, no calls will be updated after 72 hours. There is one exception; emails might have a "gap" between the pre-initiation event and "the rest" with duration of up to 24 days.

Important for *enqreg_header, enqreg_category* and *enqreg_topic*: The records in these three tables are related, and records related to the same enquiry registration will always be loaded together. Thus, you can rely on the column *dte_updated* in *enqreg_header* to detect new records in all three tables. During normal operations, an enquiry registration will never be changed after arrival to the database.