Chat

Chat (and config. of max concurrent written requests)

If your Puzzel solution handles chat (email/social) 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, several properties are defined, the most important being:

- **Maximum total concurrent** written requests: X
- Max concurrent chat/social requests
- Max concurrent email/task requests
- **Block phone** if number of written requests is greater than: Y

If an agent is connected to a phone call, a written request is not allocated to the agent.

An agent can be allocated a new written request from queue if he/she has “capacity” for more, given the agent’s values for Max total concurrent, Max concurrent chat/social and Max concurrent email/task. See more details about how it works here.

In the Puzzel internal database we create one “secondary agent” per possible written request for the user, which is the “main agent”. This is similar to Group number/Unblockable agent. If the agent can handle 4 written requests (X=4) there will be 4 secondary agents for this (main) agent.

Examples:

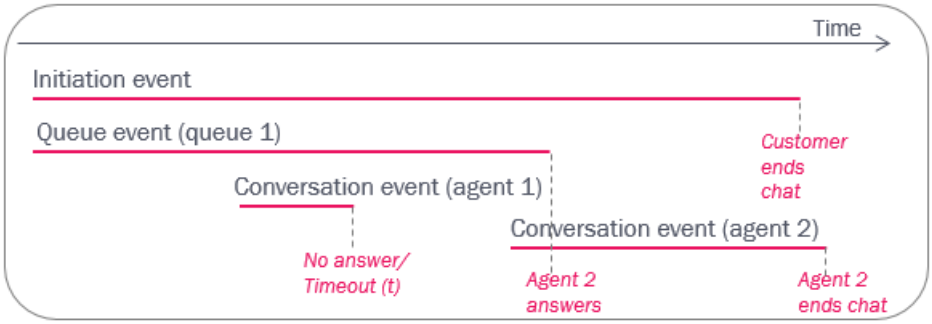
1. If Y is 0, the agent will not receive a phone call if he already has a written request.
2. If Y is 2, the agent can receive a call if the agent has maximum 2 written requests.

The number of active(open) written requests for a user is shown in brackets next to the agent’s status, e.g*Ready (0), Ready (1), Busy(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 secondary agent’s agent-id (not the same id as is used when the agent is offered a phone call). To find the secondary agent’s corresponding main 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.

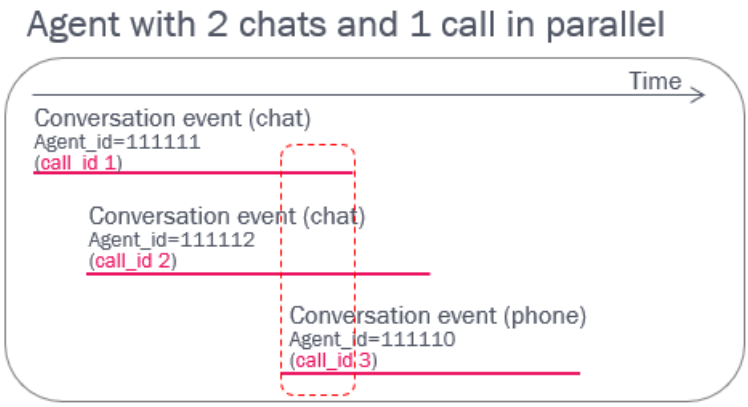


If the queue tries to send a chat (or any written request) to an agent that is logged on to queue but that has closed his/her Puzzel agent application, the queue will after a while send the chat (written request) to the next ready agent. The conversation event for the logged on agent whose Puzzel agent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

Who ended the chat?

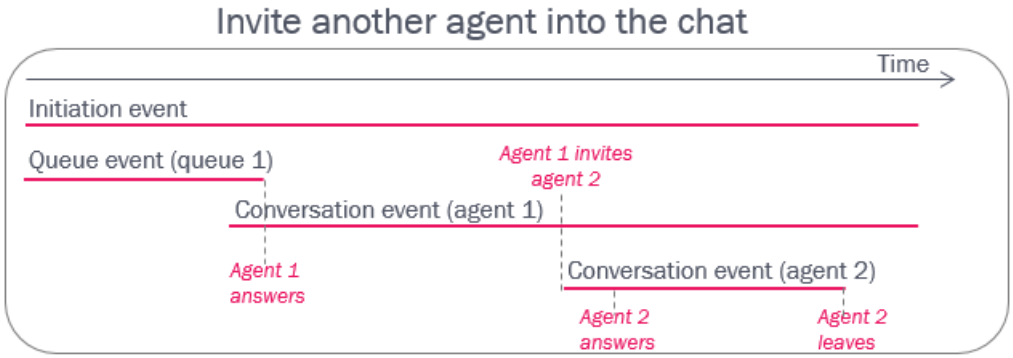
- If the person that initiated the chat ends the chat before the agent closes the chat tab (the normal situation), the Initiation event will have a Finish earlier in time than the Conversation event finish, since the agent will close the chat request tab in the agent application some seconds or minutes later.
- If the agent tries to end the chat before the chatter has disconnected, the agent will get a warning, and if agent still ends, the Initiation event will have a Finish some ms later than the Conversation event's 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).

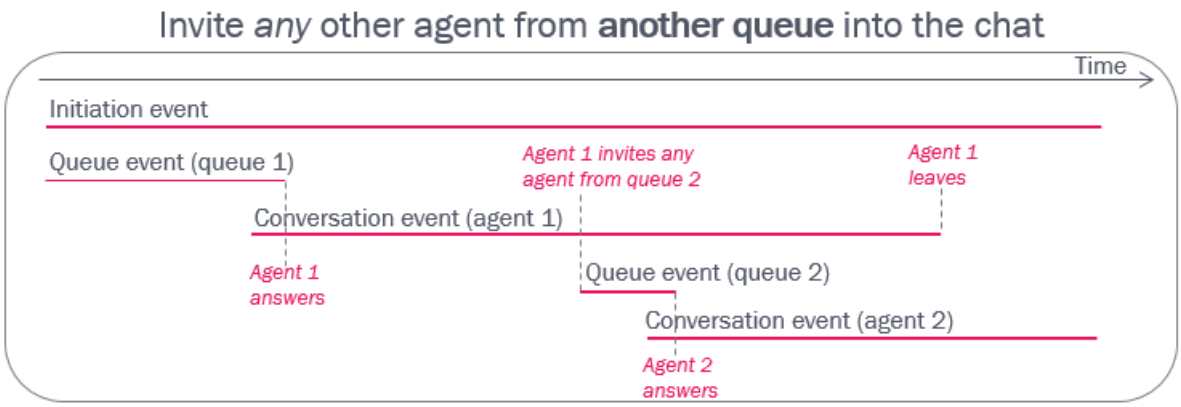


By looking in the table agents, you'll find that agent_id 11 and 12 both are «secondary agents» belonging to the same main agent, here agent_id 1.

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).



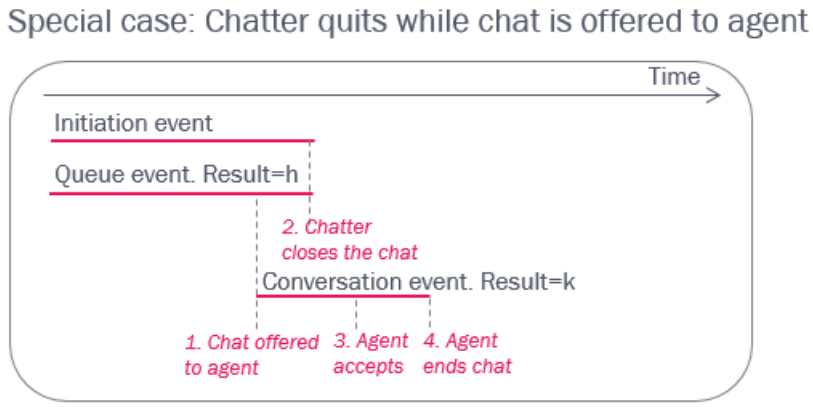
The agent may also invite any agent from another queue (the first ready agent will be offered the chat). This will result in a new Queue event and a Conversation event for the agent that gets the chat from the queue.



> **Note**
>
> 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.

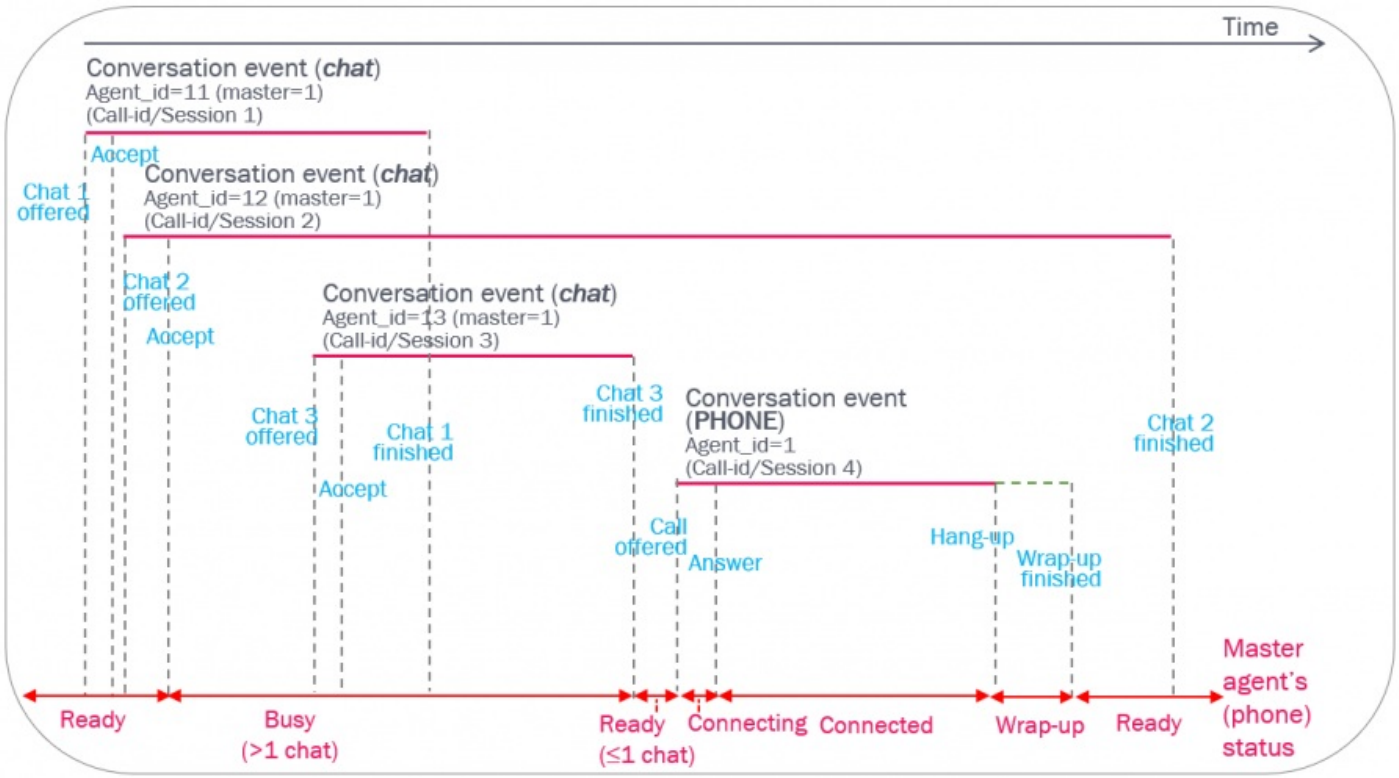
A very special case

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 written requests is greater than: 1
- Max parallel written requests: 3

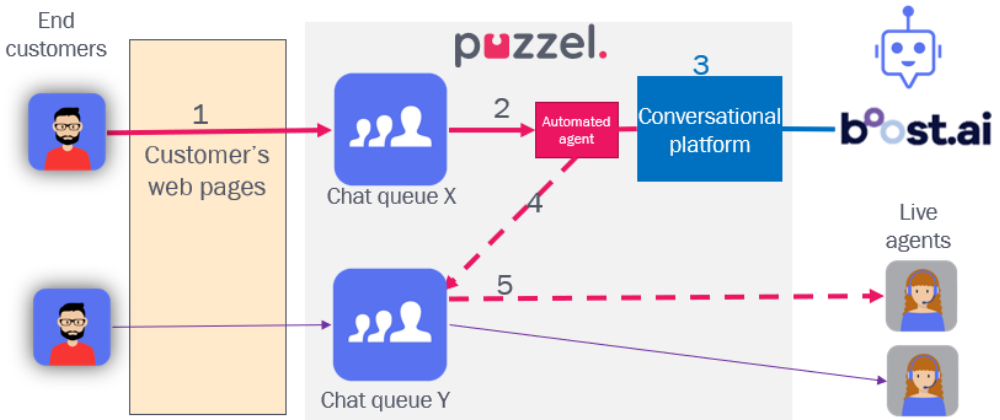


Automated agent and chat-bot

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(gs) (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.



1. End-customer starts the chat
2. Queue sends chat to the automated agent which runs a script
3. The automated agent sends chat to the *Conversational platform* which is the bridge to the external bot (which runs the natural language conversation).
4. The chatbot tells the Conversational platform to (tell the automated agent to) transfer the chatter to another queue
5. The live agent accepts the chat and continues the conversation with the end-customer