Conversations

One of the trickiest things in bot design, is to bring your various flows together in conversations. It's easy to create a bot that handles a single task, but this is rarely enough. In fact, even when your bot is designed for one task only, users may ask about other things. This is a fundamental difference between for example GUI design and bot design. In a GUI users may only interact by means of the things that are actually there. Conversations on the other hand, have no obvious borders. So even the smallest bot, needs to have at least one extra flow for dealing with things out-of-scope.

Another special (and cool) thing about conversations is that users may jump from one topic to another without following some anticipated pattern. The user may even interrupt an ongoing flow to ask for supplementary information and then expect the flow to be resumed.

The good news is that these things are exactly what Teneo is built to help you with.

Conversations vs Flows

Flows implement the logic needed to resolve a specific user's question. They can be simple (e.g. a flow that answers the question "What is your favorite color?") or complex (e.g. a flow that helps the user booking a flight), but they always represent some kind of pre-defined conversational pattern. In contrast, a full conversation is rarely foreseeable. Luckily however, you don't have to map it all out!

Let's look at an example:

**User**: I want to buy a ticket  
**Bot**: Which ticket would you like, A or B?

**User**: What is the price of ticket A? (buy ticket flow is interrupted)  
**Bot**: Ticket A is 1 million dollars!  
**Bot**: Which ticket would you like, A or B? (buy ticket flow is resumed)

**User**: Ticket B please!  
**Bot**: Great, ticket B is free!

# Flow stack

Teneo manages the active flows in a flow stack. When a flow is triggered it's put on top of the stack, and as soon as it's completed, it will be removed from the stack.

The flow stack is thus made up of all the flows that have been triggered but have not yet finished.

The top flow is the flow that is currently being processed.

## Flow stack and interruptions

Generally there is only one flow on the stack at a time. The stack may grow higher if an active flow gets interrupted. This may happen if the user asks for a clarification instead of responding to the bot's question, or responds with something that the bot does not understand. In those situations another flow will be triggered and put on top of the stack. That flow is then the active flow. Once it's completed, Teneo will drop it from the stack, and the interrupted flow will be back on top and resumed.

You can easily specify if your flow should remain on the flow stack if interrupted. Flows are only kept on the stack if the output that was interrupted was set to be [revisitable](https://www.teneo.ai/studio/flows/concepts/outputs" \l "revisitability).

## Flow stack control

Teneo manages the flow stack automatically for you. You can however access and manipulate the flow stack programmatically using the [Engine scripting API](https://www.teneo.ai/studio/scripting/concepts/engine-scripting-api). You may for instance force the stack to be cleared in certain situations. More on that here: [How to clear the flow stack](https://www.teneo.ai/studio/scripting/concepts/engine-scripting-api#clear-the-flow-stack).

# Ordering

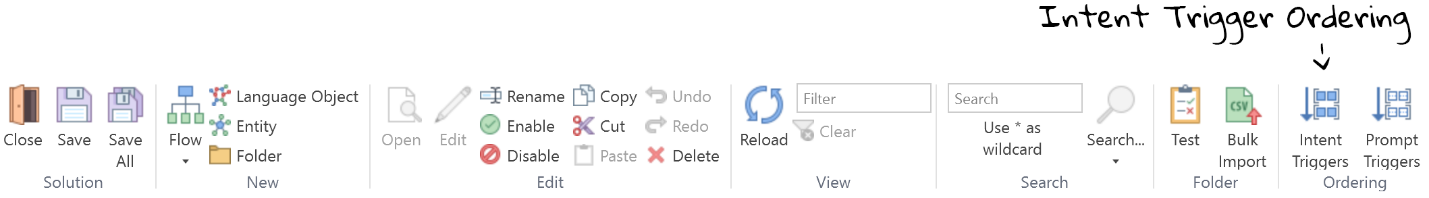
When Teneo tests a user input against your solution's triggers and transitions, it stops at the first match. It is thus crucial in which order the triggers and transitions are tested. Teneo allows you to define and modify their order. This concept is called ordering.

## Active flows first

When searching for a matching trigger, Teneo always gives priority the flow that is currently being processed. Thus, each input is first tested against the transitions at the current point of the active flow. The order in which these transitions are tested is [indicated in the flow graph](https://www.teneo.ai/studio/flows/concepts/transitions#order). Only if there is no active flow or if none of the transitions in the active flow match the user input, Teneo will go on and test the input against other flow triggers of the solution.

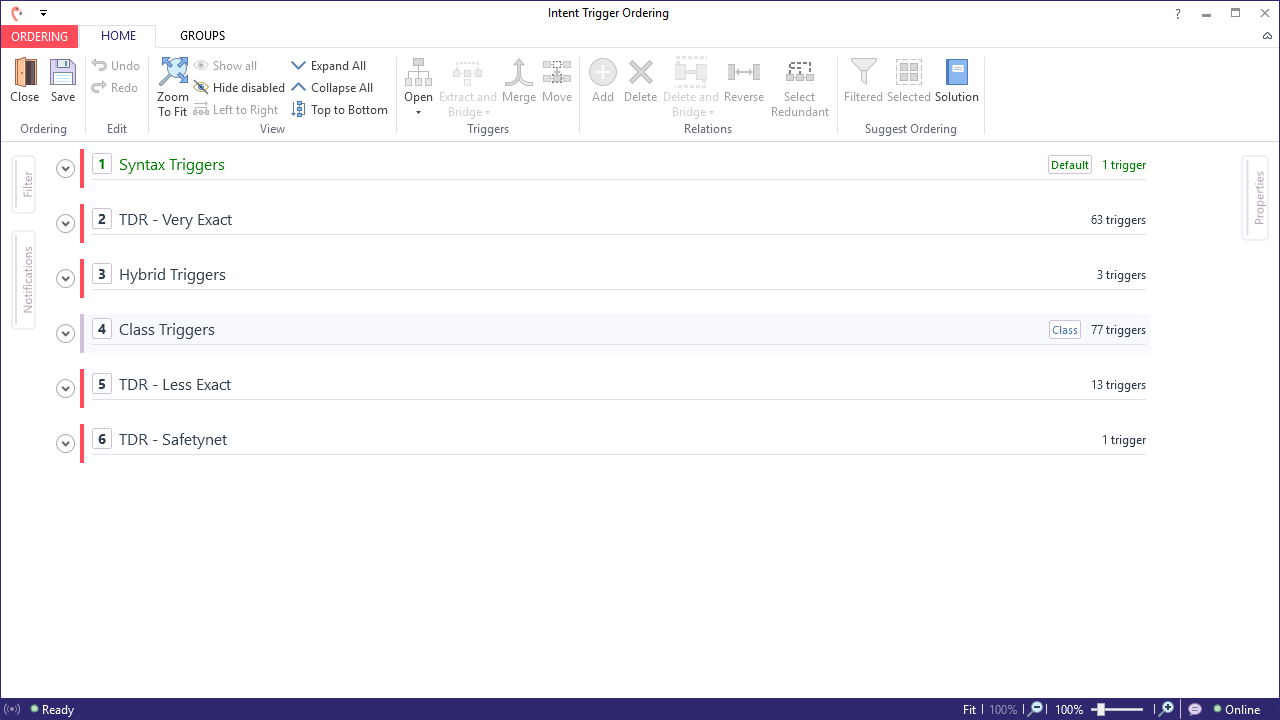
## Order groups

If there is no active flow, or no matching transition in the active flow, Teneo will go through all other flows of the solution to find a matching trigger. The order in which the triggers are tested is determined by the triggers' order group. To get an overview of the order groups of your solution, simply select the 'Intent Trigger' icon in the 'Ordering' section of the top ribbon.



This opens an overview of the existing order groups in your solution. When searching for a matching trigger, Teneo will start at the top and work its way down until it finds a matching trigger.

.



The order groups displayed above are the groups that come with Teneo Dialogue Resources (which you can use as a template when creating a new solution). Let's have a quick look at all of these groups:

1. **Syntax Triggers** - group to store syntax triggers. Because this is marked as 'default', syntax triggers you add to your solution will automatically be assigned to this group
2. **TDR - Very Exact** - contains triggers from Teneo's Dialogue Resource template solution, which are very specific.
3. **Hybrid Triggers** - this is the group you should assign [hybrid triggers](https://www.teneo.ai/studio/language-understanding/how-to/create-a-hybrid-trigger#assign-the-correct-order-group) to. Note that this assignment does not happen automatically.
4. **Class Triggers** - all class triggers you have created in your solution will automatically be assigned to this group.
5. **TDR - Less Exact** - contains the triggers from Teneo's Dialogue Resource template solution that are less specific.
6. **TDR - Safetynet** - this group contains the safetynet that comes with Teneo's Dialogue Resource template solution.

These pre-defined groups should take you a long way. However, should you discover that you need to add more order groups to your solution, you can easily do so by moving over to the 'Groups' tab of the 'Intent Trigger Ordering' window. There, you may also move triggers from one group to another, if necessary. But before you add order groups or start moving triggers, you may want to try whether 'Relative Ordering' does the job.

## Relative Ordering

If you observe that several syntax or hybrid triggers of the same order group are competing for inputs, you can define a relation between these triggers. This happens by drawing arrows between the triggers. This relative ordering clarifies the order in which the triggers should be tested. And just like with the order groups, the more specific triggers should be put before the less specific ones. Otherwise, the more specific triggers will never be tested.

Relative ordering cannot be applied to class triggers!

## Prompt Triggers

We have so far seen how to order Intent triggers, which make up for the majority of triggers in our solution. Prompt triggers are ordered separately from the intent triggers, but their ordering follows the same general principle: more specific triggers should be tested before less specific ones. In order to view or change the ordering of prompt triggers, simply select the 'Prompt Trigger' icon in the Ordering section of the top ribbon.