

Action Bot

Owner:

Drew Hamrock (drew.hamrock@appian.com) Last updated - 4/12/24

Purpose:

Assist the user in quick actions or questions around their current work. Instead of navigating to a specific record to find information or perform a related action, do this instead from this bot on the dashboard. This component uses GPT3.5 Turbo (default) and GPT4 through Azure. Click the Bot title to switch configurations. To configure the parameters of this bot, continue to [configurations](#).

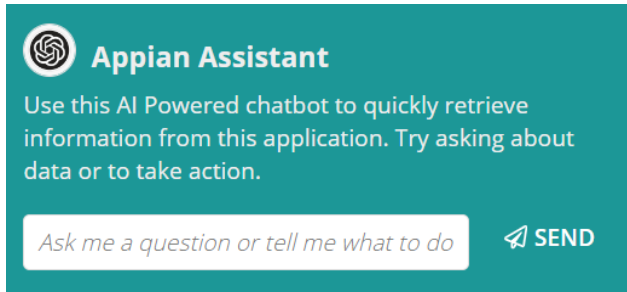
Action Prompts:

There are two types of actions: Record actions and related actions. Record and related actions play nice out of the box, but you can customize further to have additional actions based on the way I set triggers for the output response. They do not have to be record/related actions specifically, but those are easier to set up and allow for the dialogue box.

Your prompt **does not need to be exactly what your prompt data reflects** – it should understand what you want. That being said I wouldn't try to confuse it too much.

Questions:

If you would like to ask the bot a question, you can do that as well. The bot has been trained on data using the same method as in the [ChatGPT Data Bot](#) utility. Adding a "?" at the end of your question will ensure that your bot knows how to respond, but this isn't required – just good to know. This bot does not have memory of your conversation, so don't build prompts off one another.



Configurations:

To begin setup navigate to CBB_configurations. You will see details on how to set configurations for the bot. Use these as presets for the users depending on their roles. Here are examples of configuring inputs.

```
</> CBB_configurations
value: ri!configurationId,
default: {
  turnOffActions: false,
  shape: "SEMI_ROUNDED",
  chatColor: "#041E42",
  promptData: rule!CBB_flexQuery(recordType: recordType!CBB Prompt),
  recordType: recordType!WN Event,
  recordActionList: {
    recordType!WN Event.actions.newEvent,
    recordType!WN alert.actions.newAlert,
    recordType!WN absenceReport.actions.newAbsenceReport,
    recordType!WN Employee.actions.newEmployee
  },
  relatedActionList: {
    recordType!WN Employee.actions.updateEmployee,
    recordType!WN Event.actions.updateEvent,
    recordType!WN alert.actions.updateAlert,
    recordType!WN Operation.actions.updateOperation,
    recordType!WN Location.actions.updateLocation
  },
  relatedActionData: {
    rule!CBB_flexQuery(
      recordType: recordType!WN Employee,
      fields: {
        WN Employee.id,
        WN Employee.name
      },
      batchSize: 200,
    ),
    rule!CBB_flexQuery(↔),
    rule!CBB_flexQuery(↔),
    rule!CBB_flexQuery(↔),
    rule!CBB_flexQuery(↔)
  },
}
```



CBB_configurations



```
195 ▶ then: {↔},
277 equals: 3,
278 ▼ then: {
279     turnOffActions: false,
280 ▼     promptData: {
281 ▼         toString(
282 ▼             {
283 ▼                 a!map(id: 1, prompt: "Create new event", response: 1),
284 ▼                 a!map(id: 2, prompt: "Update event", response: 2)
285 ▼             }
286 ▼         )
287     },
288     recordType: recordType!WN Employee,
289 ▼     fields: {
290         WN Employee.wnMessage
291     },
292 ▼     filters: {
293 ▼         a!queryFilter(
294             field: WN Employee.id,
295             operator: ">",
296             value: 50
297         )
298     },
299 ▼     recordActionList: {
300         recordType!WN Event.actions.newEvent
301     },
302 ▼     relatedActionList: {
303         recordType!WN Event.actions.updateEvent,
304     },
305     relatedActionData: {
306
307
308 ▶     rule!CBB_flexQuery(↔),
```

Configuration Parameters:

rule!CBB_ActionBot(configurationId)

The configurationId will point to your CBB_configurations expression rule with the following parameters as presets.

displayTitle (Text): The title to be displayed above the chatbot. Default "Appian Assistant".

promptData (Any Type): The record type that contains your prompt training data. Usually a dataset with three columns: id, prompt, response. Prompt containing the keywords associated to trigger the action. The response field should match the ID field. You can use a CBB_FlexQuery wrapper around a recordType, or simply just use an toString() over an a!map() of prompt data.

recordType (Any Type): The record type that will be used to answer data related questions.

filters (Any Type): List of a!queryFilters to apply to the query. This is where you could apply a filter to a record summary using a rule input.

fields (Any Type): The fields you want to use in the data query. Default grabs all the fields in the record type. To add an additional relationship, key the relationship from the base record type. It will then grab all the related data fields from that relationship.

batchSize (Any Type): Batch size. Default 100.

recordActionList (Any Type): List of record actions. Make sure this aligns EXACTLY to the order listed in the promptData.

relatedActionList (Any Type): List of related actions. Make sure this aligns EXACTLY to the order listed in the promptData.

showAILogo (Boolean): Boolean to show AI logo. Default true.

shape (Text): Shape parameter for the card layout of the bot.

decorativeBarPosition (Text): DecorativeBarPosition parameter for card layout.

decorativeBarColor (Text): DecorativeBarColor parameter for card layout.

chatColor (Text): Hex color option for chat bot color. Default "#1C9797"

relatedActionData (Any Type): Data queries for the related action identifier you are searching for. These should be formatted using the CBB_flexQuery expression rule.

These should be in EXACT order as listed in relatedActionList. Ask yourself the question "What data would the bot need to look through to find this object?" Narrow down the fields in CBB_flexyQuery to just the records Id field and the field you will likely be searching with.

chatIconSize (Text): The size of the icon next to the chat bubbles. Default "MEDIUM"

turnOffActions (Boolean): Rule input to turn off the actions feature. Default false.

Example Prompt Data:

↑	Id	Prompt	Response
	1	Create an event	1
	2	Send an alert	2
	3	Create absence	3
	4	Create an employee	4
	5	update employee	5
	6	Update event	6
	7	Update alert	7
	8	Update operation	8
	9	Update location	9
			9 items