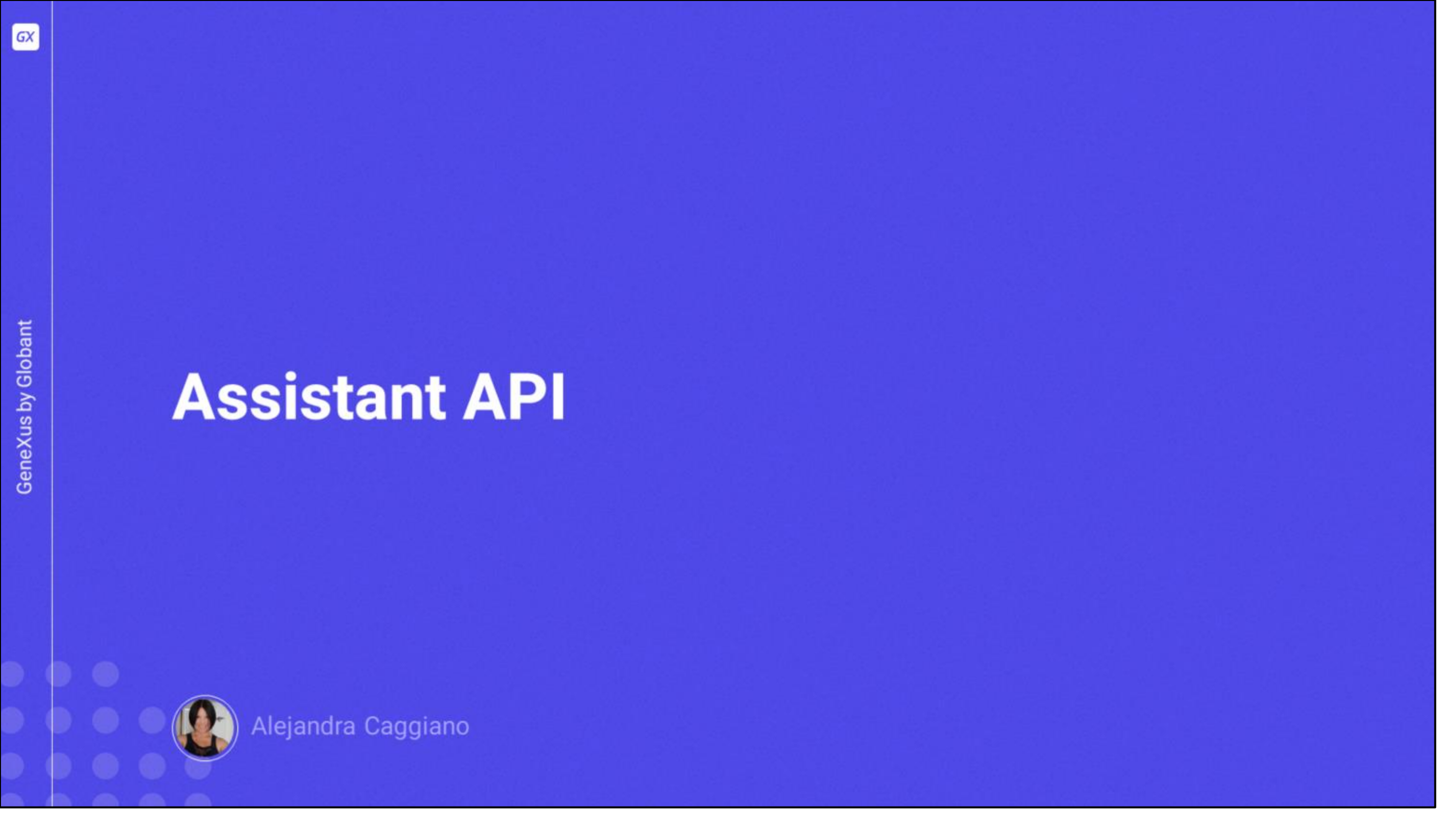
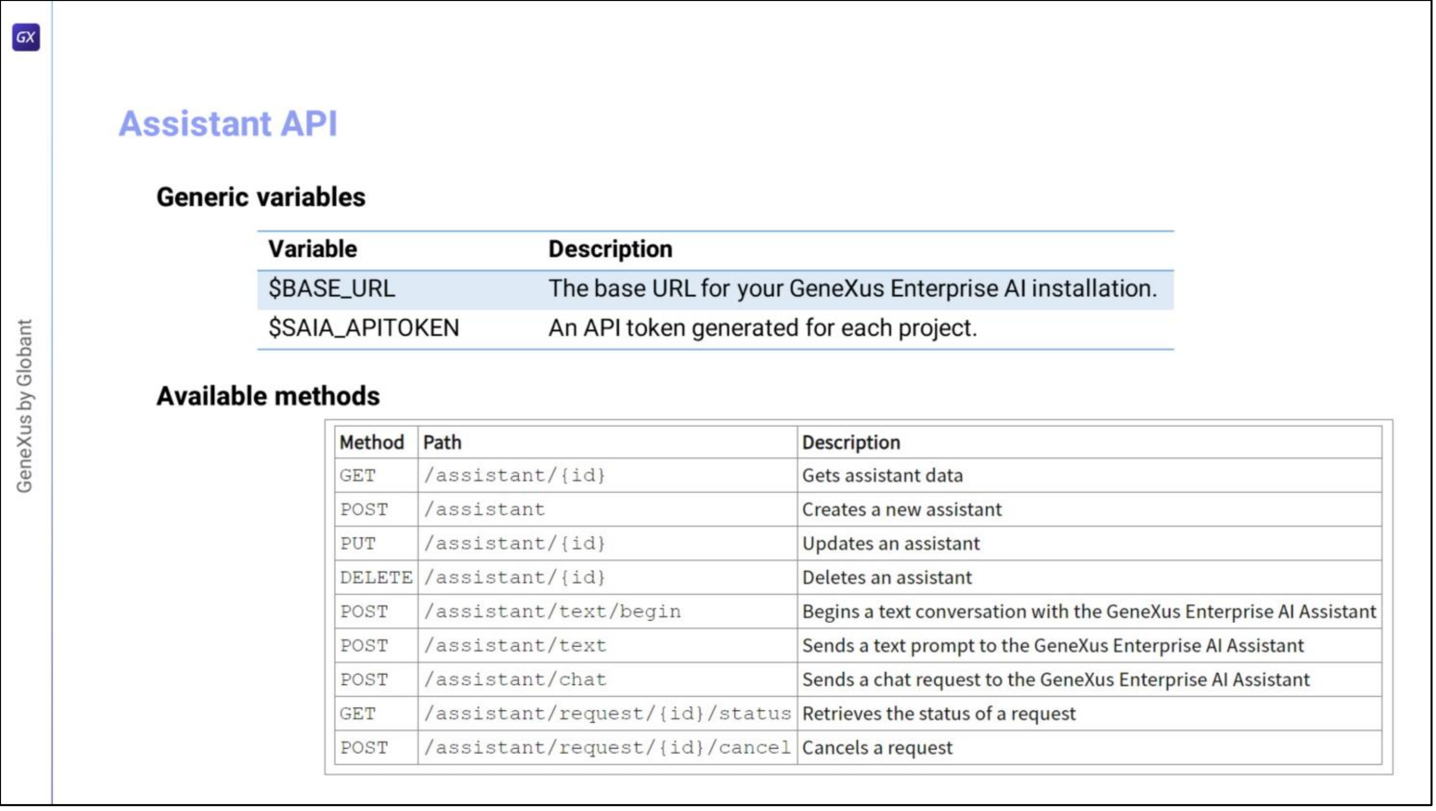


Assistant API

Alejandra Caggiano

GeneXus by Globant

Within the set of available APIs, GeneXus Enterprise AI offers one to create new assistants, modify their definitions, and retrieve their information.



Assistant API

Generic variables

Variable	Description
$BASE_URL	The base URL for your GeneXus Enterprise AI installation.
$SAIA_APITOKEN	An API token generated for each project.

Available methods

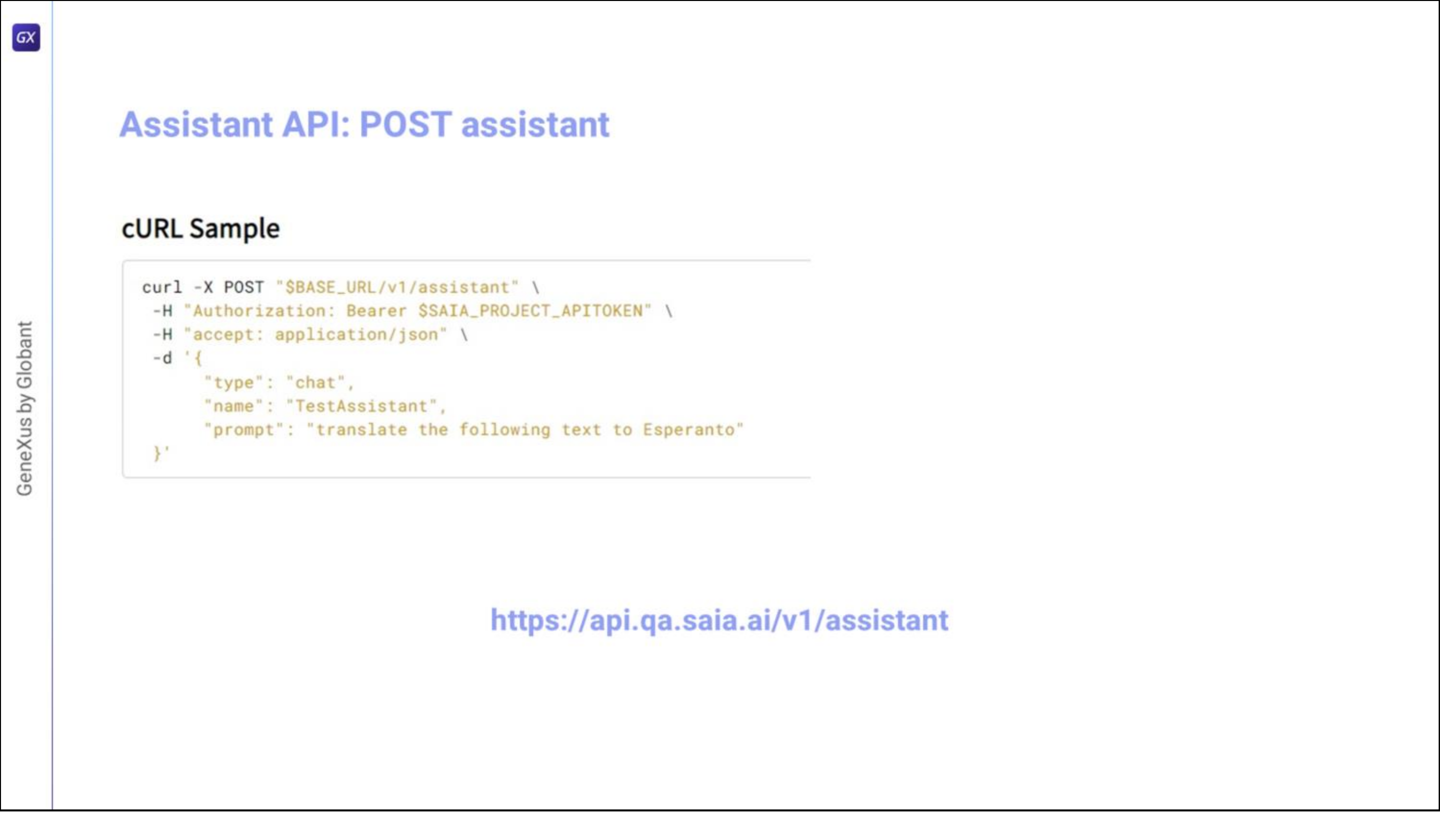
Method	Path	Description
GET	/assistant/{id}	Gets assistant data
POST	/assistant	Creates a new assistant
PUT	/assistant/{id}	Updates an assistant
DELETE	/assistant/{id}	Deletes an assistant
POST	/assistant/text/begin	Begins a text conversation with the GeneXus Enterprise AI Assistant
POST	/assistant/text	Sends a text prompt to the GeneXus Enterprise AI Assistant
POST	/assistant/chat	Sends a chat request to the GeneXus Enterprise AI Assistant
GET	/assistant/request/{id}/status	Retrieves the status of a request
POST	/assistant/request/{id}/cancel	Cancels a request

To use this API, we must consider the generic variables we already know: **&Base_URL** and **&SAIAAPiToken**.

In addition, a GeneXus Enterprise AI API token related to the scope of the organization is also required.

The methods available for this API are as follows.

For example, let's try the POST method that allows creating a new assistant.



GeneXus by Globant

Assistant API: POST assistant

cURL Sample

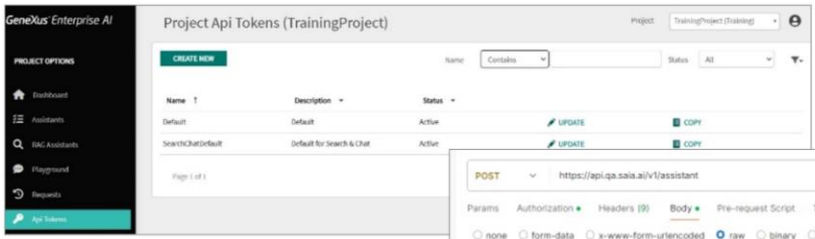
```
curl -X POST "$BASE_URL/v1/assistant" \
 -H "Authorization: Bearer $SAIA_PROJECT_APITOKEN" \
 -H "accept: application/json" \
 -d '{
      "type": "chat",
      "name": "TestAssistant",
      "prompt": "translate the following text to Esperanto"
 }'
```

https://api.qa.saia.ai/v1/assistant

The corresponding cURL sample, which is available in the GeneXus Enterprise AI technical documentation, indicates that the URL must have the following format:

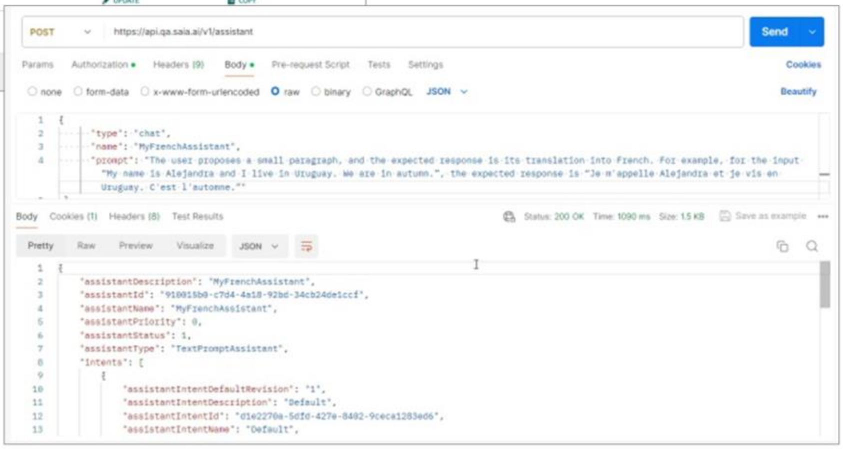
Therefore, in our example, the URL is as follows: https://api.qa.saia.ai/v1/assistant where this part corresponds to the content of the variable BASE_URL applied to our context.

Assistant API: POST assistant



Project Api Tokens (TrainingProject)

Name	Description	Status		
Default	Default	Active	UPDATE	COPY
SearchChatDefault	Default for Search & Chat	Active	UPDATE	COPY



POST https://api.qa.saia.ai/v1/assistant

Body: raw, JSON

```
{
    "type": "chat",
    "name": "MyFrenchAssistant",
    "prompt": "The user proposes a small paragraph, and the expected response is its translation into French. For example, for the input
      "My name is Alejandra and I live in Uruguay. We are in autumn.", the expected response is "Je m'appelle Alejandra et je vis en
      Uruguay. C'est l'automne.""
```

Status: 200 OK Time: 1090 ms Size: 1.5 KB

```
{
    "assistantDescription": "MyFrenchAssistant",
    "assistantId": "910015b0-c7d4-4a18-92bd-34cb24de1ccf",
    "assistantName": "MyFrenchAssistant",
    "assistantPriority": 0,
    "assistantStatus": 1,
    "assistantType": "TextPromptAssistant",
    "intents": [
        {
            "assistantIntentDefaultRevision": "1",
            "assistantIntentDescription": "Default",
            "assistantIntentId": "d1e2270a-5dfd-427e-8402-9ceca1283ed6",
            "assistantIntentName": "Default",
```

MyFrenchAssistant

"My name is Alejandra and I live in Uruguay. We are in autumn."

"Je m'appelle Alejandra et je vis en Uruguay. C'est l'automne."

Good. We enter Postman and declare the POST.

Note that the authorization type is Bearer and a Project Api Token is required.

So, from the platform, we go to the Api Tokens option and copy the default.

We go back to Postman, and define the authorization required.

In type, we enter Bearer token and paste the token.

To define the body we need to indicate the type of assistant, its name and prompt.

We intend to create a chat assistant that will return the translation into French of a short paragraph entered by the user.

Its name will be "MyFrenchAssistant" and the prompt will say that it is a personal assistant. The user enters a short paragraph and the expected response is its translation into French. For example, for the input "My name is Alejandra and I live in Uruguay. We are in autumn." the expected answer is "Je m'appelle Alejandra et je vis en Uruguay. C'est l'automne."

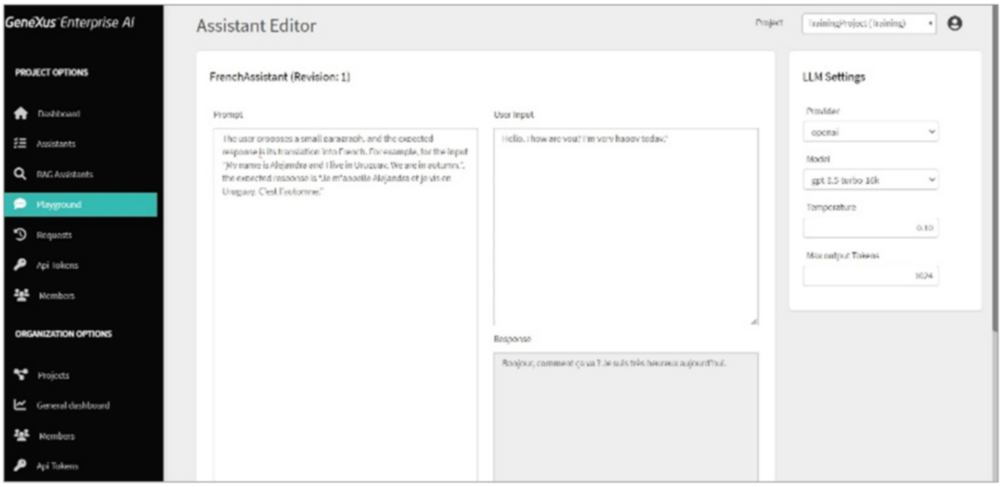
In the Body tab, we select Raw, JSON, and define the structure.

The type of assistant is Chat, its name is MyFrenchAssistant, and we paste the prompt we mentioned before.

We click on Send and see the answer with the creation of the assistant.

Among the information provided, we see that revision 1 was created; that is, the first version of the assistant definition (remember that an assistant can have several revisions). We will take special note of the Id, since it is a necessary parameter for the execution of other API methods, such as GET, PUT and DELETE.

Testing the assistant: Edit prompt



Assistant Editor

Project: TrainingProject (Training)

FrenchAssistant (Revision: 1)

Prompt

The user proposes a small paragraph, and the expected response is its translation into French. For example, for the input "My name is Alejandra and I live in Uruguay. We are in autumn.", the expected response is "Je m'appelle Alejandra et je vis en Uruguay. C'est l'automne."

User Input

Hello, how are you? I'm very happy today.

Response

Bonjour, comment ça va ? Je suis très heureux aujourd'hui.

LLM Settings

Provider: openai

Model: gpt 3.5 turbo 16k

Temperature: 0.10

Max output Tokens: 1024

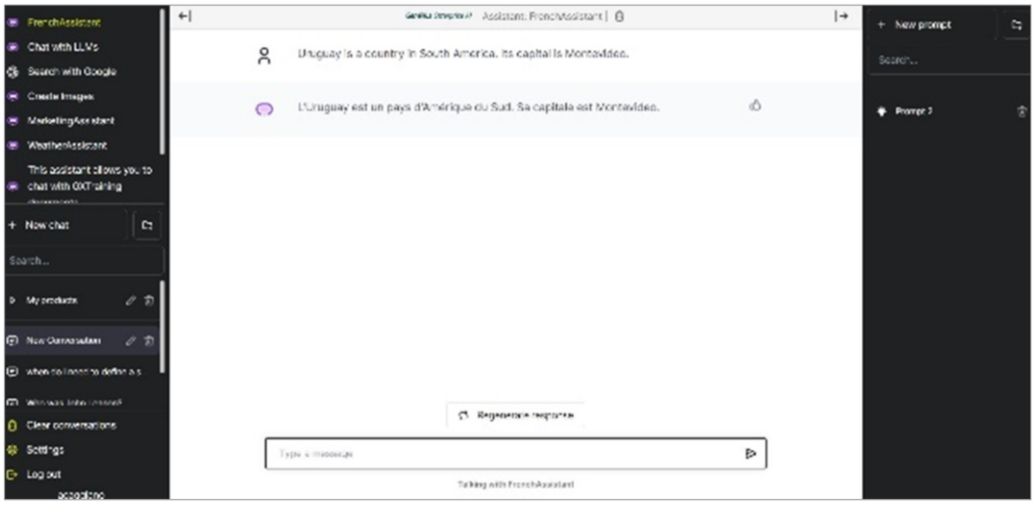
Now we want to test it, and for that we have several options.

One option is to enter directly to the platform and verify that MyFrenchAssistant is defined.

We can see the defined prompt, enter an input and test it.

For example, let's say "Hello, how are you? I am very happy today." We click on Test and see its translation into French.

Testing the assistant: Playground



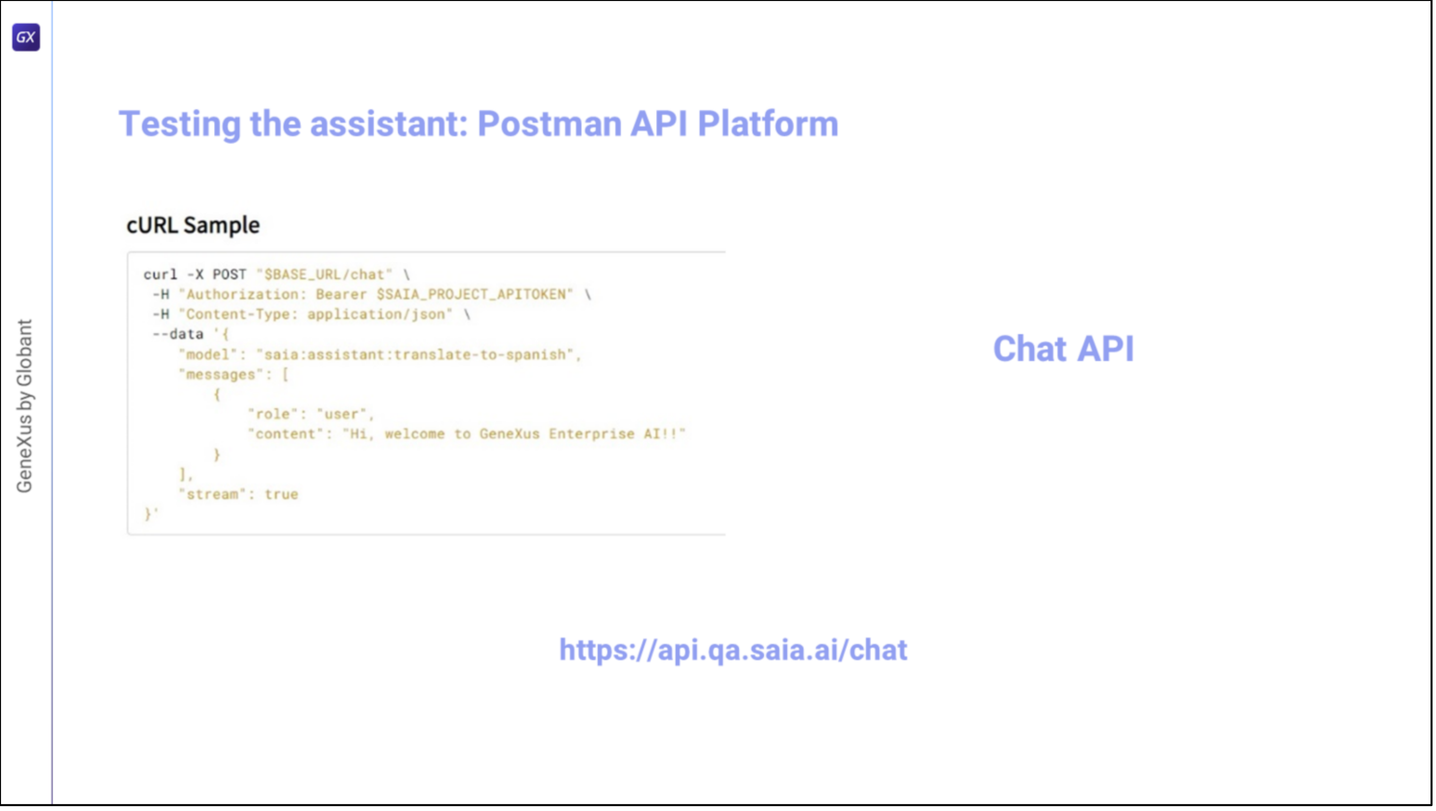
Uruguay is a country in South America. Its capital is Montevideo.

L'Uruguay est un pays d'Amérique du Sud. Sa capitale est Montevideo.

Very good. Let's test it from the Playground as well.

We select the assistant, create a new chat, and this time we say that “Uruguay is a country in South America. Its capital is Montevideo.”

And we see its translation.



GX

GeneXus by Globant

Testing the assistant: Postman API Platform

cURL Sample

```
curl -X POST "$BASE_URL/chat" \
 -H "Authorization: Bearer $SAIA_PROJECT_APITOKEN" \
 -H "Content-Type: application/json" \
 --data '{
    "model": "saia:assistant:translate-to-spanish",
    "messages": [
        {
            "role": "user",
            "content": "Hi, welcome to GeneXus Enterprise AI!!"
        }
    ],
    "stream": true
}'
```

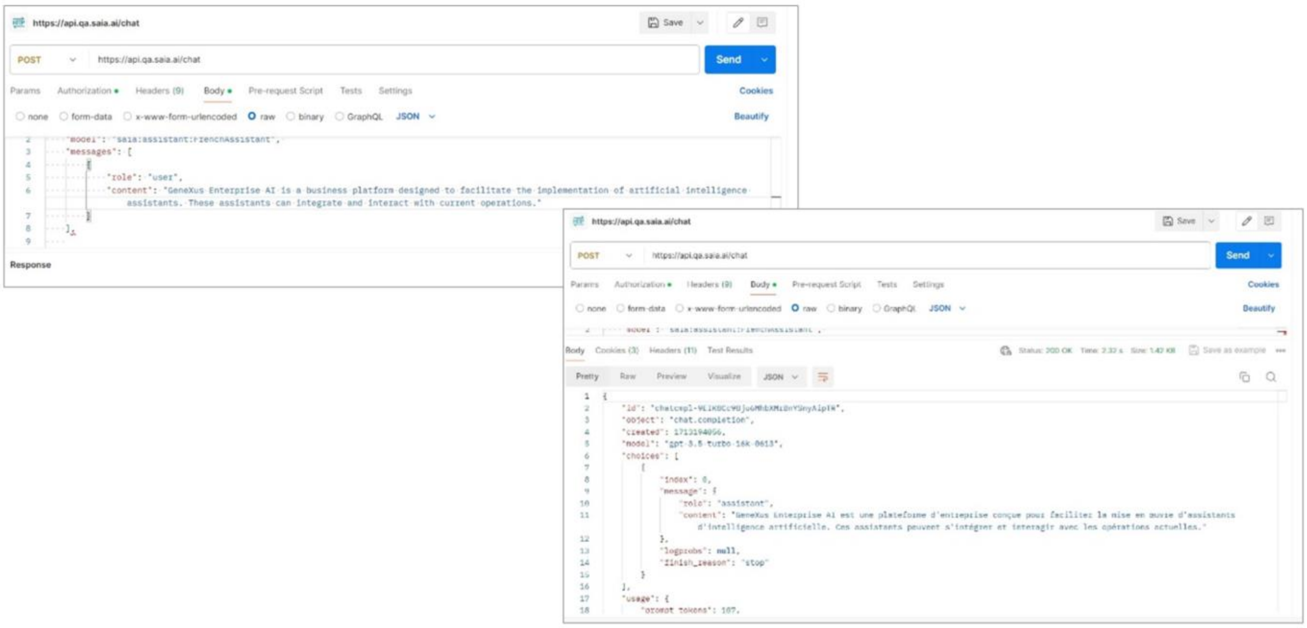
Chat API

https://api.qa.saia.ai/chat

Perfect. We also have to test it via API, and for that we are going to use the chat API. The cURL sample is the one currently shown and that we have already used before.

We are going to define a POST, and the URL will be as follows: https://api.qa.saia.ai/chat

Testing the assistant: Postman API Platform



https://api.qa.saia.ai/chat

POST https://api.qa.saia.ai/chat

Params | Authorization | Headers (9) | Body | Pre-request Script | Tests | Settings

none · form-data · x-www-form-urlencoded · [✓ raw] · binary · GraphQL · JSON

```
    "model": "saia:assistant:FrenchAssistant",
    "messages": [
        {
            "role": "user",
            "content": "GeneXus Enterprise AI is a business platform designed to facilitate the implementation of artificial intelligence
              assistants. These assistants can integrate and interact with current operations."
        }
    ],

```

Response

Body | Cookies (3) | Headers (11) | Test Results — Status: 200 OK Time: 2.33 s Size: 1.42 KB

```
{
    "id": "chatcmpl-9EIK0Cc9BJu6MhbXMiBnYSnyAlpTR",
    "object": "chat.completion",
    "created": 1713194056,
    "model": "gpt-3.5-turbo-16k-0613",
    "choices": [
        {
            "index": 0,
            "message": {
                "role": "assistant",
                "content": "GeneXus Enterprise AI est une plateforme d'entreprise conçue pour faciliter la mise en œuvre d'assistants
                    d'intelligence artificielle. Ces assistants peuvent s'intégrer et interagir avec les opérations actuelles."
            },
            "logprobs": null,
            "finish_reason": "stop"
        }
    ],
    "usage": {
        "prompt_tokens": 107,
```

We go back to Postman and define the POST.

The authorization is Bearer and we need a Project Api Token. Therefore, we go to Platform, Api Tokens, copy the Default, and paste it.

We go to Body, Raw, JSON and define the body of the request.

Remember that “Model” corresponds to the type of assistant followed by its name.

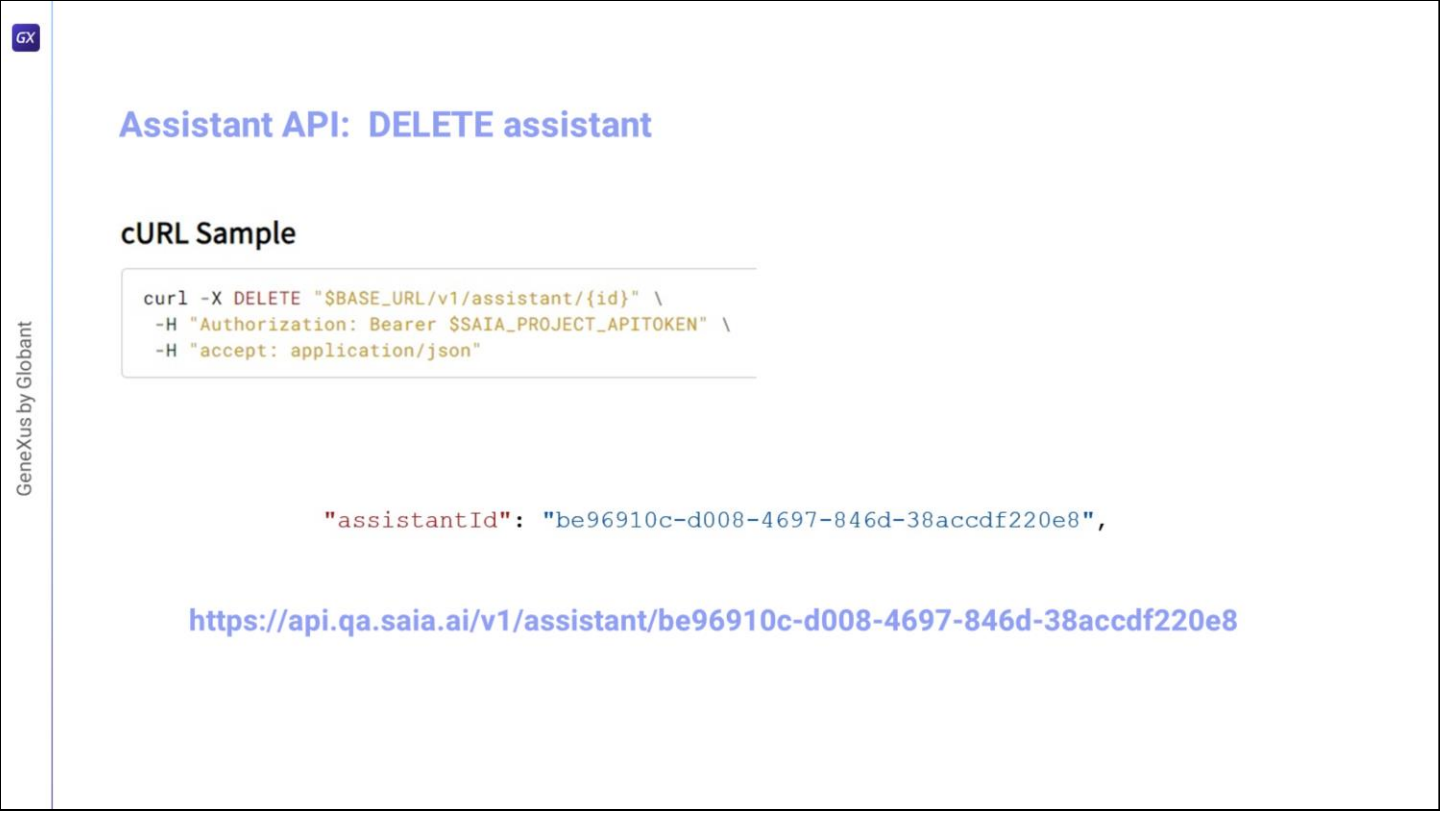
So we indicate the type “assistant” and the name “MyFrenchAssistant.”

Also, remember that Messages defines a message to be added, where “content” corresponds to the user's input.

We type as input that “GeneXus Enterprise AI is a business platform designed to facilitate the implementation of artificial intelligence assistants. These assistants can be integrated and interact with current operations.”

If necessary, we could indicate other parameters, such as determining the revision of the assistant to be used. In this case, we have only one and it is active by default, so it is not necessary to indicate it.

We select Send, and see the answer.



GeneXus by Globant

Assistant API: DELETE assistant

cURL Sample

```
curl -X DELETE "$BASE_URL/v1/assistant/{id}" \
 -H "Authorization: Bearer $SAIA_PROJECT_APITOKEN" \
 -H "accept: application/json"
```

```
"assistantId": "be96910c-d008-4697-846d-38accdf220e8",
```

https://api.qa.saia.ai/v1/assistant/be96910c-d008-4697-846d-38accdf220e8

To finish, we are going to delete the assistant that we have created and tested through all the options.

For that, we are going to use the Delete method, which requires the AssistantId as parameter.

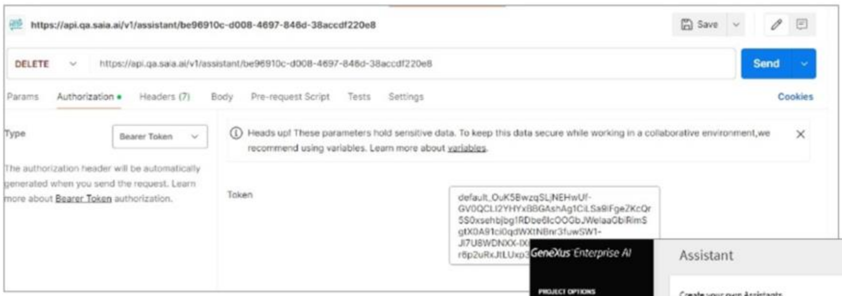
The Id of our assistant that we want to delete is the one displayed.

Therefore, the required URL is as follows:

https://api.qa.saia.ai/v1/assistant/be96910c-d008-4697-846d-38accdf220e8

We also need a Project Api Token for authorization.

Assistant API: DELETE assistant



https://api.qa.saia.ai/v1/assistant/be96910c-d008-4697-846d-38accdf220e8

DELETE https://api.qa.saia.ai/v1/assistant/be96910c-d008-4697-846d-38accdf220e8

Params | Authorization | Headers (7) | Body | Pre-request Script | Tests | Settings | Cookies

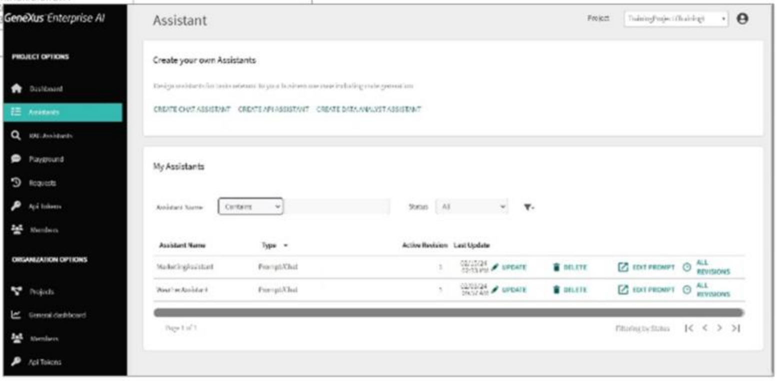
Type: Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about Bearer Token authorization.

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables.

Token:

```
default_OuK5BwzqSLjNEHwUf-
GV0QCLI2YHYxBBGAshAg1CiLSa9lFgeZKcQr
5S0xsehbjbg1RDbe6lcOOGbJWelaaGblRimS
gtX0A91ci0qdWXtNBnr3fuwSW1-
Jl7U8WDNXX-IX
r6p2uRxJtLUxp3
```



GeneXus Enterprise AI — Assistant

Create your own Assistants

CREATE CHAT ASSISTANT CREATE API ASSISTANT CREATE DATA ANALYST ASSISTANT

My Assistants

Assistant Name	Type	Active Revision	Last Update
MarketingAssistant	Prompt/Chat	1	[illegible]
[illegible]	Prompt/Chat	1	[illegible]

We go to Postman and define the request. We select Send.

There are no errors, and if we go back to the platform, we can confirm that the assistant was deleted.