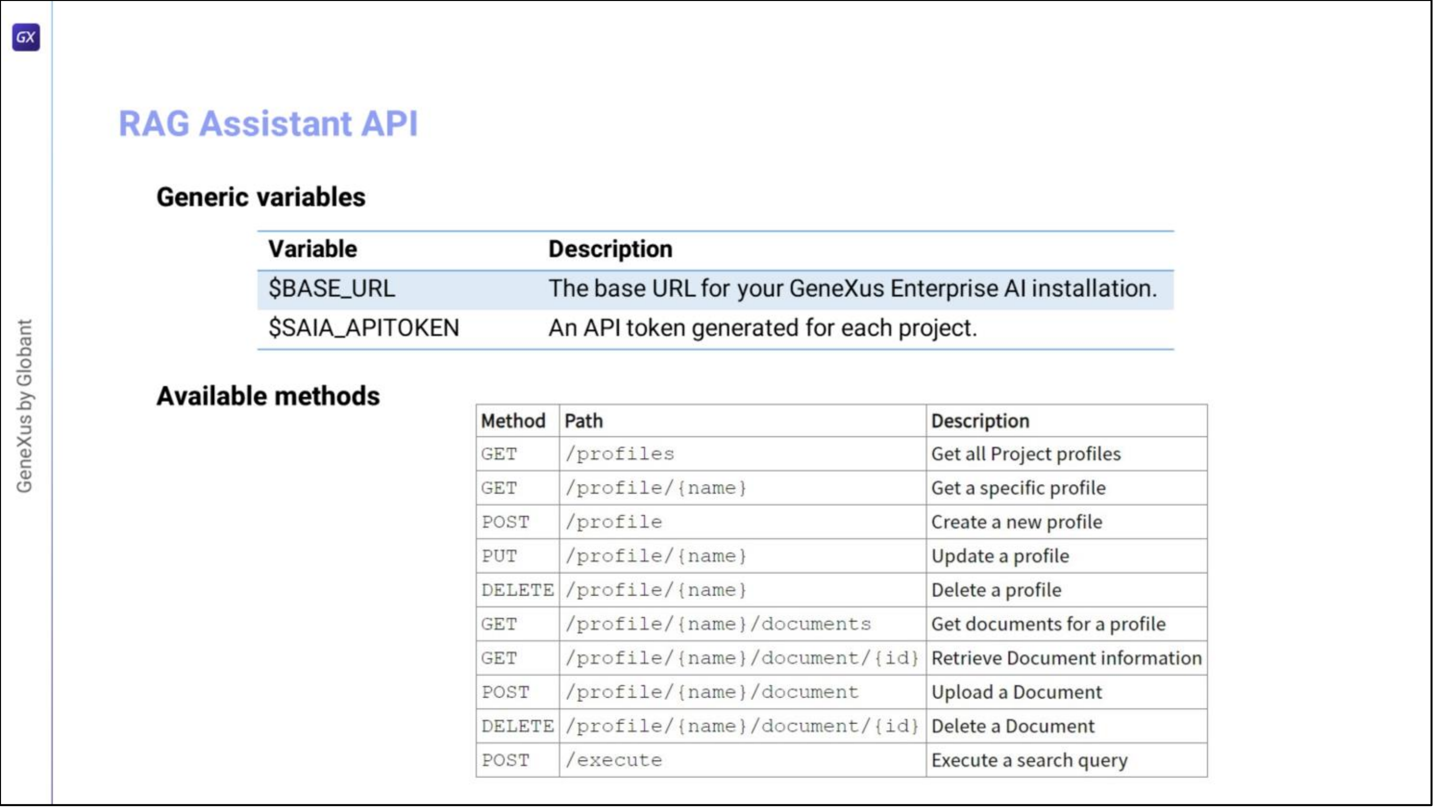
RAG Assistant API and Chat with Documents API

Alejandra Caggiano

As we know, GeneXus Enterprise AI provides a set of several APIs to interact with the defined assistants.

In this context, we are now going to learn about the API to interact with RAG Assistants.



GX

GeneXus by Globant

RAG 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	/profiles	Get all Project profiles
GET	/profile/{name}	Get a specific profile
POST	/profile	Create a new profile
PUT	/profile/{name}	Update a profile
DELETE	/profile/{name}	Delete a profile
GET	/profile/{name}/documents	Get documents for a profile
GET	/profile/{name}/document/{id}	Retrieve Document information
POST	/profile/{name}/document	Upload a Document
DELETE	/profile/{name}/document/{id}	Delete a Document
POST	/execute	Execute a search query

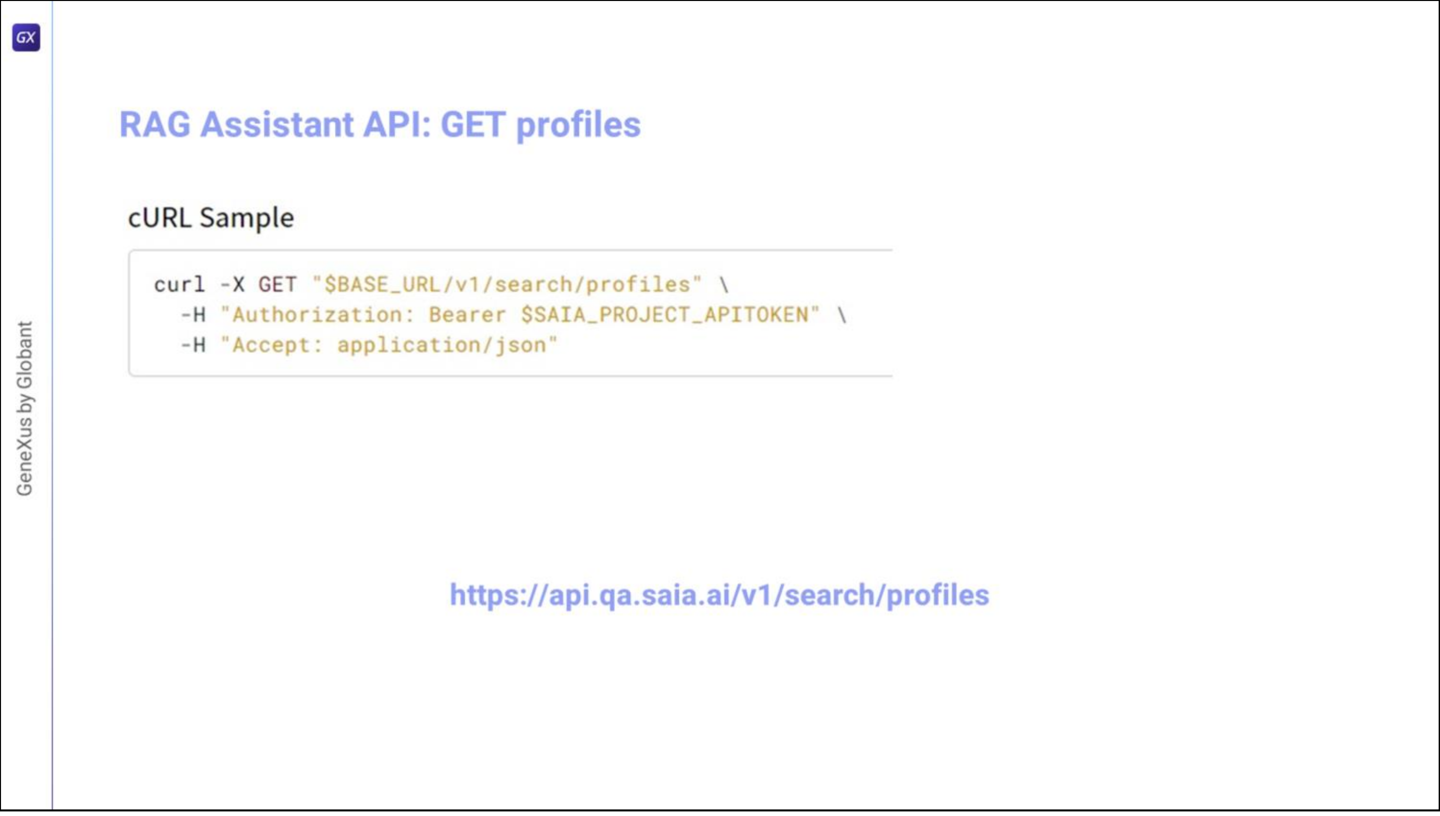
To use this API, we must consider the generic variables we already know:

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:

First, as an example, we will test the GET method that returns the list of RAG assistants defined in a project.



GeneXus by Globant

RAG Assistant API: GET profiles

cURL Sample

```
curl -X GET "$BASE_URL/v1/search/profiles" \
  -H "Authorization: Bearer $SAIA_PROJECT_APITOKEN" \
  -H "Accept: application/json"
```

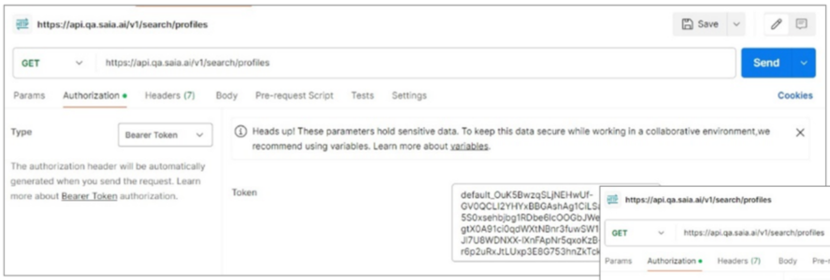
https://api.qa.saia.ai/v1/search/profiles

The corresponding cURL sample indicates that the URL must have the following format:

Therefore, in our context the URL will be as follows:
https://api.qa.saia.ai/v1/search/profiles

We can see that we need a Project Api Token.

Postman API Platform



https://api.qa.saia.ai/v1/search/profiles

GET https://api.qa.saia.ai/v1/search/profiles

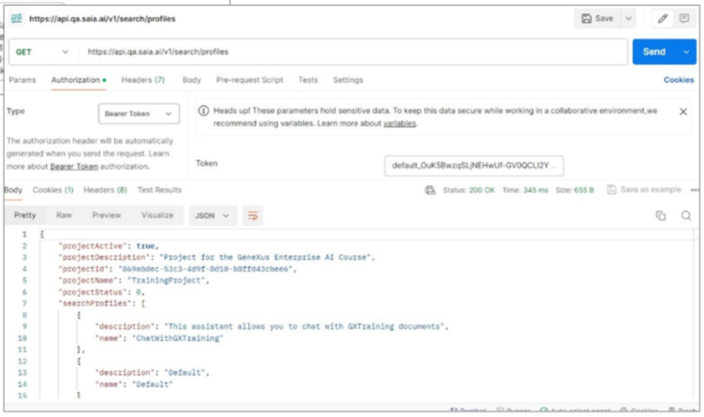
Params | Authorization ● | Headers (7) | Body | Pre-request Script | Tests | Settings | Cookies

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-GV0QCLI2YHYxBBGAshAg1CiLS 5S0xsehbjbg1RDbe6lcOOGbJWe gtX0A91ci0qdWXtNBnr3fuwSW1 JI7U8WDNXX-lXnFApNr5qxoKzB- r6p2uRxJtLUxp3E8G753hnZkTck



https://api.qa.saia.ai/v1/search/profiles

GET https://api.qa.saia.ai/v1/search/profiles

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-GV0QCLI2Y ...

Body | Cookies (1) | Headers (8) | Test Results — Status: 200 OK Time: 345 ms Size: 655 B Save as example

Pretty | Raw | Preview | Visualize | JSON

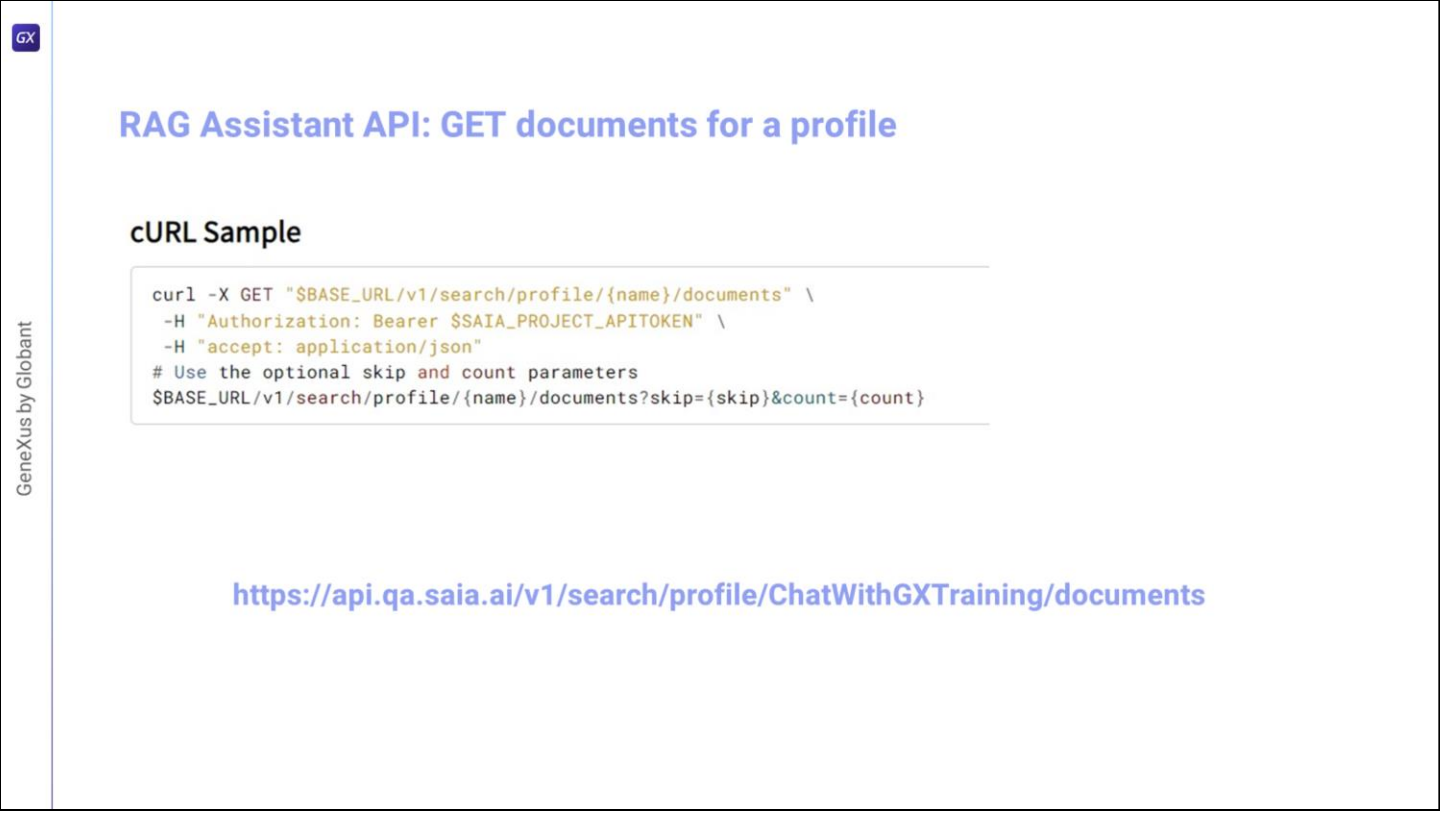
```
{
    "projectActive": true,
    "projectDescription": "Project for the GeneXus Enterprise AI Course",
    "projectId": "869abdec-53c3-4d9f-8d10-b8ffd43cbee6",
    "projectName": "TrainingProject",
    "projectStatus": 0,
    "searchProfiles": [
        {
            "description": "This assistant allows you to chat with GXTraining documents",
            "name": "ChatWithGXTraining"
        },
        {
            "description": "Default",
            "name": "Default"
        }
```

We already know the process, so from Postman, we define the GET.

And from the platform we copy the default API token. We go back to Postman and define the authorization.

We click on Send and get the response.

In our TrainingProject, the RAG Assistant named ChatWithGXTraining is defined.



RAG Assistant API: GET documents for a profile

cURL Sample

```
curl -X GET "$BASE_URL/v1/search/profile/{name}/documents" \
 -H "Authorization: Bearer $SAIA_PROJECT_APITOKEN" \
 -H "accept: application/json"
# Use the optional skip and count parameters
$BASE_URL/v1/search/profile/{name}/documents?skip={skip}&count={count}
```

https://api.qa.saia.ai/v1/search/profile/ChatWithGXTraining/documents

We will now ask for the list of documents that feeds this assistant.

For that, we are going to use the GET method that now requires the name of the assistant as parameter.

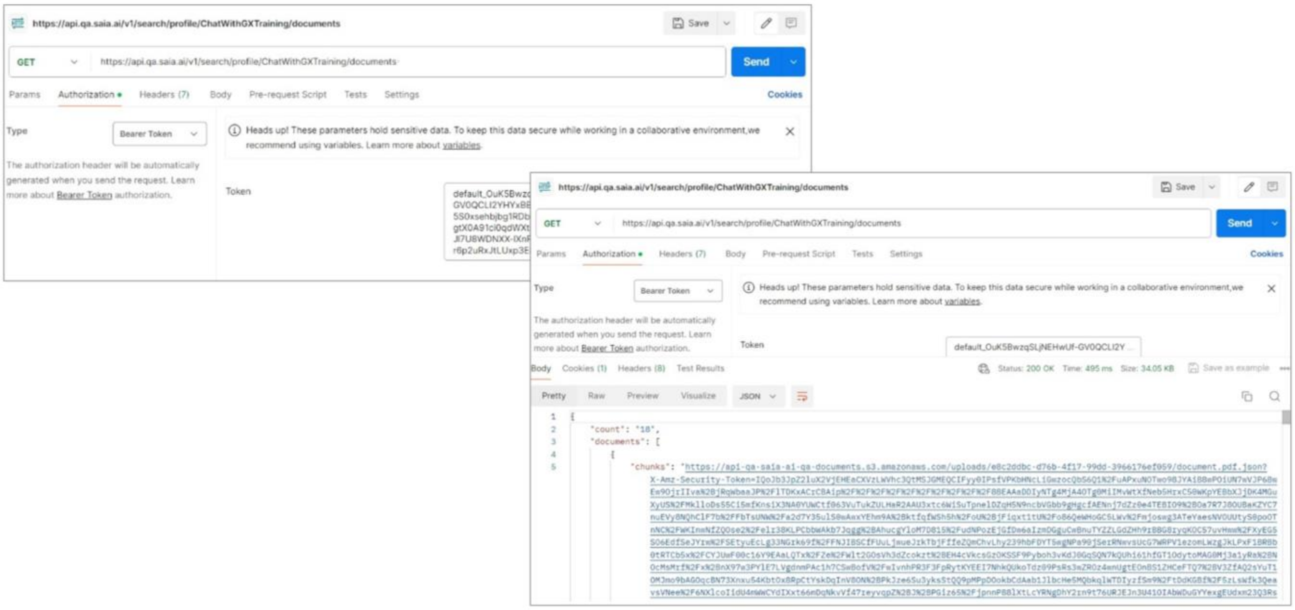
In the sample, we see that the URL has this format:

So our URL will be as follows:

https://api.qa.saia.ai/v1/search/profile/ChatWithGXTraining/documents

Once again, we need a Project Api Token.

Postman API Platform


https://api.qa.saia.ai/v1/search/profile/ChatWithGXTraining/documents

GET https://api.qa.saia.ai/v1/search/profile/ChatWithGXTraining/documents

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.

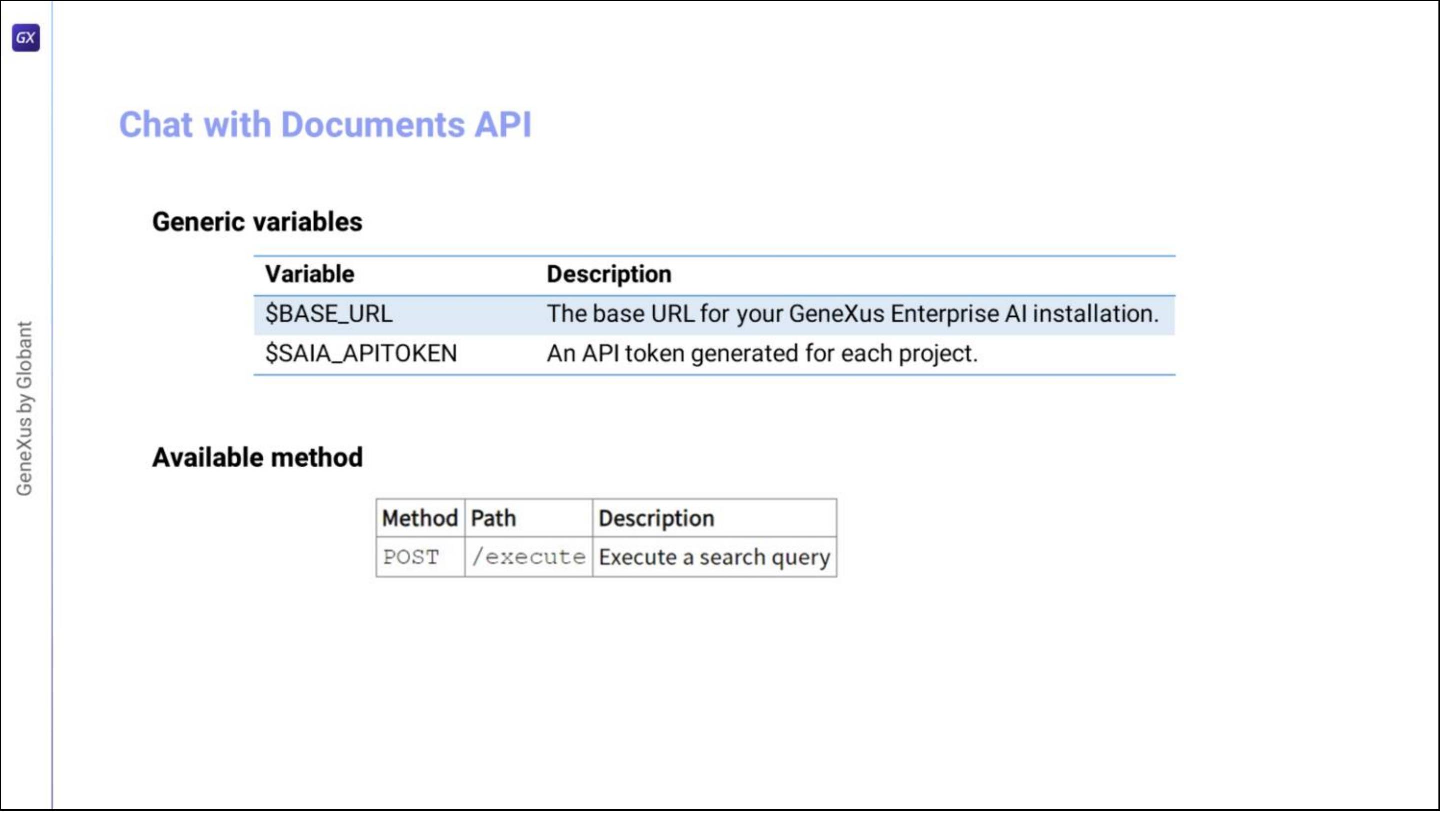
Token: default_OuK5BwzqSLjNEHwUf-GV0QCLI2Y ...

Status: 200 OK Time: 495 ms Size: 34.05 KB

```
{
    "count": "18",
    "documents": [
        {
            "chunks": "https://api-qa-saia-ai-qa-documents.s3.amazonaws.com/uploads/e8c2ddbc-d76b-4f17-99dd-3966176ef059/document.pdf.json?
                X-Amz-Security-Token=IQoJb3JpZ2luX2VjEHEaCXVzLWVhc3QtMSJGMEQCIFyy0IPsfVPKbHNcL1GwzocQbS6Q1%2FuAPxuNOTwo9BJYA1B8ePO1UN7wVJP68e
                Em9OjrIIva%2BjRqWbaaJP%2FlTDKxACrCBAip%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F8BEAAaDDIyNTg4MjA4OTg0MiIMvWtXfNeb5HrxC50WKpYEBbXJjDK4MGu
                XyUS%2FMklloDs55Ci5mfKnsiX3NA0YUWCtf063VuTukZULHaR2AAU3xtc6W1SuTpnelDZqH5N9ncbVGbb9gHgcfAENnj7dZz0e4TEBIO9%2BOa7R7J8OUBaKZYC7
                nuEVy8NQhClF7b%2FFbTsUNW%2Fa2d7Y35ulS0wAaxYEhm9A%2BktfqfWSh5h%2FoUN2BjFiqxt1tU%2Fo86QeWHoGCSLWv%2FmjoswgЗATeYaesNVOUUtyS8poOT
                nNC%2FWKInmNfZQOse2%2Felr38KLPCbbWAkb7Jqgg%2BAhucgYloM7D815%2FudNPozEjGfDm6aIzmDGguCwBnuTYZZLGdZHh9rBBG8ryqKOC57uvHmw%2FXyEG5
                SO6EdfSeJYrw%2FSEtyuEcLg33NGrk69f%2FFNJIBSCfFUuLjmueJrkTbjFffeZQmChvLhy239hbFDYT5wgNPa90jSerRNwvsUcG7WRPV1ezomLwzgJkLPxF1BRBb
                0tRTCb5x%2FCYJUwF00c16Y9EAaLQTx%2FZe%2FWlt2GOsVh3dZcokzt%2BEH4cVkcsGzOKSSF9Pyboh3vKdJ0GqSQN7kQUhi61hfGT1OdytoMAG0MjJa1yRa%2BN
                OcMsMrf%2Fx%2BnX97a3PYlE7LVgdnmPAc1h7CSwBofV%2FwIvnhPR3F3FpRytKYEEI7NhkQUkoTdz89PsRs3mZROz4mnUgtEOnBS1ZHCeFTQ7%2BV3ZfAQ2sYuT1
                OMJmo9bAGOqcBN73Xnxu54Kbt0x8RpCtYskDqInV8ON%2BPkJze6Su3yksStQQ9pMPpDOokbCdAab1JlbcHe5MQbkqlWTDIyzfSm9%2FtDdKGBf%2F5zLsWfk3Qea
                vsVNee%2F6NXlcoIIdU4mWWCYdIXxt66mDqNkvVf47reyvqpZ%2BJ%2BPGiz65%2FjpnnP8BlXtLcYRNgDhY2rn9t76URJEJn3U41OIAbWDuGYYexgEUdxm23Q3Rs
```

We go to Postman and define the request. We click on Send and see the answer.

It shows that there are 18 documents, and the URL to access each one of them.



GeneXus by Globant

Chat with Documents 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.

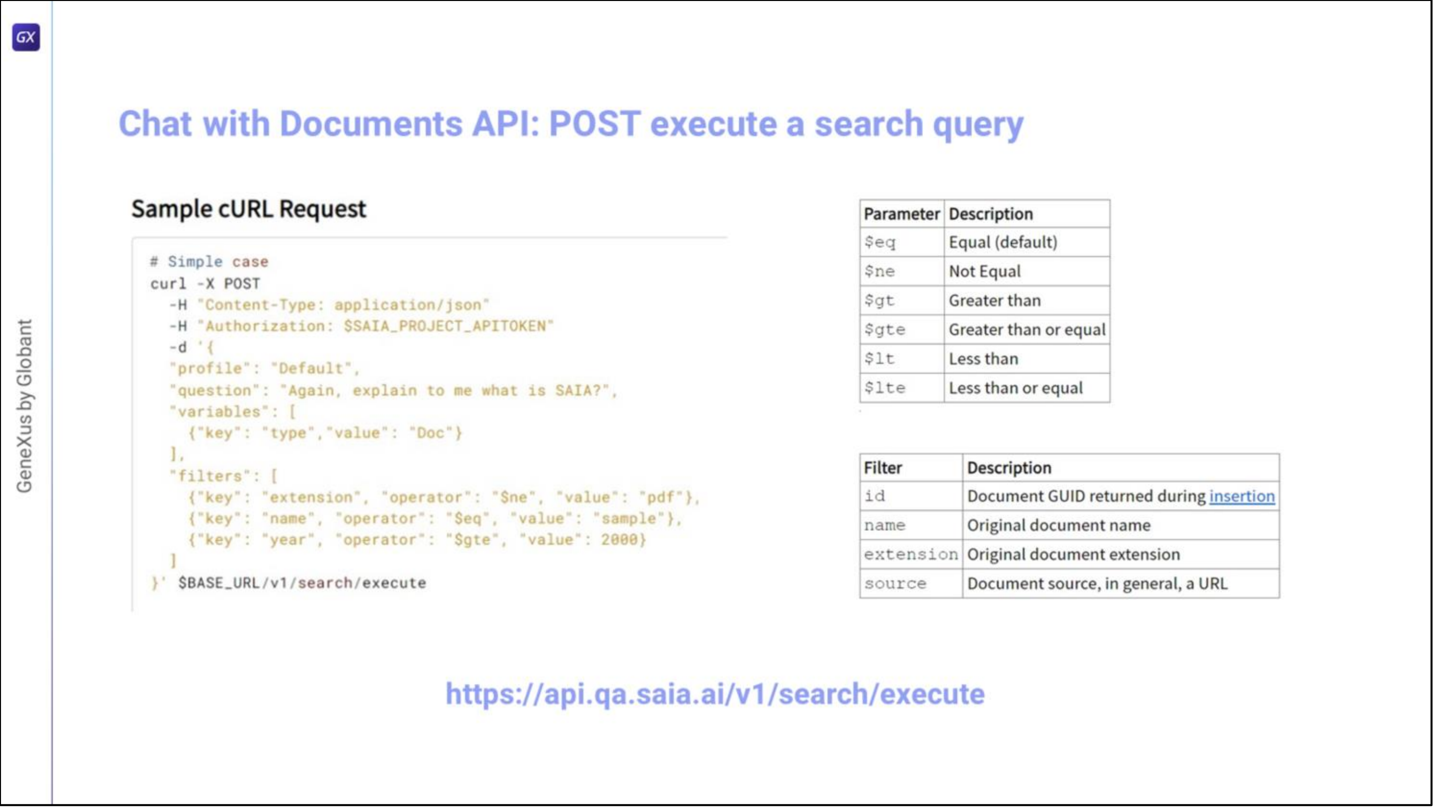
Available method

Method	Path	Description
`POST`	`/execute`	Execute a search query

Good. We are now going to make a simple query to this assistant.

For that, we are going to use the API to chat with documents. This API allows making searches or queries on the indexed content.

Its method is POST:



GX

GeneXus by Globant

Chat with Documents API: POST execute a search query

Sample cURL Request

```
# Simple case
curl -X POST
  -H "Content-Type: application/json"
  -H "Authorization: $SAIA_PROJECT_APITOKEN"
  -d '{
  "profile": "Default",
  "question": "Again, explain to me what is SAIA?",
  "variables": [
    {"key": "type","value": "Doc"}
  ],
  "filters": [
    {"key": "extension", "operator": "$ne", "value": "pdf"},
    {"key": "name", "operator": "$eq", "value": "sample"},
    {"key": "year", "operator": "$gte", "value": 2000}
  ]
}' $BASE_URL/v1/search/execute
```

Parameter	Description
`$eq`	Equal (default)
`$ne`	Not Equal
`$gt`	Greater than
`$gte`	Greater than or equal
`$lt`	Less than
`$lte`	Less than or equal

Filter	Description
`id`	Document GUID returned during insertion
`name`	Original document name
`extension`	Original document extension
`source`	Document source, in general, a URL

https://api.qa.saia.ai/v1/search/execute

If we query the cURL sample, we see that the URL will be as follows: https://api.qa.saia.ai/v1/search/execute

We also need a Project Api Token.

So again in Postman we define the POST and the corresponding authorization.

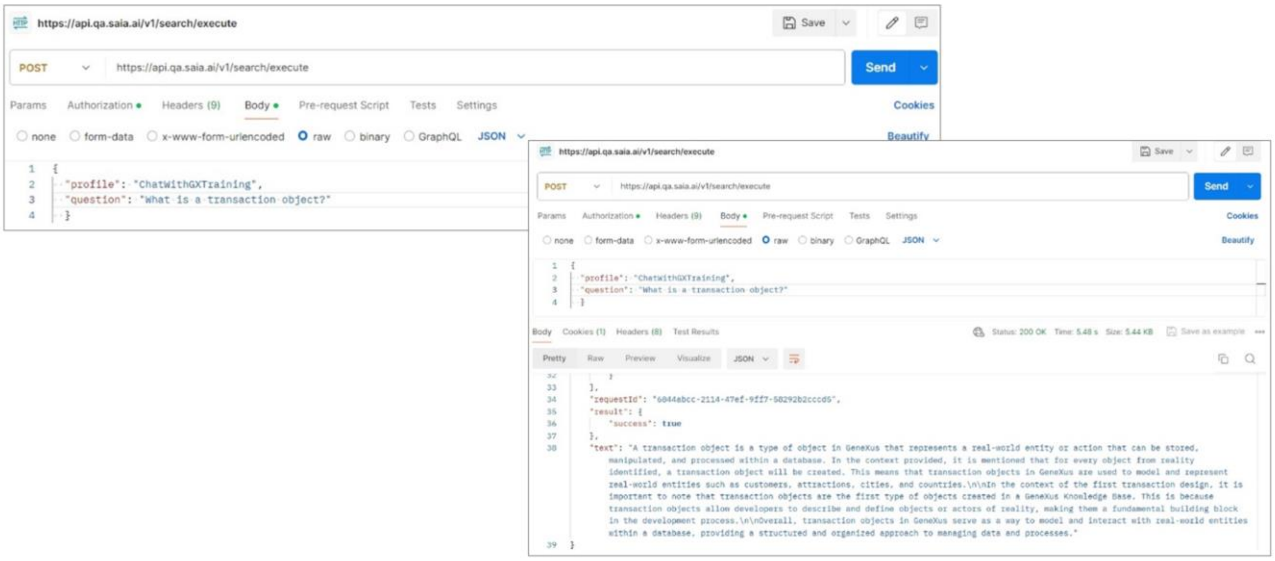
Let's see now the body of the query.

We must indicate the profile – that is, the assistant we are going to query – and declare the question.

Optionally, we can add parameters and filters to the question, as needed.

GeneXus by Globant

Postman API Platform


https://api.qa.saia.ai/v1/search/execute

POST https://api.qa.saia.ai/v1/search/execute

Body: raw, JSON

```
{
  "profile": "ChatWithGXTraining",
  "question": "What is a transaction object?"
}
```

Status: 200 OK Time: 5.48 s Size: 5.44 KB

```
        }
    ],
    "requestId": "6044abcc-2114-47ef-9ff7-58292b2cccd5",
    "result": {
        "success": true
    },
    "text": "A transaction object is a type of object in GeneXus that represents a real-world entity or action that can be stored,
        manipulated, and processed within a database. In the context provided, it is mentioned that for every object from reality
        identified, a transaction object will be created. This means that transaction objects in GeneXus are used to model and represent
        real-world entities such as customers, attractions, cities, and countries.\n\nIn the context of the first transaction design, it is
        important to note that transaction objects are the first type of objects created in a GeneXus Knowledge Base. This is because
        transaction objects allow developers to describe and define objects or actors of reality, making them a fundamental building block
        in the development process.\n\nOverall, transaction objects in GeneXus serve as a way to model and interact with real-world entities
        within a database, providing a structured and organized approach to managing data and processes."
}
```

Good. We then go to the Body, Raw, JSON tab and declare the question.

Let's define a simple query.

We indicate that we are going to query our RAG Assistant ChatWithGXTraining, and the question is “What is a Transaction object?”

To see the answer, we click on Send. We see the list of queried documents, and the final text of the answer.

GeneXus by Globant

GeneXus™ by Globant

training.genexus.com