Data retrieval with GPT Actions

Copy page

Retrieve data using APIs and databases with GPT Actions.

One of the most common tasks an action in a GPT can perform is data retrieval. An action might:

1. Access an API to retrieve data based on a keyword search
2. Access a relational database to retrieve records based on a structured query
3. Access a vector database to retrieve text chunks based on semantic search

We'll explore considerations specific to the various types of retrieval integrations in this guide.

Data retrieval using APIs

Many organizations rely on 3rd party software to store important data. Think Salesforce for customer data, Zendesk for support data, Confluence for internal process data, and Google Drive for business documents. These providers often provide REST APIs which enable external systems to search for and retrieve information.

When building an action to integrate with a provider's REST API, start by reviewing the existing documentation. You'll need to confirm a few things:

1. Retrieval methods

   Search - Each provider will support different search semantics, but generally you want a method which takes a keyword or query string and returns a list of matching documents. See Google Drive's `file.list` method for an example.

   Get - Once you've found matching documents, you need a way to retrieve them. See Google Drive's `file.get` method for an example.

2. Authentication scheme

   For example, Google Drive uses OAuth to authenticate users and ensure that only their available files are available for retrieval.

3. OpenAPI spec

   Some providers will provide an OpenAPI spec document which you can import directly into your action. See Zendesk, for an example.

   > You may want to remove references to methods your GPT *won't* access, which constrains the actions your GPT can perform.

   For providers who *don't* provide an OpenAPI spec document, you can create your own using the ActionsGPT (a GPT developed by OpenAI).

Your goal is to get the GPT to use the action to search for and retrieve documents containing context which are relevant to the user's prompt. Your GPT follows your instructions to use the provided search and get methods to achieve this goal.

Data retrieval using Relational Databases

Organizations use relational databases to store a variety of records pertaining to their business. These records can contain useful context that will help improve your GPT's responses. For example, let's say you are building a GPT to help users understand the status of an insurance claim. If the GPT can look up claims in a relational database based on a claims number, the GPT will be much more useful to the user.

When building an action to integrate with a relational database, there are a few things to keep in mind:

1. Availability of REST APIs

   Many relational databases do not natively expose a REST API for processing queries. In that case, you may need to build or buy middleware which can sit between your GPT and the database.

   This middleware should do the following:

   - Accept a formal query string
   - Pass the query string to the database
   - Respond back to the requester with the returned records

2. Accessibility from the public internet

   Unlike APIs which are designed to be accessed from the public internet, relational databases are traditionally designed to be used within an organization's application infrastructure. Because GPTs are hosted on OpenAI's infrastructure, you'll need to make sure that any APIs you expose are accessible outside of your firewall.

3. Complex query strings

   Relational databases uses formal query syntax like SQL to retrieve relevant records. This means that you need to provide additional instructions to the GPT indicating which query syntax is supported. The good news is that GPTs are usually very good at generating formal queries based on user input.

4. Database permissions

   Although databases support user-level permissions, it is likely that your end users won't have permission to access the database directly. If you opt to use a service account to provide access, consider giving the service account read-only permissions. This can avoid inadvertently overwriting or deleting existing data.

Your goal is to get the GPT to write a formal query related to the user's prompt, submit the query via the action, and then use the returned records to augment the response.

Data retrieval using Vector Databases

If you want to equip your GPT with the most relevant search results, you might consider integrating your GPT with a vector database which supports semantic search as described above. There are many managed and self hosted solutions available on the market, see here for a partial list.

When building an action to integrate with a vector database, there are a few things to keep in mind:

1. Availability of REST APIs

   Many relational databases do not natively expose a REST API for processing queries. In that case, you may need to build or buy middleware which can sit between your GPT and the database (more on middleware below).

2. Accessibility from the public internet

   Unlike APIs which are designed to be accessed from the public internet, relational databases are traditionally designed to be used within an organization's application infrastructure. Because GPTs are hosted on OpenAI's infrastructure, you'll need to make sure that any APIs you expose are accessible outside of your firewall.

3. Query embedding

   As discussed above, vector databases typically accept a vector embedding (as opposed to plain text) as query input. This means that you need to use an embedding API to convert the query input into a vector embedding before you can submit it to the vector database. This conversion is best handled in the REST API gateway, so that the GPT can submit a plaintext query string.

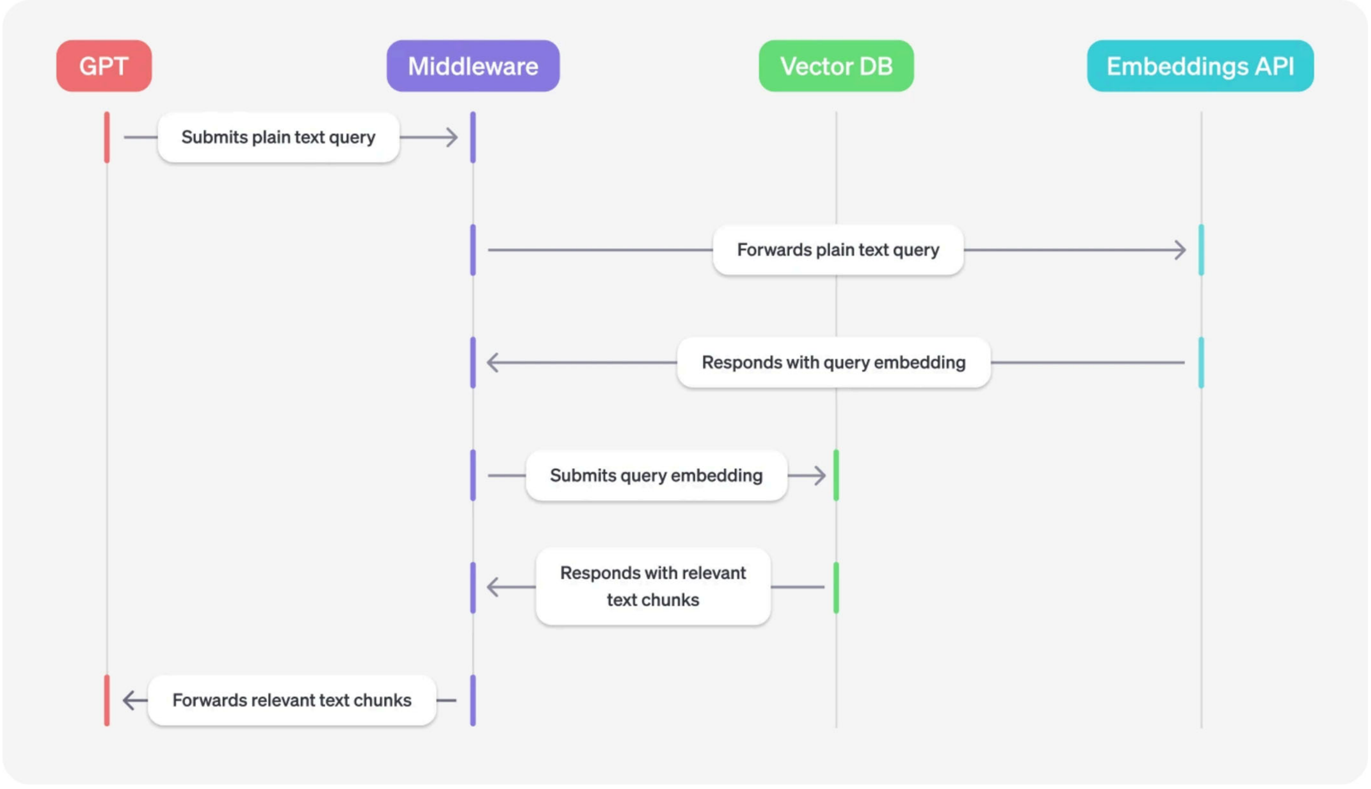
4. Database permissions

   Because vector databases store text chunks as opposed to full documents, it can be difficult to maintain user permissions which might have existed on the original source documents. Remember that any user who can access your GPT will have access to all of the text chunks in the database and plan accordingly.

Middleware for vector databases

As described above, middleware for vector databases typically needs to do two things:

1. Expose access to the vector database via a REST API
2. Convert plaintext query strings into vector embeddings



GPT
Middleware
Vector DB
Embeddings API
Submits plain text query
Forwards plain text query
Responds with query embedding
Submits query embedding
Responds with relevant text chunks
Forwards relevant text chunks

The goal is to get your GPT to submit a relevant query to a vector database to trigger a semantic search, and then use the returned text chunks to augment the response.