**Playbooks:**

While LLMs are powerful, they are general-purpose. To make a generative agent perform specific tasks reliably and consistently within a business context, the LLM needs structured guidance and operational instructions. This is precisely where the playbook comes in.

Think of a playbook as a structured instruction manual or a blueprint that guides the LLM on how to achieve a particular goal or handle a specific conversational scenario. It provides the necessary context, constraints, and operational details that transform a general-purpose LLM into a specialized, task-oriented conversational agent.

In essence, while the LLM provides the brain (the ability to understand and generate), the playbook provides the **operational intelligence, structure, and specific knowledge** needed for that brain to function effectively and reliably as a dedicated conversational agent in Dialogflow CX. It bridges the gap between the LLM's broad capabilities and the precise requirements of a conversational application. Without playbooks, a generative agent would lack the necessary guidance to consistently perform specific tasks, adhere to business logic, or interact effectively with external systems.

## Playbook data

A playbook is composed of the following data:

* **Playbook name**: a concise name in natural language that helps developers and the LLM to understand what tasks the playbook handles
* [Goals](https://cloud.google.com/dialogflow/cx/docs/concept/playbook/goal): high level description of what the playbook should accomplish
* [Instructions](https://cloud.google.com/dialogflow/cx/docs/concept/playbook/instruction): defines the process steps that should be taken to accomplish the goal
* [Examples](https://cloud.google.com/dialogflow/cx/docs/concept/playbook/example): sample conversations that are effectively few-shot prompt examples for the LLM
* [Parameters](https://cloud.google.com/dialogflow/cx/docs/concept/playbook/parameter): are used to store information about a conversation like user input, user system information, results of actions, and so on.

Playbook Types:

Two types - Task playbook and Routine playbook

### Task playbooks

*Task playbooks* are the original type of playbook. They are used to break down complex tasks into smaller, reusable sub-tasks. They are used for modeling *compositional* conversation stages, where each stage communicates through input and output parameters.

Any routine or task playbook can call another task playbook, but a task playbook cannot call another routine playbook.

### Routine playbooks

*Routine playbooks* are a new type of playbook. They are used for modeling *sequential* conversation stages, where each stage is complete and independent. They can call task playbooks to decompose larger tasks into smaller sub-tasks, and they can transition to other routine playbooks or flows.

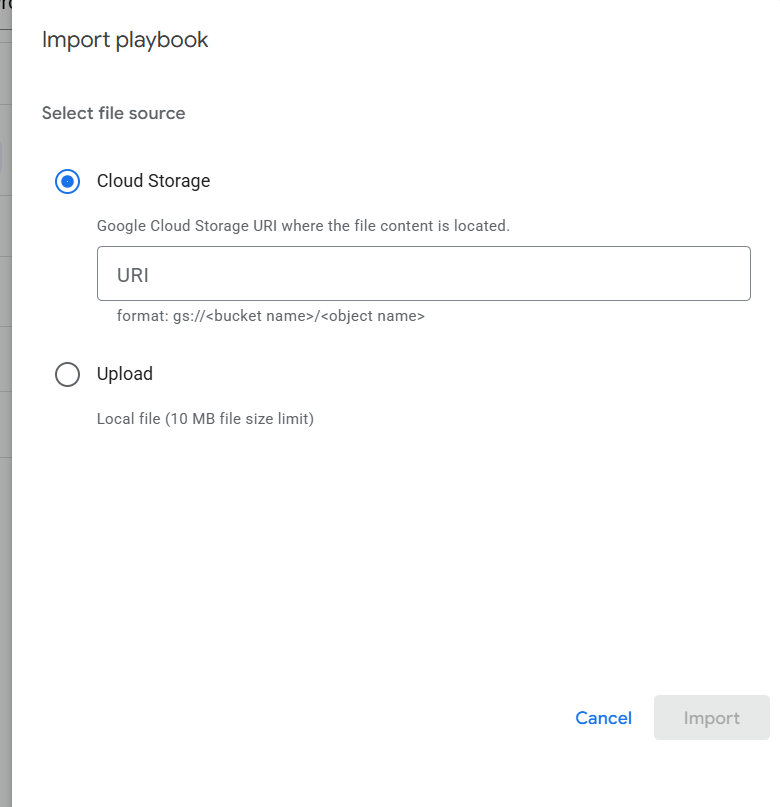
Import and export playbooks

Once you have created playbooks using the Conversational Agents console, you can export these playbooks for use in another agent. To export a playbook:

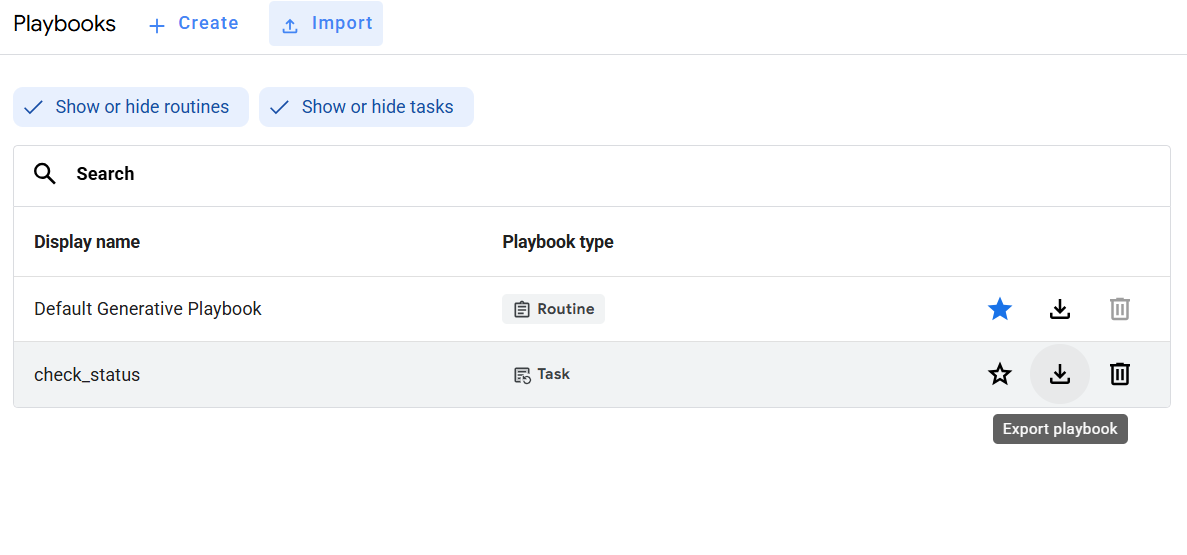
1. Navigate to the playbooks list.
2. Click the export button for the playbook you want to export.
3. Select exporting options.
4. Click **Export**.

To import a playbook you have previously exported:

1. Navigate to the playbooks list.
2. Click **Import**.
3. Select import options.
4. Click **Import**.



Export

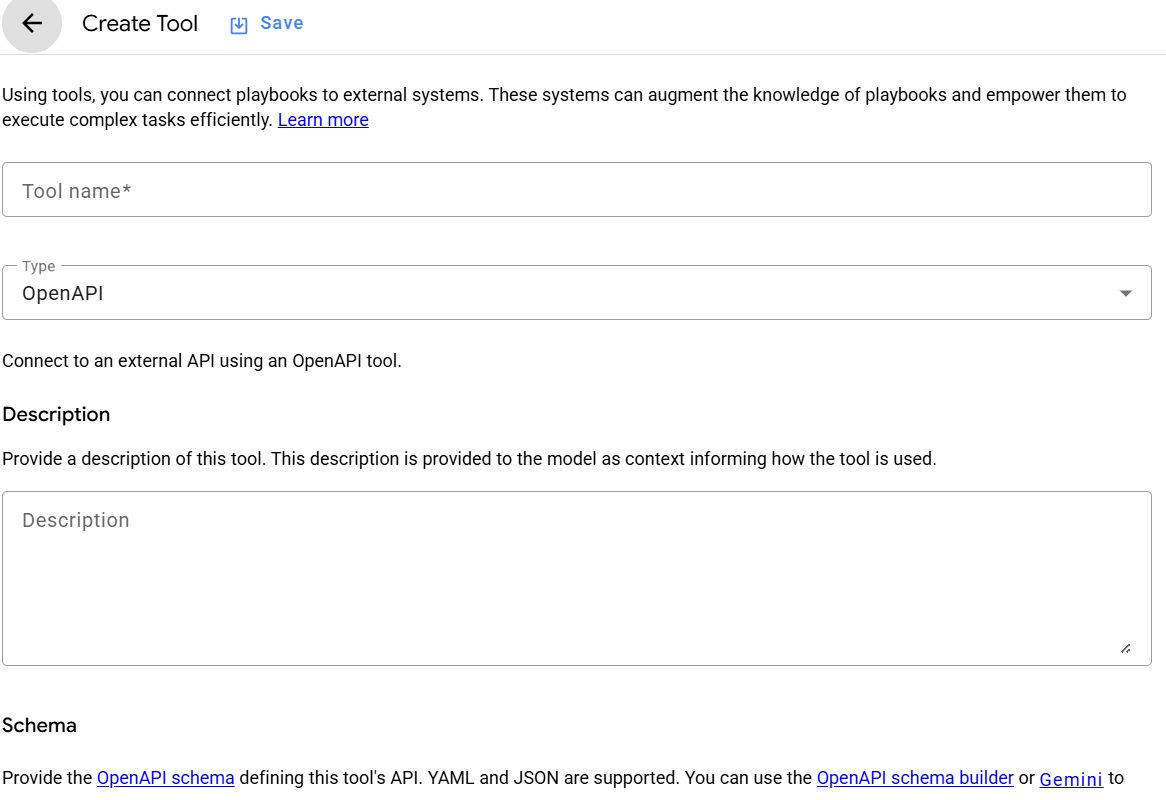


Playbook tools

Using tools, you can connect playbooks to external systems. These systems can augment the knowledge of playbooks and empower them to execute complex tasks efficiently.

OpenAPI tools

An agent can connect to an external API using an OpenAPI tool by providing the [OpenAPI](https://www.openapis.org/) schema. By default, the agent will call the API on your behalf.

w

### OpenAPI tool limitations

The following limitations apply:

* Supported parameter types are path, query, header. The cookie parameter type is not supported yet.
* Parameters defined by OpenAPI schema support the following data types: string, number, integer, boolean, array. The object type is not supported yet.
* You currently can't specify query parameters in the console example editor.
* Request and response body must be empty or JSON.

# Data store tools

Data store tools can provide AI-generated agent responses based on website content and uploaded data. Your agent can find answers to end-users' questions from your [data stores](https://cloud.google.com/dialogflow/vertex/docs/concept/data-store) during [fulfillment](https://cloud.google.com/dialogflow/cx/docs/concept/fulfillment).

## Data store tools: API

You can set up one data store of each type per tool, and the tool will query each of these data stores for answers. By default, the agent will call the [dataStoreTool](https://cloud.google.com/dialogflow/cx/docs/reference/rpc/google.cloud.dialogflow.cx.v3beta1#datastoretool) on your behalf.

There are three data store types:

* PUBLIC\_WEB: A data store that contains public web content.
* UNSTRUCTURED: A data store that contains unstructured private data.
* STRUCTURED: A data store that contains structured data (for example FAQ).