Informatica® Cloud Application Integration
July 2024

GeminiAI Prompt Chaining

Informatica Cloud Application Integration GeminiAI Prompt Chaining
July 2024

See patents at https://www.informatica.com/legal/patents.html.

Publication Date: 2024-07-30

Table of Contents

Preface

Use Gemini AI Prompt Chaining to design prompt chains and resolve them in sequence so that it provides the desired responses from a large language model (LLM). The recipe is based on Guide or HTTP request to call the process.

Introduction to GeminiAI Prompt Chaining recipe

The GeminiAI Prompt Chaining recipe is initiated using a guide or an HTTP request.
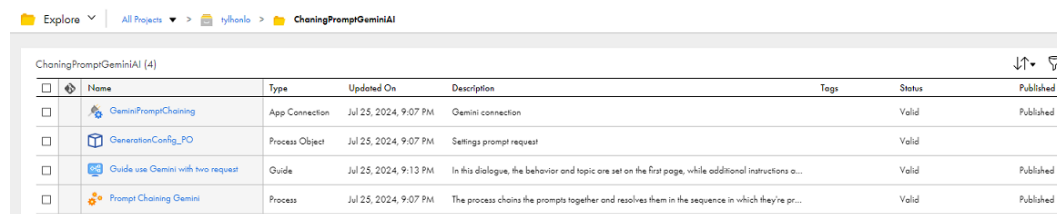
The process chains the prompts together and resolves them in the sequence in which they're provided. Chaining of these prompts augments the ability of the language model being used to deliver a highly curated response. When you send a query, the process first sets the behavior and topic, and then provides the additional instructions to answer the query. After receiving the response, you can ask a new question without changing the topic.

CHAPTER 2

GeminiAI Prompt Chaining recipe contents

The GeminiAI Prompt Chaining recipe contains multiple assets, such as a process object, an app connection, a guide, and a process.

The following image shows the assets that the GeminiAI Prompt Chaining recipe package contains:

GeminiAI Prompt Chaining recipe assets

The following table lists the assets that the GeminiAI Prompt Chaining recipe package contains:

Asset Name	Asset Type	Description
GeminiPromptChaining	App Connection	Gemini connection
GenerationConfig_PO	Process Object	Settings for the prompt request
Guide to use Gemini with two requests	Guide	When you send a query, the process first sets the behavior and topic, and then provides the additional instructions to answer the query. After receiving the response, you can ask a new question without changing the topic.
Prompt Chaining Gemini	Process	The process chains the prompts together and resolves them in the sequence in which they're provided. Chaining of these prompts augments the ability of the language model being used to deliver a highly curated response.

C H A P T E R 3

Using the GeminiAI Prompt Chaining recipe

To use the GeminiAI Prompt Chaining recipe, you must perform the following steps manually:

Step 1: Copy and access the recipe

Step 2: Configure and publish the GeminiPromptChaining connection

Step 3: Configure and publish the process

Step 4: Publish and run the guide

Copying and accessing the recipe
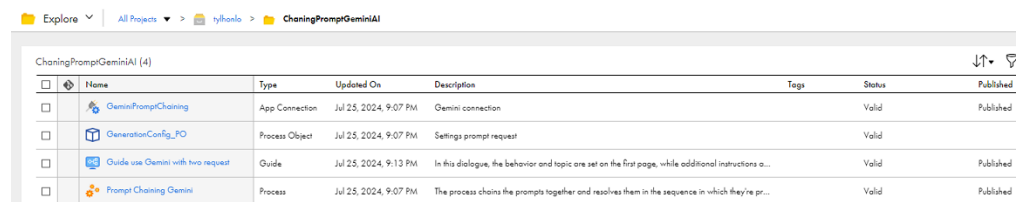
To copy and access the recipe content, perform the following steps:

1. Open the **GeminiAI Prompt Chaining** recipe and click **Use**.
2. Select the location where you want to copy the recipe, and then click **Continue**.
3. In the **Copying the recipe** dialog box, click **OK**.

   It might take some time for the recipe to get copied. You will receive a notification when the recipe is ready for use.
4. After the recipe is copied, click **Explore** to access the recipe content.
5. Navigate to the project or folder where you copied the recipe or enter the recipe name in the **Find** box.

   All the assets in the recipe are displayed as shown in the following image:

Configuring and publishing the GeminiPromptChaining connection

To configure and publish the GeminiPromptChaining connection, perform the following steps:

1. Open the **GeminiPromptChaining** connection.
2. In the **Type** field, select **Gemini**.
3. In the **Run On** field, select **Cloud Server or any Secure Agent**.
4. In the **Connection Properties** section, enter the API key in the **API_Key** property. The **API_Key** property authenticates Gemini connection requests.
5. Save, test, and publish the connection.

Configuring and publishing the process

1. Open the **Prompt Chaining Gemini** process.
2. On the **Temp Fields** tab of the Start step, the default **Model_LLM** is **gemini-1.5 pro**. You can optionally edit the model version. For information about changing the model version, see the Gemini documentation.
3. In the **Create Prompt** step, enter the prompt instructions in the **Assignments** field by updating the **Prompt_Configuration** and **Prompt_Request** fields using the Expression Editor, as shown in the following sample code:

```
For Prompt_Configuration:
 <generationConfig>
      <stopSequences>.</stopSequences>
      <candidateCount>1</candidateCount>
      <maxOutputTokens>200</maxOutputTokens>
      <temperature>0.5</temperature>
      <topP>0.5</topP>
      <topK>2</topK>
   </generationConfig>

For Prompt_Request:
<Generate_Content_Request>
   <contents>
      <parts>
         <text>Your behaviour is {$input.First_System_Prompt}, Topic is
{$input.First_User_Prompt}</text>
      </parts>
      <role>user</role>
   </contents>
 <generationConfig>
      <stopSequences>{$temp.Prompt_Configuration[1]/stopSequences}</stopSequences>
      <candidateCount>{$temp.Prompt_Configuration[1]/candidateCount }</
candidateCount>
      <maxOutputTokens>{$temp.Prompt_Configuration[1]/maxOutputTokens }</
maxOutputTokens>
      <temperature>{$temp.Prompt_Configuration[1]/temperature }</temperature>
      <topP>{$temp.Prompt_Configuration[1]/topP }</topP>
      <topK>{$temp.Prompt_Configuration[1]/topK }</topK>
   </generationConfig>
</Generate_Content_Request>
```

For the **Prompt_Configuration** field, enter values for the following properties:

Property	Description
stopSequences	Contains sequences of characters or strings that stop the model's output. It controls where the model must end its response.
candidateCount	Specifies the number of response candidates that the model must generate. For example, if the value is set to 1, the model generates one response. If set to a higher number, the model generates that many alternative responses for the same input.
maxOutputTokens	Defines the maximum number of tokens the model can generate in its response. Setting a limit ensures that the response is concise and fits within the desired length constraints.
temperature	Controls the randomness of the model's output. A lower value makes the output more deterministic, while a higher value increases randomness and creativity. For example, a temperature of 0.5 balances between deterministic and creative outputs.
topP	Determines the cumulative probability threshold for token selection. The model considers the smallest set of tokens whose cumulative probability meets or exceeds topP. For example, if topP is set to 0.1, the model considers only the top 10% most probable tokens at each step.
topK	Limits the number of the highest-probability tokens to consider during response generation. For example, if topK is set to 2, the model considers only the top 2 tokens at each step, controlling output diversity and quality.

After configuring the prompt instructions, the process send the details to the LLM to fetch the required response, and then stores the first response.

4. In the **Create Second Prompt** step, in the **Assignments** field, update the **Prompt_Request** field using the Expression Editor as shown in the following sample code:

```
<Generate_Content_Request>
<contents>
     <parts>
        <text>Your behaviour is : {$input.First_System_Prompt}, Topic is :
{$input.First_User_Prompt}</text>
     </parts>
     <role>user</role>
   </contents>
 <contents>
     <parts>
        <text>{$temp.Prompt_Response}</text>
     </parts>
     <role>model</role>
   </contents>
     <contents>
     <parts>
        <text>Your additional instructions are : {$input.Second_System_Prompt}.
Question is {$input.Second_User_Prompt}</text>
     </parts>
     <role>user</role>
   </contents>
   <generationConfig>
     <stopSequences>{$temp.Prompt_Configuration[1]/stopSequences}</stopSequences>
     <candidateCount>{$temp.Prompt_Configuration[1]/candidateCount }</
candidateCount>
     <maxOutputTokens>{$temp.Prompt_Configuration[1]/maxOutputTokens }</
maxOutputTokens>
     <temperature>{$temp.Prompt_Configuration[1]/temperature }</temperature>
     <topP>{$temp.Prompt_Configuration[1]/topP }</topP>
```

```
        <topK>{$temp.Prompt_Configuration[1]/topK }</topK>
    </generationConfig>
</Generate_Content_Request>
```
The LLM uses both the requests as an instruction to prepare the final response.

5. Save and publish the process.

Publishing and running the guide

To publish and run the guide, perform the following steps:

1. Open the **Guide to use Gemini with two requests** guide.

2. On the **Start** tab of the Start step, ensure that the **Run As** field is set to **Current User**.

3. Save and run the guide.

4. The **Instructions** page appears. Enter the requests in the **Set behavior** and **Select topic** fields. The LLM uses these requests as instructions to prepare for the final response to your query.

5. Click **Continue**.

6. On the next screen, enter any additional instruction for your request and ask your query in the **Ask Gemini** field.

   Note: You can't edit the behavior and topic fields in this page.

7. Click **Continue**.
   The final response appears.

8. Click **New question** to ask another query, or click **End** to finish.

You can also use the inline frame to embed the guide into an HTML document of a third-party application.