Informatica® Cloud Application Integration
July 2024

GeminiAI Chat with History

Informatica Cloud Application Integration GeminiAI Chat with History
July 2024

Publication Date: 2024-07-30

Table of Contents

Preface

Use the GeminiAI Chat with History recipe to maintain chat history in a file and use it as context for the next query you ask and the Large Language Model's (LLM) response.

CHAPTER 1

Introduction to GeminiAI Chat with History recipe

The GeminiAI Chat with History recipe is initiated using an HTTP request.
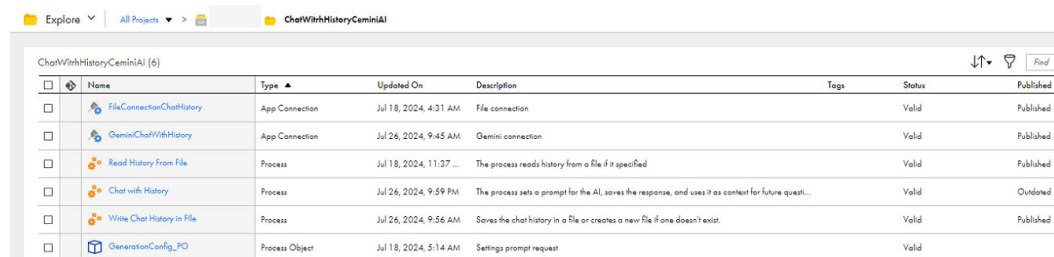
The process maintains chat history in a file and uses it as context for the last user question and LLM answer. It reads input parameters to check if the file exists, retrieves the last user question and LLM answer, prepares a request, and sends it. If the file doesn't exist, it creates one and writes the user input and LLM answer to it.

CHAPTER 2

GeminiAI Chat with History recipe contents

The GeminiAI Chat with History recipe contains multiple assets, such as a process object, two app connections, and three processes.

The following image shows the assets that the GeminiAI Chat with History recipe package contains:

GeminiAI Chat with History recipe assets

The following table lists the assets that the GeminiAI Chat with History recipe package contains:

Asset Name	Asset Type	Description
FileConnectionChatHistory	App Connection	File connection
GeminiChatWithHistory	App Connection	Gemini connection
Read History from File	Process	The process reads history from a file if specified.
Chat with History	Process	The process sets a prompt for the AI, saves the response, and uses it as context for future questions. You must set up a Secure Agent to store and retrieve the history.
Write Chat History in File	Process	Saves the chat history in a file or creates a new file if one doesn't exist.
GenerationConfig_PO	Process Object	Settings for the prompt request

C H A P T E R   3

Using the GeminiAI Chat with History recipe

To use the GeminiAI Chat with History recipe, you must perform the following steps manually:

Step 1: Copy and access the recipe

Step 2: Configure and publish the FileConnectionChatHistory connection

Step 3: Configure and publish the GeminiChatWithHistory connection

Step 4: Configure and publish the processes

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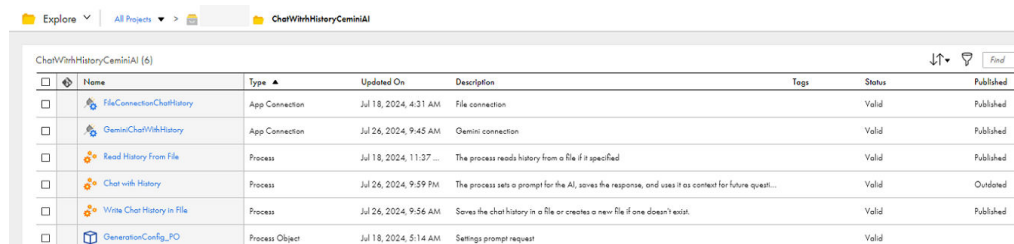
To copy and access the recipe content, perform the following steps:

1. Open the **GeminiAI Chat with History** 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 FileConnectionChatHistory connection

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

1. Open the **FileConnectionChatHistory** connection.

2. From the **Run On** list, select any Secure Agent.

3. On the **Event Targets** tab, in the **Directory** field, enter the directory path to store the files. In the **File Exists** field, select **Append**.
   The following image shows the properties on the **Event Targets** tab:
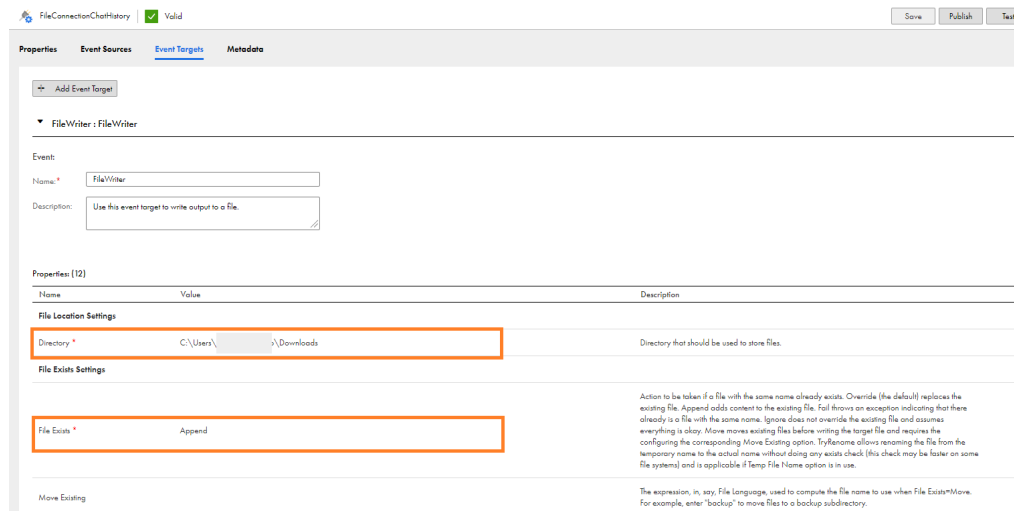
4. Save, test, and publish the connection.

Configuring and publishing the GeminiChatWithHistory connection

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

1. Open the **GeminiChatWithHistory** connection.

2. From the **Run On** list, select **Cloud Server or any Secure Agent**.

3. In the **Connection Properties** section, enter the API key in the **API_Key** property. The **API_Key** property authenticates Gemini connection requests.

4. Save, test, and publish the connection.

Configuring and publishing the processes

1. Open the **Read History from File** process.

2. On the **Start** tab of the **Start** step, select the same Secure Agent that you had selected in the **Run On** field for the **FileConnectionChatHistorydefault** connection.

3. Save and publish the process.

4. Open the **Write Chat History in File** process.

5. Optionally, in the **Prepare History to save in File** step, click the **Assignments** tab. Open the Expression Editor for the **File_Name** field and enter the format to save the file.

   The following image shows the assignments of the **File_Name** input field:
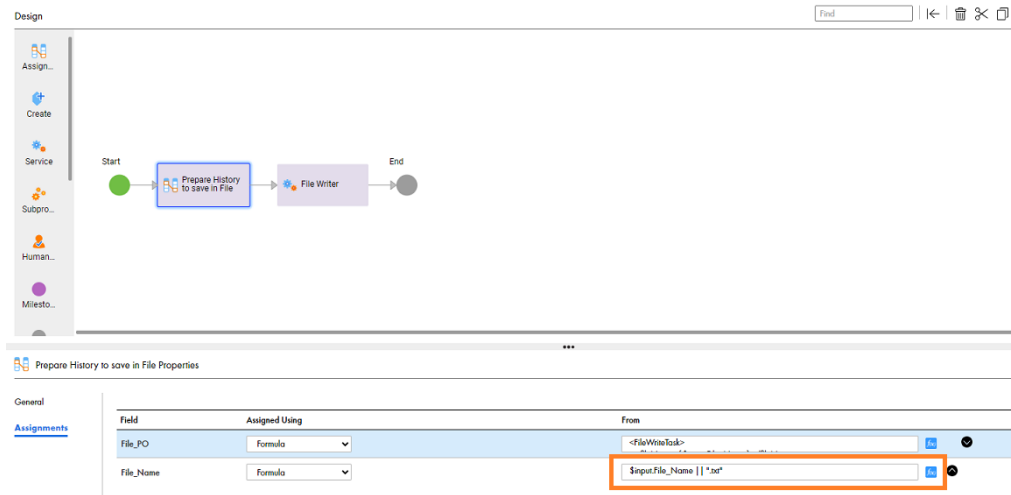
6. Save and publish the process.

7. Open the **Chat with History** main process.

   The process is called by an HTTP request using the following three input parameters mentioned in the **Input Fields** tab of the **Start** step:

   - User_Prompt

   - Path_To_File_With_History

   - User_ID
     The User_ID is used as the file name.

8. On the **Temp Field** tab of the **Start** step, the **Model_LLM** is set to **gemini-1.5 pro**. You can optionally edit the model version. For information about changing the model version, see the Gemini documentation.

9. In the **Configure Request Parameters** step, click the **Assignments** tab. Open the Expression Editor for the **Prompt_Configuration** field and enter the prompt instructions, as shown in the following sample code:

```
<generationConfig>
      <stopSequences>.</stopSequences>
      <candidateCount>1</candidateCount>
      <maxOutputTokens>200</maxOutputTokens>
      <temperature>0.5</temperature>
      <topP>0.1</topP>
      <topK>2</topK>
</generationConfig>
```

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. This property 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.

10. After configuring the prompt instructions, the process reads the file and if it has a history, you must prepare a request with history. The process reads the entire history from the file, but uses only the last question and its response.

The following image shows the assignments of the **Previous_Question** and **Previous_Answer** input fields:

The process reads the file with history. If no file exists or the file is empty, the process prepares a request without history, and then creates or updates the file. It writes the last question and its LLM response in the following format:

```
Qustion_to_LLM: user question
Answer_LLM: Gemini answer
```

If the file contains history, the process prepares the request with history but using only the last question and response. The process then updates the file and appends the new question and answer. You can include additional parsing in a specific location and also include new changes to the request.

11. Save and publish the process.

Invoking the process

The Chat with History process is called by an HTTP request using the User_Prompt, Path_To_File_With_History, and User_ID input parameters. When you invoke the process, it creates a file with history after your first question.

To invoke the Chat with History process, perform the following steps:

1. Open the **Chat with History** process and click **Actions** > **Properties Detail** > **Copy Service URL**.

2. Open a text editor and add the input fields and values to the service URL as shown in the following format:
```
<Cloud Application Integration POD URL>/active-bpel/public/rt/<API_name>?
User_Prompt=<User_Prompt>&Path_To_File_With_History=<C:/folder>/&User_ID=<User_ID>
```

3. Open a browser and paste the service URL.

The process creates or updates a file with the last question and LLM response. If no file exists or the file is empty, the process prepares a request without history. If the file has history, the process uses only the last question and response. The process then updates the file and appends the new question and answer.