Informatica® Cloud Application Integration
September 2024

Azure OpenAI Chat with History

Informatica Cloud Application Integration Azure OpenAI Chat with History
September 2024

This product includes software licensed under the terms at http://www.tcl.tk/software/tcltk/license.html, http://www.bosrup.com/web/overlib/?License, http://www.stlport.org/doc/ license.html, http://asm.ow2.org/license.html, http://www.cryptix.org/LICENSE.TXT, http://hsqldb.org/web/hsqlLicense.html, http://httpunit.sourceforge.net/doc/ license.html, http://jung.sourceforge.net/license.txt , http://www.gzip.org/zlib/zlib_license.html, http://www.openldap.org/software/release/license.html, http://www.libssh2.org, http://slf4j.org/license.html, http://www.sente.ch/software/OpenSourceLicense.html, http://fusesource.com/downloads/license-agreements/fuse-message-broker-v-5-3- license-agreement; http://antlr.org/license.html; http://aopalliance.sourceforge.net/; http://www.bouncycastle.org/licence.html; http://www.jgraph.com/jgraphdownload.html; http://www.jcraft.com/jsch/LICENSE.txt; http://jotm.objectweb.org/bsd_license.html; . http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231; http://www.slf4j.org/license.html; http://nanoxml.sourceforge.net/orig/copyright.html; http://www.json.org/license.html; http://forge.ow2.org/projects/javaservice/, http://www.postgresql.org/about/licence.html, http://www.sqlite.org/copyright.html, http://www.tcl.tk/software/tcltk/license.html, http://www.jaxen.org/faq.html, http://www.jdom.org/docs/faq.html, http://www.slf4j.org/license.html; http://www.iodbc.org/dataspace/iodbc/wiki/iODBC/License; http://www.keplerproject.org/md5/license.html; http://www.toedter.com/en/jcalendar/license.html; http://www.edankert.com/bounce/index.html; http://www.net-snmp.org/about/license.html; http://www.openmdx.org/#FAQ; http://www.php.net/license/3_01.txt; http://srp.stanford.edu/license.txt; http://www.schneier.com/blowfish.html; http://www.jmock.org/license.html; http://xsom.java.net; http://benalman.com/about/license/; https://github.com/CreateJS/EaselJS/blob/master/src/easeljs/display/Bitmap.js; http://www.h2database.com/html/license.html#summary; http://jsoncpp.sourceforge.net/LICENSE; http://jdbc.postgresql.org/license.html; http://protobuf.googlecode.com/svn/trunk/src/google/protobuf/descriptor.proto; https://github.com/rantav/hector/blob/master/LICENSE; http://web.mit.edu/Kerberos/krb5-current/doc/mitK5license.html; http://jibx.sourceforge.net/jibx-license.html; https://github.com/lyokato/libgeohash/blob/master/LICENSE; https://github.com/hjiang/jsonxx/blob/master/LICENSE; https://code.google.com/p/lz4/; https://github.com/jedisct1/libsodium/blob/master/LICENSE; http://one-jar.sourceforge.net/index.php?page=documents&file=license; https://github.com/EsotericSoftware/kryo/blob/master/license.txt; http://www.scala-lang.org/license.html; https://github.com/tinkerpop/blueprints/blob/master/LICENSE.txt; http://gee.cs.oswego.edu/dl/classes/EDU/oswego/cs/dl/util/concurrent/intro.html; https://aws.amazon.com/asl/; https://github.com/twbs/bootstrap/blob/master/LICENSE; https://sourceforge.net/p/xmlunit/code/HEAD/tree/trunk/LICENSE.txt; https://github.com/documentcloud/underscore-contrib/blob/master/LICENSE, and https://github.com/apache/hbase/blob/master/LICENSE.txt.

This product includes software licensed under the Academic Free License (http://www.opensource.org/licenses/afl-3.0.php), the Common Development and Distribution License (http://www.opensource.org/licenses/cddl1.php) the Common Public License (http://www.opensource.org/licenses/cpl1.0.php), the Sun Binary Code License Agreement Supplemental License Terms, the BSD License (http:// www.opensource.org/licenses/bsd-license.php), the new BSD License (http://opensource.org/licenses/BSD-3-Clause), the MIT License (http://www.opensource.org/licenses/mit-license.php), the Artistic License (http://www.opensource.org/licenses/artistic-license-1.0) and the Initial Developer's Public License Version 1.0 (http://www.firebirdsql.org/en/initial-developer-s-public-license-version-1-0/).

This product includes software copyright © 2003-2006 Joe Walnes, 2006-2007 XStream Committers. All rights reserved. Permissions and limitations regarding this software are subject to terms available at http://xstream.codehaus.org/license.html. This product includes software developed by the Indiana University Extreme! Lab. For further information please visit http://www.extreme.indiana.edu/.

This product includes software Copyright (c) 2013 Frank Balluffi and Markus Moeller. All rights reserved. Permissions and limitations regarding this software are subject to terms of the MIT license.

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

DISCLAIMER: Informatica LLC provides this documentation "as is" without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of noninfringement, merchantability, or use for a particular purpose. Informatica LLC does not warrant that this software or documentation is error free. The information provided in this software or documentation may include technical inaccuracies or typographical errors. The information in this software and documentation is subject to change at any time without notice.

NOTICES

This Informatica product (the "Software") includes certain drivers (the "DataDirect Drivers") from DataDirect Technologies, an operating company of Progress Software Corporation ("DataDirect") which are subject to the following terms and conditions:

1. THE DATADIRECT DRIVERS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT.
2. IN NO EVENT WILL DATADIRECT OR ITS THIRD PARTY SUPPLIERS BE LIABLE TO THE END-USER CUSTOMER FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL OR OTHER DAMAGES ARISING OUT OF THE USE OF THE ODBC DRIVERS, WHETHER OR NOT INFORMED OF THE POSSIBILITIES OF DAMAGES IN ADVANCE. THESE LIMITATIONS APPLY TO ALL CAUSES OF ACTION, INCLUDING, WITHOUT LIMITATION, BREACH OF CONTRACT, BREACH OF WARRANTY, NEGLIGENCE, STRICT LIABILITY, MISREPRESENTATION AND OTHER TORTS.

Publication Date: 2024-12-10

Table of Contents

Preface

Use *Azure OpenAI Chat with History* to learn how to maintain chat history in a file and use it as context for the next user query and the Large Language Model's (LLM) response. This guide assumes that you have an understanding of the Azure OpenAI Connector concepts.

CHAPTER 1

Introduction to Azure OpenAI Chat with History recipe

The Azure OpenAI Chat with History recipe is initiated using an HTTP request. You can maintain the chat history in a file and use it as context for the next user query and the Large Language Model's (LLM) response.

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.
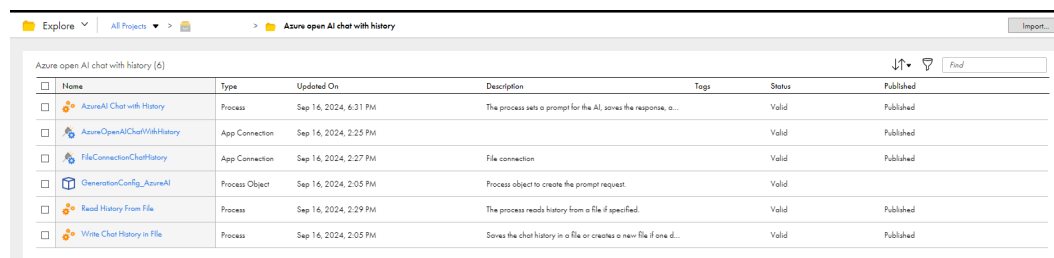
You can use the following file formats:

`.txt`, `.xml`, `.json`, `.html`, `.csv`, `.md`, and `.yaml`

Azure OpenAI Chat with History recipe contents

The Azure OpenAI Chat with History recipe contains a process object, two app connections, and three processes.

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

Azure OpenAI Chat with History recipe assets

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

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

Azure OpenAI Chat with History process

The Azure OpenAI Chat with History process is called by an HTTP request. It maintains chat history in a file and uses it as context for the last user question and LLM answer.

The process is called by an HTTP request using the following input parameters configured on 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.

After configuring the prompt instructions, the process reads the file. 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:

```
Question_to_LLM: user question
Answer_LLM: Azure OpenAI answer
```

If the file contains history, the process uses 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.

Using the Azure OpenAI Chat with History recipe

To use the Azure OpenAI 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 AzureOpenAIChatWithHistory connection

Step 4: Configure and publish the processes

Step 5: Invoke the processes

Step 1: Copy and access the recipe
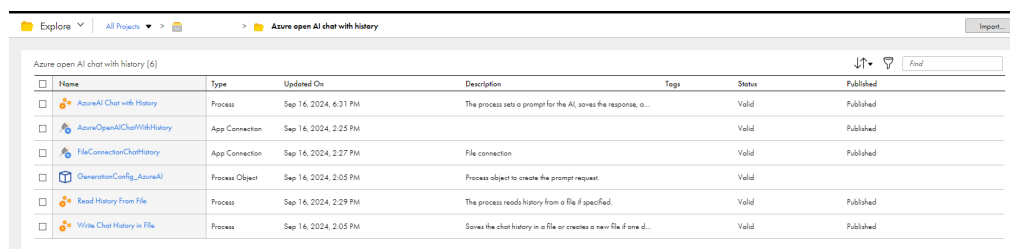
Copy the pre-configured assets in the recipe to a separate project or folder.

1. Open the **Azure OpenAI 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:

Step 2: Configure and publish the FileConnectionChatHistory connection

Configure the file directory and specify the action to be taken if a file already exists. Then, publish the FileConnectionChatHistory connection.

1. Open the **FileConnectionChatHistory** connection.
2. From the **Type** list, select **File**.
3. From the **Run On** list, select any Secure Agent.
4. On the **Event Targets** tab, in the **Directory** field, enter the directory to store the files. In the **File Exists** field, select **Append**.
5. Save, test, and publish the connection.

Step 3: Configure and publish the AzureOpenAIChatWithHistory connection

Configure the endpoint URL and API key in the AzureOpenAIChatWithHistory connection, and then publish the connection.

1. Open the **AzureOpenAIChatWithHistory** connection.
2. From the **Type** list, select **AzureOpenAI**.
3. From the **Run On** list, select **Cloud Server or any Secure Agent**.
4. In the **Connection Properties** section, enter values for the following properties:

| Property	Description
Endpoint_URL	The REST API endpoint for Azure OpenAI. You can find this value in the **Keys & Endpoint** section when examining your resource from the Azure portal. Alternatively, you can find the value in **Azure OpenAI Studio** > **Playground** > **Code View**.
API_Key	The API key to authenticate Azure OpenAI connection requests. You can find this value in the **Keys & Endpoint** section when examining your resource from the Azure portal. Alternatively, you can find the value in **Azure OpenAI Studio** > **Playground** > **Code View**.

5. Save, test, and publish the connection.

Step 4: Configure and publish the processes

Configure the deployment details of the LLM model and publish 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:
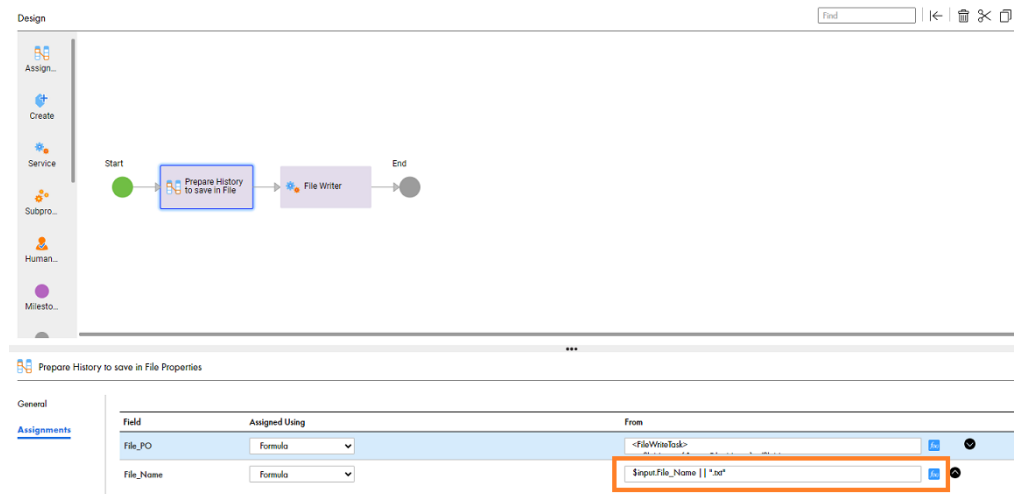
6. Save and publish the process.

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

8. On the **Temp Fields** tab of the **Start** step, enter values for the following fields:

    - In the **api_version** field, enter the API version of the LLM model. Default is **2024-06-01**. You can optionally edit the api version.

    - In the **deployment_id** field, enter the user-specific deployment ID.

9. Optionally, 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_AzureAI>
     <max_tokens>200</max_tokens>
     <temperature>1</temperature>
     <topP>1</topP>
</generationConfig_AzureAI>
```

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

Property	Description
maxTokens	Defines the maximum number of tokens that 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 0.9 balances between deterministic and creative outputs.
topP	An alternative to sampling with temperature where the model considers the results of the token with topP probability. For example, if topP is set to 0.1, the model considers only the top 10% most probable tokens at each step.

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

10. Save and publish the process.

Step 5: Invoke the process

The Chat with History process is called by an HTTP request. 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.