Solidity reference

This page describes the details of the Solidity interface to the oracle, and how to use it in your contract. We assume you are familiar with thehigh-level architecture of Galadriel .

Oracle interface

The oracle is exposed as a contract that you can call from your own contract. It exposes two types of functionality: callingLLMs and callingexternal tools .

The relevant parts of the oracle contract that you can call consist of two parts:

- Methods you call on the oracle: seeMethods you can call on the oracle
- .
- Callback methods you implement on your contract: seeCallback methods
- .

See theUsage section below on how to integrate depending on your use case.

Oracle contract methods

See source code for all the methods in theChatOracle contract .

createLlmCall

This method makes a call to OpenAI GPT-4. We will add other models, including open-source, in the future.

Arguments

- uint promptCallbackId
-
    - identifier which will be passed into the callback.

Returns:

- uint
-
    - an internal counter of the oracle, which is incremented on every call.

createFunctionCall

This method makes a call to a tool — see options below. For every tool the input and output are of string type, and you choose a tool by specifying thefunctionType parameter.

Arguments

- uint functionCallbackId
-
    - identifier which will be passed into the callback.
- string functionType
-
    - specifies which function to call. Must be one of:image_generation
- ,web_search
- .
- string functionInput
-
    - input to the function.

Returns

- uint

-
    - an internal counter of the oracle, which is incremented on every call.

The currently available tools are:

Image generation * functionType * :image_generation * Functionality: given an input prompt, run DALL-E 3 * to generate an image. * Input: a string with the prompt that is directly given as input to DALL-E. * Output: a string with the URL of the generated image. Web search * functionType * :web_search * Functionality: given a query string, run a Google search and return the list of results. * Input: a string with the search query. * Output: a string with a list of search results in JSON format.

Callback methods

These are callbacks the oracle will call once the response is available, and are described in the IChatGpt interface .

onOracleLlmResponse

Called when the result of an LLM call, created with createLlmCall , is available.

Arguments

- uint callbackId
-
    - the identifier you passed into thecreateLlmCall
- method.
- string response
-
    - the result of the LLM call.
- string errorMessage
-
    - an error message. Contains an empty string if there was no error, and a description of the error otherwise.

onOracleFunctionResponse

Called when the result of a tool call, created with createFunctionCall , is available.

Arguments

- uint callbackId
-
    - the identifier you passed into thecreateFunctionCall
- method.
- string response
-
    - the result of the tool call.
- string errorMessage
-
    - an error message. Contains an empty string if there was no error, and a description of the error otherwise.

getMessageHistoryContents and getMessageHistoryRoles

These two methods in combination are used to provide the oracle with the message history necessary for the LLM call. They are called by the oracle aftercreateLlmcall .

The methods should each return a list, one with message contents and the other listing the roles of the message authors. The list lengths should be equal for a givencallbackId .

For example, if the message history is the following:

role content system You are a helpful assistant user Hello! assistant Hi! How can I help? user How big is the Sun? ThengetMessageHistoryContents should return the following list of 4 items:

["You are a helpful assistant" , "Hello!" , "Hi! How can I help?" , "How big is the Sun?"] …andgetMessageHistoryRoles

should return the following list of 4 items:

["system" , "user" , "assistant" , "user"] The signatures of these methods should be:

getMessageHistoryRoles

Arguments:

- uint callbackId
-
    - the identifier you passed into thecreateLlmCall
- method.

Returns:

- string[]
-
    - a list of roles.

getMessageHistoryContents

Arguments:

- uint callbackId
-
    - the identifier you passed into thecreateLlmCall
- method.

Returns:

- string[]
-
    - a list of message contents.

Usage

Calling an LLM

If you want to call an LLM, you need to:

1 Call thecreateLlmCall method on the oracle contract. This will create a new LLM call. 2 Implement two methods into your contract that the oracle will call to get the message history necessary for the LLM, including the system prompt:getMessageHistoryContents and getMessageHistoryRoles . 3 Implement theonOracleLlmResponse method into your contract. This method will be called when the LLM result is available (in a separate transaction potentially tens of seconds later).

Calling an external tool

If you are calling an external tool, you need to:

1 Call thecreateFunctionCall method on the oracle contract. This will create a new tool call. 2 Implement theonOracleFunctionResponse method into your contract. This method will be called when the tool result is available (in a separate transaction potentially tens of seconds later).

Since method calls do not reference a message history, you don’t need to implement any history getter methods.

Examples

Please refer to theexample contracts , corresponding to the outlineduse cases , for complete usage examples.