AI-Engine Javascript SDk

Introduction

AI Engine SDK provides an easy-to-use interface for integrating advanced AI functionalities into your applications. With a simple API and comprehensive documentation, developers can effortlessly leverage powerful AI tools and enhance their projects.

Getting Started

Prerequisites

Before you begin, ensure you have met the following requirements:

- Node.js v20 or higher
- is installed on your system.
- A validAgentverse API key
- . You can obtain this fromAgentverse(opens in a new tab)
- .

ℹ To find out how to generate an API KEY, check out the documentation regardingAgentverse API keys(opens in a new tab)
.

Installation

You can install the AI Engine SDK using npm or yarn. Follow the commands below.

npm

install

@fetchai/ai-engine-sdk yarn

add

@fetchai/ai-engine-sdk ℹ Explore the additional resources we have on AI Engine compatible Agents:

- Make your agents AI Engine compatible
- Agent Functions
- Register Agent Functions
- DeltaV

SDK overview

Creating the AI Engine client object

To set up and initialize the AI Engine client with your API key, use the following code snippet. This client will allow you to interact with the AI Engine API, enabling you to start sessions, get function groups, and manage messages.

import { AiEngine } from

"@fetchai/ai-engine-sdk" ; const

aiEngine

=

new

AiEngine (apiKey);

Querying Function Groups in AI Engine SDK

const

functionGroups

=

await

aiEngine .getFunctionGroups (); const

functionGroupId

=

functionGroups .find ((g) =>

g .name ===

"Fetch Verified"); If you would like to use the functions in your own My Functions function group, you can use this filter instead:

const

functionGroupId

=

functionGroups .find ((g) =>

g .name ===

"My Functions");

Creating a session with the AI Engine

This code starts a session with the AI Engine using the UUID of the selected function group.

const

session

=

await

aiEngine .createSession (functionGroupId);

Starting the conversation with an arbitrary objective

await

session .start ("Find a holiday destination");

Querying new messages

This line asynchronously queries the AI engine to retrieve a list of messages related to the current session. UsesetTimeout orsetInterval to regularly check for new messages.

const

messages

=

await

session .getMessages ();

Handling Different Types of Messages

Task Selection Message (isTaskSelectionMessage

)

This message is generated when the AI engine suggests functions based on the initial objective or provides multiple options for a function.

Agent Message (isAgentMessage

)

This is a regular question from the AI Engine that the user needs to reply to with a string.

AI Engine Message (isAiEngineMessage

)

This message type doesn't require a user response; it simply notifies the user about something.

Confirmation Message (isConfirmationMessage

)

This message is sent when the AI Engine has gathered all necessary inputs for the agent's function, indicating that the context is complete.

Stop Message (isStopMessage

)

This message is sent when the session has ended, and the AI Engine no longer expects any replies from the user.

ℹ All message types (except for the AI engine message and stop message) expect a response from the user.

SDK Methods for Replying

Task Selection Message

Usesession.submitTaskSelection .

Agent Message

Usesession.submitResponse .

Confirmation Message

Use eithersession.submitConfirmation to confirm, orsession.rejectConfirmation to reject the context generated by the AI engine.

Deleting session

After finishing a conversation with the AI Engine, you can delete the session by using the following command.

await

session .delete ()

Example Usage of the SDK

The following example demonstrates how to use the AI Engine SDK to interact with the AI Engine. The script sets up the client, queries function groups, creates a session, and handles different types of messages in a loop.

Self hosted example.ts import

*

as readline from

"node:readline" ;

import

{ AiEngine , isAgentMessage , isAiEngineMessage , isConfirmationMessage , isStopMessage , isTaskSelectionMessage , }

from

"@fetchai/ai-engine-sdk" ;

const apiKey

process . env ["AV_API_KEY"] ?? "" ;

const snooze

(ms : number) => new Promise ((resolve) =>

setTimeout (resolve, ms)) ;

const main

async () =>

{ const rl = readline . promises . createInterface ({ input : process.stdin, output: process.stdout, }) ;

const aiEngine = new AiEngine (apiKey) ;

const functionGroups = await aiEngine . getFunctionGroups () ; const publicGroup = functionGroups . find ((g) => g.name ===

"Fetch Verified") ; if (publicGroup == = undefined) { throw new Error ('Could not find "Public" function group.') ; }

const session = await aiEngine . createsession (publicGroup.uuid) ; await session . start (await rl. question ("What is your objective: ")) ;

try

{ let emptyCount = 0 ; let sessionEnded = false; while

(emptyCount <

12)

{ const messages = await session . getMessages () ; if (messages . length == = 0) { emptyCount++; }

else

{ emptyCount = 0 ; }

for (const message of messages) { if (isTaskSelectionMessage (message)) { console . log ("Please select a task from the list below:") ; console . log ("") ; for (const option of message . options) { console . log ({option.key}: {option.title}) ; }

const optionIndex = parseInt (await rl. question ("\nEnter task number: "),) ;

// check the index if (optionIndex <

0 || optionIndex

= message . options . length) { throw new Error ("Invalid task number") ; }

await session . submitTaskSelection (message, [message.options[optionIndex]!,]) ; } else

if (isAgentMessage (message)) { console . log ("Agent: " , message.text) ;

const response = await rl . question ("User (enter to skip): ") ; if (response == = "exit") { break ; }

if (response != = "") { await session . submitResponse (message, response) ; } }

else

if (isAiEngineMessage (message)) { console . log ("Engine:" , message.text) ; }

else

if (isConfirmationMessage (message)) { console . log ("Confirm:" , message.payload) ;

const response = await rl . question ("\nPress enter to confirm, otherwise explain issue:\n" ,) ;

if (response == = "") { console . log ("Sending confirmation...") ; await session . submitConfirmation (message) ; }

else

{ await session . rejectConfirmation (message, response) ; } }

else

if (isStopMessage (message)) { console . log ("\nSession has ended") ; sessionEnded = true; break ; }

else

{ console . error ("???" , message) ; } }

//

if the session has concluded then break if (sessionEnded) { break ; }

await

snooze (1200) ; } } catch (e)

{ console . error ("Error" , e) ; }

finally

{ // clean up the session await session . delete () ;

// stop the readline interface rl . close () ; } };

main () ; Last updated on October 17, 2024

Was this page helpful?

You can also leave detailed feedbackon Github

AI Engine Python SDK Startup Idea analyzer

On This Page