Sign Out

Client SDKs

Bee provides SDKs for various platforms to make it easier to integrate Bee into your application. The SDKs are open source and available on GitHub.

Python

```
pip install beeai
```

Example

```
import asyncio
from beeai import Bee

async def main():
    # Create an instance of the Bee SDK with your API key
    bee = Bee("your_api_key")

    # Conversation methods
    conversations = await bee.get_conversations("me")
    print("Conversations:", conversations["conversations"])

    if conversations.get("conversations"):
        conversation_id = conversations["conversations"][0].get("id")
        if conversation_id:
            full_conversation = await bee.get_conversation("me", conversation_i
            print("Full conversation:", full_conversation)

            await bee.end_conversation("me", conversation_id)
            print("Conversation ended")

            await bee.retry_conversation("me", conversation_id)
            print("Conversation retried")

            await bee.delete_conversation("me", conversation_id)
            print("Conversation deleted")
        else:
            print("Conversation ID not found in the response.")
    else:
        print("No conversations found in the response.")

    # Fact methods
    facts = await bee.get_facts("me")
    print("Facts:", facts["facts"])

    if facts.get("facts"):
        fact_id = facts["facts"][0].get("id")
        if fact_id:
            fact = await bee.get_fact("me", fact_id)
            print("Fact:", fact)

            updated_fact = await bee.update_fact("me", fact_id, {"text": "Updat
            print("Updated fact:", updated_fact)

            await bee.delete_fact("me", fact_id)
```

```
        print("Fact deleted")
    else:
        print("Fact ID not found in the response.")
else:
    print("No facts found in the response.")

new_fact = await bee.create_fact("me", {"text": "New fact", "confirmed": Tru
print("New fact:", new_fact)

# Todo methods
todos = await bee.get_todos("me")
print("Todos:", todos["todos"])

if todos.get("todos"):
    todo_id = todos["todos"][0].get("id")
    if todo_id:
        todo = await bee.get_todo("me", todo_id)
        print("Todo:", todo)

        updated_todo = await bee.update_todo("me", todo_id, {"text": "Updat
        print("Updated todo:", updated_todo)

        await bee.delete_todo("me", todo_id)
        print("Todo deleted")
    else:
        print("Todo ID not found in the response.")
else:
    print("No todos found in the response.")

new_todo = await bee.create_todo("me", {"text": "New todo", "completed": Fa
print("New todo:", new_todo)

# Location methods
locations = await bee.get_locations("me")
print("Locations:", locations["locations"])

# Define event listeners
@bee.on("connect")
def on_connect():
    print("Connected to the socket")
    print("Waiting for events...")

@bee.on("disconnect")
def on_disconnect():
    print("Disconnected from the socket")

@bee.on("new_conversation")
def on_new_conversation(data):
    print("New conversation:", data)

@bee.on("process_conversation")
def on_process_conversation(data):
    print("Conversation begun processing:", data)

@bee.on("processed_conversation")
def on_processed_utterance(data):
    print("Conversation processed:", data)

@bee.on("interim_conversation_summary")
def on_interim_conversation_summary(data):
    print("Interim summary updated:", data)

@bee.on("new_utterance")
def on_new_utterance(data):
    print("New utterance:", data)

# Connect for real-time events
bee.connect()
```

```
    # Keep listening for events
    while True:
        await asyncio.sleep(1)

asyncio.run(main())
```

Javascript

```
npm i beeai
```

Example

```
const Bee = require('beeai');

const bee = new Bee({ apiKey: 'your_api_key' });

async function main() {
    try {
        // Conversation methods
        const conversations = await bee.getConversations('me');
        console.log('Conversations:', conversations.conversations);

        if (conversations.conversations.length > 0) {
            const conversationId = conversations.conversations[0].id;
            const fullConversation = await bee.getConversation('me', conversati
            console.log('Full conversation:', fullConversation);

            await bee.endConversation('me', conversationId);
            console.log('Conversation ended');

            await bee.retryConversation('me', conversationId);
            console.log('Conversation retried');

            await bee.deleteConversation('me', conversationId);
            console.log('Conversation deleted');
        }

        // Fact methods
        const facts = await bee.getFacts('me');
        console.log('Facts:', facts.facts);

        if (facts.facts.length > 0) {
            const factId = facts.facts[0].id;
            const fact = await bee.getFact('me', factId);
            console.log('Fact:', fact);

            const updatedFact = await bee.updateFact('me', factId, { text: 'Upd
            console.log('Updated fact:', updatedFact);

            await bee.deleteFact('me', factId);
            console.log('Fact deleted');
        }

        const newFact = await bee.createFact('me', { text: 'New fact', confirme
        console.log('New fact:', newFact);

        // Todo methods
        const todos = await bee.getTodos('me');
        console.log('Todos:', todos.todos);
```

```
        if (todos.todos.length > 0) {
            const todoId = todos.todos[0].id;
            const todo = await bee.getTodo('me', todoId);
            console.log('Todo:', todo);

            const updatedTodo = await bee.updateTodo('me', todoId, { text: 'Upd
            console.log('Updated todo:', updatedTodo);

            await bee.deleteTodo('me', todoId);
            console.log('Todo deleted');
        }

        const newTodo = await bee.createTodo('me', { text: 'New todo', complete
        console.log('New todo:', newTodo);

        // Location methods
        const locations = await bee.getLocations('me');
        console.log('Locations:', locations.locations);

        // Connect to the socket
        bee.connect();

        // Connection test
        bee.socket.on('connect', () => {
            console.log('Connected to the socket');
            console.log('Waiting for events...');
        });

        bee.socket.on('disconnect', () => {
            console.log('Disconnected from the socket');
        });

        // Listen for events
        bee.on('new_conversation', (data) => {
            console.log('New conversation:', data);
        });
        bee.on('process_conversation', (data) => {
            console.log('Conversation begun processing:', data);
        });
        bee.on('processed_conversation', (data) => {
            console.log('Conversation processed:', data);
        });
        bee.on('interim_conversation_summary', (data) => {
            console.log('Interim summary updated:', data);
        });
        bee.on('new_utterance', (data) => {
            console.log('New utterance:', data);
        });

    } catch (error) {
        console.error('Error:', error);
    }
}

main();

process.on('SIGINT', () => {
    console.log('Termination signal received. Disconnecting...');
    bee.disconnect();
    process.exit(0); // Ensure the process exits properly
});
```