Yandex AI Studio
- Концепции
  - О сервисе Yandex AI Studio
  - Model Gallery
  - Agent Atelier
  - AI Search
  - MCP Hub
  - Yandex Workflows
  - Квоты и лимиты
  - Термины и определения
- Пошаговые инструкции
- Переход с AI Assistant API на Responses API
- Совместимость с OpenAI
- Yandex Cloud ML SDK
- Справочники API
- Практические руководства
- Промптинг
- Управление доступом
- Правила тарификации
- Аудитные логи Audit Trails
- Публичные материалы
- История изменений
- Вопросы и ответы
- Решение проблем

Переход с AI Assistant API на Responses API

Статья создана Yandex Cloud   Обновлена 21 ноября 2025 г.

AI Assistant API позволял создавать AI-ассистентов, которые хранили контекст взаимодействия с пользователем в тредах, могли использовать инструменты Retrieval и WebSearch, а также получать промежуточные ответы модели.

Для новых и текущих проектов мы рекомендуем использовать Responses API — простой и гибкий интерфейс, который позволяет сохранять контекст диалога. Responses API предоставляет встроенные инструменты поиска по файлам и поиска в интернете, позволяет использовать собственные функции, вызывать внешние инструменты через MCP-серверы и обеспечивает высокую производительность.

> **Важно**
>
> С 10 декабря 2025 года функциональность AI Assistant API в Yandex AI Studio перестанет поддерживаться и будет полностью отключена 26 января 2026 года. Переведите все свои актуальные проекты на Responses API до 26 января 2026 года.

С помощью этого руководства вы сможете преобразовать существующих AI-ассистентов, построенных на основе AI Assistant API, в AI-агентов на базе Responses API.

AI-агент в Responses API — это экземпляр модели с заданной конфигурацией: инструкцией, настроенными инструментами и контекстом взаимодействия. AI-агент определяет поведение модели и способ ее взаимодействия с пользователем и другими системами.

Различия между AI Assistant API и Responses API

Понятия и инструменты, использующиеся в AI Assistant API и Responses API, различаются:

AI Assistant API	Responses API
Assistant — AI-ассистент как ресурс сервиса.	При работе через API отдельный ресурс не создается, все настройки передаются непосредственно в методе `responses.create()`. В консоли управления конфигурацию AI-агента можно сохранить с уникальным идентификатором и затем использовать в Responses API.
Thread — тред диалога.	Не существует тредов, содержащих контекст всех сообщений. Историю переписки можно передавать как контекст нового вызова `response` в поле `previous_response_id`.
Run — запуск AI-ассистента для треда.	Объект `response` — результат выполнения метода `responses.create()`. Каждый объект `response` — это аналог запуска (`Run`) AI Assistant API, в котором содержится готовый ответ.
Retrieval — инструмент поиска по поисковым индексам.	Встроенный инструмент `file_search` для поиска по файлам. Для поиска необходимо указать массив индексов Vector Store.
WebSearch — инструмент поиска в интернете.	Встроенный инструмент `web_search` для поиска в интернете. Можно указать домен и регион поиска.
Streaming — получение промежуточных ответов модели.	Метод `client.responses.stream()`.

Концептуальные различия

Основные концептуальные отличия Responses API и AI Assistant API:

1. В Responses API не существует ассистентов как отдельных ресурсов сервиса AI Studio.

AI Assistant API	Responses API
В AI Assistant API необходимо создать AI-ассистента один раз. После этого его можно запускать в разных тредах.	В Responses API для каждого запроса необходимо указывать: ◦ модель `model`; ◦ инструкции `instructions`; ◦ используемые инструменты `tools`; ◦ параметры модели (`temperature`, `max_output_tokens` и т.д.)

   Для адаптации вашего кода на Responses API воспользуйтесь одним из двух вариантов сохранения настроек модели:

   - Вынесите конфигурацию AI-ассистента из AI Assistant API в код вашего приложения.
   - В консоли управления в разделе **Agent Atelier** задайте и сохраните конфигурацию модели. После этого вы сможете использовать ее в коде вашего приложения, указывая идентификатор сохраненного агента в запросе.

2. Контекст передается не в тредах, а в сообщениях в поле `previous_response_id`.

AI Assistant API	Responses API
В AI Assistant API контекст хранится в тредах (`thread`), и каждый запуск (`run`) перечитывает его.	В Responses API реализован механизм, который позволяет передать идентификатор предыдущего сообщения в поле `previous_response_id`, чтобы учитывать историю сообщений.

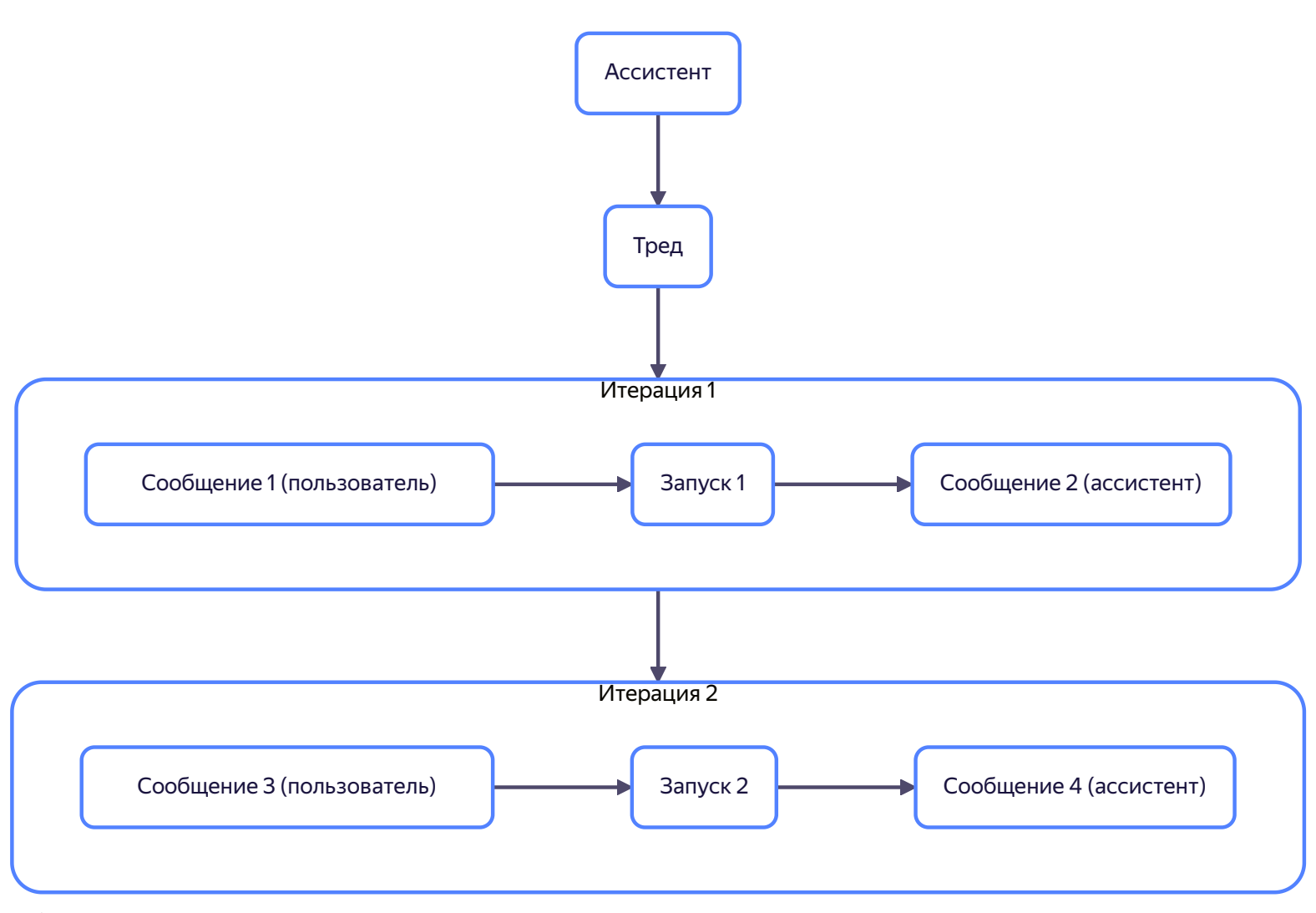
> **Примечание**
>
> Срок хранения сообщений ограничен и составляет 30 дней с момента их создания методом `responses.create()`.

3. Инструменты встроены в Responses API и не требуют подключения дополнительных библиотек.

AI Assistant API	Responses API
Retrieval и WebSearch в AI Assistant API настраиваются глобально как инструменты ассистента и используют внешние источники и отдельные поисковые индексы.	В Responses API сценарии поиска по файлам и поиска в интернете реализованы через поле `tools`, которое позволяет задавать разный набор инструментов для каждого запроса. Доступные следующие значения поля: ◦ `{"type": "file_search"}` ◦ `{"type": "web_search"}`

Как перенести простого текстового ассистента на Responses API

Схема работы AI-ассистента через AI Assistant API



Работа с ассистентом в AI Assistant API состоит из следующих этапов:

- Создание AI-ассистента, в котором хранятся настройки модели, инструменты и базовые инструкции.
- Создание треда (контейнера для диалога).
- Создание сообщения в треде (сообщение пользователя).
- Запуск ассистента для обработки треда.
- Опрос состояния запуска, чтобы дождаться завершения его выполнения.
- Получение сообщения из треда (ответ модели).

Схема работы с AI-агентом через Responses API

В Responses API AI-агент — это набор параметров в коде, а контекст предыдущего диалога передается через поле `previous_response_id`.

Логика вашего приложения должна сохранять идентификатор `response.id` как аналог треда в AI Assistant API. Чтобы получить ответ с учетом истории переписки, передавайте идентификатор последнего сообщения `response.id` в поле `previous_response_id` с каждым последующим сообщением пользователя.

Пример работы простого текстового AI-агента на Responses API:

Python SDK

```python
from openai import OpenAI

YANDEX_CLOUD_FOLDER = "<идентификатор_каталога>"
YANDEX_CLOUD_MODEL = "<URI_модели>"
YANDEX_CLOUD_API_KEY = "<API-ключ_сервисного_аккаунта>"
# или YANDEX_CLOUD_IAM_TOKEN = "<IAM-токен>"

previous_id = None  # храним ID последнего ответа ассистента

client = OpenAI(
    api_key=YANDEX_CLOUD_API_KEY,
    project=YANDEX_CLOUD_FOLDER,
    base_url="https://rest-assistant.api.cloud.yandex.net/v1",
)

print("Чат с агентом (для выхода введите 'выход')\n")

while True:
    user_input = input("Вы: ")
    if user_input.lower() in ("exit", "quit", "выход"):
        print("Чат завершен.")
        break

    response = client.responses.create(
        model=f"gpt://{YANDEX_CLOUD_FOLDER}/{YANDEX_CLOUD_MODEL}",
        input=[{"role": "user", "content": user_input}],
        instructions="Ты — текстовый агент, который ведет диалог и дает информативные ответы на вопросы пользов
        previous_response_id=previous_id,  # передаем контекст, если он есть
    )

    # сохраняем ID для следующего шага
    previous_id = response.id

    # выводим ответ агента
    print("Агент:", response.output_text)
```

Как перенести на Responses API ассистента с инструментами

Процесс переноса AI-ассистента на Responses API зависит от подключенных инструментов и режима получения результатов генерации.

Сценарии RAG с Retrieval

В сценариях поиска по файлам и внутренним базам знаний используются поисковые индексы AI Assistant API и инструмент Retrieval: AI-ассистент генерирует ответы на основе загруженных в индексы документов и возвращает метаданные использованных файлов.

В AI Assistant API инструмент Retrieval был привязан к ассистенту:

SDK

```python
# Сначала создается инструмент для работы с существующим поисковым индексом.
tool = sdk.tools.search_index(
    search_index,
    call_strategy={
        "type": "function",
        "function": {"name": "guide", "instruction": instruction},
    },
)

# Затем создается ассистент, использующий этот инструмент.
assistant = sdk.assistants.create(
    "yandexgpt",
    instruction="Ты — помощник по внутренней документации компании. Отвечай вежливо. Если информация не содержи
    tools=[tool],
)
thread = sdk.threads.create()
```

Чтобы перенести AI-ассистента с подключенным инструментом Retrieval, выполните следующие действия:

1. Все документы подключенного поискового индекса загрузите в векторное хранилище, с которым работает Responses API.
2. При формировании запроса в вашем приложении добавляйте настройки инструмента `file_search`:

Python SDK

```python
import openai
import json

YANDEX_CLOUD_FOLDER = "<идентификатор_каталога>"
YANDEX_CLOUD_MODEL = "<URI_модели>"
VECTOR_STORE_ID = "<идентификатор_хранилища_Vector_Store>"
YANDEX_CLOUD_API_KEY = "<API-ключ_сервисного_аккаунта>"
# или YANDEX_CLOUD_IAM_TOKEN = "<IAM-токен>"


client = openai.OpenAI(
    api_key=YANDEX_CLOUD_API_KEY,
    base_url="https://rest-assistant.api.cloud.yandex.net/v1",
    project=YANDEX_CLOUD_FOLDER,
)

response = client.responses.create(
    model=f"gpt://{YANDEX_CLOUD_FOLDER}/{YANDEX_CLOUD_MODEL}",
    instructions="Ты — умный ассистент. Если спрашивают про ... - ищи в подключенном индексе",
    tools=[
        {
            "type": "file_search",
            "vector_store_ids": [VECTOR_STORE_ID],
        }
    ],
    input="что такое ...",
)

print("Текст ответа:")
print(response.output_text)
print("\n" + "=" * 50 + "\n")

# Полный ответ
print("Полный ответ (JSON):")
print(json.dumps(response.model_dump(), indent=2, ensure_ascii=False))
```

Сценарии с поиском в интернете

В AI Assistant API настройки инструмента `WebSearch` задавались при создании AI-ассистента:

cURL

```json
{
  "folderId": "<идентификатор_каталога>",
  "modelUri": "gpt://<идентификатор_каталога>/yandexgpt-lite/latest",
  "instruction": "Ты — умный помощник финансовой компании. Отвечай вежливо. Для ответов на вопросы воспользуйся
  "tools": [
    {
      "genSearch": {
        "options": {
          "site": {
            "site": [
              "https://cbr.ru/",
              "https://yandex.ru/finance/currencies"
            ]
          },
          "enableNrfmDocs": true
        },
        "description": "Инструмент для получения информации об официальных курсах валют."
      }
    }
  ]
}
```

В Responses API параметры инструмента `web_search` передаются непосредственно в запросе.

Чтобы перенести AI-ассистента с инструментом `WebSearch`, в запросе передавайте настройки инструмента `file_search`:

Python SDK

```python
import openai
import json

YANDEX_CLOUD_FOLDER = "<идентификатор_каталога>"
YANDEX_CLOUD_MODEL = "<URI_модели>"
YANDEX_CLOUD_API_KEY = "<API-ключ_сервисного_аккаунта>"
# или YANDEX_CLOUD_IAM_TOKEN = "<IAM-токен>"


client = openai.OpenAI(
    api_key=YANDEX_CLOUD_API_KEY,
    base_url="https://rest-assistant.api.cloud.yandex.net/v1",
    project=YANDEX_CLOUD_FOLDER,
)

response = client.responses.create(
    model=f"gpt://{YANDEX_CLOUD_FOLDER}/{YANDEX_CLOUD_MODEL}",
    input="Сделай краткий обзор последних новостей об LLM в 2025 году — только факты, без домыслов.",
    # Передаем настройки инструментов
    tools=[
        {
            "type": "web_search",
            "filters": {
                "allowed_domains": [
                    "habr.ru",
                ],
                "user_location": {
                    "region": "213",
                }
            }
        }
      ],
    temperature=0.3,
    max_output_tokens=1000,
)
```

Получение промежуточных результатов генерации ответа

AI Assistant API позволял получать промежуточные результаты генерации ответа. Например, в (ML SDK) использовался метод `run_stream()`:

SDK

```python
run = assistant.run_stream(thread)

# Промежуточные результаты по мере генерации моделью ответа
for event in run:
    print(event._message.parts)

# Все поля окончательного результата
print(f"run {event=}")
```

Responses API также позволяет получать промежуточные результаты генерации, например, с помощью метода `responses.stream()`:

Python SDK

```python
import openai

YANDEX_CLOUD_FOLDER = "<идентификатор_каталога>"
YANDEX_CLOUD_MODEL = "<URI_модели>"
YANDEX_CLOUD_API_KEY = "<API-ключ_сервисного_аккаунта>"
# или YANDEX_CLOUD_IAM_TOKEN = "<IAM-токен>"

client = openai.OpenAI(
    api_key=YANDEX_CLOUD_API_KEY,
    base_url="https://rest-assistant.api.cloud.yandex.net/v1",
    project=YANDEX_CLOUD_FOLDER,
)

# Создаем стриминговый запрос
with client.responses.stream(
    model=f"gpt://{YANDEX_CLOUD_FOLDER}/{YANDEX_CLOUD_MODEL}",
    input="Напиши короткий тост на день рождения, дружелюбный и смешной.",
) as stream:
    for event in stream:
        # Дельты текстового ответа
        if event.type == "response.output_text.delta":
            print(event.delta, end="", flush=True)
        # Событие, показывающее, что ответ завершен
        # elif event.type == "response.completed":
        #     print("\n---\nОтвет завершен")

    # Если необходимо, можно забрать текст ответа целиком
    # final_response = stream.get_final_response()
    # print("\nПолный текст ответа:\n", final_response.output_text)
```

Была ли статья полезна?

Да | Нет

Предыдущая
← Создать MCP-сервер с нуля

Следующая
Совместимость с OpenAI →