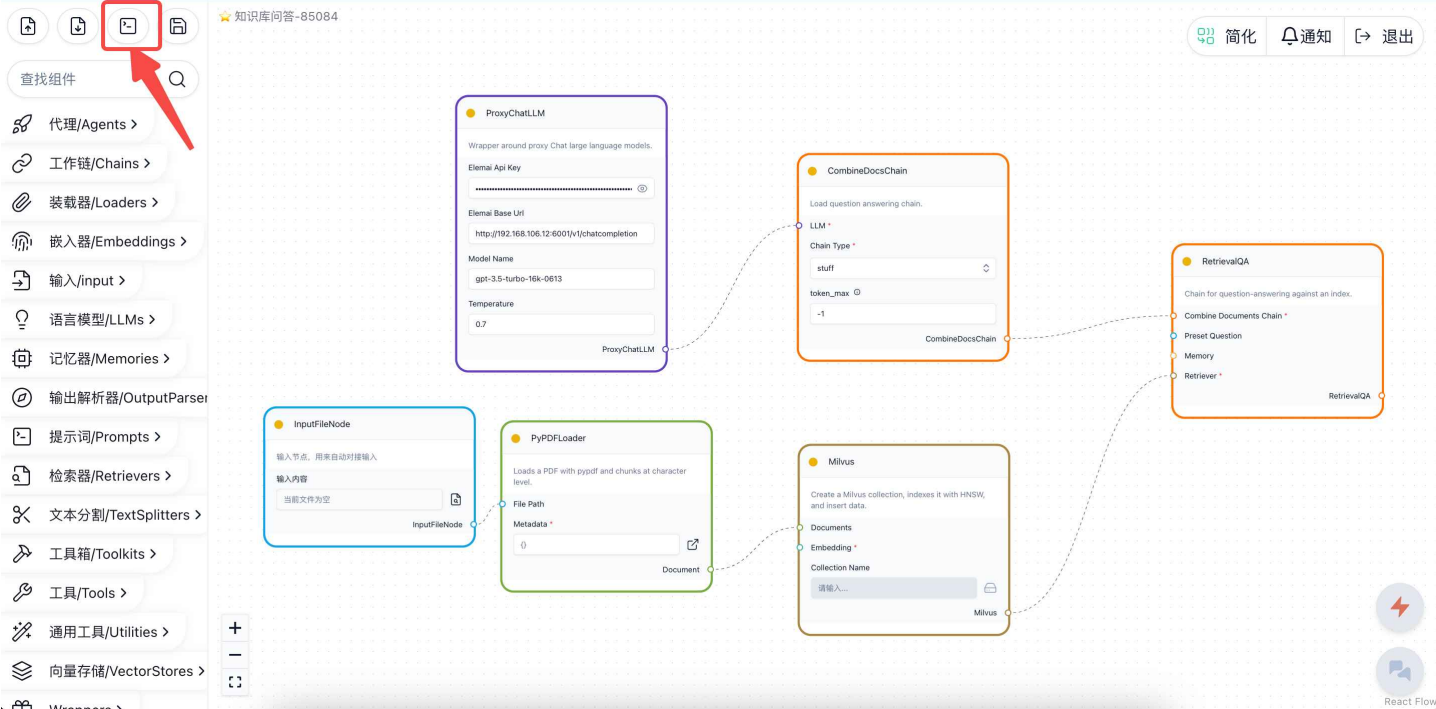


接口文档

技能执行API

查看api 接口



查看API 调用方法

生成代码，将流程集成到外部应用程序中 (打开此页面前请先build技能)。

cURLPython APIPython CodeTweaks

Copy code

```
curl -X POST \
  http://192.168.106.120:3002/api/v1/process/f558ee24-55c5-4c0d-967f-0beb27b6
  -H 'Content-Type: application/json' \
  -d '{"inputs": {"input":"","id":"ConversationChain-A1J5d"}, "tweaks": {
    "ConversationChain-A1J5d": {},
    "ProxyChatLLM-NlOT5": {}
  }}'
```

api 示例

- **cURL**: bash环境下快速测试api功能
- **Python API**: 通过Python调用RestFul API接口，本质和 **cURL**是一样的
- **Python Code**: 将bisheng作为Python 依赖，通过本地实例化技能，本地执行
- **Tweak**: 配置每个组件的变量表，可选，不传就会用技能中默认配置的，如果传可以覆盖技能中配置参数

使用说明

接口协议

1. API 服务地址是 http://IP:Port/api/v1/process/{flow_id}
IP 和Port 是服务部署的地址
2. 参数说明

	是否必填	value类型	说明	示例	备注

flow_id	是	UUID	技能ID		URL传参
inputs	是	Json	对整个技能的问题输入 json里的具体key和技能本身相关，不一定是query	{"query": "什么是金融"}	当输入节点只有一个时，id可不传
history_count	否	int	对于技能里支持Memory，选取几条历史消息进行多轮问答，默认值10		
session_id	否	str	用于session查找，当我们进行多轮时，此参数必填，且建议采用后端生成的key		每次调用，当session_id传入时，返回传入sessionid，当session_id不传时，自动生成id
tweaks	否	Json	对每个组件的控制，可以替换组件输入参数的值		当没有指定组件传参的时候，可以不传
<ul style="list-style-type: none"> ChatOpenAI-MzlaC 		Json	示例，技能中OpenAI大模型组件的配置信息，key为组件名，命名为{组件}-{id}	{"openai_api_key": "sk-xxx"} 或者 {}	当{}为空，表示保持默认值
<ul style="list-style-type: none"> ... 			每个技能中各个组件的参数均可以在调用接口时传进去，如果不传则用技能的默认配置		

3. 返回参数

返回格式采用统一格式：{"status_code":200,"status_message":"SUCCESS","data":{}}

key	类型	含义
data	object	返回内容
data.session_id	string	会话id，用来和入参对应
data.result	object	技能返回的结果
data.result.answer	str	技能统一key返回的LLM 内容
data.result.message_id	int	技能历史消息存储id

data.result.source	int	是否溯源：0 不可溯源，1 普通溯源，2有访问权限，3/4 特殊情况下的溯源
data.result.{key}	string	key是技能里组件定义的输出key，输出的内容和answer一致，唯一的区别是key不固定

Curl 示例

```

1 # cURL示例
2 # 从技能编辑页面获取到技能接口及参数，以下为示例，不代表和实际的一致
3 url="http://IP:3001/api/v1/process/"
4 flow_id="3d01af25-4aea-4193-ad14-a509ae8cacfc"
5 curl -X POST ${url}${flow_id} \
6 -H 'Content-Type: application/json' \
7 -d '{"inputs": {"query": "什么是金融", "id": "纽约时报"},
8     "tweaks": {
9         "ChatOpenAI-MzIaC": {},
10        "BingSearchRun-MzDJj": {},
11        "BingSearchAPIWrapper-UhpQW": {},
12        "ZeroShotAgent-dWiEk": {}
13    }}'
14

```

Python示例

```

1 import requests
2 from typing import Optional
3 BASE_API_URL = "http://{IP}:{port}/api/v1/process"
4 FLOW_ID = "f558ee24-55c5-4c0d-967f-0beb27b660d0"
5 # You can tweak the flow by adding a tweaks dictionary
6 # e.g {"OpenAI-XXXXX": {"model_name": "gpt-4"}}
7 TWEAKS = {
8     "ConversationChain-A1J5d": {},
9     "ProxyChatLLM-NloT5": {}
10 }
11 def run_flow(inputs: dict, flow_id: str, tweaks: Optional[dict] = None) -> dict:
12     """
13     Run a flow with a given message and optional tweaks.
14
15     :param message: The message to send to the flow
16     :param flow_id: The ID of the flow to run

```

```

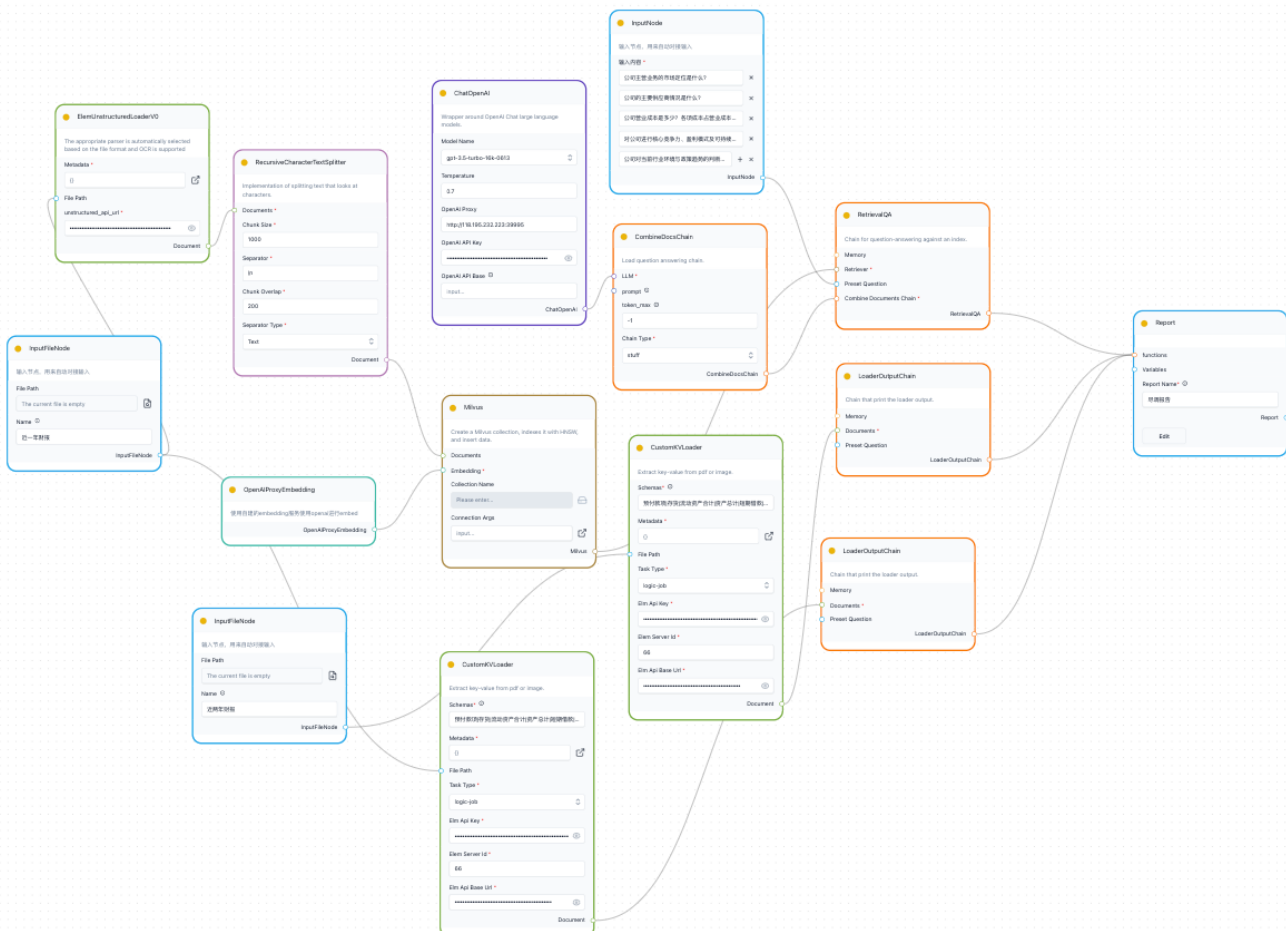
17 :param tweaks: Optional tweaks to customize the flow
18 :return: The JSON response from the flow
19 """
20 api_url = f"{BASE_API_URL}/{flow_id}"
21 payload = {"inputs": inputs}
22 if tweaks:
23     payload["tweaks"] = tweaks
24 response = requests.post(api_url, json=payload)
25 return response.json()
26 # Setup any tweaks you want to apply to the flow
27 inputs = {"input": "什么是金融"}
28 print(run_flow(inputs, flow_id=FLOW_ID, tweaks=TWEAKS))

```

实战Demo

报告生成

报告生成的技能demo



Step 1 查看api接口

```
curl -X POST \
  http://192.168.106.120:3002/api/v1/process/940a528f-eccc-4d43-aa19-55c4725645cf \
  -H 'Content-Type: application/json' \
  -d '{"inputs": {"report_name":"","id":"Report-tuc6Q"}, "tweaks": {
    "ChatOpenAI-u4bHU": {},
    "Report-tuc6Q": {},
    "InputNode-Ly9FG": {},
    "InputFileNode-ozsI2": {},
    "LoaderOutputChain-1fXw5": {},
    "InputFileNode-16U46": {},
    "LoaderOutputChain-NJksg": {},
    "RetrievalQA-oqs2F": {},
    "CombineDocsChain-0oDl0": {},
    "ElemUnstructuredLoaderV0-YNWUm": {},
    "RecursiveCharacterTextSplitter-BotGY": {},
    "CustomKVLoader-ioUcm": {},
    "CustomKVLoader-0Z4XZ": {},
    "Milvus-6HDPE": {},
    "OpenAIProxyEmbedding-g3gK1": {}
  }}'
```

Step 2 查找需要通过接口上传的Node

一般有 inputFileNode/ variableNode

通过应用会话可以看到上传参数的界面，本例子只有两个文件对应api接口里tweak的：

InputFileNode-16u46/ InputFileNode-0zsl2

近一年财报 *

当前文件为空

近两年财报 *

当前文件为空

开始

step3, 准备入参，本例子是两个文件

```
1 import requests
2 def upload_file(local_path: str):
3     server = "http://ip:port"
4     url = server + '/api/v1/knowledge/upload'
5     headers = {}
6     files = {'file': open(local_path, 'rb')}
7     res = requests.post(url, headers=headers, files=files)
8     file_path = res.json()['data'].get('file_path', '')
9     return file_path
```

```
10
11  financeA = upload_file("caibao.pdf")
12  financeB = upload_file("caibao2.pdf")
13
```

Step 4, 组装tweaks

```
1  # 所有apidemo里没有更新的都可以删除，保持代码清晰
2  tweaks = {
3      "InputFileNode-ozsI2": {"file_path": financeA},
4      "InputFileNode-16U46": {"file_path": financeB},
5      "Milvus-6HDPE": {"collection_name": "tmp", "drop_old": True} # 临时知识库，目前
                           需要手动指定collection
6  }
```

Step 5, 执行

```
1  response =
    requests.post(url="http://192.168.106.120:3002/api/v1/process/940a528f-eccc-
    4d43-aa19-55c4725645cf",
2      json={"inputs": {"report_name": "", "id": "Report-tuc6Q"}, "tweaks": tweaks})
3
4
5  print(response.text)
```

Preset Question预置问题

之前接口不支持Preset Question——即使技能中配置了Preset Question，通过接口也无法触发。升级后，调接口时只需传文件链接参数便可以触发对该文件的Preset Question问答。

Python Code

Python Code 方式不依赖服务端，不依赖数据库，通过本地json文件方式，直接将技能load进程序，然后利用Bisheng 依赖实现技能的实例化。

- 由于官方没有发布bisheng的package包，因此无法通过pip 直接安装
- 因此在本地测试时，需要考虑测试文件和Bisheng 程序目录之间的关系

```

1 from bisheng import load_flow_from_json
2
3 def _test_python_code():
4     TWEAKS = {
5         'PyPDFLoader-RJlDA': {},
6         'InputFileNode-hikjJ': {
7             'file_path':
8                 'oss 文件地址' # noqa
9         },
10        'InputNode-keWk3': {},
11        'Milvus-VzZtx': {},
12        'RetrievalQA-Y4e1R': {},
13        'CombineDocsChain-qrRE4': {},
14        'RecursiveCharacterTextSplitter-C6YSc': {},
15        'ProxyChatLLM-oWqpn': {
16            'elemapi_key':
17                'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJiaXNoZW5hX29wZW42IiwiaXhwIjoxNzEzZzU0MjUyYyQ.ww1l-GTBYJiHV3-U1JcacvW0qYPd-QMpuJIeu09_OM8' # noqa
18        },
19        'HostEmbeddings-EJq6w': {}
20    }
21    flow = load_flow_from_json('/Users/Downloads/🌟 PDF文档摘要-zxf.json',
22                               tweaks=TWEAKS)
23    # Now you can use it like any chain
24    inputs = {'query': '合同甲方是谁'}
25    print(flow(inputs))
26
27 _test_python_code()

```

示例中，

1. 我们将 OSS 文件地址通过tweak 对象传入InputFileNode这个组件。
2. 通过load_flow_from_json 获得可执行对象flow
3. Inputs 是技能输入，可以通过 flow(inputs) 直接执行



通过api和python code 都是使用flow的同步请求，因此不支持stream等用法。

QA

1. 怎么上传文件
 - a. 当前的上传，是通过接口上传到bisheng临时目录，获得临时目录地址

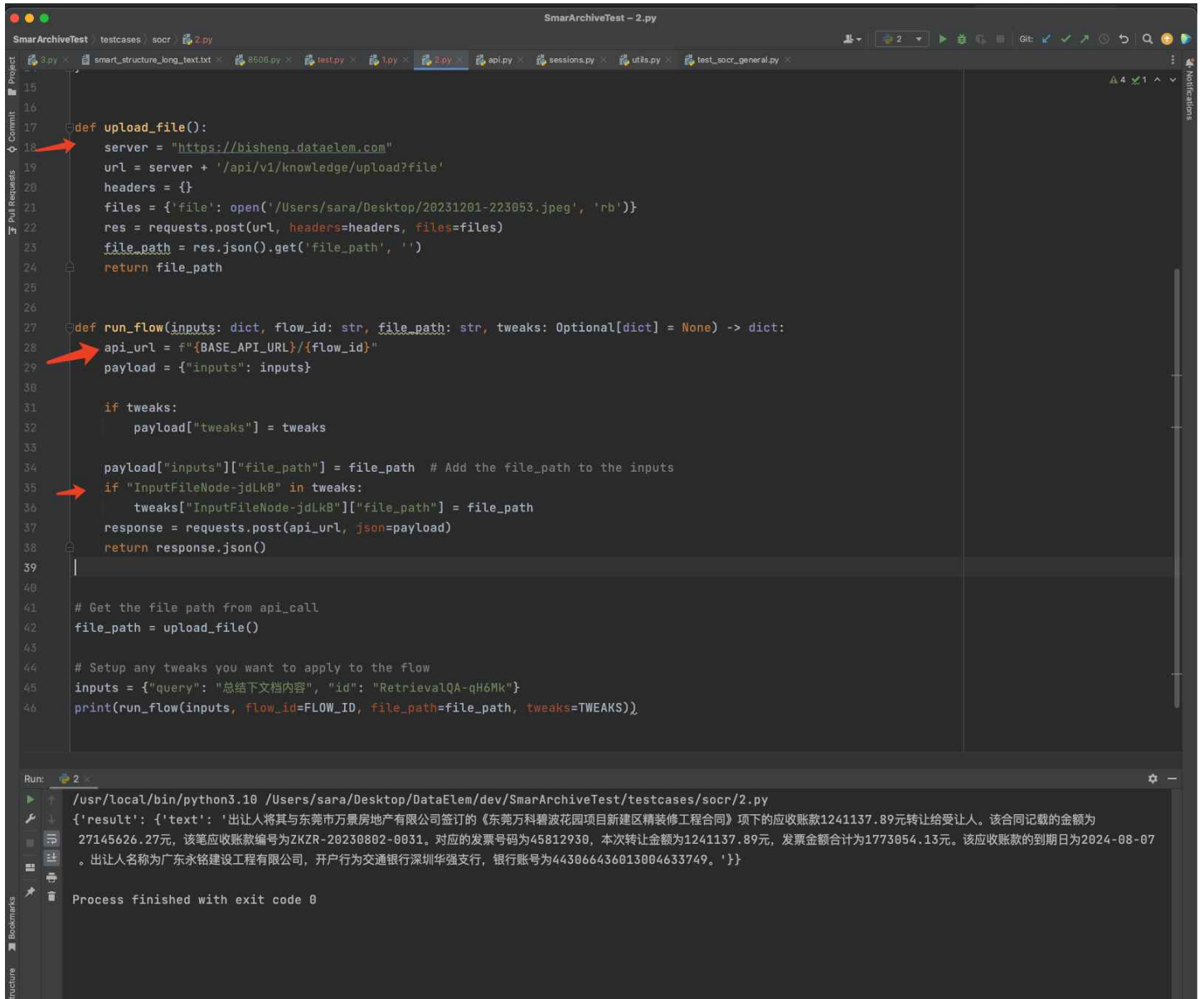
i. 接口api: /knowledge/upload

ii. 将接口返回的地址（或者直接oss地址）直接通过tweak 参数传入，以下提供一个示例

```
1 import requests
2 from typing import Optional
3
4 BASE_API_URL = "http://ip:port/api/v1/process"
5 FLOW_ID = "e4917a84-fad7-4d5d-85fd-bf2ec3c42c26"
6 TWEAKS = {
7     "CombineDocsChain-Pud2p": {},
8     "RetrievalQA-qH6Mk": {},
9     "OpenAIProxyEmbedding-yvld7": {},
10    "PyPDFLoader-SIeEa": {},
11    "InputFileNode-jdLkB": {},
12    "Milvus-T3kRH": {"collection_name": "col_333"},
13    "ChatOpenAI-qVQL6": {}
14 }
15
16
17 def upload_file():
18     server = "http://ip:port"
19     url = server + '/api/v1/knowledge/upload'
20     headers = {}
21     files = {'file': open('/Users/sara/Desktop/20231201-223053.jpeg', 'rb')}
22     res = requests.post(url, headers=headers, files=files)
23     file_path = res.json()['data'].get('file_path', '')
24     return file_path
25
26
27 def run_flow(inputs: dict, flow_id: str, file_path: str, tweaks: Optional[dict]
    = None) -> dict:
28     api_url = f"{BASE_API_URL}/{flow_id}"
29     payload = {"inputs": inputs}
30
31     if tweaks:
32         payload["tweaks"] = tweaks
33
34     payload["inputs"]["file_path"] = file_path # Add the file_path to the
    inputs
35
36     if "InputFileNode-jdLkB" in tweaks:
37         tweaks["InputFileNode-jdLkB"]["file_path"] = file_path
38     response = requests.post(api_url, json=payload)
39     return response.json()
40
41
```

```
42 # Get the file path from api_call
43 file_path = upload_file()
44
45 # Setup any tweaks you want to apply to the flow
46 inputs = {"query": "总结下文档内容", "id": "RetrievalQA-qH6Mk"}
47 print(run_flow(inputs, flow_id=FLOW_ID, file_path=file_path, tweaks=TWEAKS))
```

代码执行效果



```
SmarArchiveTest - 2.py
SmarArchiveTest testcases socr 2.py
15
16
17 def upload_file():
18     server = "https://bisheng.dataelem.com"
19     url = server + '/api/v1/knowledge/upload?file'
20     headers = {}
21     files = {'file': open('/Users/sara/Desktop/20231201-223053.jpeg', 'rb')}
22     res = requests.post(url, headers=headers, files=files)
23     file_path = res.json().get('file_path', '')
24     return file_path
25
26
27 def run_flow(inputs: dict, flow_id: str, file_path: str, tweaks: Optional[dict] = None) -> dict:
28     api_url = f"{BASE_API_URL}/{flow_id}"
29     payload = {"inputs": inputs}
30
31     if tweaks:
32         payload["tweaks"] = tweaks
33
34     payload["inputs"]["file_path"] = file_path # Add the file_path to the inputs
35     if "InputFileNode-jdLk8" in tweaks:
36         tweaks["InputFileNode-jdLk8"]["file_path"] = file_path
37     response = requests.post(api_url, json=payload)
38     return response.json()
39
40
41 # Get the file path from api_call
42 file_path = upload_file()
43
44 # Setup any tweaks you want to apply to the flow
45 inputs = {"query": "总结下文档内容", "id": "RetrievalQA-qH6Mk"}
46 print(run_flow(inputs, flow_id=FLOW_ID, file_path=file_path, tweaks=TWEAKS))
47
Run: 2
/usr/local/bin/python3.10 /Users/sara/Desktop/DataElem/dev/SmarArchiveTest/testcases/socr/2.py
{'result': {'text': '出让人将其与东莞市万景房地产有限公司签订的《东莞万科碧波花园项目新建区精装修工程合同》项下的应收账款1241137.89元转让给受让人。该合同记载的金额为27145626.27元, 该笔应收账款编号为ZKZR-20230802-0031, 对应的发票号码为45812930, 本次转让金额为1241137.89元, 发票金额合计为1773054.13元。该应收账款的到期日为2024-08-07。出让人名称为广东永铭建设工程有限公司, 开户行为交通银行深圳华强支行, 银行账号为443066436013004633749。'}}
Process finished with exit code 0
```

2. Python Code 支持报告内置问题吗?

- a. 不支持, 需要手动通过input 将预置问题挨个请求

知识库

文件上传

接口地址：<http://ip:port/api/v1/knowledge/upload>

请求方式：**POST**

返回格式：JSON

返回内容：file_url

参数	必选	类型及范围	说明
file	true	Form-data	需要通过form传参

知识库创建

接口地址：<http://ip:port/api/v2/filelib/>

请求方式：**POST**

返回格式：JSON

参数	必选	类型及范围	说明
name	true	string	知识库名称
model	true	string	embedding模型，跟系统配置有关，请确保embedding 模型在系统配置中正确配置
description	false	string	

知识库修改

接口地址：<http://ip:port/api/v2/filelib/>

请求方式：**PUT**

返回格式：JSON

参数	必选	类型及范围	说明
name	true	string	知识库名称
model	true	string	embedding模型，跟系统配置有关，请确保修改的embedding 模型在系统配置中正确配置
description	false	string	知识库描述

知识库删除

接口地址：http://ip:port/api/v2/filelib/{knowledge_id}

请求方式：**DELETE**

返回格式：JSON

获取知识库列表

接口地址：<http://ip:port/api/v2/filelib/>

请求方式：**GET**

返回格式：JSON

参数	必选	类型及范围	说明
page_size	true	int	从1 开始
page_num	true	int	

针对milvus和es进行临时文件清理

接口地址：http://ip:port/api/v2/filelib/chunk_clear

请求方式：**POST**

返回格式：JSON

参数	必选	类型及范围	说明
collection_name	false	str	明确知道需要清理milvus到哪个collection，对于非tmp开始的collection_name 慎用
index_name	false	str	明确知道需要清理es的哪个index，对于非tmp开始的index_name 慎用
flow_id	false	str	当前技能下没有进入回话列表的临时库，在技能编排测试的时候会生成
chat_id	false	str	通过应用界面使用，产生的临时表，api 调用产生的则传入对应sessionId 值。

知识库文件

文件导入

接口地址：http://ip:port/api/v2/filelib/file/{knowledge_id}

请求方式：**POST**

返回格式：JSON

参数	必选	类型及范围	说明
file	true	file	文件
knowledge_id	true	int	URL 传入
callback_url	false	string	body 传入，异步回调参数： <div><pre>1 json={ 2 "file_name": "demo.txt", 3 "file_status": 1, #1=解析中, 2=解析成功, 3=解析失败 4 "file_id": 1, 5 "error_msg": "Exception:" 6 }</pre></div>

文件列表

接口地址：http://ip:port/api/v2/filelib/file/{knowledge_id}

请求方式：**GET**

返回格式：JSON

参数	必选	类型及范围	说明
page_size	true	int	从1 开始
page_num	true	int	
knowledge_id	true	int	对应知识库id

文件删除

接口地址：http://ip:port/api/v2/filelib/file/{file_id}

请求方式：**DELETE**

返回格式：JSON

文件批量删除

接口地址：http://ip:port/api/v2/filelib/delete_file

请求方式：POST

请求参数：list直接传

参数	类型	传参方式	含义	demo
file_id	list	post	文件id列表	requests.post(url,json=[1,2,3])

返回格式：JSON

聊天Chat

免密websocket接口

现有接口是websoket方式

ws://ip:port/api/v2/chat/ws/{flow_id}

参数	类型	传参方法	参数含义	注意
chat_id	string	get	前端随机id	用来表示一个会话窗口
flow_id	string	Url 传参	技能id，对应已经上线了的技能	
knowledge_id	int	知识库id	知识库id	
tweak	json	Api 组件传参	针对特定组件进行参数设置 0.2.2.3 版本支持	需要urlencode Json对象

使用websocket 返回客户端消息

返回参数key	类型	含义	
is_bot	bool	必要，true消息在左边，false 消息在右边	
message	str/ json	返回的消息最终结果	

type	str	begin/close 表示整轮会话的开始和结束， start/stream/end 表示stream的开始和结束	
category	str	processing: 暂无特别实用 question/answer/report/system: 对应前端不同的展示样式	
user_id	int	当前用户id	
message_id	int	消息id	
source	bool	是否支持溯源， 0 不支持， 1 支持， 2有权限限制， 3 问答库， 4 QA库	
flow_id	str	（必传）当前聊天的技能id	
chat_id	str	（必传）当前会话id	

当我们完整构建一条消息，需要后端给前端发送4条消息：

type=begin, category=system

type=start, category=system

type=end, message=""

type=close, category=system

点赞

接口地址：<http://ip:port/api/v2/chat/liked>

接口请求方式： POST

返回格式： json

参数	参数含义	注意
message_id		
liked	0 初始值， 1 赞， 2 踩	

获取点赞数

接口地址：http://ip:port/api/v2/data/msg_like

接口请求方式： GET

返回格式： json

参数	含义	是否必填
liked	用户是否喜欢 0未评价/1 喜欢/2 不喜欢	是
flow_id	技能id	是
create_time	开始时间 '%Y-%m-%d %H:%M:%S'	否
create_time_end	结束时间 '%Y-%m-%d %H:%M:%S'	否

点踩后反馈建议

接口地址：<http://ip:port/api/v2/chat/comment>

接口请求方式： POST

返回格式： json

参数	参数含义	注意
message_id		
comment	评论的内容	不超过4096 大小

已解决/未解决

接口地址：<http://ip:port/api/v2/chat/solved>

接口请求方式： POST

返回格式： json

参数	参数含义	注意
chat_id		
solved	0 初始值， 1 解决, 2 未解决	

API 溯源接口

```
1
2 # 第一步执行api访问
3 def _test_python_api():
4     url = "http://127.0.0.1:3001/api/v1/process/c33c50dd-30ac-4e9b-a2c0-fd13c4bce245"
```



```

5     body_json = {
6         "inputs": {
7             "question": "SQLAgent 是什么",
8             "id": "ConversationalRetrievalChain-bZmFo"
9         },
10        "tweaks": {
11            "ConversationalRetrievalChain-bZmFo": {},
12            "ChatOpenAI-Y0pJc": {},
13            "ConversationBufferMemory-VTj8f": {},
14            "Milvus-NhYXG": {}
15        }
16    }
17    resp = requests.post(url, json=body_json)
18    # 返回实例
19    # result 对象中, source != 0 表示可以溯源
20    # 通过 message_id 查询溯源的数据
21    # {
22    #     "result": {
23    #         "answer":
24    #             "SQLAgent 是 Microsoft SQL Server 中的一个组件, 它是一个作业调度程序,
25    #             用于自动执行和管理 SQL Server 数据库中的作业和任务。SQLAgent 可以通过计划和配置来执行诸
26    #             如备份、恢复、数据清理和索引重建等操作。它还可以设置警报和通知, 以便管理员能够及时了解数据
27    #             库中的问题和事件。",
28    #         "source": 1,
29    #         "message_id": 30522
30    #     },
31    #     "session_id":
32    #         "A61GFF:538546578b93520116fe329d8df03e9805354919b50c8a946a0b47fa565ee350",
33    #     "backend": "anyio"
34    # }
35    # 0: 不溯源
36    # 1: 普通溯源
37    # 2: 有访问权限限制溯源
38    # 3: 知识库外部链接溯源
39    # 4: QA溯源
40
41    # 第二步, 获取溯源内容
42    def get_source(message_id):
43        # keywords 是表示当前答案, 有哪些关键字, 非必要
44        keywords_url = f"http://127.0.0.1:3001/api/v1/qa/keyword?message_id={message_id}"
45        keyword_resp = requests.get(keywords_url)
46
47        # chunk 获取接口, 当我们需要根据关键字进行排序, 需要将上一个关键字的结果, 拼接到
48        # keys 参数下。 ";" 分割
49        chunk_url = f"http://127.0.0.1:3001/api/v1/qa/chunk?message_id={message_id}&keys="

```

```

45     chunks = requests.get(chunk_url)
46     print(chunks)
47     #
48     # {
49 #     "data": [{
50 #         "chunk_bboxes": [{
51 #             "page": 1,
52 #             "bbox": [90, 74, 226, 85]
53 #         }], # 用来定位文本位置
54 #         "right": true,
55 #         # 双层pdf 路径
56 #         "source_url":
57 #         "http://110.16.193.170:50061/bisheng/10832?X-Amz-Algorithm=AWS4-HMAC-
        SHA256&X-Amz-Credential=minioadmin%2F20231228%2Fus-east-
        1%2Fs3%2Faws4_request&X-Amz-Date=20231228T060923Z&X-Amz-Expires=604800&X-Amz-
        SignedHeaders=host&X-Amz-
        Signature=053b4f399bbf3bdf3f4c86f750419bfde3739b4a2eee3e5490c4b63030ce66e5",
58 #         # 原始文件路径
59 #         "original_url":
60 #         "http://110.16.193.170:50061/bisheng/original/10832?X-Amz-
        Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=minioadmin%2F20231228%2Fus-east-
        1%2Fs3%2Faws4_request&X-Amz-Date=20231228T060923Z&X-Amz-Expires=604800&X-Amz-
        SignedHeaders=host&X-Amz-
        Signature=5ad4e49c5ff47aff425f472dbe21b11ce6d3ab65964292f61d1412a0ec07a3a6",
61 #         "score": 0,
62 #         "file_id": 10832,
63 #         "source": "多少专利和知识产权.pdf"
64 #     }],
65 #     "msg":
66 #     "success"
67 # }

```

特殊文档上传（同步方法）

使用场景见 [📖 RetrievalChain](#)

QA库内容上传 - 文件方式导入

内容	说明
文件	文件内容就是 问题文本
文件名	问题文本

Answer	答案文本
URL	锚点URL，知识库文章链接拼上这个QA的id

接口地址：<http://ip:port/api/v2/filelib/chunks>

接口请求方式： POST

返回格式： json

参数：

参数	是否必传	参数含义	注意
knowledge_id	是	Int 值传入知识库ID	349 测试传参
file	是	formData 传入文件	
metadata	是	对文件进行补充的metadata 采用string传入json类型	
<ul style="list-style-type: none">metadata.answer	否	QA 场景的答案， QA场景必填	
<ul style="list-style-type: none">metadata.url	否	锚点URL，知识库文章链接拼上这个QA的id	

返回参数

参数	是否必传	参数含义	注意
status_code	是	200	非200 为异常
status_message	是	success	
data	Json		
<ul style="list-style-type: none">id	int	file_id, 用来标识系统文件唯一id	
<ul style="list-style-type: none">status	int	上传状态	
<ul style="list-style-type: none">remark	str	错误原因	

Postman 示例：

	参数名		参数值	
⋮	file	String * ▾	请选择上传文件	File ▾
⋮	knowledge_id	String * ▾	349	Text ▾
⋮	metadata	String * ▾	{"url":"baidu.com","answer":""}	Text ▾

代码示例：

```
1 def test_file():
2     url = 'http://192.168.106.116:7860/api/v2/filelib/chunks'
3     data = {'knowledge_id': 349, 'metadata': "{\"url\":\"http://baidu.com\"}"}
4     file = {'file': open('/Users/huangly/Downloads/co2.pdf', 'rb')}
5
6     resp = requests.post(url=url, data=data, files=file)
7     return resp
```

QA库内容上传-string 导入

接口地址：http://ip:port/api/v2/filelib/chunks_string

接口请求方式：POST

返回格式：json

参数：

参数	是否必传	参数含义	注意
knowledge_id	是	POST body	349 测试传参
documents	是	文件List	
• page_content	是	文件内容	
• metadata	是	补充信息	
-- metadata.answer	否	QA 场景的答案，QA场景必填	
-- metadata.url	否	锚点URL，知识库文章链接拼上这个QA的id	
-- source	是	文件名	

返回参数：

--	--	--	--

参数	是否必传	参数含义	注意
status_code	是	200	非200 为异常
status_message	是	success	
data	Json		
<ul style="list-style-type: none">id	int	file_id, 用来标识系统文件唯一id	
<ul style="list-style-type: none">status	int	上传状态	
<ul style="list-style-type: none">remark	str	错误原因	

代码示例：

```
1 #demo
2 json_data = {
3     "knowledge_id": 371,
4     "documents": [
5         {
6             "page_content": "达梦有多少专利和知识产权？",
7             "metadata": {
8                 "source": "达梦有多少专利和知识产权？.txt",
9                 "url": "",
10                "answer": "达梦共有177个已获授权专利情况，293个软件著作权情况",
11                "page": 1
12            }
13        }
14    ]
15 }
16
17 url= 'http://192.168.106.116:7860/api/v2/filelib/chunks_string'
18 resp = requests.post(url=url, json=json_data)
19 resp
```

QA 溯源信息接口

接口地址：<http://ip:port/api/v2/chat/source>

接口请求方式：GET

返回格式：json

参数

--	--	--

参数	参数含义	注意
message_id	每条消息都会返回messageId	

返回参数

参数	参数含义	注意
id	溯源id	
chunk	文本块(QA 对应问题)	
meta_data	dict， 对应文本愿信息	
metadata.url	溯源信息	
metadata.answer		
metadata.title	对应文title	