拡張対話チュートリアル
はじめに

本書は、本サービスが提供する拡張対話APIによって対話機能を実現するクライアントアプリケーションを作成するためのチュートリアルです。

本書の対象読者は以下を想定しています。

- 本サービスと連携したシステムや製品開発を行う開発者

なお、本書ではPythonのサンプルコードを記載しておりますが、本サービスのAPIはREST形式のため、他の言語からもご利用いただけます。
サポートしているAPIとリクエスト・レスポンスの詳細については、APIリファレンスを参照してください。

拡張対話APIとは

以下の3つの機能を備えたLLMを使った対話機能を利用するAPIです。

- 拡張対話（ファイル添付）機能
- 拡張対話（Webコンテンツ）機能
- 拡張対話（Web検索）機能

ファイルに記載されている内容、URL参照先の内容、WEB検索結果の内容に対して質問などを行いたい場合に使用します。
なお、特にファイルやURL、検索キーワードを指定しない場合は、履歴付き対話と同等の機能になります。

機能仕様

各機能の仕様は以下の通りです。

拡張対話（ファイル添付）機能仕様

選択可能なファイルの種類は以下となります。

1. ドキュメント系ファイル
   ・テキストファイル（.txt、文字コードはUTF-8のみ利用可能）
   ・PDF ファイル（.pdf）
   ・Microsoft Word ファイル（.docx）
   ・Microsoft Excel ファイル（.xlsx、.xls）
   ・Microsoft PowerPoint ファイル（.pptx）
2. 画像系ファイル
   ・.jpg
   ・.jpeg
   ・.png
   ・.webp
   ・.gif（非アニメーション）

以下のモデルは画像入力に対応していないためご注意ください。
・cotomi-fast-v2.0
・cotomi-pro-v2.0
・cotomi-v3.0

読み込み可能なPDFは以下となります。
・PDF-1.7
・中国語、日本語、韓国語
・Type1、TrueType、Type3、CIDのフォントタイプ
・RC4、AES暗号化

ドキュメント系ファイル内に含まれる図、画像、動画は対象外となります。

送信可能なドキュメント系ファイル容量の上限は30MBです。30MBより大きい値のファイルの添付はご遠慮ください。

一度のリクエストで送信可能な画像系ファイルの枚数は50枚まで、容量の上限はbase64encode後のサイズ換算で合計1.6MB、画像ファイルのサイズ換算で合計1.2MBです。
それ以上のサイズの画像を添付した場合にはエラーになる可能性があります。

読み込み可能な文字数の上限値はありません。LLMのトークン数上限を超えない範囲でご利用ください。

空のファイルの指定はエラーが発生するためご遠慮ください。

複数のファイルを用いた対話を実施する場合はLLMのトークン数上限を超えない範囲でご利用ください。

拡張対話（Webコンテンツ）機能仕様

不正アクセスに該当するようなURL（IPアドレス形式、IPv6形式など）の指定はご遠慮ください。

URLはhttp、httpsに対応しており、参照先がHTML、テキストファイル、jsonファイルの場合読み込みが可能です。

読み込み可能なURLでエラーが発生した場合はブラウザで表示可能かどうかを確認してください。

URLの参照先に含まれる図、画像、動画は対象外となります。

以下のようなパラメータを含むURLの場合、パラメータの値を読み込むことはできません。

```
https://example.com?param=abc
```

URLに特殊文字が含まれる場合は適宜URLエンコードを実施してください。

URLの文字列長の上限は4096文字とします。

読み込み可能な文字数の上限値はありません。LLMのトークン数上限を超えない範囲でご利用ください。

参照先のサイトの文字コードはUTF-8、Shift-JIS、EUC-KR、GB2312、ISO-8859-1に対応しています。

複数のURLを用いた対話を実施する場合はLLMのトークン数上限を超えない範囲でご利用ください。

拡張対話（Web検索）機能仕様

検索キーワードは半角、もしくは全角スペース区切りで入力し、合計400文字以内、50単語以内で入力してください。

検索キーワードには半角スペース、全角スペースのみの入力は禁止とします。

Web検索結果は10000文字（約5000トークン）を上限としてLLMに入力されます。対応可能なLLMを指定してください（cotomi v3は対応しています）。指定したLLMによっては、トークン超過エラーが発生する可能性があるためご注意ください。

検索エンジンサービスには分間リクエスト上限と契約に応じた月単位でのリクエスト上限が設定されている場合があります。それぞれのリクエスト上限を超えた場合エラーが発生するためご注意ください。詳細は契約している検索エンジンのページをご確認ください。

拡張対話（Web検索）機能はWeb検索サービス処理に時間がかかるため、履歴付き対話などの他の対話機能と比べて応答までに時間がかかります。同時に大量のリクエストがあった場合、Web検索自体の負荷が高くなり処理時間が長くなる可能性があります。

チュートリアルの流れ

本チュートリアルの流れは以下です。

1. 拡張対話（ファイル添付）機能
    a. 拡張対話（ファイル添付）機能を利用する
    b. (応用)過去の会話履歴を踏まえて会話する
    c. (応用)requestsライブラリを利用したpythonプログラムで拡張対話（ファイル添付）機能を利用する

2. 拡張対話（Webコンテンツ）機能
   a. 拡張対話（Webコンテンツ）機能を利用する
   b. 拡張対話（Webコンテンツ）機能で複数のURLを用いて会話する
   c. (応用)requestsライブラリを利用したpythonプログラムで拡張対話（Webコンテンツ）機能を利用する
3. 拡張対話（Web検索）機能
   a. 拡張対話（Web検索）機能を利用する
   b. (応用)requestsライブラリを利用したpythonプログラムで拡張対話（Web検索）機能を利用する

拡張対話（ファイル添付）機能

ここでは、拡張対話（ファイル添付）機能を用いた対話について説明します。
拡張対話（ファイル添付）機能はファイル名とファイル内容をbase64エンコードしたデータを用いることで、ファイルに関する対話を実施することができる機能です。
ファイル名についた拡張子でファイルの種類を判断します。

拡張対話（ファイル添付）機能を利用する

今回は “合言葉は「馬の耳に念仏」です。” と記載された「keyword.txt」ファイルを用いて拡張対話（ファイル添付）機能を利用します。
以下のcurlコマンドを用いて、APIを呼び出します。APIの詳細は「Generative AI Cloud（SaaS）APIリファレンス」を参照してください。

```
curl -X POST https://api.genai-api.nec-cloud.com/genai-api/v1/exchat ¥
-H "Content-Type: application/json" ¥
-d "<リクエストパラメータ>" ¥
-H "Authorization: <API Key>" ¥
-H "x-nec-genai-client-id: <ユーザId>"
```

以下はリクエストパラメータの例です。addContent.contentパラメータには「keyword.txt」をBase64Encodeした結果を付与してください。

```
{
    "userContent": "合言葉は何ですか？",
    "systemContent": "日本語で回答してください。",
    "model": "cotomi-v3.0",
    "historyId": "new",
    "addContent": {
        "type": "file",
        "name": "keyword.txt",
        "content": "5ZCI6KiA6JGJ44Gv44CM6aas44Gu6ICz44Gr5b+15LuP44CN44Gn44GZ44CC"
    }
}
```

応答がJson形式で返却されます。sourceDocumentsには「keyword.txt」に関する情報が返却されます。

```
{"answer":"合言葉は「馬の耳に念仏」です。",
 "historyId":"historyid_12345678-90ab-cdef-ghij-zzz",
 "sourceDocuments":[{
  "docContent":"合言葉は「馬の耳に念仏」です。",
  "metadata":{"type":"file","fileName":"keyword.txt","overReadLimit":false}}]}
```

(応用)過去の対話履歴を踏まえて対話する

先ほどの応答で得られたhistoryIdを用いて、追加の質問を実施します。
以下のcurlコマンドを用いて、APIを呼び出します。

```
curl -X POST https://api.genai-api.nec-cloud.com/genai-api/v1/exchat ¥
-H "Content-Type: application/json" ¥
-d "<リクエストパラメータ>" ¥
-H "Authorization: <API Key>" ¥
-H "x-nec-genai-client-id: <ユーザId>"
```

以下はリクエストパラメータの例です。先ほどの応答で得られたhistoryIdをhistoryIdパラメータのキーに指定してください。

```
{
    "userContent": "合言葉を変更したいです。前回の合言葉を踏まえた上でおすすめを教えてください。",
    "systemContent": "日本語で回答してください。",
    "model": "cotomi-v3.0",
    "historyId": "historyid_12345678-90ab-cdef-ghij-zzz"
}
```

応答がJson形式で返却されます。

```
{"answer":"新しい合言葉として、前回の「馬の耳に念仏」を踏まえて、以下のような言葉は...",
```

```
"historyId":"historyid_12345678-90ab-cdef-ghij-zzz","sourceDocuments":[]}
```

(応用)requestsライブラリを利用したpythonプログラムで拡張対話（ファイル添付）機能を利用する

requestsライブラリのインストール

インストールコマンドの例

```
pip install requests
```

Pythonコードを実装する

Pythonコードの例

Pythonのエディタを開き、下記のコードを記述します。ファイル名を「extended_file.py」として保存します。

```python
import os
import sys
from base64 import b64encode
import requests

# KEYはAPIキー
KEY = 'abcdefg1234567890'
# BASEはサービスのURL
BASE = 'https://api.genai-api.nec-cloud.com/genai-api/v1'
# MODELはエイリアス名
MODEL = 'cotomi-v3.0'

# 引数でファイルを受け取る
file_path = sys.argv[1]

def func():
    file_name = os.path.basename(file_path)
    base64 = load_base64(file_path)

    url = BASE + '/exchat'
    key = 'Bearer ' + KEY
    payload = {
        "userContent": "{}の内容を教えてください。".format(file_name),
        "model": MODEL,
        "historyId": "new",
        "addContent": {
            "type": 'file',
            "name": file_name,
            "content": base64
        }
    }
    headers = { 'content-type': 'application/json',
                'x-nec-genai-client-id': 'ABCDEF',
                'Authorization': key }
    # 非ストリーム形式
    response = requests.post(url, json=payload, headers=headers)
    print(response.status_code)
    print(response.text)
    #print(response.headers)

def load_base64(path):
    with open(path, 'rb') as file:
        binary = file.read()
        base64 = b64encode(binary).decode()
    return base64

if __name__ == '__main__':
    func()

```

Pythonの実行

```
python extended_file.py <任意のファイルパス>
```

実行結果の例

```
200
{"answer": "以下に、の`test.pptx`の...",
 "historyId": "historyid_12345678-90ab-cdef-ghij-zzz",
 "sourceDocuments": [{
  "docContent": "¥n¥n人間と動物の関係は、...",
  "metadata": {
   "type": "file","fileName": "test.pptx","overReadLimit": false}}]}
```

拡張対話（Webコンテンツ）機能

ここでは、拡張対話（Webコンテンツ）機能を用いた対話について説明します。
指定されたURLページ内のテキストを取得し、テキストに関する対話を実施することができる機能です。

拡張対話（Webコンテンツ）機能を利用する

今回は、日本電気株式会社のwikipediaのURLを用いて拡張対話（Webコンテンツ）機能を利用します。
以下のcurlコマンドを用いて、APIを呼び出します。APIの詳細は「Generative AI Cloud（SaaS）APIリファレンス」を参照してください。

```
curl -X POST https://api.genai-api.nec-cloud.com/genai-api/v1/exchat ¥
-H "Content-Type: application/json" ¥
-d "<リクエストパラメータ>" ¥
-H "Authorization: <API Key>" ¥
-H "x-nec-genai-client-id: <ユーザId>"
```

以下はリクエストパラメータの例です。URLエンコードは必要に応じて実施してください。今回は未実施で実行しています。

```
{
    "userContent": "URLの内容を要約してください",
    "systemContent": "日本語で回答してください。",
    "historyId": "new",
    "model": "cotomi-v3.0",
    "addContent": {
        "type": "url",
        "content": "https://ja.wikipedia.org/wiki/日本電気"
    }
}
```

応答がJson形式で返却されます。sourceDocumentsには付与したURLに関する情報が返却されます。

```
{"answer":"日本電気株式会社（NEC）は、...",
 "sourceDocuments":[{"docContent":"日本電気 - Wikipedia...",...}]}
```

拡張対話（Webコンテンツ）機能で複数のURLを用いて会話する

日本電気株式会社のwikipediaとPC-8000シリーズのwikipediaのURLを用いて拡張対話（Webコンテンツ）機能を利用します。
以下のcurlコマンドを用いて、APIを呼び出します。APIの詳細は「Generative AI Cloud（SaaS）APIリファレンス」を参照してください。

```
curl -X POST https://api.genai-api.nec-cloud.com/genai-api/v1/exchat ¥
-H "Content-Type: application/json" ¥
-d "<リクエストパラメータ>" ¥
-H "Authorization: <API Key>" ¥
-H "x-nec-genai-client-id: <ユーザId>"
```

以下はリクエストパラメータの例です。URLエンコードは必要に応じて実施してください。今回はURLエンコードありで実行しています。

```
{
    "userContent": "2つのURLの内容を要約してください",
    "systemContent": "あなたはAIアシスタントです。質問には日本語で回答してください。",
    "historyId": "new",
    "model": "cotomi-v3.0",
    "stream": false,
    "addContents": [{
        "type": "url",
        "content": "https://ja.wikipedia.org/wiki/%E6%97%A5%E6%9C%AC%E9%9B%BB%E6%B0%97"
    },
    {
        "type": "url",
        "content": "https://ja.wikipedia.org/wiki/PC-8000%E3%82%B7%E3%83%AA%E3%83%BC%E3%82%BA"
    }]
}
```

応答がJson形式で返却されます。sourceDocumentsには付与したURLに関する情報が返却されます。

```
{
    "answer":"### 日本電気 - Wikipediaの要約¥n¥n日本電気株式会社（NEC）は、東京都港区に本社を置く住友グループの電機メーカーです。...",
    "historyId":"history_96fa2fdf-0d85-4956-a3d9-c1b83c28d856",
    "sourceDocuments":[{
        "docContent":"日本電気 - Wikipedia...",
        "metadata":{
            "type":"url",
            "url":"https://ja.wikipedia.org/wiki/%E6%97%A5%E6%9C%AC%E9%9B%BB%E6%B0%97",
            "urlTitle":"日本電気 - Wikipedia",
            "overReadLimit":true
        }
    },{
        "docContent":"PC-8000シリーズ - Wikipedia...",
        "metadata":{
            "type":"url",
            "url":"https://ja.wikipedia.org/wiki/PC-8000%E3%82%B7%E3%83%AA%E3%83%BC%E3%82%BA",
```

```
            "urlTitle":"PC-8000シリーズ - Wikipedia",
            "overReadLimit":true
        }
    }]
}
```

(応用)requestsライブラリを利用したpythonプログラムで拡張対話（Webコンテンツ）機能を利用する

requestsライブラリのインストール

インストールコマンドの例

```
pip install requests
```

Pythonコードを実装する

Pythonコードの例

Pythonのエディタを開き、下記のコードを記述します。ファイル名を「extended_url.py」として保存します。

```
import requests

# KEYはAPIキー
KEY = 'abcdefg1234567890'
# BASEはサービスのURL
BASE = 'https://api.genai-api.nec-cloud.com/genai-api/v1'
# MODELはエイリアス名
MODEL = 'cotomi-v3.0'

def func():
    url = BASE + '/exchat'
    key = 'Bearer ' + KEY
    payload = {
        "userContent": "URLの内容を要約してください",
        "model": MODEL,
        "historyId": "new",
        "addContent": {
            "type": 'url',
            "content": 'https://ja.wikipedia.org/wiki/%E6%97%A5%E6%9C%AC%E9%9B%BB%E6%B0%97',
        }
    }
    headers = { 'content-type': 'application/json',
                'x-nec-genai-client-id': 'ABCDEF',
                'Authorization': key }
    # 非ストリーム形式
    response = requests.post(url, json=payload, headers=headers)
    print(response.status_code)
    print(response.text)
    #print(response.headers)

if __name__ == '__main__':
    func()

```

Pythonの実行

```
python extended_url.py
```

実行結果の例

```
200
{"answer": "日本電気株式会社（NEC）は、...",
 "historyId": "historyid_12345678-90ab-cdef-ghij-zzz",
 "sourceDocuments": [{
  "docContent": "日本電気 - Wikipediaコンテンツに...",
  "metadata": {
   "type": "url",
   "url": "https://ja.wikipedia.org/wiki/%E6%97%A5%E6%9C%AC%E9%9B%BB%E6%B0%97",
   "urlTitle": "日本電気 - Wikipedia",
   "overReadLimit": true}}]}
```

拡張対話（Web検索）機能

ここでは、拡張対話（Web検索）機能を用いた対話について説明します。
検索キーワードをスペース区切りで入力し、検索キーワードに関する対話を実施することができる機能です。

> ⚠ Web検索機能を利用する際は、必ず管理ポータルで使用する検索エンジンサービスの種類とAPIキーが設定されていることを確認してください。
> 設定されていない場合は「管理ポータル操作ガイド（設定画面編）」を参照して設定手順を実施してください。

拡張対話（Web検索）機能を利用する

今回は、日本の世界遺産についての説明を求めるリクエストを用いて拡張対話（Web検索）機能を利用します。
以下のcurlコマンドを用いて、APIを呼び出します。APIの詳細は「Generative AI Cloud（SaaS）APIリファレンス」を参照してください。

```
curl -X POST https://api.genai-api.nec-cloud.com/genai-api/v1/exchat ¥
-H "Content-Type: application/json" ¥
-d "<リクエストパラメータ>" ¥
-H "Authorization: <API Key>" ¥
-H "x-nec-genai-client-id:<ユーザId>"
```

以下はリクエストパラメータの例です。

```
{
    "userContent": "検索結果を基に日本の世界遺産について説明してください。",
    "systemContent": "あなたはAIアシスタントです。質問には日本語で回答してください。",
    "historyId": "new",
    "model": "cotomi-v3.0",
    "stream": false,
    "addContents": [{
        "type": "web",
        "content": "日本 世界遺産"
    }]
}
```

応答がJson形式で返却されます。sourceDocumentsには検索結果のURL情報に関する情報が返却されます。

```
{"answer":"検索結果から得られた情報を基に、...",
 "sourceDocuments":[{"docContent":"日本の世界遺産一覧|文化庁文化庁の紹介政策について...",...}]}
```

(応用)requestsライブラリを利用したpythonプログラムで拡張対話（Web検索）機能を利用する

requestsライブラリのインストール

インストールコマンドの例

```
pip install requests
```

Pythonコードを実装する

Pythonコードの例

Pythonのエディタを開き、下記のコードを記述します。ファイル名を「extended_websearch.py」として保存します。

```
import requests

# KEYはAPIキー
KEY = 'abcdefg1234567890'
# BASEはサービスのURL
BASE = 'https://api.genai-api.nec-cloud.com/genai-api/v1'
# MODELはエイリアス名
MODEL = 'cotomi-v3.0'

def func():
    url = BASE + '/exchat'
    key = 'Bearer ' + KEY
    payload = {
        "userContent": "検索結果を基に日本の世界遺産について説明してください。",
        "model": MODEL,
        "historyId": "new",
        "addContent": {
            "type": 'web',
            "content": '日本 世界遺産',
        }
    }
    headers = { 'content-type': 'application/json',
                'x-nec-genai-client-id': 'ABCDEF',
                'Authorization': key }
    # 非ストリーム形式
    response = requests.post(url, json=payload, headers=headers)
    print(response.status_code)
    print(response.text)
    #print(response.headers)

if __name__ == '__main__':
    func()

```

Pythonの実行

```
python extended_websearch.py
```

実行結果の例

```
200
{”answer”: ”日本の世界遺産はユネスコによって現在26件登録...”,
 ”historyId”: ”historyid_12345678-90ab-cdef-ghij-zzz”,
 ”sourceDocuments”: [{
  ”docContent”: ”日本 の世界遺産一覧...”,
  ”metadata”: {
   ”type”: ”web”,
   ”content”: 日本 世界遺産”,
   ”results”: [{
    ”title”: ”日 本の世界遺産一覧 | 文化庁”,
    ”url”: ”https://www.bunka.go.jp/seisaku/bunkazai/shokai/sekai_isan/ichiran/”,
    ”summary”: ”<strong>世界遺産</strong>...”
    },{
    ...
    }]
   }
 }]
}
```