Restricted Individuals Consultation API — Technical Documentation

Restricted Individuals Querytion API

Documentação Técnica

Versão: 1.0 Data: 15/10/2025

■ Visão Geral

A Restricted Individuals Querytion API permite verificar se um Brazilian Individual Taxpayer Registry (CPF) possui impedimentos junto aos órgãos governamentais brasileiros. O serviço utiliza certificação digital e conexão direta com as bases oficiais do governo.

Informações Básicas

■ Authentication

A API utiliza autenticação via JWT (JSON Web Token). Você deve incluir o token em todas as requisições no header Authorization.

1. Obter Credenciais

Entre em contato com nossa equipe para receber:

Nome de usuário

Password inicial

Limit de requisições por minuto

2. Fazer Login

Endpoint: POST /auth/login

Request:

{

"username": "seu_usuario",

"password": "sua_senha"

}

Response (200 OK):

{

"access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",

"token_type": "bearer",

"expires_in": 3600

}

Example cURL:

curl -X POST https://stg.opaservices.com.br/auth/login \

-H "Content-Type: application/json" \

-d '{

"username": "seu_usuario",

"password": "sua_senha"

}'

3. Usar o Token

Inclua o token no header Authorization de todas as requisições:

Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...

■■ Importante:

Tokens expiram após 60 minutos

Armazene o token de forma segura

Renove o token antes de expirar

■ Endpoints

1. Queryr Brazilian Individual Taxpayer Registry (CPF)

Verifica se um Brazilian Individual Taxpayer Registry (CPF) possui impedimentos.

Endpoint: GET /consultar/{cpf}

Parâmetros:

Headers:

Authorization: Bearer {seu_token}

Content-Type: application/json

Response (200 OK):

{

"transaction_id": "550e8400-e29b-41d4-a716-446655440000",

"cpf": "12345678901",

"status": "LIBERADO",

"timestamp": "2025-10-16T14:30:00.123456",

"usuario": "seu_usuario"

}

Possíveis Status:

LIBERADO - Brazilian Individual Taxpayer Registry (CPF) sem impedimentos

IMPEDIDO - Brazilian Individual Taxpayer Registry (CPF) com impedimentos ativos

não encontrado - Brazilian Individual Taxpayer Registry (CPF) não consta na base

indefinido - Status não determinado

Example cURL:

curl -X GET http://172.171.218.109:8000/consultar/12345678901 \

-H "Authorization: Bearer seu_token_aqui"

Example Python:

import requests

```
url = "https://stg.opaservices.com.br/consultar/12345678901"
headers = {
"Authorization": "Bearer seu_token_aqui"
}
response = requests.get(url, headers=headers)
data = response.json()
print(f"Status: {data['status']}")
print(f"Transaction ID: {data['transaction_id']}")
```

2. Queryr Transaction

Recupera os detalhes de uma consulta anterior.

Endpoint: GET /transacao/{transaction_id}

Parâmetros:

Response (200 OK):

```
{
"transaction_id": "550e8400-e29b-41d4-a716-446655440000",
"usuario": "seu_usuario",
"cpf": "12345678901",
"status": "LIBERADO",
"timestamp": "2025-10-16T14:30:00.123456",
"tempo_resposta_ms": 234.56,
"ip_origem": "192.168.1.100"
}
```

Example cURL:

```
curl -X GET https://stg.opaservices.com.br/transacao/550e8400-e29b-41d4-a716-446655440000 \
-H "Authorization: Bearer seu_token_aqui"
```

3. History de Querys

Lista as últimas consultas realizadas pelo usuário.

Endpoint: GET /historico?limite=10

Query Parameters:

Response (200 OK):

```
{
"usuario": "seu_usuario",
"total_transacoes": 156,
"transacoes": [
{
```

"transaction_id": "550e8400-e29b-41d4-a716-446655440000",

"cpf": "12345678901",

"status": "LIBERADO",

"timestamp": "2025-10-16T14:30:00.123456",

"tempo_resposta_ms": 234.56

},

{

"transaction_id": "660e8400-e29b-41d4-a716-446655440001",

"cpf": "98765432109",

"status": "IMPEDIDO",

"timestamp": "2025-10-16T14:25:00.654321",

"tempo_resposta_ms": 189.12

}

]

}

Example cURL:

curl -X GET "https://stg.opaservices.com.br/historico?limite=20" \

-H "Authorization: Bearer seu_token_aqui"

4. Health Check

Verifica se a API está funcionando.

Endpoint: GET /health

Response (200 OK):

{

"status": "ok",

"database": "connected",

"timestamp": "2025-10-16T14:30:00.123456"

}

Example cURL:

curl -X GET https://stg.opaservices.com.br/health

■ Examples de Uso

Example Completo em Python

import requests

import json

from datetime import datetime, timedelta

class APIImpedidos:

def __init__(self, base_url, username, password):

```
self.base_url = base_url
self.username = username
self.password = password
self.token = None
self.token_expira = None
def login(self):
"""Realiza login e obtém token JWT"""
url = f"{self.base_url}/auth/login"
payload = {
"username": self.username,
"password": self.password
}
response = requests.post(url, json=payload)
if response.status_code == 200:
data = response.json()
self.token = data["access_token"]
self.token_expira = datetime.now() + timedelta(seconds=data["expires_in"])
print("■ Login realizado com sucesso!")
return True
else:
print(f"■ Error no login: {response.status_code}")
return False
def _verificar_token(self):
"""Verifica se o token ainda é válido"""
if not self.token or datetime.now() >= self.token_expira:
print("■■ Token expirado, fazendo novo login...")
return self.login()
return True
def consultar_cpf(self, cpf):
"""Query um Brazilian Individual Taxpayer Registry (CPF)"""
if not self._verificar_token():
return None
url = f"{self.base_url}/consultar/{cpf}"
headers = {"Authorization": f"Bearer {self.token}"}
response = requests.get(url, headers=headers)
```

```
if response.status_code == 200:
return response.json()
elif response.status_code == 429:
print("■■ Limit de requisições excedido. Aguarde.")
return None
else:
print(f"■ Error: {response.status_code} - {response.text}")
return None
def consultar_lote(self, cpfs):
"""Query múltiplos Brazilian Individual Taxpayer Registry (CPF)s"""
resultados = []
for i, cpf in enumerate(cpfs, 1):
print(f"Queryndo {i}/{len(cpfs)}: {cpf}")
resultado = self.consultar_cpf(cpf)
if resultado:
resultados.append(resultado)
# Evita rate limit
if i % 10 == 0:
print("■ Pausa de 60 segundos...")
import time
time.sleep(60)
return resultados
# Uso
api = APIImpedidos(
base_url=" https://stg.opaservices.com.br",
username="seu_usuario",
password="sua_senha"
)
# Fazer login
api.login()
# Queryr um Brazilian Individual Taxpayer Registry (CPF)
resultado = api.consultar_cpf("12345678901")
if resultado:
print(f"Brazilian Individual Taxpayer Registry (CPF): {resultado['cpf']}")
print(f"Status: {resultado['status']}")
print(f"Transaction ID: {resultado['transaction_id']}")
```

```
# Queryr vários Brazilian Individual Taxpayer Registry (CPF)s
cpfs = ["12345678901", "98765432109", "11122233344"]
resultados = api.consultar_lote(cpfs)
print(f"\nTotal processado: {len(resultados)}")
for r in resultados:
print(f"{r['cpf']}: {r['status']}")
Example em JavaScript/Node.js
const axios = require('axios');
class APIImpedidos {
constructor(baseUrl, username, password) {
this.baseUrl = baseUrl;
this.username = username;
this.password = password;
this.token = null;
this.tokenExpira = null;
}
async login() {
try {
const response = await axios.post(`${this.baseUrl}/auth/login`, {
username: this.username,
password: this.password
});
this.token = response.data.access_token;
this.tokenExpira = Date.now() + (response.data.expires_in * 1000);
console.log('■ Login realizado com sucesso!');
return true;
} catch (error) {
console.error('■ Error no login:', error.response?.data);
return false;
}
}
async verificarToken() {
if (!this.token || Date.now() >= this.tokenExpira) {
console.log('■■ Token expirado, fazendo novo login...');
return await this.login();
```

```
}
return true;
}
async consultarCpf(cpf) {
if (!(await this.verificarToken())) {
return null;
}
try {
const response = await axios.get(
`${this.baseUrl}/consultar/${cpf}`,
{
headers: {
Authorization: `Bearer ${this.token}`
}
}
);
return response.data;
} catch (error) {
if (error.response?.status === 429) {
console.error('■■ Limit de requisições excedido');
} else {
console.error('■ Error:', error.response?.data);
}
return null;
}
}
}
// Uso
(async () => {
const api = new APIImpedidos(
'https://stg.opaservices.com.br/',
'seu_usuario',
'sua_senha'
);
await api.login();
const resultado = await api.consultarCpf('12345678901');
```

```
if (resultado) {
console.log('Brazilian Individual Taxpayer Registry (CPF):', resultado.cpf);
console.log('Status:', resultado.status);
console.log('Transaction ID:', resultado.transaction_id);
}
})();
```

Example em C#

```
using System;
using System.Net.Http;
using System.Text;
using System.Text.Json;
using System.Threading.Tasks;
public class APIImpedidos
{
private readonly string baseUrl;
private readonly string username;
private readonly string password;
private string token;
private DateTime tokenExpira;
private readonly HttpClient client;
public APIImpedidos(string baseUrl, string username, string password)
{
this.baseUrl = baseUrl;
this.username = username;
this.password = password;
this.client = new HttpClient();
}
public async Task Login()
{
var loginData = new
{
username = this.username,
password = this.password
};
var content = new StringContent(
```

```
JsonSerializer.Serialize(loginData),
Encoding.UTF8,
"application/json"
);
var response = await client.PostAsync(
$"{baseUrl}/auth/login",
content
);
if (response.IsSuccessStatusCode)
{
var result = await response.Content.ReadAsStringAsync();
var data = JsonSerializer.Deserialize(result);
token = data.GetProperty("access_token").GetString();
var expiresIn = data.GetProperty("expires_in").GetInt32();
tokenExpira = DateTime.Now.AddSeconds(expiresIn);
Console.WriteLine("■ Login realizado com sucesso!");
return true;
}
Console.WriteLine($"■ Error no login: {response.StatusCode}");
return false;
}
public async Task QueryrCpf(string cpf)
{
if (DateTime.Now >= tokenExpira)
{
Console.WriteLine("■■ Token expirado, fazendo novo login...");
await Login();
}
client.DefaultRequestHeaders.Authorization =
new System.Net.Http.Headers.AuthenticationHeaderValue("Bearer", token);
var response = await client.GetAsync($"{baseUrl}/consultar/{cpf}");
if (response.IsSuccessStatusCode)
{
return await response.Content.ReadAsStringAsync();
}
Console.WriteLine($"■ Error: {response.StatusCode}");
```

```
return null;
}
}
// Uso
class Program
{
static async Task Main(string[] args)
{
var api = new APIImpedidos(
"https://stg.opaservices.com.br",
"seu_usuario",
"sua_senha"
);
await api.Login();
var resultado = await api.QueryrCpf("12345678901");
Console.WriteLine(resultado);
}
}
```

■■ Códigos de Error

Example de Resposta de Error

```
{
"detail": "Brazilian Individual Taxpayer Registry (CPF) inválido. Deve conter 11 dígitos."
}
```

■ Rate Limiting

Cada usuário possui um limite de requisições por minuto configurado em seu cadastro.

Comportamento

Limit padrão: 10 requisições/minuto

O contador reseta a cada minuto

Ao exceder o limite, retorna erro 429

Resposta ao Exceder Limit

```
{
"detail": "Limit de requisições excedido. Tente novamente em 45 segundos."
}
```

Boas Práticas

Implemente retry com backoff exponencial

Processe em lotes respeitando o limite

Armazene resultados em cache quando possível

Monitore o header X-RateLimit-Remaining (se disponível)

■ Documentação Interativa

Acesse a documentação interativa Swagger UI:

URL: https://stg.opaservices.com.br/docs

Nela você pode:

■ Ver todos os endpoints

■ Testar requisições diretamente

■ Ver exemplos de request/response

■ Exportar especificação OpenAPI

■ Segurança

Recomendações

Nunca compartilhe suas credenciais

Use HTTPS em produção (configure certificado SSL)

Armazene tokens de forma segura (variáveis de ambiente)

Implemente rotação de senhas regularmente

Monitore acessos não autorizados

Use IP whitelist quando possível

Armazenamento Seguro de Credenciais

■ Não faça:

username = "meu_usuario" # Hardcoded no código

password = "minha_senha"

■ Faça:

import os

username = os.getenv("API_USERNAME")

password = os.getenv("API_PASSWORD")

■ Suporte

Canais de Atendimento

Informações para Suporte

Ao entrar em contato, tenha em mãos:

Nome de usuário

Transaction ID (se aplicável)

Timestamp do erro

Mensagem de erro completa

■ Changelog

Versão 3.0.0 (Atual)

■ Authentication JWT

■ Rate limiting por usuário

■ Sistema de transações com UUID

■ Logs detalhados de acesso

■ Suporte a consultas simultâneas

■ Pool de conexões otimizado

■ Termos de Uso

A API destina-se exclusivamente para consultas legítimas

É proibido revender ou redistribuir o acesso

O uso indevido pode resultar em bloqueio permanente

Os dados são fornecidos conforme disponibilidade do governo

Não nos responsabilizamos por decisões tomadas com base nos dados

■ FAQ

P: Quanto tempo o token JWT é válido? R: 60 minutos para usuários comuns

P: Posso consultar o mesmo Brazilian Individual Taxpayer Registry (CPF) múltiplas vezes? R: Sim, mas cada consulta conta para o rate limit.

P: Os resultados são em tempo real? R: Sim, consultamos diretamente a base oficial do governo.

P: Posso aumentar meu rate limit? R: Sim, entre em contato com o suporte comercial.

P: Como faço backup das transações? R: Use o endpoint /historico periodicamente para exportar seus dados.

P: A API aceita CNPJ? R: Não aceita CNPJ porque CNPJ não pode ser beneficiario de proramas sociais.

Última atualização: 16 de Outubro de 2025 Versão do documento: 1.0 Versão da API: 3.0.0