API for Checking Players on Blocked Status

Technical Documentation

Version: 3.0
Date: 11/11/2025

🎯 Overview

The Blocked Status Query API allows you to check if a CPF (Brazilian taxpayer ID) has any restrictions with Brazilian government agencies. The service uses digital certification and a direct connection to official government databases.

Basic Information

Item	Descrição
URL Base/STG	https://stg.opaservices.com.br
URL Base/PROD	https:// prevent-api.plugnp.com
Protocol	HTTPS
Format	JSON
Auth	JWT Bearer Token
Timeout	30 segundos

🔐 Authentication

The API uses JWT (JSON Web Token) authentication. You must include the token in all requests in the `Authorization` header.

1. Get Credentials

Contact our team to receive:

- Username
- Initial password
- Request limit per minute

2. Login

Endpoint: POST /auth/login

Request:

```
{
  "username": "your_user”
  "password": "your_passw"
}
```

Response (200 OK):

```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "token_type": "bearer",
  "expires_in": 3600
}
```

Exemplo cURL:

```
curl -X POST https://stg.opaservices.com.br/auth/login \
  -H "Content-Type: application/json" \
  -d '{
    "username": "your_user",
    "password": "your_passw"
  }'
```

3. Use the Token

Include the token in the Authorization header of all requests:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

⚠️ Important:

- Tokens expire after 60 minutes
- Store the token securely
- Renew the token before it expires

📡 Endpoints

1. Lookup CPF

Checks whether a CPF has any restrictions.

Endpoint: GET /consultar/{cpf}

Parameters

Parameter	Type	Required	Description
`cpf`	string	Yes	CPF to be queried (with or without formatting)

Headers:

Authorization: Bearer {your_token}
Content-Type: application/json

Response (200 OK):

Example with a CPF that is **not** a beneficiary of social programs and is **not** in the centralized self-exclusion database:


```
{
  "transaction_id": "550e8400-e29b-41d4-a716-446655440000",
  "cpf": "12345678901",
  "status": " NAO_IMPEDIDO",
  “motivo”: ""
  "timestamp": "2025-10-16T14:30:00.123456",
  "usuario": "your_user”
}
```


Example with a CPF that is **not** a beneficiary of social programs


```
{
  "transaction_id": "550e8400-e29b-41d4-a716-446655440000",
  "cpf": "12345678901",
  "status": "IMPEDIDO",
  “motivo”: "PROGRAMA_SOCIAL"
  "timestamp": "2025-10-16T14:30:00.123456",
  "usuario": "your_user"
}
```

Example with a CPF that is in the centralized self-exclusion database:

```
{
  "transaction_id": "550e8400-e29b-41d4-a716-446655440000",
  "cpf": "12345678901",
  "status": "IMPEDIDO",
  “motivo”: " AUTOEXCLUSAO_CENTRALIZADA"
  "timestamp": "2025-10-16T14:30:00.123456",
  "usuario": "your_user"
}
```

Possible status values

- NAO_IMPEDIDO – CPF without restrictions
- IMPEDIDO – CPF with active restrictions
- nao encontrado – CPF not found in the database
- indefinido – Status could not be determined

Possible motivo values

- PROGRAMA_SOCIAL – CPF blocked for being a beneficiary of social programs
- AUTOEXCLUSAO_CENTRALIZADA – CPF is in the centralized self-exclusion database
- "" – CPF is not blocked

cURL Example:

```
curl -X GET http://172.171.218.109:8000/consultar/12345678901 \
  -H "Authorization: Bearer your_token_here"
```

Python Exemple:

```
import requests

url = "https://stg.opaservices.com.br/consultar/12345678901"
headers = {
    "Authorization": "Bearer your_token_here"
}

response = requests.get(url, headers=headers)
data = response.json()

print(f"Status: {data['status']}")
print(f"Transaction ID: {data['transaction_id']}")
```

2. Lookup Transaction

Retrieves the details of a previous lookup.

Endpoint: GET /transacao/{transaction_id}

Parameters

Parameter	Type	Required	Description
transaction_id	string	Yes	Transaction UUID

Response (200 OK):


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


cURL Example:

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

3. Lookup History

Lists the most recent lookups performed by the user.

Endpoint: GET /historico?limite=10

Query parameters

Parameter	Type	Required	Default	Description
limite	integer	No	10	Maximum number of records

Response (200 OK):


```
{
  "usuario": "your_user ",
  "total_transacoes": 156,
  "transacoes": [
    {
      "transaction_id": "550e8400-e29b-41d4-a716-446655440000",
      "cpf": "12345678901",
      "status": "NAO_IMPEDIDO",
      "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
    }
  ]
}
```


Exemplo cURL:

```
curl -X GET "https://stg.opaservices.com.br/historico?limite=20" \
  -H "Authorization: Bearer your_token_here"
```

4. Health Check

Checks whether the API is up and responding.

Endpoint: GET /health

Response (200 OK):


```
{
  "status": "ok",
  "database": "connected",
  "timestamp": "2025-10-16T14:30:00.123456"
}
```


Exemplo cURL:

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

💡 Usage Examples

Full Example in 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):
        """ Log in and obtain a JWT token."""
        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 successful!")
            return True
        else:
            print(f"❌ Login error: {response.status_code}")
            return False

    def _verificar_token(self):
        """ Check if the token is still valid."""
        if not self.token or datetime.now() >= self.token_expira:
            print("⚠️ Token expired, logging in again...")
            return self.login()
        return True

    def consultar_cpf(self, cpf):
        """CPF lookup"""
        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("⚠️ Rate limit exceeded. Please wait.")
            return None
        else:
            print(f"❌ Error: {response.status_code} - {response.text}")
            return None

    def consultar_lote(self, cpfs):
        """Lookup multiples CPFs"""
        results = []

        for i, cpf in enumerate(cpfs, 1):
            print(f"Querying {i}/{len(cpfs)}: {cpf}")
            result = self.consultar_cpf(cpf)

            if result:
                results.append(result)

            # Avoid rate limit
            if i % 10 == 0:
                print("⏳ 60-second pause...")
                import time
                time.sleep(60)

        return results


# Usage
api = APIImpedidos(
    base_url=" https://stg.opaservices.com.br",
    username="your_user",
    password="your_passw"
)

# login
api.login()

# Lookup one CPF
resultado = api.consultar_cpf("12345678901")
if result:
    print(f"CPF: {result['cpf']}")
    print(f"Status: {result['status']}")
    print(f"Status: {result[motivo]}")
    print(f"Transaction ID: {result['transaction_id']}")
```

```
# lookup multiples CPFs
cpfs = ["12345678901", "98765432109", "11122233344"]
results = api.consultar_lote(cpfs)

print(f"\nTotal processado: {len(results)}")
for r in result:
    print(f"{r['cpf']}: {r['status']}")
```

⚠️ Errors Codes

Code	Description	Solution
200	Success	–
400	Invalid CPF	Check CPF format (11 digits)
401	Invalid or expired token	Log in again
403	Inactive user	Contact support
404	Resource not found	Check URL and transaction ID
429	Rate limit exceeded	Wait for the time indicated in the message
500	Internal server error	Contact support

Error Response Example

```
{
  "detail": "Invalid CPF. It must contain 11 digits."
}
```

🚦 Rate Limiting

Each user has a per-minute request limit configured in their account.

Behavior

- Default limit: 100 requests/minute
- Counter resets every minute
- When the limit is exceeded, the API returns HTTP 429

Response when limit is exceeded

```
{
  "detail": "Rate limit exceeded. Try again in 45 seconds."
}
```

Best Practices

- Implement retries with exponential backoff
- Process batches while respecting the limit
- Cache results whenever possible
- Monitor the X-RateLimit-Remaining header (if available)

📊 Interactive Documentation

Access the interactive Swagger UI:

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

There you can:

- ✅ View all endpoints
- ✅ Test requests directly
- ✅ See request/response examples
- ✅ Export the OpenAPI specification

🔒 Security

Recommendations

- Never share your credentials
- Use HTTPS in production (configure SSL certificate)
- Store tokens securely (environment variables)
- Implement regular password rotation
- Monitor for unauthorized access
- Use IP whitelisting when possible

Secure Credential Storage

❌ Do NOT:

```
username = "your_user"  # Hardcoded in the code
password = "your_pasww"
```

✅ Do:

```
import os
username = os.getenv("API_USERNAME")
password = os.getenv("API_PASSWORD")
```

📝 Changelog

Version 3.0.0 (Current)

✅ JWT authentication
✅ Per-user rate limiting
✅ Transaction system with UUID
✅ Detailed access logs
✅ Support for concurrent queries
✅ Optimized connection pool

📄 Terms of Use

- The API is intended exclusively for legitimate lookups
- It is forbidden to resell or redistribute access
- Misuse may result in permanent blocking
- Data is provided based on government system availability
- We are not responsible for decisions made based on the data

---

🎓 FAQ

Q: How long is the JWT token valid?
A: 60 minutes for regular users.

Q: Can I query the same CPF multiple times?
A: Yes, but each query counts towards the rate limit.

Q: Are the results real-time?
A: Yes, we query the official government database directly.

Q: Can I increase my rate limit?
A: Yes, contact the commercial support team.

Q: How can I back up the transactions?
A: Use the /historico endpoint periodically to export your data.

Q: Does the API accept CNPJ?
A: No, CNPJ is not accepted because CNPJ cannot be a beneficiary of social programs.

Last updated: November 11, 2025
Document version: 3.0
API version: 3.0.0