Guia de Integração com PJe via MNI: Consulta de Processos e Download de Documentos

Este guia apresenta os procedimentos para integração com o sistema PJe (Processo Judicial Eletrônico) através do padrão MNI (Modelo Nacional de Interoperabilidade), com foco na consulta de processos e download de documentos.

Sumário

1. Introdução ao MNI
2. Tribunais Disponíveis
3. Autenticação
4. Consulta de Processos
5. Download de Documentos
6. Exemplos Completos em Python
7. Tratamento de Erros
8. Considerações Finais

Introdução ao MNI

O Modelo Nacional de Interoperabilidade (MNI) é o padrão definido pelo CNJ para a troca de informações processuais entre sistemas. Ele utiliza tecnologia WebService baseada em SOAP/WSDL, permitindo a integração entre diferentes sistemas e o PJe.

As principais operações disponibilizadas pelo MNI são:

- `entregarManifestacaoProcessual`: criar processos ou anexar documentos
- `consultarProcesso`: visualizar informações de processos
- `consultarAvisosPendentes`: verificar comunicações pendentes
- `consultarTeorComunicacao`: acessar conteúdo de comunicações

Diferentemente de APIs REST modernas, o MNI usa um endpoint único com diferentes operações.

Tribunais Disponíveis

Abaixo estão os endpoints WSDL dos principais tribunais que implementaram o MNI:

Tribunais Estaduais

- **TJDFT**: https://tjdf199.tjdft.jus.br/pje/intercomunicacao?wsdl
- **TJRN**:

    - 1ª instância: https://pje.tjrn.jus.br/pje1grau/intercomunicacao?wsdl
    - 2ª instância: https://pje.tjrn.jus.br/pje2grau/intercomunicacao?wsdl
- **TJMG**: http://pje.tjmg.jus.br/pje/intercomunicacao?wsdl
- **TJMT**:
    - 1ª instância: http://pje.tjmt.jus.br/pje/intercomunicacao?wsdl
    - 2ª instância: http://pje2.tjmt.jus.br/pje2/intercomunicacao?wsdl
- **TJRO**:
    - 1ª instância: http://pje.tjro.jus.br/pg/intercomunicacao?wsdl
    - 2ª instância: http://pje.tjro.jus.br/sg/intercomunicacao?wsdl
- **TJRR**: http://pje.tjrr.jus.br/pje/intercomunicacao?wsdl
- **TJPB**:
    - 1ª instância: https://pje.tjpb.jus.br/pje/intercomunicacao?wsdl
    - 2ª instância: https://pje.tjpb.jus.br/pje2g/intercomunicacao?wsdl
- **TJPE**:
    - 1ª instância: https://www.tjpe.jus.br/pje/intercomunicacao?wsdl
    - 2ª instância: https://www.tjpe.jus.br/pje2g/intercomunicacao?wsdl
- **TJMA**:
    - 1ª instância: https://pje.tjma.jus.br/pje/intercomunicacao?wsdl
    - 2ª instância: https://pje2.tjma.jus.br/pje2g/intercomunicacao?wsdl
- **TJGO**: https://www.tjgo.jus.br/pje/intercomunicacao?wsdl
- **TJBA**: https://pje.tjba.jus.br/intercomunicacao?wsdl
- **TJRS**: http://www.tjrs.jus.br/pje/intercomunicacao?wsdl
- **TJCE**:
    - 1ª instância: https://pje.tjce.jus.br/pje1grau/intercomunicacao?wsdl
    - 2ª instância: https://pje.tjce.jus.br/pje2grau/intercomunicacao?wsdl
- **TJES**: https://sistemas.tjes.jus.br/pje/intercomunicacao?wsdl

Justiça Federal

- **TRF5**: https://pje.trf5.jus.br/pje/intercomunicacao?wsdl
- **Seções Judiciárias**:
    - Alagoas: https://pje.jfal.jus.br/pje/intercomunicacao?wsdl
    - Ceará: https://pje.jfce.jus.br/pje/intercomunicacao?wsdl
    - Paraíba: https://pje.jfpb.jus.br/pje/intercomunicacao?wsdl
    - Pernambuco: https://pje.jfpe.jus.br/pje/intercomunicacao?wsdl

- Sergipe: https://pje.jfse.jus.b/pje/intercomunicacao?wsdl
- Rio Grande do Norte: https://pje.jfrn.jus.br/pje/intercomunicacao?wsdl

Outros órgãos

- **CNJ**: https://www.cnj.jus.br/pjecnjinterno/intercomunicacao?wsdl
- **CJF (TNU)**: http://www2.cjf.jus.br/pje/intercomunicacao?wsdl
- **Ambiente de testes (CNJ)**: https://wwwh.cnj.jus.br/pjemni-2x/intercomunicacao?wsdl

Autenticação

Cada requisição ao MNI exige autenticação via CPF/CNPJ e senha. Essas credenciais são as mesmas utilizadas para acesso ao sistema PJe via navegador.

O acesso aos documentos respeita o nível de sigilo e as permissões do usuário autenticado no sistema.

Consulta de Processos

Para consultar um processo, é necessário usar a operação `consultarProcesso`. Abaixo está a estrutura do XML e um exemplo prático em Python.

Estrutura XML

```xml
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ser="http://s
    <soapenv:Header/>
    <soapenv:Body>
        <ser:consultarProcesso>
            <idConsultante>SEU_CPF_OU_CNPJ</idConsultante>
            <senhaConsultante>SUA_SENHA</senhaConsultante>
            <numeroProcesso>NUMERO_PROCESSO_CNJ</numeroProcesso>
            <movimentos>true</movimentos>
            <incluirCabecalho>true</incluirCabecalho>
            <incluirDocumentos>true</incluirDocumentos>
        </ser:consultarProcesso>
    </soapenv:Body>
</soapenv:Envelope>
```

Exemplo em Python

python

```python
import requests
from zeep import Client

# Configurações
MNI_URL = "https://pje.tjce.jus.br/pje1grau/intercomunicacao?wsdl"  # Substitua pelo tribunal d
CPF = "SEU_CPF"
SENHA = "SUA_SENHA"
NUMERO_PROCESSO = "0000000-00.0000.0.00.0000"  # Número CNJ do processo

# Criar cliente SOAP
client = Client(MNI_URL)

# Parâmetros da consulta
params = {
    'idConsultante': CPF,
    'senhaConsultante': SENHA,
    'numeroProcesso': NUMERO_PROCESSO,
    'movimentos': True,
    'incluirCabecalho': True,
    'incluirDocumentos': True
}

# Executa a consulta
try:
    with client.settings(strict=False, xml_huge_tree=True):
        response = client.service.consultarProcesso(**params)

    # Verifica se a consulta foi bem-sucedida
    if response.sucesso:
        print(f"Processo consultado com sucesso: {NUMERO_PROCESSO}")
        # Processa os documentos disponíveis
        if hasattr(response.processo, 'documento'):
            documentos = response.processo.documento
            if not isinstance(documentos, list):
                documentos = [documentos]

            print(f"Encontrados {len(documentos)} documentos no processo")
            for doc in documentos:
                print(f"ID: {doc.idDocumento}, Tipo: {doc.tipoDocumento}, Descrição: {doc.descr
    else:
        print(f"Falha na consulta: {response.mensagem}")
except Exception as e:
    print(f"Erro na consulta: {str(e)}")
```

Download de Documentos

Após identificar os documentos do processo, você pode baixá-los individualmente. O download é feito também através da operação `consultarProcesso`, mas incluindo o parâmetro `documento` com o ID do documento desejado.

Estrutura XML

```xml
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ser="http://s
    <soapenv:Header/>
    <soapenv:Body>
        <ser:consultarProcesso>
            <idConsultante>SEU_CPF_OU_CNPJ</idConsultante>
            <senhaConsultante>SUA_SENHA</senhaConsultante>
            <numeroProcesso>NUMERO_PROCESSO_CNJ</numeroProcesso>
            <documento>ID_DO_DOCUMENTO</documento>
        </ser:consultarProcesso>
    </soapenv:Body>
</soapenv:Envelope>
```

Exemplo em Python

python

```python
def baixar_documento(num_processo, id_documento):
    """Função para baixar um documento específico do processo"""

    # Parâmetros da consulta
    params = {
        'idConsultante': CPF,
        'senhaConsultante': SENHA,
        'numeroProcesso': num_processo,
        'documento': id_documento
    }

    try:
        with client.settings(strict=False, xml_huge_tree=True):
            response = client.service.consultarProcesso(**params)

        if response.sucesso and hasattr(response.processo, 'documento'):
            # O documento retornado vem codificado em base64
            doc = response.processo.documento
            if isinstance(doc, list):
                # Procurando o documento na lista
                for item in doc:
                    if getattr(item, 'idDocumento', '') == id_documento:
                        doc = item
                        break

            if hasattr(doc, 'conteudo') and doc.conteudo:
                # Determinar extensão com base no mimetype
                extensao = '.pdf'  # Padrão
                if hasattr(doc, 'mimetype'):
                    if doc.mimetype == 'application/pdf':
                        extensao = '.pdf'
                    elif doc.mimetype == 'text/html':
                        extensao = '.html'
                    # Adicione mais mimetypes conforme necessário

                # Salvar arquivo
                with open(f"{id_documento}{extensao}", 'wb') as f:
                    f.write(doc.conteudo)
                print(f"Documento {id_documento} salvo com sucesso.")
                return True
            else:
                print(f"Documento {id_documento} não contém conteúdo.")
        else:
            print(f"Falha ao baixar documento: {response.mensagem if hasattr(response, 'mensage

        return False
```

```
    except Exception as e:
        print(f"Erro ao baixar documento {id_documento}: {str(e)}")
        return False

# Exemplo de uso
id_documento = "12345"  # Substitua pelo ID real
baixar_documento(NUMERO_PROCESSO, id_documento)
```

Exemplos Completos em Python

Baixando Todos os Documentos de Um Processo

python

```python
def baixar_todos_documentos(num_processo):
    """Função para baixar todos os documentos de um processo"""

    # Consultar o processo primeiro para obter a lista de documentos
    params = {
        'idConsultante': CPF,
        'senhaConsultante': SENHA,
        'numeroProcesso': num_processo,
        'incluirDocumentos': True
    }

    try:
        with client.settings(strict=False, xml_huge_tree=True):
            response = client.service.consultarProcesso(**params)

        if response.sucesso and hasattr(response.processo, 'documento'):
            documentos = response.processo.documento
            if not isinstance(documentos, list):
                documentos = [documentos]

            print(f"Iniciando download de {len(documentos)} documentos...")

            # Criar diretório para os documentos (opcional)
            import os
            dir_name = f"processo_{num_processo.replace('.', '_').replace('-', '_')}"
            os.makedirs(dir_name, exist_ok=True)

            # Baixar cada documento
            for doc in documentos:
                id_doc = doc.idDocumento
                print(f"Baixando documento {id_doc}...")

                # Consultar documento específico
                doc_params = {
                    'idConsultante': CPF,
                    'senhaConsultante': SENHA,
                    'numeroProcesso': num_processo,
                    'documento': id_doc
                }

                with client.settings(strict=False, xml_huge_tree=True):
                    doc_response = client.service.consultarProcesso(**doc_params)

                if doc_response.sucesso:
                    doc_item = doc_response.processo.documento
                    if not isinstance(doc_item, list):
```

```
                    doc_item = [doc_item]

                # Encontrar o documento correto na lista
                for item in doc_item:
                    if getattr(item, 'idDocumento', '') == id_doc and hasattr(item, 'conteu
                        # Determinar extensão
                        extensao = '.pdf'
                        if hasattr(item, 'mimetype'):
                            if item.mimetype == 'application/pdf':
                                extensao = '.pdf'
                            elif item.mimetype == 'text/html':
                                extensao = '.html'

                        # Salvar arquivo
                        filename = f"{dir_name}/{id_doc}{extensao}"
                        with open(filename, 'wb') as f:
                            f.write(item.conteudo)
                        print(f"  Documento {id_doc} salvo como {filename}")
                        break
            else:
                print(f"  Falha ao baixar documento {id_doc}")

        print("Download de documentos concluído.")
    else:
        print(f"Falha na consulta do processo: {response.mensagem if hasattr(response, 'mer

except Exception as e:
    print(f"Erro ao processar documentos: {str(e)}")

# Exemplo de uso
baixar_todos_documentos(NUMERO_PROCESSO)
```

Implementação Completa com Tratamento de Erros e Documentos Vinculados

python

```python
import os
import time
import requests
from zeep import Client
from zeep.exceptions import Fault

# Configurações
MNI_URL = "https://pje.tjce.jus.br/pje1grau/intercomunicacao?wsdl"  # Substitua pelo tribunal c
CPF = "SEU_CPF"
SENHA = "SUA_SENHA"

# Mapeamento de tipos MIME para extensões de arquivo
mime_to_extension = {
    'application/pdf': '.pdf',
    'image/jpeg': '.jpg',
    'image/png': '.png',
    'application/zip': '.zip',
    'text/plain': '.txt',
    'application/msword': '.doc',
    'application/vnd.ms-excel': '.xls',
    'application/vnd.openxmlformats-officedocument.wordprocessingml.document': '.docx',
    'application/vnd.openxmlformats-officedocument.spreadsheetml.sheet': '.xlsx',
    'text/html': '.html',
}

class PJeClient:
    def __init__(self, wsdl_url, cpf, senha):
        self.wsdl_url = wsdl_url
        self.cpf = cpf
        self.senha = senha
        self.client = None
        self.conectar()

    def conectar(self):
        try:
            self.client = Client(self.wsdl_url)
            print("Cliente SOAP criado com sucesso.")
        except Exception as e:
            print(f"Erro ao criar cliente SOAP: {str(e)}")
            raise

    def consultar_processo(self, num_processo, incluir_documentos=True, incluir_movimentos=True
        """Consulta informações do processo"""
        params = {
            'idConsultante': self.cpf,
            'senhaConsultante': self.senha,
```

```
            'numeroProcesso': num_processo,
            'movimentos': incluir_movimentos,
            'incluirCabecalho': True,
            'incluirDocumentos': incluir_documentos
        }

        try:
            with self.client.settings(strict=False, xml_huge_tree=True):
                response = self.client.service.consultarProcesso(**params)

            if response.sucesso:
                return response
            else:
                print(f"Falha na consulta: {response.mensagem}")
                return None

        except Fault as e:
            print(f"Erro SOAP: {str(e)}")
            return None
        except Exception as e:
            print(f"Erro inesperado: {str(e)}")
            return None

    def baixar_documento(self, num_processo, id_documento, diretorio=None):
        """Baixa um documento específico"""
        params = {
            'idConsultante': self.cpf,
            'senhaConsultante': self.senha,
            'numeroProcesso': num_processo,
            'documento': id_documento
        }

        try:
            with self.client.settings(strict=False, xml_huge_tree=True):
                response = self.client.service.consultarProcesso(**params)

            if not response.sucesso:
                print(f"Falha ao baixar documento: {response.mensagem}")
                return False

            # Extrair documento da resposta
            doc = None

            if hasattr(response.processo, 'documento'):
                docs = response.processo.documento
                if not isinstance(docs, list):
                    docs = [docs]
```

```
            # Buscar documento pelo ID
            for item in docs:
                if getattr(item, 'idDocumento', '') == id_documento:
                    doc = item
                    break

        if not doc or not hasattr(doc, 'conteudo') or not doc.conteudo:
            print(f"Documento {id_documento} não encontrado ou sem conteúdo.")
            return False

        # Determinar extensão com base no mimetype
        mimetype = getattr(doc, 'mimetype', 'application/octet-stream')
        extensao = mime_to_extension.get(mimetype, '.bin')

        # Preparar nome do arquivo
        filename = f"{id_documento}{extensao}"
        if diretorio:
            os.makedirs(diretorio, exist_ok=True)
            filename = os.path.join(diretorio, filename)

        # Salvar arquivo
        with open(filename, 'wb') as f:
            f.write(doc.conteudo)

        print(f"Documento {id_documento} salvo como {filename}")
        return True

    except Exception as e:
        print(f"Erro ao baixar documento {id_documento}: {str(e)}")
        return False

def baixar_todos_documentos(self, num_processo, diretorio=None, incluir_vinculados=True):
    """Baixa todos os documentos de um processo"""
    if not diretorio:
        diretorio = f"processo_{num_processo.replace('.', '_').replace('-', '_')}"

    os.makedirs(diretorio, exist_ok=True)

    # Primeiro, consulta o processo para obter a lista de documentos
    response = self.consultar_processo(num_processo)
    if not response or not hasattr(response.processo, 'documento'):
        print("Não foi possível obter a lista de documentos.")
        return False

    # Lista para armazenar todos os IDs de documentos (principais e vinculados)
    todos_documentos = []
```

```python
# Processar documentos principais
documentos = response.processo.documento
if not isinstance(documentos, list):
    documentos = [documentos]

# Adicionar documentos principais à lista
for doc in documentos:
    id_doc = doc.idDocumento
    todos_documentos.append({
        'id': id_doc,
        'tipo': getattr(doc, 'tipoDocumento', 'Desconhecido'),
        'descricao': getattr(doc, 'descricao', ''),
        'vinculado_a': None
    })

    # Processar documentos vinculados, se existirem
    if incluir_vinculados and hasattr(doc, 'documentoVinculado'):
        vinculados = doc.documentoVinculado
        if not isinstance(vinculados, list):
            vinculados = [vinculados]

        for vinc in vinculados:
            id_vinc = getattr(vinc, 'idDocumento', None)
            if id_vinc:
                todos_documentos.append({
                    'id': id_vinc,
                    'tipo': getattr(vinc, 'tipoDocumento', 'Desconhecido'),
                    'descricao': getattr(vinc, 'descricao', ''),
                    'vinculado_a': id_doc
                })

# Baixar cada documento
print(f"Iniciando download de {len(todos_documentos)} documentos...")

documentos_baixados = 0
for info_doc in todos_documentos:
    id_doc = info_doc['id']
    tipo_doc = info_doc['tipo']
    descricao = info_doc['descricao']
    vinculado_a = info_doc['vinculado_a']

    # Informação para o usuário
    vinc_info = f" (vinculado ao documento {vinculado_a})" if vinculado_a else ""
    print(f"Baixando documento {id_doc} - {tipo_doc} - {descricao}{vinc_info}...")

    # Criar subdiretório para documentos vinculados (opcional)
```

```
            dir_doc = diretorio
            if vinculado_a:
                dir_doc = os.path.join(diretorio, f"vinculados_a_{vinculado_a}")
                os.makedirs(dir_doc, exist_ok=True)

            # Baixar o documento
            if self.baixar_documento(num_processo, id_doc, dir_doc):
                documentos_baixados += 1

            # Pausa para não sobrecarregar o servidor
            time.sleep(1)

        print(f"Download concluído. {documentos_baixados} de {len(todos_documentos)} documentos
        return documentos_baixados > 0

# Exemplo de uso
def main():
    pje = PJeClient(MNI_URL, CPF, SENHA)

    # Consultar um processo
    numero_processo = "0000000-00.0000.0.00.0000"  # Substitua pelo número real
    response = pje.consultar_processo(numero_processo)

    if response:
        print(f"Detalhes do processo {numero_processo}:")
        print(f"Classe: {getattr(response.processo, 'classeProcessual', 'N/A')}")
        print(f"Órgão Julgador: {getattr(getattr(response.processo, 'orgaoJulgador', {}), 'desc

        # Baixar todos os documentos
        pje.baixar_todos_documentos(numero_processo)

if __name__ == "__main__":
    main()
```

Tratamento de Erros

Ao trabalhar com a API MNI do PJe, é importante implementar um bom tratamento de erros. Alguns dos erros comuns incluem:

1. **Erro de autenticação**: Credenciais incorretas ou expiradas
2. **Processo não encontrado**: Número do processo incorreto ou sem permissão de acesso
3. **Documento não encontrado**: ID de documento inválido
4. **Timeout**: O servidor pode demorar para responder em processos grandes
5. **Erros de estrutura XML**: Inconsistências no formato da requisição

Estratégias para lidar com erros:

- Implemente retry com backoff exponencial para erros temporários
- Verifique a conectividade com o servidor antes de iniciar operações em lote
- Mantenha logs detalhados para depuração
- Monitore o uso de memória ao baixar documentos grandes

Considerações Finais

Limitações e Boas Práticas

1. **Performance**: O MNI não foi projetado para consultas em massa. Limite o número de requisições simultâneas.
2. **Sistemas de armazenamento**: Diferentes tribunais podem usar sistemas diferentes (DB-Storage ou JCR-Storage), o que pode afetar a forma como os documentos são identificados e acessados.
3. **Versões do PJe**: Diferentes tribunais podem estar em diferentes versões do PJe, o que pode causar variações nas respostas.
4. **Cache**: Considere implementar cache para consultas frequentes para reduzir a carga nos servidores do tribunal.
5. **Documentação**: Mantenha-se atualizado com a documentação oficial do CNJ sobre o MNI, pois o padrão pode evoluir.

Requisitos para Implementação

Para implementar a integração com o PJe via MNI, você precisará:

1. **Credenciais válidas**: CPF/CNPJ e senha de acesso ao PJe
2. **Bibliotecas SOAP**: Para Python, a biblioteca zeep é recomendada
3. **Tratamento de XML**: Capacidade de processar documentos XML complexos
4. **Armazenamento**: Espaço suficiente para salvar documentos, que podem ser grandes

Próximos Passos

- Explore outras operações do MNI, como `entregarManifestacaoProcessual` para peticionamento
- Implemente monitoramento de prazos usando `consultarAvisosPendentes`
- Considere desenvolver uma interface gráfica para facilitar o uso por não-programadores
- Integre com sistemas de gestão documental para organização automática dos documentos baixados

---

Este guia fornece uma base sólida para começar a integração com o PJe via MNI. Lembre-se que cada tribunal pode ter particularidades em sua implementação, então esteja preparado para adaptações.