República de Moçambique

API de Assinatura Avançada

Integração com a API de Assinatura Avançada

Especificação do Uso de Tokens JWT

Autor:
Augusto de Hollanda Vieira Guerner

2026

Sumário

Resumo

Este documento descreve o processo de integração com a API de Assinatura Avançada do Governo de Moçambique, com foco na autenticação e autorização por meio de tokens JWT. São apresentados os requisitos para registro de Provedores de Identidade externos, a estrutura esperada dos tokens de acesso, exemplos de uso das APIs de assinatura e verificação de documentos, bem como a implementação de um IdP mínimo para fins de teste.

1 Introdução

Este documento tem por objetivo responder às perguntas exigidas por Moçambique para o uso da API. Será detalhado como registrar um IdP externo no sistema, como estruturar um token de acesso, como requisitar a API e como construir um IdP básico para teste.

2 Como registrar o IdP externo

Para que um IdP externo seja reconhecido pelo sistema, deve-se configurar a variável de ambiente `ISSUERS_FOR_JWT_VALIDATION`. Essa variável estabelece o mapeamento entre o emissor do token (`iss`) e a URL onde a chave pública correspondente pode ser obtida.

A seguir está um exemplo de uma possível definição:

```
ISSUERS_FOR_JWT_VALIDATION: >-
{
    "http://dev-idp:5000": "http://dev-idp:5000/keys",
    "https://outro-idp": "https://outro-idp/chave-publica"
}
```

Note que é um dicionário simples, em que a chave é uma string e o valor também. Será visto mais adiante que a chave é o campo iss do token e o valor é a url onde se deve ter a chave pública do IdP externo. Destaque-se que a chave privada é usada pelo provedor de identidade para emitir o token de acesso.

3 Estrutura do Token JWT

O token possui dois tipos de campos: os obrigatórios e os opcionais. Além disso, o token esperado pela aplicação é do tipo JWT (para mais informações veja este site https://www.jwt.io).

3.1 Campos Obrigatórios

O token JWT utilizado na integração deve conter um conjunto mínimo de campos obrigatórios, responsáveis por identificar o emissor, o titular e a validade temporal do token. A ausência ou formatação incorreta de qualquer um desses campos invalida o token para fins de autenticação e autorização.

3.1.1 Campo `iss` (Issuer)

O campo `iss` identifica o emissor do token JWT.

Sua principal finalidade é permitir que o sistema consumidor verifique se o token foi emitido por uma autoridade confiável previamente registrada. O valor deste campo deve corresponder exatamente a um dos emissores configurados no sistema.

Formato esperado:

- Tipo: String
- Normalmente uma URL ou identificador único

Exemplo:

```
"iss": "http://dev-idp:5000"
```

3.1.2 Campo `iat` (Issued At)

O campo `iat` indica a data e hora em que o token foi emitido.

Este campo é utilizado para validar a idade do token e auxiliar na prevenção de reutilização indevida de tokens antigos.

Formato esperado:

- Tipo: Number
- Timestamp Unix (segundos desde 01/01/1970 UTC)

Exemplo:

```
"iat": 1706959200
```

3.1.3 Campo `exp` (Expiration Time)

O campo `exp` define a data e hora de expiração do token.

Após o momento indicado por este campo, o token deve ser considerado inválido pelo sistema consumidor.

Formato esperado:

- Tipo: Number
- Timestamp Unix (em segundos)
- Deve ser maior que o valor do campo `iat`

Exemplo:

```
"exp": 1706962800
```

3.1.4 Campo `name`

O campo `name` representa o nome completo do titular do token.

Este campo é utilizado para identificação nominal do usuário nos sistemas consumidores.

Formato esperado:

- Tipo: String
- Preferencialmente o nome completo, sem abreviações

Exemplo:

```
"name": "João da Silva"
```

3.1.5 Campo `email`

O campo `email` indica o endereço de correio eletrônico do titular do token.

Este campo pode ser utilizado como meio de contato e como identificador único do usuário.

Formato esperado:

- Tipo: String
- Deve obedecer ao padrão RFC 5322

Exemplo:

```
"email": "joao@gmail.com"
```

3.1.6 Campo `nuit`

O campo `nuit` corresponde ao Número Único de Identificação Tributária do titular.

É utilizado para fins de identificação fiscal.

Formato esperado:

- Tipo: String ou Number
- Deve conter apenas dígitos

Exemplo:

```
"nuit": "123456789"
```

3.1.7 Campo `nuic`

O campo `nuic` corresponde ao Número Único de Identificação Civil do cidadão.

Este identificador é utilizado para identificação civil única.

Formato esperado:

- Tipo: String ou Number
- Deve conter apenas dígitos

Exemplo:

```
"nuic": "123456789"
```

3.1.8 Campo `nuib`

O campo `nuib` representa o Número Único de Identificação Bancária do titular.

É utilizado para identificação bancária.

Formato esperado:

- Tipo: String ou Number
- Deve conter apenas dígitos

Exemplo:

```
"nuib": "123456789"
```

3.1.9 Campo `bi`

O campo `bi` indica o número do Bilhete de Identidade do titular.

Este é um documento oficial de identificação pessoal.

Formato esperado:

- Tipo: String
- Pode conter letras e números

Exemplo:

```
"bi": "110101234567A"
```

É obrigatório que ao menos um dos identificadores `NUIT`, `NUIC`, `NUIB` ou `BI` esteja presente no token.

3.2 Campos Opcionais do Token JWT

Além dos campos obrigatórios, o token JWT pode conter campos opcionais que complementam a identificação do titular, sem impactar a validade estrutural do token.

3.2.1 Campo `chosen_name`

O campo `chosen_name` representa o nome escolhido pelo titular, distinto do nome civil registrado oficialmente no campo `name`.

Este campo permite que os sistemas consumidores utilizem o nome pelo qual o usuário prefere ser identificado, respeitando identidade social, nome social ou nome de uso corrente, sem a necessidade de alterar o nome civil utilizado para fins legais e administrativos.

Formato esperado:

- Tipo: String

Exemplo:

```
"chosen_name": "João"
```

3.3 Prefixo para Campos do Token JWT

Considerando a definição dos campos obrigatórios e opcionais do token JWT, o sistema permite a utilização de um prefixo nos nomes dos campos, com o objetivo de possibilitar a reutilização de identificadores já reservados ou evitar conflitos com campos existentes.

Como exemplo, pode-se utilizar o prefixo `idmz_`, fazendo com que os campos passem a ser identificados como `idmz_name`, `idmz_email`, entre outros.

A definição do prefixo a ser utilizado deve ser realizada por meio da configuração da seguinte variável de ambiente:

```
PREFIX_FOR_JWT_VALIDATION: IDMZ_
```

Apesar da disponibilidade dessa funcionalidade, recomenda-se que, nesta fase inicial de integração, o uso de prefixos não seja adotado, de modo a simplificar a implementação e reduzir a complexidade do sistema.

4 Uso da API

Esta seção descreve o processo de utilização da API de Assinatura Avançada, abrangendo tanto a funcionalidade de assinatura quanto a de verificação de documentos. Para um entendimento mais aprofundado sobre todas as possibilidades oferecidas pelo

sistema — especialmente no que se refere à configuração e à estampa visual da assinatura — recomenda-se a consulta à documentação específica do assinador.

4.1 Assinador

Para a assinatura de um documento no formato PDF, é obrigatória a utilização de um token JWT válido, emitido por um Provedor de Identidade (IdP) previamente cadastrado no sistema. O token deve conter, no mínimo, todos os atributos obrigatórios definidos neste documento e deve estar dentro de seu período de validade.

Antes de realizar a requisição, é fundamental verificar se o token não se encontra expirado e se todos os campos estão corretamente formatados, de modo a evitar falhas no processo de assinatura.

4.1.1 Exemplo de Requisição utilizando `curl`

```
curl -H "Authorization: Bearer {JWT}" \
     -X POST \
     -F "field_name=teste" \
     -F "file=@arquivo.pdf" \
     https://assinadoravancado.gov.mz/api/signer/pdf/1/sign \
     --output documento_assinado.pdf
```

4.1.2 Exemplo de Requisição utilizando Python

```python
import requests

# Configuracoes
url = "https://assinadoravancado.gov.mz/api/signer/pdf/1/sign"
jwt_token = "{JWT}"  # Substituir pelo JWT valido
arquivo_entrada = "arquivo.pdf"
arquivo_saida = "documento_assinado.pdf"

headers = {
    "Authorization": f"Bearer {jwt_token}"
}

files = {
    "file": open(arquivo_entrada, "rb")
}

data = {
```

```
    "field_name": "teste"
}

# Requisicao POST
response = requests.post(
    url,
    headers=headers,
    files=files,
    data=data
)

# Encerramento do arquivo apos o envio
files["file"].close()

# Verificacao da resposta
if response.status_code == 200:
    with open(arquivo_saida, "wb") as f:
        f.write(response.content)
    print("Documento assinado com sucesso!")
else:
    print("Erro ao assinar documento")
    print("Status:", response.status_code)
    print("Resposta:", response.text)
```

Em caso de falha na assinatura, recomenda-se a leitura atenta da mensagem de erro retornada pela API, pois ela pode indicar problemas relacionados à validade do token, à formatação dos campos ou ao próprio documento enviado. No uso do `curl`, pode ser necessário remover o parâmetro `-output` para que a mensagem de erro seja exibida diretamente no terminal.

4.2 Verificador

Para a verificação das assinaturas de um documento no formato PDF, não é necessária a utilização de um token JWT. O serviço de verificação retorna um objeto no formato JSON contendo as informações detalhadas de cada assinatura presente no documento, incluindo dados sobre validade, integridade e cadeia de certificação.

A seguir é apresentado um exemplo de requisição utilizando a ferramenta `curl`:

```
curl -X POST \
https://assinadoravancado.gov.mz/api/verifier/pdf/1/verify \
-F file=@documento_assinado.pdf
```

Caso seja necessária a implementação em Python, pode-se reaproveitar a estrutura do código apresentada na subseção anterior, adaptando apenas a URL e removendo a necessidade de autenticação por token JWT.

5 Exemplo de Provedor de Identidade

Esta seção apresenta a implementação de um Provedor de Identidade (IdP) simples para fins de teste, desenvolvido em Python. O objetivo deste IdP é permitir a emissão de tokens JWT e a disponibilização da chave publica correspondente para validação dos tokens emitidos.

Para o funcionamento adequado do IdP, são necessários dois componentes fundamentais:

- A geracao de tokens JWT assinados com chave privada;
- A exposicao da chave publica para validacao dos tokens pelo sistema consumidor.

5.1 Geracao do Par de Chaves

Inicialmente, deve-se criar um par de chaves criptograficas do tipo RSA, sendo a chave privada utilizada para a assinatura dos tokens JWT e a chave publica disponibilizada para verificacao da assinatura.

A leitura das chaves pode ser realizada conforme o exemplo a seguir:

```
with open("public-key.pem", "rb") as f:
    PUBLIC_KEY_BYTES = f.read()

with open("private-key.pem", "rb") as f:
    PRIVATE_BYTES = f.read()
```

5.2 Rota de Emissao de Token JWT

Com as chaves carregadas, o proximo passo consiste na criacao de uma rota responsavel pela emissao dos tokens JWT. Essa rota pode ser implementada utilizando frameworks como Flask ou FastAPI.

A seguir apresenta-se um exemplo de rota utilizando Flask:

```
@application.route("/issue", methods=["GET", "POST"])
def issue_token():
    params = request.form.to_dict(flat=True)

```

```
    default_attributes = {
        "iss": "http://dev-idp:5000",
        "exp": int(time()) + 3600,
        "iat": int(time()),
        "chosen_name": "Maria",
        "name": "John␣Doe",
        "email": "john.doe@example.com",
        "bi": "110100006699B",
    }

    jws = JsonWebSignature(algorithms=["RS256"])
    return jws.serialize_compact(
        {"alg": "RS256", "kid": KID},
        json.dumps(default_attributes),
        PRIVATE_BYTES
    ).decode(), 200
```

Ressalta-se que, neste exemplo, a rota sempre gera o mesmo token, uma vez que se trata apenas de uma implementacao demonstrativa destinada a fins de teste e validacao da integracao.

5.3 Rota de Exposicao da Chave Publica

Por fim, e necessario disponibilizar uma rota responsavel por expor a chave publica do IdP. Essa chave sera utilizada pelo sistema consumidor para verificar a assinatura dos tokens JWT emitidos.

Um exemplo de implementacao dessa rota e apresentado a seguir:

```
@application.get("/keys")
def get_keys():
    return jsonify({
        "keys": [
            JsonWebKey.import_key(
                PUBLIC_KEY_BYTES.decode(),
                {"kty": "RSA", "kid": KID}
            ).as_dict()
        ]
    })
```

E de fundamental importancia que o valor configurado na variavel de ambiente de registro do IdP externo, conforme descrito na Secao 2, corresponda exatamente a esta

rota de exposicao da chave publica, garantindo assim a correta validacao dos tokens JWT emitidos.

6 Considerações Finais

A integracao com a API de Assinatura Avancada pressupoe uma divisao clara de responsabilidades entre o cliente integrador e o sistema da Republica de Mocambique.

E de responsabilidade do cliente integrador a correta implementacao do seu Provedor de Identidade (IdP), incluindo, mas nao se limitando, aos seguintes aspectos:

- A geracao do token JWT, respeitando integralmente a estrutura, os campos obrigatorios e opcionais, bem como os formatos definidos neste documento;
- A gestao segura das chaves criptograficas, garantindo o uso adequado da chave privada exclusivamente para a assinatura dos tokens emitidos;
- A exposicao da chave publica por meio de uma rota acessivel (endpoint), que permita a validacao dos tokens JWT emitidos;
- A garantia de que os tokens gerados sejam validos, nao expirados e contenham informacoes consistentes, corretas e veridicas.

Por outro lado, compete ao sistema da Republica de Mocambique:

- Registrar corretamente os clientes (IdPs externos) no sistema, configurando de forma adequada o mapeamento entre o valor do campo `iss` presente no token JWT e a URL de exposicao da chave publica correspondente;
- Garantir que apenas Provedores de Identidade previamente registrados, devidamente confiaveis e autorizados sejam aceitos para fins de validacao de tokens;
- Manter as configuracoes atualizadas sempre que houver alteracoes nos endpoints de exposicao das chaves publicas dos clientes integradores.

O correto funcionamento da integracao depende do cumprimento rigoroso das responsabilidades atribuidas a ambas as partes, assegurando elevados niveis de seguranca, interoperabilidade e confiabilidade no processo de assinatura e verificacao de documentos digitais.