1. Introdução

Esta documentação visa explicar e exemplificar os parâmetros aceitos e os respectivos retornos da `API` de assinatura de `PDF`s utilizando Certificados de Assinatura Única (CertAU) do SCDM.

1.1. Mudanças na documentação

Data	O que foi alterado
29/07/2025	Primeira versão.

2. URLs do serviço e requisitos para consumo da API

A `URL` da `API` de assinatura é a seguinte:

- `https://assinadoravancado.gov.mz/api/signer/pdf/<version>/<path>` (Produção)

Onde `<version>` corresponde à versão da `API` (veja mais em **2.1. Versionamento**) e `<path>` corresponde à ação específica dentro da `API` de assinatura. Logo, `<path>` pode ser substituído por:

- **sign**: assina um documento utilizando CertAU (detalhado em **3.2. Assinatura de documentos PDF**).
- **sign/hash/sha256/<hash_value>**: assina um *hash* de um documento gerado pelo algoritmo `SHA-256`.
- **sign/hash/sha512/<hash_value>**: assina um *hash* de um documento gerado pelo algoritmo `SHA-512`.

Os dois últimos *endpoints* são detalhados em **3.3. Assinatura com privacidade**.

2.1. Versionamento

Essa `API` sofrerá modificações sucessivas durante o seu ciclo de vida, de modo que, com o passar do tempo, novas funcionalidades serão adicionadas. De forma semelhante, funcionalidades podem ser removidas. Quaisquer mudanças que gerem incompatibilidade com versões anteriores serão lançadas como uma nova *major release*.

O formato de versionamento segue o padrão semver. Porém, o usuário informa apenas a versão `major` como exemplificado acima.

Exemplos:

- `https://assinadoravancado.gov.mz/api/signer/pdf/1/sign`: A versão com `major` igual a `1` mais atualizada será chamada.

IMPORTANTE: No momento, apenas a versão `1` está disponível.

OBS: No futuro próximo será disponibilizado documentação através do swagger.

2.2. Registro do IdP no INTIC e Formato do Token

2.2.1 Registro do IdP no INTIC

Antes de mais nada, o cliente (instituição lendo este documento) deve: (i) possuir um provedor de identidade capaz de gerar tokens JWT assinados digitalmente que represente a autenticação e autorização de um usuário para realizar assinaturas em seu nome; e (ii) registrar no INTIC a URL do provedor de identidade que publicamente informa, em formato JWK, a(s) chave(s) pública que o INTIC deve utilizar para verificar a assinatura digital dos tokens.

Exemplo de um endpoint expondo chaves públicas no formato JWK:


```
{
  "keys": [
    {
      "kid": "Z-Kk30jR2HNXmard3mDxN4MDLvksWsuDne8bcvxv4_g",
      "kty": "RSA",
      "alg": "RSA-OAEP",
      "use": "enc",
      "n": "rlMEidli43jRGeVHhdvS6... (omitido por brevidade)",
      "e": "AQAB",
      "x5c": [
        "MIIClzCCAX8CBgGRkN8tojANBgkq... (omitido por brevidade)"
      ],
      "x5t": "BeqAbBx47QtSFHNF9fPa5Vas8k4",
      "x5t#S256": "lJx-1RC6TFxdX2G3853bnq6LHYWJIQODrs8G97-OSdM"
    },
    {
```

```
      "kid": "NAGnMDubPi0pT1MTGPXZYCnPaLmT6b-FTAKGf-SjDGc",
      "kty": "RSA",
      "alg": "RS256",
      "use": "sig",
      "n": "whmU3RLAt0IT21... (omitido por brevidade)",
      "e": "AQAB",
      "x5c": [
        "MIIClzCCAX8CBgGRkN8r... (omitido por brevidade)"
      ],
      "x5t": "nSe6OST9H9mBWXDynqbdu2_cRdY",
      "x5t#S256": "GFIs7x5I1EQ3tBCZDAXDmKb8Z6xFqTOLiyCBLR_dK-0"
    }
  ]
}
```


O registro de URLs do provedor de identidade deve ser realizado através de solicitação para o Departamento de Licenciamento e Certificação do INTIC, i.e., enviando e-mail para `dlc@intic.gov.mz`. Deve ser enviado, obrigatoriamente, um token de exemplo (seguindo o formato especificado nesta documentação na seção 2.2.2), bem como a URL onde as chaves JWK serão disponibilizadas.

OBS: O processo de registro do novo IdP no sistema de assinatura digital envolve manipular o ambiente de produção do assinador, o que só será feito em horários oportunos. Desta forma, atender o vosso pedido pode levar alguns dias.

Para a utilização da `API` de assinatura, o *token* enviado deve: (i) seguir o formato especificado pelo INTIC (mais informações na seção 2.2.2); e (ii) a chave pública para verificação da assinatura **deve estar disponível publicamente na URL registrada previamente**.

2.2.2 Formato do token

2.2.2.1 Padronização dos atributos

Atributos obrigatórios (além daqueles especificados na RFC 7519):

- `iss`
- `name`
- `email`
- `birthdate`
- `nuic`, `nuit` ou `nuib`
- `exp`
- `iat`

OBS: no futuro o BI também será adicionado.

Atributos opcionais:

- `chosen_name`

2.2.2.2 Explicação dos atributos:

Atributos padronizados pelo OpenID Connect:

- `iss` -> URL do provedor de identidade
- `name` -> nome completo
- `email`
- `birthdate` -> no formato ISO 8601-1: `YYYY-MM-DD`, também aceito sem `-`

Atributos padronizados pelo RFC:

- `exp` -> Data de expiração do JWT (formato Unix Timestamp)
- `iat` -> Data de emissão do JWT (formato Unix Timestamp)

Atributos não padronizados pelo OpenID Connect:

- `nuic`, `nuit` ou `nuib` -> apenas letras e números
- `chosen_name` -> nome social

2.2.2.3 Lidando com conflitos

Caso a instituição já emita tokens e faça uso dos atributos especificados acima, permite-se que os atributos sejam prefixados com `intic_`, por exemplo:

- `intic_iss`
- `intic_name`
- `intic_email`
- `intic_birthdate`

- `intic_nuic`, `intic_nuit`, ou `intic_nuib`
- `intic_chosen_name`

A inclusão do prefixo `intic_` visa minimizar possíveis transtornos, garantindo que, caso alguma instituição utilize algum dos atributos especificados para outros propósitos ou em outro formato, as adaptações para uso do assinador via API sejam as menos intrusivas possível.

2.2.2.4 Ordem de interpretação dos atributos pelo assinador

Primeiramente, o assinador buscará por cada atributo com prefixo `intic_`. Caso não encontre, buscará pelo atributo sem prefixo. Por exemplo, buscará primeiro por `intic_nuic` e, não encontrando, prosseguirá para `nuic`.

É possível usar o prefixo `intic_` para alguns atributos e outros não. Fica a cargo do emissor do token decidir.

2.2.2.5 Obrigatoriedade dos atributos

Todos os atributos são obrigatórios, exceto `chosen_name` (ou `intic_chosen_name`). Todos os atributos obrigatórios devem estar contidos no token, com ou sem o prefixo `intic_`.

2.2.2.6 Exemplo de JWT

A seguir está um exemplo de JWT válido:

```
{
    "iss": "https://idp.assinadoravancado.gov.mz/realms/IDMZ",
    "sub": "5287c5ea-deba-45a2-9459-226fe0a560cb",
    "aud": "your-client-id",
    "exp": 1754763043,
    "iat": 1754762743,
    "name": "John Doe",
    "email": "john.doe@example.com",
    "birthdate": "1990-01-01",
    "nuit": "1234567890",
}
```

Vale comentar que o `exp` é a data de expiração do JWT que está no formato Unix Timestamp (algo como 2082758400). Já o `iat` é a data de emissão do JWT que também está no formato Unix Timestamp (algo como 1696012800)

A seguir está um exmplo de um JWT inválido:

```
{
    "iss": "https://idp.assinadoravancado.gov.mz/realms/IDMZ",
    "sub": "5287c5ea-deba-45a2-9459-226fe0a560cb",
    "aud": "your-client-id",
    "exp": 1754763043,
    "iat": 1754762743,
    "name": "John Doe",
    "email": "john.doe@example.com",
    "birthdate": "01-01-1990",
}
```

Observe que ele é inválido, pois não tem `nuic`, `nuit` ou `nuib` (pelo menos um destes atributos é obrigatório) e o `birthdate` está no formato errado.

3. Utilização do assinador

Na presente seção, trata-se da utilização do assinador, com exemplos.

3.1. Chamada de API

Para o serviço de assinatura, deve-se enviar uma requisição para a `URL` de assinatura (explicitada na seção 2), no *path* desejado. Segue um exemplo de uma requisição via cURL:

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@/home/edu/Downloads/arquivo.pdf" \
${opcoes_assinador} \
https://assinadoravancado.gov.mz/api/signer/pdf/1/sign \
--output documento_assinado.pdf
```

Na chamada acima, suponha que `$access_token` é uma variável de ambiente que armazena o *token* de acesso obtido pela aplicação após a autenticação do usuário no Provedor de Identidade.

Suponha também que `$opcoes_assinador` seja uma variável de ambiente que contenha opções do assinador (explicadas em **3.2.1 Opções do assinador para a rota `/sign`**).

O que se faz no restante da requisição é: especificar o caminho do arquivo a ser enviado para a assinatura, (em `-F "file=@/home/edu/Downloads/arquivo.pdf"`); especificar a `URL` com a rota de assinatura (nesse caso, em `https://assinadoravancado.gov.mz/api/signer/pdf/1/sign`) e finalmente, o nome com o qual o arquivo que retorna da chamada será salvo (em `--output documento_assinado.pdf`).

Segue um exemplo de requisição para a rota de assinatura de *hashes*:

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
https://assinadoravancado.gov.mz/api/signer/pdf/1/sign/hash/sha256/ \
ca978112ca1bbdcafac231b39a23dc4da786eff8147c4e72b9807785afee48bb
```

Veja que o padrão é o mesmo, apenas com o *hash* sendo informado junto ao *path* da rota. O_token_ de acesso continua sendo obrigatório.

3.2. Assinatura de documentos PDF (rota /sign)

Para a rota `/sign`, deve-se enviar o documento a ser assinado. Segue novamente um exemplo de requisição válida para a rota `/sign`:

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@contrato_de_estagio.pdf" \
${opcoes_assinador} \
https://assinadoravancado.gov.mz/api/signer/pdf/1/sign \
--output contrato_de_estagio_assinado.pdf
```

No exemplo, estamos enviando ao assinador o arquivo `contrato_de_estagio.pdf` e o retorno será salvo no arquivo `contrato_de_estagio_assinado.pdf`

Retorno: O retorno da rota `/sign` será um arquivo PDF assinado digitalmente, com artefato visual ou não, conforme as opções escolhidas pelos parâmetros configurados em `$opcoes_assinador`. Todos os parâmetros possíveis para customização são detalhados na seção **3.2.1. Opções do assinador para a rota `/sign`**

Segue um exemplo mais concreto de uma chamada à rota `/sign`:

```
curl -H "Authorization: Bearer \
eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJVVWZzcVpEb1pCaHRoVE50UTFzVElna2x1aEdnNDRkaWNNdzVOc0QtYk \
RIn0.eyJleHAiOjE2NzkzNjA2ODcsImlhdCI6MTY3OTM2MDM4NywiYXV0aF90aW1lIjoxNjc5MzU3NzQ2LCJqdGkiOiJiNzViZWZkYi1m \
NTJkLTRmYTktOTFhaXTGDRbht2nzmJ4fcdWvEXYPKCNVPXdgKTniMePU5q003OvVhcOtJgj3w_n1kvUrLzi5zxtv_JxM5mrh7CsJGyXhf \
Ri2THwRxxBOZ2_h5WQJA0l6T7rharDwFVvGeYwzUICkZ3iIPCQQtwyhdlTXJXOXUv3L4k1w9fd2yXeyOjajZd8nz38unhkA5elY9qXbJh \
szLhJdjPxvufnOYNVTeoCOLtxj5oVWXj2NaLw"
-X POST \
-F "stamp_enable=true" \
-F "file=@arquivo.pdf" \
https://assinadoravancado.gov.mz/api/signer/pdf/1/sign \
--output teste_assinado.pdf
```

Acima, `$access_token` está explícito. Ademais, a opção do assinador `"stamp_enable"` é enviada. Como será explicado no próximo tópico, ela habilita/desabilita a estampa visual da assinatura.

3.2.1 Opções do assinador para a rota `/sign`

O assinador conta com algumas opções que modificam características da assinatura (visualmente ou não). Para cada opção, o próprio assinador define valores padrão. Esses valores padrão ainda podem ser sobrescritos ao nível de variáveis de ambiente do serviço (nível de infraestrutura).

Porém, caso a aplicação deseje customizar sua assinatura, basta especificar o(s) parâmetro(s) na chamada de `API` à rota `/sign` do assinador. Nesse caso, os valores padrão serão sobrescritos pelos enviados via requisição.

Parâmetro	valor padrão	Valores possíveis	Descrição
sign_reason	None	string qualquer	Texto que explica o comprometimento do assinador perante o documento.

Parâmetro	valor padrão	Valores possíveis	Descrição
sign_contact_info	None ou valor definido na variável de ambiente SIGN_CONTACT_INFO	string qualquer	Contato do sistema assinador, que pode ser utilizado por um receptor do documento para proceder à validação do mesmo.
sign_user_auth_time	None	Número inteiro positivo	Inteiro que indica a estampa temporal (em Unix timestamp) da última autenticação do usuário no sistema.
stamp_enable	'False' ou valor definido na variável de ambiente STAMP_ENABLE	'True', 'False'	Define se a assinatura terá ou não uma estampa visual.
stamp_QR_code_enable	'False' ou valor definido na variável de ambiente STAMP_QR_CODE_ENABLE	'True', 'False'	Define se a assinatura terá ou não QR code na estampa
stamp_font_family	'Noto Sans' ou valor definido na variável de ambiente STAMP_FONT_FAMILY	'Noto Sans', 'Noto Serif Display', 'Noto Serif', 'Nunito', 'Playfair', 'Display', 'Poppins', 'Roboto'	Define o tipo de fonte a ser usado para a descrição textual da estampa
stamp_font_size	10 ou valor definido na variável de ambiente STAMP_FONT_SIZE	Número inteiro positivo	Define tamanho da fonte do texto.
stamp_border_width	2 ou valor definido na variável de ambiente STAMP_BORDER_WIDTH	Valores inteiros positivos	Define a largura da borda entorno da estampa
stamp_inner_scaling	'stretch-to-fit' ou valor definido na variável de ambiente STAMP_INNER_SCALING	'none', 'stretch-fill', 'stretch-to-fit', 'shrink-to-fit'	Define como a caixa de texto vai escalar (crescer) em relação ao container principal.
stamp_box_margins	'0, 0, 0, 0' ou valor definido na variável de ambiente STAMP_BOX_MARGINS	quádrupla de valores inteiros positivos	Define a margem da caixa principal da estampa. Os valores definem, respectivamente, as margens esquerda, direita, superior e inferior.
stamp_text_box_margins	'0, 0, 0, 0' ou valor definido na variável de ambiente STAMP_TEXT_BOX_MARGINS	quádrupla de valores inteiros positivos	Define a margem da caixa de texto da estampa. Os valores da quádrupla definem, respectivamente, margens esquerda, direita, superior e inferior.
stamp_text_box_vertical_align	'top' ou valor definido na variável de ambiente STAMP_TEXT_BOX_VERTICAL_ALIGN	'bottom', 'mid', 'top'	Define o alinhamento vertical do texto
stamp_text_box_horizontal_align	'left' ou valor definido pela variável de ambiente STAMP_TEXT_BOX_HORIZONTAL_ALIGN	'left', 'mid', 'right'	Define o alinhamento horizontal do texto
stamp_box_size	'287, 50' ou valor definido pela variável de ambiente STAMP_BOX_SIZE	tupla de valores inteiros	Define, respectivamente, a largura e a altura da estampa
stamp_qr_code_position	'right' ou valor definido pela variável de ambiente STAMP_QR_CODE_POSITION	'right', 'left', 'top', 'bottom'	Define a posição do QR code em relação à estampa
stamp_timestamp_format	'%d/%m/%Y %H:%M:%S' ou valor definido pela variável de ambiente STAMP_TIMESTAMP_FORMAT	formato de timestamp	Define o formato do timestamp que será mostrado na estampa com QR Code
stamp_align_by	'center' ou valor definido pela variável de ambiente STAMP_ALIGN_BY	'center', 'left'	Define a posição que será considerada para posicionar a estampa na página

Parâmetro	valor padrão	Valores possíveis	Descrição
stamp_align_x	-143 ou valor definido pela variável de ambiente STAMP_ALIGN_X	Número inteiro	Define o posicionamento horizontal da estampa a partir da posição definido em stamp_align_by
stamp_align_y	5 ou valor definido pela variável de ambiente STAMP_ALIGN_Y	Número inteiro positivo	Define o posicionamento vertical da estampa a partir da aresta inferior da página
stamp_background_rgba	(255, 255, 255, 255) ou valor definido pela variável de ambiente STAMP_BACKGROUND_RGBA	Define a cor e a opacidade do fundo da assinatura, que preenche toda a caixa principal.	
stamp_page	0 ou valor definido pela variável de ambiente STAMP_PAGE	Número que represente posições de páginas no documento	Define em qual página será feita a estampa
pades_allow_further_sign	'True' ou valor definido pela variável de ambiente PADES_ALLOW_FURTHER_SIGN	'True', 'False'	Define política de assinaturas futuras no documento
stamp_message	"Documento assinado digitalmente." ou valor definido pela variável de ambiente STAMP_MESSAGE	Texto	Texto que será exibido na estampa
stamp_url	"" ou valor definido pela variável de ambiente STAMP_URL	URL de verificação	URL que será utilizada para o redirecionamento pela estampa.
stamp_image (arquivo)	Não há padrão. Deve ser enviada via requisição.	Arquivo enviado via requisição (PNG, PDF, JPG)	Imagem que pode ser utilizada como estampa ou como fundo da estampa.
stamp_vertical_text_enable	'False' ou valor definido pela variável de ambiente STAMP_VERTICAL_TEXT_ENABLE	'True' ou 'False'	Habilita/desabilita texto escrito na vertical.
extra_information	Não há	String representando um JSON	Campo arbitrario de informações armazenadas nos logs de assinatura.
send_email	'False' ou valor definido pela variável de ambiente SEND_EMAIL	'True' ou 'False'	Habilita/desabilita o envio de emails quando um documento é assinado
attach_signed_document	'False' ou valor definido pela variável de ambiente EMAIL_ATTACH_SIGNED_DOCUMENT	'True' ou 'False'	Habilita/desabilita a anexação do documento assinado no e-mail enviado

Abaixo, são explicados todos os parâmetros citados na tabela, a(s) característica(s) que eles modificam na assinatura e os valores possíveis para cada um. Nos exemplos, omitir-se-á o *token* de acesso (supondo que exista uma variável de ambiente `$access_token` que o armazene), assim como a `URL` do serviço (supondo que exista uma variável de ambiente `$document_signer_url` que armazene essa `URL`).

3.2.1.1. sign_reason

Define o comprometimento do assinador perante o documento e/ou a razão pela qual a assinatura foi feita (ex.: 'Eu concordo com o que está descrito neste contrato'). Durante o processo de assinatura, essa descrição é salva no dicionário de assinatura da estrutura do `PDF`, mais especificamente na entrada `/Reason` desse dicionário.

Valores possíveis

- **'' (padrão)**: A entrada `/Reason` da assinatura não é preenchida.
- **String**: Texto que será colocado no campo `/Reason`.

Exemplo com sign_reason

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "sign_reason=Eu concordo com os termos do presente contrato" \
```

```
${document_signer_url}/sign \
--output teste_assinado.pdf"
```

Observe que o serviço de verificação, ao ler a assinatura, retorna essa informação, como explicado na seção **3.2.1** da documentação do serviço de verificação.

3.2.1.2. sign_contact_info

Define o contato do sistema de assinatura, para que o receptor do documento consiga buscar a fonte da assinatura, conseguindo realizar então a validação do documento. Essa descrição é armazenada, durante o processo de assinatura na API, no dicionário de assinatura do PDF, mais especificamente na entrada `/ContactInfo`. Assim como a opção explicada acima, esse parâmetro também é retornado no serviço de verificação, como explicado na seção **3.2.1** do serviço de verificação.

Valores possíveis

- **'' (padrão)**: A entrada `/ContactInfo` da assinatura não é preenchida.
- **String**: Texto que será colocado como contato do sistema de assinatura.

Exemplo com contact_info

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "sign_contact_info=https://assinadoravancado.gov.mz/" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```

3.2.1.3. sign_user_auth_time

Informa à `API` a estampa temporal da última autenticação do usuário no sistema (no padrão Unix timestamp). Esse *timestamp* é utilizado durante o processo de assinatura para o preenchimento da entrada `/Prop_AuthTime` do dicionário de assinatura, que informa o tempo, em segundos, desde a última autenticação do usuário no sistema.

Valores possíveis

- **'' (padrão)**: A entrada `/Prop_AuthTime` da assinatura não é preenchida.
- **Inteiro**: Número que será utilizado para cálculo da entrada `/Prop_AuthTime`.

O serviço verificador apresenta essa entrada do dicionário de assinatura em `"auth_time"`, como mostrado na seção **3.2.1** da documentação do verificador.

Exemplo com sign_user_auth_time

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "sign_user_auth_time=1693455424" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```

Note que o tempo está indicado no formato Unix Timestamp, e significa `quinta-feira, 31 de agosto de 2023 às 01:17:04 GMT-03:00`

3.2.1.4. stamp_enable

Define se a assinatura terá artefato visual (estampa). O valor padrão é 'False', mas se configurado como 'True' sem as demais opções referentes à estampa, a mesma será inserida com os valores padrão. O artefato visual pode ter diferentes características (a serem tratadas nos próximos parâmetros). Porém, nenhuma delas irá ser aplicada caso a estampa esteja desativada (i.e. caso este parâmetro seja indicado como False).

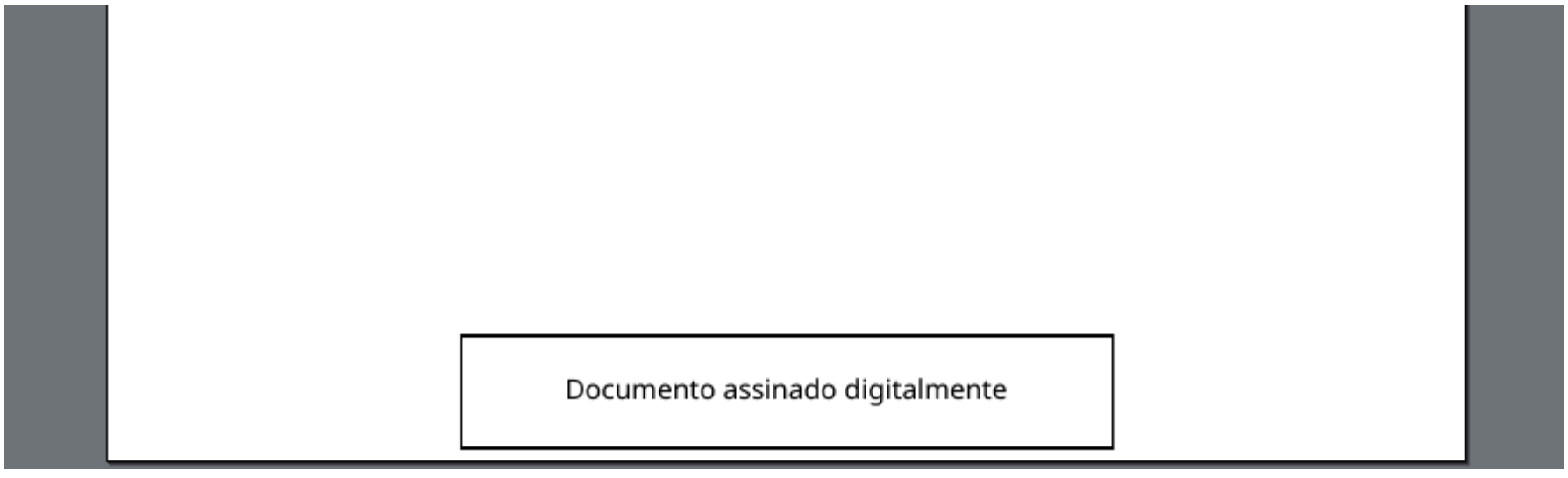
Valores possíveis

- **False (padrão):** Não haverá artefato visual na assinatura. Todos os outros parâmetros referentes à estilização da estampa serão ignorados.
- **True:** O artefato é habilitado e sua estilização se dará pela definição dos parâmetros que definem estas características (mostrados abaixo).

Exemplo com stamp_enable = True

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```

O resultado é mostrado abaixo. Como apenas a estampa foi ativada, os demais atributos são preenchidos com o valor padrão definido nos demais parâmetros.



Observação importante sobre os tipos de artefato do assinador

Como opções do assinador, temos dois tipos de artefato. Um deles é específico para estampas que contenham apenas informação textual. Esse tipo de estampa é ativado quando passamos o parâmetro `stamp_enable` com o valor `True`. Outra opção de artefato é para estampas que contenham tanto informação textual quanto um *QR Code*. Esta, por sua vez, é ativada quando habilitamos o parâmetro `stamp_QR_code_enable`, explicado em detalhes abaixo.

3.2.1.5. stamp_QR_code_enable

Define se a assinatura terá um QR code junto ao artefato visual. O QR code pode ter diferentes características (a serem tratadas nos próximos parâmetros referentes a ele). Porém, nenhuma delas irá ser aplicada caso o QR code esteja desativado (i.e. caso este parâmetro seja indicado como False).

Observação: Caso a estampa esteja desativada em `stamp_enable`, esse parâmetro será ignorado.

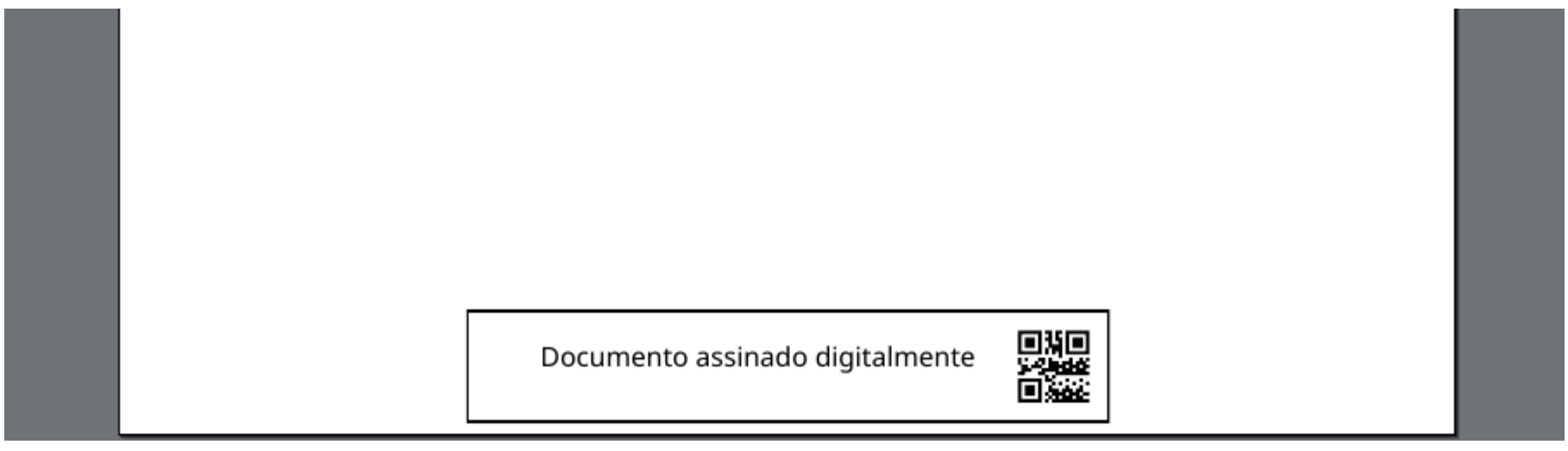
Valores possíveis

- **False (padrão):** Não haverá QR code. Todos os outros parâmetros referentes à estilização de QR code serão ignorados.
- **True:** O QR code é habilitado e sua estilização se dará pela definição dos parâmetros que definem estas características (mostrados abaixo).

Exemplo com stamp_QR_code_enable = True

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_QR_code_enable=True" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```

O resultado é mostrado abaixo, onde se vê o `QR Code` adicionado ao artefato visual.



Note que esse `QR Code` não aponta para lugar algum. Para setar uma `URL` de apontamento, o parâmetro `stamp_url` (explicado abaixo) deve ser utilizado.

3.2.1.6. stamp_font_family

Define a fonte que será utilizada na parte textual da estampa. Caso a estampa esteja desativada em `stamp_enable`, esse parâmetro será ignorado.

Observação: Por enquanto, o assinador suporta algumas fontes específicas, descritas abaixo.

Valores possíveis

Valor que definirá a fonte utilizada para a escrita do texto da estampa. Por enquanto, as seguintes fontes são suportadas:

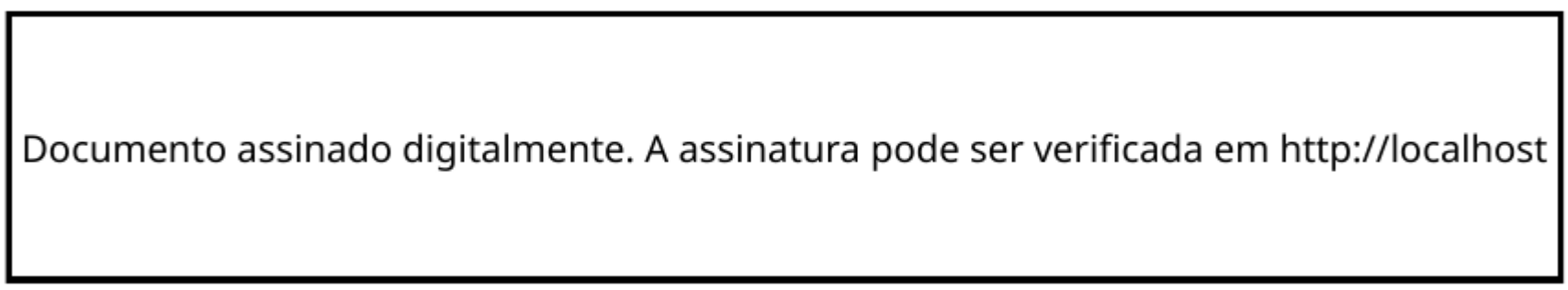
- **Noto Sans (padrão)**
- **Noto Serif Display**
- **Noto Serif**

- **Nunito**
- **Playfair Display**
- **Poppins**
- **Roboto**

Seguem exemplos com as possíveis fontes:

Exemplo com a fonte Noto Sans

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_font_family=Noto Sans" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```



Exemplo com a fonte Noto Serif Display

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_font_family=Noto Serif Display" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```

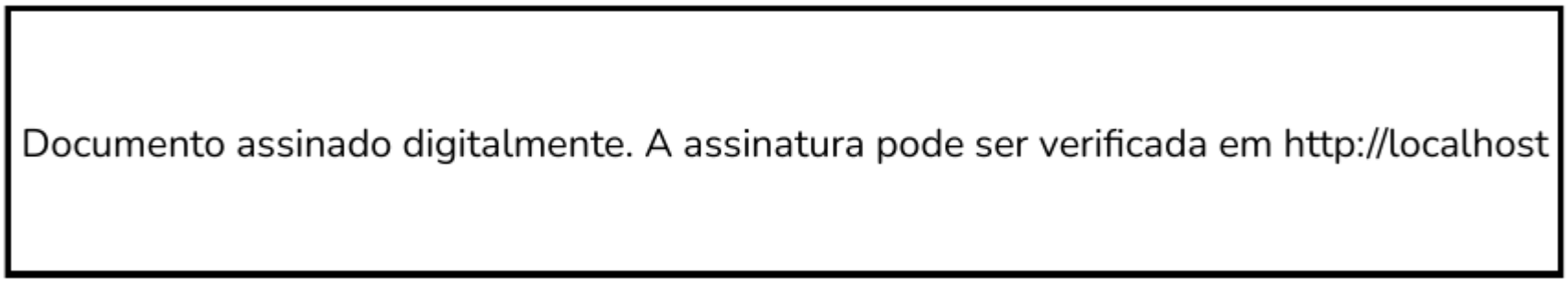
Exemplo com a fonte Noto Serif

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_font_family=Noto Serif" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```

Exemplo com a fonte Nunito

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
```

```
-F "stamp_font_family=Nunito" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```



Exemplo com a fonte Playfair Display

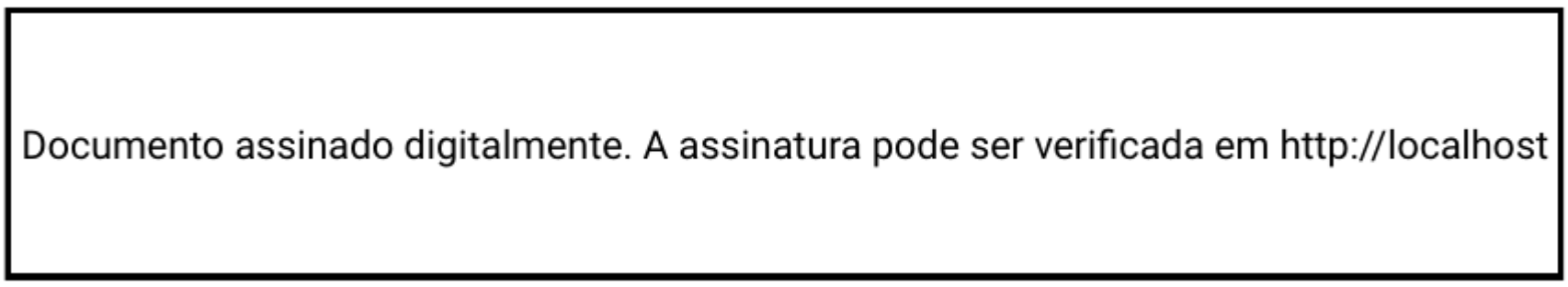
```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_font_family=Playfair Display" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```

Exemplo com a fonte Poppins

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_font_family=Poppins" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```

Exemplo com a fonte Roboto

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_font_family=Roboto" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```



3.2.1.7. stamp_font_size

Define o tamanho da fonte que será utilizada na parte textual da estampa. Caso a estampa esteja desativada em `stamp_enable`, esse parâmetro será ignorado. **Observação:** Esse valor pode ou não ser respeitado conforme o que for definido em `stamp_inner_scaling` (explicado abaixo e com exemplos utilizando o parâmetro `stamp_font_size`).

Valores possíveis

- **Números inteiros:** Número inteiro que define o tamanho da fonte do texto da estampa.

3.2.1.8. stamp_border_width

Define a largura da borda da caixa principal do artefato visual.

Observação: Caso a estampa esteja desativada em `stamp_enable`, esse parâmetro será ignorado.

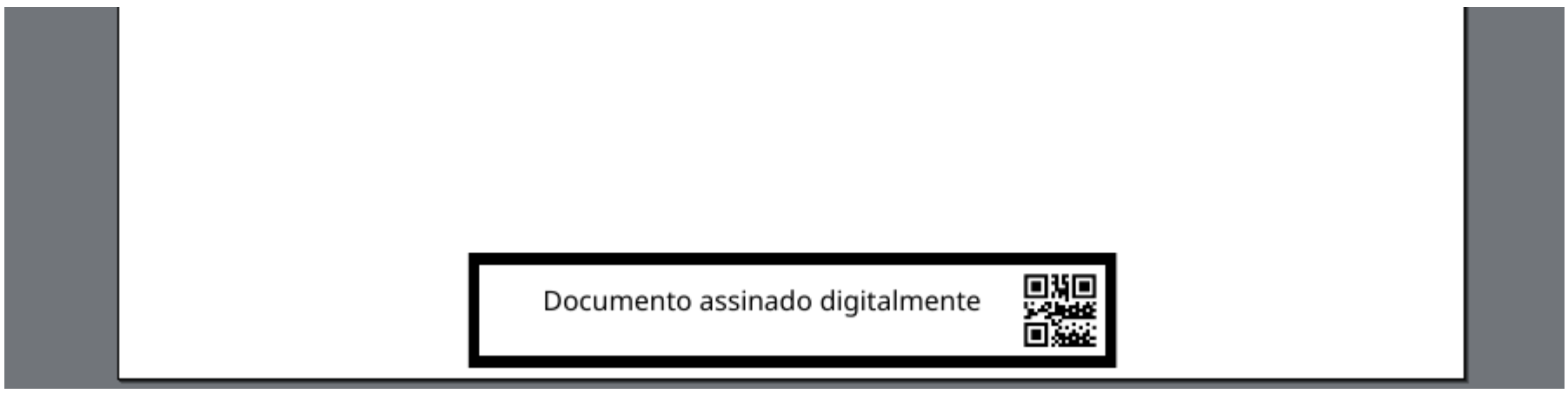
Valores possíveis

- **Números inteiros positivos:** Número que indique a largura da borda.

Exemplo com stamp_border_width = 10

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_QR_code_enable=True" \
-F "stamp_border_width=10" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```

O resultado é mostrado abaixo, onde se vê que o contorno da caixa está mais largo em relação aos exemplos anteriores (que usam largura padrão de 2).

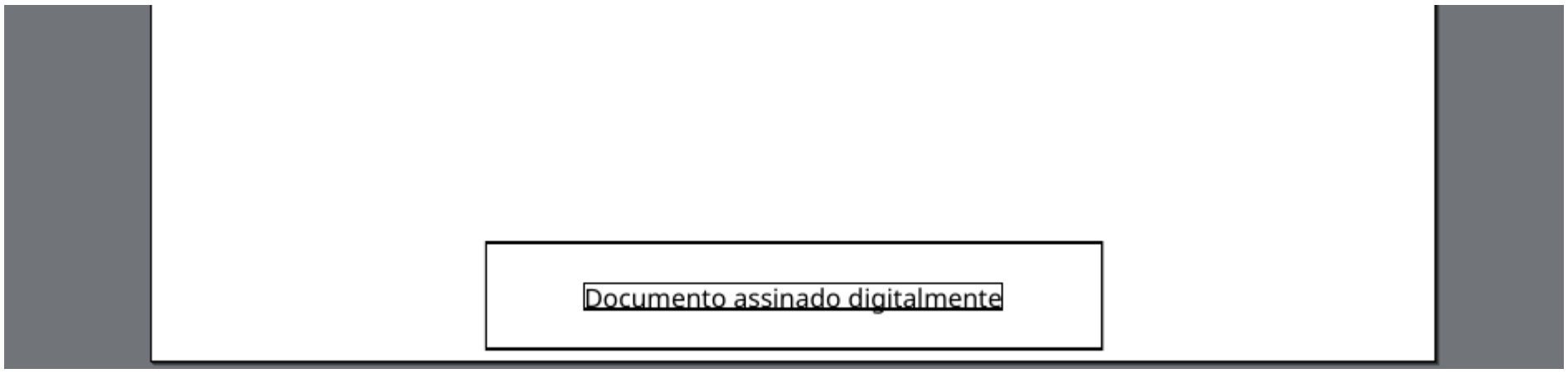


Observação importante sobre a estrutura do artefato visual

Antes de explicar as demais opções, deve-se observar como se dá a estilização da estampa visual da assinatura no assinador.

Primeiramente, temos uma **caixa principal**, que engloba todos os demais elementos visuais. Dentro dela, temos uma **caixa textual**, que engloba o texto que será adicionado ao artefato. Essa caixa textual pode ser posicionada e receber tamanhos diferentes de acordo com parâmetros explicados a seguir.

Segue uma ilustração dessas caixas.



Nos exemplos de algumas das opções abaixo, assim como na imagem acima, as caixas internas (caixas de texto) foram marcadas com uma borda, apenas para permitir que a visualização do alinhamento fosse possível. Essa borda da caixa textual é omitida das opções de assinatura e não há parâmetros para alteração dessa característica.

3.2.1.9. stamp_inner_scaling

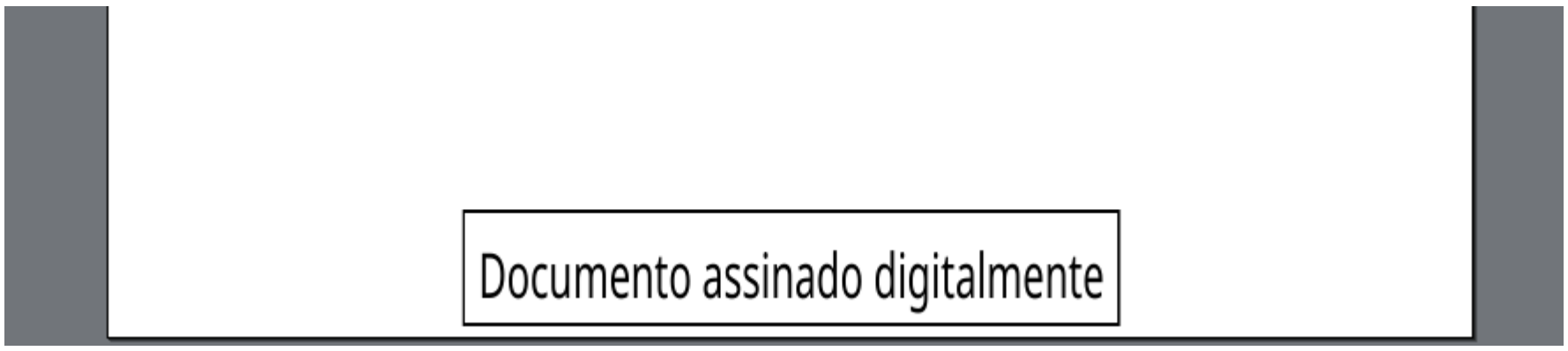
Inner Scaling define como a caixa de texto da estampa irá se comportar em relação à caixa principal em relação à escala (tamanho total).

Valores possíveis

- **stretch-fill (padrão):** Escala até preencher todo o *container*.
- **stretch-to-fit:** Escala o conteúdo da caixa interna até encontrar limite vertical ou horizontal da caixa principal.
- **shrink-to-fit:** Reduz o tamanho da caixa de texto, mantendo a relação altura x largura da caixa interna, quando a mesma ultrapassa o tamanho da caixa principal. Quando isso não ocorre, a caixa de texto seguirá com seu tamanho normal.
- **none:** Não escala a caixa de texto em relação à caixa principal.

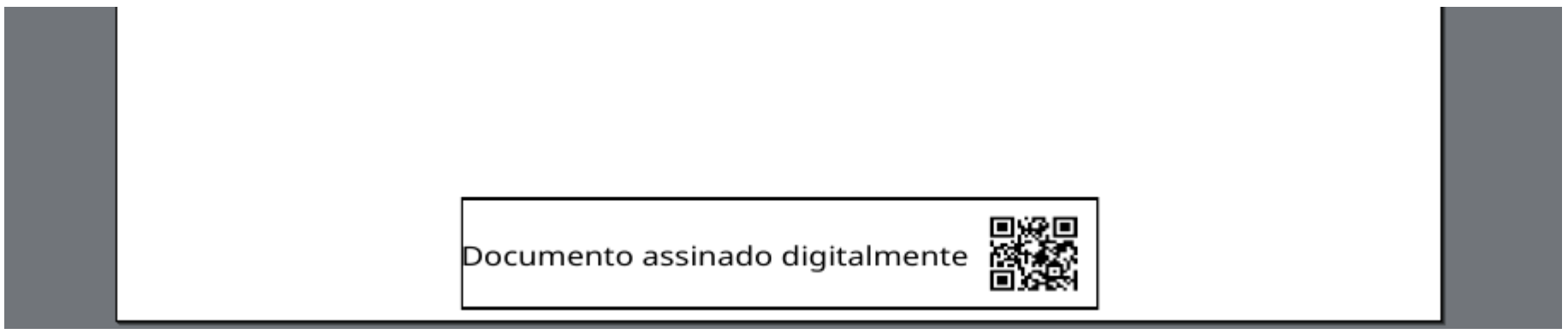
Exemplo com stamp_inner_scaling = stretch-fill

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_inner_scaling=stretch-fill" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```



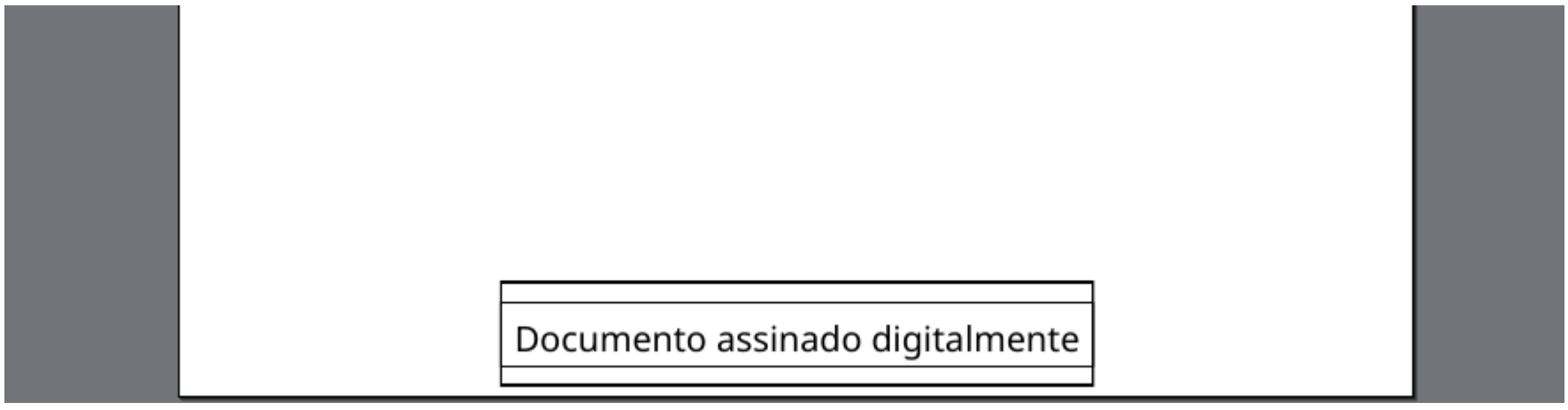
Note que a box interna cresceu tanto vertical quanto horizontalmente até achar os limites da caixa principal. Veja um exemplo usando QR code:

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_QR_code_enable=True" \
-F "stamp_inner_scaling=stretch-fill" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```



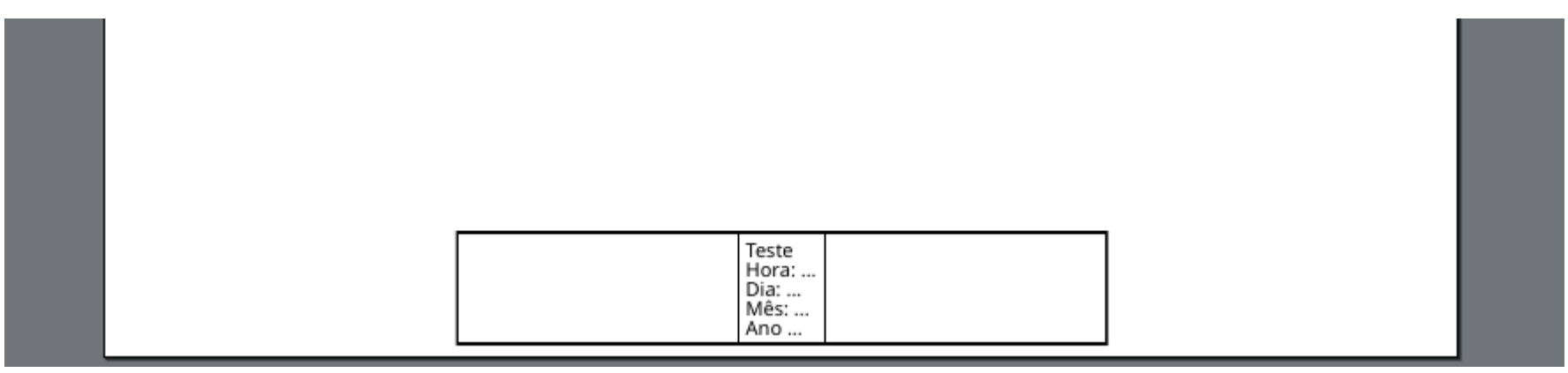
Exemplo com stamp_inner_scaling = stretch-to-fit

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_inner_scaling=stretch-to-fit" \
${document_signer_url}sign \
--output teste_assinado.pdf"
```



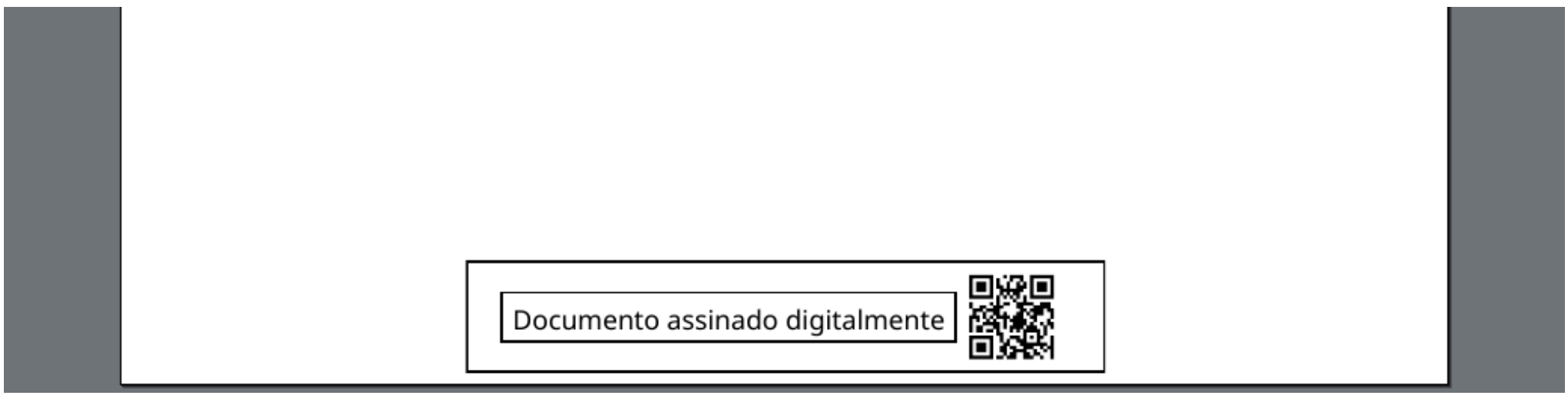
Note que a caixa interna foi escalada de forma que, como o texto é maior horizontalmente do que verticalmente, preencheu a caixa principal horizontalmente antes de preencher verticalmente. Veja outro exemplo em que a caixa textual preencherá a principal verticalmente.

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_inner_scaling=stretch-to-fit" \
-F "stamp_message=Teste\nHora: ...\nDia: ...\nMês: ...\nAno ..." \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```



Agora, um exemplo com `stamp_inner_scaling=stretch-to-fit` e QR code habilitado.

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_inner_scaling=stretch-to-fit" \
-F "stamp_QR_code_enable=True" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```

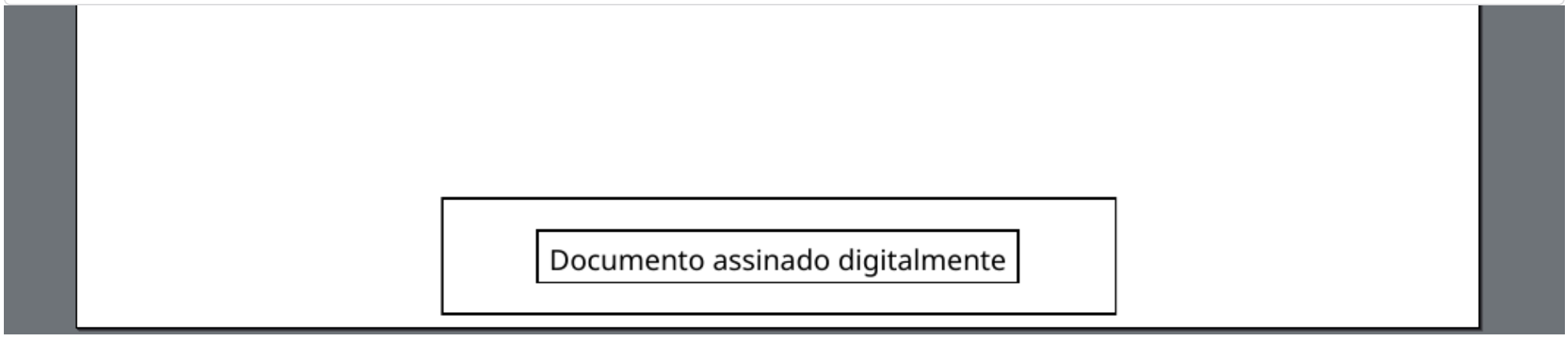


Note que a alteração parece não ter feito efeito sobre a caixa de texto. Porém, quando a opção QR code está habilitada, o próprio QR code é incluído como parte da caixa de texto para algumas regras específicas. Outra observação importante é que o QR code em si exige um espaçamento entre ele e demais artefatos visuais. Por isso, uma margem é automaticamente criada em torno do QR code quando o mesmo é habilitado. Com isso, visualmente, não há diferença entre utilizar `stretch-to-fit`, `shrink-to-fit` e None quando temos o QR code habilitado.

Exemplo com stamp_inner_scaling = shrink-to-fit

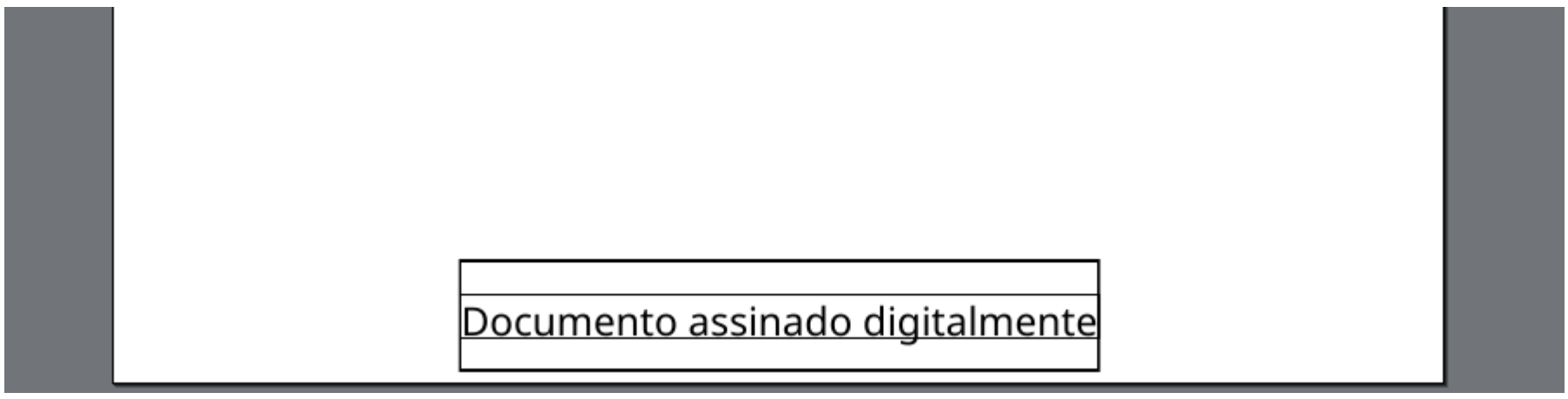
```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_inner_scaling=shrink-to-fit" \
```

```
${document_signer_url}/sign \
--output teste_assinado.pdf"
```



Notável que a caixa de texto manteve suas medidas, isso porque a opção `shrink-to-fit` prioriza o tamanho original da caixa. Apenas se ela ultrapassar o tamanho da caixa principal, será reduzida. Segue um exemplo em que o tamanho da fonte foi aumentado a um valor grande, o que, propositalmente, faria com que a caixa interna excedesse o tamanho da externa.

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_inner_scaling=shrink-to-fit" \
-F "stamp_font_size=99" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```



Note que, como a caixa interna seria maior e não caberia na externa, o parâmetro `shrink-to-fit` habilita a redução da caixa de texto, que, mantendo proporção entre suas dimensões, é diminuída para o tamanho da principal.

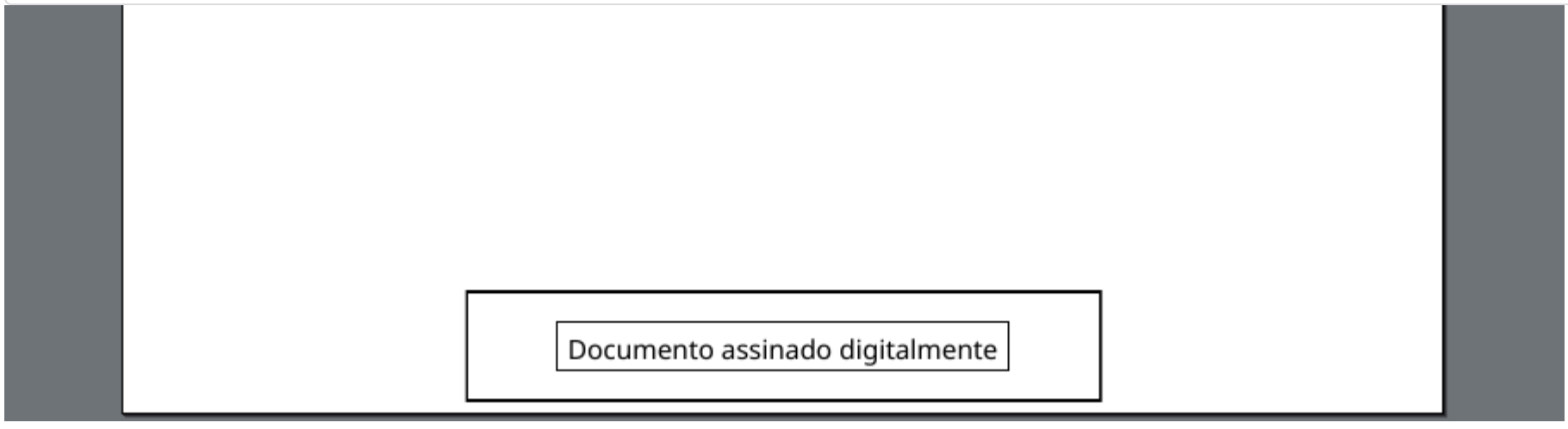
Exemplo com stamp_inner_scaling = none

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_inner_scaling=none" \
-F "stamp_font_size=99" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```

Veja que não houve redução da caixa de texto e a mesma vazou da caixa principal. Vejamos outro exemplo em que a caixa interna não será aumentada.

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_inner_scaling=none" \
```

```
${document_signer_url}/sign \
--output teste_assinado.pdf"
```



3.2.1.10. stamp_box_margins

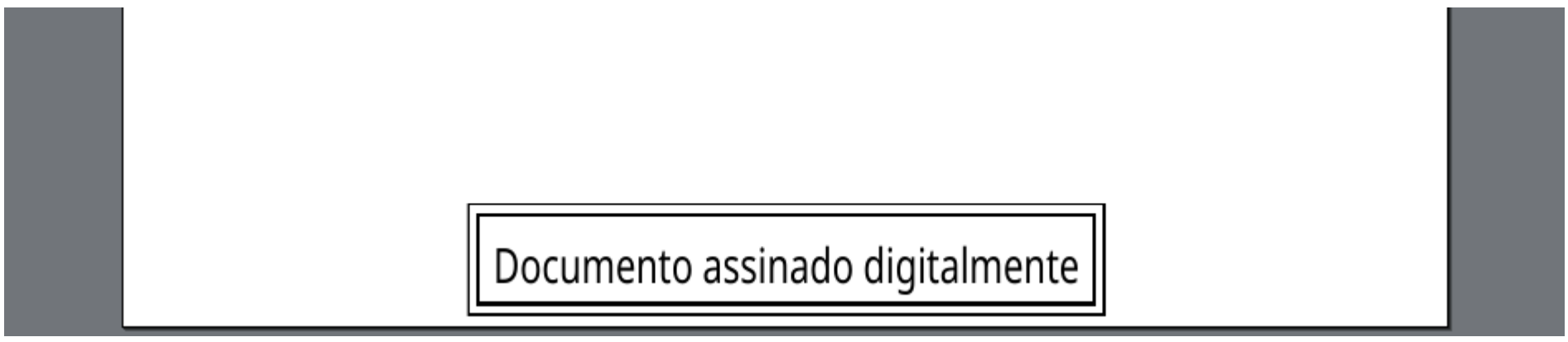
Define as margens esquerda, direita, superior e inferior da caixa principal da estampa, sendo que o valor padrão para todos eles é zero.

Valores possíveis

- (int, int, int. int): Valores inteiros que definem as margens esquerda, direita, superior e inferior.

Exemplo com stamp_box_margins = 5, 5, 5, 5

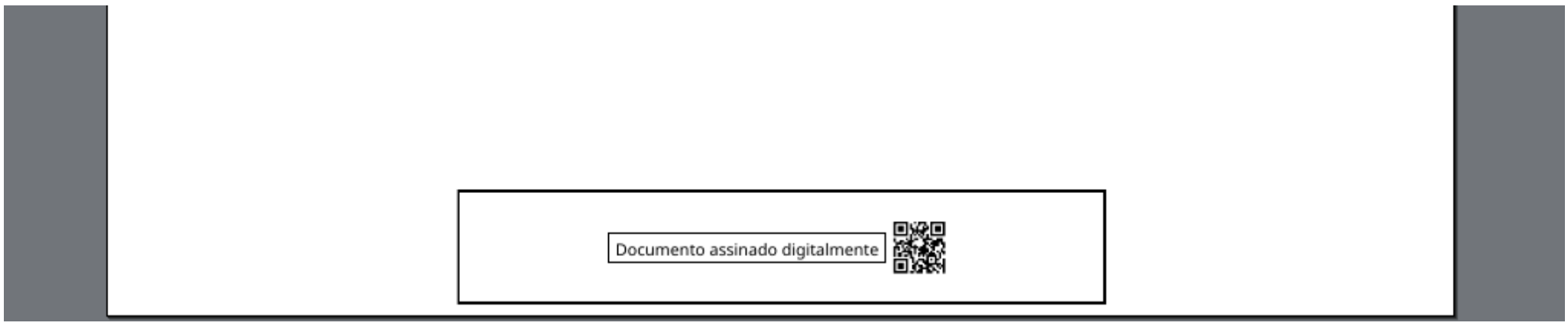
```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_inner_scaling=stretch-fill" \
-F "stamp_box_margins=5, 5, 5, 5" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```



Note que, mesmo passando o parâmetro `stamp_inner_scaling=stretch-fill`, que deveria, como no exemplo já apresentado, preencher a box principal com a box de texto. Porém, `stamp_box_margins` prevalece sobre o `stamp_inner_scaling`.

Veja agora um exemplo aplicando `stamp_box_margins` e utilizando QR code:

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_qr_code_enable=True" \
-F "stamp_box_margins=10, 10, 10, 10" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```



Nota-se que, além do espaço padrão do QR code, mais 10 unidades de borda foram adicionadas, o que resultou na diminuição do conjunto de texto e do QR code em si.

3.2.1.11. stamp_text_box_margins

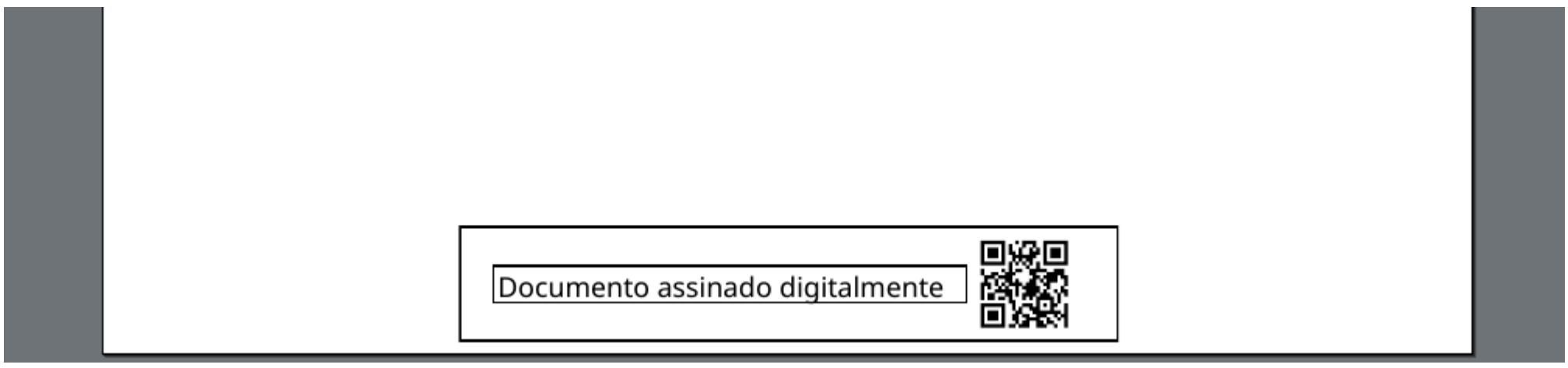
Também é possível inserir margens dentro do contexto da caixa interna, sendo que o valor do parâmetro passado corresponde às especificações de `stamp_box_margins`. O valor padrão desse atributo é 5, 5, 5, 5.

Valores possíveis

- (int, int, int. int): Valores inteiros que definem as margens esquerda, direita, superior e inferior.

Exemplo com stamp_text_box_margins = 0, 7, 0, 0 e QR code ativo

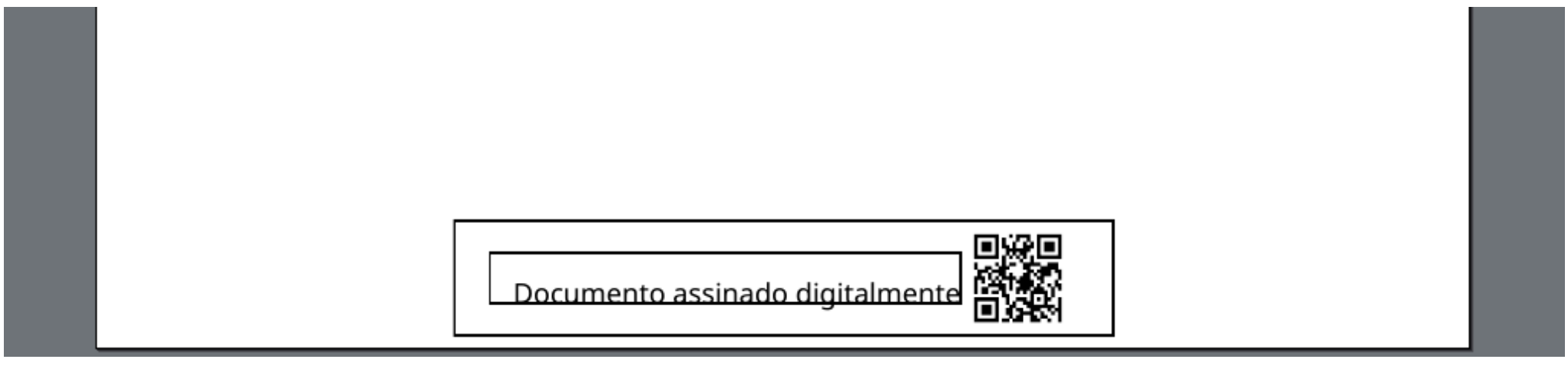
```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_qr_code_enable=True" \
-F "stamp_text_box_margins=2, 10, 2, 2" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```



Note que, neste caso, o desejo era adicionar um espaçamento à direita, para separar o texto do QR code.

Veja outro exemplo, em que as margens foram aplicadas na parte superior e esquerda.

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_qr_code_enable=True" \
-F "stamp_text_box_margins=10, 10, 0, 0" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```



3.2.1.12. stamp_text_box_vertical_align

Define o posicionamento vertical da caixa textual da estampa em relação à sua caixa principal.

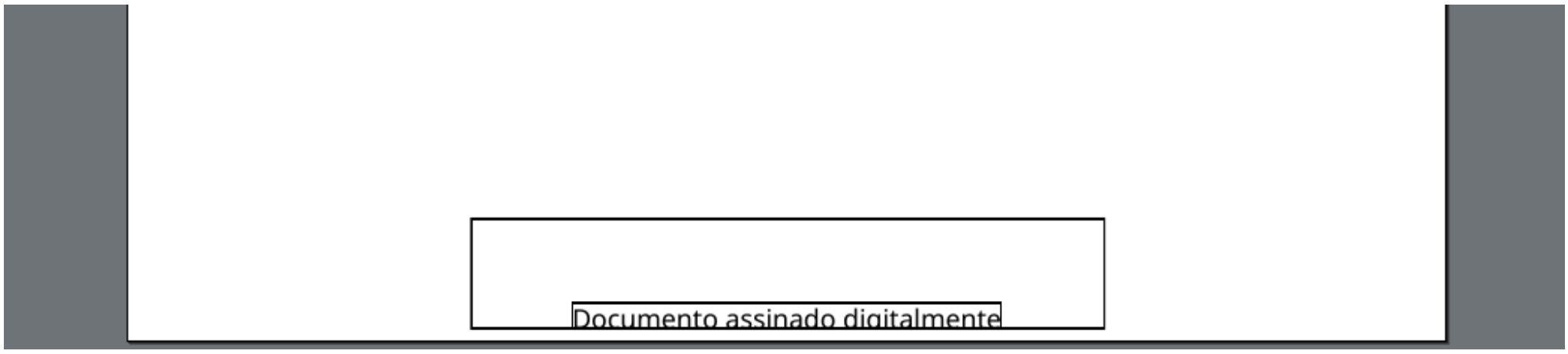
Observação: Caso a estampa esteja desativada em `stamp_enable`, esse parâmetro será ignorado. Caso o QR code esteja ativo, o QR code é considerado como parte integrante da caixa de texto e, portanto, essa opção não alterará o layout, como será visto em exemplo.

Valores possíveis

- **mid (padrão):** Caixa textual fica no centro da caixa.
- **bottom:** Caixa textual fica na parte inferior da caixa.
- **top:** Caixa textual fica na parte superior da caixa.

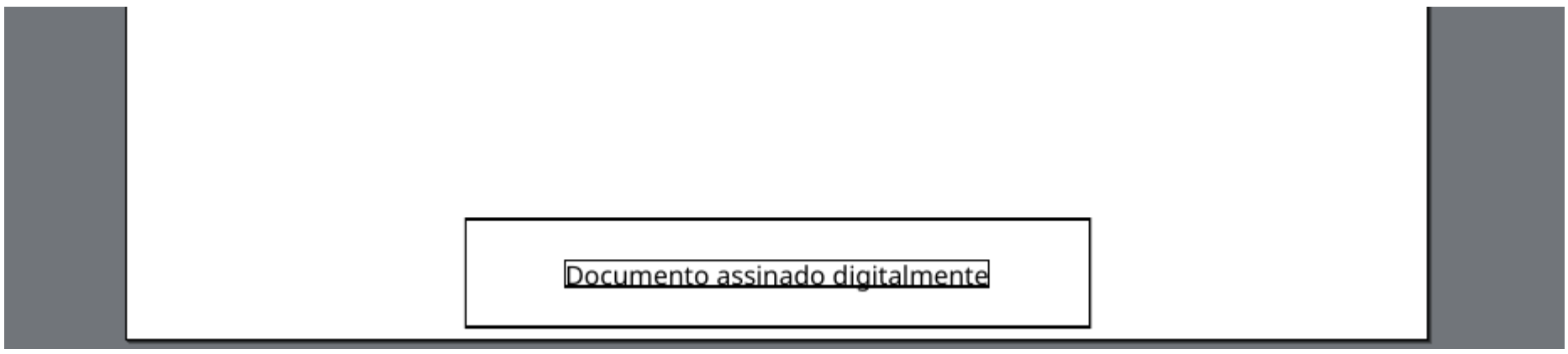
Exemplo com stamp_text_box_vertical_align = bottom

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_text_box_vertical_align=bottom" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```



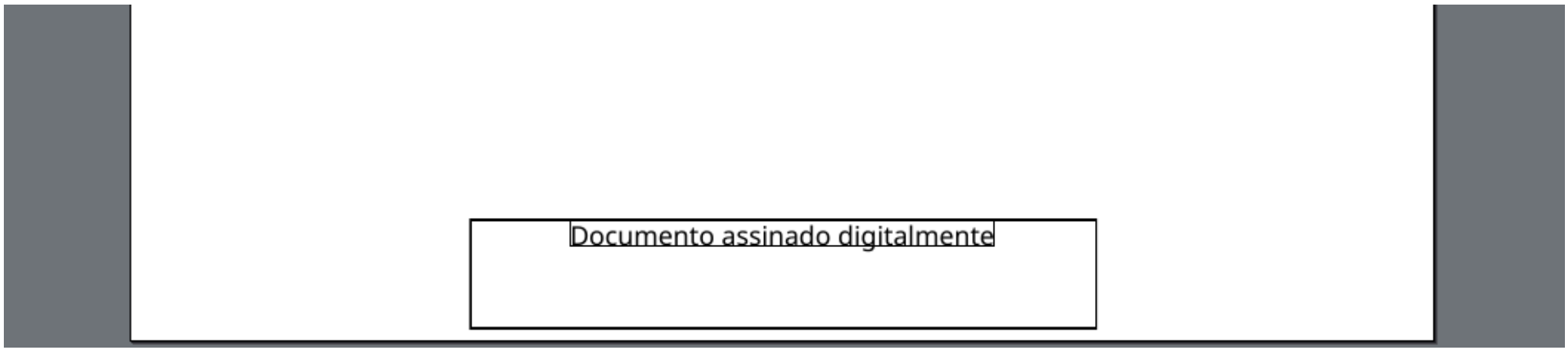
Exemplo com stamp_text_box_vertical_align = mid

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_text_box_vertical_align=mid" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```



Exemplo com stamp_text_box_vertical_align = top

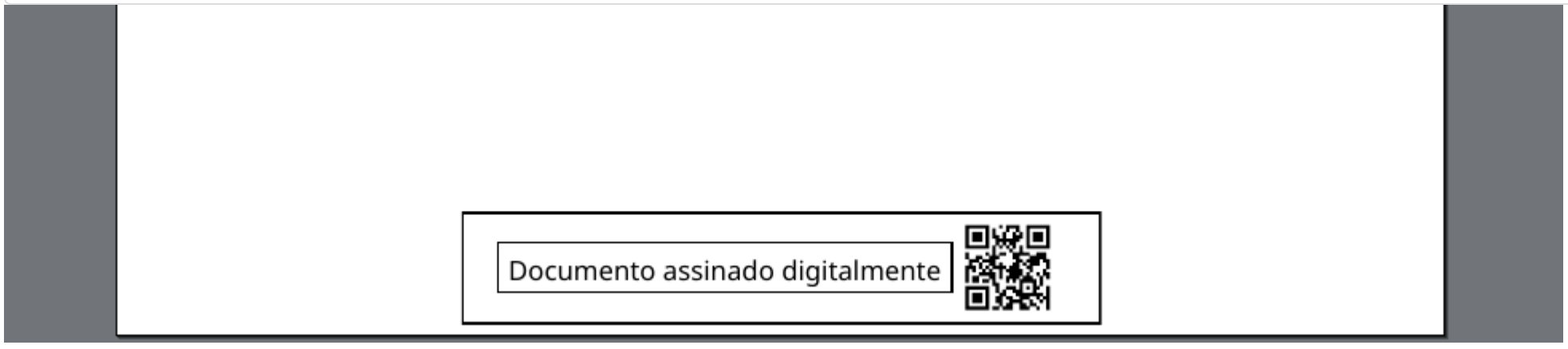
```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_text_box_vertical_align=top" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```



Exemplo com stamp_text_box_vertical_align = bottom e stamp_qr_code_enable=True

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_qr_code_enable=True" \
-F "stamp_text_box_vertical_align=bottom" \
```

```
${document_signer_url}/sign \
--output teste_assinado.pdf"
```



Observe que o QR code é entendido como a caixa interna, e que, por já receber seu tamanho conforme a caixa externa, mesmo não aparentemente, cumpre o `stamp_text_box_vertical_align=bottom`. O mesmo ocorre com os valores `top` e `mid`.

3.2.1.13. stamp_text_box_horizontal_align

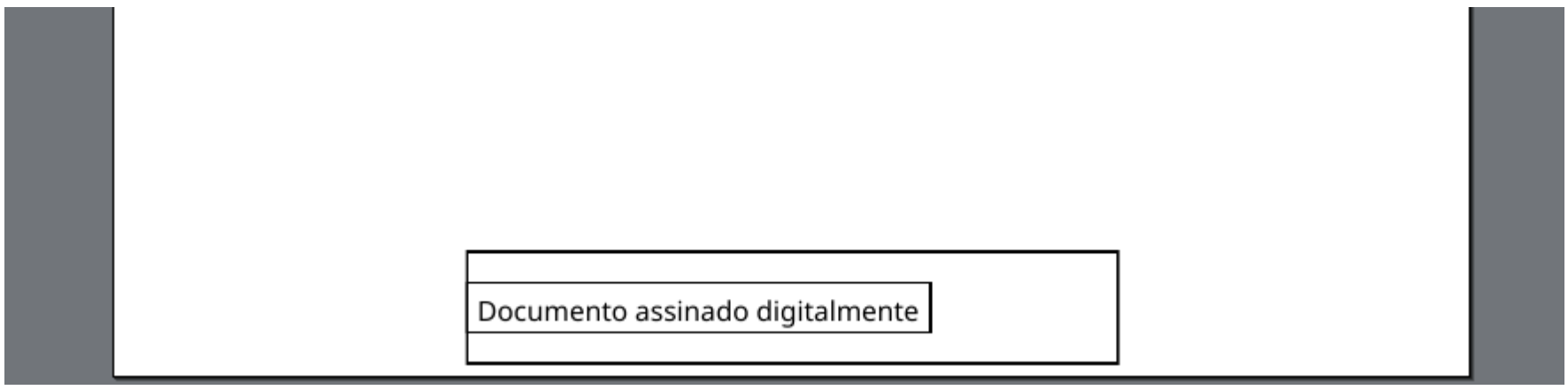
Esse parâmetro funciona da mesma forma que o anterior, porém trabalha com o alinhamento horizontal da caixa interna.

Valores possíveis

- **mid (padrão):** Caixa textual fica no centro da caixa principal.
- **left:** Caixa textual posicionada à esquerda da caixa principal.
- **rigth:** Caixa textual posicionada à direita da caixa principal.

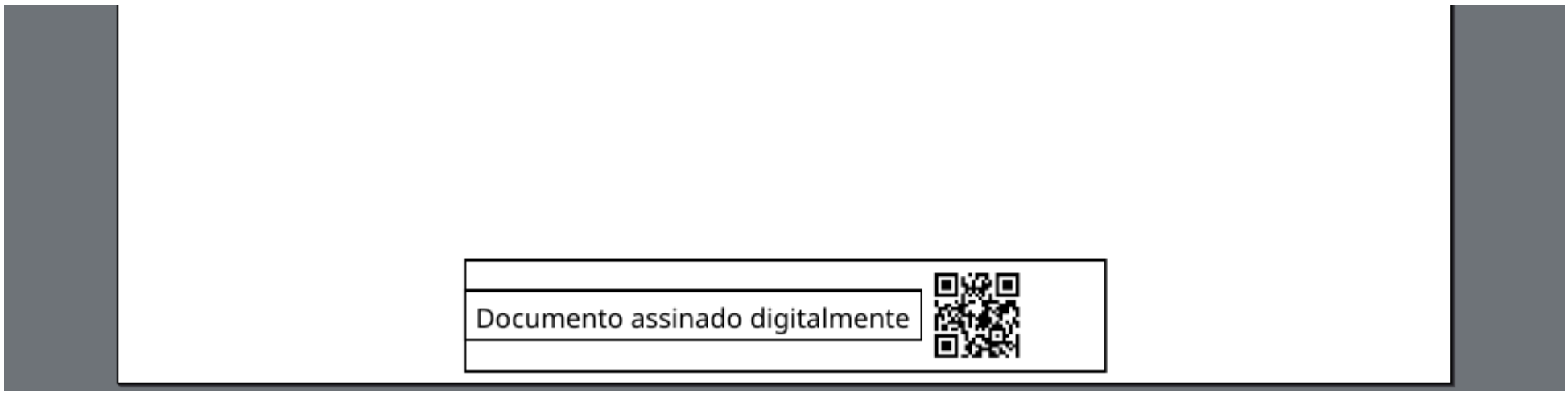
Exemplo com stamp_text_box_horizontal_align = left

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_text_box_horizontal_align=left" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```



Exemplo com stamp_text_box_horizontal_align=left e `stamp_qr_code_enable=True`

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_qr_code_enable=True" \
-F "stamp_text_box_horizontal_align=left" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```



Nota-se que, assim como na opção anterior, o QR code é integrado a box de texto para alinhamento seguindo o atributo. O resultado é que tanto a parte textual quanto o próprio QR code são posicionados à esquerda.

3.2.1.14. stamp_box_size

Esse parâmetro define o tamanho da caixa principal da assinatura nas duas dimensões, primeiro em x (horizontal) e depois em y (vertical). O valor padrão desse parâmetro é `287, 50`.

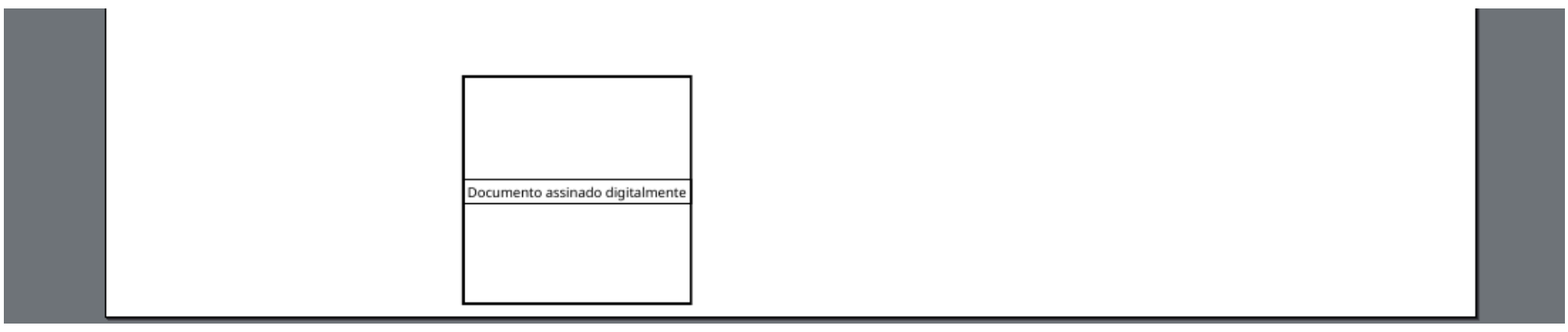
Observação: Se o valor passado for maior que o valor da dimensão da página, o assinador diminui automaticamente a caixa. Porém, deve-se ter cuidado ao utilizar o parâmetro `stamp_align_by`, que modifica o ponto de referência do posicionamento da box (próxima opção a ser explicada).

Valores possíveis

- **int, int:** Valores inteiros que definem o tamanho da caixa em x e y, respectivamente.

Exemplo com stamp_box_size = 100, 100

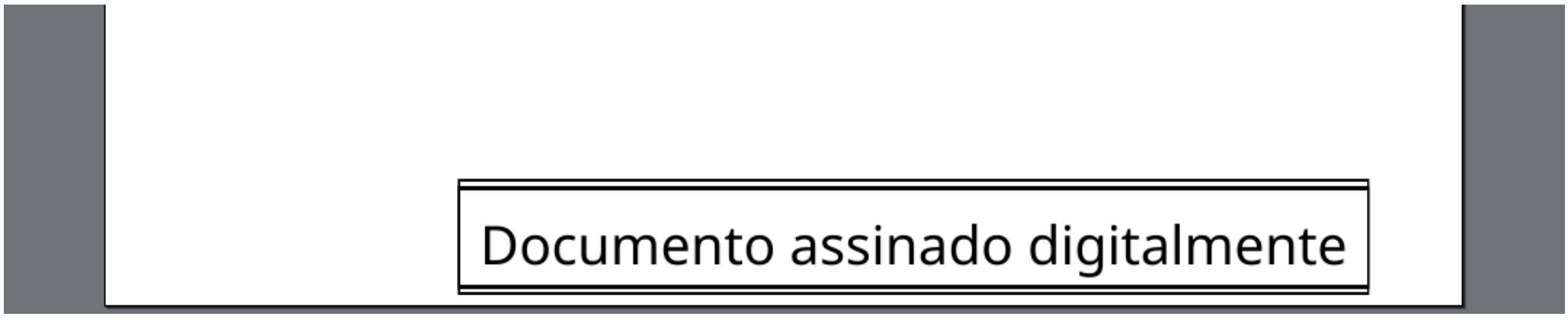
```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_box_size=100, 100" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```



Note que o tamanho da caixa externa setado foi menor que o tamanho original da caixa interior. Como o parâmetro `stamp_inner_scaling` é por padrão definido como `stretch-to-fit`, o conteúdo da caixa interior é ajustado ao tamanho da caixa externa.

Exemplo com stamp_box_size = 400, 50

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_box_size=400, 50" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```



3.2.1.15. stamp_qr_code_position

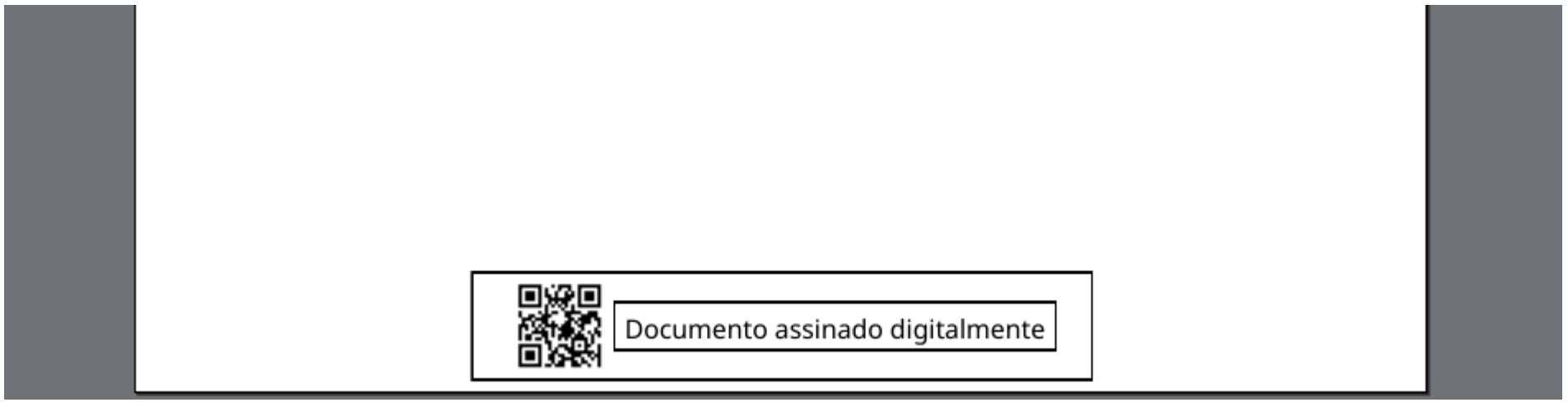
Determina a posição do QR code em relação à estampa. Pode ficar acima, abaixo, na esquerda ou direita.

Valores possíveis

- **top:** QR code fica posicionado no topo da estampa.
- **bottom:** QR code posicionado na parte inferior da estampa.
- **left:** QR code posicionado à esquerda da estampa.
- **right:** QR code posicionado à direita da estampa.

Exemplo com stamp_QR_code_position = left

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_QR_code_enable=True" \
-F "stamp_QR_code_position=left" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```



Exemplo com stamp_QR_code_position = top

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_QR_code_enable=True" \
-F "stamp_QR_code_position=top" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```

3.2.1.16. stamp_timestamp_format

Determina o formato do timestamp salvo na estampa.

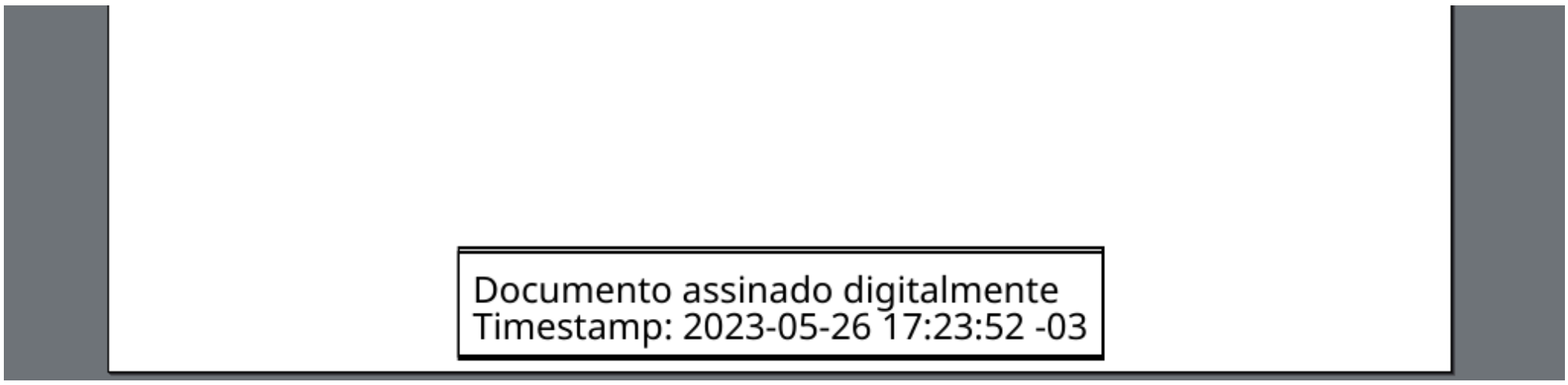
Valores possíveis

- **regex:** Expressão Regular que contenha o formato desejado para o *timestamp* da assinatura.

Observação importante sobre a ativação do *timestamp*: Para que o *timestamp* apareça na estampa, é necessário que, na mensagem prevista para o parâmetro `stamp_message` (explicado abaixo), seja adicionada a seguinte *string* de substituição: `%(ts)s`. Isso fará com que essa *string* seja substituída pelo *timestamp* formatado seguindo o formato setado neste parâmetro.

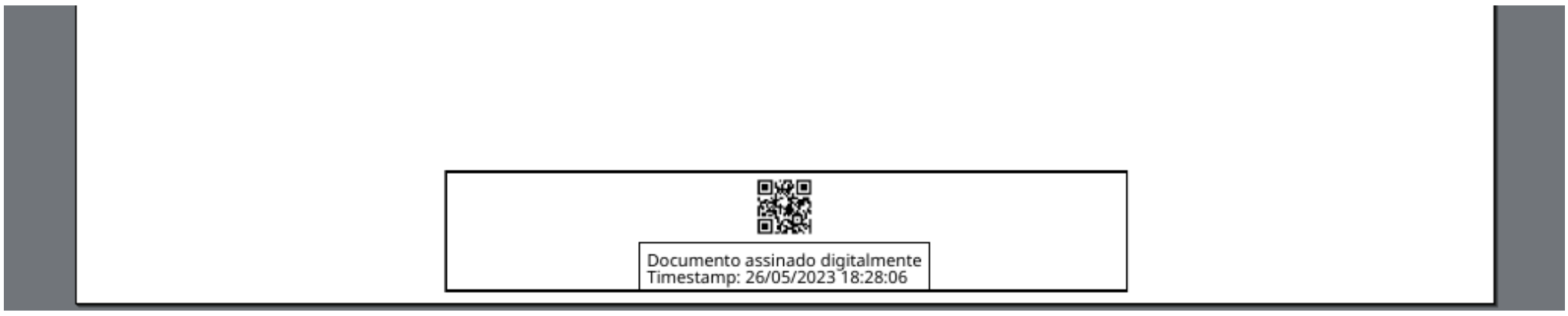
Exemplo com stamp_timestamp_format = %Y-%m-%d %H:%M:%S %Z

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_timestamp_format=%Y-%m-%d %H:%M:%S %Z" \
-F \"stamp_message=Documento assinado digitalmente\nTimestamp: %(ts)s" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```



Exemplo com stamp_timestamp_format = %d/%m/%Y %H:%M:%S

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_QR_code_enable=True" \
-F "stamp_QR_code_position=top" \
-F "stamp_timestamp_format=%d/%m/%Y %H:%M:%S" \
-F "stamp_message=Documento assinado digitalmente\nTimestamp: %(ts)s" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```



3.2.1.17. stamp_align_by

Parâmetro que define o ponto de partida do alinhamento horizontal da caixa principal da assinatura em relação à página do documento PDF. Por padrão, a caixa é alinhada à esquerda, com `stamp_align_by=left`.

Valores possíveis

- **left**: Ponto de partida do alinhamento é a borda esquerda da página do PDF.
- **center**: Ponto de partida do alinhamento é o centro horizontal da página do PDF.

Observação: Esse alinhamento horizontal com ponto de partida configurado neste parâmetro depende também do valor do atributo `stamp_align_x`. Para os exemplos, `stamp_align_x` será utilizado com o valor zero.

Exemplo com stamp_align_by = center

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_align_x=0" \
-F "stamp_align_by=center" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```

Note que o posicionamento da caixa principal na horizontal é feito a partir do centro da página. Como se tem zero de deslocamento, a borda esquerda da box fica exatamente no centro da página.

Exemplo com stamp_align_by = left

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_align_x=0" \
-F "stamp_align_by=left" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```

3.2.1.18. stamp_align_x

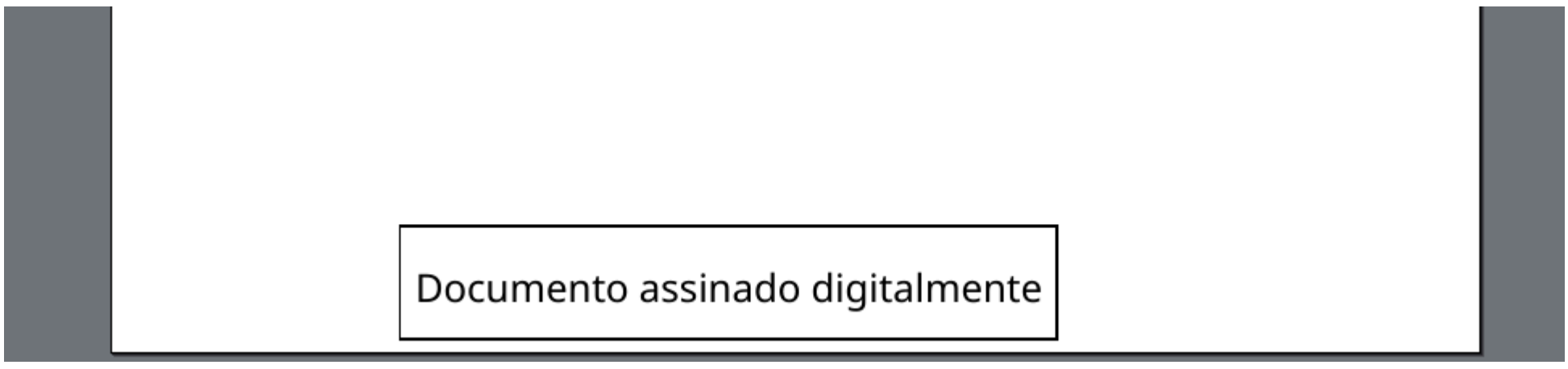
Esse parâmetro define o deslocamento da estampa horizontalmente a partir do ponto de partida definido em `stamp_align_by`.

Valores possíveis

- **valores positivos**: Valor inteiro positivo que define o deslocamento da box da estampa.

Exemplo com stamp_align_by = left e stamp_align_x = 50

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_align_x=125" \
-F "stamp_align_by=left" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```



Neste caso, vê-se que a caixa foi movida 125 unidades, na horizontal, a partir da esquerda.

Exemplo com stamp_align_by = center e stamp_align_x = 45

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_align_x=45" \
-F "stamp_align_by=center" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```

Neste segundo exemplo, a partir do centro horizontal da página, a caixa é movida 45 unidades para a direita.

3.2.1.19. stamp_align_y

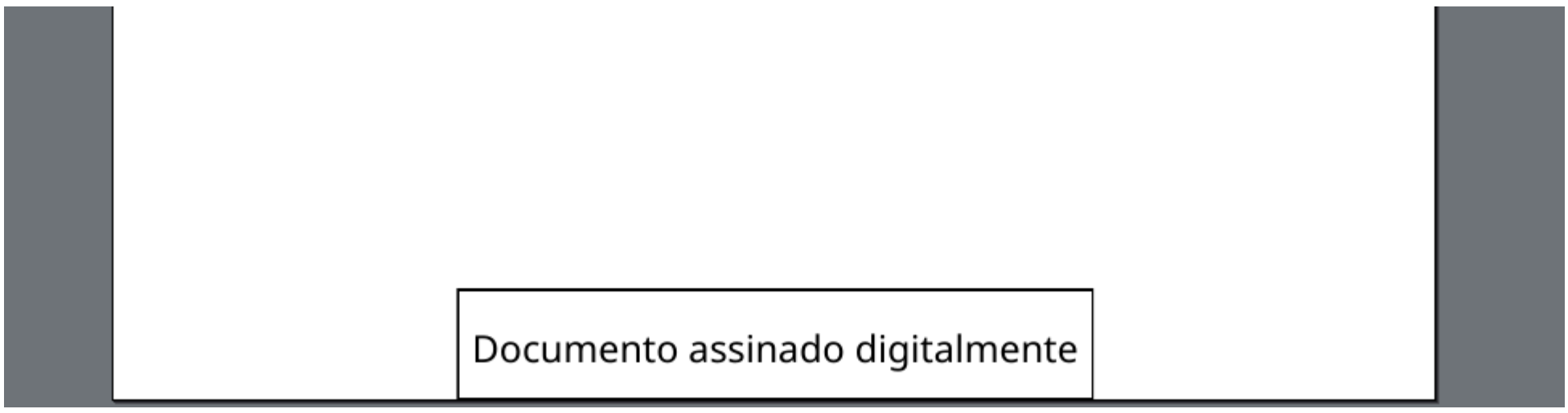
Esse parâmetro define o deslocamento vertical da caixa principal da estampa em relação à borda inferior da página.

Valores possíveis

- **valores positivos**: Valor inteiro positivo que define o deslocamento vertical da box principal da estampa em relação à borda inferior da página.

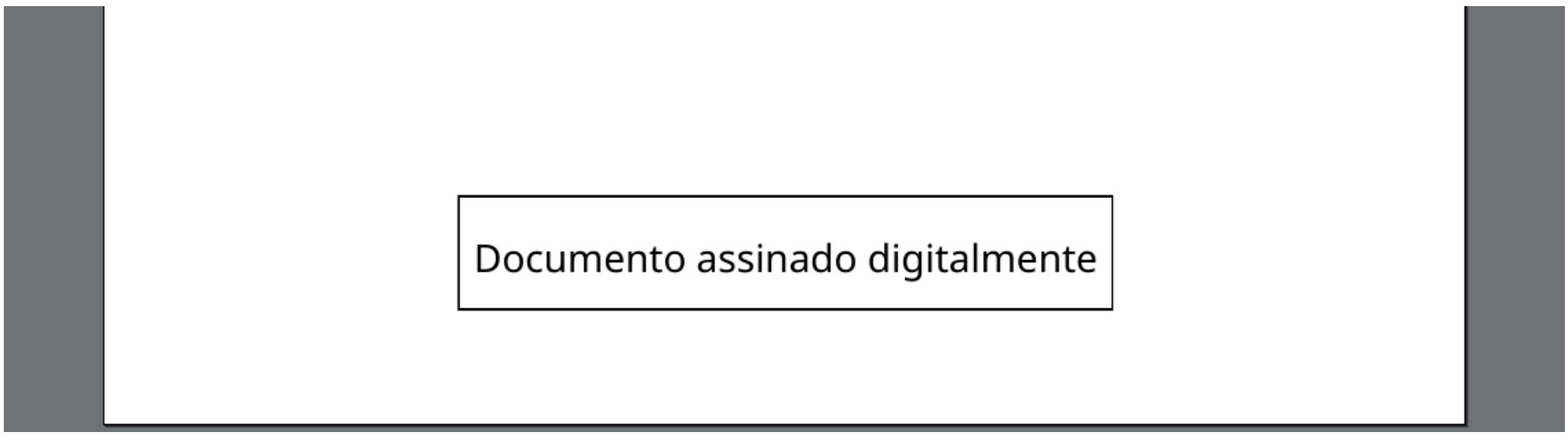
Exemplo com stamp_align_y = 0

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_align_y=0" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```



Exemplo com stamp_align_y = 50

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_align_y=50" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```



3.2.1.20. stamp_background_rgba

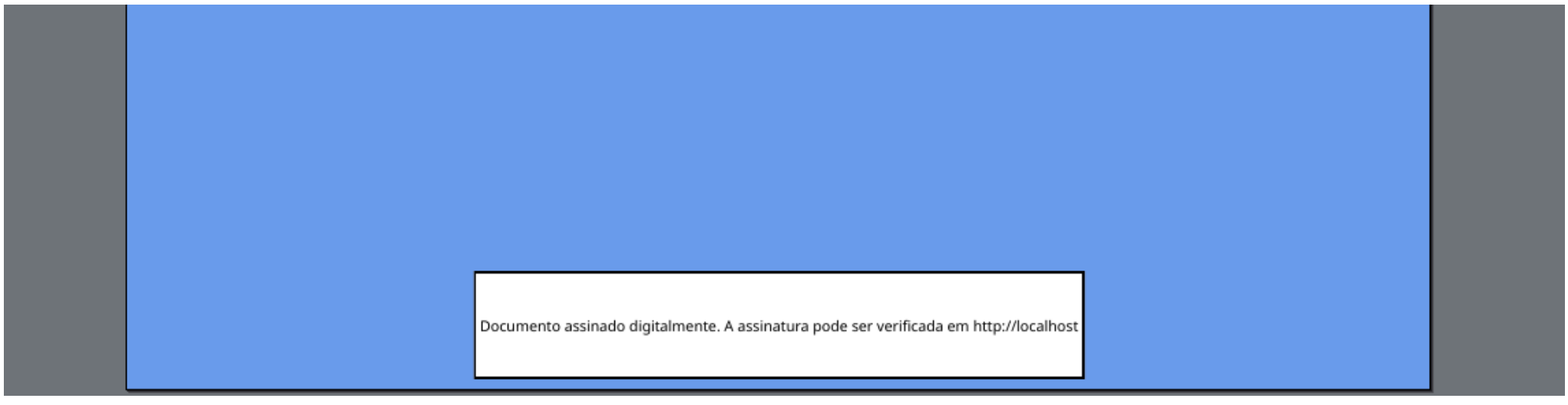
Esse parâmetro descreve a cor e a opacidade, no formato RGBA, do fundo da estampa da assinatura.

Valores possíveis

- **(int, int, int, int)**: Valor inteiro positivo entre 0 e 255, que descreve, em ordem, os tons de vermelho (**R**ed), verde (**G**reen), azul (**B**lue) e opacidade (**A**lpha) do fundo da assinatura.

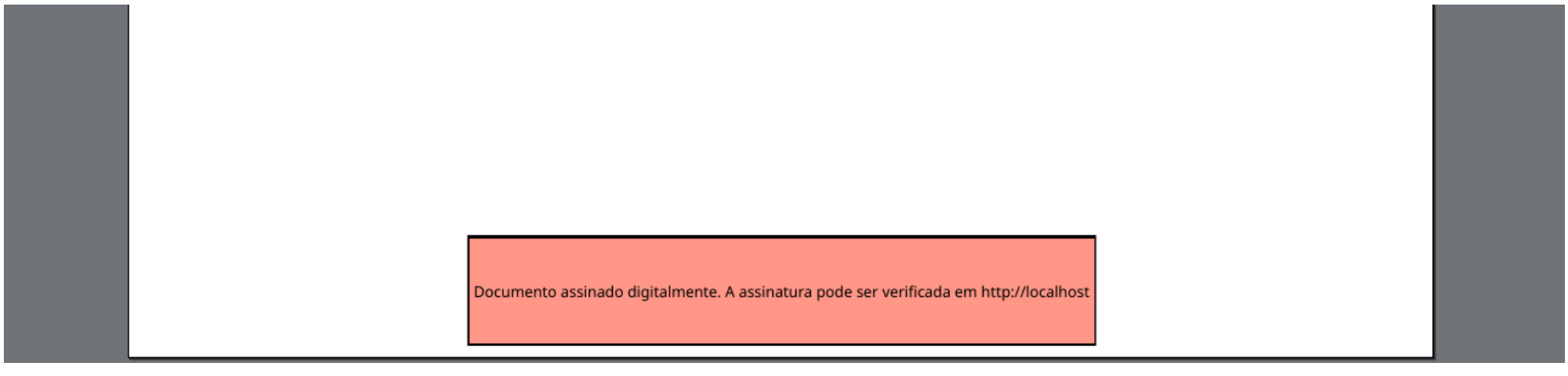
Exemplo com stamp_background_rgba = 255, 255, 255, 255

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_page=0" \
-F "stamp_background_rgba=255, 255, 255, 255" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```



Exemplo com stamp_background_rgba = '255, 99, 71, 165'

```
curl -H "Authorization: Bearer {access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_page=0" \
-F "stamp_background_rgba=255, 99, 71, 165" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```



3.2.1.21. stamp_page

Esse atributo configura qual a página do documento PDF em que a estampa da assinatura será inserida, iniciando por zero.

Observação: Caso o valor seja negativo, a estampa será inserida contando as páginas do documento de trás para frente. Ou seja, caso o valor seja -1, o artefato visual será inserido na última página, se o valor for -2, na penúltima e assim por diante.

Valores possíveis

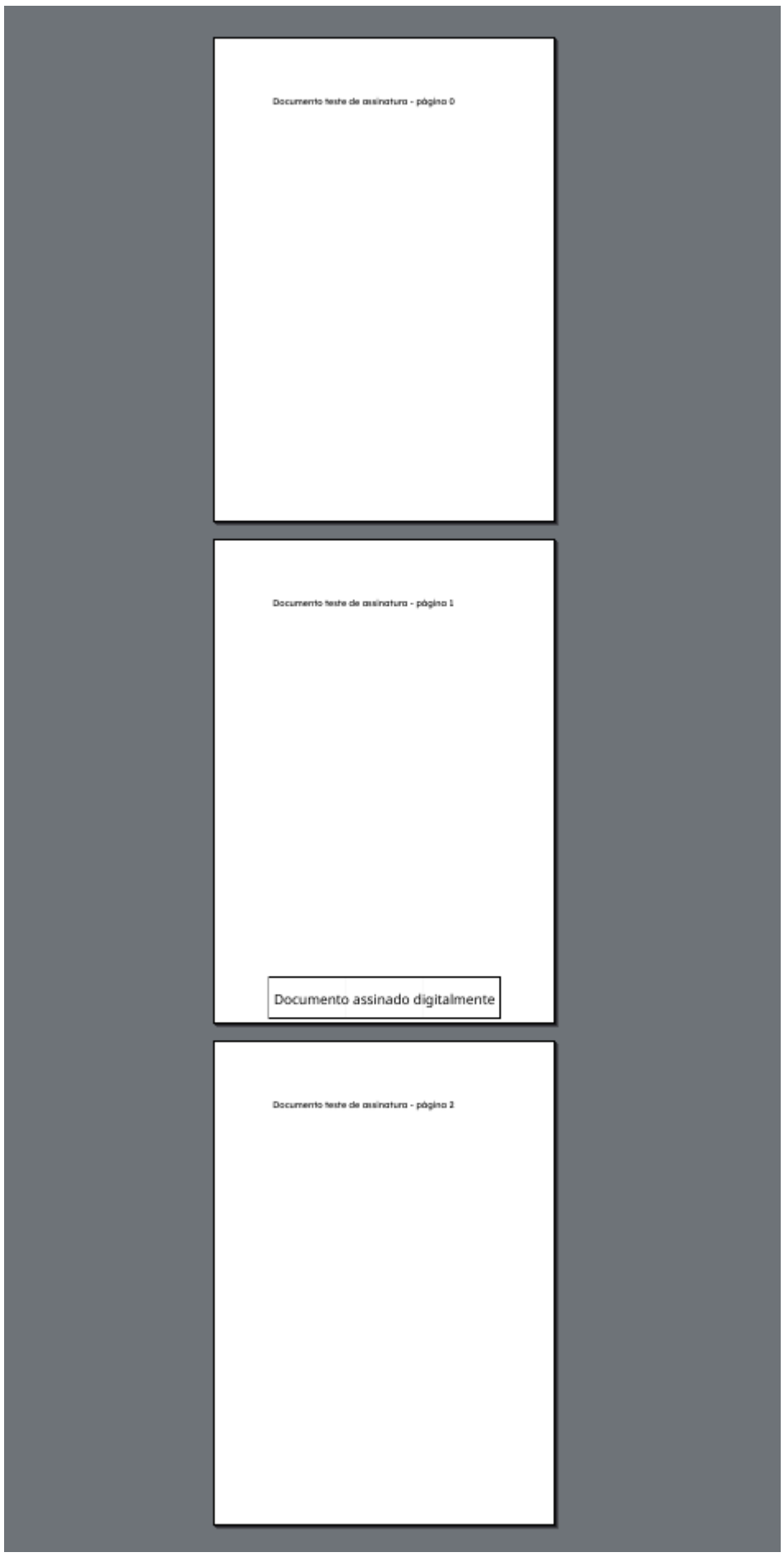
- **valores inteiros**: Valor inteiro positivo ou negativo.

Exemplo com stamp_page = 0

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_page=0" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```

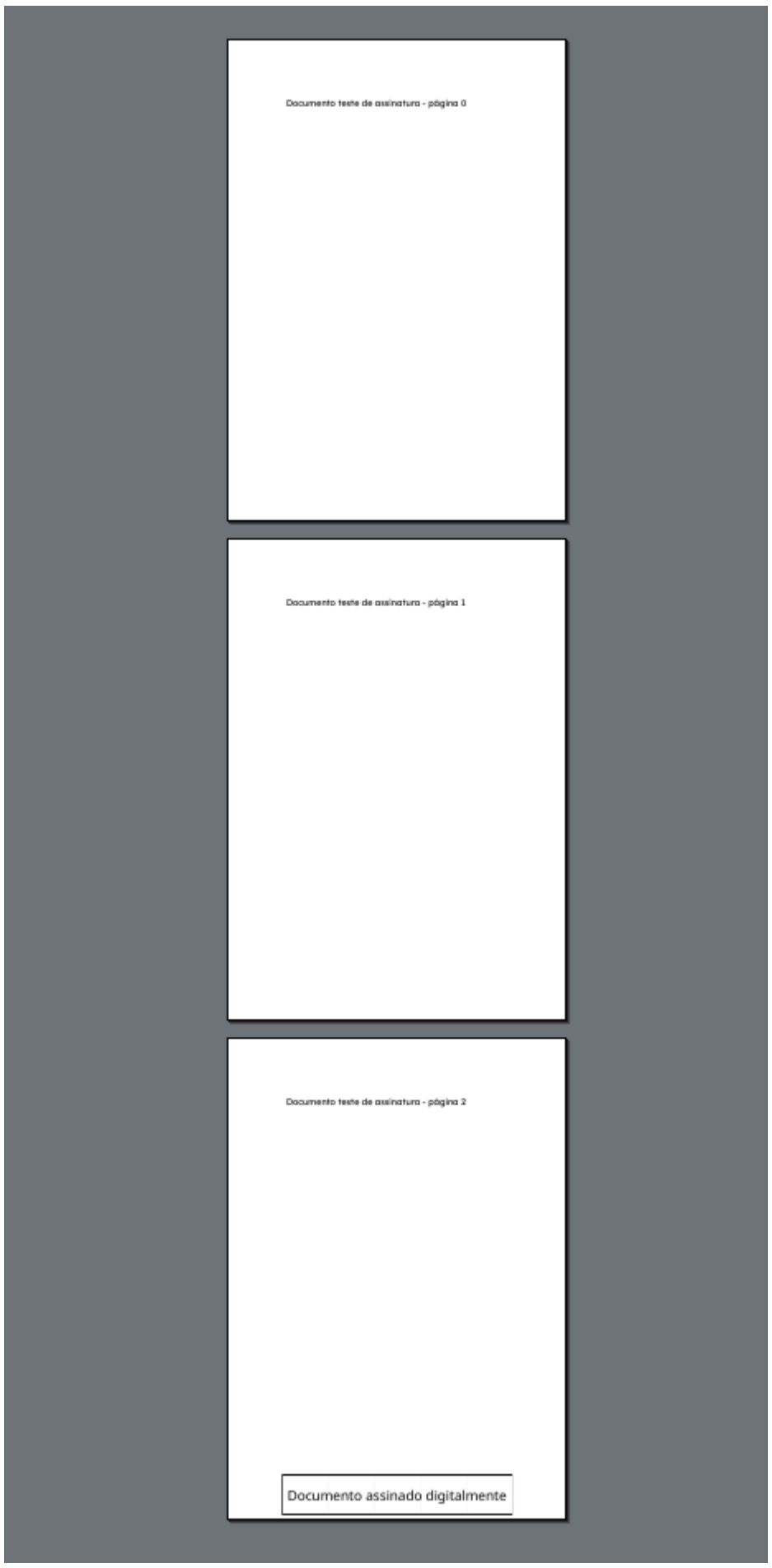
Exemplo com stamp_page = 1

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_page=1" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```



Exemplo com stamp_page = -1

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_page=-1" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```

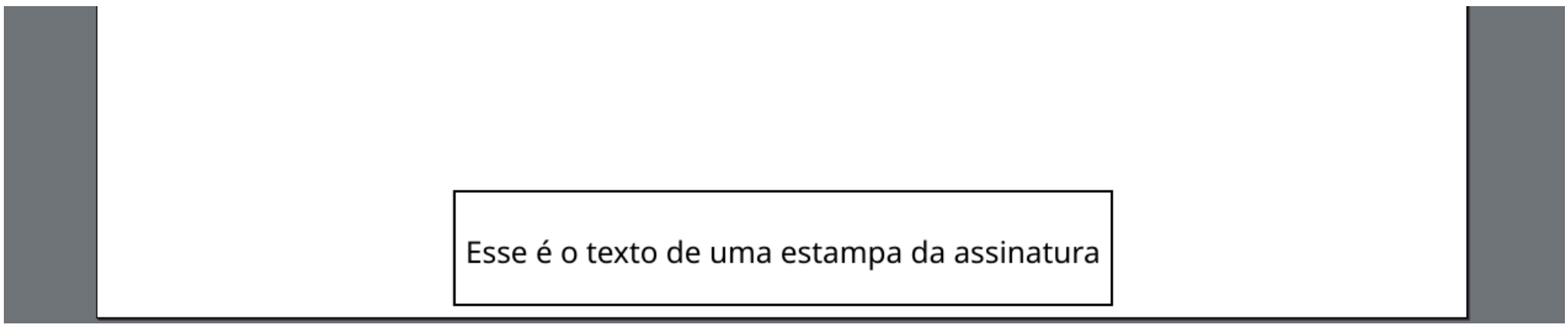


3.2.1.22. stamp_message

Atributo que configura a mensagem textual a ser impressa na estampa. Esse parâmetro foi utilizado em exemplos anteriores.

Exemplo com stamp_message = 'Esse é o texto de uma estampa da assinatura'

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_page=0" \
-F "stamp_message=Esse é o texto de uma estampa da assinatura" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```



3.2.1.23. stamp_url

Esse parâmetro especifica a URL que será utilizada na estampa, tanto para o QR code quanto para as demais. Em ambos os casos, se esse parâmetro for especificado, será inserida na estampa uma ação de clique e redirecionamento, que irá direcionar o usuário para a URL determinada.

Valores possíveis

- **''**: Nenhum evento de clique e redirecionamento será configurado para a assinatura.
- **texto**: Será adicionado um evento de clique e redirecionamento para a URL especificada à estampa.

Observação importante: Caso `stamp_QR_code=True`, o QR code gerado na estampa terá o valor da URL especificado. Caso o `stamp_message` contenha a *substring* `&(url)` e `stamp_url` for especificado, `&(url)` será substituído pelo valor de `stamp_url`.

Exemplo com stamp_url = https://www.google.com/

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_url=https://www.google.com/" \
-F "stamp_message=Verifique essa assinatura em &(url)" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```

3.2.1.24. pades_allow_further_sign

Atributo que configura se o documento pode ou não receber assinaturas posteriores. Isso altera a política de modificação do documento. Se duas pessoas devem assinar esse documento, por exemplo, a primeira assinatura deve ter o valor desse parâmetro como `True`.

Valores possíveis

- **True (padrão)**: Documento pode ser assinado após a assinatura corrente.
- **False**: Documento não pode ser mais assinado após a assinatura corrente, ela fechará o documento para modificações.

3.2.1.25. stamp_image (arquivo)

Arquivo a ser enviado via requisição, que pode ser utilizado tanto como estampa estática (sem texto ou QR code além da própria imagem) quanto como *background* (fundo do texto ou do QR code).

Caso o parâmetro `stamp_message` seja enviado com algum texto, o arquivo de imagem enviado em `stamp_image` será usado como *background* e o texto de `stamp_message` será inserido normalmente na estampa, acima da imagem de `stamp_image`.

Porém, também é possível utilizar a imagem como uma estampa estática. Ou seja, sem informação adicional em `stamp_message` adicionada na estampa. Nesse caso, apenas a imagem será inserida como artefato visual e, portanto, deve conter todas as informações necessárias para a estampa.

Observações importantes: No caso de estampa estática, é obrigatório que o parâmetro `stamp_message` seja passado com valor vazio `''`, indicando que a imagem de `stamp_image` deve ser usada como estampa estática.

Tanto no caso de imagens para *background* quanto no caso de estampas estáticas, são aceitos os tipos PDF, PNG e JPG.

Seguem exemplos com as possibilidades apresentadas.

Exemplo com stamp_image PNG, stamp_message e QR_code_enable=True (imagem será utilizada como fundo da estampa)

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_image=@INTIC-logo.jpg" \
-F "stamp_enable=True" \
-F "stamp_QR_code_enable=True" \
-F "stamp_message=Verifique essa assinatura" \
```

```
${document_signer_url}/sign \
--output teste_assinado.pdf"
```

Nota-se que a imagem foi utilizada como fundo da estampa, pois o parâmetro `stamp_message` continha um texto. Veja agora um exemplo com um documento cujas páginas são coloridas.

Exemplo com stamp_image PNG, stamp_message e QR_code_enable=True

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_image=@INTIC-logo.png" \
-F "stamp_enable=True" \
-F "stamp_QR_code_enable=True" \
-F "stamp_message=Verifique essa assinatura" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```

Observa-se que a imagem, sendo um background, obedece às regras de `stamp_inner_scaling` e, portanto, não ocupa todo o espaço da box da estampa. Nesse caso, é necessário ajustar o tamanho da box. O próximo exemplo trata esse problema.

Exemplo com stamp_image PNG, stamp_message e ajuste do tamanho da caixa principal

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_image=@INTIC-logo.png" \
-F "stamp_enable=True" \
-F "stamp_QR_code_enable=True" \
-F "stamp_message=Verifique essa assinatura" \
-F "stamp_box_size=100, 55" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```

Abaixo, um exemplo em que se envia um arquivo PDF como `stamp_image`, informando que `stamp_message` deve ser nulo.

Exemplo com stamp_image PDF e stamp_message=''

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_image=@iekuatiara-stamp.pdf" \
-F "stamp_enable=True" \
-F "stamp_QR_code_enable=True" \
-F "stamp_message=" \
-F "stamp_box_size=256, 50" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```

PDF é um formato vetorizado de imagem, portanto, não perde resolução quando a estampa é aproximada no visualizador do documento assinado. Veja a diferença de definição entre um JPG, um PNG e um PDF enviados como `stamp_image` e configurados como estampa estática.

JPG como estampa estática

PNG como estampa estática

PDF como estampa estática

Caso se deseje uma estampa rotacionada, basta enviar o arquivo `stamp_image` já rotacionado, apenas ajustando o tamanho da box externa com `stamp_box_size` e o seu posicionamento com `stamp_align_by`, `stamp_align_x` ou `stamp_align_y`.

Exemplo de stamp_image com PDF rotacionado

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_align_x=3" \
-F "stamp_align_y=100" \
-F "stamp_box_size=30, 151" \
-F "stamp_image=@image_pdf.pdf" \
-F "stamp_QR_code_enable=True" \
-F "stamp_message=" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```

3.2.1.26. stamp_vertical_text_enable

Esse parâmetro habilita a renderização vertical do texto da estampa. Por padrão, o texto é renderizado na direção horizontal, da esquerda para a direita. Caso esta opção seja habilitada, o texto será renderizado na direção vertical, sendo escrito de cima para baixo.

Valores possíveis

- **False (padrão)**: Desabilita a renderização vertical do texto.
- **True**: Habilita a renderização do texto na vertical.

Exemplo com stamp_vertical_text=True

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F "stamp_QR_code_enable=True" \
-F "stamp_align_by=left" \
-F "stamp_align_x=3" \
-F "stamp_align_y=50" \
-F "stamp_box_size=20, 250" \
-F "stamp_enable=True" \
-F "stamp_message=Estampa com texto vertical" \
-F "stamp_vertical_text_enable=True" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```

Também é possível obter um texto interno escrito horizontalmente dentro da estampa, com a mesma sendo exibida rotacionada. Para isso, pode-se utilizar a opção de estampa estática, enviando uma `stamp_image` em PDF, como pode ser visto no último exemplo de **3.4.26.** (exemplo de estampa rotacionada).

3.2.1.27. extra_information

Esse parâmetro possibilita o armazenamento de informações adicionais dentro dos `logs` de assinatura. Para utilizá-lo, é necessário criar um `JSON` contendo as informações desejadas.

Valores possíveis:

- Pode-se utilizar um JSON contendo qualquer chave desejada.

Exemplo com extra_information={"first_name": "John", "last_name"}:

```
 curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "stamp_enable=True" \
-F 'extra_information={"first_name": "John", "last_name": "Smith"}' \
```

```
${document_signer_url}/sign \
--output teste_assinado.pdf"
```

3.2.1.28. send_email

Esse parâmetro controla o envio de e-mails ao usuário quando ele assina um documento.

Valores possíveis

- **False (padrão)**: Desabilita o envio de email.
- **True**: Habilita o envio de email.

Exemplo com send_email=True:

```
 curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "send_email=True" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```

3.2.1.29. attach_signed_document

Este parâmetro regula a inclusão do documento assinado como anexo nos emails enviados. Se **send_email** for definida como **False**, este parâmetro será ignorado.

Valores possíveis

- **False (padrão)**: Desabilita a anexação de documento.
- **True**: Habilita a anexação de documento.

Exemplo com attach_signed_document=True:

```
 curl -H "Authorization: Bearer ${access_token}" -X POST \
-F "file=@teste_arquivo.pdf" \
-F "send_email=True" \
-F "attach_signed_document=True" \
${document_signer_url}/sign \
--output teste_assinado.pdf"
```

3.2.2 Envio de email

O retorno do assinador inclui o cabeçalho "X-Email-Sent", o qual pode conter os valores "True" ou "False", indicando se um email foi enviado ou não.

No caso da utilização do cURL, o header poderá ser visto por meio da opção `-v`.

3.3. Assinatura com privacidade (rotas `/sign/hash`)

Opções disponíveis:

- https://assinadoravancado.gov.mz/api/signer/pdf/1/sign/hash/sha256/<hash_value>
- https://assinadoravancado.gov.mz/api/signer/pdf/1/sign/hash/sha512/<hash_value>

Em ambas, `<hash_value>` corresponde ao *hash* do documento, que será assinado.

Uma das fases em um processo de assinatura digital avançada de documentos é executar um algoritmo de assinatura (ex.: RSA) a partir do *hash* do documento a ser assinado. Esta fase é executada por essas rotas.

Esta opção é especialmente útil quando a aplicação cliente deseja realizar uma assinatura com privacidade. Ou seja, a `API` não terá acesso ao documento original, apenas ao seu resumo criptográfico.

Um parâmetro opcional da rota `/sign/hash` é o `sign_time`, que deve ser no formato Unix Timestamp. O objetivo é permitir que o assinante comunique a `API` de assinatura, que irá informar a Autoridade Certificadora (AC) o horário exato em que o usuário confirmou sua intenção de realizar a assinatura digital. Este é o horário que o cliente desta `API` inseriu na entrada M do dicionário de assinatura, conforme especificado na `ISO 32000-2` na Tabela 255 Entries in a signature dictionary. Conforme informado na Declaração de Práticas de Certificação (DPC) da AC CertAU, aceita-se um `sign_time` de até 60 segundos anteriores ao horário atual da AC. Caso o `sign_time` seja anterior a 60 segundos, um erro será retornado. Caso o `sign_time` seja informado e respeitando o intervalo máximo de 60 segundos, a AC setará o horário de início de validade do certificado de assinatura única (o campo `not_before`) para 1 segundo antes do `sign_time`. Por último, caso o `sign_time` não seja informado ou esteja no futuro, o campo `not_before` será o horário atual da AC.

Segue um exemplo de requisição para uma das rotas de assinatura de *hash*:

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
https://assinadoravancado.gov.mz/api/signer/pdf/1/sign/hash/sha256/ \
ca978112ca1bbdcafac231b39a23dc4da786eff8147c4e72b9807785afee48bb
```

Caso se queira passar o `sign_time`, a requisição deve ser feita assim:

```
curl -H "Authorization: Bearer ${access_token}" -X POST \
-F sign_time="${sign_time}" \
https://assinadoravancado.gov.mz/api/signer/pdf/1/sign/hash/sha256/ \
ca978112ca1bbdcafac231b39a23dc4da786eff8147c4e72b9807785afee48bb
```

Vale destacar novamente que o valor do `sign_time` deve ser Unix Timestamp.

Retorno: Nesse caso, o retorno será o *hash* assinado, em formato PKCS#7. Segue um exemplo de retorno:

```
-----BEGIN PKCS7-----
MIIabQYJKoZIhvcNAQcCoIIaXjCCGloCAQExDzANBglghkgBZQMEAgEFADALBgkq
hkiG9w0BBwGgghdQMIIEnzCCA4egAwIBAgIUKbh984hEyH58hEmLRT9nzi/DfScw
DQYJKoZIhvcNAQELBQAwYjELMAkGA1UEBhMCQlIxCzAJBgNVBAgTAlNDMQ8wDQYD
VQQKEwZMYWJTRUMxDzANBgNVBAsTBkxhYlNFQzEkMCIGA1UEAxMbQUMgQ2VydEFV
IExhYlNFQyB2MSBIT01PTE9HMCAXDTI0MDkwMzEyMDAxMloYDzIxMjMwNjE2MTQy
OTE5WjAcMRowGAYDVQQDDBFFZHVhcmRvIFBlcm90dG9uaTCCASIwDQYJKoZIhvcN
... (omitido para brevidade) ...
RUMxJDAiBgNVBAMTG0FDIENlcnRBVSBMYWJTRUMgdjEgSE9NT0xPRwIUKbh984hE
yH58hEmLRT9nzi/DfScwDQYJKoZIhvcNAQELBQAEggEAZ85heQwlfxRAV0D2yXls
hOCDX2gYYeyLa5CJsBDo9CbDSL1xEmc1PJ5Wl4j8kkBy9zRu/6eXaVCYooS1kDHH
GeQ8i3175ag9/IoIrSHxZ4T/vqzsVEK/eWj98JRwQQLLqkuzDEh+IsLofIeeaEG6
i8CHJ0VXC7zFpQxJqZytFyBl7RasoGG65jReYLKhh9czvCs7BE+z0UlRQPQ7bgpf
xmPeAuGQHedlUOif7JhjUaVfjTFnimiHh0IF0yJbyvebFsWIjuIthxWKbLofvNjL
v+mzfvjYJSfl+lN2bf5FEfIgt5gNSePYtK2rCpdo8jZcRoFbzLw1tcyCOjXWit7M
JQ==
-----END PKCS7-----
```

Essa estrutura contém tanto a assinatura do *hash*, quanto a cadeia de certificados até o CertAU que assinou os dados. Para uma melhor visualização, podem-se usar ferramentas que decodificam esse `PKCS#7`. Clique aqui para ver essa estrutura decodificada na ferramenta LAPO dot it.

3.4. Erro na assinatura

Podem acontecer alguns erros específicos para assinatura de documentos na rota `/sign`:

- **PDF criptografado**: O PDF enviado para a assinatura é criptografado.
  - **Código de status HTTP retornado**: 534
  - **Mensagem de erro:**

```
"PDF criptografado"
```

- **PDF mal estruturado**: O PDF enviado para a assinatura não possui uma estrutura correta, o que impossibilita a inserção de assinatura digital avançada.
  - **Código de status HTTP retornado**: 533
  - **Mensagem de erro:**

```
"PDF mal estruturado"
```

- **Problemas com a estampa de assinatura**: O arquivo enviado para estampa possui algum problema, o que impossibilita a assinatura.
  - **Código de status HTTP retornado**: 422
  - **Mensagem de erro:**

```
"Erro com a estampa da assinatura"
```

Já nas rotas de assinaturas de *hashes*, pode ocorrer:

- **Erro com o hash enviado para assinatura**: O *hash* enviado não contém o formato esperado, sendo inválido para assinatura.

    - **Código de status HTTP retornado**: 400
    - **Mensagem de erro:**

```
"Digest {digest_algorithm} inválido, deve ser no formato {example_digest}""
```

- **Formato inválido do sign_time**: O formato do sign_time enviado não é Unix Timestamp (valores inteiros).
    - **Código de status HTTP retornado**: 400
    - **Mensagem de erro:**

```
"Parâmetro sign_time deve ser no formato Unix timestamp."
```

- **sign_time muito no passado**: sign_time está muito no passado, ou seja, o valor é anterior ou igual a now() - 60 seg.
    - **Código de status HTTP retornado**: 400
    - **Mensagem de erro:**

```
"sign_time é mais antigo do que o esperado: {sign_time} < {lower_limit} (mínimo esperado)"
```

Ademais, um erro generalista pode ocorrer:

- **Erro geral:** Em caso de erro geral durante o processo de assinatura.
    - **Código de status HTTP retornado**: 500

3.5. Sobre os leitores de PDF

Alguns leitores de documentos não implementam as especificações da ISO 32000-1 (ISO com as normas que regem a estrutura e funcionalidades do documento PDF). Por isso, ao lerem um PDF e renderizarem o mesmo, não entendem assinaturas e artefatos visuais.

Alguns visualizadores (visualizador de *responses* do Postman) ignoram a estampa visual. Outros mostram a estampa, porém não permitem outras funcionalidades. Exemplos disso são os navegadores (Chrome, Opera, Firefox, etc.), nos quais estampas habilitadas com evento de clique e redirecionamento por URL, por exemplo, não funcionam.

Porém, em leitores que implementam as especificações, tais como o Adobe Reader e o Okular, a estampa é reconhecida, assim como demais funcionalidades relacionadas a ela.

3.6. Log de documentos assinados

3.6.1 Chamada de API

O histórico de todos os documentos assinados por um usuário pode ser obtido através de um token de acesso representando tal usuário. Um exemplo de chamada usando `curl`

```
curl -H "Authorization: Bearer ..."
-X POST
https://assinadoravancado.gov.mz/api/logs/search
```

Caso haja a necessidade de filtrar quais logs serão retornados, tal chamada pode ser utilizada

```
curl -H "Authorization: Bearer ..."
-H "Content-Type: application/json"
-d "FILTERS"
-X POST
 https://assinadoravancado.gov.mz/api/logs/search
```

Acima, `FILTERS` corresponde a um JSON contendo as opções que serão utilizadas para filtrar os logs. Tais opções estão detalhadas em **3.6.3 opções da API**

3.6.2 Campos dos logs

Campo	Tipo	Descrição
id	int	identificador único de cada log
hmac	string	hash dos conteúdos do log. Está em codificado em hexadecimal
user_id	string	identificador único do usuário ao qual o log pertence

Campo	Tipo	Descrição
x509_certificate	string	certificado x509 utilizado para durante a assinatura. Está no formato PEM
file_name	string	nome do arquivo assinado
file_size_bytes	int	tamanho em bytes do arquivo assinado
time_to_sign_ms	int	duração do processo de assinatura
number_of_signatures	int	quantidade de assinaturas presentes no documento assinado
metadata	JSON	Metadados relacionados ao documento assinado. Detalhados em **3.4.2.1 Metadados dos documentos**
operation	string	String definindo a operação relacionada ao log. Valores possíveis são: `PDF_SIGNATURE`.
creation_date	UNIX Timestamp	data de criação do log imutável.
content_snippet	string	trecho do conteúdo presente no documento
access_token	string	token de acesso do usuário ao qual o log está relacionado
extra_information	string (JSON)	campo arbitrário contendo de informações armazenadas nos logs.
identity_provider_iss	string	URL do provedor de identidade utilizado para autenticar o usuário. Durante a assinatura, esse valor é obtido por meio do atributo "iss" do token de acesso.

3.6.2.1 Metadados dos documentos

O atributo `metadata` mencionado na secção anterior contém os seguintes parâmetros para documentos PDF:

Campo	Tipo	Descrição
author	string	autor do documento
creation_date	UNIX Timestamp	Data de criação do documento
modification_date	UNIX Timestamp	data da última modificação feita no arquivo. Como a assinatura modifica o arquivo, essa data corresponde a quando o log foi criado
source_app	string	aplicação na qual o documento foi criado
title	string	título do documento
subject	string	assunto do documento
pages	int	número de páginas do documento

Como tais atributos não são obrigatórios, podem existir PDFs nos quais estes atributos são nulos.

3.6.3 opções da API

Nome	tipo	valor padrão	descrição
sort_by	string	id	nome do campo que será utilizado para ordenar os logs
sort_order	string	desc	Indica se os logs deverão ser ordenados em ordem crescente (asc) ou decrescente (desc)
page	int	1	número da página de resultados retornados.
page_size	int	30	quantidade máxima de logs retornados
selected_columns	list[str]	Todos os campos definidos em **3.6.2 campos dos logs**, exceto `access_token`	Campos que serão retornados pela API.

Nome	tipo	valor padrão	descrição
filters	list[tuple[str, str, str]]	None	Lista de filtros que serão aplicados a todos os logs. Apenas logs que obedecem tais filtros serão retornados.

3.6.3.1 page e page_size

Para lidar com grandes quantidades de dados, a API segmenta os logs em páginas e entrega somente a página solicitada na requisição. Nesse sentido, o parâmetro `page` determina qual página será fornecida, ao passo que o parâmetro `page_size` estabelece a quantidade de logs a serem incluídos na resposta.

Exemplo:

```
curl -H "Authorization: Bearer ..."
-H "Content-Type: application/json"
-d '{"page": 1, "page_size": 100}'
-X POST
 https://assinadoravancado.gov.mz/api/logs/search
```

Para retornar todos os logs, deve-se fazer múltiplas requisições a api (incrementando `page`), até que nenhum log seja retornado.

3.6.3.2 selected_columns

O parâmetro `selected_columns` indica que campos de informação estarão dentro dos logs retornadas.

Com exceção do `access_token`, todos os campos definidos em **3.6.2 campos dos logs** podem ser retornados.

Para especificar os campos retornados, deve-se utilizar uma lista de strings indicando os campos desejados.

Por exemplo, `["id", "creation_date"]` resultaria em logs no formato:

```
{
  "logs": [ // Logs do usuário
    {
      "creation_date": 1732669826,
      "id": 4
    },
    {
      "creation_date": 1732669825,
      "id": 3
    },
    {
      "creation_date": 1732669824,
      "id": 2
    },
    {
      "creation_date": 1732669823,
      "id": 1
    }
  ],
  "total_count_in_db": 4 // Número total de logs de assinaturas feitas pelo usuário.
}
```

Um exemplo utilizando `curl`:

```
curl -H "Authorization: Bearer ..."
-H "Content-Type: application/json"
-d '{"selected_columns": ["id", "creation_date"]}'
-X POST
 https://assinadoravancado.gov.mz/api/logs/search
```

3.6.3.3 filters

O parâmetro `filters` é utilizado para definir uma lista de filtros que todos os logs retornados devem obedecer.

3.6.3.4 Definição de um filtro

Um filtro consiste em três elementos denominados: campo, operador e valor. Esses elementos são empregados para definir uma comparação entre o campo e o valor através de um operador específico.

Tais elementos são organizados em listas no seguinte formato:

```
[ "campo", "operador", "valor" ]
```

Por exemplo, essa lista

```
[ "id", "<", "10"]
```

equivale a essa expressão booleana

```
id < 10
```

3.6.3.5 Composição de um filtro

nome	tipo	valores possíveis
campo	string	Qualquer um dos campos definidos em **3.6.2 campos dos logs**, exceto `access_token`, `user_id` e `extra_information`.
operador	string	`<, <=, =, !=, >, >=, LIKE, ILIKE`
valor	string | number | boolean | null	Pode ser uma string, um número, um valor booleano ou null. OBS: caso o valor corresponda a uma data, deve-se utilizar o formato UNIX Timestamp.

3.6.3.6 Interação entre múltiplos filtros

Quando múltiplos filtros são definidos, a api somente retornará logs que obedecem a todos os filtros.

Por exemplo, esses filtros

```
["id", "<", "10"]
["creation_date", "<", 1732669823]
```

seriam equivalentes a essa expressão booleana

```
id < 10 AND creation_date < 1732669823
```

3.6.3.7 Usando filtros em uma requisição

Para filtrar os logs retornados, deve-se colocar todos os filtros dentro de uma lista identificada pelo nome `filters`.

```
curl -H "Authorization: Bearer ..."
-H "Content-Type: application/json"
-d '{"filters": [["id", "<", 100], ["file_name", "LIKE", "%plano%"]]}'
-X POST
 https://assinadoravancado.gov.mz/api/logs/search
```

3.6.4 retorno da API para logs

O retorno sempre terá código 200 e será um `json` em caso de sucesso, ou código 500 em caso de erro.

Um exemplo de retorno:

```
{
  "logs": [
    {
      "content_snippet": null,
```

```
  "creation_date": 1732669826,
  "database_version": "4",
  "extra_information": null,
  "file_name": "documento.pdf",
  "file_size_bytes": 33276,
  "hmac": "a3f5cecdbb16a9acc96eff847c99624eed7d97f422602f597ef5fca46c235f71474aa6a6a4bb0544ff3bbac44028106f80eead6f
  "id": 4,
  "identity_provider_iss": "http://keycloak:8080",
  "metadata": {
    "author": "Mauro Mangas",
    "creation_date": 1232549888,
    "modification_date": 1732669825,
    "pages": 1,
    "source_app": "Acrobat Distiller 8.1.0 (Macintosh); pyHanko 0.25.1",
    "subject": null,
    "title": "pdf_janela"
  },
  "number_of_signatures": 1,
  "operation": "PDF_SIGNATURE",
  "time_to_sign_ms": 202,
  "user_id": "00489369014",
  "x509_certificate": "-----BEGIN CERTIFICATE-----MIIF4DCCBMigAwIBAgIUb9+sjcIb4dTAPFErM5jMAQKcDZ4wDQYJKoZIhvcNAQELB(
    }
  ],
  "total_count_in_db": 4
}
```


Comments