MANUAL E LAYOUT API DE RECEBIMENTOS PIX

SUMÁRIO
MANUAL E LAYOUT API DE RECEBIMENTOS

PIX

Introdução

Com o **Pix,** é possível **receber pagamentos e transferências, 24 horas por dia**, **365 dias no ano, inclusive aos finais de semana e feriados**. O crédito **cai na hora**, inclusive entre **diferentes instituições.**

Integração

O **objetivo deste manual é orientar** o desenvolvedor sobre como realizar integração da **API de Recebimentos PIX.**

Objetivo

Este manual apresentará o modelo de acesso às **APIs - Interfaces de Programação de Aplicativos da Organização Bradesco**, com foco no processo de autenticação e autorização de aplicações do servidor. Será demonstrado o **passo a passo** para **automatizar o uso das APIs.**

Nesse modelo, a autorização de acesso considerará os recursos acessados pertencentes à **aplicação servidora e o token** de acesso será emitido para a própria aplicação, e **não para um usuário final.**

O padrão de autorização adotado será o **JSON Web Token (JWT) Profire for Oauth 2.0 Client Authentication and Authorization Grants.**

Resumo

Para atender ao objetivo acima, o manual **apresentará os procedimentos de como:**

1. **Gerar/Obter** o ID da aplicação;
2. **Gerar** o par de chaves - privada e pública;
3. **Gerar o JWT** assertion para requisição do Acess-Token;
4. **Obter o Access Token JWT** (auth/server/v1/token);
5. **Gerar a assinatura** do request de serviço;
6. Chamar o **serviço de exemplo (/jwt-service).**

PASSO A PASSO

1. GERAR ID DA APLICAÇÃO

Para geração do ID da aplicação, é necessário enviar os dados da empresa (nome, CNPJ, dados para contato) e informações sobre seu Aplicativo/Website (finalidade e Endpoints que serão utilizados) juntamente com a chave pública gerada, conforme detalhado nos itens abaixo, para o seguinte e-mail: plataforma.api@bradesco.com.br.

2. GERAR PAR DE CHAVES

Para gerar a **chave privada e a chave pública autoassinada**, use o **OpenSSL. Recomenda-se usar a versão mais atual.**

Observação:
O OpenSSL pode ser executado em um ambiente **Linux** (exemplo: MobaXterm, Putty) ou por meio do **git-bash.exe** instalado com o **Git client.**

O **par de chaves deverá ser gerado com a validade 1.125 dias** (aproximadamente, 3 anos), com o tamanho mínimo de 2.048 bits, usando um dos algoritmos suportados:

- ✓ **RS256** - RSASSA-PKCS1-v1_5 usando SHA-256;
- ✓ **RS384** - RSASSA-PKCS1-v1_5 usando SHA-384;
- ✓ **RS512** - RSASSA-PKCS1-v1_5 usando SHA-512.

Para gerar, como exemplo, execute o seguinte comando:

```
openssl req -new -x509 -sha256 -newkey rsa:2048 -nodes -keyout <nome do arquivo da chave privada> -days <vigência da chave> -out <nome do arquivo do certificado> -subj <subject DN do certificado>
```

EXEMPLO: openssl req -new -x509 -sha256 -newkey rsa:2048 -nodes –keyout parceiro. teste.com.key.pem -days 365 -out parceiro.teste.com.cert.pem -subj "/C=BR/ST=SP/O=Parceiro Teste"

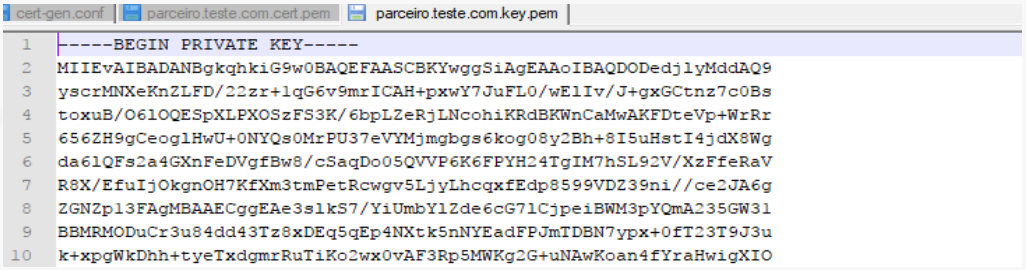
OBSERVAÇÃO: trocar "parceiro" pelo nome da organização e gerar a chave com um nome associado à sua organização.

Cada execução do comando, embora tenha os mesmos parâmetros, gera uma nova chave privada em um novo certificado. A chave pública fica incorporada no certificado gerado.

Esse comando gerará dois arquivos: a chave privada e a chave pública, em formato PEM (Base 64).

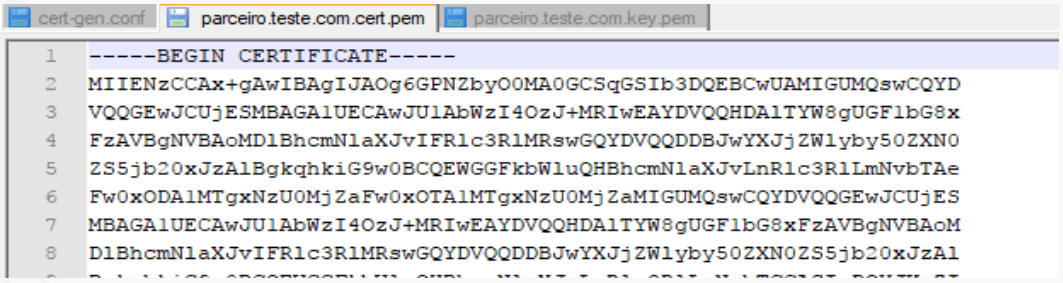
Para conferir a chave privada, **abra o arquivo em um editor de texto.**
A primeira linha do arquivo deve ter o texto **"BEGIN PRIVATE KEY".**

EXEMPLO: parceiro.teste.com.key.pem.



Na chave pública, o texto deve começar com "BEGIN CERTIFICATE".

EXEMPLO: parceiro.teste.com.key.pem.



OBSERVAÇÃO

Esse arquivo é um **certificado que tem a chave pública** e alguns metadados adicionais, como o nome de quem gerou o par de chaves, entre outros.

A **chave privada é de responsabilidade de sua empresa** e deve ser armazenada de forma segura e nunca deverá ser fornecida a terceiros.

A **chave pública, e somente a chave pública, deve ser enviada em dois e-mails separados** da seguinte maneira:

1. Chave pública zipada com senha;
2. Senha de descompactação em um txt anexo.

À Organização Bradesco pelo e-mail: plataforma.api@bradesco.com.br

OBSERVAÇÃO: para o consumo das APIs, utiliza-se duas requisições:

- ✓ **Obtenção do access-token;**
- ✓ **Consulta ao endpoint do serviço.**

Os passos seguintes **descreverão como realizar essas duas requisições de forma manual, utilizando-se da ferramenta Postman.** Os fluxos devem ser desenvolvidos para funcionar de **forma automatizada nos sistemas consumidores.**

OBTER ACCESS-TOKEN

1.1 Gerar JWT

O JWT é utilizado na primeira requisição para obtenção de token de acesso.

O desenvolvedor deverá gerar um JWT de acesso e fazer a assinatura com um dos algoritmos indicados **(exemplo: RS256)** com sua chave privada. O JWT é formado por dois JSONs, sendo eles: **"header.json"** e **"payload.json".**

Header.json:

```
{
   "alg": "RS256",
   "typ": "JWT"
}
```

Payload.json:

```
{
   "aud" : "<endereço do serviço em que o token será gerado>",
   "sub" : "<id client do cliente >",
   "iat" : "<data de geração deste jwt, no formato NumericDate>",
   "exp" : "<data de expiração deste jwt, no formato NumericDate>",
   "jti" : "<nonce – numérico(18) random, a cada chamada deve ser trocado, ex: unix current date
in milliseconds >",
   "ver" : "1.1"
}
```

Exemplo de Payload preenchido:

```
{
  "aud" : "https://proxy.api.prebanco.com.br/auth/server/v1.1/token",
  "sub" : "bc7ccf09-8a85-4be6-y67e-82bf11737994", <id cliente fornecido pelo banco>
  "iat" : "1574094116",     <data atual em segundos>
  "exp" : "1576686116",     <data atual adicionando 1 mês à frente, em segundos>
  "jti" : "1574094116000",   < numérico(18) random, a cada chamada deve ser trocado, ex: unix
current date in milliseconds >
  "ver" : "1.1"
}
```

Com esses dois **JSONs formatados**, realize o **encode de ambos para Base 64 e os concatene**, separando-os com o caractere "." (ponto). **O formato será:**

Header(base64) + "." + Payload(base64).

Assine string gerada. Concatene a assinatura, separando-a com **o caractere "." (ponto), o formato final será:**

Header(base64) + "." + Payload(base64) + "." + Assinatura(base64).

Exemplo JWT:

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOiJodHRwczovL3Byb3h5LmFwaS5w
cmViYW5jby5jb20uYnIvYXV0aC9zZXJ2ZXIvdjEuMS90b2tlbiIsInN1YiI6ImJiN2NjZjA5LThh
ODUtNGJlNi1hNjdlLTgyYmYxMTczNzk5NCIsImlhdCI6IjE1ODg5NDU3MDIiLCJleHAiOiIxN
TkxNTM3NzAyIiwianRpIjoiMTU4ODk0NTcwMjIzMyIsInZlciI6IjEuMSJ9.RhJRerMlUsOqvnV
cJRfU-g4i9yZGd3nIfUWhd8hUBAkFyriLpOblvorXI3YRUe5JJUPGhj36vW7uKNPIL-
l9a8RQs-lnPJd20v-
dyYznvMYva7IpoC6tm3byO56qcCky4tocCnGFTA_dRXnTvJIEPPJSK5GjBpAYWAtcfF5hj
mzpOsjUhoizOWTTEnda7UAiSo5DaExpV2ngj_99sRCAf6JhbtHr6Wiw_B5aJdeuzhq9dbp3
To7nggUOeG3bcpX1SdtDECn60S6s7USB2JE3Cb08O_1_Ri3NBzqZ5nLxpKw35ONe8Q7
O2Wfu2zG8O04INAdjp0gBbhWCwe38ljqoKQ
```

1.2 Request para Obtenção do Access-Token

O **access-token** é de uso único e deve ser gerado a cada nova chamada das APIs. Para realização do teste, recomendamos usar o Postman.

Antes de criar a requisição, certifique-se que o **"SSL certificate verification"** no Postman está **DESATIVADO.**

File → Settings → General → SSL certificate verification

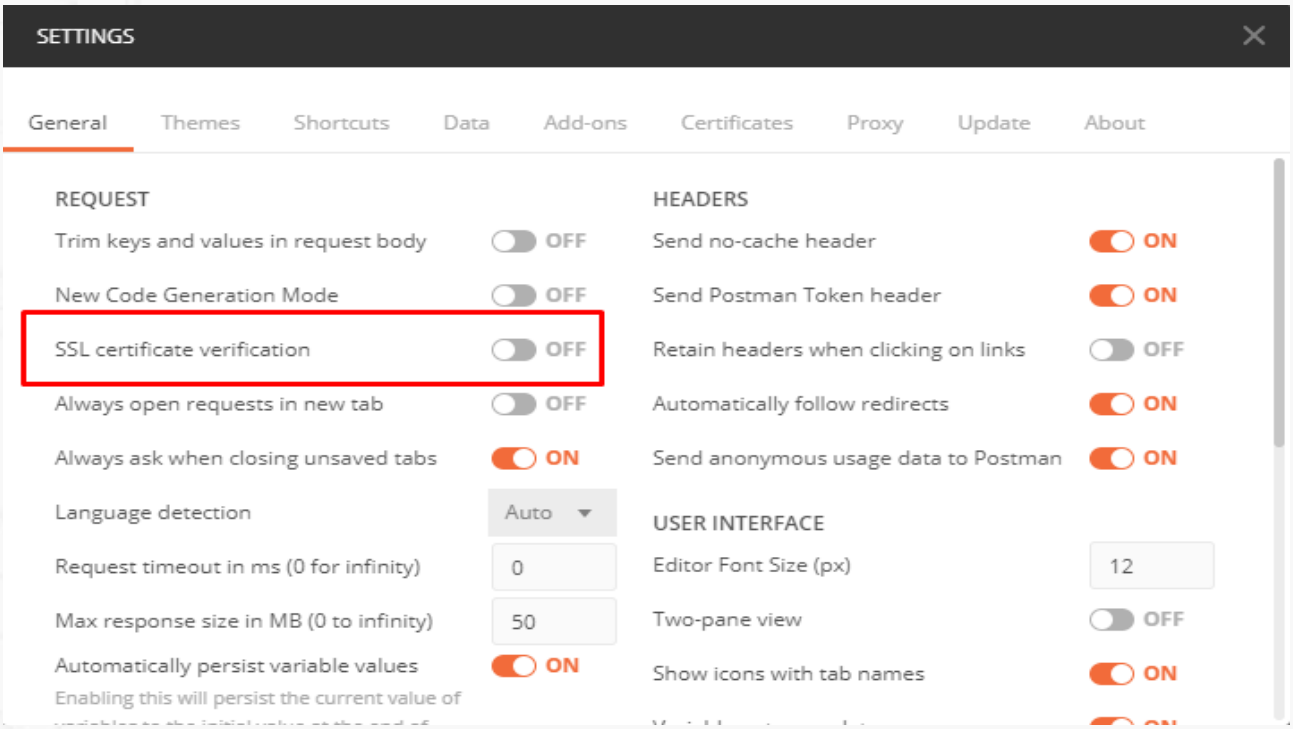


Figura 1. DESATIVANDO SSL CERTIFICATE VERIFICATION

A requisição do Postman para obtenção do token deverá conter as informações de "método" e "headers", conforme abaixo para a URL de serviço:

https://proxy.api.prebanco.com.br/auth/server/v1.1/token

Method POST
URL: https://proxy.api.prebanco.com.br/auth/server/v1.1/token
Body: selecionar "x-www-form-urlencoded"
Key: grant_type = urn:ietf:params:oauth:grant-type:jwt-bearer
Key: assertion = <JWT assinado (HASH gerado anteriormente)>

Exemplo JWT:

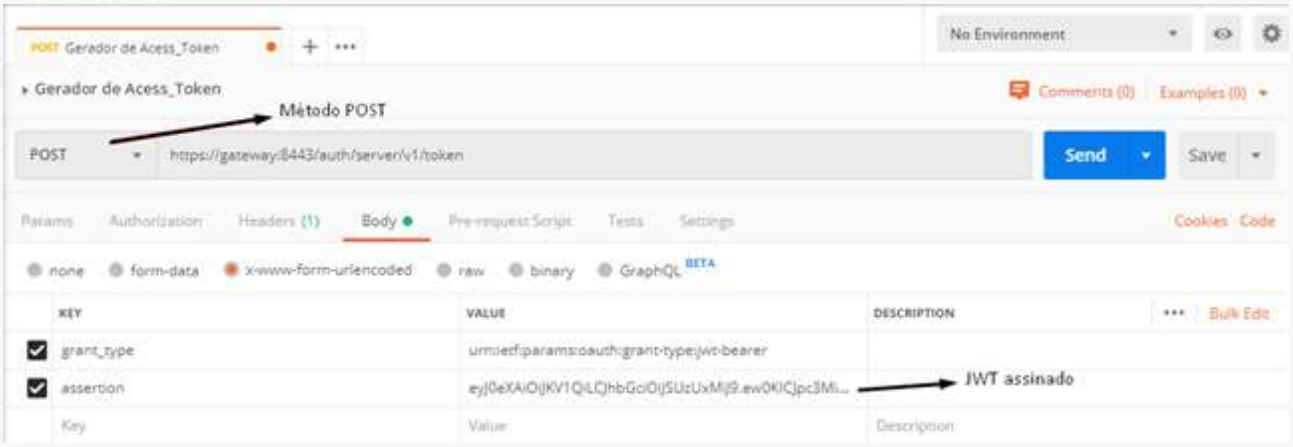


Figura 2. EXEMPLO DE REQUEST POSTMAN

O Postman adicionará o header Content-Type automaticamente:

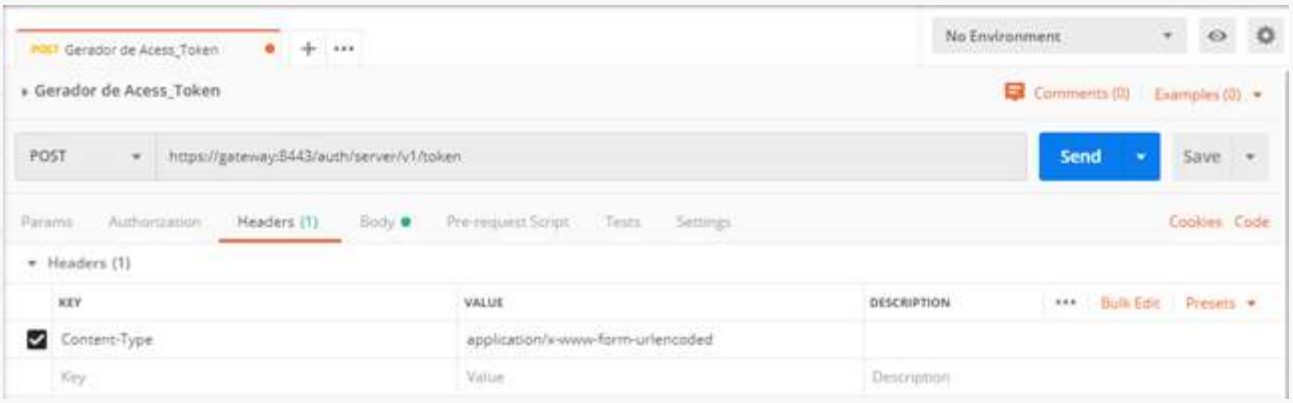


Figura 3. HEADER CONFIGURADO PELO POSTMAN

Após enviar a requisição ao serviço, o access-token gerado será retornado da seguinte forma:


```
{
 "access_token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzUxMiJ9.ew0KICJpc3MiOiAiaHR0cHM6Ly8yMDEuNy4xMjcuMTcxOjg0NDMvYXV0aC9zZXJ2ZXIvdjEvdG9rZW4iLA0KICJhdWQiOiAiaHR0cHM6Ly9sYWI5LnByZWJhbmNvLmNvbS5icjo4NDQzIiwNCiAiZXhwIjogMTU3NzI4MTIwOSwNCiAiaWF0IjogMTU3NDY4OTM1NSwNCiAic2NwIjogIm9vYiIsDQogInZlciI6ICIxLjAiLA0KICJhdXRoVHlwZSI6ICJzZXJ2ZXIiLA0KICJhdXRoRGF0YSI6ICJleUowZVhBaU9pSktWMVFpTENKamRIa2lPaUpLVjFRaUxDSmhiR2NpT2lKU1UwRXRUMEZGVUNJc0ltVnVZeUk2SWtFeU5UWkRRa010U0ZNMU1USWlmUS5BSjIxWWJkSDE2UDRPYlBJaGFmV0hqMldBX0tFQTVSZnRIVzlQQWFQVHFyM2tjamktMHJ5NFNSX2Jwb3p1RnJGR0MzNEY2bVdtdEl1VFJ3dG5pSEpVVGNhQk5jeUlVYzd2cEctZWZyWWsxVzNKaERzMTF4TUZ3OFN1Sk9CRk14M0tDZGM4STBKMGdON1FTVnlCSllKcEFITE83VTJsMHphQzc3YzlZY3dSLUFsZ1ZleUZzaHE4cHlqU0VaZXJtMDJLS1MwMTBJN09iVTE2WWgtNHRDQktSQ1VPT2Z4bnhnZFp2RXg2UUxNOHByeE81OTJSNmFCY0Fkb1lmM3JHRmdielQ0YkF0RjdkbXBmZ2pacnZfT2stNkU1dU9DVjVyZ1JSNEFrMkZ1Y0ZENHhKNWs2Mlp5Qm5FOS1sMU96QkJKWVMtS09fS3pfa0RQWGFaQ2hya3ctb3ZWS3lBOXdHS0dPN1hGb3l3eUl6VXk5dTZzSlpHNFVwWGxxZ3R5MWdKZFRfTzlKc0NISU1zZldHYVU1S3RnendaWkxER2N0Q0N3eFZsejJMM1Vxd1loTlgxWjhCR3Q1NVl0aUw1djRFNzVBUWZ6SzZ2a2U4WE5SZmdGcW5sSlJ4eW9nWE5EZ1JQNG41dGJJOVBLTmJhWll0SkVscXpGbWhBZElzcjVqOHNNZ1Z2cnNHVHhJelR0Y0ZMdUJCMFJMQkg3SFNQSXRiOW96eGRRaGZtN01lcGctQ0VldjhRTzdpaTUd19jQTF5a043cUdPaUxaNGR2OU4xT3NwR2piRmpXRmlEZUxVTXJHX0tvc0VIRkFYc3JMZzFOUjVGLURSUDIzeERLRkZJTzY1ektmTmZvTGt2Vld4bnFyckdhVGx5aXl2dVhhZlZZTHNSM3ViWXN1MjZNWS55cGkyWmdyblF6QUR1UEVyOGtVNWdBLnR2am5KMkVQOXVPMXRZcGtZbnlPY0JHTUJkdGh1Znd0NHJuT3h6b1BacUotbmRtSXE2LVFlMnpuYjVTUE1Va3pCOTN3MXdwU2hldTVXdlFqdUFwNUtyd2RyaEtmRUdmRjFmUjVaYXlHVlNrYnZLeXp2X2JaN2h5U21RZTVPT0FyaEhxbmF1MzRONFNoU0RuTmdKX1ZFbUdnVmNFMEJPQ1VpY1hHMjROMDZZLUw3UnZQR18xakY5S01xWE14OWd3YmREeEpFOEFzUzlDYy1aSFd2SHozemlXV2hKdDIxTVNPQWxZeVFoNWR0TXpDYmw1ZFRzeHNaeTEwQzhwaEs2UG1QemZZQndLS1ltS05WV1U1VXNyZEpQVHpPaHBISUNDcTd5UWhvSkFYY2dXNm1tYTdadGJkREllSEt3ay0tZTVFbnphT1NReTB0N3owUmQtSEdOWDZGYnNhRml0dFJUV1paQmFXNTI5X0hPcENNfRUkwNE5ta0daOC15MkdubzlBSThtdFRUUTZTVDg2R0JFYVdjWENIODhBenhOOTl6M052dTdLSG9wYndzR1luTEZnYX
```

```
oxdTROaWtFejdCdFhQVGR1bDZRZ3VJaDRkMzJWNGlWZjlEVEhKYmJDa1FweXAxVExLUkZJcm51VDRrZFZBWVdpQS1KNHk4enRsUU93V290T
FNUQkI2TkpJanYxeS1ZUTc3QlFkYTlJNjBtdWlucUx4ay1ja0l5QnNXRGl1QW9xckdlUy1tT0puR1VNYzVMbkYzN1k4RU9fVUh2WGc4MVlhdFZWZ
2ZFWHVUU2djNGFPV1JtYVFxV1lzV1V5YUtfbTNvcjBqa0pXaE9sbkFQV1JnSk13dVpsQlFPa2hfVmVScTFacFQyUFJWMW14emVpcEh0MnNodz
ZGd3l3MlotZ2NrTmk2a3NEOWhTNDJWaHd1c3RtTnlVUVo2OUtiR2xqLTZHZl94d1B0SklLaGVTNDZoQ0RsOFAyMjVqSHlPZjF2QXZsZFpPWXFJ
dE5sWDZfT1F0M20xMXRhSHpFdzluRm5ZaUxpcThSNmN0c0RWck5xYU8xSDZIX1J0Ymc1eEY2NmpQZk5PS2lfNDBXX1F4U2REQVhia003SDV
2cFpLZlhjak5sNnFWYXNPbE1wNC1pWUVESFBqdGRsSGJpSlR3ZUtISjI3c0V1WlQ2VXFTYjdCOGl3cmxPTmNCZU4wdFBXbzJycTR3amJDY1l2Z
ElydTZKZ1hVWlBpRlNGZEFDaGRPTUI4WDNfRVduWnprSWFfdVd0SGY2X2ZlaUJMcHMwOExFWkRJaWQ0LVRqRC1nMkNISHdSR2pXTC1pVDd
pdEV0U2ZuS0g2d3c3SU1WV2d1VEl3UlhIbnpVSENUWjc0WVNkY3dEODFEc0NLSktHQ0l4eVZIUlhRaGdPRHdyWjkwMUs2bHNxa2ItWTR3Mk
taY2hNaUFxUVZQSjRQQldKbkd2RkVXSU9TbXdFUmlFRWczMkkzT0tyT183dEFjY1NrUWZxSUN5bU41LWNaTTR1dlRBRjdhMkhFbjMxTEtEOHc
zYXFseWNBV2RWNExpT1ZYZXVGbmtJU1RrMFRaVDQuaTZ5Q0VoMlZSc1FIWUFzVTBkOUpITWdDa2FyUV9HWm9Fa3ZrNmkzT1ViNCINCn0.R
r6C_uCohQMlmT0fCv-HUmn-WLoRfgwP4euYbJz2plsm_CsSwwEgZn3XQWbWGC7QuAL6wdkMOkoSeOQbwdNUNukyjuDVXJEh_W6e-
wvBaE8TWrSS-
CijJ9X9Xdp93uObT99YaYSmGj22bLGtGdl86su7GYI5ORzoFKyYkuACsi1Q0AbqLbrKugYgNfRSFGzatkcjO9eeKqsJ4xUdPIEbgBbRLdEntkcs3MP
NU-XKt9eGv1opOg1JkKciibOfcP1HJHDiAcFHxLqmPGhrVJ8Wu99y3nWwzfyvj9MilqK7OjJ_63ROZQyTV-
Fb7jbh0sGA9i3TrcQGrnFJ_CCayfctC4ah4KrF02J-UcQ1xS52ezeNdv1jFlgsKpZT8dfN-vpgyxuJTR9mq9HDCGJQyjf-
ivoaF7CnTy8peSLY9Knzxl4vVFpdBAgeKh7qOn058-TZu2Y06d8mEfd3xet_m1wE4A85-
7anbw2uKI1PTAQiEmPuY46xAH1FSXq97fcUYaeE7Z695if_yipyZsWbntOnU-
9dzCm4tu76FnDc45Q6qd7UqDCXDXSFFnxhe9eqX5dCll0O3eDgU09_pCd5qAQyprLqTkWe5ib-8aG5jotBsPENQNnxie57ol8DR2y_k-
dCjwPQ1RRjYHM9Xd8gqeivP6WnAx4wR2kFfo8FNndfLpo",
  "token_type": "Bearer",
  "expires_in": 2591854.0,
  "scope": "oob"
}
```


OBSERVAÇÃO: esse **access-token** será utilizado para realizar as chamadas nos endpoints dos serviços. A geração de um novo access-token somente deve ser feita após o mesmo expirar, conforme data recebida no retorno no atributo **"expires_in".**

O efeito colateral de gerações contínuas é se deparar com o erro abaixo:

```
{
   "code": 0,
   "message": "unexpected error",
   "details": [
      {
         "name": "internalCode",
         "value": "FRWK0103"
      },
      {
         "name": "internalMessage",
         "value": "EXCEDIDO O LIMITE DE SESSÕES ABERTAS SIMULTÂNEAS PARA ESTE USUÁRIO. PARA ABRIR UMA NOVA
SESSÃO, FECHE A SESSÃO QUE ESTÁ ATIVA NO TERMINAL APIFRONT OU UMA SESSÃO ATIVA EM OUTRO TERMINAL."
      }
   ]
}
```

2. Chamada à API do Serviço

Neste manual, utilizaremos o **endpoint de exemplo (/jwt-service).** A resposta do recurso é **"API acessada com sucesso!"** e pode ser utilizada para o teste de autenticação na camada de segurança.

2.1 Gerar Arquivo request.txt

Será necessário que crie um arquivo com o nome request.txt.

Nesse arquivo, deverão ser incluídas algumas informações que serão utilizadas na chamada do postman, tais como: **método, URL, parâmetros, endpoint e, inclusive, o access token obtido anteriormente no serviço:**

https://<endereco_do_ambiente>/auth/server/v1.1/token.

As **informações no arquivo devem ser inseridas**, sendo que a cada informação nova assumirá uma nova linha. O arquivo **request.txt** deverá estar no mesmo diretório que a chave pública e a chave privada, **observando o seguinte padrão:**

```
VERBO[1] – linha 1
<URI da chamada>[2] – linha 2
<Parâmetros que estão sendo utilizados na URL>[3] – linha 3
<body>[4] – linha 4...
<Valor do access-token>[5]
<Nonce>[6]
<Timestamp>[7]
<Algoritmo que está sendo utilizado>[8]
```

É possível que os **parâmetros e token ocupem várias linhas do arquivo,** assim uma nova informação deverá ser colocada logo abaixo da outra.

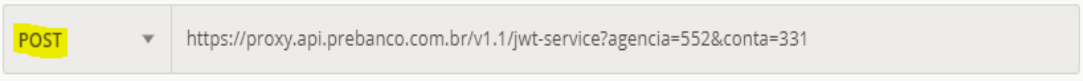
É necessário que a ordem de informações no arquivo **permaneça exatamente conforme o modelo acima.**

O arquivo **"request.txt",** depois de preenchido e devidamente assinado (gerar um hash de assinatura), **será utilizado na chamada do endpoint de exemplo** (/jwt-service).

Segue exemplo de **preenchimento do arquivo** que será assinado e de como ficará a chamada no **Postman.**

Linha 1: método utilizado no Postman.
POST[1] - linha 1 do arquivo.

Colocaremos esse mesmo verbo na chamada Postman:



Linha 2: URI da chamada (endpoint à API consultada).
/v1.1/jwt-service[2] - linha 2 do arquivo.

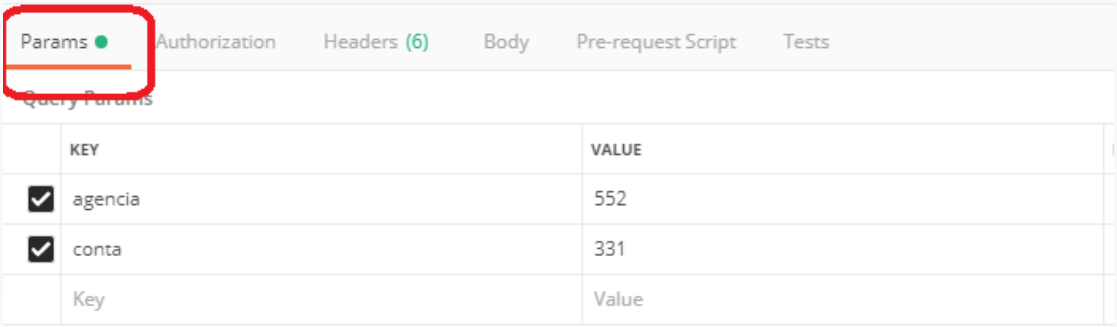
E no Postman:

Linha 3: exemplo de parâmetros da chamada.
agencia=552&conta=331[3] - linha 3 do arquivo.

Então, no Postman, temos os parâmetros que estão sendo utilizados, podemos inserir esses valores na URI após o "?" ou na aba "Params" do Postman.

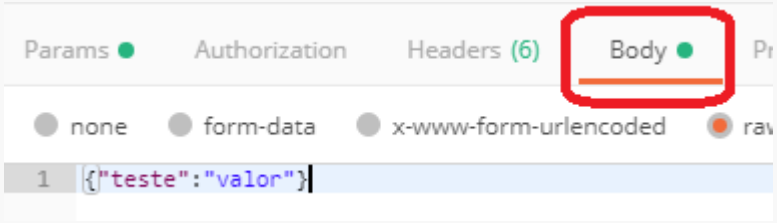
URI:

Aba Params:



Linha 4: body da chamada.
{"teste":"valor"}[4]

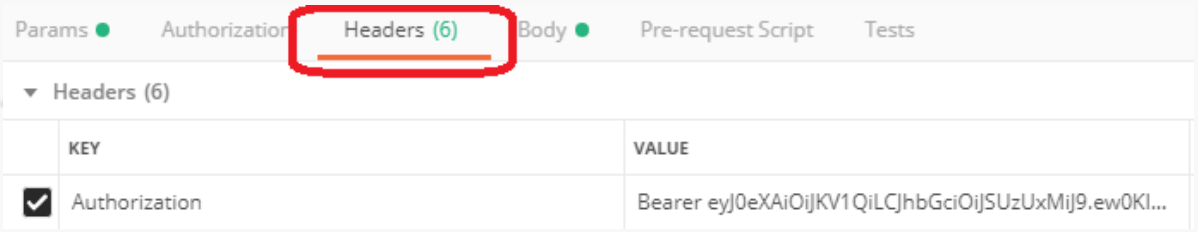
O body dessa chamada deve ser "raw" "JSON" no Postman.



No caso de um endpoint especifico não possuir "parâmetros" ou "body" na requisição, a linha do arquivo request.txt correspondente deverá ficar em branco.

Linha 5: access-token gerado nos passos anteriores.
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzUxMiJ9.ew0KICJ...[5]

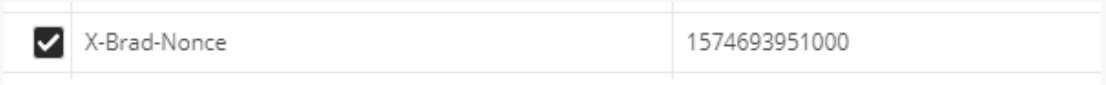
No **Postman,** devemos inserir esse valor no cabeçalho da chamada, que será referente à chave **"Authorization",** e o Campo Valor deve ser iniciado com **"Bearer[espaço]** valor do access-token da **sequinte forma:**



Linha 6: nonce (valor numérico (18) aleatório, que poderá ser utilizado uma única vez para cada chamada. Nesse caso, está sendo utilizada a data atual em milissegundos).

1574693951000[6]

Esse mesmo valor deve ser inserido no Postman, equivalente à chave "X-Brad-Nonce", no "header" da chamada, ficando conforme imagem abaixo:



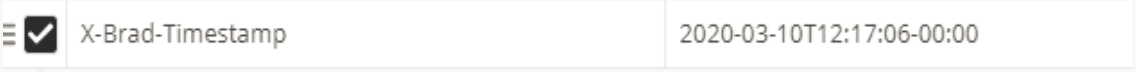
Linha 7: timestamp (referem-se à data e à hora que está sendo efetuada a chamada para o endpoint).

2019-11-25T11:23:00-00:00 [7]

Formato "AAAA-MM-DDThh:mm:ss-00:00", sendo:

- ✓ AAAA = ano com quatro caracteres, exemplo "2019", referindo-se ao ano atual;
- ✓ MM = mês com dois caracteres, exemplo "11", referindo-se ao mês de novembro;
- ✓ DD = dia com dois caracteres, exemplo "25", referindo-se ao dia 25;
- ✓ T = texto fixo;
- ✓ hh = hora com dois caracteres, exemplo "11", referindo-se às 11 da manhã;
- ✓ mm = minutos com dois caracteres, exemplo "23", referindo-se aos 23 minutos daquela hora;
- ✓ ss = segundos com dois caracteres, exemplo "00";
- ✓ -00:00 = texto fixo.

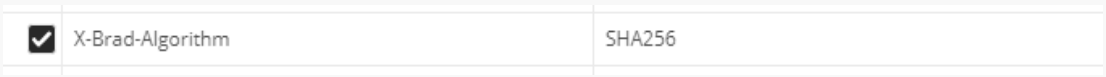
Esse mesmo valor deve inserido no cabeçalho da chamada, sendo a sua chave "X-Brad-Timestamp", ficando no padrão ilustrado abaixo:



Linha 8: algoritmo que está sendo utilizado.

SHA256[8]

No cabeçalho da chamada, o valor é correspondente à chave: "X-Brad-Algorithm", conforme imagem abaixo:



Então, o arquivo para assinatura ficará da seguinte forma:

```
POST
/v1.1/jwt-service
agencia=552&conta=331
{"teste":"valor"}
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzUxMiJ9.ew0KICJ...
1574693951000
2019-11-25T11:23:00-00:00
SHA256
```

Observação: no exemplo acima, estamos preenchendo tanto body quanto os parâmetros da requisição, porém, caso não seja utilizado body ou parâmetros, deve-se deixar a linha referente ao que não está sendo utilizado em branco.

Abaixo, ilustração de como deve ser feito um arquivo de assinatura, quando não está sendo passado o body na chamada:

```
POST
/v1.1/jwt-service
agencia=552&conta=331

eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzUxMiJ9.ew0KICJ...
1574693951000
2019-11-25T11:23:00-00:00
SHA256
```

Note que, nesse caso, deixamos a linha referente ao body em branco.

OBSERVAÇÃO: é importante considerar a quebra de linha do padrão **Unix (\n: LF – LineFeed)** no arquivo request.

O padrão de quebra de linha do Windows **(\r\n CR – CarriageReturn, LF- LineFeed)** ou de algum outro sistema **(\r CR – CarriageReturn)** causará uma assinatura inválida.

Ou seja, o arquivo deve ser salvo no padrão de quebra linha do **Linux**. Caso seja salvo em outro padrão de quebra linha, será retornado na chamada um erro de assinatura inválida.

2.2 Gerar Assinatura request.txt

Então, agora que o arquivo de assinatura está completo, será necessário assiná-lo. Como exemplo, utilizaremos o nome desse arquivo como "request.txt".
Para assinar esse arquivo, é necessário salvá-lo no mesmo diretório em que estão as chaves pública e privada, como mostra a imagem abaixo:

```
$ ls
request.txt   suporte.pem   suporte.teste.com.key.pem
```

Após isso, execute o comando em um ambiente Linux para gerar a assinatura. A assinatura deve estar no padrão SHA256, o comando utilizado será o seguinte:

```
echo -n "$(catArquivo_de_assinatura.txt)" | openssl dgst -sha256 -keyform pem -sign
nome_da_chave_privada.key.pem -out arquivo_de_saída.txt.256
```

Exemplo de execução do comando utilizando o arquivo "request.txt" e a chave privada "suporte.teste.com.key.pem":

```
echo -n "$(cat request.txt)" | openssl dgst -sha256 -keyform pem -sign
suporte.teste.com.key.pem -out request.txt.256
```

Após executado, **não será retornada nenhuma mensagem pelo terminal**, porém será criado um novo arquivo **com extensão ".256"** que, no nosso caso, será o arquivo **"request.txt.256",** conforme ilustra a imagem abaixo:

```
$ ls
request.txt   request.txt.256   suporte.pem   suporte.teste.com.key.pem
```

Após isso, será necessário codificar esse arquivo para Base 64, utilizando o comando:

```
base64  nome_do_arquivo
```

O exemplo abaixo ilustra como codificar o arquivo "request.txt.256":

```
base64 –wrap=0 request.txt.256
```

Após executar esse comando, será retornada a string da assinatura desse arquivo em Base 64, conforme ilustra a imagem abaixo:

```
$ base64 --wrap=0 request.txt
U1QKL2p3dC1zZXJ2aWNlCmFnZW5jaWE9NTUyJmNvbnRhPTMzMQp7InRlc3RlIjoidmFsb3IifQpleUow
ZVhBaU9pSktWMVFpTENKaGJHY2lPaUpTVXpVeE1pSjkuZXcwS0lDSi4uLgoxNTc0NjkzOTUxMDAwCjIw
MTkxMTI1VDExOjIzOjAwLTAwOjAwClNIQTIlNgoKIwo=
[i392610@AT-CL-MT-023 teste-suporte]$
```

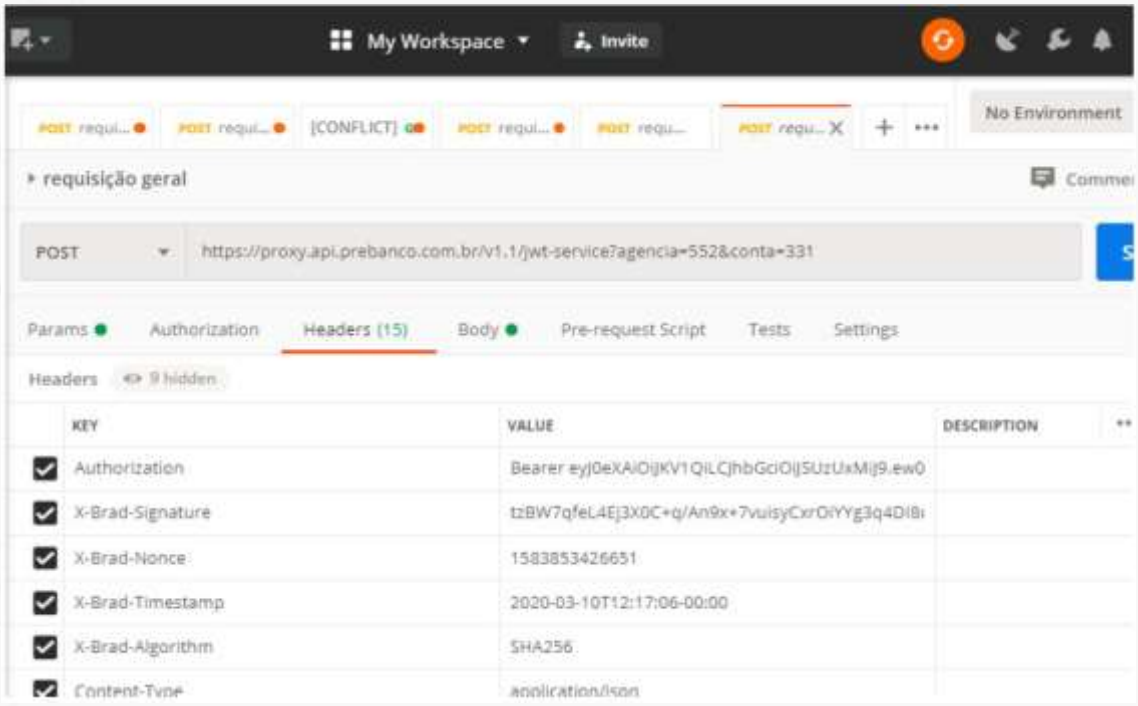
A string gerada será utilizada para fazer a chamada. Essa string é equivalente à chave "X-Brad-Signature" na chamada do Postman.

☑	X-Brad-Signature	U1QKL2p3dC1zZXJ2aWNlCmFnZW5jaWE9NTUyJmNvb...

Observação: antes de colar a string no campo do valor do X-Brad-Signature, é necessário deixá-lo em uma única linha; caso contrário, sua chamada retornará erro de assinatura inválida.

2.3 Realizar Chamada à API do Serviço

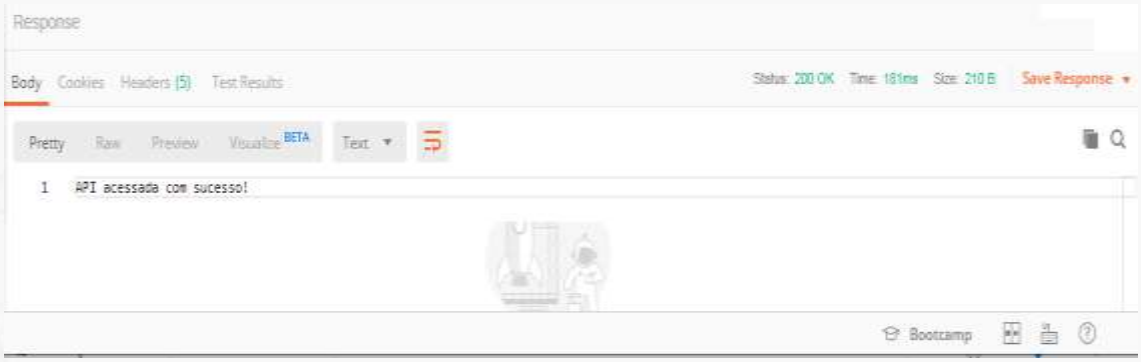
Então, obtemos o cabeçalho completo do Postman da seguinte forma:



Explicação de cada campo:

Chave	Valor
Authorization	Bearer access-token.
X-Brad-Signature	Valor da assinatura em Base 64, em linha única.
X-Brad-Nonce	Valor do nonce que foi inserido no arquivo de assinatura.
X-Brad-timestamp	Timestamp inserido no arquivo de assinatura.
X-Brad-Algoritmo	SHA256 (valor fixo).

Após os valores serem preenchidos, a chamada à **API "/jwt-service"** retornará com a seguinte resposta: **API acessada com sucesso!**



3. Suporte

Em **caso de dúvidas ou necessidade de suporte**, após seguir os procedimentos deste manual, entre em contato com a nossa **Central de Suporte, pelo seguinte e-mail:**

suporte.api@bradesco.com.br

Enviando as seguintes informações:

- ✓ Hash (assinatura) do jwt utilizado para geração do token;
- ✓ Collection (postman) de requisição de token;
- ✓ Collection (postman) chamada da api jwt-service;
- ✓ Arquivo request.txt;
- ✓ CNPJ;
- ✓ Nome da empresa que contratou o serviço junto ao Bradesco.

Conceito PIX

O **Pix é a solução de pagamento instantâneo**, criada e gerida pelo **Banco Central** do Brasil (BACEN), que proporciona a realização de **transferências e de pagamentos.**

O Pix é um meio de pagamento assim como o de boleto, **TED, DOC, transferências entre contas de uma mesma instituição e cartões de pagamento (débito, crédito e pré-pago).** A diferença é que o Pix permite que qualquer tipo de transferência e de pagamento seja realizada em qualquer dia, incluindo fins de semana e feriados, e a qualquer hora.

Chave Pix

A chave é um "apelido" utilizado para identificar sua conta. Ela representa o endereço da sua conta no Pix. Os quatro tipos de Chaves Pix que você pode utilizar são:

- ✓ CPF/CNPJ;
- ✓ e-mail;
- ✓ número de telefone celular; ou
- ✓ chave aleatória.

A chave efetua o vínculo de uma dessas informações básicas às **informações completas que identificam a conta transacional do cliente** (identificação da Instituição Financeira ou de pagamento, número da Agência, número da conta e tipo de conta).

A **chave aleatória é uma forma de você receber um Pix**, sem precisar informar quaisquer dados pessoais ao pagador. Ele será um conjunto de números, letras e símbolos gerados, aleatoriamente, que identificará sua conta e poderá ser utilizado para o recebimento de recursos.

Acesse os Canais Eletrônicos do Bradesco e realize o cadastramento da sua Chave Pix o mais rápido possível.

QR Code

Por meio dessa nova solução de pagamento instantâneo criada pelo Banco Central, sua empresa poderá gerar **QR Codes** e compartilhar com seus clientes a imagem ou URL para facilitar o pagamento. Existem dois tipos de **QR Code: Estático e Dinâmico**. Ambos servem para receber um ou mais Pix e podem ser gerados pela Instituição Financeira ou de pagamento, na qual você possua conta. Podem ser disponibilizados em papel ou em meio eletrônico. Ambos foram normatizados pelo BACEN por meio do BR Code.

O **QR Code Estático** permitirá receber pagamentos sem precisar cadastrar um valor fixo, o que permitirá ao pagador informar o valor no momento em que for realizado o pagamento, lembrando que esse tipo não possui data de vencimento ou expiração.

O **QR Code Dinâmico** apresentará as informações específicas daquela transação, como data de vencimento ou expiração, valor e multa, sendo ideal para transações únicas.

Conciliação

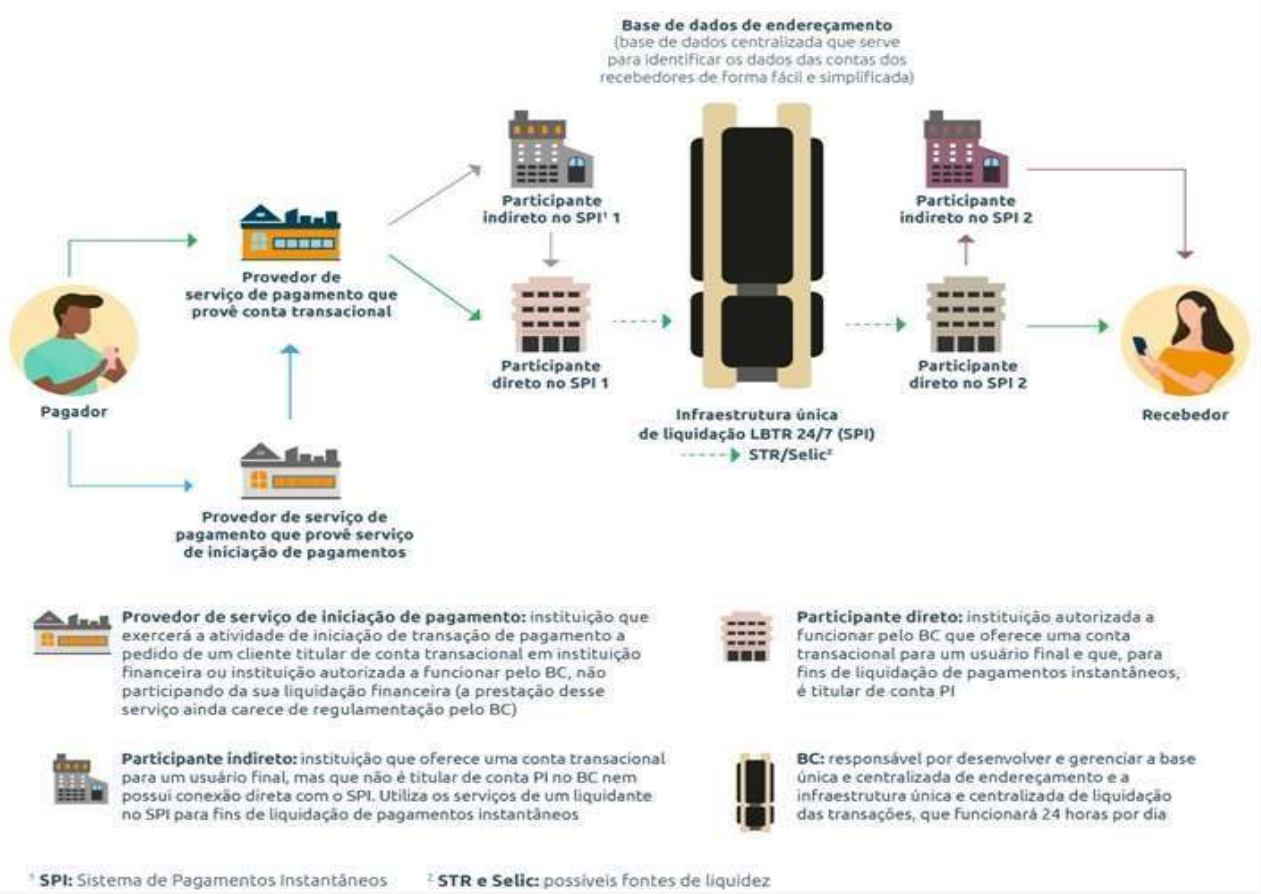
Disponibilizaremos a informação no formato e tempo ideal para atender à demanda da sua empresa, podendo ser imediato ou sob demanda.

APIREC

Trata-se de **API de Recebimentos Pix** normatizada pelo BACEN, dessa forma, todas as instituições seguirão o mesmo modelo.

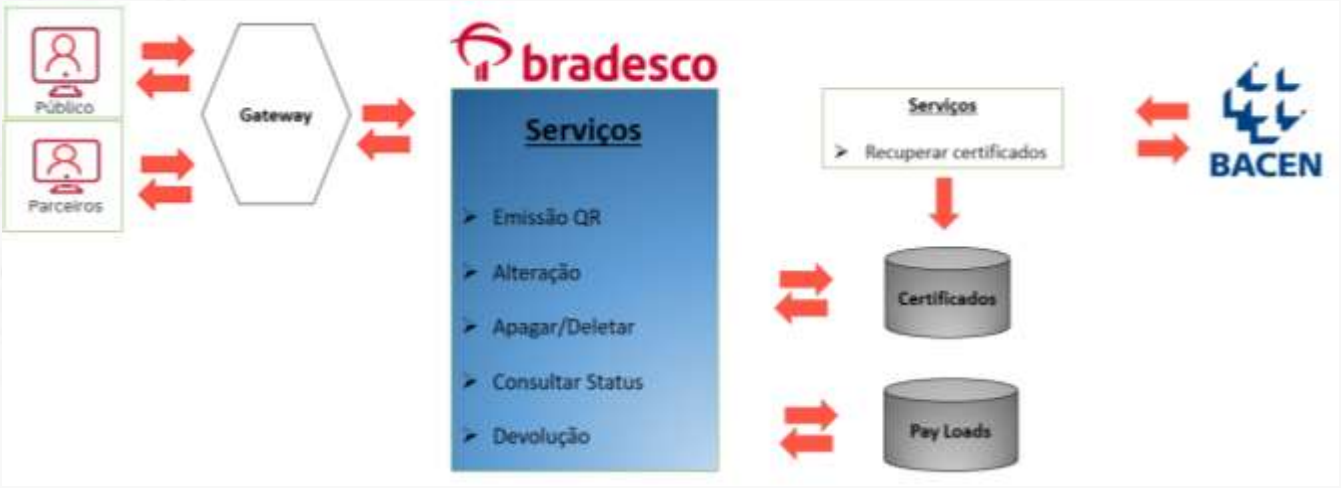
Este documento **detalha as funcionalidades disponibilizadas hoje** e serão atualizadas em conformidade com o **Órgão regulador.**

Modelo de Negócio criado pelo BACEN



Fonte: https://www.bcb.gov.br/.

Macrofluxo de Processamento Bradesco



Emissão de Qr Code Estático

API responsável por efetuar a emissão de um QR Code Estático. No momento da emissão do **QR Code**, serão validados os campos obrigatórios descritos abaixo e as regras de negócio de acordo com norma do **BACEN**.

Parâmetros de Entrada

Parâmetro	Tipo de Dado	Presença	Descrição	Observação
chave_pix	string	obrigatório	Chave Pix do recebedor cadastrada no Sistema DICT -BACEN.	Limitada 77 caracteres.
tx_id	string	opcional	ID de identificação do documento entre instituições e o cliente emissor.	O Campo Identificador (txid) é obrigatório e determina o identificador da transação, transactionID. Ele deve ser único para cada Chave Pix do recebedor. Via API, o identificador será gerado, automaticamente, pelo Bradesco e devolvido no retorno da API.
Mensagem	string	opcional	Mensagem descritiva usada na emissão do tipo estático.	-
valor.original	string	opcional	Valor nominal/cadastro do QR Code.	Se o valor não for preenchido, o pagador deverá informar no momento do pagamento.

Códigos de Status

Código	Descrição
200	Requisição realizada com sucesso.
4**	Chave não encontrada.
4**	Erros nos campos informados ou erro de validação.
4**	Dados informados estão fora do escopo definido para o campo.
500	Erro Inesperado. Entre em contato com o suporte Bradesco.

Emissão de Qr Code Dinâmico

API responsável por efetuar a emissão de um **QR Code Dinâmico**. No momento da emissão do **QR Code**, serão validados os campos obrigatórios descritos abaixo e as regras de negócio de acordo com norma do BACEN. O **Campo Identificador (txid) é obrigatório e determina o identificador da transação, transactionID. Ele deve ser único para cada Chave Pix do recebedor**. O identificador será gerado via API, automaticamente, pelo Bradesco e devolvido no retorno da mesma.

Parâmetros de Entrada

Parâmetro	Tipo de Dado	Presença	Descrição	Observação
Reutilizável	boolean	obrigatório	Indicador se o QR Code pode ser reutilizável.	Padrão de retorno é false.
chave_pix	string	obrigatório	Chave Pix do recebedor cadastrada no Sistema DICT - BACEN.	Limitada 77 caracteres.
tx_id	string	opcional	ID de identificação do documento entre instituições e o cliente emissor.	O Campo Identificador (txid) é obrigatório e determina o identificador da transação, transactionID. Ele deve ser único para cada Chave Pix do recebedor. Via API, o identificador será gerado, automaticamente,e pelo Bradesco e devolvido no retorno da API.
calendario.expiracao	string	opcional	Data e Hora de expiração do documento - yyyy-MM-dd'T'hh:mm:ss.z'Z'.	Indica os segundos, a data e a hora-limite para pagamento do Qr Code. Não pode ser preenchido junto com o calendario.vencimento. Caso nenhuma das datas sejam preenchidas, assumiremos a data de expiração após 30 dias da data de criação do QR Code.
calendario.vencimento	string	opcional	Data de vencimento do documento - yyyy-MM-dd.	Não pode ser preenchido junto com o calendario.expiracao. Caso nenhuma das datas sejam preenchidas, assumiremos a data de expiração após 30 dias da data de criação do QR Code.
calendario.recebivel_apos_vencimento	boolean	opcional	Trata-se de um campo booleano ou uma flag. Se preenchido true, significa que a cobrança pode ser paga após o vencimento. Se o campo não estiver preenchido, assume-se o valor false, ou seja, a cobrança não pode ser paga após o vencimento. Essa semântica se aplica somente a calendario.vencimento.	Padrão de retorno é false.

info_adicionais	array	opcional	-	
info_adicionais.nome	string	opcional	Nome da chave da informação (Tamanho máximo= 50).	Se for enviado, é necessário o envio do campo info_adicionais.valor.
info_adicionais.valor	string	opcional	Valor da informação (Tamanho máximo = 200).	Se for enviado, é necessário o envio do campo info_adicionais.nome.
valor.original	string	obrigatório	Valor nominal do QR Code.	Se o valor não for preenchido, o pagador deverá informar no momento do pagamento.
valor.final	string	opcional	Valor nominal final do QR Code.	-
valor.permite_alteracao	boolean	opcional	Indicador se valor da cobrança pode ser alterado durante o pagamento.	Padrão de retorno é false.
valor.juros	string	opcional	Valor do juros ao mês por atraso de pagamento do QR Code.	Aguardando definição do BACEN.
valor.multa	string	opcional	Valor da multa por atraso de pagamento do QR Code.	Aguardando definição do BACEN.
valor.desconto	string	opcional	Valor do desconto aplicado para pagamento do QR Code.	Aguardando definição do BACEN.
pagador.cpf	string	opcional	Número do Documento Cadastro de Pessoa Física.	Apenas CPF. (não enviar o pagador.cnpj).
pagador.cnpj	string	opcional	Número do Cadastro Nacional da Pessoa Jurídica.	Apenas CNPJ (não enviar o pagador.cpf).
pagador.nome	string	opcional	Nome do pagador do QR Code.	
solicitacao_pagador	string	opcional	Mensagem de solicitação do pagador ao emissor.	-

Códigos de Status

Código	Descrição
200	Requisição realizada com sucesso.
4**	Chave não encontrada.
4**	Erros nos campos informados ou erro de validação.
4**	Não encontra nenhum conteúdo seguindo os parâmetros fornecidos na entrada.
4**	Dados informados estão fora do escopo definido para o campo.
500	Erro Inesperado. Entre em contato com o suporte Bradesco.

Alterar Qr Code Dinâmico

API responsável por permitir a alteração de um **QR Code Dinâmico** por meio do **transaction ID (tx_id),** desde que o QR esteja ativo (disponível para pagamento/a vencer). Uma vez realizado o pagamento, **o QR Code Dinâmico não poderá ser alterado.**

Parâmetros de Entrada

Parâmetro	Tipo de Dado	Presença	Descrição	Observação
chave_pix	string	Obrigatório	Chave Pix do recebedor cadastrada no Sistema DICT -BACEN.	Limitada 77 caracteres.
tx_id	string	Obrigatório	TransactionID que será alterado.	Criado pelo Bradesco e informado no retorno da emissão do QR Code.
calendario.expiracao	string	Opcional	Data e Hora de expiração do documento - yyyy- MM-dd'T'hh:mm:ss.z'Z'.	Indica os segundos, a data e a hora-limite para pagamento do Qr Code. Não pode ser preenchido junto com calendario.vencimento. Caso nenhuma das datas sejam preenchidas, assumiremos a data de expiração após 30 dias da data de criação do QR Code.
calendario.vencimento	string	Opcional	Data de vencimento do documento - yyyy- MM-dd.	Não pode ser preenchido junto com calendario.expiracao. Caso nenhuma das datas sejam preenchidas, assumiremos a data de expiração após 30 dias da data de criação do QR Code.
calendario.recebivel_apos_vencimento	boolean	Opcional	Trata-se de um campo booleano ou uma flag. Se preenchido true, significa que a cobrança pode ser paga após o vencimento. Se o campo não estiver preenchido, assume- se o valor false, ou seja, a cobrança não pode ser paga após o vencimento. Essa semântica se aplica somente a calendario.vencimento.	Padrão de retorno é false.
info_adicionais	array	Opcional	-	-
info_adicionais.nome	string	Opcional	Nome da chave da informação (Tamanho máximo= 50).	Se for enviado, é necessário o envio do campo info_adicionais.valor.
info_adicionais.valor	string	Opcional	Valor da informação (Tamanho máximo = 200).	Se for enviado, é necessário o envio do campo info_adicionais.nome.
valor.original	string	Obrigatório	Valor nominal do QR Code.	Se o valor não for preenchido, o pagador deverá informar no momento do pagamento.
valor.final	string	Opcional	Valor nominal final do QR Code.	-
valor.permite_alteracao	boolean	Opcional	Indicador se o valor da cobrança pode ser alterado durante o pagamento.	Padrão de retorno é false.
valor.juros	string	Opcional	Valor do juros ao mês por atraso de pagamento do QR Code.	Aguardando definição do BACEN.
valor.multa	string	Opcional	Valor da multa por atraso de pagamento do QR Code.	Aguardando definição do BACEN.
valor.desconto	string	Opcional	Valor do desconto aplicado para pagamento do QR Code.	Aguardando definição do BACEN.
pagador.cpf	string	Opcional	Número do Documento Cadastro de Pessoa Física.	Apenas CPF (não enviar o pagador.cnpj).
pagador.cnpj	string	Opcional	Número do Cadastro Nacional da Pessoa Jurídica.	Apenas CNPJ (não enviar o pagador.cpf).
pagador.nome	string	Opcional	Nome do pagador do QR Code.	-
solicitacao_pagador	string	Opcional	Mensagem de solicitação do pagador ao emissor.	-

Códigos de Status

Código	Descrição
200	Requisição realizada com sucesso.
4**	Documento já alterado e status não será alterado.
4**	Erros nos campos informados ou erro de validação.
500	Erro Inesperado. Entre em contato com o suporte Bradesco.

Apagar/Remover Qr Code Dinâmico por Documento

API responsável por permitir apagar um payload que representa um **QR Code Dinâmico.** Uma vez apagado, ele torna-se inapto para o pagamento na Instituição pagadora (PSP pagador). O status do documento será alterado para **"removido pelo usuário recebedor".**

Vale ressaltar que a opção de **apagar/remover o QR Code** será utilizada quando a emissão de um **QR Code Dinâmico foi realizada erroneamente**. A informação será removida para pagamento, porém o QR emitido permanecerá na base histórica.

Parâmetros de Entrada

Parâmetro	Tipo de Dado	Presença	Descrição	Observação
tx_id	string	opcional	TransactionID que será excluído.	Criado pelo Bradesco e informado no retorno da emissão do QR Code.

Códigos de Status

Código	Descrição
200	Requisição realizada com sucesso.
4**	Documento já apagado e status não será alterado.
4**	Erros nos campos informados ou erro de validação.
500	Erro Inesperado. Entre em contato com o suporte Bradesco.

Consultar/Recuperar Qr Code Dinâmico por Documento Específico

API responsável por permitir recuperar as informações de um payload que representa um **QR Code Dinâmico, por meio do TransactionID gerado pelo Bradesco.**

Parâmetros de Entrada

Parâmetro	Tipo de Dado	Presença	Descrição	Observação
tx_id	string	opcional	TransactionID que será recuperado.	Criado pelo Bradesco e informado no retorno da emissão do QR Code.

Códigos de Status

Código	Descrição
200	Requisição realizada com sucesso.
4**	Documento já apagado e status não será alterado.
4**	Erros nos campos informados ou erro de validação.
500	Erro Inesperado. Entre em contato com o suporte Bradesco.

Consultar/Recuperar Lista de Qr Code Dinâmico

API responsável por permitir recuperar as informações de uma lista de payload por meio dos dados abaixo:

Parâmetros de Entrada

Parâmetro	Tipo de Dado	Presença	Descrição	Observação
data_inicio_criacao	string	opcional	Data início de emissão do Pix.	-
data_final_criacao	string	opcional	Data final de emissão do Pix.	-
pagador.cpf	string	opcional	CPF do pagador, informar o número desse documento para consulta.	-
pagador.cnpj	string	opcional	CNPJ do pagador, informar o número desse documento para consulta.	-
paginacao.pagina	string	opcional	Informar número da página que será consultada.	Se não informada, enviar como 0.
paginacao.tamanho	string	opcional	Quantidade máxima de registros informada na página.	Apenas a última página pode retornar quantidade menor que 100 (default é 100).

Códigos de Status

Código	Descrição
200	Requisição realizada com sucesso.
4**	Documento já apagado e status não será alterado.
4**	Erros nos campos informados ou erro de validação.
500	Erro Inesperado. Entre em contato com o suporte Bradesco.

Devolução de Qr Code Dinâmico

API responsável por permitir devolução do valor pago. No processo criado, o pagamento de um QR Code Dinâmico permitirá devolução em casos de acordo comercial, insatisfação com o bem/serviço ou no caso de um erro no pagamento.

A devolução ocorrerá sempre pelo recebedor. A API permitirá que o recebedor, ao iniciar a devolução, não tenha os dados da conta do pagador. A devolução poderá ocorrer em até 90 dias da data do pagamento.

Parâmetros de Entrada

Parâmetro	Tipo de Dado	Presença	Descrição	Observação
tx_id	string	obrigatório	Identificação do documento Pix.	Gerado pelo Bradesco.
id_devolucao	string	obrigatório	Até 35 posições.	Gerado pelo cliente.
valor	string	opcional	Valor da devolução. Inserir no formato: 99.99.	Não sendo informado, o valor considerado será o total pago.

Códigos de Status

Código	Descrição
200	Requisição realizada com sucesso.
4**	Documento já apagado e status não será alterado.
4**	Erros nos campos informados ou erro de validação.
500	Erro Inesperado. Entre em contato com o suporte Bradesco.