Pagamentos Instantâneos

Especificações técnicas e de negócio do ecossistema de pagamentos instantâneos brasileiro

Anexo III – Manual das Interfaces de Comunicação

Versão 1.2

BANCO CENTRAL DO BRASIL

SUMÁRIO

Apresentação

Este anexo detalha as interfaces dos sistemas que compõem o ecossistema de Pagamentos Instantâneos operado pelo Banco Central do Brasil. Nesse documento estão disponíveis todas as informações necessárias para que os prestadores de serviços de pagamentos (PSP) possam efetuar liquidações e consulta/gerenciamento de chaves de liquidação.

Estas especificações fazem parte de um trabalho em desenvolvimento. Nenhuma informação aqui apresentada deve ser considerada final.

Termos de Uso

Estas especificações são apresentadas “AS-IS”.

Referências

Estas especificações baseiam-se, referenciam, e complementam onde aplicável, os seguintes documentos:

Referência	Origem
Especificações técnicas e de negócio do ecossistema de pagamentos instantâneos brasileiro	https://www.bcb.gov.br/estabilidadefinanceira/forumpagamentosinstantaneo
Manual de Segurança do SFN	https://www.bcb.gov.br/estabilidadefinanceira/comunicacaodados
RFC 7230: Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing	https://tools.ietf.org/html/rfc7230
RFC 7231: Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content	https://tools.ietf.org/html/rfc7231
RFC 7807: Problem Details for HTTP APIs	https://tools.ietf.org/html/rfc7807
IANA: Hypertext Transfer Protocol (HTTP) Status Code Registry	https://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml
IANA: Message Headers	https://www.iana.org/assignments/message-headers/message-headers.xhtml
IANA: Internet Media Types	https://www.iana.org/assignments/media-types/media-types.xhtml
IANA: Media Type “application/xml”	https://www.iana.org/assignments/media-types/application/xml
OpenAPI	https://www.openapis.org/
Licença Apache 2.0	https://www.apache.org/licenses/LICENSE-2.0.txt

Sugestões, críticas ou pedidos de esclarecimento de dúvidas podem ser enviados ao BC por meio do e-mail pagamentosinstantaneos@bcb.gov.br.

1. Manual das interfaces do ecossistema de pagamentos instantâneos brasileiro

Introdução

Conforme descrito no documento “Especificações técnicas e de negócio do ecossistema de pagamentos instantâneos brasileiro”, existem dois grandes sistemas participantes do ecossistema de pagamentos instantâneos: um responsável por serviços relacionados à liquidação, e à gestão da conta de pagamentos instantâneos; outro responsável pela gestão e consulta de chaves de liquidação. As interfaces desses sistemas são referenciadas nesse documento por **ICOM** (Interface de Comunicação) e **DICT** (Diretório de Identificadores de Contas Transacionais), respectivamente.

1.1. Conectividade

A comunicação entre o PSP e os sistemas de Pagamentos Instantâneos se dá pela Rede do Sistema Financeiro Nacional (RSFN). A conexão do PSP com a RSFN observará as regras e padrões dispostos no Manual de Redes do SFN.

O PSP deve se conectar à API exclusivamente por meio de protocolo **HTTP versão 1.1** utilizando **TLS versão 1.2**, com autenticação mútua obrigatória no estabelecimento da conexão TLS. Deve ser suportada *a Cipher Suite ECDHE-RSA-AES-128-GCM-SHA256 (0xc02f)*.

Para a autenticação do cliente, o PSP deve utilizar certificado ICP-Brasil no padrão SPB, gerado conforme especificado no Manual de Segurança do SFN. Tal certificado deve ser previamente ativado junto ao Banco Central, conforme descrito no documento "Especificações técnicas e de negócio do ecossistema de pagamentos instantâneos brasileiro".

Para garantir desempenho e disponibilidade necessários ao sistema, o BCB poderá alterar quando necessário o destino dos nomes DNS, resolver um mesmo nome para IPs diferentes de acordo com o PSP cliente ou ainda usar a estratégia de *round-robin* via DNS para balanceamento de carga. Portanto, os clientes HTTP do PSP devem sempre respeitar o TTL (*Time To Live*) dos servidores DNS. A falha em respeitar o TTL pode causar indisponibilidade no acesso à API.

2.ICOM – Interface de Comunicação

2.1. API

A API fornece *endpoints* HTTP aos PSPs, tanto para entrega quanto para recebimento de mensagens. Não há, portanto, fornecimento de serviços HTTP pelos participantes para tráfego das mensagens ISO 20022.

Esta seção apresenta regras gerais do uso da API. Informações detalhadas sobre os endpoints e exemplos são apresentados na seção Endpoints.

2.1.1.Tratamento de erros e códigos de retorno

A API utiliza o padrão descrito no RFC 7807 para notificar erros nas requisições ou problemas internos, respondendo com *media type* "`application/problem+xml`".

```
HTTP/1.1 403 Forbidden
Content-Type: application/problem+xml

<?xml version="1.0" encoding="UTF-8"?>
<problem xmlns="urn:ietf:rfc:7807">
  <type>https://icom.pi.rsfn.net.br/api/v1/error/forbidden</type>
  <title>Acesso não permitido.</title>
</problem>
```

Erros da classe `4xx` (400-499) normalmente indicam falha na requisição feita pelo PSP. Nesse caso, PSP deve verificar a situação e não submeter a mesma requisição novamente. A falha em atender itens do manual em que consta a palavra "deve" implicará erro da classe `4xx`.

Erros da classe `5xx` (500-599) normalmente indicam algum problema na API. Contudo, há erros dessa classe causados pelo PSP.

Erros `5xx` devem ser notificados ao suporte do SPI a não ser que haja determinação contrária no manual (em algumas situações de erro `503`, por exemplo).

Excepcionalmente, em caso de erro grave da classe `5xx`, a API pode retornar conteúdo no formato "`text/plain`".

A API pode retornar diversos erros. **Entre eles**:

Código	Descrição	Observações
400	Bad Request	
403	Forbidden	

404	Not Found	
405	Method Not Allowed	
406	Not Acceptable	Ocorre se o PSP utilizar o cabeçalho `Accept` e a API não aceitar os tipos requeridos
408	Request Timeout	
410	Gone	Ocorre quando o PSP tentar acessar um recurso já removido. Usado pelo endpoint de leitura de mensagens.
411	Length Required	Ocorre quando o PSP não preencher `Content-Length` ou `Transfer-Encoding`
413	Payload Too Large	Ocorre quando houver requisições com conteúdo muito extenso. O limite aceito pela API será suficiente para a operação normal do sistema.
414	URI Too Long	
415	Unsupported Media Type	Ocorre quando o PSP falhar em usar o Content-Type ou Content-Encoding no formato correto.
429	Too Many Requests	
500	Internal Server Error	
502	Bad Gateway	
503	Service Unavailable	Ocorre quando houver muitas requisições simultâneas abertas por um mesmo PSP. Nesse caso, haverá mais informações no corpo da resposta. Ocorre caso haja outro tipo de indisponibilidade.

2.1.2.Content-Type e codificação

A API recebe e entrega mensagens no formato **XML** com codificação **UTF-8**.

Sempre que houver conteúdo em requisições, o PSP deve preencher o cabeçalho "`Content-Type`" com MIME type "`application/xml`" e parâmetro "`charset`" com valor "`utf-8`". Ex: "`application/xml; charset=utf-8`".

A falha no preenchimento do cabeçalho causa erro `415` (Unsupported Media Type).

A API, com exceção de resposta a erros (conforme seção "Tratamento de erros e códigos de retorno"), retorna conteúdo no formato XML com o cabeçalho preenchido de acordo com as regras acima.

2.1.3.Conexão persistente, Content-Length e Transfer-Encoding

O PSP deve privilegiar o uso de conexões HTTP **persistentes** conforme definido na seção 6.3 do RFC 7230. É admitido o uso de conexões não persistentes quando a frequência de requisições for baixa.

Para isso, o PSP deve enviar **um dos seguintes cabeçalhos** que possibilitam a interpretação do tamanho do corpo, quando existente:

- Content-Length; OU
- Transfer-Encoding sempre terminando com "`chunked`".

A falha no preenchimento dos cabeçalhos causa erro `411` (Length Required).

Aberturas frequentes de novas conexões impõem à API e PSPs penalidades importantes, especialmente quando feitas através de TLS. Fechamentos e aberturas muito frequentes com a API podem causar erro `503` (Service Unavailable).

Manter a conexão aberta para o envio de mensagem deverá ser avaliado por cada PSP dependendo da frequência de requisições. Fechamentos frequentes com alto fluxo de mensagens reduzirá o desempenho do sistema.

2.1.4.Conexões simultâneas

O PSP pode abrir conexões simultâneas à ICOM.

Há, no entanto, limites ao número de conexões por participante. A abertura de muitas conexões simultâneas pode causar erro `503` (Service Unavailable).

2.1.5.Compactação

O PSP e a API devem aceitar conteúdo no formato **gzip**. Para isso, devem aceitar as seguintes possibilidades de cabeçalhos:

- "`Transfer-Encoding`" com a sequência de valores "`gzip`" e "`chunked`"; OU
- "`Content-Length`" com o tamanho do corpo E "`Content-Encoding`" com o valor "`gzip`".

Exemplos válidos:

```
POST /api/v1/in/11111111/msgs HTTP/1.1
Host: icom.pi.rsfn.net.br
Transfer-Encoding: gzip, chunked
Content-Type: application/xml; charset=utf-8

[xml compactado com gzip]

---
```

```
POST /api/v1/in/11111111/msgs HTTP/1.1
Host: icom.pi.rsfn.net.br
Content-Length: 1874
Content-Encoding: gzip
Content-Type: application/xml; charset=utf-8

[xml compactado com gzip]
```

Não há suporte a outros tipos de compressão de dados. Tentativa de uso de outros formatos resultam em erro `400` (Bad Request) ou `415` (Unsupported Media Type).

O suporte a gzip é obrigatório. O seu uso não é obrigatório, mas recomendável.

2.1.6.Host

Conforme descrito na seção 5.4 do RFC 7230, é obrigatório o envio do cabeçalho "`Host`" em todas as requisições HTTP.

2.1.7.User-Agent

O cabeçalho "`User-Agent`" pode ser usado pelo PSP para enviar informações adicionais sobre o cliente. Seu conteúdo será guardado em log junto às demais informações da requisição e pode ser usado para auxiliar o diagnóstico de problemas.

2.1.8.Cabeçalhos adicionais

A API pode recusar uma requisição que contenha cabeçalhos diferentes de:

- Host
- Content-Length
- Transfer-Encoding
- Content-Type
- Content-Encoding
- User-Agent
- Accept: se presente, deve sempre ser "application/xml" ou "*/*".
- Accept-Encoding: se presente, deve sempre ser "gzip".

2.1.9.Exemplos

A seção atual fornece exemplos de algumas combinações de requisições e respostas.

A seção "Endpoints" descreve com detalhes os endpoints fornecidos pela API.

POST com conteúdo compactado e Content-Length:

```
-->
POST /api/v1/in/11111111/msgs HTTP/1.1
Host: icom.pi.rsfn.net.br
Content-Type: application/xml; charset=utf-8
Content-Length: 1874
Content-Encoding: gzip
User-Agent: instancia 1; versao 1.1

[xml compactado com gzip]

<---
HTTP/1.1 201 Created
PI-ResourceId: UEkBbgR78VqKK/uvuq9O0r9bqXyBH9Ur
POST com conteúdo compactado e Transfer-Encoding:
-->
POST /api/v1/in/11111111/msgs HTTP/1.1
Host: icom.pi.rsfn.net.br
Content-Type: application/xml; charset=utf-8
Transfer-Encoding: gzip, chunked
User-Agent: instancia 1; versao 1.1

[xml compactado com gzip]

<---
HTTP/1.1 201 Created
PI-ResourceId: UEkBbgR78VqKK/uvuq9O0r9bqXyBH9Ur
POST com conteúdo não compactado e Content-Length:
-->
POST /api/v1/in/11111111/msgs HTTP/1.1
Host: icom.pi.rsfn.net.br
Content-Type: application/xml; charset=utf-8
Content-Length: 4587
User-Agent: instancia 1; versao 1.1

[xml]

<---
HTTP/1.1 201 Created
PI-ResourceId: UEkBbgR78VqKK/uvuq9O0r9bqXyBH9Ur
Erro:
-->
POST /api/v1/in/11111111/msgs HTTP/1.1
Host: icom.pi.rsfn.net.br
Content-Type: application/xml; charset=utf-16
Content-Length: 8976

[xml]

<---
HTTP/1.1 400 Bad Request
Content-Length: ???
Content-Type: application/problem+xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<problem xmlns="urn:ietf:rfc:7807">
  <type>https://icom.pi.rsfn.net.br/api/v1/error/charset</type>
  <title>Charset XML inválido.</title>
</problem>
```

2.2. Endpoints

A URL base da API é:

- `https://icom.pi.rsfn.net.br` para o ambiente de **produção**;
- `https://icom-h.pi.rsfn.net.br` para o ambiente de **homologação**.

2.2.1.Entrada: envio de mensagens pelo PSP

```
POST /api/v1/in/{ispb}/msgs
```

Transmite uma mensagem ao SPI.

`{ispb}` corresponde ao código ISPB **do PSP**.

Nesse momento não é realizada validação sintática ou de assinatura do XML. A API recebe a mensagem e a armazena para processamento.

2.2.1.1. REQUISIÇÃO

O corpo da requisição deve conter o XML da mensagem correspondente.

A API admite que o PSP envie a mesma mensagem mais de uma vez. Nesse caso, a mensagem será gravada novamente na ICOM e processada de acordo com as regras sobre idempotência descritas no documento “Especificações técnicas e de negócio do ecossistema de pagamentos instantâneos brasileiro”.

2.2.1.2. RESPOSTA

Códigos:

Código	Descrição
201	A mensagem foi gravada.
4xx	Erro do PSP. A mensagem não foi gravada.
503	Ler seções sobre conexão persistente e conexões simultâneas. A mensagem não é gravada.
5xx	Erro da API. A mensagem pode ou não ter sido gravada.

Cabeçalhos:

Cabeçalho	Descrição
PI-ResourceId	Identificação da mensagem gravada.

A API retorna um código `201` (Created) se o envio for bem-sucedido. Nesse caso, a mensagem foi gravada no SPI e será processada em seguida. A API ainda retorna um cabeçalho "`PI-ResourceId`" descrito em outra seção.

Se a API retornar erro da classe `4xx`, a mensagem não é gravada no SPI. Portanto, o PSP deve avaliar e corrigir o erro e enviar a mensagem novamente, se for o caso.

Se a API retornar erro da classe `5xx`, a mensagem **pode ou não** ter sido gravada. Uma vez que o sistema de pagamento tem comportamento idempotente, o PSP deve enviar a mensagem novamente quando possível.

2.2.1.3. PI-RESOURCEID

O cabeçalho "`PI-ResourceId`" retornado em algumas respostas da API identifica um armazenamento de mensagem no SPI. O PSP deve aceitar conteúdo em formato Base64 com até 32 caracteres (equivalente a 24 bytes).

A referência retornada em "`PI-ResourceId`" é usada como identificador da mensagem na API e no corpo das mensagens `admi.002` enviadas pelo SPI ou PSP quando ocorrer erro de validação ou assinatura.

Caso o PSP envie uma mesma mensagem mais de uma vez, cada envio retornará um "`PI-ResourceId`" distinto.

2.2.1.4. EXEMPLO

```
-->
POST /api/v1/in/11111111/msgs HTTP/1.1
Host: icom.pi.rsfn.net.br
Content-Type: application/xml; charset=utf-8
Transfer-Encoding: gzip, chunked

[xml compactado com gzip]

<---
HTTP/1.1 201 Created
PI-ResourceId: UEkBbgR78VqKK/uvuq9O0r9bqXyBH9Ur
```

2.2.2.Saída: recebimento de mensagens pelo PSP

O processo de leitura das mensagens da API destinadas ao PSP exige um algoritmo mais complexo a fim de garantir todas as entregas ao PSP.

É responsabilidade do PSP buscar as mensagens disponíveis a ele. Para isso, deve iniciar a leitura da saída através de um GET como o que segue:

```
GET /api/v1/out/{ispb}/stream/start
```

`{ispb}` corresponde ao código ISPB **do PSP**.

2.2.2.1. RESPOSTA

Códigos:

Código	Descrição
200	Há uma mensagem disponível. A mensagem vem no corpo da resposta.
204	Não há mensagens disponíveis ao PSP.
4xx	Erro do PSP.
503	Ler seções sobre conexão persistente e conexões simultâneas.
5xx	Erro da API.

Cabeçalhos:

Cabeçalho	Descrição
PI-ResourceId	Identificação da mensagem. Apenas quando o código é 200.
PI-Pull-Next	Caminho para leitura da próxima mensagem. Quando o código de retorno for `200` ou `204`.

A API retorna um dos seguintes códigos:

- `200`: quando houver uma mensagem disponível. Nesse caso, a mensagem será enviada no corpo da resposta. Além disso, retorna o cabeçalho adicional "`PI-ResourceId`" com a identificação da mensagem na API conforme descrito na seção PI-ResourceId.
- `204`: quando não houver nenhuma mensagem disponível para o PSP.

Para as respostas `200` ou `204`, a API retorna também o cabeçalho "`PI-Pull-Next`" com o caminho que o PSP deve usar para a próxima leitura. O caminho pode ser um URI relativo ou absoluto. Exemplo:

- `/api/v1/out/11111111/stream/f0d744d792d6`
- `https://icom.pi.rsfn.net.br/api/v1/out/11111111/stream/92bb31c8a623`

Exemplo (com alguns cabeçalhos omitidos):

```
--->
GET /api/v1/out/11111111/stream/start HTTP/1.1
<---
HTTP/1.1 200 OK
PI-ResourceId: UEkBbgR78VqKK/uvuq9O0r9bqXyBH9Ur
```

```
PI-Pull-Next: /api/v1/out/11111111/stream/f0d744d792d6

[xml]
```

O PSP não deve assumir qualquer formato no conteúdo do cabeçalho "`PI-Pull-Next`". Os exemplos de conteúdo neste manual podem não corresponder ao formato real da API. Atualizações na implementação da API podem mudar o formato do conteúdo sem aviso prévio.

2.2.2.2. ITERAÇÕES SEGUINTES

Após a resposta ao "`GET (...)/stream/start`", o PSP deve gravar a mensagem recebida e utilizar o caminho retornado no cabeçalho "`PI-Pull-Next`" para prosseguir a leitura:

```
--->
GET /api/v1/out/11111111/stream/f0d744d792d6 HTTP/1.1
<---
HTTP/1.1 200 OK
PI-Pull-Next: /api/v1/out/11111111/stream/c7e3d74bca8e
...
```

O retorno obedecerá às mesmas regras da resposta ao "`GET (...)/stream/start`".

Cada GET deve usar como caminho o retornado pelo "`PI-Pull-Next`" da requisição imediatamente anterior.

O PSP deve repetir esse processo até que deseje interromper o processo de leitura.

A falha em usar o retornado em "`PI-Pull-Next`" fará a API apresentar erros ou comportamento imprevisto, entre eles a entrega excessiva de mensagens em duplicidade.

2.2.2.3. INTERRUPÇÃO

Quando o PSP desejar interromper a leitura de mensagens, deve finalizar o processo através de uma requisição como a que segue:

```
DELETE /api/v1/out/11111111/stream/c7e3d74bca8e HTTP/1.1
```

O caminho deve ser o retornado no "`PI-Pull-Next`" da requisição anterior.

Caso bem-sucedido, o DELETE pode retornar código `200` (OK) ou `410` (Gone). Demais erros devem ser tratados como descrito na seção sobre tratamento de erros.

O método, além de liberar recursos na API, tem a finalidade de informar à ICOM que as últimas mensagens entregues ao PSP foram corretamente guardadas, marcando-as como efetivamente lidas. Portanto, a falha em finalizar a leitura com DELETE pode causar a entrega posterior de mensagens em duplicidade ou ainda atraso na entrega de algumas mensagens.

2.2.2.4. ALGORITMO

De forma simplificada, o algoritmo para leitura de mensagens pelo PSP pode ser ilustrado da seguinte forma:

```
URL = "https://icom.pi.rsfn.net.br/api/v1/out/11111111/stream/start"

ENQUANTO (em operação)
  R = HTTP.GET(URL)
  URL = R.Cabeçalhos("PI-Pull-Next")
  SE (R.Codigo = 200) ENTÃO GravaMensagem(R.Corpo)
FIM

HTTP.DELETE(URL)
```

2.2.2.5. EXEMPLO

Exemplo do processo de leitura (com cabeçalhos omitidos):

1. Início da leitura:

```
-->
GET /api/v1/out/11111111/stream/start
```

2. Retorno com uma mensagem

```
<--
HTTP/1.1 200 OK
PI-Pull-Next: /api/v1/out/11111111/stream/c2e3a665c1a9
PI-ResourceId: UEkBbgR78VqKK/uvuq9O0r9bqXyBH9Ur

[mensagem]
```

3. Segunda iteração:

```
-->
GET /api/v1/out/11111111/stream/c2e3a665c1a9
```

4. Retorno após alguns segundos sem nenhuma mensagem

```
<--
HTTP/1.1 204 No Content
PI-Pull-Next: /api/v1/out/11111111/stream/e839cf0015bb
```

5. Terceira iteração

```
-->
GET /api/v1/out/11111111/stream/e839cf0015bb
```

6. Retorno

```
<--
HTTP/1.1 200 OK
PI-ResourceId: dTkENva+UQ6gnwMrojr1NLxC/3Ot32bd
PI-Pull-Next: /api/v1/out/11111111/stream/8e20a682c86f

[mensagem]
```

7. Fim da leitura

```
-->
DELETE /api/v1/out/11111111/stream/8e20a682c86f
```

8. Resposta

```
<---
HTTP/1.1 200 OK
```

2.2.2.6. PROCESSAMENTO DA MENSAGEM

O PSP deve usar o menor tempo possível entre as requisições de leitura. Dessa forma, não é recomendável que faça processamento complexo nesse momento.

É recomendável que o PSP apenas a armazene e notifique outro elemento do sistema para processar a mensagem de forma assíncrona.

A demora entre requisições aumenta a latência do sistema e pode causar envio excessivo de mensagens em duplicidade.

2.2.2.7. LONG POLLING

Em qualquer tentativa de leitura (`GET`), a API não responde imediatamente caso não haja nenhuma mensagem disponível ao PSP. Ao invés disso, aguarda por alguns segundos até que uma mensagem esteja disponível.

Após o tempo decorrido sem nenhuma mensagem ao PSP, a API retornará um código `204` (No Content). A resposta 204 retorna, também, um cabeçalho "`PI-Pull-Next`" que deve ser usado na próxima requisição.

O "PI-Pull-Next" de um `204` pode ser idêntico ao retornado anteriormente, a critério da API. É importante que o PSP sempre obedeça ao que vier nesse cabeçalho.

2.2.2.8. DUPLICAÇÃO EM MENSAGENS RECEBIDAS PELO PSP

Ocasionalmente, o PSP poderá receber uma mensagem já recebida anteriormente, mesmo que o PSP faça uma requisição com o "PI-Pull-Next" correto. Nesse caso, o "PI-ResourceId" da mensagem será o mesmo e o PSP pode utilizá-lo para identificar a duplicidade.

2.2.2.9. LEITURA SIMULTÂNEA

Caso o PSP deseje consumir mensagens simultaneamente deve iniciar cada processo através do "`GET /api/v1/out/{ispb}/stream/start`". Dessa forma, cada requisição ao recurso `stream/start` pode fazer a API criar um "stream" de leitura.

2.3. Pendências

Esta seção apresenta assuntos em discussão que poderão causar mudanças importantes no manual no futuro.

Multipart

O protocolo HTTP permite o tráfego de conteúdo em qualquer formato. A versão atual do manual admite XML compactado. Contudo, o protocolo ainda limita a transmissão de apenas uma mensagem por requisição. Isso, aliado à necessidade do estilo de transmissão requisição-resposta adotado pelo protocolo HTTP, pode exigir uma grande quantidade de conexões e requisições abertas simultâneas.

A possibilidade de tráfego de mais de uma mensagem em uma mesma rodada de requisição-resposta abre caminho para maior eficiência no uso de rede, especialmente com PSPs com alto tráfego de mensagens.

Está em estudo a possibilidade de uso, em algumas situações, do *media type* "`multipart/mixed`".

Limites

Alguns pontos do manual determinam que haverá limitações em quantidade de conexões, requisições, ciclos de fechamento e abertura de novas requisições e tamanho do corpo da requisição HTTP.

Esses limites serão determinados posteriormente.

Rate Limit

Há mecanismos que auxiliam os clientes no acompanhamento de cotas de requisição. É possível que a API adote o descrito no documento RateLimit Header Fields for HTTP.

3. DICT – Diretório de Identificadores de Contas Transacionais

3.1. OpenAPI

O DICT tem sua API e Endpoints especificados conforme o padrão OpenAPI, versão 3.0.0. A documentação HTML pode ser acessada no arquivo "API do Diretório de Identificadores de Contas Transacionais.zip", ou através do arquivo de definição do OpenAPI disponível em https://github.com/bacen/pix-dict-api.

3.2. Códigos Públicos do DICT

O DICT expõe o código necessário para integração em dois projetos do GitHub:

Repositório	Finalidade
https://github.com/bacen/pix-dict-api	Arquivo com a especificação OpenAPI que define a interface do DICT.
https://github.com/bacen/pix-dict-quickstart	Projeto "quickstart" para auxiliar os desenvolvedores a fazer as primeiras iterações com o DICT, registrando uma chave e consultando o diretório. **IMPORTANTE**: Esse projeto é disponibilizado com o propósito didático, e não deve ser utilizado como base para construção da integração final do PSP com o DICT.

Todo código público do DICT é disponibilizado sob a licença Apache 2.0.

3.3. Pendências

A versão atual da Interface do DICT não utiliza assinaturas. Qualquer requisição utilizando o campo `<Signature>` não terá garantia de processamento, tampouco serão assinadas as respostas das operações por parte do DICT. A assinatura das mensagens, contudo, será obrigatória na versão final da API.

Histórico de revisão

Data	Versão	Descrição das alterações
14/11/2019	1.0	Versão inicial.
17/01/2020	1.1	Conectividade: - Inclusão de referência ao Manual de Redes da RSFN - Atualização do link para o Manual de Segurança RSFN - Detalhamento do comportamento do DNS Endpoints: - Mudança da URL base de produção e inclusão da URL de homologação Endpoint Entrada: - Esclarecimento sobre idempotência no envio de mensagens duplicadas. Endpoint Saída: - Esclarecimento sobre interrupção. Diversos: - Cabeçalhos SPI-ResourceId e SPI-PullNext alterados para PI-ResourceId e PI-PullNext.
17/02/2020	1.2	Incluída a interface do DICT. Renomeado o arquivo de “Manual da Interface de Comunicação” para “Manual das Interfaces de Comunicação”.