



pix
powered by Banco Central

Manual das Interfaces de Comunicação

Versão 1.12

SUMÁRIO

Apresentação	4
Referências	5
1. Manual das interfaces do ecossistema de pagamentos instantâneos brasileiro	6
Introdução	6
1.1. Conectividade	7
1.2. Canais de transmissão de mensagens	7
2. ICOM – Interface de Comunicação do SPI	8
2.1. API	8
2.1.1. Tratamento de erros e códigos de retorno	8
2.1.2. Content-Type e codificação	9
2.1.3. Conexão persistente, Content-Length e Transfer-Encoding	10
2.1.4. Conexões simultâneas	10
2.1.5. Compactação	10
2.1.6. Host	11
2.1.7. User-Agent	12
2.1.8. Cabeçalhos adicionais	12
2.1.9. Exemplos	12
2.2. Endpoints	14
2.2.1. Entrada: envio de mensagens pelo PSP	14
2.2.1.1. Requisição	14
2.2.1.2. Resposta	15
2.2.1.3. PI-ResourceId	15
2.2.1.4. Exemplos	16
2.2.1.5. Limitação de tráfego de mensagens e operações	17
2.2.2. Saída: recebimento de mensagens pelo PSP	19
2.2.2.1. Resposta	20
2.2.2.2. Iterações seguintes	21

2.2.2.3.	Interrupção.....	22
2.2.2.4.	Algoritmo.....	22
2.2.2.5.	Exemplo.....	23
2.2.2.6.	Processamento da mensagem	24
2.2.2.7.	Long polling	24
2.2.2.8.	Duplicação em mensagens recebidas pelo PSP	25
2.2.2.9.	Leitura simultânea.....	25
2.2.2.10.	Otimizando o processo de leitura em cenários de carga – resumo geral	25
2.2.3.	Catálogo: versões das mensagens aceitas e enviadas pela API	26
2.2.3.1.	Versões das mensagens aceitas na Entrada.....	26
2.2.3.2.	Versões das mensagens enviadas na Saída.....	26
2.2.3.3.	Exemplos	26
3.	DICT – Diretório de Identificadores de Contas Transacionais.....	28
3.1.	Interface	28
3.2.	Segurança.....	28
3.3.	OpenAPI.....	28
3.4.	Códigos Públicos do DICT	28
4.	ARQ – Serviço de Transferência de Arquivos	29
4.1.	Interface	29
4.2.	API	29
	Histórico de revisão.....	30

Apresentação

Este anexo detalha as interfaces de comunicação dos sistemas operados pelo Banco Central do Brasil que compõem o ecossistema de pagamentos instantâneos, a saber: o Sistema de Pagamentos Instantâneos (SPI) e o Diretório de Identificadores de Contas Transacionais (DICT).

Referências

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

Referência	Origem
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
Regulamento do Pix	https://www.bcb.gov.br/estabilidadefinanceira/exibenormativo?tipo=Resolu%C3%A7%C3%A3o%20BCB&numero=1

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

Existem dois grandes sistemas operados pelo BCB que participam do ecossistema de pagamentos instantâneos: o Sistema de Pagamentos Instantâneos (SPI), responsável por serviços relacionados à liquidação e à gestão da Conta Pagamentos Instantâneos; e o Diretório de Identificadores de Contas Transacionais (DICT), responsável pela gestão e consulta de chaves de identificação de contas transacionais dos usuários finais.

As interfaces desses sistemas são referenciadas nesse documento por **ICOM** (Interface de Comunicação do SPI) e **DICT** (Diretório de Identificadores de Contas Transacionais), respectivamente.

1.1. Conectividade

A comunicação entre o PSP e os sistemas operados pelo BCB que compõem o ecossistema 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](#).

Para garantir desempenho e disponibilidade necessários aos sistemas, 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.

1.2. Canais de transmissão de mensagens

Para o tratamento racional das ordens de pagamento que cursam pelo SPI, a ICOM provê duas estruturas apartadas para transmissão de mensagens:

- Canal primário de transmissão de mensagens (CPM) ou simplesmente **Canal primário**: canal de transmissão por meio do qual serão enviadas as ordens para as quais o usuário final possui expectativa de liquidação imediata. Exemplos: pagamentos não agendados, pagamentos feitos nos caixas de estabelecimentos comerciais, devoluções etc. Nesses casos, o cliente pagador espera que a liquidação ocorra em poucos segundos (~10s).
- Canal secundário de transmissão de mensagens (CSM), ou simplesmente **Canal secundário**: canal de transmissão por meio do qual serão enviadas as ordens para as quais o usuário final não possui expectativa de liquidação imediata. Exemplos: pagamentos agendados, pagamentos feitos em grandes lotes, Pix cobrança com data de vencimento etc. Nesses casos, o cliente pagador espera que a liquidação ocorra em uma data específica, mas sem nenhuma expectativa com relação ao horário exato do pagamento.

2. ICOM – Interface de Comunicação do SPI

2.1. API

A API do **SPI** 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 do **SPI**. 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: com valor “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 o cabeçalho “Content-Encoding” com valor “gzip”.

Exemplos válidos:

```
POST /api/v1/in/11111111/msgs HTTP/1.1
Host: icom.pi.rsfn.net.br
Transfer-Encoding: chunked
Content-Encoding: gzip
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 no canal primário não é obrigatório, mas altamente recomendável.

No canal secundário o uso de compactação gzip é obrigatório.

O uso de “Transfer-Encoding” com a sequência de valores “gzip” e “chunked” não é mais recomendado e poderá ser desabilitado no futuro. Recomendamos usar o cabeçalho “Content-Encoding” para enviar conteúdo em gzip.

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.

O conteúdo do cabeçalho deverá seguir **sem a porta**.

- **Exemplo** de uso suportado:
 - “Host: servico.pi.rsfn.net.br”
- **Exemplos** de uso **não suportados**:

“Host: servico.pi.rsfn.net.br:16422”

- “Host: servico.pi.rsfn.net.br:443”

O envio com a porta poderá causar erros na conexão.

POST com conteúdo compactado e Transfer-Encoding:

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

[xml compactado com gzip]

<---
HTTP/1.1 201 Created
PI-ResourceId: UEkBbgR78VqKK/uvuq900r9bqXyBH9Ur
```

POST com conteúdo não compactado e Content-Length:

```
-->
POST /api/v1/in/11111111/messages 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/uvuq900r9bqXyBH9Ur
```

Erro:

```
-->
POST /api/v1/in/11111111/messages 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 do **SPI** é:

- Para o ambiente de **produção do SPI**
 - `https://icom.pi.rsfn.net.br:16422` para o canal primário;
 - `https://icom-sec.pi.rsfn.net.br:17422` para o canal secundário.
- Para o ambiente de **homologação do SPI**
 - `https://icom-h.pi.rsfn.net.br:16522` para o canal primário;
 - `https://icom-sec-h.pi.rsfn.net.br:17522` para o canal secundário.

2.2.1. Entrada: envio de mensagens pelo PSP

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

Transmite uma ou mais mensagens ao **SPI**.

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

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

2.2.1.1. REQUISIÇÃO

O *endpoint* admite o envio de uma ou mais mensagens.

O envio de apenas uma mensagem deve ser feito utilizando "Content-Type" com valor "application/xml; charset=utf-8".

Para enviar múltiplas mensagens, a API admite Content-Type do tipo "multipart/mixed". Nesse caso, cada parte deve conter o cabeçalho "Content-Type" com o valor "application/xml; charset=utf-8". Há um limite de 10 mensagens em uma mesma requisição.

A API admite que o PSP envie a(s) mesma(s) mensagem(s) mais de uma vez. Nesse caso, será(ão) gravada(s) novamente na ICOM e processada(s) de acordo com as regras sobre idempotência.

2.2.1.2. RESPOSTA

Códigos:

Código	Descrição
201	A gravação foi realizada com sucesso.
4xx	Erro do PSP. Não houve gravação de nenhuma mensagem.
503	Ler seções sobre conexão persistente e conexões simultâneas . Não houve gravação.
5xx	Erro da API. A(s) mensagem(s) pode(m) ou não ter sido gravada(s). Com o envio de múltiplas mensagens, é possível que alguma tenha sido gravada e outra não.

Cabeçalhos:

Cabeçalho	Descrição
PI-ResourceId	Identificação da(s) mensagem(s) gravada(s).

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

Se a API retornar erro da classe 4xx, nenhuma mensagem é gravada no **SPI**. Portanto, o PSP deve avaliar e corrigir o erro e realizar o envio novamente, se for o caso.

Se a API retornar erro da classe 5xx, a(s) mensagem(s) **pode(m) ou não** ter sido gravada(s). No caso de envio com *multipart*, é possível que algumas mensagens tenham sido gravadas e outras não. Uma vez que o **SPI** tem comportamento idempotente, o PSP deve tentar o envio novamente quando possível.

2.2.1.3. PI-RESOURCEID

O cabeçalho "PI-ResourceId" – protocolo de armazenamento de mensagem no SPI – retornado em algumas respostas da API identifica o armazenamento de cada mensagem no **SPI**.

A identificação de cada mensagem tem até 32 caracteres em formato Base64 (equivalente a 24 bytes). No *endpoint* de entrada, quando a requisição tratar de várias mensagens (*multipart*), esse cabeçalho carregará as identificações de cada mensagem separadas por vírgula (",").

Nos casos de *multipart* no *endpoint* de saída (seção 0), cada parte carregará como cabeçalho o seu ResourceId.

As referências retornadas em “PI-ResourceId” são usadas como identificador das mensagens 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.

No caso de mensagens entregues pela API ao PSP, quando houver envio em duplicidade, o ResourceId será idêntico em todas as entregas da mesma mensagem.

2.2.1.4. EXEMPLOS

Envio de mensagem avulsa:

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

[xml compactado com gzip]

<---
HTTP/1.1 201 Created
PI-ResourceId: UEkBbgR78VqKK/uvuq900r9bqXyBH9Ur
```

Envio de múltiplas mensagens com *multipart*:

```
-->
POST /api/v1/in/11111111/messages HTTP/1.1
Host: icom.pi.rsfn.net.br
Content-Type: multipart/mixed; boundary="simple boundary"
Transfer-Encoding: chunked
Content-Encoding: gzip

preamble. será ignorado.
--simple boundary
Content-type: application/xml; charset=utf-8

<xml>Mensagem 1</xml>
--simple boundary
Content-type: application/xml; charset=utf-8

<xml>Mensagem 2</xml>
--simple boundary--

<---
HTTP/1.1 201 Created
PI-ResourceId: UEkBbgR78VqKK/uvuq900r9bqXyBH9Ur,dTkENva+UQ6gnwMroj1NLxC/30t32bd
```

Corpo compactado via gzip

2.2.1.5. LIMITAÇÃO DE TRÁFEGO DE MENSAGENS E OPERAÇÕES

A fim de proteger o ecossistema contra falhas de implementação ou uso indevido do trilha de pagamentos instantâneos, a API implementa uma estratégia de limitação de tráfego adaptada do conceito de [token bucket](#).

Nesse processo, um saldo de tokens individualizado para cada PSP é creditado automaticamente ao longo do tempo até um determinado limite e debitado quando do processamento de mensagens do PSP. Se o saldo de um PSP for completamente consumido, a API rejeitará quaisquer novas mensagens até que se torne novamente positivo. Nesse caso, retornará o código HTTP 429 (Too Many Requests) e cabeçalho "Retry-After" preenchido com a previsão de tempo, em segundos, em que o saldo se tornará novamente positivo.

O débito do saldo de tokens é atualizado durante o processamento das mensagens, e não na recepção. Dessa forma, pode haver um pequeno atraso entre o recebimento de mensagens e o consumo desse saldo. Uma vez que mensagens podem ser enviadas em pacotes (*multipart*) e que cada uma pode carregar múltiplas operações, o saldo de tokens admite valor negativo.

Os parâmetros desse mecanismo, para cada PSP, são:

Canal	Horário	Tamanho Máximo do Bucket	Acréscimo Automático por Segundo
Primário	—	3750	750
Secundário (Regra Geral)	08:00:00 – 23:59:59	350	35
Secundário (Regra Geral)	00:00:00 – 07:59:59	5000	350
Secundário (trck.002, booktransfer)	—	1500	500

O Banco Central do Brasil poderá, a seu critério, alterar os parâmetros acima para um participante ou um conjunto de participantes. Para o canal secundário, nos horários próximos as mudanças de valores do bucket, não recomendamos o envio massivo de mensagens. O saldo é consumido da seguinte forma:

- 1 token para cada operação contida em mensagens pacs.008 e trck.002
- 0,2 token para cada operação contida em mensagens pacs.002
- 1 token para cada mensagem que não seja pacs.008, trck.002 ou pacs.002

Essa estratégia implementa limites para proteção do ecossistema, sendo aplicada a todas as mensagens cursadas nos canais primário e secundário da ICOM, e não exige

o PSP de atentar para a classificação adequada das prioridades dos pagamentos nas mensagens pacs.008.

Buscando otimizar a distribuição do envio de mensagens no canal secundário ao longo do período da madrugada, apresentamos um exemplo a seguir: Considere que devem ser enviadas 1.000.000 de operações, **idealmente distribuídas** ao longo das 8 horas, em requisições de 5.000 operações, sendo 10 mensagens via multipart e 500 operações por mensagem. Neste exemplo, o intervalo ideal entre cada requisição deve ser de 144 segundos.

De forma a ilustrar melhor o funcionamento desse mecanismo, segue tabela com uma simulação de um PSP enviado mensagens em diferentes cenários no decorrer do tempo, considerando, por exemplo, uma reposição de 500 tokens/s e um bucket de 2500:

Segundo	Operações Enviadas	Debitado do Bucket	Creditado no Bucket	Saldo do Bucket
1	500	500	500	2500
2	1000	1000	500	2000
3	3500	3500	500	-1000
4	Saldo T-1 do Bucket negativo * A API rejeita qualquer mensagem	0	500	-500
5	Saldo T-1 do Bucket negativo * A API rejeita qualquer mensagem	0	500	0
6	Saldo T-1 do Bucket é zero * A API rejeita qualquer mensagem	0	500	500
7	1000	1000	500	-500
8	Saldo T-1 do Bucket negativo * A API rejeita qualquer mensagem	0	500	0
9	Saldo T-1 do Bucket é zero * A API rejeita qualquer mensagem	0	500	500
10	500	500	500	500
11	100	100	500	900
12	50	50	500	1350
(T+n)	(...) quantidades menores que o saldo em T-1	(...)	500	Até o limite de 2500

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**.

O PSP pode optar por receber mensagens avulsas (uma por requisição) ou receber pacotes com uma ou mais mensagens em uma mesma resposta. A escolha é feita através do cabeçalho "Accept". Quando o cabeçalho for inexistente ou seu valor for "application/xml", a API retornará apenas uma mensagem. Quando seu valor for "multipart/mixed", poderá retornar múltiplas mensagens em uma mesma resposta.

A API retorna, no máximo, 10 mensagens a cada requisição com *multipart*.

2.2.2.1. RESPOSTA

Códigos:

Código	Descrição
200	Há uma ou mais mensagem(s) disponível(s). A(s) mensagem(s) 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(s) mensagem(s). Apenas quando o código é 200.
PI-Pull-Next	Caminho para leitura da(s) próxima(s) mensagem(s). Quando o código de retorno for 200 ou 204.

A API retorna um dos seguintes códigos:

- 200: quando houver mensagem disponível. Nesse caso, a(s) mensagem(s) será(ão) enviada(s) no corpo da resposta. Além disso, retorna o cabeçalho adicional "PI-ResourceId" com a identificação da(s) mensagem(s) 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:16422/api/v1/out/11111111/stream/92bb31c8a623>

Exemplos (com alguns cabeçalhos omitidos):

Leitura avulsa:

```
--->
GET /api/v1/out/11111111/stream/start HTTP/1.1
<---
HTTP/1.1 200 OK
PI-ResourceId: UEkBbgR78VqKK/uvuq900r9bqXyBH9Ur
PI-Pull-Next: /api/v1/out/11111111/stream/f0d744d792d6

[xml]
```

Leitura admitindo multipart:

```
--->
GET /api/v1/out/11111111/stream/start HTTP/1.1
Accept: multipart/mixed
<---
HTTP/1.1 200 OK
PI-Pull-Next: /api/v1/out/11111111/stream/f0d744d792d6
Content-Type: multipart/mixed; boundary="simple boundary"

--simple boundary
Content-type: application/xml; charset=utf-8
PI-ResourceId: UEkBbgR78VqKK/uvuq900r9bqXyBH9Ur

[xml 1]
--simple boundary
Content-type: application/xml; charset=utf-8
PI-ResourceId: dTkENva+UQ6gnwMrojr1NLxC/30t32bd

[xml 2]
--simple boundary--
```

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 SEGUINTE

Após a resposta ao “GET (...) /stream/start”, o PSP deve gravar a(s) mensagem(s) recebida(s) 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”. O cabeçalho “Accept” também admite os valores “multipart/mixed” se o PSP desejar receber mais de uma mensagem em uma mesma resposta.

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 imprevisível, 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:16422/api/v1/out/11111111/stream/start"
```

```
ENQUANTO (em operação)
  R = HTTP.GET(URL)
  URL = R.Cabeçalhos("PI-Pull-Next")
  SE (R.Codigo = 200) ENTÃO GravaMensagens(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/uvuq900r9bqXyBH9Ur

[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/30t32bd  
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.

No caso de leituras que admitam mais de uma mensagem na resposta (*multipart*), a API responde ao PSP assim que houver ao menos uma mensagem disponível (200) ou após o tempo máximo de espera (204). A API não aguarda, portanto, por acúmulo de

mensagens antes da resposta. Dessa forma, caso houver baixo volume de mensagens, uma parte das respostas com conteúdo carregará apenas uma mensagem, mesmo em formato *multipart*.

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, a sua identificação (ResourceId) será a mesma e o PSP pode utilizá-la 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.

2.2.2.10. OTIMIZANDO O PROCESSO DE LEITURA EM CENÁRIOS DE CARGA – RESUMO GERAL

- É altamente recomendável desacoplar o recebimento das mensagens do seu processamento (idealmente assíncrono).
- Faça uso de múltiplas threads de consumo (atualmente limitadas a 6 por participante por canal).
- Se o limite de *threads* for ultrapassado, o participante recebe erro 429, o que representa desperdício de recursos. Se isso estiver acontecendo de forma frequente, diminua o número de threads.
- Garanta que cada *stream* seja finalizado corretamente com DELETE, para evitar mensagens duplicadas.
- Certifique-se que esteja aceitando compactação (*Accept-Encoding: gzip*).
- Habilite o uso de *multipart* (*Accept: multipart/mixed*). Em conjunto com a compactação isso traz um bom ganho (a requisição inteira é compactada). Mesmo sem considerar essa compactação, em cenário de alta latência o uso de *multipart* ajuda muito.

2.2.3. Catálogo: versões das mensagens aceitas e enviadas pela API

As versões atuais das mensagens da API podem ser consultadas por dois *endpoints* "GET", um para as mensagens de entrada e outro para as mensagens de saída. Em ambos o retorno será um XML.

2.2.3.1. VERSÕES DAS MENSAGENS ACEITAS NA ENTRADA

Versões atuais das mensagens que são aceitas pela API na entrada.

```
GET /api/v1/in/catalog
```

2.2.3.2. VERSÕES DAS MENSAGENS ENVIADAS NA SAÍDA

Versões atuais das mensagens que são enviadas pela API aos participantes na saída.

```
GET /api/v1/out/catalog
```

2.2.3.3. EXEMPLOS

Exemplos de consultas do catálogo (com cabeçalhos omitidos):

```
-->
GET /api/v1/in/catalog HTTP/1.1
<---
HTTP/1.1 200 OK
Content-Type: application/xml; charset=utf-8
---
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Catalog>
  <Message>camt.060.spi.1.1</Message>
  <Message>camt.060.spi.1.2</Message>
  <Message>pacs.002.spi.1.3</Message>
  <Message>pacs.004.spi.1.2</Message>
  <Message>pacs.008.spi.1.4</Message>
  <Message>pibr.001.spi.1.1</Message>
</Catalog>
```

```
-->
GET /api/v1/out/catalog HTTP/1.1
<---
HTTP/1.1 200 OK
Content-Type: application/xml; charset=utf-8
---
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Catalog>
  <Message>admi.002.spi.1.3</Message>
  <Message>camt.014.spi.1.0</Message>
```

```
<Message>camt.052.spi.1.1</Message>  
<Message>camt.053.spi.1.1</Message>  
<Message>camt.054.spi.1.2</Message>  
<Message>pacs.002.spi.1.3</Message>  
<Message>pacs.004.spi.1.2</Message>  
<Message>pacs.008.spi.1.3</Message>  
<Message>pibr.002.spi.1.1</Message>  
<Message>reda.016.spi.1.1</Message>  
<Message>reda.017.spi.1.0</Message>  
</Catalog>
```

3. DICT – Diretório de Identificadores de Contas Transacionais

3.1. Interface

A interface do DICT proverá todas as funcionalidades para manutenção e obtenção de dados da conta transacional vinculada a uma chave. Será uma interface RESTful, síncrona, via HTTPS. As definições e requisitos de autenticação dessa interface estão alinhados aos da ICOM. O uso do cabeçalho Host também deve seguir o definido na seção 2.1.6. O corpo das requisições e das respostas, quando necessário, será enviado em formato XML.

3.2. Segurança

Todas as requisições feitas ao DICT deverão ser assinadas digitalmente pelos participantes, exceto as consultas a chaves. Todas repostas do DICT serão assinadas, sem exceções.

Os detalhes do processo de assinatura do DICT estão descritos no Manual de Segurança.

3.3. 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”, ou através do arquivo de definição do OpenAPI disponível em <https://github.com/bacen/pix-dict-api>.

3.4. 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. 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.

4.ARQ – Serviço de Transferência de Arquivos

4.1. Interface

A API de transferência é usada pelos PSPs para buscar arquivos gerados pelo SPI, DICT e outros sistemas que forneçam arquivos para os participantes.

As definições e requisitos de autenticação dessa interface estão alinhados aos da ICOM. O uso do cabeçalho Host também deve seguir o definido na seção 2.1.6.

Os arquivos disponibilizados no ARQ ficam disponíveis por 24 horas após sua solicitação e criação. Passado esse período, o arquivo é automaticamente removido. Caso necessário, poderá ser feita nova solicitação com envio de nova mensagem ao SPI.

4.2. API

A URL base da API é:

- <https://arq.pi.rsfn.net.br:1130> para o ambiente de **produção**; e
- <https://arq-h.pi.rsfn.net.br:1130> para o ambiente de **homologação**.

A busca de arquivos é feita através de requisições GET a caminhos especificados na documentação do sistema de origem.

A API admite o uso de cabeçalhos HTTP “Range” para retomada de downloads interrompidos.

Os arquivos entregues seguirão com os cabeçalhos “ETag” e “Last-Modified”. A API suporta os cabeçalhos “If-None-Match” e “If-Modified-Since” para detectar mudanças em arquivos obtidos previamente, possibilitando que se obtenha a nova versão apenas se houver alguma atualização.

Os erros retornados obedecem ao padrão descrito no [RFC 7807](#), alinhado com a seção 2.1.1 deste documento.

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".
28/07/2020	1.3	Retiradas referências ao documento de especificações e alterado o título e nome do arquivo. DICT: Modificada a sessão do DICT para referenciar assinatura de mensagens. ICOM: - Remoção de "Transfer-Encoding: gzip, chunked" para envio compactado. Recomendação de uso de "Content-Encoding" para essa finalidade. - Admitidos novos cabeçalhos.

		- Adoção de envio e recebimento de múltiplas mensagens em uma mesma requisição (multipart). - Incluído o endpoint Catálogo: versões das mensagens aceitas e enviadas pela API - Remoção da seção "Pendências" ARQ: incluída documentação sobre o serviço de transferência de arquivos.
12/08/2020	1.4	- Alterado padrão visual - Adicionada seção com dicas para melhoria de desempenho na leitura de mensagens em situação de carga - Alteração nas portas HTTPS para os serviços ICOM e ARQ.
19/08/2020	1.5	Correção na documentação de PI-ResourceID nos casos de requisições <i>multipart</i> no <i>endpoint</i> de saída.
06/10/2020	1.6	- API ARQ: informação sobre uso de cabeçalhos ETag, "ETag", "Last-Modified", "If-None-Match" e "If-Modified-Since". - Cabeçalho Host deverá ser enviado sem a porta (ICOM, DICT e ARQ).
14/10/2020	1.7	Criação da seção 2.2.1.5 -Limitação da Quantidade Máxima de Transações Agendadas por Unidade de Tempo. Ajuste pontual na definição do PI-Resource-ID, na seção 2.2.1.3.
30/04/2021	1.8	Criação da seção 2.2.1.6 – Limitação de Tráfego de Mensagens e Operações.
04/01/2021	1.9	Alteração nos valores da seção 2.2.1.5 – Limitação da quantidade máxima de transações agendadas por unidade de tempo.
17/10/2022	1.10	Inclusão do texto na seção 2.2.1.6. para alteração dos parâmetros de balde dos participantes, a critério do BACEN.

31/03/2023	1.11	- Alteração nos valores do bucket e do acréscimo de tokens na seção 2.2.1.6. – com validade a partir de 02/05/2023 - Criação da seção 1.2 - Canais de transmissão de mensagens. – com validade a partir de 29/10/2023 - Alterações nos textos das seções 2.2, 2.2.1.5, 2.2.1.6, 2.2.2.10 para inclusão das especificações relativas ao canal secundário. – com validade a partir de 29/10/2023
23/12/2025	1.12	- Remove texto desatualizado na seção Apresentação - Remove menções as datas passadas de implantação do canal secundário nas seções 1.2, 2.2 e 2.2.1.5 - Remove todo o item desatualizado “Limitação da quantidade máxima de transações agendadas por unidade de tempo” - Adiciona bucket para trck.002 - Adiciona informação sobre o tempo de exclusão dos arquivos no ARQ