OPEN BANKING - GT DA CONVENÇÃO - INTERFACES

ALTERAÇÕES ACORDADAS NO GT INTERFACES PARA A PROPOSTA FEBRABAN

Apresentação

Este documento é um *draft* criado com o objetivo de consolidar as alterações propostas nas reuniões do GT da Convenção – Interfaces.

O texto original da proposta Febraban, contido no repositório do *GitHub* encontra-se apenas parcialmente reproduzido aqui, essencialmente nos pontos que levantaram questionamentos.

Será utilizada a notação [...] para indicar a existência de texto não reproduzido neste documento porém existente no repositório da documentação(*slate*), no endereço eletrônico abaixo:

https://febraban.github.io/Open-Banking-/#introducao

Para facilitar, os itens alterados possuem um *(asterisco) em seu título, facilmente localizáveis no sumário.

Adicionalmente, este material contém o anexo "Anexo: Guia de Versionamento", com uma proposta padronizada de itens que caracterizariam a quebra de contrato em APIs, acordada neste GT.

As alterações propostas em que houve acordo entre os participantes do GT Interfaces estão em destaque.

Alterações em que não se obteve acordo ou maioria qualificada estão em caixas com a formatação a seguir e portanto deverão ser deliberadas no conselho:

PONTO PARA DELIBERAÇÃO NO CONSELHO – VOTAÇÃO NO GT INTERFACES:

X VOTOS PARA OPÇÃO ABCD

Y VOTOS PARA OPÇÃO XPTO

O grupo deverá revisar as alterações contidas neste documento.

Considerações adicionais e correções, além de abordadas em reunião, poderão ser enviadas para o seguinte grupo de destinatários:

gislaine_almeida@banrisul.com.br

cristiano.leao@citi.com

flavio.ricardo-alves@itau-unibanco.com.br

ricardo-augusto.santos@btgpactual.com

thiagoribas@bb.com.br

Sumário

1 Introdução*

Este documento é o resultado do trabalho conjunto do GT da Convenção – Interfaces, formado por integrantes da ABBC, ABCD, Abecs, ABFintechs, Abipag, Abranet, Camara e-net, OCB, BACEN e Febraban, baseado na proposta inicial apresentada pela Febraban (Federação Brasileira de Bancos) para implementação do Sistema Financeiro Aberto no Brasil (Open Banking). A documentação a seguir visa estabelecer padrões de desenvolvimento de APIs por parte das instituições financeiras, instituições de pagamento e demais instituições autorizadas pelo Banco Central do Brasil, além de conter informações adicionais para melhor entendimento do uso dos dados disponibilizados.

Nossa proposta tem como base os modelos do Open Banking do Reino Unido e Austrália, com as devidas adaptações ao cenário bancário brasileiro.

2 Padrões

Estes padrões representam a versão 1.0.0, a qual fornece uma visão alto nível dos padrões. Consulte a seção versionamento para obter mais informações sobre como as versões são gerenciadas com o padrão.

Observe que, nesta proposta, as palavras-chave DEVEM, NÃO DEVEM, NECESSÁRIAS, RECOMENDADO, PODE e OPCIONAL, devem ser interpretadas conforme descrito na RFC2119.

2.1 Princípios*

Os seguintes princípios técnicos não exaustivos constituem a base para o desenvolvimento e implementação das APIs para o Open Banking no Brasil.

2.1.1 Princípio 1: Segurança*

A adoção de mecanismos de segurança no design e implementação das APIs do Open Banking no Brasil deverá considerar os padrões aplicáveis a cada uma de suas fases, visando a proteção e a disponibilidade do ecossistema como um todo, considerando clientes, participantes e os dados específicos compartilhados em cada fase.

2.1.2 Princípio 2: RESTful APIs

A API irá aderir aos conceitos de *RESTful* API sempre que for possível e sensato.

2.1.3 Princípio 3: Padrões existentes*

Os padrões existentes serão adotados sempre que sua aplicação for relevante/apropriada e desde que não violem nenhum dos demais princípios, com foco na experiência do desenvolvedor e do usuário, e ainda, prevendo a extensibilidade, resiliência e a evolução do Open Banking no Brasil.

2.1.4 Princípio 4: ISO 20022*

Os *payloads* das APIs serão desenvolvidos utilizando como base os elementos e componentes de mensagem ISO 20022, que poderão ser modificados, caso necessário, para deixar o *payload* mais simples e/ou atender às características locais, tal como implementado em diferentes jurisdições.

2.1.5 Princípio 5: Extensibilidade

Os fluxos das APIs serão estendidos para atender a casos de uso mais complexos em futuros releases, e, portanto, esse princípio será mantido em mente durante o design, e os procedimentos serão detalhados durante a implementação.

2.1.6 Princípio 6: Códigos de Status

A API usará dois códigos de status que atendem a dois propósitos diferentes: (i) o HTTP *status code* reflete o resultado da chamada da API e (ii) um campo status em alguns *resource payloads* reflete o status dos *resources* nos casos de acesso *write* (p.ex. iniciação de pagamento).

2.1.7 Princípio 7: Identificadores únicos*

Um recurso REST deverá ter um identificador exclusivo que possa ser usado para identificar o recurso, com formato e padrão a ser definido a partir da Fase 2 do Open Banking no Brasil. Esses identificadores exclusivos são usados para criar URLs para identificar e endereçar recursos específicos.

2.1.8 Princípio 8: Categorização dos requisitos de implementação

Quando um requisito estiver sendo implementado por um transmissor e/ou um receptor, uma categorização diferente será aplicada. As funcionalidades, *endpoints* e campos em cada recurso serão categorizados como 'Obrigatório', 'Condicional' ou 'Opcional'.

2.1.9 Princípio 9: Agnósticas*

As APIs serão agnósticas à implementação onde elas poderão ser consumidas independente das tecnologias adotadas no ecossistema, porém com aderência aos princípios contidos nesta documentação.

2.2 Versionamento*

O controle de versão contemplará 4 tipos de lançamento (p.ex. *major*, *minor*, *patch* e *release candidate*) e terá prazos definidos para lançamento e implementação de novas versões *major*, bem como suporte de versões anteriores.

No link Anexo: Guia de Versionamento, integrante desta documentação, estão documentados os casos previstos em que uma nova versão de API poderá vir a quebrar o contrato estabelecido.

- ***Major***: inclui novas características da implementação, mudanças e correções a serem incorporadas, que poderão vir a quebrar o contrato.
    - P.ex. v1.0.0, v2.0.0.
- ***Minor***: pequenas mudanças nos elementos já existentes, com manutenção da compatibilidade e sem quebra de contrato.
    - P.ex. v1.1.0, v1.2.0
- ***Patch***: esclarecimentos às especificações publicadas pelo diretório, não incluem alterações funcionais.
    - P.ex. v1.1.1, v1.1.2
- ***Release candidate***: versões de pré-lançamento de qualquer patch futuro, *minor* ou *major*.
    - P.ex. v1.0.0-rc, v1.0.0-rc2

Será definido um cronograma de novas versões dos padrões para que os participantes consigam se planejar e desenvolver novas APIs, com cada um dos lançamentos tendo um prazo pré-estabelecido para ser implementado pelos participantes, mitigando, desta forma, o risco de múltiplas versões.

Não serão feitos mais do que um lançamento de versão *major* em um período de 6 meses. No entanto, serão previstas exceções para atender às alterações urgentes que não podem esperar até a próxima versão principal (*major*). Lançamentos de versões *minor* e *patch* podem ocorrer a qualquer momento.

Lançamentos *minor* não podem configurar em quebra de contrato, impactar significativamente *endpoints* e/ou exigir manutenção crítica.

Por fim, credenciais de acesso associadas às APIs deverão ser agnósticas à versão.

2.3 Estrutura da URI

A estrutura da URI para os *endpoints* deve ser implementada conforme abaixo:<host> / open-banking / <api> / <versão> / <recurso>

Os componentes desta estrutura de URI estão descritos abaixo:

- ***Host***: O *host* de API da entidade financeira implementadora é um endereço base definido pela entidade transmissora de dados.
- "***open-banking***": Esta é uma *string* constante que representa a finalidade desta API.
- **API**: A API que será consumida (p.ex. *channels*).
- **Versão**: O número da versão da API. Na URI a versão deve ser precedida pela letra "v" seguida pelo número da versão a ser consumida (p.ex. v1, v2, v25).
- **Recurso**: O recurso a ser consumido dentro de uma API. Utilizando como exemplo a API *channels*, a mesma possui 4 recursos:
  - *banking-agents*
  - *branches*
  - *electronic-channels*
  - *phone-channels*

A versão *minor* será repassada apenas no *header* do *payload* de resposta, orientando a instituição receptora sobre quais serão os dados no retorno.

Como exemplo, para realizar o consumo do método *electronic-channels* da API *channels* na versão 1, a URI ficaria com a seguinte estrutura:

<host>/open-banking/channels/v1/electronic-channels

2.4 Cabeçalhos HTTP*

Cabeçalhos HTTP suportados e suas funções.

2.4.1 Cabeçalho de requisição

Nome do cabeçalho	Descrição	Obrigatório
Content-Type	Representa o formato do payload de requisição, por padrão/default definido como application/json;charset UTF-8. Obrigatório para chamadas PUT e POST. Os transmissores poderão implementar tratamento para outros padrões, sendo obrigatório apenas o suporte ao padrão.	Não

Nome do cabeçalho	Descrição	Obrigatório
Accept	Especifica o tipo de resposta. O default/padrão definido é application/json;charset UTF-8, mínimo a ser implementado de modo obrigatório no ecossistema do Open Banking e o mesmo a ser adotado caso o cabeçalho não seja especificado. Os transmissores poderão implementar tratamento para outros padrões. Se for informado um valor não suportado pelo transmissor, será retornado o código HTTP 406.	Não
Accept-Encoding	Especifica os tipos de encoding(geralmente algoritmo de compressão) que são suportados pelo cliente, com previsão de suporte ao gzip por parte dos transmissores, sendo que o padrão é a transmissão dos dados não compactados e esta orientação aplica-se aos Dados Abertos.	Não
If-Modified-Since	Condiciona o resultado da requisição para que o recurso só seja enviado caso tenha sido atualizado após a data fornecida. Utiliza o padrão da RFC 7232, sessão 3.3: *If-Modified-Since* do protocolo HTTP.	Não

2.5 Códigos de resposta HTTP*

Os códigos de resposta HTTP devem ser utilizados conforme tabela abaixo.

Códigos

Situação	Código HTTP	Notas	POST	GET	DELETE
Consulta concluída com sucesso.	200 OK.		Sim	Sim	Não
Execução normal. A solicitação foi bem sucedida.	201 Created.	A operação resulta na criação de um novo recurso.	Sim	Não	Não
Operação de exclusão concluída com sucesso.	204 No Content.		Não	Não	Sim
A requisição foi malformada, omitindo atributos obrigatórios,	400 Bad Request.	A operação solicitada não será realizada.	Sim	Sim	Sim

Situação	Código HTTP	Notas	POST	GET	DELETE
seja no *payload* ou através de atributos na URL.					
Cabeçalho de autenticação ausente/inválido ou *token* inválido.	401 Unauthorized.	A operação foi recusada devido a um problema de autenticação.	Sim	Sim	Sim
O *token* tem escopo incorreto ou uma política de segurança foi violada.	403 Forbidden.	A operação foi recusada devido a falta de permissão para execução.	Sim	Sim	Sim
O recurso solicitado não existe ou não foi implementado.	404 Not Found.		Sim	Sim	Sim
O consumidor tentou acessar o recurso com um método não suportado.	405 Method Not Allowed.		Sim	Sim	Sim
A solicitação continha um cabeçalho Accept diferente dos tipos de mídia permitidos ou um conjunto de caracteres diferente de UTF-8.	406 Not Acceptable.		Sim	Sim	Sim
Indica que o recurso não está mais disponível.	410 Gone.		Sim	Sim	Sim
A operação foi recusada porque o payload está em um formato não suportado pelo endpoint.	415 Unsupported Media Type.		Sim	Não	Não
A solicitação foi bem formada, mas não pôde ser processada devido à lógica de negócios específica da solicitação.	422 Unprocessable Entity.	Se aplicável ao *endpoint*, espera-se que esse erro resulte em um *payload* de erro.	Sim	Sim	Não

Situação	Código HTTP	Notas	POST	GET	DELETE
A operação foi recusada, pois muitas solicitações foram feitas dentro de um determinado período ou o limite global de requisições concorrentes foi atingido.	429 Too Many Requests.	A limitação é um Requisito Não Funcional. O titular dos dados deve incluir o cabeçalho Retry-After na resposta indicando quanto tempo o consumidor deve esperar antes de tentar novamente a operação.	Sim	Sim	Sim
Ocorreu um erro no *gateway* da API ou no microsserviço.	500 Internal Server Error.	A operação falhou.	Sim	Sim	Sim
O serviço está indisponível no momento.	503 Service Unavailable.		Sim	Sim	Sim
O servidor não pôde responder em tempo hábil.	504 Gateway Timeout.	Retornado se ocorreu um tempo limite, mas um reenvio da solicitação original é viável (caso contrário, use 500 *Internal Server Error*).	Sim	Sim	Sim

2.6 Convenções de *payload*

Esta seção do padrão descreve as estruturas padrões de requisição e resposta para todos os *endpoints* das APIs, assim como as convenções de nomenclatura para os atributos.

2.6.1 Estrutura de requisição

Cada requisição deve ser um objeto JSON contendo um objeto data para armazenar os dados primários da requisição.

[...]

2.6.2 Estrutura de resposta

Cada *endpoint* retornará um objeto JSON contendo os atributos abaixo:

[...]

O objeto *links* irá conter *hypermedia* (referências para recursos relacionados) para outros recursos da API requisitada.

2.6.3 Convenções de nomenclatura de atributos

2.6.3.1 Caracteres válidos em nomes de atributos

[...]

2.6.3.2 Estilo de nomeação de atributos

[...]

2.6.4 Convenções de propriedade dos atributos

2.6.4.1 Tipos de dados dos atributos

[...]

2.6.4.2 Atributos Obrigatórios / Opcionais

[...]

2.6.4.3 Atributos vazios / nulos*

Um atributo omitido (ou seja, um atributo que não está presente no *payload*) será considerado equivalente a um atributo que esteja presente com o valor *0*.

Uma *string* vazia (“”) não será considerada equivalente a *null*.

Na situação onde o campo a ser informado no *payload* seja obrigatório e a Instituição, seja consumidora no envio ou transmissora no retorno, não a possuir, deve-se implementar o valor padronizado: “NA” - Não se Aplica.

O valor booleano false não será considerado equivalente a *null*. Os atributos booleanos opcionais, por definição, possuirão três valores possíveis: verdadeiro (*true*), falso (*false*) e indeterminado (*null*).

2.7 Convenções de nomenclatura

Todos os nomes devem ser autoexplicativos, sem redundância de termos e sem ambiguidade de entendimento, além de seguir o padrão *Lower Camel* Case (primeira letra de cada termo maiúscula, com exceção do primeiro termo, que fica todo em minúsculas e sem espaços ou pontuações entre os termos). Ex: “*areaCode*”.

[...]

2.8 Tipos de dados comuns*

2.8.1 Propriedades

Tipo	Descrição	Exemplos válidos
AmountString	- Uma *string* que representa um valor monetário. - Um número positivo, zero ou negativo. - Sem o símbolo da moeda. - Com pelo menos 1 e no máximo 16 dígitos antes do ponto decimal. - Com no mínimo 2 dígitos (mais dígitos são permitidos, porém não obrigatórios). - Sem formatação adicional. Ex: Separador de milhar.	"1.37" "54.85" "3456928.98" "-2387.02"
Boolean	- Valor booleano padrão.	true false
CurrencyString	- Uma *string* que representa a abreviação da moeda conforme especificação ISO-4217.	"BRL" "USD" "EUR"
DateTimeString	- Uma *string* com data e hora conforme especificação RFC-3339.	"2020-07-21T08:30:00-03:00"
DurationString	- Uma *string* que representa um período de duração conforme especificação ISO-8601.	"P23DT23H" "PT2H30M"
Enum	- Uma *string* que representa um domínio de valores - Todos os possíveis valores são definidos. - Os valores devem estar em letras maiúsculas. - Espaços em branco devem ser substituídos por _. - Artigos e preposições devem ser removidos. - Não devem possuir caracteres acentuados.	"PRIMEIRA_OPCAO" "OUTRA_OPCAO_EXISTENTE"
Integer	- Números inteiros.	-1 0 1
RateString	- Uma string que representa um valor percentual, tendo como referência que 100% é igual ao valor 1. - Com pelo menos 1 e no máximo 16 dígitos antes do ponto decimal. - Com no máximo 16 dígitos após o ponto decimal. - Sem formatação adicional. Ex: Separador de milhar.	"0.01" "0.1" "-0.05" "-0.98365"
String	- Padrão de texto UTF-8 sem restrição de conteúdo.	"Uma *string* qualquer."
TimeString	- Uma *string* que representa a hora conforme especificação RFC-3339.	"12:00:00+00:00" "09:00:00+03:00"
URIString	- Uma *string* que representa URI válida.	"http://www.google.com.br"

PONTO PARA DELIBERAÇÃO NO CONSELHO – VOTAÇÃO NO GT INTERFACES – PADRÃO PARA DATETIMESTRING E TIMESTRING:

4 VOTOS PARA **UTC TIME FORMAT** (HH:MM:SS.SSSZ) – UTC 0

1996-12-20T00:39:57Z

2 VOTOS PARA **LOCAL TIME WITH UTC OFFSET FORMAT** (HH:MM:SS.SSS+/-HH:MM)

1996-12-19T16:39:57-08:00

2.9 Paginação

Cada recurso de cada API pode possuir ou não paginação, caso a quantidade de registros retornados justifique a mesma. A paginação estará disponível e deverá funcionar independente se o recurso permite filtros por query ou POST. Isso é, filtros e paginação são aplicados de forma independente.

2.9.1 Parâmetros de Requisição

[...]

2.9.2 Atributos de Resposta

2.9.3 Links*

O objeto *links* passará por revisão subsequente de modo a atender as próximas Fases do Open Banking, em especial a partir da Fase 2.

No objeto links, serão retornadas *hypermedia* (referências para os recursos relacionados) de paginação conforme parâmetros abaixo:

[...]

2.10 Estabilidade do ID*

Dentro desses padrões, a serem melhor especificados a partir da Fase 2 do Open Banking no Brasil, os IDs de recursos são necessários para atender ao seguinte:

- O ID de um recurso deve ser especificado no *endpoint* de uma API apenas para obter detalhes do recurso ou para realizar alterações no mesmo.
- Se o ID for especificado nos padrões do Open Banking, então ele é obrigatório e deverá ser fornecido pela entidade implementadora da API de acordo com o padrão definido.
- Se um ID for especificado, o mesmo deverá ser totalmente desconectado de significados com outras entidades. Por exemplo, um ID não deve ser uma combinação de outros campos ou uma *string* que possa ter conteúdo sensível que possa ser extraído.
- Os IDs devem ser únicos, e sua padronização será detalhada a partir da Fase 2 do Open Banking no Brasil, porém sua unicidade pode estar dentro de um contexto. Por exemplo, um ID de conta corrente deve ser único, porém apenas dentro do contexto de conta corrente.

- Nos *payloads* o nome de campo "id" nunca deverá ser utilizado. Cada campo ID deverá ter um nome significativo, dessa forma independentemente de onde o ID for utilizado entre múltiplos *endpoints*, ele sempre irá se referir ao seu objeto principal. Por exemplo, IDs para conta deverão ser representados no JSON como "*accountId*".

2.11 Extensibilidade

Os padrões de Open Banking podem não cobrir todas as possibilidades de objetos retornados ou APIs que os participantes desejam expor. Os participantes podem ter o desejo de realizar inovações sobre os padrões definidos oferecendo mais dados afim de atender demandas específicas de mercado. É nossa intenção que os padrões definidos não apenas permitam estas extensões como também sirvam como base para futuras alterações na própria definição dos padrões.

No entanto, é importante que um participante que esteja querendo estender as APIs não impeça um consumidor que foi projetado para consumir apenas o *endpoint* padrão funcione corretamente.
Para atender tanto as demandas de quem deseja estender as APIs (participantes) quanto as demandas de quem irá realizar o consumo (consumidor das APIs), foram definidos os critérios abaixo.
É possível estender os padrões nos seguintes aspectos:

- O participante pode oferecer **uma API** completamente nova que não está coberta nos padrões definidos
- O participante pode oferecer **novos *endpoints*** em uma API que já foi definida no padrão
- O participante pode oferecer **campos de entrada e retorno opcionais** para um *endpoint* que já foi definido no padrão

2.11.1ID dos participantes*

Participantes que desejam estender os padrões devem adicionar seu prefixo para identificar todas as extensões. Campos adicionais no retorno de *endpoints* existentes ou em novos *endpoints* devem usar o prefixo do participante. O prefixo deve ser no formato exposto ao lado (4 letras) e não devem haver prefixos duplicados entre os participantes.
Nesta documentação, quando tivermos que nos referir ao prefixo do participante, o termo <PID> será utilizado.
[...]

2.11.2Novas APIs

Quando a extensão for a criação de uma nova API, o participante deve adicionar seu prefixo a URI antes do nome da nova API, conforme exemplo abaixo.

[...]

2.11.3Novos *endpoints* em APIs existentes

Quando o participante desejar adicionar um novo *endpoint* em uma API já especificada no padrão, o participante deve incluir seu <PID> como prefixo do recurso que será implementado.

[...]

2.11.4Campos de retorno adicionais em um *endpoint* existente

[...]

2.11.5Parâmetros query adicionais

Quando for adicionado um novo parâmetro de *query* a um *endpoint* existente, o novo parâmetro deve ter o prefixo <PID>-, evitando assim colisões com parâmetros já existentes.

[...]

2.11.6Filtro de Dados

Opcionalmente, a entidade transmissora de dados poderá realizar filtro de dados através de *query* de entrada, baseado em campos que julgue relevante para a melhor experiência do cliente.

A informação de quais possibilidades estarão disponíveis (*query parameter*) deverá constar em documentação adicional disponibilizada pela entidade transmissora.

2.11.7Extensão do versionamento

Como descrito na seção versionamento, o versionamento existe apenas no nível das APIs e não no nível dos *endpoints*, no entanto caso seja necessário realizar versionamento de um *endpoint* customizado, o participante poderá utilizar o header x-<PID>-v para que o consumidor possa especificar qual versão do *endpoint* está requisitando.

3 Glossário

[...]

4 Segurança - Consumo

Visão geral

As APIs de Open Banking estão divididas em dois escopos:

- open-data

- customer-data

[...]

[AJUSTES DO GT DA CONVENÇÃO - SEGURANÇA]

5 Segurança – Participantes

[...]

[AJUSTES DO GT DA CONVENÇÃO - SEGURANÇA]

6 APIs Comuns

6.1 API de Status*

GET /discovery/status

Obtém a descrição referente ao código de *status* retornado pelas APIs.

[...]

O comando acima retorna uma estrutura *json* como exemplificada abaixo, e no caso em que o *status* devolvido seja *PARTIAL_FAILURE*, o *array unavailableEndpoints* deve conter a lista de *endpoints* indisponíveis:

[...]


```
{
  "data": {
    "status": "OK",
    "explanation": "Retorno com Sucesso",
    "detectionTime": "2020-01-01T01:00:00+00:00",
    "expectedResolutionTime": "2020-01-01T01:00:00+00:00",
    "updateTime": "2020-01-02T01:00:00+00:00",
    "unavailableEndpoints": [],
  },
  "links": {
    "self": "https://api.banco.com.br/open-banking/discovery/v1/status"
  },
  "meta": {
    "totalRecords": 1,
    "totalPages": 1
  }
}
```

6.2 API de Outages*

GET /discovery/outages

Obtêm a lista de indisponibilidade agendada para os serviços.

Na estrutura de retorno exemplificada abaixo, no caso em que o parâmetro *isPartial* devolvido seja *true*, o *array unavailableEndpoints* deve conter a lista de *endpoints* indisponíveis:

[...]

```
{
  "data": {
    "outages": [
      {
        "outageTime": "2020-01-01T01:00:00+00:00",
        "duration": "PT2H30M",
        "isPartial": false,
        "explanation": "Atualização do API Gateway"
        "unavailableEndpoints": [],
      }
    ]
  },
  "links": {
    "self": "https://api.banco.com.br/open-banking/discovery/v1/outages"
  },
  "meta": {
    "totalRecords": 1,
    "totalPages": 1
  }
}
```

6.3 Especificação em OAS 3.0

7 API - Canais de Atendimento

7.1 Dependências próprias

GET /channels/<versão>/branches
[...]

7.2 Canais de atendimento eletrônico

GET /channels/<versão>/electronic-channels
[...]

7.3 Canais de atendimento telefônico

GET /channels/<versão>/phone-channels
[...]

7.4 Correspondentes bancários

GET /channels/<versão>/banking-agents
[...]

7.5 Especificação em OAS 3.0

[...]

8 API - Produtos e Serviços

8.1 Contas pessoa física

GET /products-services/<versão>/personal-accounts
[...]

8.2 Contas pessoa jurídica

GET /products-services/<versão>/business-accounts
[...]

8.3 Empréstimos pessoa física

GET /products-services/<versão>/personal-loans
[...]

8.4 Empréstimos pessoa jurídica

GET /products-services/<versão>/business-loans
[...]

8.5 Financiamento Pessoa Física

GET /products-services/<versão>/personal-financings
[...]

8.6 Financiamento Pessoa Jurídica

GET /products-services/<versão>/business-financings

[...]

8.7 Antecipação de recebíveis Pessoa Física

GET /products-services/<versão>/personal-invoice-financings

[...]

8.8 Antecipação de recebíveis Pessoa Jurídica

GET /products-services/<versão>/business-invoice-financings

[...]

8.9 Cartão de crédito de pessoa física

GET /products-services/<versão>/personal-credit-cards

[...]

8.10 Cartão de crédito de pessoa jurídica

GET /products-services/<versão>/business-credit-cards

[...]

8.11 Especificação em OAS 3.0

9 API – Admin*

As APIs administrativas são recursos que podem ser consumidos apenas pelo diretório para avaliação e controle da qualidade dos serviços fornecidos pelas instituições financeiras.

9.1 Métricas

GET /admin/<versão>/metrics

[...]

9.1.1 Visão Geral

Este *endpoint* possibilita ao diretório consultar estatísticas operacionais das APIs disponibilizadas pelas instituições financeiras, a fim de avaliar a qualidade dos serviços fornecidos ao usuário final.

[...]

PROPOSTA DE ALTERAÇÃO ACORDADA NO GT INTERFACES – ALTERAÇÃO NO PAYLOAD DE RESPOSTA:

Conforme deliberado em reunião, a API de métricas será de consumo do diretório com retornos por API.

Está mantido o agrupamento dos *HTTP STATUS CODES*.

10 Schemas

10.1 ResponseBranchesList

[...]

10.2 BranchesBrand

[...]

10.3 BranchesCompanies

[...]

10.4 Branch

[...]

10.5 BranchIdentification

[...]

10.6 BranchPostalAddress

[...]

10.7 BranchAvailability*

Nome	Tipo	Obrigatório	Descrição
standard	Array	Sim	Lista com os dias da semana.
» weekday	string	Sim	Dia da semana.

Nome	Tipo	Obrigatório	Descrição
» openingTime	[UTCHour]	Sim	Horário de abertura na dependência.
» closingTime	[UTCHour]	Sim	Horário de encerramento na dependência.

PONTO DE ATENÇÃO!

Ajustar *openingTime* e *closingTime* para que usem o tipo *TimeString* da seção dos dados comuns após deliberação do conselho.

11 Requisitos não funcionais

11.1 Nível de serviço (SLA)

O suporte eficaz da disponibilidade do Open Banking mantém níveis consistentes de serviços do sistema. Níveis de Serviço definidos:

- Cada *endpoint* da API deve estar disponível 95% do tempo durante cada período de 24 horas.
- Cada *endpoint* da API deve estar disponível 99.5% do tempo durante cada período de 3 meses.
- Cada *endpoint* da API deve retornar o primeiro byte de resposta dentro de 1000ms por 95% das requisições.
- O tempo de resposta será medido por um cliente externo com uma latência de rede máxima de 50 ms (tempo para o primeiro byte).

Informativamente, esse nível de serviço representa aproximadamente um *downtime* máximo de 0,5% por trimestre, o que corresponde a 18s por hora, 7,2min (432s) por dia, 3,6h (216m) por mês e 10,8h (648m) por trimestre.

A definição de um período de indisponibilidade é qualquer período de tempo em que qualquer um dos *endpoints* da API definidos na norma é incapaz de fornecer uma resposta confiável a uma solicitação construída de forma apropriada.

11.1.1Checagem de disponibilidade:*

A disponibilidade é checada no *endpoint* GET /discovery/status, conforme documentada no item API de Status.

A cada 30 segundos, a API de status é requisitada com *timeout* de 1s.

- Será considerado *uptime*, se o retorno for:
  - OK.

- Será considerado *downtime*, se o retorno for:
  - *PARTIAL_FAILURE;*
  - *SCHEDULED_OUTAGE:*
    - Se a requisição for realizada entre o período de 01h e 07h, o contador de *SCHEDULED_OUTAGE* é iniciado com 30 segundos acrescidos;
    - Cada nova requisição vai adicionando 30 segundos mais ao contador de *SCHEDULED_OUTAGE*, até que uma requisição volte outro valor ou a requisição for feita depois das 07h.
  - *UNAVAILABLE*:
    - Se a requisição for realizada entre o período de 07h e 01h;
    - Se serviço não responder a requisição;
    - O contador de *downtime* é iniciado com 30 segundos acrescidos;
    - Cada nova requisição adicionará 30 segundos a mais ao contador de *downtime*, até que uma requisição retorne OK.

O *downtime* deve ser calculado como o número total de segundos simultâneos por requisição da API, por período de 24 horas, começando e terminando à meia-noite, que qualquer *endpoint* da API não esteja disponível, dividido por 86.400 (total de segundos em 24 horas) e expresso como uma porcentagem.

A disponibilidade é calculada sendo 100% menos a quantidade em percentual da indisponibilidade.

Não será considerado como *downtime*:

- Uma indisponibilidade por mês, por 3h entre 01h e 07h, desde que reportado com uma semana de antecedência ao diretório;
- Por tempo não definido, a qualquer momento e sem notificação em caso de resolução de problemas de segurança, desde que aprovado pelo Diretório. Neste caso, as instituições devem garantir o emprego dos melhores esforços para a resolução do problema.

11.2 Nível de desempenho

O desempenho do *endpoint* da API será medido no tempo de resposta de cada solicitação, desde o recebimento da solicitação até a entrega da resposta.

Espera-se que o detentor dos dados garanta que a medição do tempo de resposta ocorra o mais próximo possível do receptor dos dados, embora algumas camadas técnicas não estejam no controle do detentor dos dados.

À luz destas considerações, a exigência de desempenho para os detentores dos dados é:

- APIs de alta prioridade (*status/outages*) devem manter percentil 95 em no máximo 1000ms.
- APIs de média prioridade (*channels/products-services*) devem manter percentil 95 em no máximo 1500ms.

- APIs Admin (ex. *metrics*) devem manter percentil 95 em no máximo 4000ms.

P. ex. Em um dia que a API Produtos e Serviços receba 10.000 chamadas, pelo menos 9.500 delas deveriam ter sido respondidas dentro de um prazo inferior a 1500ms.

11.3 Limites de tráfego de requisições*

Os limites de tráfego serão estabelecidos utilizando as seguintes métricas:

- Transações por Segundo (TPS) - o número de transações simultâneas a cada segundo;
- Número de chamadas - o número de chamadas de *endpoint* iniciadas por uma duração especificada.

Cada instituição transmissora, deverá garantir os seguintes limites mínimos de tráfego abaixo especificados para as APIs de Dados Públicos – Fase 1, os quais serão revisados e ajustados em decorrência dos indicadores de uso das APIs, com revisão prevista imediatamente antes da entrada da Fase 2:

- 500 Requisições por minuto por receptora (via endereço IP);
- 300 TPS globalmente.

As chamadas que excedam os seguintes limites de tráfego poderão ser enfileiradas ou rejeitadas por um detentor de dados sem impacto em seu desempenho ou requisitos de disponibilidade.

Requisições que ultrapassem os limites estabelecidos poderão ser rejeitadas utilizando o HTTP *status code*: 429 *Too Many Requests*.

12 Guia Operacional

O Guia Operacional dá suporte a doadores e receptores, trazendo informações sobre Desempenho, Disponibilidade, Processo de Registro e Revogação, Estrutura do Repositório de Informações, Resolução de Problemas, Comunicação de Mudanças, *CheckList*, Glossário e Referências.

Ele poderá ser acessado clicando aqui.

13 Anexo: Guia de Versionamento*

Este anexo tem como objetivo detalhar quando mudanças nas APIs do Open Banking serão consideradas disruptivas (*breaking changes*) exigindo a criação de uma nova versão maior (major) e quando uma mudança poderá ser tratada como não disruptiva (*non breaking changes*) podendo ser criada uma versão menor (*minor*) para comportá-la.

Mudanças em APIs REST podem ocorrer no contrato da API (Open API 3.0) tendo efeitos mais visíveis aos consumidores e, portanto, sendo mais fácil de se mapear os seus impactos; ou podem acontecer em suas regras de negócio/implementação precisando de uma análise mais complexa de quando a mudança traz impactos aos atuais consumidores das APIs.

Nota: Os exemplos contidos neste anexo são meramente ilustrativos e não refletem as APIs definidas nas especificações do Open Banking.

13.1 Mudanças no Contrato

Por se basear nos mecanismos de transmissão de mensagem HTTP, uma API REST é composta de vários elementos que podem ser alterados causando impactos aos seus consumidores. Por via de regra, alterações de APIs que exigem que os consumidores alterem os seus sistemas para poder continuar utilizando a funcionalidade devem necessariamente ser feitas com um versionamento do tipo *major*. Mudanças que não necessitem desenvolvimento por parte dos consumidores poderão ser tratadas como *minor*.

No *request* de uma API, são pontos de alteração relevantes: verbo, *schema*, recurso, *pathParams*, *queryParams*, *headers* e o formato do *body*. No *response*, as alterações podem ocorrer no *status code*, *headers* e no formato do *body*. Sendo que dependendo da API nem todos os elementos estão presentes.

Existe ainda o cenário em que APIs definem *callbacks* (*webhooks*). Neste caso alterações em path*h*Params, queryParams, *headers* e *body*, além do próprio uso do *webhook* podem impactar o consumidor.

Estrutura de um *request*

```
VERBO schema://host:porta/open-banking/api/versão/recurso/pathParam?queryParam=ipsum
Header: ipsum loren

{JSON BODY}
```

Estrutura de um *response*

```
HTTP Status
Header: ipsum loren

{JSON BODY}
```

Estrutura de um *callback*

```
VERBO schema://callbackUri/recurso/pathParam?queryParam=ipsum
Header: ipsum loren

{JSON BODY}
```

Exemplo:

```
GET https://api.banco.com.br/open-banking/customers/v1/customers/12345678900/accounts?type=CONTA_CORRENTE
Authorization: Bearer token
```

```
200 Success
Content-Type: application/json

{
  "data": {
    "accounts": [
      "branch": "0001",
      "account": "123456"
    ]
  }
  "meta": {
    "totalRecords": 1,
    "totalPages": 1
  }
}
```

API que lista as contas do cliente de id “12345678900” filtrando apenas as contas do tipo “CONTA_CORRENTE”. Neste exemplo o verbo utilizado foi o *GET*, o *schema* o HTTPS, o caminho a API é a “*customers*”, a versão “v1”, o recurso “*customers*”, existe um *pathParam* que identifica o cliente, um *queryParam* que filtra os tipos de contas retornados, um *header* informando o *token* de autorização e a requisição não possui um *body*.

A resposta desta API contém um *status* de sucesso, um *header* identificando o tipo de conteúdo retornado e um *body* contendo a lista de contas do cliente.

13.1.1Lista de Mudanças disruptivas:

BC1. Remover um recurso;

BC2. Remover um verbo de um recurso (operação);

BC3. Alterar o verbo de um recurso (operação);

BC4. Remover um *path*;

BC5. Remover um parâmetro do *request* (*pathParam* e/ou *queryParam*);

BC6. Renomear um parâmetro (*queryParam* e/ou *body*)

BC7. Adicionar um parâmetro obrigatório no *request* (*pathParam*, *queryParam* e/ou *body*)

BC8. Alterar local onde um parâmetro é recebido

BC9. Alterar um parâmetro do tipo *enum* (adicionar ou remover valores)

BC10. Remover o suporte a um tipo de conteúdo (*content-type*) previamente aceito no *request*;

BC11. Remover o suporte a um tipo de conteúdo (*content-type*) previamente aceito no *response*;

BC12. Adicionar um *header* obrigatório no *request*;

BC13. Remover um *header* do *response*;

BC14. Alterar a estrutura do *body* (*request*, *response* e/ou *webkook*);

BC15. Alterar o tipo de um parâmetro (*pathParam*, *queryParam*, *header* e/ou *body*);

BC16. Alterar o formato de um parâmetro (*pathParam*, *queryParam*, header e/ou *body*);

BC17. Aumentar as restrições de um parâmetro no *request* (*pathParam*, *queryParam*, *header* e/ou *body*);

BC18. Diminuir as restrições de um parâmetro no *response* (*body*);

BC19. Alterar o valor padrão de um parâmetro opcional;

BC20. Alterar a forma de representar *arrays* nos parâmetros (*queryParam*)

BC21. Adicionar um novo código HTTP de resposta;

BC22. Remover um código HTTP de resposta;

BC23. Modificar o código HTTP de uma resposta existente;

BC24. Adicionar/Remover o uso de *callbacks* (*webhooks*) assíncrono

13.1.2Lista Mudanças não disruptivas:

NBC1. Adicionar um novo recurso na API;

NBC2. Adicionar um novo verbo em um recurso existente (operação);

NBC3. Adicionar um novo *path*;

NBC4. Tornar um parâmetro obrigatório do *request* (*pathParam* e/ou *queryParam*) em opcional;

NBC5. Adicionar novos parâmetros opcionais no *request* (*header* ou *body*)

NBC6. Adicionar novos parâmetros no response (*header* ou *body*)

BC1. Remover um recurso

Remover um recurso é uma mudança disruptiva pois indisponibiliza um conjunto de funcionalidades utilizadas pelos consumidores atuais da API, exigindo que eles adequem o seu negócio e implementação para se adequar a ausência do recurso removido.

Exemplo: a API originalmente define três recursos e na nova versão exclui o recurso 2.

Versão 1

```
/api/v1/recurso1
/api/v1/recurso2
/api/v1/recurso3
```

Versão 2

```
/api/v2/recurso1
/api/v2/recurso3
```

BC2. Remover um verbo de um recurso (operação)

Remover um verbo de um recurso (operação) é uma mudança disruptiva pois indisponibiliza uma funcionalidade utilizada pelos consumidores atuais da API, exigindo que eles adequem o seu negócio e implementação para se adequar a ausência da operação removida.

Exemplo: a API originalmente define um recurso com três operações e na nova versão uma das operações foi removida.

Versão 1

```
GET /api/v1/recurso1
GET /api/v1/recurso1/{id}
PUT /api/v1/recurso1/{id}
```

Versão 2

```
GET /api/v2/recurso1
GET /api/v2/recurso1/{id}
PUT /api/v2/recurso1/{id}
```

BC3. Alterar o verbo de um recurso (operação)

Alterar um verbo de um recurso (operação) é uma mudança disruptiva pois exige que os consumidores atuais da API alterem a sua implementação para continuar consumindo um recurso. Geralmente, uma mudança no verbo da operação é acompanhada de uma mudança na regra de negócio.

Exemplo: a API originalmente define um recurso com o verbo PUT e na nova versão ele é alterado para PATCH.

Versão 1

```
PUT /api/v1/recurso1/{id}
```

Versão 2

```
PATCH /api/v2/recurso1/{id}
```

BC4. Remover um *path*

Em uma API um path (recurso + subrecurso) pode ser utilizado como uma forma de facilitar o acesso de consulta e/ou edição a um subrecurso. Removê-lo implica na necessidade de alteração na implementação dos consumidores que precisarão modificar a forma de acesso a informação.

Exemplo: a API originalmente define um path para acessar apenas os dados do subrecurso2 com facilidade. A nova versão remove este path.

Versão 1

```
GET /api/v1/recurso1/{id}
GET /api/v1/recurso1/{id}/subrecurso1
GET /api/v1/recurso1/{id}/subrecurso2
```

Versão 2

```
GET /api/v2/recurso1/{id}
GET /api/v2/recurso1/{id}/subrecurso1
```

BC5. Remover um parâmetro do *request* (*pathParam* e/ou *queryParam*)

Remover um parâmetro do *request* de uma API, seja ele um *pathParam* ou *queryParam*, faz com a implementação dos consumidores precise ser alterada ou até revista dado que uma condição para o funcionamento do seu negócio pode ter sido removida.

Exemplo 1: a API originalmente define um *pathParam* que identifica o recurso sendo. A nova versão não possui mais este parâmetro, não sendo mais possível identificar o recurso.

Versão 1

```
GET /api/v1/recurso1/{id}/subrecurso1
```

Versão 2

```
GET /api/v2/recurso1/subrecurso1
```

Exemplo 2: a API originalmente define como *queryParam* um filtro. A nova versão não possui mais este filtro.

Versão 1

```
GET /api/v1/recurso1?filter=lorem
```

Versão 2

```
GET /api/v2/recurso1
```

BC6. Renomear um parâmetro (queryParam e/ou body)

Renomear um parâmetro, seja ele um *queryParam* ou um parâmetro no *body*, exige que os consumidores alterem a sua implementação para continuar utilizando a API.

BC7. Adicionar um parâmetro obrigatório no *request* (*pathParam*, queryParam e/ou *body*)

Adicionar um parâmetro, seja ele um *pathParam*, *queryParam* ou um *parâmetro* no *body*, exige que os consumidores alterem a sua implementação para continuar utilizando a API.

BC8. Alterar local onde um parâmetro é recebido

Alterar o local onde um parâmetro é recebido exige que os consumidores alterem a sua implementação para continuar utilizando a API.

BC9. Alterar um parâmetro do tipo *enum* (adicionar ou remover valores)

Um *enum* define os valores válidos para um parâmetro e alterar a lista exige que os consumidores alterem a sua implementação.

BC10. Remover o suporte a um tipo de conteúdo (*content-type*) previamente aceito no request

BC11. Remover o suporte a um tipo de conteúdo (*content-type*) previamente aceito no response

BC12. Adicionar um *header* obrigatório no request;

BC13. Remover um *header* do *response*

BC14. Alterar a estrutura do *body* (*request*, *response* e/ou *webkook*);

BC15. Alterar o tipo de um parâmetro (*pathParam*, *queryParam*, *header* e/ou *body*);

BC16. Alterar o formato de um parâmetro (*pathParam*, *queryParam*, *header* e/ou *body*)

BC17. Aumentar as restrições de um parâmetro no request (*pathParam*, *queryParam*, *header* e/ou *body*)

BC18. Diminuir as restrições de um parâmetro no response (*body*);

BC19. Alterar o valor padrão de um parâmetro opcional;

BC20. Alterar a forma de representar *arrays* nos parâmetros (*queryParam*)

BC21. Adicionar um novo código HTTP de resposta;

BC22. Remover um código HTTP de resposta;

BC23. Modificar o código HTTP de uma resposta existente;

BC24. Adicionar/Remover o uso de *callbacks* (*webhooks*) assíncrono