

Boletín CIMPRA N°535

31 de enero de 2024

QR interoperables para pagos
con tarjeta de crédito / Interfaz
Estandarizada de Pagos



BANCO CENTRAL
DE LA REPÚBLICA ARGENTINA

Boletín CIMPRA N° 535

QR interoperable para pagos con tarjeta de crédito / Interfaz Estandarizada de Pagos

Introducción

Mediante el punto 10 de la Comunicación “A” 7769 (receptado en el punto 4.4. de las normas sobre “Sistema Nacional de Pagos – Servicios de pago”) se establece que toda imagen impresa, dispositivo o terminal que sea provista o facilitada por un adquirente o agregador que acepte pagos con tarjetas de crédito mediante la lectura de un código QR, debe permitir que toda billetera digital interoperable inscripta en el “Registro de billeteras digitales interoperables” pueda efectuar pagos con las tarjetas de crédito asociadas en el caso de que los comercios las acepten a través del correspondiente agregador o adquirente.

Este documento incorpora adecuaciones a la Interfaz Estandarizada de Pagos (IEP) definida en primera instancia en el boletín CIMPRA 525 y a los QR dinámicos que no utilicen la “API resolve” (boletín CIMPRA 530), enriqueciendo sus criterios operativos y realizando cambios en la mensajería del ecosistema a fin de extender a los pagos con tarjeta de crédito a través de códigos QR la interoperabilidad que hoy existe en los pagos con transferencia (PCT). La arquitectura de los esquemas presentados se desarrolló y consensuó en las “mesas técnicas de trabajo realizadas en el marco de la CIMPRA.

Adicionalmente, es importante destacar que la IEP no se altera a nivel estructura y nombres de atributo, asegurando así la retrocompatibilidad con lo desarrollado y consensuado para PCT (boletín CIMPRA 525 y posteriores).

QR interoperable, versión pago con tarjeta de crédito

Índice

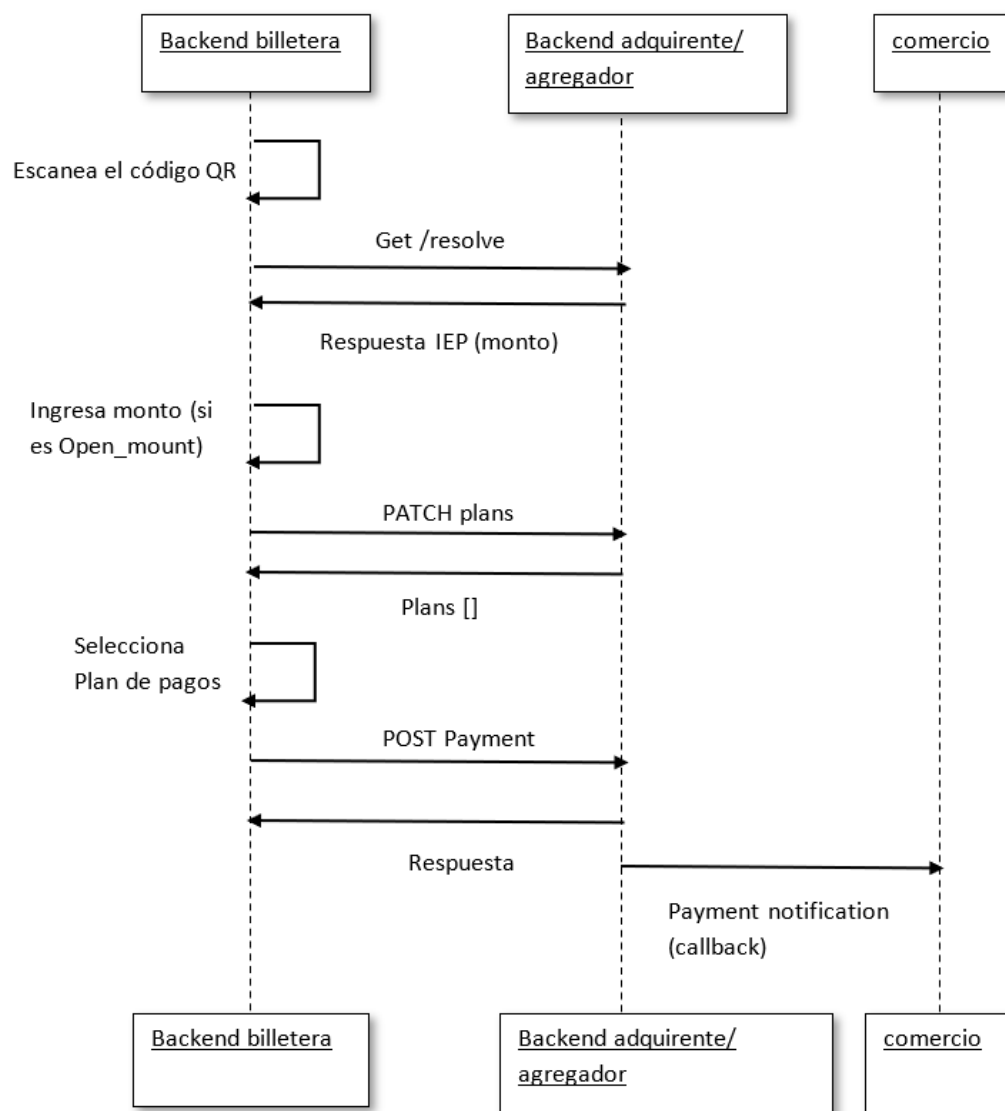
- 1. Solicitud de Pago Pasiva**
 - 1.1. Flujo de pago con utilización de “API resolve”**
 - 1.2. Flujo de pago sin utilización de “API resolve”**
 - 1.3. Flujo de devolución de pago**
- 2. Resolución del pago**
 - 2.1. Lectura de QR (desde billetera hacia adquirente/agregador)**
 - 2.1.1. QR con utilización de “API resolve” (boletín CIMPRA 525)**
 - 2.1.1.1. Monto abierto**
 - 2.1.2. QR sin utilización de “API resolve” (boletín CIMPRA 530)**
 - 2.2. Obtención de planes (desde billetera hacia adquirente/agregador)**
 - 2.2.1. PATCH plans**
 - 2.3. Envío intención de pago (desde billetera hacia adquirente/agregador)**
 - 2.3.1. POST Payment**
 - 2.4. Notificación de pago (desde adquirente/agregador hacia billetera)**
 - 2.4.1. POST payments/notify**
- 3. Anexos**
 - 3.1. Pospago**
 - 3.1.1. GET Payment by ID**
 - 3.1.2. GET Order by ID**
 - 3.2. Acceptor token**
 - 3.3. Códigos de respuesta**
 - 3.4. Estándar de tipos definidos**
 - 3.5. Estados de operación**
 - 3.6. Otras consideraciones**
 - 3.7. Swagger de APIs**

1. Solicitud de Pago Pasiva

1.1. Flujo de pago con utilización de API resolve

El esquema del flujo de pago involucra tres roles: billetera digital, adquirente/agregador y el comercio. La mayor cantidad de interacciones se da entre el backend¹ de la billetera con el del adquirente/agregador.

El propósito de este esquema es proporcionar un flujo de pagos eficiente, que involucra consultas a la Interfaz Estandarizada de Pagos (IEP): identificar al adquirente/agregador, tipo de monto (abierto o cerrado) y si el adquirente/agregador permite operar con tarjeta de crédito. Las consultas al backend del adquirente/agregador tienen que ver con planes de pago, monto de cuota y notificación de pago realizado, si fue aceptado o rechazado.



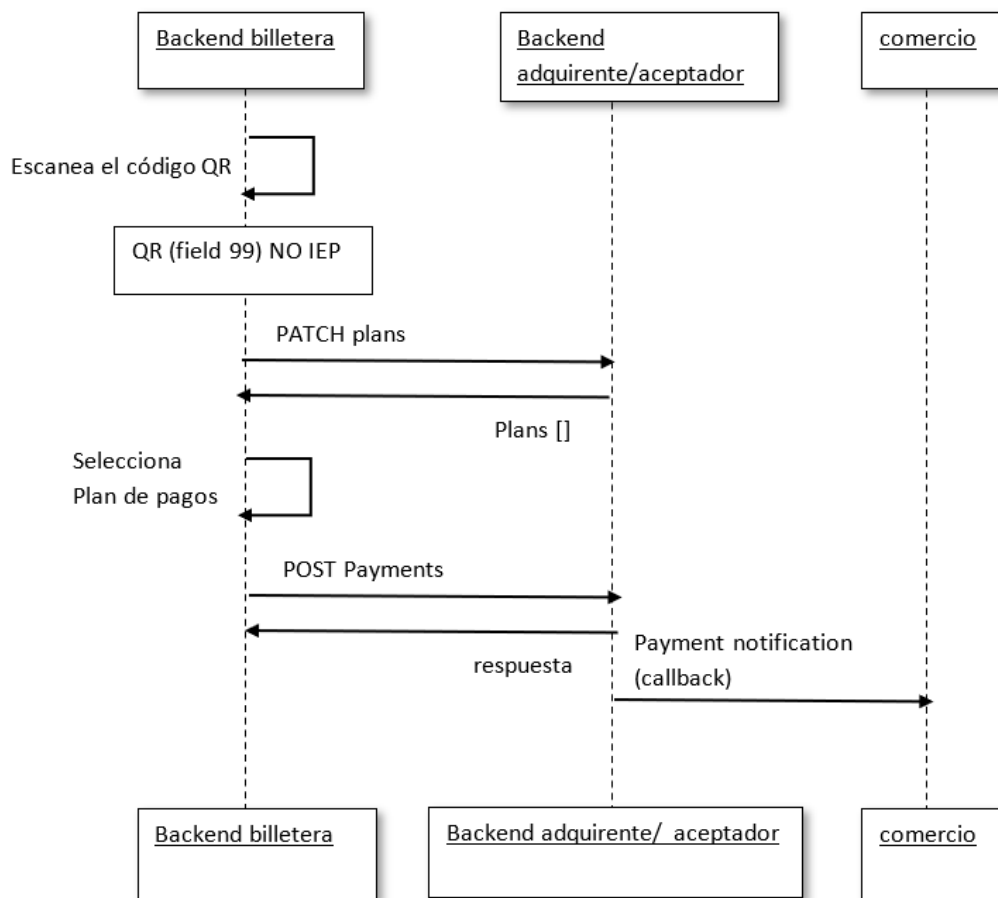
¹ Ejecuta funciones que no son visibles para el usuario final, pero de importancia para el buen funcionamiento y desarrollo del proceso.

1. Ante la lectura de un código QR (estático o dinámico), la billetera enviará el contenido completo del QR a su backend para ser procesado.
2. El backend de la billetera analizará el QR bajo estándar EMVCo Merchant Presented Code, y determinará el adquirente/ agregador mediante la búsqueda de qué Merchant Account Information contiene el QR (identificadores 26 a 49, ya que las demás posiciones ya se encuentran reservadas por el propio standard EMVCo o por la resolución del BCRA). Cada template contiene bajo el id 00 como identificador universal de fácil identificación un dominio invertido del adquirente/ agregador (para mayor detalle consultar boletín CIMPRA 525).
3. Una vez procesado el QR en el backend de la billetera y confirmado que el adquirente/agregador utiliza la IEP, se realizará un GET de información a la IEP para la obtención de mayor información para poder iniciar el pago.
 - a. Respuesta de la IEP con información. La respuesta del monto desde la IEP puede ser a monto cerrado o abierto, incluyendo información acerca de si el comercio permite operar o no con tarjetas de crédito.
4. En caso de aceptar tarjetas de crédito, la billetera solicita los planes con los cuales el adquirente/agregador permite operar. El adquirente/agregador devuelve los planes, junto a los montos de las cuotas calculados por el adquirente/agregador.
5. Se produce la confirmación del usuario del pago a realizar (selección de plan de pagos), ejecutando desde la billetera la iniciación del pago. Este paso contempla una respuesta síncrona de aprobación o rechazo, o una respuesta de error.
 - a. La billetera podrá consultar el estado de ese pago en el caso de que no haya recibido respuesta o información.

1.2. Flujo de pago sin utilización de API resolve

El esquema del flujo de pago involucra tres roles: billetera digital, adquirente/agregador y el comercio. La mayor cantidad de interacciones se da entre el backend de la billetera con el del adquirente/agregador.

El propósito de este esquema es proporcionar un flujo de pagos eficiente, que al escanear el código QR la billetera pueda: identificar al adquirente/ agregador; informarse de que el adquirente/agregador no tiene IEP, monto de la transacción y si el adquirente/agregador permite operar con tarjeta de crédito. Las consultas al backend del adquirente/ agregador tienen que ver con planes de pago, monto de cuota y notificación de monto de cuota y notificación de pago realizado, si fue aceptado o rechazado.



1. El backend de la billetera analizará el QR bajo estándar EMVCo Merchant Presented Code, y determinará si el adquirente/agregador utiliza IEP (entre campos 43 a 46, subcampo 99 por boletín CIMPRA 530)
2. Si no utiliza IEP el campo aparecerá en 00, y deberá enviar la consulta de planes al backend del adquirente/agregador, con la información del monto, moneda, informado en los campos del QR.
3. El adquirente/agregador devuelve los planes disponibles, junto a los montos de las cuotas calculados.
4. Se produce la confirmación del usuario y una posterior iniciación del pago, con la respuesta síncrona de aprobación o rechazo. Se puede consultar el estado de ese pago por el caso de que no haya recibido respuesta o información.

Ejemplo QR DATA sin IEP:

00 02 01
01 02 12
26 27

00 11 27260448213 CUIT
01 08 10000000

BRAND (0 es que no tiene la marca. 1 es que tiene la marca)

43 35

00 10 com.fiserv URL

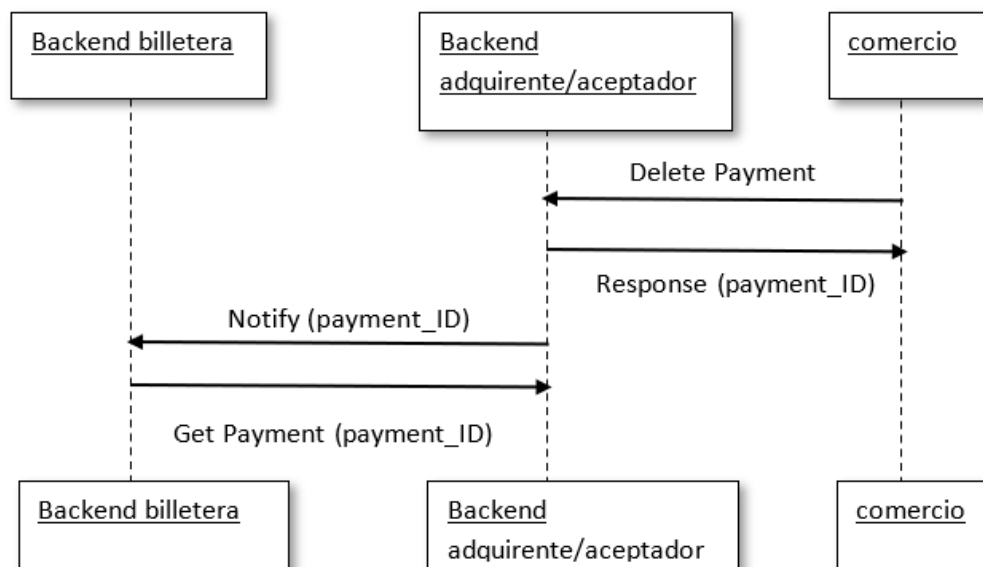
98 11 30692264785	CUIT COELSA
99 02 00	NO IEP
50 15 00 11 27260448213	CUIT
51 26 00 22 0000068000000002222956	CVU
52 04 5812	MCC
53 03 032	CURRENCY
54 06 100.00	AMOUNT
58 02 AR	COUNTRY CODE
59 13 POSNET.SA	MERCHANT NAME
60 12 VILLA GESELL	MERCHANT CITY
61 05 07165	POSTAL CODE
62 59	
01 12 0001000000001	BILL NUMBER
05 21 123456789012345678901	ID QR
07 08 26044821	DEVICE POS
08 02 00	PURCHASE
80 50	
00 10 com.fiserv	URL
01 05 02.01	VERSION
02 02 01	INSTALLMENTS (Ex. = 07 --> Ahora 12. =12 --> 12 cuotas)
03 12 220830154145	DATETIME (YYMMDDHHMMSS UTC -3)
04 01 1	DEVICE (POS, SISTEMA PROPIO, CLOVER, etc.)
82 93	
0005ecc010180MEUCIQDqU+T+mCEnJLCgUQMFPy0nkM2qjpCHRYTI//rm5ldwcQlgGz75Nu4p2vNJBGaCLGJRL6AiSz	
+G	
83 29 0005ecc020116Wx39AT9eDcUOQig=	
63 04 0e06	

Posición de los brands campo 26, subcampo 01.

1. Visa crédito
2. Visa débito
3. Mastercard
4. Mastercard debit
5. Maestro
6. Cabal
7. Amex
8. Tarjeta naranja

1.3. Flujo de devolución de pago

El único actor que puede iniciar el flujo de devoluciones es el comercio, pidiéndole al adquirente/agregador la devolución de un pago; luego este, será quien notifique a la billetera la devolución.



1. El comercio inicia una solicitud de devolución de pago adquirente/agregador.
2. El backend del adquirente/agregador recibe la solicitud y procesa la devolución.
3. Procesada la devolución, el backend del adquirente/agregador responde al comercio con el "payment_ID"
 - a. Notifica a la billetera el "payment_ID" de la devolución.
 - b. La billetera podrá consultar el estado de ese pago realizando un "GET Payment"² con el "payment_id" que recibió previamente.

² Método utilizado para consultar respecto a un pago específico.

2. Resolución del pago

2.1. Lectura de QR (desde billetera hacia adquirente/agregador)

Al momento de la lectura del QR, la billetera deberá identificar si el adquirente/agregador utiliza IEP o no, tal cual se define en el boletín CIMPRA 530.

2.1.1. QR con utilización de “API resolve” (boletín CIMPRA 525)

Una vez interpretado el QR, y detectado que utiliza IEP, la billetera realizará un “GET resolve” para solicitar la información necesaria para poder posteriormente cursar el pago. Estos QR podrán ser de monto abierto o monto cerrado.

Ejemplo de QR monto abierto:

```
{
  "status": "open_amount",
  "administrator": {
    "identification_number": "string",
    "name": "string"
  },
  "collector": {
    "account": "string",
    "identification_number": "string",
    "mcc": "string",
    "name": "string",
    "postal_code": "C1064AAD"
  },
  "order": {
    "id": "string",
    "items": {
      "currency_id": "ARS",
      "description": "Producto 1",
      "quantity": 1,
      "title": "string",
      "unit_price": 10000.99
    },
    "total_amount": 10000.99
  },
  "payment_methods_allowed": [ //los payment_methods_allowed deben ir explicitos
    porque son los que permiten a la billetera saber que metodos de pago ofrecer//
    {
      "id": "CARD",
      "restrictions": {
        "min_amount_allowed": 10000.99,
        "max_amount_allowed": 10000.99,
        "max_bins_allowed": "1235678"
      }
    }
  ],
  "additional_info": {}
}
```

2.1.2. QR sin utilización de “API resolve” (boletín CIMPRA 530)

Una vez interpretado el QR, y detectado que no utiliza IEP, la billetera deberá avanzar al siguiente paso de obtención de planes de pago.

Tener en cuenta que para los QR sin utilización de API resolve las billeteras deberán completar los siguientes datos con la información del QR DATA.

Ejemplo:

```
order_id: "123456789012345678901", // FISERV QRdata: 62. 05 QR ID
value: "100.0", // FISERV QRdata: 53 CURRENCY
currency: "ARS" // FISERV QRdata: 54 AMOUNT
```

2.2. Obtención de planes (desde billetera hacia adquirente/agregador)

La billetera solicita los planes con los cuales el adquirente/agregador permite operar.

2.2.1. PATCH plans

Endpoint: /orders/{order_id}/plans

Sentido: Billetera >> Adquirente/agregador

Timeout: 15000 ms

Header

x-request-id: guid

authorization: bearer (access_token)

Request:

```
{
  "bins": [
    {
      "original_bin": "1235678",
      "issuer_id": "string",
      "type": "CREDIT",
      "brand_id": "string"
    }
  ],
  "amount": {
    "value": 10000.99,
    "currency": "ARS"
  },
  "additional_info": {}
}
```

Response Status 200:

```
{
  "supported_bins": [
    {
      "brand_id": "string",
      "type": "CREDIT",
      "original_bins": [
        "1235678"
      ],

```

```

    "plans": [
      {
        "id": "string",
        "type": "AHORA",
        "description": "string",
        "installments": 1,
        "total_amount": {
          "value": 10000.99,
          "currency": "ARS"
        },
        "installment_amount": {
          "value": 10000.99,
          "currency": "ARS"
        },
        "financial_info": {
          "total_fianancial_cost": "80.00",
          "nominal_annual_rate": "80.00"
        },
        "requiered_fields": [ //el adquirente/agregador deberá informar a la
billetera los campos obligatorios a través de requiered_fields
          "payment_method.card.holder.name"
        ]
      }
    ]
  },
  "unsupported_bins": [
    "1235678"
  ],
  "additional_info": {}
}

```

Response Status > 4XX / 5XX:

```

{
  "code": "string",
  "message": "string",
  "additionalProp1": {}
}

```

2.3. Envío intención de pago (desde billetera hacia adquirente/agregador)

El usuario ya ha seleccionado la forma de pago, dando lugar a la iniciación del pago por parte de la billetera.

2.3.1. POST Payment

Endpoint: /orders/{order_id}/payments

Sentido: Billetera >> Adquirente/agregador

Observación: el objeto payment_method debe proponerse con una estructura que quede escalable para la posterior etapa de tokenización con las marcas. El flujo de pago de adquirente/agregador contempla 3 escenarios:

- ✓ card_data: Envío de datos planos de la tarjeta para cuando la billetera o el habilitador tecnológico que se utilice sea "PCI compliance"
- ✓ acceptor_token: implica enviar un token custom de cada adquirente/ agregador, utilizado y disponible solo para ese pago en ese momento.
- ✓ brand_token: utiliza el token de marca

Timeout: 15000 ms

Header

x-request-id: guid

x-idempotency-key: guid

authorization: bearer (access_token)

Request:

```
{
  "plan": {
    "id": "string",
    "type": "AHORA",
    "description": "string",
    "installments": 1,
    "total_amount": {
      "value": 10000.99,
      "currency": "ARS"
    },
    "installment_amount": {
      "value": 10000.99,
      "currency": "ARS"
    }
  },
  "payment_method": {
    "card": {
      "holder": {
        "name": "string",
        "identification_type": "DNI",
        "identification_number": "string"
      },
      // solo una de las 3 opciones "card_data", "acceptor_token", "brand_token" //
      "card_data": {
        "number": "string", //numero de tarjeta
        "security_code": "string", //cvv
        "expiration_month": 2, //mes de vencimiento
        "expiration_year": 9999, //año de vencimiento
        "entry_mode": "manual" //cof, manual
      },
      "acceptor_token": "string", //en caso de requerir un acceptor token, se deberá
      obtener el mismo a través de la api /payments/tokens
      "brand_token": {
        "original_bin": "string", //bin del pan original
        "original_last4": "string", //últimos 4 del pan original
        "token": "string",
        "cryptogram": "string",
        "security_indicator": "string", //Indicador de nivel de seguridad ej:
        "par": "string" //Hash
        "expiration_month": 2, //mes de vencimiento
      }
    }
  }
}
```

```

        "expiration_year": 9999, //año de vencimiento
        "token_requestor_id": "string"
    },
},
"wallet": {
    "name": "string", //nombre de la wallet que inicia el pago ej: Mercado pago,
    Cuenta DNI, Santander, Naranja X, Galicia, MOD0, etc.
    "provider": "string", //nombre del proveedor de infraestructura que permite
    hacer ese pago como por ej el valor es MOD0
    "brand_wallet_id": "string", //id específico que la marca de tarjeta asigna a
    cada billetera, aplica especialmente para VISA y MASTER
    "user": {
        "email": "user@example.com",
        "phone": "string",
        "device_id": "string",
        "identification_type": "DNI",
        "identification_number": "string"
    }
}
},
"additional_info": {}
}

```

Response Status 200:

```

{
    "payment_id": "string",
    "order_id": "string",
    "status": "APPROVED",
    "status_code": "APPROVED",
    "amount": {
        "value": 10000.99,
        "currency": "ARS"
    },
    "authorized_amount": {
        "value": 10000.99,
        "currency": "ARS"
    },
    "plan": {
        "id": "string",
        "type": "AHORA",
        "description": "string",
        "installments": 1,
        "total_amount": {
            "value": 10000.99,
            "currency": "ARS"
        },
    },
    "installment_amount": {
        "value": 10000.99,
        "currency": "ARS"
    },
    "financial_info": {
        "total_fianancial_cost": "80.00",
        "nominal_annual_rate": "80.00"
    }
},

```

```

"card": {
  "original_bin": "1235678",
  "original_last4": "string",
  "type": "string",
  "brand_id": "string",
  "issuer_id": "string",
  "holder": {
    "name": "string",
    "identification_type": "DNI",
    "identification_number": "string"
  }
},
"wallet": {
  "name": "string",
  "provider": "string",
  "brand_wallet_id": "string",
  "user": {
    "email": "user@example.com",
    "phone": "string",
    "device_id": "string",
    "identification_type": "DNI",
    "identification_number": "string"
  }
},
"authorization_code": "string", //ej: "123456"
"refunds": [
  {
    "amount": {
      "value": 10000.99,
      "currency": "ARS"
    },
    "created_at": "2023-10-26T17:26:03.953Z"
  }
],
"created_at": "2023-10-26T17:26:03.953Z",
"updated_at": "2023-10-26T17:26:03.953Z",
"additional_info": {}
}

```

Response Status > 4XX / 5XX Generic error response:

```

{
  "code": "string",
  "message": "string",
  "additionalProp1": {}
}

```

2.4. Notificación de pago (desde adquirente/agregador hacia billetera)

Una vez que el adquirente/agregador confirme que la operación concluyó, el adquirente/agregador comunicará el resultado de la intención de pago a la billetera.

El adquirente/agregador notificará siempre a la billetera los siguientes estados: APPROVED, REJECTED, REFUNDED, CHARGED BACK; el único estado que no será notificado sin previa consulta por parte de la billetera será el de PROCESSING.

2.4.1. POST payments/notify

Request:

```
{  
  "payment_id": "string",  
  "domain_reverse": "string"  
}
```

Response Status 204: OK

Response Status > 4XX / 5XX

Generic error response:

```
{  
  "code": "string",  
  "message": "string",  
  "additionalProp1": {}  
}
```

3. Anexos

3.1. Pospago

3.1.1. GET Payment by ID

La billetera podrá realizar una consulta de un pago a través del GET Payment by ID para asegurarse el estado final de la operación, dado que puede haber contracargos, cancelaciones o devoluciones de pago.

Response Status 2XX:

```
{
  "payment_id": "string",
  "order_id": "string",
  "status": "PROCESSING",
  "status_code": "PROCESSING",
  "amount": {
    "value": 10000.99,
    "currency": "ARS"
  },
  "authorized_amount": {
    "value": 10000.99,
    "currency": "ARS"
  },
  "plan": {
    "id": "string",
    "type": "AHORA",
    "description": "string",
    "installments": 1,
    "total_amount": {
      "value": 10000.99,
      "currency": "ARS"
    },
    "installment_amount": {
      "value": 10000.99,
      "currency": "ARS"
    },
    "financial_info": {
      "total_fianancial_cost": "80.00",
      "nominal_annual_rate": "80.00"
    }
  },
  "card": {
    "original_bin": "1235678",
    "original_last4": "string",
    "type": "string",
    "brand_id": "string",
    "issuer_id": "string",
    "holder": {
      "name": "string",
      "identification_type": "DNI",

```

```

        "identification_number": "string"
    },
    },
    "wallet": {
        "name": "string",
        "provider": "string",
        "brand_wallet_id": "string",
        "user": {
            "email": "user@example.com",
            "phone": "string",
            "device_id": "string",
            "identification_type": "DNI",
            "identification_number": "string"
        }
    },
    },
    "authorization_code": "string",
    "refunds": [
        {
            "amount": {
                "value": 10000.99,
                "currency": "ARS"
            },
            "created_at": "2023-10-26T19:50:04.954Z"
        }
    ],
    "created_at": "2023-10-26T19:50:04.954Z",
    "updated_at": "2023-10-26T19:50:04.954Z",
    "additional_info": {}
}

```

Response Status > 4XX / 5XX Generic error response:

```

{
  "code": "string",
  "message": "string",
  "additionalProp1": {}
}

```

3.1.2. GET Order by ID

Como opción adicional, la billetera podrá consultar todos los pagos relacionados a una orden a través del endpoint GET/orders/{order_id}/payments.

Response Status 2XX:

```

[
  {
    "payment_id": "string",
    "order_id": "string",
    "status": "PROCESSING",
    "status_code": "PROCESSING",
    "amount": {
      "value": 10000.99,
      "currency": "ARS"
    }
  }
]

```

```

},
"authorized_amount": {
  "value": 10000.99,
  "currency": "ARS"
},
"plan": {
  "id": "string",
  "type": "AHORA",
  "description": "string",
  "installments": 1,
  "total_amount": {
    "value": 10000.99,
    "currency": "ARS"
  },
  "installment_amount": {
    "value": 10000.99,
    "currency": "ARS"
  },
  "financial_info": {
    "total_fianancial_cost": "80.00",
    "nominal_annual_rate": "80.00"
  }
},
"card": {
  "original_bin": "1235678",
  "original_last4": "string",
  "type": "string",
  "brand_id": "string",
  "issuer_id": "string",
  "holder": {
    "name": "string",
    "identification_type": "DNI",
    "identification_number": "string"
  }
},
"wallet": {
  "name": "string",
  "provider": "string",
  "brand_wallet_id": "string",
  "user": {
    "email": "user@example.com",
    "phone": "string",
    "device_id": "string",
    "identification_type": "DNI",
    "identification_number": "string"
  }
},
"authorization_code": "string",
"refunds": [
  {
    "amount": {
      "value": 10000.99,
      "currency": "ARS"
    },
    "created_at": "2023-10-26T19:56:13.912Z"
  }
]

```

```

    }
  ],
  "created_at": "2023-10-26T19:56:13.912Z",
  "updated_at": "2023-10-26T19:56:13.912Z",
  "additional_info": {}
}
]

```

Response Status > 4XX / 5XX Generic error response:

```

{
  "code": "string",
  "message": "string",
  "additionalProp1": {}
}

```

3.2. Acceptor token

En aquellos casos donde se deba utilizar el token de adquirente/agregador, la billetera lo obtendrá a través de las siguiente API /payments/tokens

Para realizar el post, se podrá enviar tanto card_data, como brand_token (solo se deberá enviar uno de los dos).

Request:

```

{
  "card": {
    "holder": {
      "name": "string",
      "identification_type": "DNI",
      "identification_number": "string"
    }, //se envía una sola de las siguientes opciones
    "card_data": {
      "number": "string",
      "security_code": "string",
      "expiration_month": 12,
      "expiration_year": 9999,
      "entry_mode": "MANUAL"
    },
    "brand_token": {
      "original_bin": "string", //bin del pan original
      "original_last4": "string", //ultimos 4 del pan original
      "token": "string",
      "cryptogram": "string",
      "security_indicator": "string", //Indicador de nivel de seguridad ej:
      "par": "string" //Hash
      "expiration_month": 2, //mes de vencimiento
      "expiration_year": 9999, //año de vencimiento
      "token_requestor_id": "string"
    }
  }
}

```

Response Status 2XX:

```
{
  "token": "string",
  "created_at": "2023-11-09T19:36:03.456Z",
  "expiry_at": "2023-11-09T19:36:03.456Z"
}
```

Response Status > 4XX / 5XX**Generic error response:**

```
{
  "code": "string",
  "message": "string",
  "additionalProp1": {}
}
```

3.3. Códigos de respuesta

Las respuestas deben responderse con códigos HTTP apropiados para todas las API.

Código	Respuesta	Descripción
200	Ok	La petición fue exitosa. También se utiliza para peticiones a colecciones vacías
400	Bad Request	Existe algún problema en el formato del request
401	Unauthorized	Problemas de autenticación: el request no contiene credenciales o las mismas son inválidas
403	Forbidden	El request es correcto, pero el usuario no tiene permisos suficientes
404	Not Found	No se encontraron {entidad}
500	Internal Server Error	Algún error inesperado sucedió en el servidor
502	Bad Gateway	El servicio que provee la API está ok, pero un servicio del cual depende respondió con un error interno
503	Service Unavailable	El servidor no está disponible debido a algún problema operativo propio del servicio,

		como una sobrecarga
504	Gateway timeout	El servicio que provee la API está OK, pero un servicio del cual depende no está contestando dentro del tiempo esperado

3.4. Estándar de tipos definidos

Atributo	enum
brand_id	VISA, MASTER, AMEX, CABAL, MAESTRO, NARANJA, SUCREDITO, ARGENCARD, DISCOVER, NATIVA
card.type	DEBIT, CREDIT, PREPAID
identification_type	DNI, CUIT, CUIL
entry_mode	MANUAL, COF
plans.type	AHORA: Plan financiado por el emisor con tasa fijada por el gobierno. EMISOR: Plan financiado por el emisor. ADQUIRENTE: Plan financiado por el adquirente. COMERCIO: Plan financiado por el comercio. ALIMENTAR: xx ZETA: Plan financiado por Naranja.
currency	ARS
issuer_id	Detallar tomando como base los listados de entidades financieras del BCRA y el registro de empresas no financieras emisoras de tarjetas de crédito o compra. https://www.bcra.gob.ar/SistemasFinanciero sYdePagos/Sistema_financiero_nomina_de_entidades.asp?bco=AAA00&tipo=1 https://www.bcra.gob.ar/SistemasFinanciero sYdePagos/Emisoras_tarjetas_credito_comp ra.asp

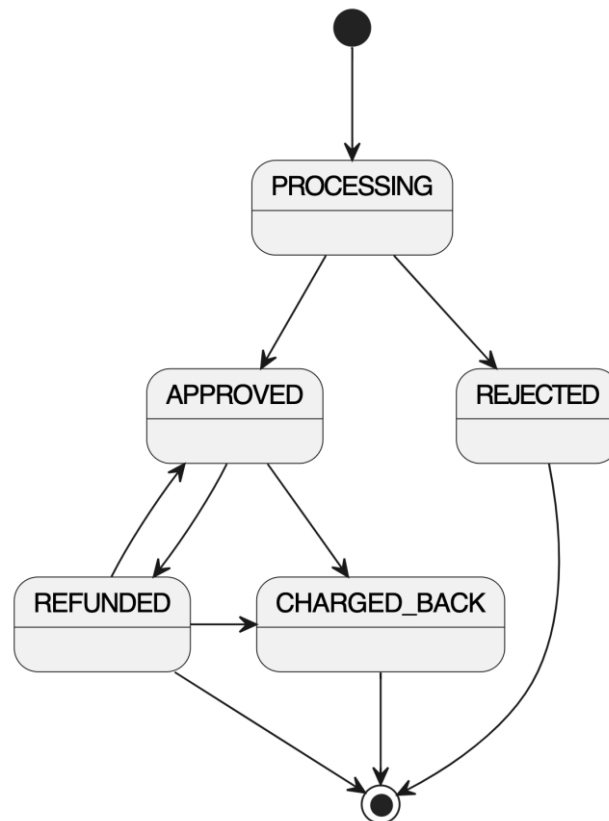
3.5. Estados de operación

Se contemplan los siguientes estados para las operaciones:

- PROCESSING
- APPROVED

- REJECTED
- REFUNDED
- CHARGED BACK

Se define la siguiente máquina de estados:



```

@startuml
[*] -down-> PROCESSING
PROCESSING -down-> APPROVED
APPROVED -down-> REFUNDED
APPROVED -> CHARGED_BACK
REFUNDED -> CHARGED_BACK
REFUNDED -left-> APPROVED
PROCESSING -> REJECTED
REJECTED -> [*]
REFUNDED -> [*]
CHARGED_BACK -down-> [*]
@enduml
  
```

En el status code se obtendrá mayor información del estado informado, pudiendo ser:

Status_code	Descripción
processing	

approved	
rejected_insufficient_funds	fondos insuficientes, excede límites,
rejected_call_for_auth	
rejected_declined	motivos 05, otros como fraude
rejected_invalid_card	Cualquier error relacionado a la tarjeta: número incorrecto, tarjeta denunciada, validaciones de tokenización, tarjeta expirada, cvv incorrecto,
rejected_invalid_transaction	cuotas no soportadas, bin incorrecto, tipo de tarjeta incorrecto (debit, credit)
rejected_invalid_merchant	comercio no configurado correctamente
rejected_system_error	emisor fuera de línea, adquirente fuera de línea
rejected_invalid_order	orden cancelada, la orden ya fue pagada,
rejected_max_attempts	máximo de intentos alcanzados
refunded	devolución y anulación
refunded_partially	
charged_back	

3.6. Otras consideraciones

- Zona horaria: UTC-3
- Autenticación para servicios expuestos entre adquirente/agregador y billetera: los servicios que sean consumidos *"frontend to backend"* deberán contar con credenciales de autenticación distintas a las utilizadas para consumir servicios *"backend to backend"*. Las credenciales de autenticación podrán tener un tiempo de vida y un mecanismo para su rotación.
- Servicios de *"notify"* a las billeteras: al momento de realizar la integración entre adquirentes/agregadores y billeteras, se solicitará a esta última informar la URL en la cual deberá recibir las notificaciones por parte del adquirente/agregador.

3.7. Swagger de APIs

```
openapi: 3.0.3
info:
  title: Interfaz Estandarizada de Pagos - QR
  version: 0.0.1
  description: This contract define the APIs exposed by acquirers to any wallet
operating QR payment solutions inside Argentina.
```

```

    contact:
      name: Tomás Mingo Suarez
      email: tomas.mingo@globant.com
  externalDocs:
    description: find IEP Specification here
    url: https://www.bcra.gob.ar/pdfs/sistemasfinancierosydepagos/SNP3525.pdf
  paths:
    /resolve:
      get:
        operationId: getResolve
        summary: Recives the qr string and return the standarized answer for T3 , TD and
TC
        tags:
          - Resolve
        parameters:
          - $ref: '#/components/parameters/QRData'
          - $ref: '#/components/parameters/AccessToken'
          - $ref: '#/components/parameters/TraceId'
        responses:
          '2XX':
            $ref: '#/components/responses/Resolve200'
          '4XX':
            $ref: '#/components/responses/GenericError'
          '5XX':
            $ref: '#/components/responses/GenericError'
    /orders/{order_id}/plans:
      patch:
        operationId: patchPlansByOrderId
        summary: Recives the order_Id, amount (if the order has the 'status' field with
'open_amount' value) and bins and return all available plans for the defined order.
        tags:
          - Orders
        parameters:
          - $ref: '#/components/parameters/TraceId'
          - $ref: '#/components/parameters/OrderId'
        requestBody:
          required: true

```

```

    content:
      application/json:
        schema:
          $ref: '#/components/schemas/PlansRequest'
  responses:
    '2XX':
      $ref: '#/components/responses/Plans200'
    '4XX':
      $ref: '#/components/responses/GenericError'
    '5XX':
      $ref: '#/components/responses/GenericError'
/orders/{order_id}/payments:
  post:
    operationId: postPaymentsByOrderId
    summary: Execute a payment from the given order_id
    tags:
      - Orders
    parameters:
      - $ref: '#/components/parameters/IdempotencyId'
      - $ref: '#/components/parameters/TraceId'
      - $ref: '#/components/parameters/OrderId'
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/PaymentsRequest'
    responses:
      '2XX':
        $ref: '#/components/responses/Payments200'
      '4XX':
        $ref: '#/components/responses/GenericError'
      '5XX':
        $ref: '#/components/responses/GenericError'
  get:
    operationId: getPaymentsByOrderId
    summary: Get a payment from the given order_id

```

```

tags:
  - Orders
parameters:
  - $ref: '#/components/parameters/TraceId'
  - $ref: '#/components/parameters/OrderId'
responses:
  '2XX':
    $ref: '#/components/responses/GetPayments200'
  '4XX':
    $ref: '#/components/responses/GenericError'
  '5XX':
    $ref: '#/components/responses/GenericError'
/payments/{payment_id}:
get:
  operationId: getPaymentsByPaymentId
tags:
  - Payments
parameters:
  - $ref: '#/components/parameters/TraceId'
  - $ref: '#/components/parameters/PaymentId'
responses:
  '2XX':
    $ref: '#/components/responses/Payments200'
  '4XX':
    $ref: '#/components/responses/GenericError'
  '5XX':
    $ref: '#/components/responses/GenericError'
/payments/notify:
post:
  operationId: postPaymentsNotifyUpdate
  summary: Endpoint exposed by any wallet provier to recive a notification that
something changes in a particular payment
tags:
  - Wallet
requestBody:
  required: true
  content:

```

```

        application/json:
          schema:
            $ref: '#/components/schemas/PaymentNotifyUpdate'
      responses:
        '204':
          description: OK
        '4XX':
          $ref: '#/components/responses/GenericError'
        '5XX':
          $ref: '#/components/responses/GenericError'
    /payments/tokens:
      post:
        operationId: postPaymentsTokenization
        summary: Generate an Acceptor Token to be used in a payment.
        tags:
          - Payments
        parameters:
          - $ref: '#/components/parameters/TraceId'
        requestBody:
          required: true
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/TokenRequest'
        responses:
          '2XX':
            $ref: '#/components/responses/Tokenization200'
          '4XX':
            $ref: '#/components/responses/GenericError'
          '5XX':
            $ref: '#/components/responses/GenericError'

components:
  schemas:

    # Requests

```

```

TokenRequest:
  type: object
  properties:
    card:
      type: object
      properties:
        holder:
          $ref: '#/components/schemas/CardHolder'
    oneOf:
      - $ref: '#/components/schemas/CardData'
      - $ref: '#/components/schemas/BrandToken'
PlansRequest:
  type: object
  properties:
    bins:
      type: array
      items:
        $ref: '#/components/schemas/Bin'
    amount:
      $ref: '#/components/schemas/Amount'
    additional_info:
      type: object
  required:
    - order_id
    - bins
    - amount
PaymentsRequest:
  type: object
  properties:
    plan:
      $ref: '#/components/schemas/Plan'
    payment_method:
      $ref: '#/components/schemas/PaymentMethod'
    additional_info:
      type: object
  required:
    - plan

```

```

    - payment_method
PaymentNotifyUpdate:
  type: object
  properties:
    payments_id:
      type: string
    domain_reverse:
      type: string
      description: Same field exposed in the QR
  required:
    - payment_id
    - domain_reverse

# Responses

Resolve:
  type: object
  properties:
    status:
      type: string
      description: Define if the amount is open to be added by the wallet or not
      enum:
        - "open_amount"
        - "closed_amount"
    administrator:
      $ref: '#/components/schemas/Administrator'
    collector:
      $ref: '#/components/schemas/Collector'
    order:
      $ref: '#/components/schemas/Order'
    payment_methods_allowed:
      type: array
      items:
        $ref: '#/components/schemas/PaymentMethodIEP'
    additional_info:
      type: object
  required:

```

```

    - status
    - administrator
    - collector
    - order
    - payment_methods_allowed
PlansResponse:
  type: object
  properties:
    supported_bins:
      type: array
      items:
        $ref: '#/components/schemas/SupportedBin'
    unsupported_bins:
      type: array
      items:
        $ref: '#/components/schemas/BinField'
    additional_info:
      type: object
GetPaymentsResponse:
  type: array
  items:
    $ref: '#/components/schemas/PaymentsResponse'
PaymentsResponse:
  type: object
  properties:
    payment_id:
      type: string
    order_id:
      type: string
    status:
      $ref: '#/components/schemas/PaymentStatus'
    status_code:
      $ref: '#/components/schemas/PaymentStatusCode'
    amount:
      $ref: '#/components/schemas/Amount'
    authorized_amount:
      $ref: '#/components/schemas/Amount'

```

```

    plan:
      $ref: '#/components/schemas/Plan'
    card:
      $ref: '#/components/schemas/Card'
    wallet:
      $ref: '#/components/schemas/Wallet'
    authorization_code:
      type: string
    refunds:
      type: array
      items:
        $ref: '#/components/schemas/Refund'
    created_at:
      type: string
      format: date-time
    updated_at:
      type: string
      format: date-time
    additional_info:
      type: object
  required:
    - payment_id
    - order_id
    - status
    - status_code
    - amount
    - authorized_amount
    - plan
    - created_at
    - updated_at
  TokenizationResponse:
    type: object
  properties:
    token:
      type: string
    created_at:
      type: string

```

```

        format: date-time
    expiry_at:
        type: string
        format: date-time
Error:
    type: object
    additionalProperties: true
    properties:
        code:
            type: string
        message:
            type: string
    required:
        - code
        - message

# Objects

Administrator:
    type: object
    properties:
        identification_number:
            type: string
        name:
            type: string
    required:
        - identification_number
        - name
Collector:
    type: object
    properties:
        account:
            type: string
        indentification_number:
            type: string
        mcc:
            type: string

```

```

    name:
      type: string
    postal_code:
      type: string
      example: "C1064AAD"
  required:
    - account
    - identification_number
    - mcc
    - name
    - postal_code
Order:
  type: object
  properties:
    id:
      type: string
    items:
      $ref: '#/components/schemas/Items'
    total_amount:
      $ref: '#/components/schemas/AmountField'
  required:
    - id
    - total_amount
Items:
  type: object
  properties:
    currency_id:
      $ref: '#/components/schemas/CurrencyField'
    description:
      type: string
      example: "Producto 1"
    quantity:
      type: integer
      example: 1
    title:
      type: string
    unit_price:

```

```

    $ref: '#/components/schemas/AmountField'
  required:
    - currency_id
    - description
    - title
    - unit_price
PaymentMethodIEP:
  type: object
  properties:
    id:
      type: string
      enum:
        - "CARD"
        - "TRANSFER"
    restrictions:
      type: object
      properties:
        min_amount_allowed:
          $ref: '#/components/schemas/AmountField'
        max_amount_allowed:
          $ref: '#/components/schemas/AmountField'
        max_bins_allowed:
          $ref: '#/components/schemas/BinField'
      required:
        - id
Bin:
  type: object
  properties:
    original_bin:
      $ref: '#/components/schemas/BinField'
    issuer_id:
      $ref: '#/components/schemas/IssuerIdField'
    type:
      $ref: '#/components/schemas/CardTypeField'
    brand_id:
      $ref: '#/components/schemas/BrandIdField'
  required:

```

```

    - original_bin
    - issuer_id
    - type
    - brand_id
Amount:
  type: object
  properties:
    value:
      $ref: '#/components/schemas/AmountField'
    currency:
      $ref: '#/components/schemas/CurrencyField'
  required:
    - amount
SupportedBin:
  type: object
  properties:
    brand_id:
      $ref: '#/components/schemas/BrandIdField'
    type:
      $ref: '#/components/schemas/CardTypeField'
    original_bins:
      type: array
      items:
        $ref: '#/components/schemas/BinField'
    plans:
      type: array
      items:
        allOf:
          - $ref: '#/components/schemas/Plan'
          - $ref: '#/components/schemas/RequieredFields'
  required:
    - brand_id
    - type
    - original_bin
    - plans
RequieredFields:
  type: object

```

```

    properties:
      requiered_fields:
        type: array
        description: Any optional field from the POST /payments endpoint could be
added on this list. If a field is reported here, that field will be considered
mandatory in the context of this particular order. In the case of nested fields all
objects will be included using a dot "." as separator
        items:
          type: string
          example: "payment_method.card.holder.name"
        readOnly: true
  Plan:
    type: object
    properties:
      id:
        type: string
      type:
        $ref: '#/components/schemas/PlanTypeField'
      description:
        type: string
        description: A description ready to be rendered in the frontend
      installments:
        type: integer
        example: 1
      total_amount:
        $ref: '#/components/schemas/Amount'
      installment_amount:
        $ref: '#/components/schemas/Amount'
      financial_info:
        $ref: '#/components/schemas/FinancialInfo'
    required:
      - id
      - type
      - description
      - installments
      - total_amount
      - installment_amount

```

```

FinancialInfo:
  type: object
  properties:
    total_fianancial_cost:
      $ref: '#/components/schemas/PercentageField'
    nominal_annual_rate:
      $ref: '#/components/schemas/PercentageField'
  readOnly: true
PaymentMethod:
  type: object
  properties:
    card:
      type: object
      properties:
        holder:
          $ref: '#/components/schemas/CardHolder'
    oneOf:
      - $ref: '#/components/schemas/AcceptorToken'
      - $ref: '#/components/schemas/CardData'
      - $ref: '#/components/schemas/BrandToken'
    wallet:
      $ref: '#/components/schemas/Wallet'
  required:
    - card
    - wallet
CardData:
  type: object
  properties:
    card_data:
      type: object
      properties:
        number:
          type: string
        security_code:
          type: string
        expiration_month:
          type: integer

```

```

        minimum: 1
        maximum: 12
    expiration_year:
        type: integer
        minimum: 1000
        maximum: 9999
    entry_mode:
        type: string
        enum:
            - MANUAL
            - COF
    required:
        - number
        - security_code
        - expiration_month
        - expiration_year
        - entry_mode
    required:
        - card_data
BrandToken:
    type: object
    properties:
        brand_token:
            type: object
            properties:
                original_bin:
                    $ref: '#/components/schemas/BinField'
                original_last4:
                    type: string
                token:
                    type: string
                cryptogram:
                    type: string
                security_indicator:
                    type: string
                    description: Security level indicator according to brand standards (for
example ECI for VISA)

```

```

    par:
      type: string
    expiration_year:
      type: integer
      minimum: 1000
      maximum: 9999
    expiration_month:
      type: integer
      minimum: 1
      maximum: 12
    token_requestor_id:
      description: Token Requestor ID
      type: string
    required:
      - original_bin
      - original_last4
      - token
      - cryptogram
      - security_indicator
  required:
    - brand_token
  AcceptorToken:
    type: object
    properties:
      acceptor_token:
        type: string
    required:
      - acceptor_token
  CardHolder:
    type: object
    properties:
      name:
        type: string
      identification_type:
        $ref: '#/components/schemas/IdentificationTypeField'
      identification_number:
        type: string

```

```

Card:
  type: object
  properties:
    original_bin:
      $ref: '#/components/schemas/BinField'
    original_last4:
      type: string
    type:
      type: string
    brand_id:
      $ref: '#/components/schemas/BrandIdField'
    issuer_id:
      $ref: '#/components/schemas/IssuerIdField'
    holder:
      $ref: '#/components/schemas/CardHolder'
  required:
    - original_bin
    - original_last4
Wallet:
  type: object
  properties:
    name:
      type: string
    provider:
      type: string
    brand_wallet_id:
      type: string
    user:
      $ref: '#/components/schemas/User'
  required:
    - name
    - provider
User:
  type: object
  properties:
    email:
      type: string

```

```

        format: email
    phone:
        type: string
    device_id:
        type: string
    identification_type:
        $ref: '#/components/schemas/IdentificationTypeField'
    identification_number:
        type: string
Refund:
    type: object
    properties:
        amount:
            $ref: '#/components/schemas/Amount'
        created_at:
            type: string
            format: date-time
    required:
        - amount
        - created_at

# Fields

PaymentStatus:
    type: string
    enum:
        - PROCESSING
        - APPROVED
        - REJECTED
        - REFUNDED
        - CHARGED_BACK
PaymentStatusCode:
    type: string
    enum:
        - PROCESSING
        - APPROVED
        - REJECTED_INSUFFICIENT_FUNDS

```

- REJECTED_CALL_FOR_AUTH
- REJECTED_DECLINED
- REJECTED_INVALID_CARD
- REJECTED_INVALID_TRANSACTION
- REJECTED_INVALID_MERCHANT
- REJECTED_SYSTEM_ERROR
- REJECTED_INVALID_ORDER
- REJECTED_MAX_ATTEMPTS
- REFUNDED
- REFUNDED_PARTIALLY
- CHARGED_BACK

IdentificationTypeField:

type: string

enum:

- DNI
- CUIL

PercentageField:

type: string

format: percentage

example: "80.00"

PlanTypeField:

type: string

example: "AHORA"

description: This value must be taken from the standarized list of plan types.

The list itself is part of the IEP documentation.

AmountField:

type: number

format: amount

example: 10000.99

CurrencyField:

type: string

format: currencyCodeISO4217

default: "ARS"

example: "ARS"

BinField:

type: string

format: bins

```

    example: "1235678"
IssuerIdField:
  type: string
  description: This value must be taken from the standarized list of card issuers
ids. The list itself is part of the IEP documentation and is based on the BCRA's
'Nomina de
Entidades' (https://www.bcra.gob.ar/SistemasFinancierosYdePagos/Sistema\_financiero\_nomi
na\_de\_entidades.asp?bco=AAA00&tipo=1)
CardTypeField:
  type: string
  enum:
    - CREDIT
    - DEBIT
    - PREPAID
BrandIdField:
  type: string
  description: This value must be taken from the standarized list of card brands
ids. The list itself is part of the IEP documentation.
parameters:
  QRData:
    name: data
    in: query
    schema:
      type: string
  AccessToken:
    name: access_token
    in: query
    schema:
      type: string
    deprecated: true
  IdempotencyId:
    name: idempotency-id
    in: header
    description: Id for idempotency recognition. Used to recognize several attemps
of the same requests. In a retry logic the same request-id must be sent.
    schema:
      type: string

```

```

TraceId:
  name: trace-id
  in: header
  description: Id for tracing purpose. Used to trace the request through every
internal system. Must be different on every transaction or attempt.
  schema:
    type: string
OrderId:
  name: order_id
  in: path
  description: Unambiguous order identifier
  schema:
    type: string
  required: true
PaymentId:
  name: payment_id
  in: path
  description: Unambiguous payment identifier
  schema:
    type: string
  required: true
responses:
  Resolve200:
    description: Successful response
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/Resolve'
  Plans200:
    description: Successful response
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/PlansResponse'
  Payments200:
    description: Successful response
    content:

```

```

    application/json:
      schema:
        $ref: '#/components/schemas/PaymentsResponse'
  GetPayments200:
    description: Successful response
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/GetPaymentsResponse'
  Tokenization200:
    description: Successful response
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/TokenizationResponse'
  GenericError:
    description: Generic error response
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/Error'

securitySchemes:
  bearerAuth:
    type: http
    scheme: bearer

security:
  - bearerAuth: []

```