Transferencias 3.0

Transferencias inmediatas pull **Consentimiento**

Versión 2023

Documento Técnico

Comisión Interbancaria
para los Medios de Pago
de la República Argentina

BANCO CENTRAL
DE LA REPÚBLICA ARGENTINA

Índice:

1. Objetivo 2
2. Convenciones de notación 2
3. Principios 2
    a. RESTful APIs 2
    b. ISO 20022 2
4. Mapeo de riesgos y sugerencias en flujo de transacciones 3
5. Normas de seguridad 4
6. Flujos de OAUTH2+PKCE 6
    Seguridad en el flujo de autorización de operaciones sobre la cuenta 7
        7. Flujo de OAuth 2.0 Authorization code + PKCE 10
        10. Flujo transaccional 12
7. Régimen de mantenimiento y actualización del documento 16
8. Referencias normativas 16
9. Anexos 16
    Anexo I – APIs 16
        a. Alta 16
        b. Intercambio de code por tokens 18
        c. Baja / Desvinculación 20
    Anexo II – Proceso de gestión de certificados para consentimiento 21
    Anexo III – Llaves públicas 22

Aclaración: El presente documento fue realizado en colaboración entre los participantes de las mesas de trabajo, para esta primera salida a producción. Se deja asentado que:

- En el mismo se presenta un modelo de validación en terceros, el cual fue acordado por los participantes de la mesa, pero con el objetivo de migrar a un modelo de introspección, donde el token sea validado por el proveedor de la cuenta.
- Queda sujeto a revisión los tiempos de duración de los tokens estipulados en esta versión del documento.
- Se aclara que en esta versión del documento se considera como entidad enrolante únicamente a las billeteras digitales en su versión mobile. Para MVPs posteriores, independientemente del canal a utilizar, son aplicables los mismos requisitos y recomendaciones.

1.Objetivo

En atención a la Comunicación "A" 7514 publicada el 19.05.22, este documento propone detallar técnicamente cómo será la interacción de consentimiento y servicios transaccionales. Este documento es una propuesta conjunta y colaborativa entre entidades participantes de la CIMPRA, por lo que los cambios a este documento y las decisiones arquitecturales deben realizarse en el marco de la misma.

2. Convenciones de notación

Las palabras clave "deberá" (shall), "no deberá" (shall not), "debería" (should), "no debería" (should not) y "puede" (may) presentes en este documento deben interpretarse de acuerdo con las pautas descritas en RFC2119.

3. Principios

a. RESTful APIs

La API se adherirá a los conceptos de API RESTful siempre que sea posible y sensato.

b. ISO 20022

Los payloads (contenidos) de las APIs se desarrollarán utilizando elementos y componentes de mensajes ISO 20022 como base, que pueden modificarse, si es necesario, para simplificar la carga útil y/o cumplir con las características locales, según se implemente en diferentes jurisdicciones.

4. Mapeo de riesgos y sugerencias en flujo de transacciones

Riesgos y sugerencias

1. **Uso de certificados mTLS sin control de emisión**
    - Impacto
        - Alguien puede hacerse pasar por una entidad financiera o PSP legítima.
    - Mitigación
        - P2P con ceremonias descentralizadas.

2. **Robo de token de sesión**
    - Impacto
        - Técnicamente no existe un mecanismo de proof-of-possession.
        - Impacto en negocio:
            - Aprobación de "n" operaciones indebidas por tiempo de vida de la sesión.
    - Mitigación
        - Las APIs del ResourceServer (pagos y etc.) no deben ser públicas. La conexión entre la billetera y el administrador tiene que ser 1:1 y la conexión entre el administrador y el proveedor de cuenta igual. El administrador siempre debe estar entre la billetera y el proveedor de cuenta.

3. **Interceptación y/o robo de refresh token**
    - Impacto
        - Tener el leak del secret y permitir el uso indebido del refresh token.
        - Impacto en negocio:
            - Aprobación de "n" operaciones indebidas por tiempo de vida de la sesión.
    - Mitigación
        - Rotar refresh token en cada uso.
        - Nota: Complejidad en la solución. La dificultad para sincronizar puede tener un impacto negativo en la experiencia del usuario, las billeteras

deben estar preparadas para salvar el nuevo refresh token cuando sea rotacional. Tener una regla de expiración del refresh token. Las billeteras tienen que preparar el flujo para el caso donde el refresh token esté revocado/expirado.

4. **Adulteración de parámetros**
    - Impacto
        - Emitir autorización con datos y/o alcances diferentes a los solicitados originalmente.
    - Mitigación
        - Tener siempre los 3 scopes definidos dentro de este documento, y que el Authorization Server siempre permita emitir tokens únicamente con los scopes definidos dentro del documento. Para que la mitigación sea efectiva, tanto al momento de iniciar el proceso de autenticación contra el servidor de autorización, este último deberá validar los scopes permitidos de la misma forma que al momento de solicitar el access token invalidando la petición si esto no ocurre.

5. Normas de seguridad

a. Ceremonias P2P para el MVP

Para que ocurra el mTLS en las integraciones, es necesario **garantizar que el proceso se realizará con ceremonias P2P, donde va a ser hecha la configuración y validación en ambos lados, para permitir integraciones en el MVP**.

b. Estándar de certificado

i. Servidor

El certificado de servidor debe emitirse para proteger y autenticar el canal TLS utilizado por las API que consumirán las aplicaciones cliente de las entidades, o sea, la pantalla de consentimiento.

El certificado del servidor debe enviarse con la cadena intermedia según RFC 5246 ítem 7.4.2.

El certificado de servidor debe contener los siguientes atributos:

Distinguished Name

- organizationName (OID 2.5.4.10) [Opcional]
- commonName (OID 2.5.4.3): FQDN ou Wildcard [Opcional]
- serialNumber (OID 2.5.4.5: Nro. de serie): DEBE estar presente y DEBE contener el tipo y número de identificación del titular, expresado como texto y respetando el siguiente formato y codificación: "[tipo de documento]" "[nro. de documento]" [Obligatorio]

ii. Cliente

El Certificado de Cliente debe contener los siguientes atributos:

Distinguished Name

- serialNumber (OID 2.5.4.5: Nro. de serie): DEBE estar presente y DEBE contener el tipo y número de identificación del titular, expresado como texto y respetando el siguiente formato y codificación: "[tipo de documento]" "[nro. de documento]"

Req Extensions

- basicConstraints = CA:FALSE [Obligatorio]
- subjectAltName = <DNS NAME> [Obligatorio]
- keyUsage = critical,digitalSignature,keyEncipherment [Obligatorio]
- extendedKeyUsage: clientAuth [Obligatorio]

Todas las comunicaciones backend del flujo de generación de token, deben realizarse utilizando el certificado de cliente (mTLS).

Se aclara que ambos certificados detallados son a modo ilustrativo, queda a disposición de cada proveedor de cuenta definir los mismos.

En los ambientes productivos, ambos certificados Cliente y servidor deben ser emitidos por una entidad certificadora (CA) pública reconocida, no deberá permitirse certificados autofirmados.

En los ambientes de homologación, ambos certificados tienen la posibilidad de ser autofirmados.

c. Direccionamiento

El proceso de direccionamiento a la página del proveedor de cuenta para la autenticación del cliente se definen dos soluciones[1]:

- Solución transitoria: Utilizar un navegador externo, con posterior redireccionamiento a la billetera (deep linking), una vez concluido el proceso de enrolamiento de cuentas, garantizando el proceso embebido definido en el documento funcional.

- Solución definitiva: Utilizar Custom Tabs / SF SafariViewController. Solo en dispositivos con versiones del sistema operativo que no soporten esta solución, se realizará la apertura de un navegador externo con deep linking a la billetera una vez concluido el proceso de enrolamiento de cuentas.

6. Flujos de OAUTH2+PKCE

- El método a utilizar es OAuth 2.0, Grant type Authorization Code Flow + PCKE + mTLS (RFC 6749 y RFC 7636).
- El **client_id** a generar identificará a la billetera en la que se enrolará la cuenta.
- Los scopes propuestos inicialmente a tener son: "openid", "offline_access", "accounts.debit".
- El servicio de autenticación enviará un access token y refresh token con el formato JSON Web Token estándar (RFC 7519 y RFC 9068).

[1] Para la salida a producción de la primera fase (entidades que se encuentran dentro del Registro de Billeteras Digitales Interoperables del BCRA), las entidades como mínimo deberán haber implementado y probado la solución transitoria, en caso de no llegar con los plazos para implementar la solución definitiva. Para fases siguientes de implementación, las entidades deben tener implementado para su salida a producción la solución definitiva (Custom Tabs / SF SafariViewController).

- Cada proveedor de cuenta sólo podrá mantener vigente un único refresh token para un mismo momento del tiempo para la relación proveedor de cuenta origen, cliente, cuentas consentidas y entidad enrolante.

Seguridad en el flujo de autorización de operaciones sobre la cuenta

- Para esta primera salida, se propone que el administrador autorice las operaciones antes de enviarlas al proveedor de cuentas para que sean ejecutadas. Esto nos permite reducir el impacto en la transaccionalidad sin afectar a los mecanismos offline ni mensajería host.
    - Para ello, el JWT no debe ser opaco, sino que debe seguir el formato JSON Web Token estándar (RFC 7519 y RFC 9068), y los administradores deben tener las herramientas para su validación y verificación bien definidas.

- Para la autorización de la operación, el administrador (en alto nivel):
    - Validará el JWT, chequeando que está bien formado, que fue creado por el proveedor de la cuenta (vía el chequeo de su firma), entre otras cosas definidas en RFC 7519 y RFC 9068.
    - Verificará el contenido del JWT, chequeando que no esté expirado, que el scope es correcto, que el client es correcto, que la audiencia es correcta, entre otras cosas definidas en RFC 7519 y RFC 9068.

- Detalles de claims en JWT

  Los JWT están compuestos por tres partes: header, payload y signature.

  Según lo descrito en el siguiente link, RFC 7519: JSON Web Token (JWT), y acordado en las mesas de trabajo en el ámbito de la CIMPRA, los claims que compondrán el payload del access token son los siguientes[2]:

[2] Los claims descritos en este documento están sujetos a futuras actualizaciones de los servidores de OAuth.

- **iss** (issuer): /identifica a quien generó el token/ será el código de la entidad proveedora de la cuenta – son de 5 dígitos (los bancos tienen dos ceros a la izquierda). Nómina de entidades , Registro de proveedores de servicios de pago
    - **iss_bcra_id (recomendado)**: será el código de la entidad proveedora de la cuenta – son de 5 dígitos (los bancos tienen dos ceros a la izquierda). Nómina de entidades , Registro de proveedores de servicios de pago.
- **sub** (subject): /identifica al usuario – de la billetera y de la cuenta a enrolar/ será el CUIT/CUIL del dueño de la cuenta. CUIT/CUIL sin guiones.
    - **user_cuit (recomendado)**: /identifica al usuario – de la billetera y de la cuenta a enrolar/ será el CUIT/CUIL del dueño de la cuenta. CUIT/CUIL sin guiones.
- **aud** (audience): /identifica para quien se genera ese token JWT/ para EEFF será también el código de la entidad, pero en este caso, dicha entidad será la entidad enrolante. En caso de PSP, será el código de PSP en el registro del BCRA – son de 5 dígitos (los bancos tienen dos ceros a la izquierda). Nómina de entidades , Registro de proveedores de servicios de pago.
    - **aud_bcra_id (recomendado):** /identifica para quien se genera ese token JWT/ para EEFF será también el código de la entidad, pero en este caso, dicha entidad será la entidad enrolante. En caso de PSP, será el código de PSP en el registro del BCRA – son de 5 dígitos (los bancos tienen dos ceros a la izquierda). Nómina de entidades , Registro de proveedores de servicios de pago.
- **exp** (expiration time): /duración del token/ parte entera del número de segundos transcurridos desde el 01/01/1970 hasta el momento en el cual expira el token generado, la diferencia entre este valor y el valor del claim iat no debe superar las 3 horas.
- **nbf** (not before time): /desde cuándo se puede usar el token/ se analizó este punto y se decidió no agregar este claim obligatoriamente, dado que en este caso no tiene relevancia. [Opcional]
- **iat** (issued at time): /momento en que fue emitido el token/ parte entera del número de segundos transcurridos desde el 01/01/1970 hasta el momento en el cual se generó el token.

- **scope**: /indica qué operaciones puede hacer aquel a quien se emita el token/ deben figurar openid[3], offline_access y accounts.debit. Van separados por espacios dentro de un string.
- **accounts:** claim específico /indica contra qué puede operar aquel a quien se emita el token/ se informarán las cuentas enroladas de la entidad proveedora, se propone utilizar la clave única de cada cuenta a las cuales el usuario dio su consentimiento. En caso de informarse más de una cuenta, se realizará en un array compuesto por la totalidad de CBU/CVU consentidos (ej: {cbu1, cbu2, cbuN}).
- **jti** (JWT ID): es un id para el token que el proveedor de la cuenta puede bloquear. Es opcional para el estándar de OAuth (links: JSON Web Token (JWT); JSON Web Token Claims).
- **trace_id**: /claim utilizado para contar con una trazabilidad de las operaciones/ longitud: 16 posiciones, alfanumérico.

A la fecha de emisión de este documento existen participantes cuya versión de OAuth permite realizar modificaciones en los claims específicos de este documento y otros que no se encuentran frente a esta restricción. Para solucionar este inconveniente, los administradores desarrollarán un **"if"** en su esquema de validación, el cual permitirá operar con ambas versiones en simultáneo.

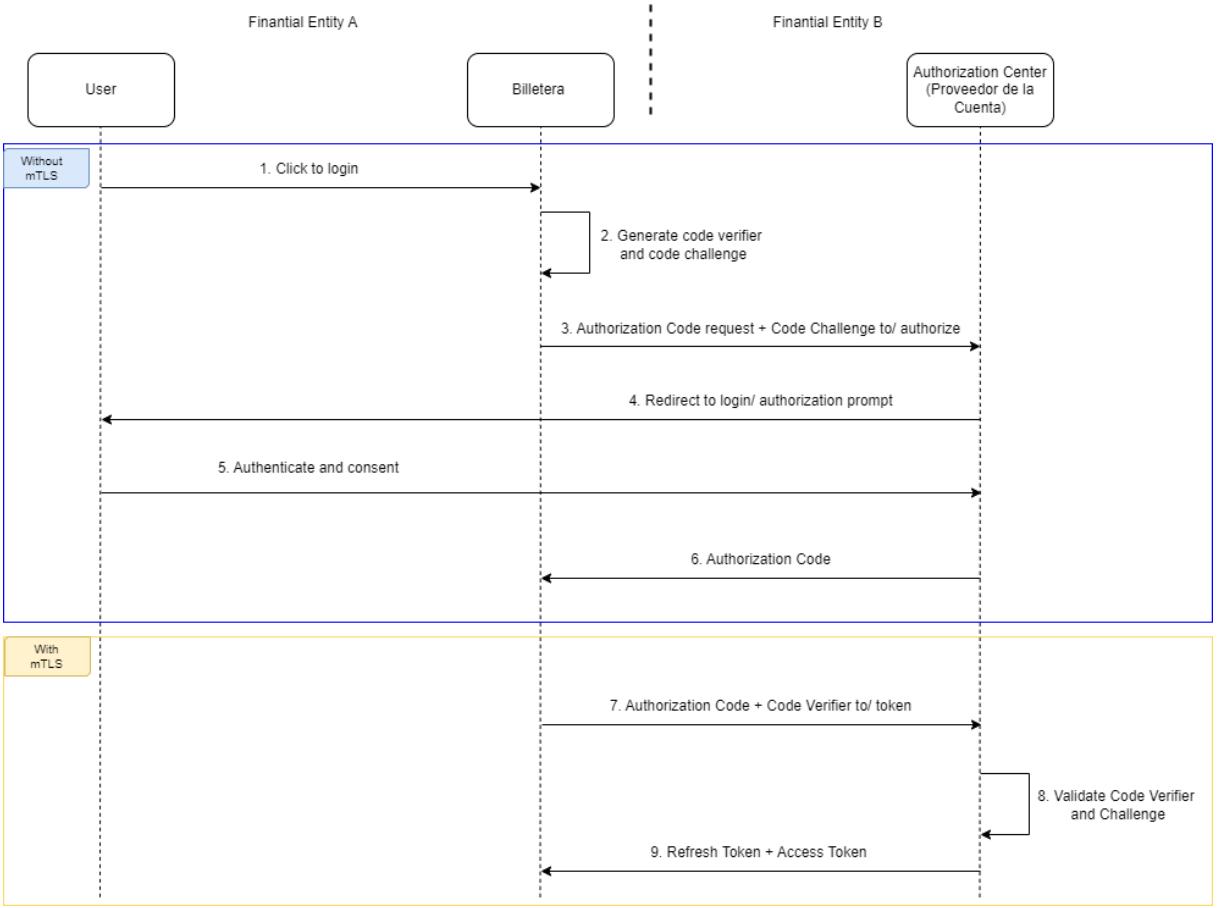
Por lo tanto, se estableció un período de convivencia de 6 meses desde la salida a producción de transferencias pull, en el cual la inclusión de los claims identificados como "recomendados" será opcional, para aquellos participantes cuya versión de OAuth no sea restrictiva frente dichos cambios. Para quienes hoy no puedan realizar modificaciones, la inclusión de estos claims será obligatoria, y podrán operar debido al **"if"** que utilizarán los administradores, el cual validará primero la existencia de estos claims específicos, y en caso de no figurar tomará el dato anterior.

Concluido el período de convivencia, los claims que figuran en este documento como recomendados pasarán a ser de inclusión obligatoria. Y en caso de no ser incluidos, los tokens generados serán inválidos.

[3] Permite obtener un perfil básico del cliente. Es un scope obligatorio, pero que no deberá exponer UserInfo ya que podría tener implicancias en la Ley de Privacidad de Datos Personales. Dado que para compartir información del cliente, se necesitaría un consentimiento previo, en esta instancia no se podrá utilizar de manera productiva. OpenID Connect Core 1.0

Apéndice

7. Flujo de OAuth 2.0 Authorization code + PKCE



Paso a Paso:

1. El usuario hace click en iniciar sesión dentro de la aplicación e inicia el proceso de enrolamiento de cuentas.
2. La billetera crea un code_verifier criptográficamente aleatorio y, a partir de esto, genera un code_challenge. En la solicitud de autorización se debe especificar el parámetro code_challenge_method=S256; si no se especifica el parámetro se lo tomará como plain y será rechazado.
3. La billetera redirige al usuario al servidor de autorización (del proveedor de la cuenta) junto con el code_challenge.*
4. El proveedor de la cuenta redirige al usuario al inicio de sesión y autorización.
5. El usuario se autentica usando una de las opciones de inicio de sesión configuradas y puede ver una página de consentimiento que enumera los permisos que le otorgará a la aplicación.
6. El proveedor de cuenta almacena el code_challenge y redirige al usuario a la aplicación con un código de autorización, que es válido para un solo uso.

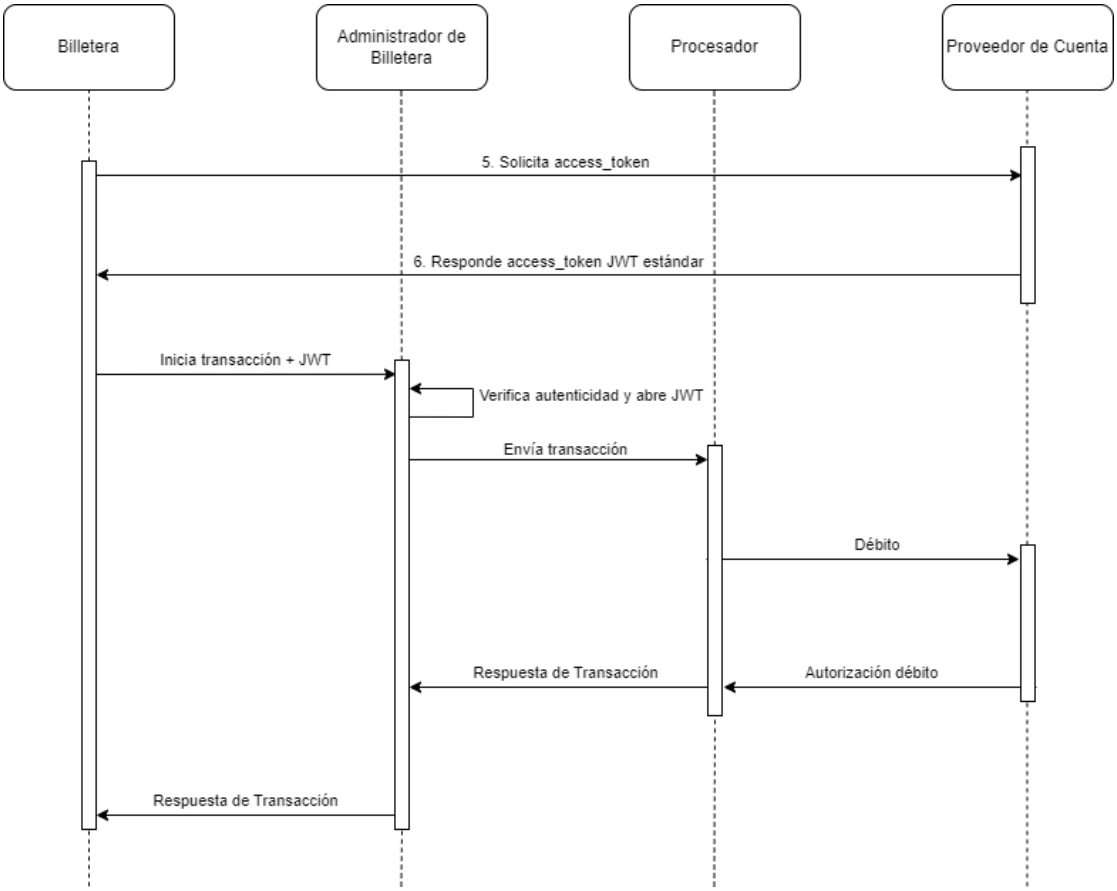
7. La billetera envía este código de autorización y el code_verifier (creado en el paso 2) al proveedor de la cuenta.
8. El proveedor de cuenta valida el code_challenge y code_verifier.*
9. El proveedor de cuenta responde a la billetera con un access token, el cual contiene un tiempo de expiración y un refresh token.

(*) Aclaraciones:
Se deberá incluir en el paso 3, de manera obligatoria para la billetera, el user_identifier (CUIT del usuario) que viajará como un parámetro más.
En el paso 8 se sugiere que el proveedor de cuenta valide también que el CUIT que viajó en el user_identifier sea igual al de quien está ingresando sus credenciales.

Recursos útiles:

- Authorization Code Flow with Proof Key for Code Exchange (PKCE)
- Estándares OAuth 2.0

10. Flujo transaccional



Paso a paso:

1. Billetera solicita access token por medio del refresh antes de iniciar la transacción (paso "5").
2. Billetera recibe access token "fresco" y listo para ser utilizado (paso "6").
3. Billetera inicia la transacción y envía el access token "fresco" a la operación junto a la información de la transacción (el access token se envía en el header authorization (RFC 6750)).
4. El administrador valida y verifica la autenticidad de la operación en base al access token recibido.
    i. Si el JWT es válido, se envía la transacción al procesador. El procesador envía la operación de Débito al Proveedor de cuenta, que luego envía el resultado al administrador, y éste a la billetera.
    ii. Si el JWT no es válido, se rechaza la operación y se envía esa información a la billetera. El mensaje de error en este caso es 401 unauthorized.
5. El procesador envía el débito y recibe la autorización.

6. Flujo de desvinculación

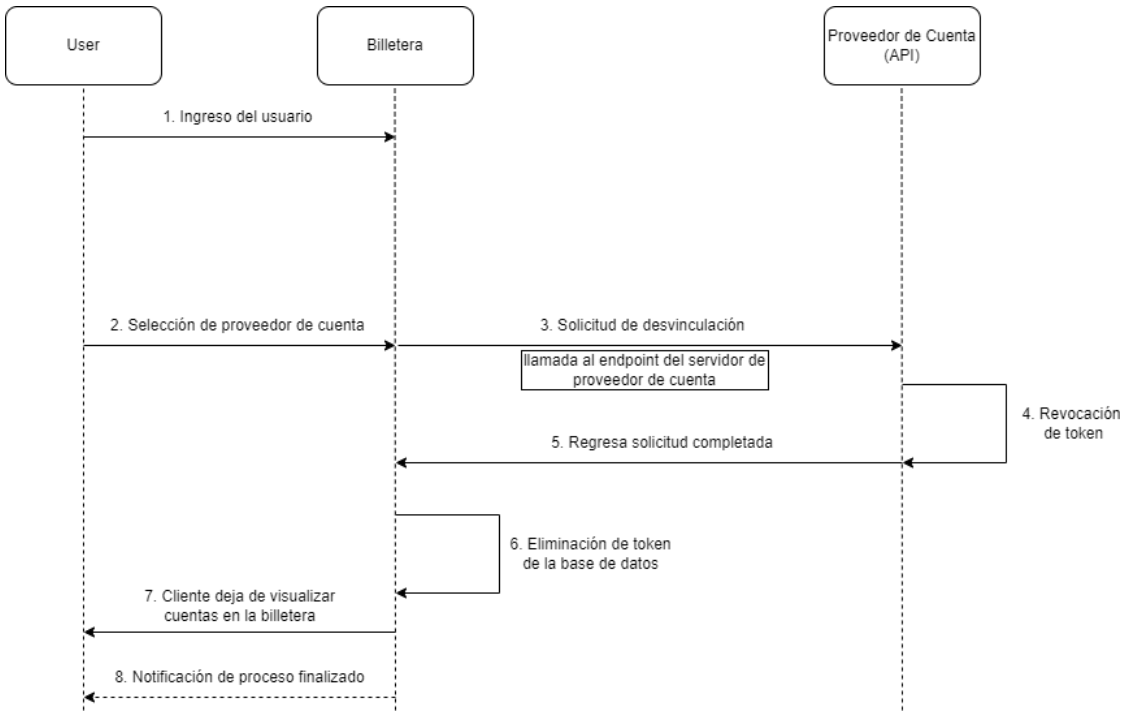
La desvinculación de las cuentas se puede generar a través de la misma billetera o desde un canal del proveedor de cuenta.

Independientemente de dónde se inicie el proceso de desvinculación, al momento de efectivizar la revocación, se dan de baja todas aquellas cuentas enroladas del proveedor de cuenta en la billetera.

En caso de no haber enrolado la totalidad de las cuentas y, que posteriormente el cliente desee enrolar una cuenta nueva del mismo proveedor, se deberá revocar el refresh token vigente de las cuentas previamente enroladas y realizar el enrolamiento con consentimiento nuevamente.

Para el proceso de desvinculación se solicitarán los datos asociados de "client_id", "client_secret" y "refresh_token". Serán eliminados el **"refresh token"** y el **"access token"**. La conexión back end to back end estará siempre protegida por mTLS.

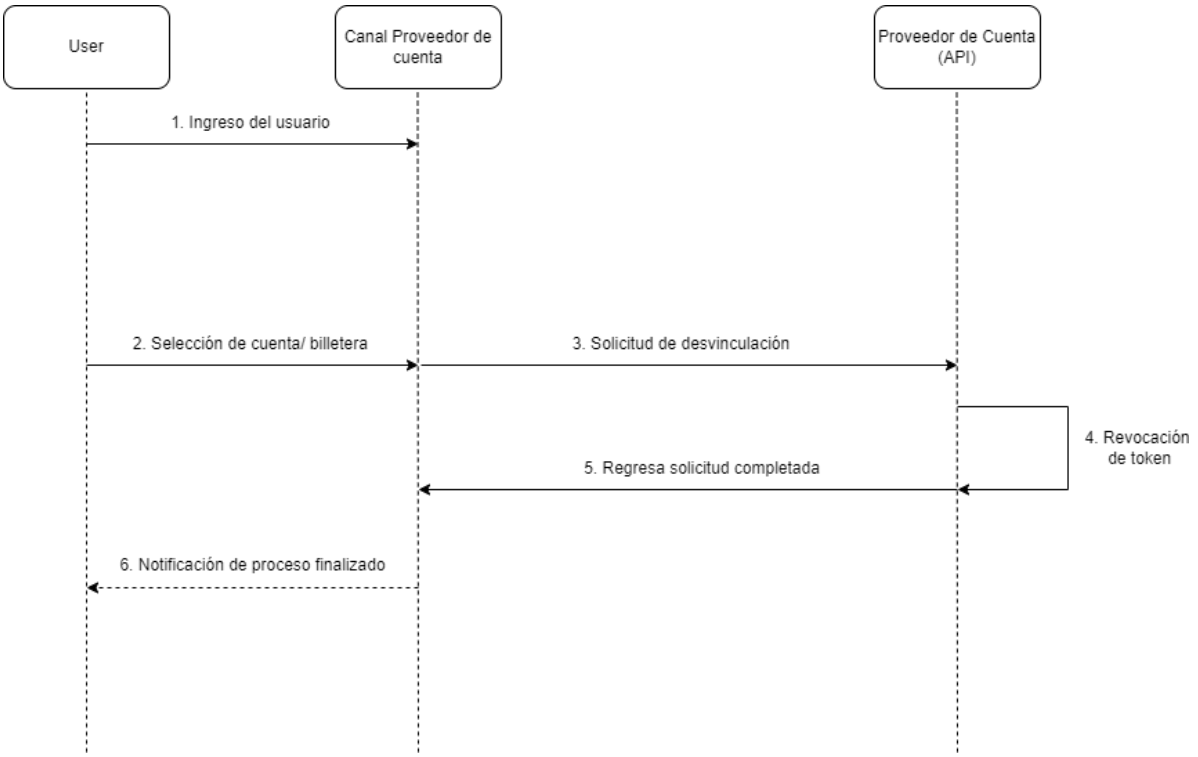
i. Flujo de desvinculación por billetera



Paso a paso:

1. El usuario ingresa a su aplicación de billetera.
2. El usuario selecciona al proveedor de cuenta que desea desvincular y se encuentre enrolado en la billetera.
3. La billetera genera la solicitud de desvinculación dirigida a la API provista por el proveedor de cuenta (ver API de revoke token en anexo).
4. La API del proveedor de cuenta revoca refresh token asociado al client_id y al usuario.
5. La API del proveedor de cuenta informa solicitud de desvinculación completada de forma exitosa a la billetera.
6. La billetera elimina el refresh token y el access token de su base de datos y se desvincula la cuenta.
7. El usuario deja de visualizar las cuentas desvinculadas en la billetera.
8. Notificación al usuario de proceso finalizado con éxito.

i. Flujo de desvinculación desde un canal del proveedor de cuenta



Paso a paso:

1. El usuario ingresa al canal del proveedor de cuenta.
2. El usuario selecciona qué cuenta/billetera desea desvincular.
3. Se genera la solicitud de desvinculación dirigida a proveedor de cuenta, desde el canal alternativo.
4. La API del proveedor de cuenta revoca el refresh token y access token asociado al client_id.
5. La API del proveedor de cuenta informa solicitud de desvinculación completada de forma exitosa al canal alternativo.
6. Notificación al usuario de proceso finalizado con éxito.

Aclaración: Este proceso no contempla la notificación de la baja a la billetera. Esta última deberá interpretar la desvinculación de la cuenta cuando solicite un nuevo access token y el refresh token sea rechazado. El mensaje de error en este caso es **401 unauthorized**. Para poder operar, la billetera deberá enrolar nuevamente las cuentas.

7. Régimen de mantenimiento y actualización del documento

Para garantizar la eficacia permanente del proceso de consentimiento y servicios transaccionales, el presente documento debe mantenerse actualizado por medio de revisiones periódicas.

Los participantes serán responsables de detectar posibles cambios en los estándares de las herramientas informáticas en uso, a la par de notificarlos al equipo de medios de pago del BCRA.

Posteriormente, el BCRA se encargará de coordinar una mesa de trabajo en el ámbito de la CIMPRA donde se plantee y dialogue la identificación de cambios en el proceso con los participantes, a fin de definir un plan de adecuación.

Este régimen de mantenimiento y actualización debe procurar que se distribuya el documento actualizado a todos los participantes del ecosistema, detallando las modificaciones realizadas junto con las fechas de implementación y vigencia.

8. Referencias normativas

[RFC2818] - HTTP Over TLS: https://datatracker.ietf.org/doc/html/rfc2818

[RFC5246] - The Transport Layer Security (TLS) Protocol Version 1.2 https://www.rfc-editor.org/rfc/rfc5246.txt

9. Anexos

Anexo I – APIs

a. Alta

Login prompt

```
/authorize
    response_type=code&
    code_challenge=CODE_CHALLENGE&
    code_challenge_method=S256&
    user_identifier=CUIT del usuario
    client_id=YOUR_CLIENT_ID&
    redirect_uri=YOUR_CALLBACK_URL&
    scope=SCOPE&
    state=STATE
```

`response_type`	“**code**” Indica el tipo de credencial que devolverá en nuestro caso code.
`code_challenge`	Desafío generado desde code_verifier.
`code_challenge_method`	Método utilizado para generar el desafío (p. ej., S256). La especificación PKCE define dos métodos, S256 y plain. Se utilizará S256.
`user_identifier`	CUIT del usuario logueado en el cliente que inició el flujo, se debe incluir de manera obligatoria por parte de la billetera.Se sugiere que el proveedor de cuenta valide que el CUIT que viajó en este claim sea igual al de quien está ingresando sus credenciales.
`client_id`	Id generado para la billetera que estará consumiendo (Código de la entidad BCRA).
`redirect_uri`	La URL a la que se deberá redirigir el navegador después de que el usuario haya otorgado la autorización. El código de autorización estará disponible en el parámetro code. Se deberá registrar la URL de callback permitida en el servidor.
`scope`	“openid offline_access accounts.debit” será el único scope habilitado para operar.
`state`	Alfanumérico arbitrario generado por el cliente para prevenir ataques CSRF, deberá enviarse en la redirect uri parámetro state.

Response:
Inicio del flujo de consentimiento en el proveedor de cuenta (Login)

Flujo completado

Response:

En caso exitoso, recibirá una respuesta HTTP 302. El código de autorización se incluye al final de la URL (`redirect_uri`)

```
HTTP/1.1 302 Found
Location: CALLBACK_URL?code=AUTHORIZATION_CODE&state=xyzABC123
```

b. Intercambio de code por tokens

Intercambio authorization_code por tokens

```
/token
curl --request POST \
  --url '/token' \
  --header 'content-type: application/x-www-form-urlencoded' \
  --data grant_type=authorization_code \
  --data 'client_id=YOUR_CLIENT_ID' \
  --data 'client_secret=YOUR_CLIENT_SECRET' \

  --data code_verifier=YOUR_GENERATED_CODE_VERIFIER \
  --data code=YOUR_AUTHORIZATION_CODE \
  —-data user_identifier \
  --data 'redirect_uri=https://YOUR_APP/callback'
```

`grant_type`	authorization_code
`code_verifier`	Clave criptográfica para verificar el code challenge que se envía en el primer paso.
`code`	authorization_code
`client_id`	Id generado para la billetera que estará consumiendo (Código de la entidad BCRA).
`client_secret`	Clave de la billetera dentro de OAuth.
`redirect_uri`	URL que se envió en el primer paso.
`user_identifier`	CUIT del usuario logueado en el cliente que inició el flujo, se debe validar si el mismo coincide con el usuario a quien se emitió el token.

Response

```
{
  "access_token":"eyJz93a...k4laUWw",
  "refresh_token":"GEbRxBN...edjnXbL",
  "token_type":"Bearer",
  "expires_in":10800,
}
```

`access_token`	access_token JWT
`refresh_token`	Para solicitar un nuevo access_token .
`token_type`	Tipo del token Bearer.
`expires_in`	Tiempo de duración del token expresado en segundos.

Intercambio de authorization code por tokens

API `/token`

```
curl --request POST \
 --url '/token' \
 --header 'content-type: application/x-www-form-urlencoded' \
 --data grant_type=refresh_token \
 --data 'client_id=YOUR_CLIENT_ID' \
 --data 'client_secret=YOUR_CLIENT_SECRET' \
 --data refresh_token=YOUR_REFRESH_TOKEN
```

grant_type	refresh_token
client_id	Id generado para la billetera que estará consumiendo (Código de la entidad BCRA)
client_secret	Clave de la billetera dentro de oAuth.
refresh_token	Refresh token del usuario que está solicitando un nuevo juego de access_token&refresh_token.

Response

```
{
  "access_token":"eyJz93a...k4laUWw",
  "refresh_token":"GEbRxBN...edjnXbL",
  "token_type":"Bearer",
  "expires_in":10800
}
```

access_token	access_token JWT
refresh_token	Nuevo refresh token (opcional).
token_type	Tipo del token Bearer.
expires_in	Tiempo de duración del token expresado en segundos.

c. Baja / Desvinculación

Revocación

```
curl --request POST \
  --url '/oauth/revoke' \
  --header 'content-type: application/x-www-form-urlencoded' \
  --data '{ "client_id": "YOUR_CLIENT_ID", "client_secret":
"YOUR_CLIENT_SECRET", "token": "YOUR_REFRESH_TOKEN" }'
```

`client_id`	Id generado para la billetera que estará consumiendo (Código de la entidad BCRA).
`client_secret`	Clave de la billetera dentro de OAuth.
`token`	Refresh token a revocar.
`token_type_hint`	Tipo de bearer token. Valor: refresh_token. Su uso es opcional.

Anexo II – Proceso de gestión de certificados para consentimiento

Este proceso aplica para el intercambio de credenciales y URLs de los endpoints para el proceso de consentimiento y enrolamiento de cuentas.

El proceso de gestión de certificados para el ambiente de homologación y para el de producción son equivalentes. A la fecha de emisión del presente documento se ha logrado definir únicamente un proceso manual, por lo cual se ha determinado avanzar con el mismo a modo de proceso transitorio, hasta que sea implementada una solución automática.

Se aclara que las credenciales serán intercambiadas entre proveedor de cuenta y entidad enrolante, pudiendo ser delegada en un tercero la entrega del token si la entidad proveedora de la cuenta así lo decida.

a. Proceso transitorio (vigencia hasta puesta en producción solución automática).

Al momento de realizar la conexión, ya sea en rol de proveedor de cuenta o billetera digital, la gestión de los certificados será canalizada a través de los administradores, mediante las casillas de email de integraciones definidas.

Los administradores velarán por el ecosistema, facilitando el primer contacto entre las partes (entidad enrolante y proveedor de cuenta), para que luego se compartan los certificados entre sí.

Se aclara que el administrador no gestionará certificados de terceros, únicamente será un facilitador en el proceso de conexión entre las partes.

b. Proceso definitivo de gestión de certificados para consentimiento (pendiente de definición)

Se evalúa la viabilidad de desarrollar un proceso automático de gestión de certificados para consentimiento.

A la fecha de salida del presente documento se encuentra pendiente de definición su implementación.

c. Canales de contacto con los administradores

Las entidades se deberán comunicar con su administrador; a continuación, se detallan los canales de contacto de cada uno:

Coelsa: integraciones@coelsa.com.ar

Link: T3.0-certificaciones@redlink.com.ar

Prisma: newpay-homologaciones@prisma.com

Anexo III – Llaves públicas

Cada administrador informará a los participantes como deberán compartir las llaves públicas.

Formato de llave pública a compartir:
Las llaves deberán ser del tipo RSA 2048 bits. El formato que se deberá cargar es del tipo PEM X.509 notación ASN.1 que sigue el estándar PKCS#1. Se firmará utilizando el algoritmo RS256 descrito en RFC 7518.