IDLAC Broker Modelo Regional

DOCUMENTACIÓN TÉCNICA
Parte I: Instalación, Opciones de despliegue e Integración

Versión	Modificación	Fecha	Modificado por:
1.0	Versión inicial	2025-09-04	Mariana Silvera
1.1	Incluye URL well-known de Testing y logout uri. Mapeo de Claims	2025-10-06	Mariana Silvera
1.2	Incluye documentación final Fase 1	2025-12-04	Mariana Silvera
1.3	Separa documentación técnica del broker del Modelo	2025-12-10	JP García

Documento en permanente evolución elaborado por el Grupo de trabajo de Red Gealc con apoyo del Consorcio Ciudadano Digital Regional (BID, Banco Mundial, CoDevelop y OEA)

1. Contents

2. Objetivo del documento

Este documento propone ser una guía para la implementación del broker modelo de identificaciones digitales de América Latina y El Caribe. En este documento se describen los aspectos técnicos del broker modelo, componente esencial para lograr una interoperabilidad transfronteriza estandarizada de identificaciones digitales según los lineamientos del modelo IdLAC.

Este documento complementa al documento "Modelo de Identificación Digital de América Latina y el Caribe (IdLAC)" donde se describe el modelo acordado por todos los países.

3. Introducción y Contexto

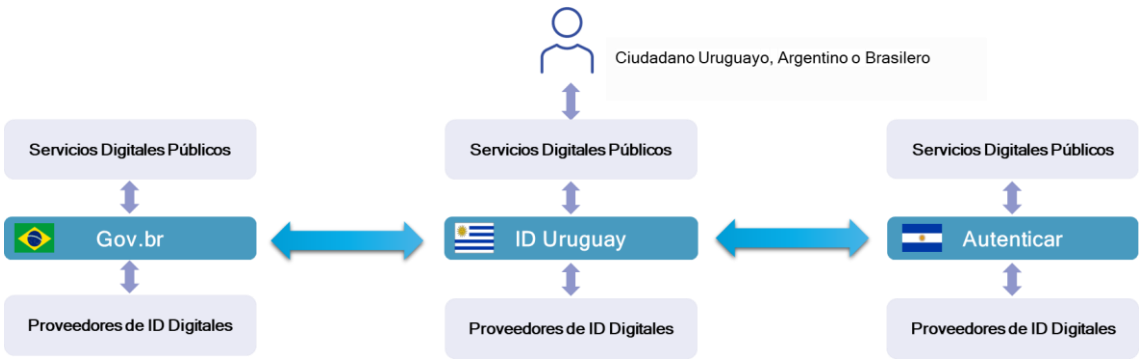
3.1. Broker de Identificaciones Digitales

En la era digital, la autenticación segura y eficiente de los usuarios en servicios en línea es crucial para el buen funcionamiento de las entidades públicas y privadas. En este contexto, la adopción de un broker de identificaciones digitales, se plantea como una solución eficaz para gestionar de manera centralizada y confiable distintos mecanismos de identificación, permitiendo que los usuarios accedan a múltiples servicios mediante una sola credencial de identificación.

Un broker de identificaciones digitales no solo facilita y fortalece la identificación o autenticación en servicios digitales, sino que también simplifica la operación de las instituciones, que ya no requieren desarrollar o mantener sus propios mecanismos de autenticación. Además, habilita un ecosistema público-privado de identificaciones digitales interoperables a nivel nacional.

En un contexto regional, la integración entre brokers de identificaciones digitales compatibles de diversos países habilita en forma simple, segura y estandarizada la interoperabilidad de identificaciones digitales. De esta forma, una persona que posea una identificación digital confiable en un país podrá utilizarla para acceder a servicios digitales de otros países integrados sin necesidad de poseer una credencial adicional en cada territorio .

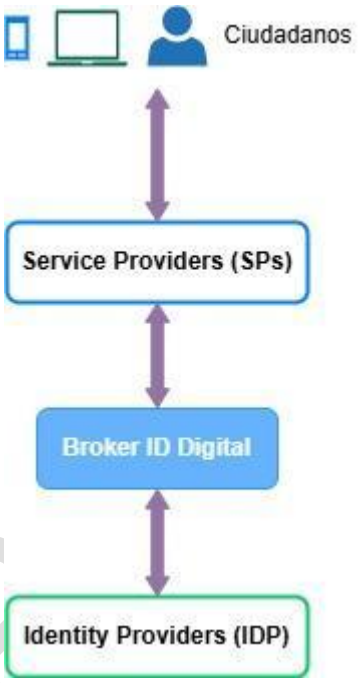
El siguiente diagrama ejemplifica el escenario de integración entre los brokers de identificación de Uruguay (ID Uruguay), Argentina (Autenticar) y Brasil (gov.br). En este esquema, una persona argentina o brasileña puede autenticarse en cualquier servicio digital uruguayo utilizando el prestador de identificación digital de su país a través de la plataforma ID Uruguay.



El objetivo de este proyecto es desarrollar y poner a disposición un bróker de identificaciones digitales modelo para ser implementado por los países de la región, evitando que cada país desarrolle el suyo, asegurando el uso de tecnologías, estándares y protocolos actualizados y alineados a EIDAS 2.0.

3.2. ID Broker

Un bróker de identificaciones digitales genera un ecosistema a nivel país que simplifica y fortalece el uso de identificaciones digitales. El siguiente esquema muestra cómo se posiciona un bróker de identificaciones digitales en el ecosistema:



+

Un Broker de identificaciones digitales es una plataforma tecnológica que se posiciona entre proveedores de identificaciones digitales (IDP, Identity Providers) aptos para el ecosistema y sistemas digitales (portales, aplicaciones móviles, sistemas de gestión, servicios en línea, tiendas, etc).

De esta forma, cuando un usuario ingresa a un sistema informático (SP) y necesita identificarse (autenticarse) a través del bróker accede a un conjunto de prestadores de identificación aptos, utiliza el que desee y, nuevamente a través del bróker, regresa al sistema informático.

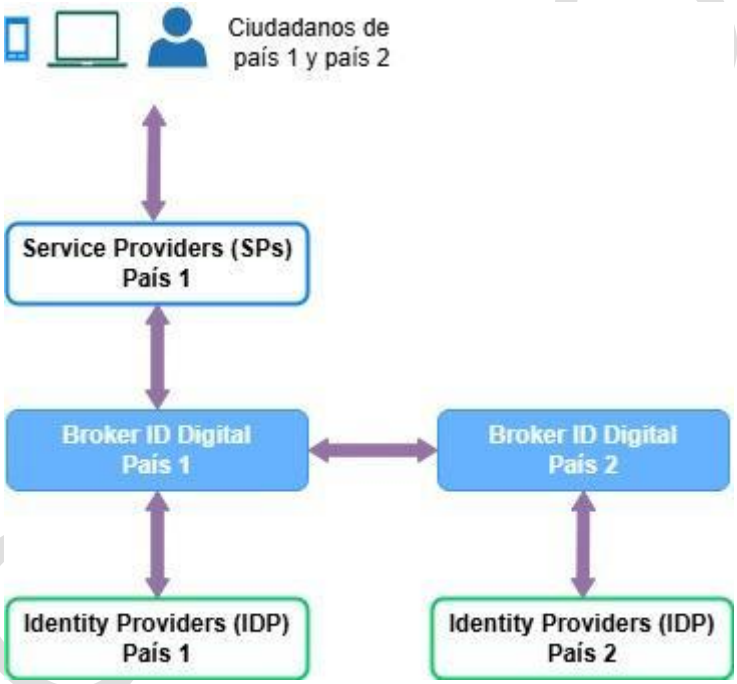
Es importante resaltar algunos aspectos fundamentales:

- El bróker debe actuar como un Single Sign On (SSO), es decir, una vez el usuario inicia sesión a través de un proveedor de identificación, la misma se debe mantener activa para todos los sistemas integrados al bróker (SPs) hasta que el usuario no cierre la sesión o se finalice luego de determinado tiempo definido por una política específica.
- La normativa asociada al bróker determina, entre otras cosas, las condiciones que deben cumplir los proveedores de identificación (IDP) para estar habilitados para el

ecosistema, en función de las necesidades y requerimientos que poseen los sistemas (SP) integrados al bróker.

- Los usuarios se autentican en su proveedor de identificación (IDP), las credenciales nunca salen del IDP.
- El ecosistema puede tener diferentes niveles de seguridad o confianza en las identificaciones digitales dependiendo de cómo el usuario se registró y validó su identificación y cómo se autenticó utilizando su IDP.

Con el fin de simplificar y realizar un paralelismo con las identificaciones físicas, se habla de identificación digital como un sinónimo de autenticación que, técnicamente se compone de identificación y verificación. El bróker genera un ecosistema de identificación digital a nivel país, el sector privado puede estar presente como IDP, pero también como SP. Adicionalmente la bróker habilita y facilita la integración regional, integrando dos brokers de diferentes países. En este caso, cada bróker se comporta como un ecosistema de IDPs integrados. El siguiente esquema muestra esta situación:



Integrando el bróker del país 1 con el del país 2 se crea la posibilidad de que un ciudadano del país 2 utiliza proveedores de identificación digital de su país para ingresar digitalmente al país 1 con los mismos niveles de confianza y seguridad.

3.3. Protocolos de Integración

Los protocolos están definidos en el documento del Modelo IdLAC, a continuación se detallan lo que tiene que ver con el broker modelo:

- La integración entre el broker los SPs, IDPs y otros brokes se va a resolver mediante el protocolo **OIDC (OpenID Connect).**

- En la versión 2 del broker se habilitará el protocolo OIDC4VP (OpenID Connect for Verifiable Presentations) para integrar IDPs - billeteras electrónicas con Credenciales Verificables de identificación digital.
- El bróker asegurará el inicio de sesión unificado **(SSO – Single Sign-On)** a partir de la autenticación en cada proveedor de identificación.
- Tanto los SPs como IdPs pueden ser sistemas web o aplicaciones móviles.
- Cuando un usuario cierra la sesión se debe cerrar en todos los SPs e IDPs que utilizó.

3.4. Niveles de Confianza para las Identificaciones Digitales

El detalle de los niveles de confianza en las identificaciones están definidos en el documento del Modelo IdLAC. A continuación, se presenta un resumen:

El nivel de seguridad (NID) es el mínimo entre dos variables:

1. **Nivel de Registro (RID)**, puede ser bajo (1), medio (2) o alto (3). Un registro 0 es un registro que no fue validado por el usuario, por lo que todavía no puede ser utilizado.
2. **Nivel de Autenticación (AE)**, puede ser bajo (1), medio (2) o alto (3). Todos los proveedores de identificación deben implementar políticas de contraseñas que aseguren que el usuario solamente puede utilizar contraseñas consideradas fuertes.

La siguiente tabla muestra un resumen de los niveles de seguridad (NID) en función del nivel de registro (RID) y el nivel de autenticación (AE):

Nivel de Seguridad (NID)		**Autenticación (AE)**			
		0 Pass débil	1 Pass Fuerte	2 Pass fuerte + 2FA	3 Certificado Digital o 2FA fuerte
Registro (RID)	0 No confirmado	-	-	-	-
	1 Correo o teléfono confirmado	-	Bajo	Bajo	-
	2 Identificación Validada	-	Bajo	Medio	-
	3 Validación biométrica y renovación	-	-	-	Alto

3.5. Requisitos Funcionales

- Mecanismo para configurar SPs e IDPS
- Gestión de perfiles de usuario administradores
- Auditoría de eventos:
    - Registro detallado de actividades de autenticación de usuarios que pasan a través del bróker.
    - Auditoría de la actividad de los usuarios administradores

4. Instalación de ID Broker Modelo

A continuación, se detallan los pasos a seguir y la información que se requiere intercambiar para poder dar de alta una instancia del ID Broker Modelo en un nuevo país.

4.1. Proceso de Instalación e integración

1. **Instalación del ID Broker Modelo en infraestructura propia del nuevo País**

    1.1. Equipo implantador se basa en el capítulo: Implantación de ID Broker en infraestructura propia
    1.2. Una vez instalado, se tienen definidas las URIs requeridas para poder integrar la nueva instancia del Broker con los SPs e IDPs:
        1.2.1. Well-known URL
        1.2.2. Redirect_uri
        1.2.3. Logout_uri

    Si se utilizará la plataforma de pruebas de integración, omitir este paso e ir directamente al paso 2.

2. **Integración de Proveedores de Identificación del país (IDPs)**

    2.1. Validar los prerrequisitos con el IDP.
    2.2. Los IDPs deben completar el Formulario de alta de IDP, indicando toda la información solicitada.
    2.3. ID Broker Modelo provee la información necesaria a través del mismo formulario de alta.
    2.4. Personalización de los IDPs a nivel de interfaz de usuario.

3. **Integración de Proveedores de Servicios del país (SPs)**

    3.1. Validar los prerrequisitos con el SP.
    3.2. Los SPs deben completar el Formulario de alta de SP, indicando toda la información solicitada.

3.3. ID Broker País entrega información necesaria a través del mismo formulario de alta.
3.3.1. Para cada SP el ID Broker Modelo otorga un client_id y un client_secret.

4. Personalización de la interfaz de usuario (logo y textos)

4.1. El equipo implantador personaliza el Logo y texto de la pantalla de selección de IDPs de acuerdo a la guía de estilos.

5. Validación y Pruebas

5.1. País realiza configuraciones y pruebas de conectividad entre su Broker y el SP e IDP del país.
5.2. El equipo implantador ejecutará las pruebas que crea necesarias para validar el correcto funcionamiento de cada integración.

5. Opciones de despliegue

5.1. Implantación de ID Broker en infraestructura propia

ID Broker es una solución basada en contenedores Docker, pudiendo ser instalada en infraestructura propia de los países de la región, ya sea Cloud u onPremise.

5.2. Docker Compose Entregable

Se crea este componente para que la instalación de las aplicaciones requeridas por el proyecto sea completamente agnóstica, por lo cual, todas las aplicación requeridas se encuentran contenerizadas y todo orquestado a través de un docker-compose (docker-compose-entregable.yml)

Pre-requisitos

Para asegurar el correcto funcionamiento se debe tener en cuenta:

- agregar las siguientes entradas en el archivo **hosts** de tu sistema
  - 127.0.0.1 localhost idplocal.com broker-lb.com
- Tener instalado:
  - Docker
  - Docker Compose

Aplicaciones Orquestadas

- postgres
- redis
- idplocal
- broker
- broker-lb
- ssp
- backoffice

Ejecución

Para iniciar el proyecto, se debe ejecutar el docker compose entregable desde la raíz del proyecto y de la siguiente manera:

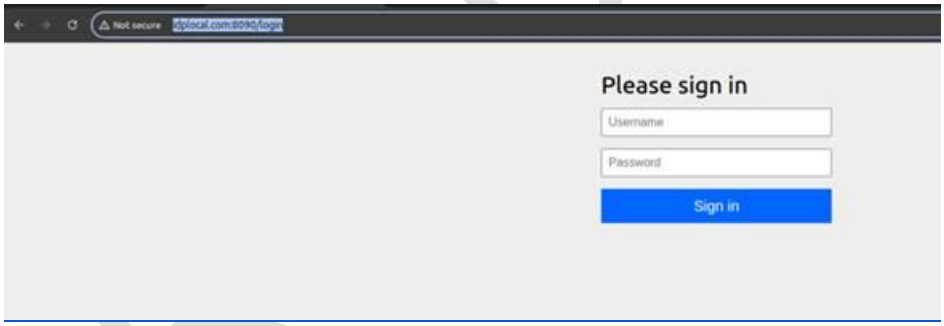
docker compose -f docker-compose-entregable.yml up --build --scale broker=n

donde n, es el número de instancias que se desean escalar del broker (debe ser mayor o igual a 1)

Direcciones clave del proyecto

Luego de la ejecución del proyecto, las siguientes dirección son clave para probar que todo haya levantado correctamente:

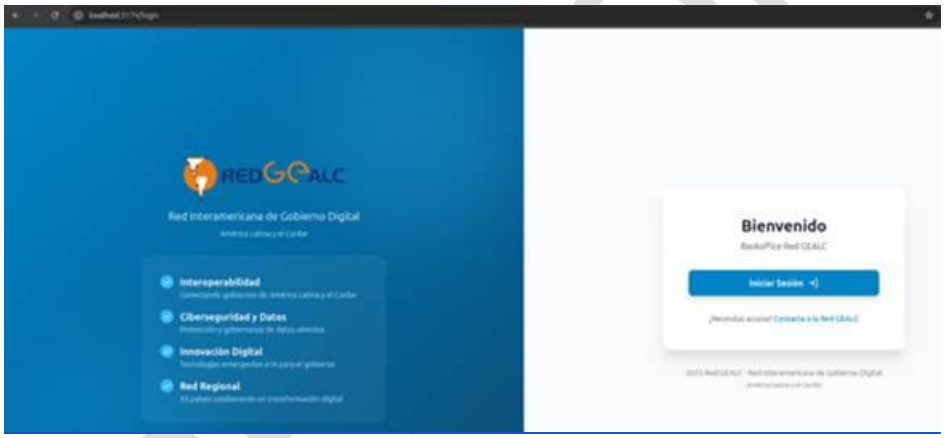
- idplocal
  - Usuario backoffice test: user_ae3_rid3
  - Pass backoffice test: pw_ae3_rid3
  - Usuario ssp test: user_ae0_rid0
  - Pass ssp test: pw_ae0_rid0
  - Url: http://idplocal.com:8090/login



- broker-lb
  - URL: https://broker-lb.com/idp-selection

- ssp
  - URL: http://localhost:5173/success

- backoffice
  - URL: http://localhost:5174/login



Detener

Para detener el proyecto, se debe detener el docker compose entregable desde la raíz del proyecto y de la siguiente manera:

docker compose -f docker-compose-entregable.yml down

5.3. Configuración Inicial de IdP, SP y Administrador para el Broker y Backoffice

Introducción

Para que el broker y el backoffice puedan utilizarse por primera vez, es necesario contar al menos con:

- Un primer Identity Provider (IdP) configurado
- Un primer Service Provider (SP) configurado
- Un primer usuario administrador del backoffice asociado al IdP local

En la instalación estándar se entregan scripts de ejemplo que crean un IdP de prueba, un SP de prueba y un administrador inicial. Estos datos permiten que el sistema arranque y que el backoffice sea accesible desde el primer día.

Se recomienda considerar estos registros como datos de ejemplo únicamente. Una vez que cada país u organización configure sus propios IdP, SP y administradores, se deben eliminar los registros de prueba para evitar que queden credenciales de laboratorio activas en ambientes reales.

Este documento explica qué información se necesita y cómo crear los primeros IdP, SP y administrador **como si no existieran scripts de ejemplo**, es decir, generando los scripts desde cero. Luego se puede comparar con los scripts entregados y adaptarlos.

El flujo de alta mediante backoffice ya está descrito de forma general en el apartado de "Alta, baja y modificación de IdP, SP y usuarios administradores", que se toma como base para esta guía.

Alcance

- Entender qué datos se requieren para el primer IdP, el primer SP y el primer administrador.
- Definir la información técnica y de organización que debe recopilar el equipo de integración.
- Proponer plantillas de scripts SQL para crear los registros iniciales en la base de datos.
- Explicar cómo validar que la configuración funciona y cómo continuar luego con el alta de nuevos IdP, SP y administradores desde el backoffice.

No se detalla aquí la instalación de la base de datos, ni el despliegue del broker o del backoffice, ni la configuración de red.

Precondiciones

Antes de crear el primer IdP, SP y administrador se asume:

- El esquema de base de datos del broker está creado.

- Se tiene acceso de escritura a la base (por ejemplo mediante un usuario técnico o herramienta de administración).
- Se conocen las URL del broker y de los sistemas externos que actuarán como IdP y SP.
- Se dispone de al menos un IdP OIDC externo funcional o de un IdP local de pruebas.
- Se conoce qué persona o equipo será el primer administrador del backoffice y qué identificador usará (claim `sub` emitido por el IdP).

Información necesaria para el primer Identity Provider (IdP)

El IdP representa un proveedor de identidad externo que autentica a los usuarios y emite tokens OIDC consumidos por el broker.

Datos de endpoints OIDC

Corresponden a los campos visibles en el formulario "Nuevo Proveedor de Identidad":

- **Issuer**
  - URL base del IdP, por ejemplo `https://idp.example.gov`
  - Debe coincidir exactamente con el valor `iss` presente en los tokens emitidos por el IdP.
- **Client ID**
  - Identificador del cliente registrado en el IdP para el broker.
  - Se obtiene al registrar el broker como aplicación OIDC en el IdP.
- **Client Secret**
  - Secreto asociado al client id anterior.
  - Debe protegerse como credencial sensible. Ver https://pyxisjira.atlassian.net/wiki/spaces/PTRID/pages/edit-v2/6403359009#Generar-client_secret-de-IdP-(AES-GCM)
- **Authorization Endpoint**
  - URL del endpoint de autorización del IdP, por ejemplo `https://idp.example.gov/oauth2/authorize`.
- **Token Endpoint**
  - URL del endpoint de token, por ejemplo `https://idp.example.gov/oauth2/token`.

- **Userinfo Endpoint**
  - URL del endpoint de userinfo, por ejemplo `https://idp.example.gov/userinfo`.
- **JWKS URI**
  - URL de las claves públicas del IdP, por ejemplo `https://idp.example.gov/oauth2/jwks`.
- **Scopes**
  - Lista de scopes mínimos que el broker necesita solicitar al IdP, por ejemplo `openid profile email`

Datos de organización del IdP

Tomados del bloque "Información de la Organización" del formulario:

- **Nombre de Organización**
  - Nombre descriptivo que verá el usuario final, por ejemplo `ExampleGov – Test`.
- **Identificador IdP**
  - Identificador único interno para el broker, por ejemplo `example-idp`.
  - Se utiliza en configuraciones y logs, por lo que conviene que sea estable y legible.
- **País (código ISO)**
  - Código de país ISO 3166 de dos letras, por ejemplo `UY`, `BR`, `EX`.
- **Contacto Técnico**
  - Correo de contacto para incidentes y coordinación técnica.
- **Descripción**
  - Texto libre que describe el IdP, su alcance o ambiente.
- **Nivel AE Máximo**
  - Máximo nivel de Aseguramiento de Entidad que este IdP puede declarar.
- **NID Mínimo**
  - Valor mínimo de NID que se aceptará desde este IdP.
- **Prioridad de Visualización**

- Número que define el orden en que se mostrará el IdP en la pantalla de selección del broker.
- Valores más bajos se suelen mostrar primero.

- **URL del Logo**
  - URL HTTPS al logo que se mostrará para este IdP.
- **Activo**
  - Indica si el IdP está disponible para autenticación.
- **IdP Extranjero**
  - Indica si el IdP pertenece a otro país u organización distinta de la que opera el broker.

Además, en la tabla `identity_providers` existen campos de auditoría y de configuración interna, como:

- `broker_registered_at`, `broker_updated_at`

Plantilla SQL para crear el primer IdP

Ajustar los valores entre corchetes a la realidad de cada entorno.

```
INSERT INTO public.identity_providers (
broker_active,
broker_display_priority,
broker_idp_country,
broker_max_ae_level,
broker_registered_at,
broker_updated_at,
broker_idp_type,
broker_idp_identifier,
authorization_endpoint,
broker_logo_url,
issuer,
jwks_uri,
token_endpoint,
userinfo_endpoint,
client_secret,
```

```
scopes,
broker_description,
broker_organization_name,
broker_technical_contact,
client_id,
broker_foreign_idp,
broker_minimum_nid,
broker_idp_wellknown
) VALUES (
true,
[PRIORIDAD_VISUALIZACION],
'[PAIS_ISO]',
[NIVEL_AE_MAXIMO],
NOW(),
NOW(),
'GOVERNMENT',
'[IDENTIFICADOR_IDP]',
'[AUTHORIZATION_ENDPOINT]',
'[URL_LOGO]',
'[ISSUER]',
'[JWKS_URI]',
'[TOKEN_ENDPOINT]',
'[USERINFO_ENDPOINT]',
'[CLIENT_SECRET]',
'[SCOPES]',
'[DESCRIPCION]',
'[NOMBRE_ORGANIZACION]',
'[CONTACTO_TECNICO]',
'[CLIENT_ID]',
[ES_IDP_EXTRANJERO_FALSE_O_TRUE],
[NID_MINIMO],
'[URL_WELLKNOWN]'
```

);

Información necesaria para el primer Service Provider (SP)

El SP representa una aplicación cliente que delega autenticación en el broker.

Datos técnicos del SP

Campos visibles en el formulario "Nuevo Proveedor":

- **Client ID**
  - Identificador único del SP, por ejemplo `sp-client-int`.
- **Client Secret**
  - Secreto del SP frente al broker.
  - Debe almacenarse cifrado mediante BCrypt en la base. Ver: https://pyxisjira.atlassian.net/wiki/spaces/PTRID/pages/edit-v2/6403359009#Generar-client_secret-de-SP-(BCrypt)
- **Nombre del Cliente**
  - Nombre descriptivo que verá el usuario durante el flujo de consentimiento.
- **Métodos de Autenticación**
  - Lista de métodos separados por espacio, por ejemplo `client_secret_basic client_secret_post`.
- **Grant Types**
  - Tipos de grant habilitados, típicamente `authorization_code refresh_token`.
- **Scopes**
  - Scopes que el SP puede solicitar al broker, por ejemplo `openid profile email document address phone auth_info nid`.
- **Redirect URIs**
  - Lista JSON con las URL de redirección válidas, por ejemplo `["https://sp.example.gov/callback"]`.
- **Post Logout Redirect URIs**
  - Lista JSON con URL adonde regresar al usuario después del cierre de sesión.

Datos de organización del SP

- **Nombre de Organización**

- Entidad responsable del servicio, por ejemplo `Ministerio de Pruebas Digitales`.

- **ID de Organización**
  - Identificador legible y único, por ejemplo `MIN-PRUEBAS-001`.
- **País**
  - Código ISO de país, por ejemplo `UY`.
- **Contacto Técnico**
  - Correo para incidencias y coordinación.
- **Activo**
  - Indica si el SP puede utilizar el broker.
- **Es Backoffice**
  - Marca si el SP corresponde al propio backoffice.
  - En la mayoría de los SP de negocio este valor será falso y solo deberá de haber un SP marcado como backoffice al mismo tiempo.

En la tabla `service_provider` existen además campos internos:

- `broker_service_description`
- `broker_requires_explicit_consent`
- `broker_default_consent_duration_days`
- `broker_registered_at` y campos de auditoría.

Plantilla SQL para crear el primer SP

```
INSERT INTO public.service_provider (
client_id,
client_id_issued_at,
client_secret,
client_name,
client_authentication_methods,
authorization_grant_types,
redirect_uris,
post_logout_redirect_uris,
scopes,
```

```
broker_organization_name,
broker_organization_id,
broker_organization_country,
broker_technical_contact,
broker_service_description,
broker_active,
broker_requires_explicit_consent,
broker_default_consent_duration_days,
broker_registered_at
) VALUES (
'[CLIENT_ID_SP]',
NOW(),
'[CLIENT_SECRET_BCRYPT]',
'[NOMBRE_CLIENTE]',
'client_secret_basic client_secret_post',
'authorization_code refresh_token',
'["[URL_REDIRECT_1]", "[URL_REDIRECT_2]"]',
'["[URL_POST_LOGOUT_1]"]',
'[SCOPES]',
'[NOMBRE_ORGANIZACION]',
'[ID_ORGANIZACION]',
'[PAIS_ISO]',
'[CONTACTO_TECNICO]',
'[DESCRIPCION_SERVICIO]',
true,
false,
365,
NOW()
);
```


Información necesaria para el primer usuario administrador

El backoffice utiliza usuarios administrativos que se identifican por el claim `sub` emitido por el IdP local.

Datos requeridos

- **Sub**
  - Valor exacto del claim `sub` del usuario que será administrador.
  - Se puede obtener realizando un login de prueba en el IdP y observando el token.
  - Ejemplo de formato: `CR-ID-CRI330016`.

Además de la tabla de usuarios, existe una tabla de asociación con roles. Para el primer administrador se asigna el rol `ADMIN`.

Plantilla SQL para crear el primer administrador

```
-- Crear usuario administrador
INSERT INTO backoffice_users (sub, created_at, updated_at)
VALUES ('[SUB_ADMIN_PRIMARIO]', NOW(), NOW());

-- Asignar rol ADMIN
INSERT INTO backoffice_user_roles (role, user_id, created_at, updated_at)
VALUES (
'ADMIN',
(SELECT id FROM backoffice_users WHERE sub = '[SUB_ADMIN_PRIMARIO]'),
NOW(),
NOW()
);
```

Una vez creado el usuario, este podrá acceder al backoffice autenticándose mediante el IdP local que emite ese `sub`.

Generación de secretos y cifrados para cargas manuales

En condiciones normales, cuando se dan de alta IdP y SP desde el backoffice, la aplicación se encarga de:

- Codificar el `client_secret` de los SP con BCrypt.
- Cifrar los `client_secret` de los IdP (y cualquier otro dato sensible configurado para AES GCM) usando el `EncryptionService` del broker.

Solo es necesario hacer estos pasos manuales cuando:

- Se crean scripts SQL iniciales desde cero para poblar la base.

- Se corrige algún registro directamente en base de datos en un ambiente controlado.

Para estos casos se proporcionan dos clases utilitarias:

- `BCryptPasswordGenerationTest` para generar secretos de SP.
- `AESGCMEncryptionGenerationTest` para cifrar secretos de IdP u otros datos sensibles con AES GCM.

A continuación se describe cómo utilizarlas.

Generar client_secret de SP (BCrypt)

Los SP almacenan su secreto en la base en formato BCrypt.

Utilizar la clase:

BCryptPasswordGenerationTest

Código relevante:

```
String rawPassword = "mi-sp-secret";
String encoded = passwordEncoder.encode(rawPassword);
System.out.println(encoded);
```

Pasos:

1. Cambiar `rawPassword` por el secreto en texto plano.
2. Ejecutar el test desde el IDE o con Maven.
3. Copiar el valor impreso (ej: `{bcrypt}$2a$10$...`).
4. Usarlo en el script SQL del SP:

```
client_secret = '{bcrypt}$2a$10$...'
```

Generar client_secret de IdP (AES-GCM)

Los IdP almacenan su secreto en un valor cifrado con el `EncryptionService`.
Para esto se usa:

AESGCMEncryptionGenerationTest

La clase permite:

- Definir un texto claro (`plainText`)
- Generar o reutilizar una clave AES-GCM

- Obtener un valor cifrado compatible con el broker

Ejemplo típico:

String plainText = "mi-idp-secret";

String passphrase = "frase-segura";

String existingBase64Key = null; // genera una nueva

String existingBase64Salt = null; // genera un nuevo salt

Pasos:

1. Poner en `plainText` el secreto real del IdP.
2. Ejecutar el test.
3. Guardar:
    - `AES GCM Value` → va directo en la base (`client_secret`)
    - `Base64 Key` y `Salt` → deben configurarse en el `EncryptionService` del ambiente
4. SQL típico:

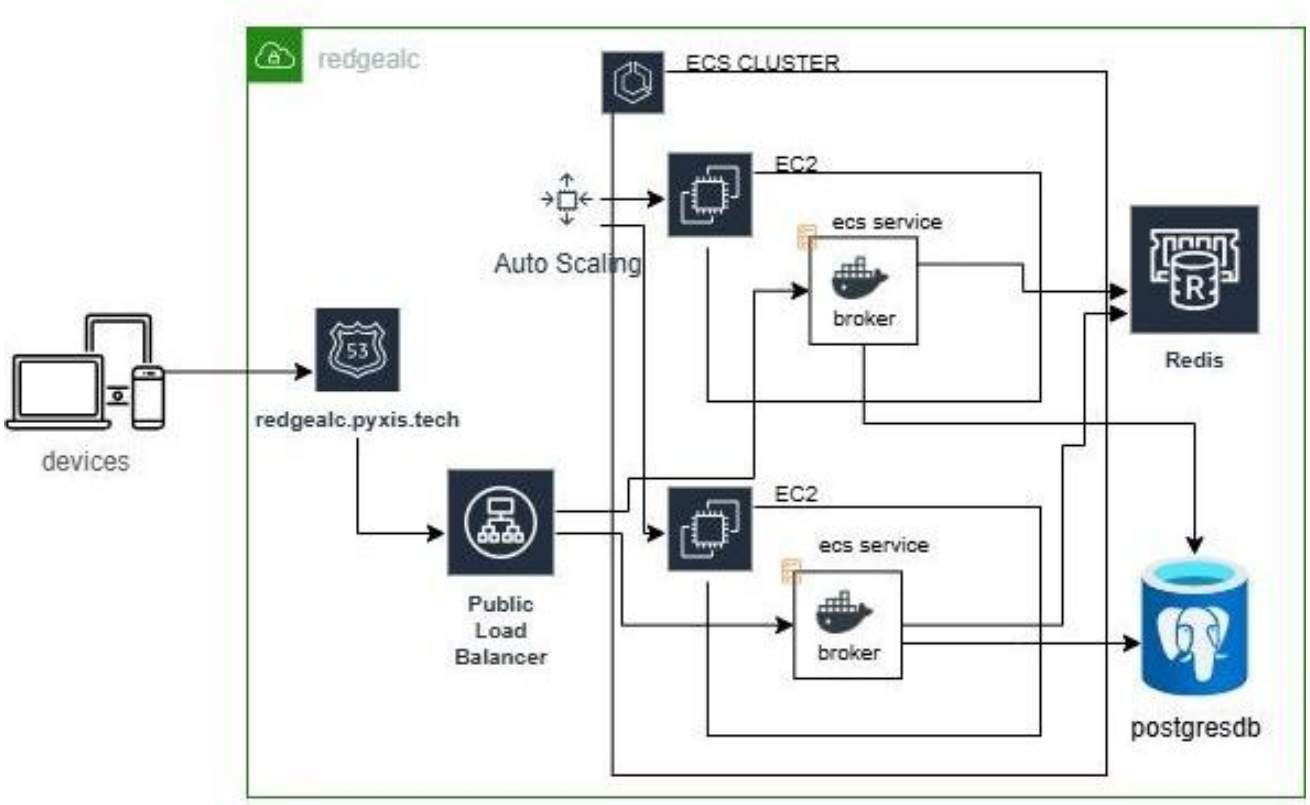
client_secret = 'iV9GjsclZIcYlurWt4d03SFPm4mDdQ...'

Importante: si ya existe una clave AES-GCM en el ambiente, cargarla en `existingBase64Key` y no generar una nueva.

5.4. Plataforma para pruebas de Integración

Se cuenta con la siguiente arquitectura disponible en infraestructura del proveedor de desarrollo (Pyxis) para pruebas con los países de la región.

La misma podrá ser utilizada en las etapas de validación de las integraciones, previo a una instalación en infraestructura propia del país.



Componentes desplegados:

- ECS Cluster: cluster de contenedores Docker
- EC2: nodos de cluster ECS
- Ecs_service: contenedores de cluster ECS
- AutoScaling Group: componente para escalado de instancias ID Broker
- Redis: base de datos de cache
- PostgreDB: base de datos de configuración de ID Broker
- ALB: Load Balancer para alta disponibilidad de la solución
- Route53: DNS para resolución de nombres.

6. Integraciones con ID Broker

A continuación, especificamos los requerimientos y características necesarias para integrar tanto servicios finales (Service Providers) como proveedores de identidades (IDPs).

6.1. Alta de proveedor de identidades en ID Broker (IDP)

En este caso, se integrará un proveedor de identidades (u otro Broker de identidades) que permitirá autenticar a personas ya registradas en él, en los servicios donde esté disponible y permitido (Service Provider).

6.1.1. Prerrequisitos

Esta sección asume que el lector conoce los conceptos y las tecnologías involucradas:

- Criptografía de clave pública, certificados digitales, SSL.
- Protocolo de comunicación Open ID Connect (OIDC)
- Tokens JWT (RFC 7519)

A su vez, antes de comenzar la integración, la organización debe contar con:

***Entorno técnico listo**:*

- Proveedor de identidades a integrar, que soporte el estándar OpenID Connect (OIDC) 2.0, entregando los claims definidos por la solución ID Broker.
- Soporte para HTTPS/TLS habilitado en los endpoints expuestos.
- Soporte para nivel de autenticación avanzado.

ID Broker proporciona la siguiente información:

- Redirect URI
- Logout URI

Datos para proporcionar al ID Broker:

- Client_id
- Secret
- Well-known URL (según protocolo OIDC)
- URL de logout/cierre de sesión (opcional, si soporta Single Logout)
- Alcances (scopes) requeridos: típicamente openid, profile, email.

***Accesos y roles**:*

- Contacto técnico definido para la integración del IDP en el ID Broker.
- Certificados o claves públicas, si el IDP requiere validación de tokens vía JWKs/JWT firmados.

Nivel de integración:

- Especificar soporte para Single Sign-On
- Especificar soporte para logout

Los detalles de la información de intercambio antes mencionada deberán ser presentadas a través de un formulario adjunto a esta guía.

6.2. Alta de servicio como Service Provider (SP)

6.2.1. Prerrequisitos

Esta sección asume que el lector conoce los conceptos y las tecnologías involucradas:

- Criptografía de clave pública, certificados digitales, SSL.
- Protocolo de comunicación Open ID Connect (OIDC)
- Tokens JWT (RFC 7519)

A su vez, antes de comenzar la integración, la organización debe contar con:

Entorno técnico listo:

- Aplicaciones web o servicios a integrar que soporten el estándar OpenID Connect (OIDC) y el procesamiento de los claims definidos por la solución ID Broker.
- URL de acceso pública (callback/redirect URL) donde recibirá los tokens emitidos por ID Broker.
- Soporte para HTTPS/TLS habilitado en los endpoints expuestos.

Datos para proporcionar al ID Broker:

- Nombre del Service Provider
- URLs de redirección autorizadas (redirect_uri)
- URL de logout/cierre de sesión (opcional, si soporta Single Logout)

Accesos y roles:

- Contacto técnico definido para la integración del SP en el ID Broker.

Nivel de integración:

- En esta primera instancia el ID Broker (MVP) no toma decisiones sobre las características de la autenticación devueltas por el IDP.
- Especificar soporte para Signle Sign-On
- Especificar soporte para logout

Los detalles de la información de intercambio antes mencionada deberán ser presentadas a través de un formulario adjunto a esta guía.

6.3. Alta del Broker Modelo como Proveedor OpenID Connect en un Service Provider externo

Objetivo

Esta sección describe cómo integrar el **Broker Modelo** como proveedor de identidad OpenID Connect dentro de una aplicación externa, en adelante el **Service Provider** o **SP**.

El alcance es el flujo estándar de autenticación por código de autorización con soporte opcional de refresh token.

Requisitos previos

Antes de comenzar, se requiere:

1. Que el SP soporte el protocolo OpenID Connect sobre OAuth 2.0, con flujo authorization_code.
2. Que el SP permita configurar:
    - un valor de Issuer
    - los endpoints OIDC
    - un par client_id y client_secret
    - una o varias redirect URIs
    - uno o varios scopes
3. Contar con un canal de comunicación con el equipo del Broker RedGealc para:
    - solicitar la creación del SP en el Broker
    - recibir el client_id y el client_secret
    - acordar las URLs de redirección y cierre de sesión

Datos de configuración del Broker (ambiente de pruebas)

Para el ambiente de pruebas del Broker, los datos principales se obtienen del documento .well-known/openid-configuration.

Issuer

https://broker.redgealc.pyxis.tech (testing)

Endpoints principales

Authorization endpoint: https://broker.redgealc.pyxis.tech/oauth2/authorize

Token endpoint: https://broker.redgealc.pyxis.tech/oauth2/token

UserInfo endpoint: https://broker.redgealc.pyxis.tech/userinfo

End session endpoint: https://broker.redgealc.pyxis.tech/connect/logout

JWKS URI: https://broker.redgealc.pyxis.tech/oauth2/jwks

Discovery document: https://broker.redgealc.pyxis.tech/.well-known/openid-configuration

Grant types soportados por el Broker

Según el documento well known, el Broker soporta:

authorization_code

client_credentials

refresh_token

Para la integración de un SP que usa al Broker como IdP se recomienda utilizar el flujo authorization_code. El uso de refresh_token puede habilitarse en función de las necesidades de sesión del SP.

Métodos de autenticación de cliente soportados

client_secret_basic

client_secret_post

client_secret_jwt

private_key_jwt

tls_client_auth

self_signed_tls_client_auth

En la mayoría de las integraciones típicas se utilizan client_secret_basic o client_secret_post, según las capacidades de la plataforma del SP.

Algoritmo de firma del ID Token

El documento well known indica:

id_token_signing_alg_values_supported: ["RS256"]

El SP debe estar preparado para validar ID tokens firmados con RS256 usando las claves publicadas en el JWKS del Broker.

Scopes soportados

El well known declara al menos:

scopes_supported: ["openid"]

Además, en el ejemplo de configuración de un SP se utilizan los scopes:

openid profile email phone auth_info

El scope openid es **obligatorio**. Los scopes adicionales permiten recibir atributos enriquecidos del usuario, siempre que estén habilitados para ese SP.

Datos de configuración específicos del SP

Cuando un nuevo SP desea integrarse con el Broker, el equipo del Broker realiza el alta interna en el backoffice. A modo de referencia, el siguiente ejemplo corresponde a un SP de pruebas asociado a IdUruguay:

- Organización: `Ministerio de Pruebas Digitales`
- Identificador interno del SP en el Broker: `SP-GUB_UY-TEST-001`
- País de la organización: `UY`
- Correo de contacto técnico: `admin@sp-test.uy`
- Nombre del cliente: `SP GUB-UY`
- Grant types habilitados: `authorization_code` y `refresh_token`
- Métodos de autenticación de cliente permitidos: `client_secret_basic` y `client_secret_post`
- Scopes asignados al SP: `openid profile email phone auth_info`
- Redirect URIs de ejemplo:

[

"https://auth-dev.id-uruguay.com:8444/federated-identity/authorization-code",

"https://mi-dev.id-uruguay.com:3000/"

]

- Post logout redirect URIs de ejemplo:

· [

· "https://mi-staging.iduruguay.gub.uy/",

· "https://auth-staging.iduruguay.gub.uy/federated-identity/logout"

]

El valor de `client_secret` se almacena en el Broker utilizando hashing bcrypt, por lo que **no se expone en la base de datos en texto plano**. El equipo del Broker entrega el `client_secret` al SP únicamente por un canal seguro, por ejemplo intercambio manual cifrado, gestor de secretos compartido u otro mecanismo acordado.

Para cada nuevo SP externo, el equipo del Broker proveerá al integrador los siguientes datos concretos:

- `client_id`
- `client_secret`
- listado de `redirect_uris` autorizadas
- scopes habilitados
- post logout redirect URIs si aplica

Pasos generales de configuración en el SP

Los pasos exactos dependen de la plataforma del SP, pero en general se requiere la siguiente configuración:

1. **Crear una nueva conexión OIDC u OpenID Provider** en la plataforma del SP.
2. Configurar el **Issuer** con el valor del Broker.
3. Configurar manualmente los endpoints o permitir que el SP los descubra desde el documento well known:
    - authorization endpoint
    - token endpoint
    - userinfo endpoint
    - end session endpoint
    - jwks uri
4. Cargar las credenciales asignadas por el Broker:
    - `client_id`

- `client_secret`
- método de autenticación del cliente (por ejemplo `client_secret_basic`).

5. Registrar las **redirect URIs** que se utilizarán en los flujos de autenticación. Deben coincidir exactamente con las que se configuraron en el Broker.

6. Definir el flujo de OAuth:
   - `response_type = code`
   - `grant_type = authorization_code`
   - opcionalmente `refresh_token`.

7. Seleccionar los **scopes** que se solicitarán, incluyendo al menos `openid` y, según las necesidades del SP, `profile`, `email`, `phone` y `auth_info` si están habilitados.

8. Configurar, si la plataforma lo permite, la validación de ID token:
   - firma RS256
   - obtención automática de claves desde el JWKS del Broker
   - verificación del issuer y del audience (`aud`) igual al `client_id`.

Pruebas básicas de integración

Una vez completada la configuración, se recomienda verificar:

1. Que el SP muestra correctamente el enlace o botón de inicio de sesión con el Broker RedGealc.

2. Que el usuario es redirigido a la URL del Broker, se autentica en su IDP y luego retorna al SP a través de una de las redirect URIs configuradas.

3. Que el SP recibe un código de autorización, puede canjearlo por un ID token y un access token en el token endpoint, y valida correctamente la firma y los campos del ID token.

4. Que el cierre de sesión en el SP dispara una llamada al end session endpoint del Broker y se respeta la post logout redirect URI configurada.

Errores frecuentes

Algunos errores habituales en esta etapa son:

- `invalid_redirect_uri` o similar, cuando la redirect URI utilizada por el SP no coincide exactamente con las registradas en el Broker.

- `invalid_client` o `unauthorized_client`, cuando el `client_id` o el `client_secret` son incorrectos o el método de autenticación no coincide con lo configurado.

- `invalid_scope`, cuando el SP solicita scopes que no están habilitados para su registro.

- Fallos en la validación del ID token por no comprobar correctamente el `issuer` o por no obtener las claves del JWKS del Broker.

Ante cualquiera de estos errores, se recomienda revisar la configuración y, en caso de dudas, contactar al equipo del Broker indicando el mensaje de error completo y el `client_id` afectado.

6.4. Configurar nuevo IDP - Desarrollar claim extractor

Introducción

Cada Identity Provider (IDP) entrega información del usuario con estructuras y formatos propios. El broker tiene que unificar estos formatos para operar de forma consistente. Cuando la respuesta de un nuevo IDP no coincide con alguno de los formatos conocidos, es necesario implementar un Claim Extractor.

Un Claim Extractor es el componente que lee los claims del `user_info` de un IDP y extrae los atributos que el broker requiere. Su función es mapear la estructura propia del IDP hacia el modelo del broker.

El Claim Extractor debe identificar, como mínimo, los siguientes atributos:

- document
- ae
- nid
- rid

Estos valores son obligatorios para que el broker pueda construir los claims de identidad del usuario.

Desarrollo de Claim Extractor

El desarrollo implica en extender la clase abstracta `FederatedClaimExtractorBase`, la cual participa directamente en el proceso de autenticación y construye las claims a partir del user_info del IDP.

La implementación debe proveer los métodos definidos en la interfaz:

```
public interface FederatedClaimExtractor extends Converter<Map<String, Object>,
Map<String, Object>> {

    String getName();

    Integer getAE(Map<String, Object> source);

    Integer getRID(Map<String, Object> source);

    Integer getNID(Map<String, Object> source);

    Map<String, String> getDocument(Map<String, Object> source);

    Map<String, Object> getAdditionalClaims(Map<String, Object> source);
}
```

El método `getName` define el identificador del Claim Extractor y es el valor utilizado en las application properties para asociarlo a un IDP específico.

El resto de los métodos extrae atributos del usuario (`ae`, `rid`, `nid`, `document`). El método `getAdditionalClaims` permite mapear claims adicionales provenientes del IDP con claims estándar OpenID para exponerlas en el broker.

Una vez finalizado el proceso de autenticación, las claims generadas quedan registradas dentro de la aplicación bajo los siguientes atributos:

- `broker.document`
- `broker.ae`
- `broker.nid`
- `broker.rid`
- `broker.sub`

El atributo `broker.sub` se utiliza posteriormente como `sub` del usuario en la emisión de tokens y user_info. Su valor se construye a partir del documento siguiendo el formato definido por el broker:
`${pais-documento}-${tipo-documento}-${numero-documento}`

El broker expone tanto en el `id_token` como en el `user_info` las claims generadas por el extractor, junto con las claims estándar identificadas durante el proceso.

Para que el Claim Extractor sea detectado por el sistema, basta con declararlo como un Spring Bean dentro del contexto de la aplicación. El broker seleccionará automáticamente el extractor apropiado según la configuración del IDP.

Configuración

Para que el broker utilice un Claim Extractor durante el proceso de autenticación, es necesario asociar dicho extractor con el issuer del IDP.

Esta asociación se define en las application properties de la siguiente manera:

```
broker:
  converters:
        configs:
        - name: ${claim_extractor_name}
        accepted-issuer-uris:
        - ${issuer_uri_1}
        - ${issuer_uri_2}
```

El valor `name` debe coincidir con el nombre retornado por el método `getName()` del Claim Extractor. Cada entrada en `accepted-issuer-uris` indica un issuer válido para el cual el extractor puede ser aplicado.

Durante la autenticación, el broker selecciona el primer converter acepte el issuer del IDP. Una vez identificado, utiliza ese Claim Extractor para generar las claims a partir del `user_info`.

Ejemplo - GUB UY

A modo de ejemplo se presenta el claim extractor utilizado para el mapeo de claims de GUB UY.

La respuesta del user info de GUB UY tiene el formato:

```
{
        "sub": "123456",
        "name": "Nombre y Apellido",
        "given_name": "Nombre",
        "family_name": "Apellido",
        "nickname": "uy-ci-12345678",
        "email": "nomre.apellido@mail.com",
```

"email_verified": true,
"pais_documento": {
"codigo": "uy",
"nombre": "Uruguay"
},
"tipo_documento": {
"codigo": 68909,
"nombre": "C.I."
},
"numero_documento": "12345678",
"ae": "urn:uce:ae:1",
"nid": "urn:uce:nid:1",
"rid": "urn:uce:rid:2"
}

GUB UY Claim extractor

Para identificar los campos de esta respuesta se implementa el claim extractor

```
@Component
public class FederatedClaimExtractorGubUy extends FederatedClaimExtractorBase {

    private static final String NAME = "gubuy";

    public FederatedClaimExtractorGubUy() {
    super(NAME);
    }

    @Override
    public Integer getAE(Map<String, Object> source) {
    return Optional.ofNullable((String) source.get("ae"))
    .map(this::parseUrnIntegerValue)
    .orElse(null);
    }
```

```
@Override
public Integer getRID(Map<String, Object> source) {
return Optional.ofNullable((String) source.get("rid"))
.map(this::parseUrnIntegerValue)
.orElse(null);
}

@Override
public Integer getNID(Map<String, Object> source) {
return Optional.ofNullable((String) source.get("nid"))
.map(this::parseUrnIntegerValue)
.orElse(null);
}

@Override
public Map<String, String> getDocument(Map<String, Object> source) {
Map<String, String> documentMap = new HashMap<>();
String documentType = getDocumentType(source);

if (documentType != null) {
documentMap.put("document_type", documentType);
}

String documentId = getDocumentNumber(source);
if (documentId != null) {
documentMap.put("document_id", documentId);
} else {
throw new InvalidIdpResponseException("Could not extract document_id from source
claims");
}

String documentCountryCode = getDocumentCountryCode(source);
if (documentCountryCode != null) {
```

```
        documentMap.put("document_country", documentCountryCode);
        }

        return documentMap;
        }

        private Integer parseUrnIntegerValue(String urn) {
        String[] splited = urn.split(":");
        String urnValue = splited[splited.length - 1];
        return Integer.valueOf(urnValue);
        }

        private String getDocumentType(Map<String, Object> source) {
        return Optional.ofNullable((Map<String, Object>) source.get("tipo_documento"))
        .map(documentTypeMap -> (String) documentTypeMap.get("nombre"))
        .map(documentTypeName -> documentTypeName.replaceAll("\\.", ""))
        .orElseThrow(() -> new InvalidIdpResponseException("Could not extract
document_type from source claims"));
        }

        private String getDocumentNumber(Map<String, Object> source) {
        return (String) source.get("numero_documento");
        }

        private String getDocumentCountryCode(Map<String, Object> source) {
        return Optional.ofNullable((Map<String, Object>) source.get("pais_documento"))
        .map(documentTypeMap -> (String) documentTypeMap.get("codigo"))
        .map(String::toUpperCase)
        .orElseThrow(() -> new InvalidIdpResponseException("Could not extract
document_country from source claims"));
        }
}
```

Por ultimo se configura el claim extractor para que actue cuando el issuer es el ambiente de testing de GUB UY

broker:

converters:

configs:

- name: gubuy

accepted-issuer-uris:

- "https://auth-testing.iduruguay.gub.uy/oidc/v1"

Caso por defecto: interoperabilidad automática entre Brokers

Cuando el Broker recibe información de usuario (`userinfo`) desde un IDP federado, selecciona el Claim Extractor correspondiente en función del issuer configurado.
Si no existe una configuración explícita para ese issuer, el sistema utiliza automáticamente el extractor por defecto (`FederatedClaimExtractorDefault`).

El extractor por defecto implementa el modelo estándar definido por el Broker y espera que el IDP entregue los atributos mínimos en la estructura común:

- `document.document_type`
- `document.document_id`
- `document.document_country`
- `ae`
- `nid`
- `rid`

Este extractor construye entonces los claims internos `broker.*`, incluyendo `broker.sub`, de acuerdo con el formato oficial del Broker

.

6.5. Integración automática entre Brokers

Cuando el IDP remoto es **otro Broker MOdelo**,, este ya expone los atributos en el formato estándar.
Por lo tanto:

- no es necesario desarrollar un Claim Extractor específico
- no es necesario realizar configuraciones adicionales
- la federación entre brokers funciona en modo **plug & play**

El extractor por defecto actúa como mecanismo de compatibilidad nativo entre Brokers, garantizando que cualquier instancia pueda federarse con otra sin requerir desarrollo adicional.

Integración entre brokers

Se quiere configurar Broker A como IDP de Broker B actuando como SP.

Para esto se tiene que:

- Paso 1 - En bróker B→ Configurar IDP con información de Broker A
- Paso 2 - En bróker A → Configurar SP con información de Broker B

Paso 1 - En bróker B → Configurar IDP con informaicón de Broker A

En consola de backoffice de bróker B, dar de alta un IDP con la información del bróker A como IDP
Por ejemplo:

campo	valor
Endpoints OIDC	
Issuer	https://int-broker.redgealc.pyxis.tech
Client ID	broker-testing
Client Secret	broker-TESTING

Authorization Endpoint	`https://int-broker.redgealc.pyxis.tech/oauth2/authorize`
Token Endpoint	`https://int-broker.redgealc.pyxis.tech/oauth2/token`
Userinfo Endpoint	`https://int-broker.redgealc.pyxis.tech/userinfo`
JWKS URI	`https://int-broker.redgealc.pyxis.tech/oauth2/jwks`
Scopes	`openid profile email phone auth_info`
Well-known URL	https://int-broker.redgealc.pyxis.tech/.well-known/openid-configuration
Información de la Organización	
Nombre de Organización	Broker A
Identificador IDP	broker-int-idp
País (Código ISO)	PA
Contacto Técnico	
Descripción	Broker A actuando como IDP
Nivel AE Máximo	3

NID Mínimo	3
Prioridad de Visualización	10
URL del Logo	
Activo	SI
IDP Extranjero	SI

Paso 2 - En bróker A- Configurar SP con información de Broker B

En consola de backoffice de bróker A dar de alta un SP con la información del bróker B como SP

campo	valor
Client ID	broker-testing
Client Secret	broker-TESTING
Nombre del Cliente	Broker Testing SP
Métodos de Autenticación	client_secret_basic client_secret_post
Grant Types	authorization_code refresh_token
Scopes	openid profile email phone auth_info

Redirect URIs	https://broker.redgealc.pyxis.tech/login/oauth2/code/broker-int-idp
Post Logout Redirect URIs	https://broker.redgealc.pyxis.tech/logout
Información de la Organización	
Nombre de Organización	Broker B SP
ID de Organización	broker-test-sp
País	CL
Contacto Técnico	
Activo	SI
Es Backoffice	NO
Descripción del servicio	Broker B actuando como SP