MINISTERIO DE LA PRESIDENCIA
ESTADO PLURINACIONAL DE BOLIVIA

AGETIC
agencia de gobierno electrónico y tecnologías de información y comunicación

Ciudadanía Digital

Especificaciones técnicas para el servicio de Autenticación

Versión 2.0.0

Contenido

1. DESCRIPCIÓN

Introducción

En este documento se describen todos los pasos que deben seguir los sistemas que utilizarán la plataforma de ciudadanía digital como medio de autenticación.

Las especificaciones del presente documento están basados en OpenID Connect, que es una capa de identidad basada en las especificaciones del protocolo OAuth 2.0 que define mecanismos para obtener y usar los token de acceso.

Para una mayor comprensión de los protocolos, véase:

Especificación del protocolo OAuth 2.0 https://tools.ietf.org/html/rfc6749 Especificación del protocolo OpenId Connect: https://openid.net/specs/openid-connect-core-1_0.html

Terminología

La siguiente terminología es usada en el resto del documento y busca ser compatible con la terminología utilizada en OpenId Connect.

Roles

- Propietario del recurso (Resource Owner) Es el ciudadano que tiene acceso a un navegador web y es capaz de dar acceso a sus recursos protegidos. Se puede utilizar el término usuario, ciudadano o End-User.
- Cliente (Client) Es el Sistema Cliente que hace peticiones a recursos protegidos en nombre del propietario del recurso y con la autorización del mismo. También se pueden usar los términos: sistema cliente, aplicación cliente para hacer referencia a este concepto.
- Proveedor de Identidad (Identity Provider) Es el Sistema Proveedor de Identidad que es el responsable de validar las credenciales del ciudadano (Authorization endpoint) y generar tokens de acceso (Token endpoint). Se utilizan los términos servidor de autenticación de Ciudadanía Digital, Provider, Servidor de Autorización (en terminología referente a OAuth 2.0)
- Recurso protegido (Protected Resource) Son los recursos que tiene acceso el Resource Owner tras obtener un token de acceso.

Tokens

- ID token Un ID token es un JSON Web Token (JWT) que contiene información sobre el

proceso de autenticación del End-User en el servidor de autenticación. (https://openid.net/specs/openid-connect-core-1_0.html)

- Access token Los Access tokens son credenciales emitidas por el servidor de autenticación para un sistema cliente, y tienen como fin permitir a estos últimos el acceso a recursos protegidos. Un Access Token es un string opaco que representa el acceso a ciertos datos y puede ser utilizado por un tiempo limitado. (https://tools.ietf.org/html/rfc6749#section-1.4)
- Refresh token Los Refresh tokens son credenciales emitidas por el servidor de autenticación para un sistema cliente, y tienen como fin la obtención de nuevos Access tokens, cuando estos expiran o se vuelven inválidos. (https://tools.ietf.org/html/rfc6749#section-1.5)

Scopes

Indican los recursos que el cliente quiere acceder a nombre del usuario. Pueden entenderse como permisos que el usuario final autorizará para que una determinada aplicación cliente pueda realizar operaciones.

Estos scopes o permisos pueden ser públicos (public), privados (private) o internos (internal). Los scopes tipificados como públicos son aquellos que pueden ser implementados por instituciones del Estado, ajenas a los detalles de implementación del ecosistema de Ciudadanía Digital. Los scopes privados e internos están destinados a ser parte de la implementación interna del ecosistema de Ciudadanía Digital.

Se listan a continuación los scopes públicos:

Nombre	Descripción	Tipo	
profile	Permiso para acceder al Perfil público del usuario (nro_documento, nombre completo)	público	required
nombre	Permiso para acceder al Nombre completo del usuario	público	`deprecated`
documento_identidad	Permiso para acceder al número de documento de identidad del usuario	público	`deprecated`
fecha_nacimiento	Permiso para acceder a la fecha de nacimiento del usuario. Pedirlo sólo si se hará uso de ella dentro del sistema cliente.	público	
email	Permiso para acceder a la dirección de correo electrónica del usuario. Pedirlo sólo si se hará uso de él dentro del sistema cliente.	público	
celular	Permiso para acceder al número de celular del usuario. Pedirlo sólo si se hará uso de él dentro del sistema cliente.	público	
openid	Permiso para acceder a la información de la autenticación del usuario	público	required
offline_access	Permiso para acceder a refresh token	público	

Discovery

Acorde a la especificación de OpenID Connect (https://openid.net/specs/openid-connect-discovery-1_0.html), el mecanismo de autenticación expone un endpoint para descubrir la configuración del

proveedor de identidad. La configuración se encuentra en notación de objetos Javascript, JSON.

La configuración se encuentra en:

`https://<base-url-proveedor-identidad>/.well-known/openid-configuration`

La respuesta tendrá un aspecto similar al siguiente:

```
{
    "authorization_endpoint":"https://<base-url-proveedor-identidad>/auth",
    "claims_parameter_supported":true,
    "claims_supported":[
        "sub",
        "status",
        "profile",
        "nombre",
        "documento_identidad",
        "fecha_nacimiento",
        "email",
        "address",
        "celular"
    ],
    "code_challenge_methods_supported":[
        "plain",
        "S256"
    ],
    "end_session_endpoint":"https://<base-url-proveedor-identidad>/session/end",
    "check_session_iframe":"https://<base-url-proveedor-identidad>/session/check",
    "grant_types_supported":[
        "authorization_code",
        "refresh_token",
        "client_credentials"
    ],
    "id_token_signing_alg_values_supported":[
        "HS256",
        "PS256",
        "RS256",
        "ES256"
    ],
    "issuer":"https://<base-url-proveedor-identidad>",
    "jwks_uri":"https://<base-url-proveedor-identidad>/jwks",
    "registration_endpoint":"https://<base-url-proveedor-identidad>/reg",
    "request_object_signing_alg_values_supported":[
        "HS256",
        "RS256",
        "PS256",
        "ES256",
        "EdDSA"
    ],
    "request_parameter_supported":false,
    "request_uri_parameter_supported":true,
    "require_request_uri_registration":true,
    "response_modes_supported":[
        "form_post",
        "fragment",
        "query",
        "web_message",
        "jwt",
        "query.jwt",
        "fragment.jwt",
        "form_post.jwt",
        "web_message.jwt"
    ],
    "response_types_supported":[
        "code"
    ],
    "scopes_supported":[
```

```
        "openid",
        "offline_access",
        "status",
        "profile",
        "nombre",
        "documento_identidad",
        "fecha_nacimiento",
        "email",
        "address",
        "celular"
    ],
    "subject_types_supported":[
        "public",
        "pairwise"
    ],
    "token_endpoint_auth_methods_supported":[
        "client_secret_basic",
        "client_secret_post",
        "none"
    ],
    "token_endpoint":"https://<base-url-proveedor-identidad>/token",
    "userinfo_endpoint":"https://<base-url-proveedor-identidad>/me",
    "userinfo_signing_alg_values_supported":[
        "HS256",
        "PS256",
        "RS256",
        "ES256"
    ],
    "authorization_signing_alg_values_supported":[
        "HS256",
        "PS256",
        "RS256",
        "ES256"
    ],
    "introspection_endpoint":"https://<base-url-proveedor-identidad>/token/introspection",
    "introspection_endpoint_auth_methods_supported":[
        "client_secret_basic",
        "client_secret_post",
        "none"
    ],
    "introspection_signing_alg_values_supported":[
        "HS256",
        "PS256",
        "RS256",
        "ES256"
    ],
    "revocation_endpoint":"https://<base-url-proveedor-identidad>/token/revocation",
    "revocation_endpoint_auth_methods_supported":[
        "client_secret_basic",
        "client_secret_post",
        "none"
    ],
    "id_token_encryption_alg_values_supported":[
        "A128KW",
        "A256KW",
        "ECDH-ES",
        "ECDH-ES+A128KW",
        "ECDH-ES+A256KW",
        "RSA-OAEP"
    ],
    "id_token_encryption_enc_values_supported":[
        "A128CBC-HS256",
        "A128GCM",
        "A256CBC-HS512",
        "A256GCM"
    ],
    "userinfo_encryption_alg_values_supported":[
        "A128KW",
```

```
        "A256KW",
        "ECDH-ES",
        "ECDH-ES+A128KW",
        "ECDH-ES+A256KW",
        "RSA-OAEP"
    ],
    "userinfo_encryption_enc_values_supported":[
        "A128CBC-HS256",
        "A128GCM",
        "A256CBC-HS512",
        "A256GCM"
    ],
    "introspection_encryption_alg_values_supported":[
        "A128KW",
        "A256KW",
        "ECDH-ES",
        "ECDH-ES+A128KW",
        "ECDH-ES+A256KW",
        "RSA-OAEP"
    ],
    "introspection_encryption_enc_values_supported":[
        "A128CBC-HS256",
        "A128GCM",
        "A256CBC-HS512",
        "A256GCM"
    ],
    "authorization_encryption_alg_values_supported":[
        "A128KW",
        "A256KW",
        "ECDH-ES",
        "ECDH-ES+A128KW",
        "ECDH-ES+A256KW",
        "RSA-OAEP"
    ],
    "authorization_encryption_enc_values_supported":[
        "A128CBC-HS256",
        "A128GCM",
        "A256CBC-HS512",
        "A256GCM"
    ],
    "request_object_encryption_alg_values_supported":[
        "A128KW",
        "A256KW"
    ],
    "request_object_encryption_enc_values_supported":[
        "A128CBC-HS256",
        "A128GCM",
        "A256CBC-HS512",
        "A256GCM"
    ],
    "backchannel_logout_supported":true,
    "backchannel_logout_session_supported":true,
    "frontchannel_logout_supported":true,
    "frontchannel_logout_session_supported":true,
    "claim_types_supported":[
        "normal"
    ],
    "version":"2.0.0"

}
```


Clientes

Todo sistema que utiliza la plataforma de Ciudadanía Digital como medio de autenticación es

denominado Cliente. Indistintamnte se pueden usar los términos **client** (propio de la terminología OAuth 2.0 y OpenId Connect), **sistema cliente**, **aplicación cliente**.

Tipos de clientes

Un cliente debe pasar por un proceso de registro que finalmente otorga unas credenciales de acceso que permitirán al cliente autenticarse contra el Proveedor de Identidad.

Se puede clasificar al sistema cliente como:

- Confidencial: es una aplicación que se ejecuta sobre un servidor protegido y puede almacenar de manera segura los secretos relacionados a las credenciales del cliente.
- Público: es una aplicación que se ejecuta principalmente sobre un dispositivo del usuario final o en un navegador web y por esa naturaleza es incapaz de almacenar de manera segura los secretos relacionados a las credenciales del cliente

Perfiles de clientes

- Aplicaciones web: clientes confidenciales que ejecuta su código principalmente sobre un protegido servidor back-end server. El servidor puede almacenar los secretos de manera segura.
- Aplicaciones del navegador: Se asumen como clientes públicos con el código ejecutándose principalmente en el navegador web del usuario.
- Aplicaciones nativas: Se asumen también como clientes públicos con el código ejecutándose en el dispositivo del usuario final; principalmente las aplicaciones móviles y de escritorio.

Registro del sistema cliente

Es recomendable que antes de la implementación del mecanismo de autenticación de Ciudadanía Digital en ambientes de producción, se realice la integración en entorno de pruebas.

Para ello, es necesario solicitar el registro del sistema cliente al correo ciudadania@agetic.gob.bo indicando los siguientes datos:

- Perfil cliente: El tipo de aplicacion del sistema cliente, estos estan descritos en el apartado anterior: perfiles de clientes.
- Uri de redirección: Es la Uri donde se redirigirá al usuario final tras completar el proceso de autenticación. Debe estar sobre https y no debe contener caracteres especiales, como querys o fragmentos de url (en el caso de ser app movil enviar el activity).
- Uri de redirección tras el logout: Es la Uri donde se redirigirá al usuario final después de cerrar sesión en el proveedor de identidad. Debe estar sobre https y no debe contener

caracteres especiales, como querys o fragmentos de url (en el caso de ser app movil enviar el activity).

- Nombre: Nombre del sistema cliente
- Scopes: Lista de permisos que se solicitarán al usuario final para su autorización a nombre del sistema cliente (estos estan descritos en la sección scopes).
- ¿Va a utilizar refresh token? Sí o no
- Metodo de autenticacion para acceder al endpoint token: Los valores puede ser:
    - basic
    - post
    - none (para aplicaciones moviles)
- Contactos: Correo electrónico institucional de la persona que servirá de contacto para recibir información sobre el presente mecanismo.
- Url de términos y condiciones (**opcional**): Es la Url en el dominio del sistema cliente que expone los términos y condiciones del uso del mismo. Se mostrará como un enlace en las pantallas de autenticación.
- Url política de privacidad (**opcional**): Es la Url en el dominio del sistema cliente que expone la política de privacidad del mismo. Se mostrará como un enlace en las pantallas de autenticación.
- Url del logo del sistema cliente (**opcional**): Es la Url que conduce al logo que identifica al sistema cliente. Se mostrará en las pantallas de autenticación y será escalado acorde al diseño de estas.

Una vez realizado el registro se enviarán las credenciales del sistema cliente al contacto de la entidad. Las credenciales del sistema cliente contendrán los datos `client_id`, `client_secret`, los mismos deben ser conservados de manera segura, ya que estos datos serán necesarios en el proceso de obtención del token de acceso y el token de identidad por parte del sistema cliente.

Autenticación y Autorización endpoints

Authorization endpoint

El endpoint de autorización permite al sistema cliente ser autorizado por parte del usuario final.

El usuario debe inicialmente autenticarse en el Proveedor de Identidad y posteriormente autorizar al cliente por lo que este paso se lo realiza en un navegador web con la interacción del usuario final.

Para el proceso de autenticación y autorización se deberá seguir con los siguientes pasos :

1. Generar valores aleatorios:

Para proteger la seguridad de los ciudadanos mediante la prevención de ataques de falsificación de solicitudes, se debe crear un par de tokens de sesión únicos para mantener el estado entre el sistema cliente y Proveedor de Identidad. Estos dos tokens son denominados `state` y `nonce`

Los valores deben ser generados de manera aleatoria y debe ser conservados para ser validados más adelante (se sugiere generar una cadena de al menos 30 caracteres).

Sólo si se está integrando una aplicacion movil adicionalmente debe generar los siguentes códigos:

- code_verifier: Es recomendable que sea una cadena aleatoria criptográfica de alta entropía que utiliza los caracteres no reservados [A-Z] / [a-z] / [0-9] / "-" / "." / "_" / "~", con una longitud mínima de 43 caracteres y una longitud máxima de 128 caracteres.

Ejemplo:

```
code_verifier =
2AcLLc82Tdu8HUESuVxJeL28DGavoDQfpJGSjLLC3FfJfpwtWR0efZaTUgHRL3wTvkwUJ9nwuTd9QXSA
```

- code_challenge: Para generar el code_challenge se recomienda utilizar la codificación SHA256, para la generación de este código es necesario usar la siguiente formula:

```
code_challenge = BASE64URL_ENCODE(SHA256(ASCII(code_verifier)))
```

Ejemplo:

```
code_challenge = mKlsmdDCnEeIatFQbv0CJugIeiaIcU_IfKfuBOfQeJE
```

2. Enviar solicitud al Proveedor de Identidad

Siempre en el navegador web, se debe enviar la petición HTTP al valor del campo authorization_endpoint que se obtiene del punto anterior, con los siguientes parámemtros:

Parámetro	Condición	Descripción
client_id	requerido	Es el identificador del Sistema Cliente, este identificador se obtiene cuando se registra el Sistema Cliente en el Sistema Proveedor de Identidad.
response_type	requerido	Parámetro utilizado en el flujo openID connect y siempre deberá ser code.
redirect_uri	requerido	Es la url que recibirá el código de acceso tras la autenticación en el proveedor de identidad. Debe estar codificado en formato url y debe ser la misma que se proporcionó en el registro del sistema cliente.
scope	requerido	Los scopes enviados debe ser un subconjunto de los scopes solicitados en la creación del cliente. Los scopes deben estar separados por un espacio.
state	requerido	Valor del token de estado y de sesión única creada en el paso anterior.
nonce	requerido	Valor aleatorio generado en el sistema cliente. Es utilizada para la posterior verificación del id_token y evita replay attacks.
prompt	opcional	`prompt=login`: *este parametro sirve asegurarse de que el Usuario final esta presente en la sesión actual (si se envía este parámetro el proveedor de identidad siempre solicitará las credenciales al ciudadano en el inicio de sesión)*. `prompt=consent`: *este parametro sirve para solicitar refresh_token*

Parámetro	Condición	Descripción
code_challenge_method	opcional	Algoritmo con el que se obtubo el hash para generar el code_challenge, si utilizo SHA256 enviar el valor: S256 **(este valor es requerido sóli si el cliente es una aplicación movil)**

Ejemplo de solicitud:

```
https://<base-url-proveedor-identidad>/auth?
response_type=code&client_id=mGntc6oKqoFMCRZArKDad&state=509ccc2713049e6efea071a9c34f6f45&nonce=2313
01a1afe20d88ca963ee84c3929c3&redirect_uri=https://dominio/app/login&scope=openid%20profile
```

Ejemplo de solicitud que siempre pide las credenciales al ciudadano:

```
https://<base-url-proveedor-identidad>/auth?
response_type=code&client_id=mGntc6oKqoFMCRZArKDad&state=509ccc2713049e6efea071a9c34f6f45&nonce=2313
01a1afe20d88ca963ee84c3929c3&redirect_uri=https://dominio/app/login&scope=openid
%20profile&prompt=login
```

Ejemplo de solicitud de una aplicacion movil

```
https://<base-url-proveedor-identidad>/auth?
response_type=code&client_id=yhOiNSDcWeXQEz2Hs7DeW&state=509ccc2713049e6efea071a9c34f6f45&redirect_u
ri=net.openid.appauthdemo:/oauth2redirect&scope=openid
%20profile&code_challenge=mKlsmdDCnEeIatFQbv0CJugIeiaIcU_IfKfuBOfQeJE&code_challenge_method=S256
```

Una vez solicitada la URL el usuario final se autenticará en el dominio del Proveedor de Identidad y autorizará al sistema cliente.

3. Redirección al sistema cliente

Una vez que el ciudadano haya procedido a autenticarse en el proveedor de Identidad el mismo redirigirá al Sistema Cliente (a la url registrada como redirect_uri) con un código de autorización (authorization code), los códigos state y nonce, en caso de éxito; por el contrario, un código de error en caso de que la autenticación no fuera exitosa, petición malformada o se produjera un error.

Sobre el código de autorización es necesario mencionar que es un código intermedio, opaco y que puede ser usado **solo una vez**, es necesario para obtener el token de acceso y, eventualmente, un token de refresh.

En caso de retorno exitoso el sistema cliente debe verificar que el valor del parámetro state y el valor del parámetro nonce que recibe del proveedor de identidad coincide con el generado en paso anterior. Esto da la certeza de que la petición no ha sido fruto de alún tipo de ataque de falsificación de solicitud.

Ejemplo de retorno con autenticación exitosa:

```
https://<sistema-cliente/callback>?
code=bzjggVXIHPrqZDAwlTAC5W6Z2BkrNXefHPi3jCqGkcG&state=509ccc2713049e6efea071a9c34f6f45&nonce=2sad12
aa292kdhj1l34dks2&session_state=pe-3iNh4OnD_uGuKj91nrDsQsS09KJpUCB2cvBaBfvQ
```

Ejemplo de retorno con autenticación NO exitosa:

```
https://<sistema-cliente/callback>?error=consent_required&error_description=client%20not
%20authorized%20for%20End-User%20session
%20yet&state=509ccc2713049e6efea071a9c34f6f45&nonce=2sad12aa292kdhj1l34dks2&session_state=pe-
```

```
3iNh4OnD_uGuKj91nrDsQsS09KJpUCB2cvBaBfvQ
```

Token endpoint

Para intercambiar el código de autorización (authorization code) por el token de acceso y el token de identidad se debe enviar una petición POST via HTTPS con los parámetros necesarios.

La url donde se debe enviar la solicitud POST es la siguiente:

```
https://<base-url-proveedor-identidad>/token
```

Los parámetros necesarios que se deben enviar son:

Parámetro	tipo	Condición	Descripción
Authorization	header	`requerido` si el metodo de autenticacion del cliente es `basic`	El id_client y secret (ambos obtenidos en el registro del sistema cliente) deben ser pasados en la cabecera mediante la autenticación básica (el secret debe estar codificado con urlEncode)
code	body	requerido	Es el código de acceso obtenido en el punto anterior
redirect_uri	body	requerido	Es la misma uri utilizada en el paso anterior
grant_type	body	requerido	valor: `authorization_code`
code_verifier	body	requerido si es una app movil	Valor generado en el paso anteriror
client_id	body	requerido si el tipo de autenticaion del cliente es `post` o es aplicacion movil	Identificador del cliente, obtenido al crear el cliente
client_secret	body	requerido si el tipo de autenticacion del cliente es `post`	Credencial del sistema cliente obtenido en el registro

Ejemplo de solicitud si el tipo de autenticación es `basic`:

```
curl -X POST \
  https://<base-url-proveedor-identidad>token \
  -H 'authorization: Basic
M1ZOS0x4UkNjUVF4US05NEphcGhjOnBabjAzbG1PaE1Ga0E0Y0h4V2REY0xYSHBtOXlsMU9CSkNnRTFtRDFoYmFOaXdBYWxPbHAx
WkJRQjVJNzVUX2dFWGcwYm5vZFN4eUxuOU8yVzFGR1F3' \
  -H 'cache-control: no-cache' \
  -H 'content-type: application/x-www-form-urlencoded' \
  -d 'grant_type=authorization_code&code=KhYAjeCwU3BOchZZhFVr-
ukqFOrc4jAF_9ZUwMhQU32&redirect_uri=https://<sistema-cliente>/callback'
```

Ejemplo de solicitud si el tipo de autenticación es `post`:

```
curl -X POST \
  https://<base-url-proveedor-identidad>/token \
  -H 'cache-control: no-cache' \
  -H 'content-type: application/x-www-form-urlencoded' \
  -d 'grant_type=authorization_code&code=1LVvS3_X8QdIOJejBtAmvBEUzdAy-
QYTtIb4SpEY29c&redirect_uri=https://<sistema-cliente>/callback&client_id=W5ulM-
2xLKtXpBxahfTQQ&client_secret=lY9MSMzTTn0tp4avTBrQ44CnicKpoLEATG-QfcyJWOI1kIrVjqpcqPKps8-
eMruYqd6vdzu6RiuL7Q9inOwkpQ'
```

Ejemplo de solicitud si la aplicación es movil:

```
curl -X POST \
  https://<base-url-proveedor-identidad>/token \
  -H 'content-type: application/x-www-form-urlencoded' \
```

```
  -d 'grant_type=authorization_code&redirect_uri=net.openid.appauthdemo%3A
%2Foauth2redirect&code_verifier=2AcLLc82Tdu8HUESuVxJeL28DGavoDQfpJGSjLLC3FfJfpwtWR0efZaTUgHRL3wTvkwU
J9nwuTd9QXSA&code=64GrCF1ps00cU1d1jp5d8mDlQjYRhojH6JEMn6yu5l5&client_id=yhOiNSDcWeXQEz2Hs7DeW'
```

En caso de éxito el Proveedor de identidad devolverá un objeto JSON con los siguientes parámetros:

Parámetro	Descripción
access_token	Es un token que servirá acceder a los recurso protegidos.
expires_in	Tiempo de vida del token de acceso.
id_token	Token de identidad.
scope	scopes a los que se tiene permiso.
token_type	Identifica el tipo de token devuelto.

Ejemplo de respuesta:

```
{
    "access_token": "6m4IKQt3M98vLu4qWTwyNlcMk5ZVXV_fITSDXLuke30",
    "expires_in": 86400,
    "id_token": "eyJhbGciOiJSUzI1NiIsInR5...", // JWT
    "scope": "openid profile",
    "token_type": "Bearer"
}
```

Refresh token

Los tokens de actualización o refresh tokens se pueden usar para obtener un nuevo token de acceso cuando el que se tenía previamente ya ha caducado. Por esta naturaleza, es útil asimismo para acceder a los recursos protegidos sin que el usuario esté presente y, en algunos casos, se pueden otorgar permisos de acceso sin que el usuario tenga que autenticarse nuevamente.

Para acceder a un token de actualización **es necesario haber solicitado éste en el proceso de registro del sistema cliente**, entonces, sólo si la mencionada condición es satisfecha:

- Cuando crea la solicitud de autenticación debe adicionaren el parametro scope `offline_access` y el parametro `prompt=consent`.

Ejemplo de solicitud de autenticación:

```
https://<base-url-proveedor-identidad>/auth?
response_type=code&client_id=mGntc6oKqoFMCRZArKDad&state=509ccc2713049e6efea071a9c34f6f45&nonce=2313
01a1afe20d88ca963ee84c3929c3&redirect_uri=https://<sistema-cliente>/callback&scope=openid%20profile
%offline_access&prompt=consent
```

- Para solicitar un nuevo `access_token` utilizando un `refresh_token` enviar una petición POST via HTTPS con los parámetros necesarios, es enviado de manera similar al token endpoint, visto en el apartado anterior.

La url donde se debe enviar la solicitud POST es la siguiente:

```
https://<base-url-proveedor-identidad>/token
```

Los parámetros necesarios que se debe enviar son:

Parámetro	tipo	Condición	Descripción
Authorization	header	`requerido` si el metodo de autenticacion del cliente es `basic`	El id_client y secret (ambos obtenidos en el registro del sistema cliente) deben ser pasados en la cabecera mediante la autenticación básica (el secret debe estar codificado con urlEncode)
grant_type	body	requerido	valor: `refresh_token`
refresh_token	body	requerido	Valor del refresh token
client_id	body	requerido si el tipo de autenticaion del cliente es `post` o es aplicacion movil	Identificador del cliente, obtenido al crear el cliente
client_secret	body	requerido si el tipo de autenticacion del cliente es `post`	Credencial del sistema cliente obtenido en el registro

Ejemplo de solicitud si el tipo de autenticación es `basic`:

```
curl -X POST \
  https://<base-url-proveedor-identidad>/token \
  -H 'authorization: Basic
LVVhdmYzUTYxTEotOEU0T0VzZWFDOnh2bl8zRmtURTlvV1gxbHMzOFRnR2Q4Ymx6QzNtRXNxbVFyVXNBbXhQR3I3MC1hZnlwTWR4
US1iQXFSWjhnRUFzbmJ4eElSVGJVYnBXQnZ5dTExYy1n' \
  -H 'cache-control: no-cache' \
  -H 'content-type: application/x-www-form-urlencoded' \
  -d 'grant_type=refresh_token&refresh_token=mpTDZhhH5WnxOFidb8uZh1vwrJX3N9M4XMAPYQ0ewgb'
```

Ejemplo de solicitud si el tipo de autenticación es `post`:

```
curl -X POST \
  https://<base-url-proveedor-identidad>/token \
  -H 'cache-control: no-cache' \
  -H 'content-type: application/x-www-form-urlencoded' \
  -d
'grant_type=refresh_token&refresh_token=mpTDZhhH5WnxOFidb8uZh1vwrJX3N9M4XMAPYQ0ewgb&client_id=W5ulM-
2xLKtXpBxahfTQQ&client_secret=lY9MSMzTTn0tp4avTBrQ44CnicKpoLEATG-QfcyJWOI1kIrVjqpcqPKps8-
eMruYqd6vdzu6RiuL7Q9inOwkpQ'
```

En caso de éxito el Proveedor de Identidad devolverá un objeto JSON con los siguientes parámetros:

Parámetro	Descripción
access_token	Es un token que servirá acceder a los recurso protegidos.
expires_in	Tiempo de vida del token de acceso.
id_token	Token de identidad.
refresh_token	Token de actuaización.
scope	scopes a los que se tiene permiso.
token_type	Identifica el tipo de token devuelto.

Ejemplo de respuesta:

```
{
    "access_token": "OyJM-ikIfrjOSQr98u6_eyznfDKJlOSpu-6Vc5PU79m",
    "expires_in": 86400,
    "id_token": "eyJhbGciOiJSUzI1NiIsInR5c...", // JWT
    "refresh_token": "mpTDZhhH5WnxOFidb8uZh1vwrJX3N9M4XMAPYQ0ewgb",
    "scope": "openid offline_access profile",
    "token_type": "Bearer"
}
```

Token introspection

La extensión de introspección de tokens define un mecanismo para que los servidores de recursos protegidos obtengan información sobre los tokens de acceso. Con esta especificación, los servidores de recursos pueden verificar la validez de los tokens de acceso y encontrar otra información, como qué usuario y qué ámbitos están asociados con el token.

Enviar una petición POST via HTTPS con los parámetros necesarios.

La url donde se debe enviar la solicitud POST es la siguiente:

```
https://<base-url-proveedor-identidad>/token/introspection
```

Los parámetros necesarios que se debe enviar son:

Parámetro	tipo	Condición	Descripción
Authorization	header	`requerido` si el metodo de autenticacion del cliente es `basic`	El id_client y secret (ambos obtenidos en el registro del sistema cliente) deben ser pasados en la cabecera mediante la autenticación básica (el secret debe estar codificado con urlEncode)
token	body	requerido	Valor del access token
client_id	body	requerido si el tipo de autenticaion del cliente es `post` o es aplicacion movil	Identificador del cliente, obtenido al crear el cliente
client_secret	body	requerido si el tipo de autenticacion del cliente es `post`	Credencial del sistema cliente obtenido en el registro

Ejemplo de solicitud:

```
curl -X POST \
  https://<base-url-proveedor-identidad>/token/introspection \
  -H 'authorization: Basic
LVVhdmYzUTYxTEotOEU0T0VzZWFDOnh2bl8zRmtURTlvV1gxbHMzOFRnR2Q4Ymx6QzNtRXNxbVFyVXNBbXhQR3I3MC1hZnlwTWR4
US1iQXFSWjhnRUFzbmJ4eElSVGJVYnBXQnZ5dTExYy1n' \
  -H 'cache-control: no-cache' \
  -H 'content-type: application/x-www-form-urlencoded' \
  -d token=OyJM-ikIfrjOSQr98u6_eyznfDKJlOSpu-6Vc5PU79m
```

En caso de éxito el Proveedor de Identidad devolverá un objeto JSON con los siguientes parámetros:

Parámetro	Descripción
active	Indica si el token resentado esta activo o no, valor booleano
sub	Identificador del usuario que autorizó el token
client_id	Identificador del cliente al que se emitio el token
exp	Marca de tiempo que indica cuándo caducará este token.
iat	Marca de tiempo cuando se genero el token
iss	proveedor de identidad
scope	scopes a los que se tiene permiso.
token_type	Identifica el tipo de token devuelto.

Ejemplo de respuesta:

```
{
    "active": true,
    "sub": "274047a7-649a-4b7c-8a8...",
    "client_id": "-Uavf3Q61LJ-8E4OEseaC",
    "exp": 1600480389,
    "iat": 1600393989,
    "iss": "https://<base-url-proveedor-identidad>",
    "scope": "openid profile",
    "token_type": "Bearer"
}
```

Errores

Los errores pueden suscitarse en la interacción con el proveedor por dos canales:

- Frontend channel
- Backend channel

Frontend channel

Cuando las acciones se realizan en un navegador. En este caso, siempre que sea posible, se redirige al usuario a la URL de redirección y se adjunta a ésta el error que debe ser parseado por el sistema cliente. Por ejemplo: `https://example-app.com/cb?error=invalid_scope`.

Se devuelve el error y un campo `error_description`, que no debe ser mostrado al usuario. En vez, el sistema cliente debe mostrar su propio error en función a la documentación del protocolo. El error que muestre el sistema cliente debe ayudar al usuario a qué acción tomar.

Authorization request errors:

Client_id no reconocido:

Código: `invalid_client: client_id`
Código HTTP: `400`
Descripción: `Client_id` inválido porque el enviado no se encuentra en los registros del servidor.
¿Redirige?: No
Dominio donde se muestra el error: proveedor, no se puede redirigir.
Acciones a tomar por el cliente: Debe proporcionar el `client_id` que se le ha otorgado cuando ha registrado el cliente.
Acciones a tomar por el usuario: Ninguna
error_description: `client is invalid`

Url de redirección inválida:

Código: `redirect_uri_mismatch`
Código HTTP: `400`

Descripción: Url de redirección inválida porque la enviada no coincide con la registrada
¿Redirige?: No
Dominio donde se muestra el error: proveedor, no se puede redirigir.
Acciones a tomar por el cliente: Debe proporcionar la url de redirección correcta, exactamente igual que se ha registrado.
Acciones a tomar por el usuario: Ninguna
error_description: `redirect_uri did not match any of the client\'s registered redirect_uris`

***El Usuario cancela la solicitud*:**
Código: `access_denied`
Código HTTP: `302`
Descripción: El usuario ha cancelado la solicitud, interrumpiendo el flujo.
¿Redirige?: Sí
Dominio donde se muestra el error: cliente
Acciones a tomar por el cliente: Puede mostrar un mensaje que indique que la operación ha sido cancelada.
Acciones a tomar por el usuario: Ninguna
error_description: `End-User aborted interaction`

REPONSE_TYPE no soportado:
Código: `unsupported_response_type`
Código HTTP: `302`
Descripción: El `response_type` no es soportado por el proveedor
¿Redirige?: Sí
Dominio donde se muestra el error: cliente
Acciones a tomar por el cliente: Revisar el `response_type` enviado en la solicitud, que debe ser igual al valor `code`.
Acciones a tomar por el usuario: Ninguna
error_description: `unsupported response_type requested`

Solicitud de scopes invalido:
Código: `invalid_scope_error`
Código HTTP: `302`
Descripción: Alguno de los scopes solicitados no se encuentra en la lista de scopes solicitada en la creación del cliente.
¿Redirige?: Sí
Dominio donde se muestra el error: cliente

Acciones a tomar por el cliente: Revisar la lista de scopes permitidos para el cliente, enviar scopes válidos.
Acciones a tomar por el usuario: Ninguna
error_description: `requested scope is not whitelisted`

Backend channel

Cuando las acciones se realizan dentro del servidor del sistema cliente hacia el proveedor de identidad. En este caso los errores deben ser atendidos por el sistema cliente y no es recomendable que sean presentados de la misma manera al usuario final, si fuera el caso.

Token request errors

Cliente invalido:

Código: `invalid_client`
Código HTTP: `401`
Descripción: La autenticación del cliente falló
Acciones a tomar por el cliente: Debe proporcionar las credenciales correctas del cliente, si la autenticación del cliente es tipo `basic` verificar que el secreto del cliente (`client_secret`) se está enviando codificado con `urlencode`.
Acciones a tomar por el usuario: Ninguna
error_description: `client authentication failed`

GRANT_TYPE no soportado:

Código: `unsupported_grant_type`
Código HTTP: `400`
Descripción: El `grant_type` no es soportado por el proveedor
Acciones a tomar por el cliente: Debe proporcionar el `grant_type` correcto. Si está solicitando `access_token` enviar `authorization_code`, si está solicitando `refresh_token` enviar el valor `refresh_token`.
Acciones a tomar por el usuario: Ninguna
error_description: `unsupported grant_type requested`

Parámetro code o refresh token invalido:

Código: `invalid_grant`
Código HTTP: `400`
Descripción: El parámetro `code` o `refresh token` es incorrecto. Esto puede deberse a que el valor del parámetro es invalido, expiró o ya fue consumido.
Acciones a tomar por el cliente: Debe proporcionar un valor válido del parámetro.

Acciones a tomar por el usuario: Ninguna
error_description: `grant request is invalid`

Userinfo request errors

Token invalido:

Código: `invalid_token`
Código HTTP: `401`
Descripción: El `access_token` enviado es inválido sea porque no es igual al generado o porque ha expirado
Acciones a tomar por el cliente: Debe proporcionar un token válido.
Acciones a tomar por el usuario: Ninguna
error_description: `invalid token provided`