**OpenID Connect**

OpenID Connect es un protocolo de identidad simple y de estándar abierto creado sobre el protocolo OAuth 2.0 que permite a las aplicaciones cliente confiar en la autenticación realizada por un proveedor de OpenID Connect para verificar la identidad de un usuario.

OpenID Connect utiliza OAuth 2.0 para la autenticación y la autorización y, a continuación, crea las identidades que identifican a los usuarios de forma exclusiva. Las aplicaciones cliente también pueden obtener información básica de perfil acerca de un usuario de modo interoperativo de tipo REST de los proveedores de OpenID Connect.

OpenID Connect es un protocolo de autenticación basado en la especificación en OAuth 2.0. Mediante el uso de un token JWT la aplicación puede obtener información del usuario según el flujo definido en las especificaciones OAuth 2.0. Para dar un marco inicial y simplificado, cada vez que los clientes ingresen a una aplicación web que se encuentre federada con OpenID Connect y no cuenten con una sesión disponible en la aplicación, será redirigido al OpenID Provider para realizar el proceso de autenticación. Una vez que el proceso de login sea exitoso retornará a la aplicación la información del cliente en  
formato JSON, llamado ID Token. Si bien OpenID Connect soporta distintos flujos de autenticación, el presente documento se basa en la autenticación mediante el flujo Authorization Code.

**Referencias**

|  |  |
| --- | --- |
| **Estándares OAuth 2.0** | |
| **RFC** | **Descripción** |
| [http://The OAuth 2.0 Authorization Framework [RFC 6749]](https://tools.ietf.org/html/rfc6749) | Define las especificaciones del framework OAuth 2.0 |
| [http://JSON Web Token (JWT) [RFC 7519](https://tools.ietf.org/html/rfc7519) | Define las especificaciones del JSON Web Token |
| [JSON Web Algorithms (JWA) [RFC 7518]](https://tools.ietf.org/html/rfc7518) | Define las especificaciones de los algoritmos para la firma del JWT |
| https://tools.ietf.org/html/rfc7515 | Define las especificaciones de la firma del JWT |

|  |  |
| --- | --- |
| **Especificaciones OpenId Connect** | |
| **Especificación** | **Descripción** |
| [OpenID Connect Core 1.0](http://openid.net/specs/openid-connect-core-1_0.html) | Define las especificaciones de OpenID Connect 1.0 |
| [OpenID Connect Discovery 1.0](http://openid.net/specs/openid-connect-discovery-1_0.html) | Define como los Clients obtienen la información del OpenID provider |

Se presenta las especificaciones para la federación de aplicaciones Web de Telecom Personal mediante protocolo Open ID Connect (OIDC).

Se detallan las especificaciones y los casos de uso para la federación de aplicaciones Web mediante el protocolo OpenID Connect (OIDC). Bajo este escenario se delega la autenticación de los clientes, identificados mediante el número de linea y pin, a la plataforma del IDP (en este caso Access Manager de Forgerock) que cumple el rol de OpenID Provider (OP). De esta manera las distintas aplicaciones Web no almacenarán, ni tendrán conocimiento de las credenciales del usuario, ya que esta responsabilidad estará centralizada en la plataforma de Access Management.

**OpenId Connect**

[OpenID Connect](http://openid.net/specs/openid-connect-core-1_0.html) es un protocolo de autenticación basado en la especificación en [OAuth 2.0](https://tools.ietf.org/html/rfc6749). Mediante el uso de un token JWT la aplicación puede obtener información del usuario según el flujo definido en las especificaciones OAuth 2.0. Para dar un marco inicial y simplificado, cada vez que los clientes ingresen a una aplicación web que se encuentre federada con OpenID Connect y no cuenten con una sesión disponible en la aplicación, será redirigido al OpenID Provider para realizar el proceso de autenticación. Una vez que el proceso de login sea exitoso retornará a la aplicación la información del cliente en  
formato JSON, llamado ID Token. Si bien OpenID Connect soporta distintos flujos de autenticación, el presente documento se basa en la autenticación mediante el flujo Authorization Code.

**Nomenclatura**

* *Authorization Server*: también referenciado en el documento como OpenID Provider (OP). Servicio encargado de autenticar a un usuario, identificando la aplicación solicitante, y generar un token JWT con la información de la identidad.
* *Relay Party*: Representa a la aplicación que va a federación contra el OpenID Provider. Es la parte que confía. Es la aplicación protegida, que recurre al proveedor de autenticación para autenticar a un usuario.
* *Línea / PIN*: es el mecanismo de autenticación actual, donde la línea corresponde a un número telefónico, y el PIN corresponde a una clave numérica generada por el cliente para asegurar que es quien dice ser.
* User Agent: En este caso hace referencia al navegador web.
* Token JWT: es la abreviatura de JSON Web Token. Definido por el [RFC 7519], se utiliza en el contexto de Open ID Connect para enviar datos entre aplicaciones o servicios y garantizar que sean válidos y seguros. Estos tokens van firmados por el emisor para asegurar su autenticidad.
* Nonce: string generado por la aplicación y enviado en la llamada de autenticación al Authorization Server, el mismo es utilizado por la aplicación para certificar que la respuesta corresponde a la petición inicial.
* *State*: string utilizado único generado por la aplicación asociado a la sesión del usuario. El mismo es usado para prevenir el ataque por Cross-Site Request Forgery [RFC6749] (CSRF)
* *Scope*: Parámetro que representa la información a solicitada por la aplicación al Authorization Server. Las aplicaciones especifican el scope con valor openid para establecer que es llamada OpenIDConnect y también lo utilizan como para solicitar claims.
* *Claim*: Representa la información a obtener del usuario final en token JWT, en este caso, el cliente se encuentra asociado a una linea. De acuerdo al estándar OpenID Connect también se encuentran los  claims .

**Diagrama de Flujo Authorization Code**

|  |
| --- |
| **https://docuit.personal.corp/download/attachments/14779004/image2018-1-26%2014%3A56%3A35.png?version=1&modificationDate=1516989729000&api=v2** |
| Diagrama de flujo Authorization Code |

**Información OIDC requerida**

Dentro del contexto de una aplicación que va a realizar la federación OpenID Connect es requerido que se provea de la siguiente información.

**Datos de la aplicación**

Bajo el escenario descripto en este documento, cada aplicación debe estar registrada en el OpenID Provider para poder identificarse y autenticarse en el proceso de solicitud de token, debido a esto la aplicación debe poseer un client\_id y el client\_secret.   
Por otra parte, a momento que se registre la aplicación en el OpenID Provider, la misma debe informar el parámetro de redirección redirect\_uri, el mismo es utilizando por el Authorization Server para devolver el Authorization Code como se ve en el Diagrama de flujo Authorization Code.  
  
A modo de resumen se lista a continuación los parámetros de configuración básica de la aplicación.

|  |  |
| --- | --- |
| **Parámetros** | **Descripción** |
| client\_id | Identificador de la aplicación |
| client\_secret | Password de la aplicación |
| redirect\_uri | URL redirección de la aplicación utilizada por Authorization Server para retornar la respuesta en la solicitud del Authentication Request |
| scope | Información a solicitar en el token |

**EndPoint OpenId Provider Metadata**

Por otro lado como buena práctica recomendada, las aplicaciones deben consultar el endpoint OIDC Provider Metadata definido en el estandar [OpenID Discovery](http://openid.net/specs/openid-connect-discovery-1_0.html) para tomar la configuración del OpenID Provider, a lo largo del documento la información de configuración de OP la llamaremos Discovery Document. Considerando lo antes mencionado, las aplicaciones deben realizar un haciendo un HTTPS GET al endpoint  especificado, por ejemplo, .well-known/openidconfiguration.

Ejemplo: En el caso de Access Management como Identity Provider, puede obtenerse de:

Realm tabpins:

[https://sesiond.telecom.com.ar/openam/oauth2/realms/root/realms/tabpins/.well-known/openid-configuration](http://openamdesa1.personal.com.ar:8080/openam/oauth2/realms/root/realms/tabpins/.well-known/openid-configuration)

[https://sesiont.telecom.com.ar/openam/oauth2/realms/root/realms/tabpins/.well-known/openid-configuration](http://openamdesa1.personal.com.ar:8080/openam/oauth2/realms/root/realms/tabpins/.well-known/openid-configuration)

|  |
| --- |
| **well-known/openidconfiguration** |
| {"request\_parameter\_supported":true,"claims\_parameter\_supported":false,"introspection\_endpoint":"https://sesiont.telecom.com.ar:443/openam/oauth2/realms/root/realms/tabpins/introspect","check\_session\_iframe":"https://sesiont.telecom.com.ar:443/openam/oauth2/realms/root/realms/tabpins/connect/checkSession","scopes\_supported":["openid","profile"],"issuer":"https://sesiont.telecom.com.ar:443/openam/oauth2/tabpins","id\_token\_encryption\_enc\_values\_supported":["A128CBC-HS256","A256CBC-HS512"],"acr\_values\_supported":[],"authorization\_endpoint":"https://sesiont.telecom.com.ar:443/openam/oauth2/realms/root/realms/tabpins/authorize","request\_object\_encryption\_enc\_values\_supported":["A256GCM","A192GCM","A128GCM","A128CBC-HS256","A192CBC-HS384","A256CBC-HS512"],"rcs\_request\_encryption\_alg\_values\_supported":["RSA-OAEP","RSA-OAEP-256","A128KW","RSA1\_5","A256KW","dir","A192KW"],"claims\_supported":["num\_line","status"],"rcs\_request\_signing\_alg\_values\_supported":["ES384","HS256","HS512","ES256","RS256","HS384","ES512"],"token\_endpoint\_auth\_methods\_supported":["client\_secret\_post","private\_key\_jwt","client\_secret\_basic"],"token\_endpoint":"https://sesiont.telecom.com.ar:443/openam/oauth2/realms/root/realms/tabpins/access\_token","response\_types\_supported":["code token id\_token","code","code id\_token","device\_code","id\_token","code token","token","token id\_token"],"request\_uri\_parameter\_supported":true,"rcs\_response\_encryption\_enc\_values\_supported":["A256GCM","A192GCM","A128GCM","A128CBC-HS256","A192CBC-HS384","A256CBC-HS512"],"end\_session\_endpoint":"https://sesiont.telecom.com.ar:443/openam/oauth2/realms/root/realms/tabpins/connect/endSession","rcs\_request\_encryption\_enc\_values\_supported":["A256GCM","A192GCM","A128GCM","A128CBC-HS256","A192CBC-HS384","A256CBC-HS512"],"version":"3.0","rcs\_response\_encryption\_alg\_values\_supported":["RSA-OAEP","RSA-OAEP-256","A128KW","A256KW","RSA1\_5","dir","A192KW"],"userinfo\_endpoint":"https://sesiont.telecom.com.ar:443/openam/oauth2/realms/root/realms/tabpins/userinfo","id\_token\_encryption\_alg\_values\_supported":["RSA1\_5"],"jwks\_uri":"https://sesiont.telecom.com.ar:443/openam/oauth2/realms/root/realms/tabpins/connect/jwk\_uri","subject\_types\_supported":["public"],"id\_token\_signing\_alg\_values\_supported":["ES384","HS256","HS512","ES256","RS256","HS384","ES512"],"registration\_endpoint":"https://sesiont.telecom.com.ar:443/openam/oauth2/realms/root/realms/tabpins/register","request\_object\_signing\_alg\_values\_supported":["ES384","HS256","HS512","ES256","RS256","HS384","ES512"],"request\_object\_encryption\_alg\_values\_supported":["RSA-OAEP","RSA-OAEP-256","A128KW","RSA1\_5","A256KW","dir","A192KW"],"rcs\_response\_signing\_alg\_values\_supported":["ES384","HS256","HS512","ES256","RS256","HS384","ES512"]} |

El contenido de la configuración retornada es en formato JSON. Utilizando este mecanismo donde se puede tomar la información de manera dinámica. La aplicación no debería tener información del OpenID Provider hard-coded. A continuación se muestran alguno de los parámetros básicos de la configuración presentada.

|  |  |
| --- | --- |
| **Key** | **Descripcion** |
| authorization\_endpoint | URL del Authorization Endpoint |
| token\_endpoint | URL del Token Endpoint |
| userinfo\_endpoint | URL del UserInfo Endpoint |
| issuer | URL del identificador del issuer |
| jwks\_uri | URL del JSON Web Key Set. Permite a las aplicaciones consultar el certificado (clave pública) del OpenID Provider para poder validar la firma del ID Token |
| id\_token\_signing\_alg\_values\_supported | JSON array con la lista de algoritmos de firma del ID Token |

Dirigirse a [OpenID Provider Metadata](http://openid.net/specs/openid-connect-discovery-1_0.html" \l "ProviderMetadata) donde se especifica el resto de la información disponible basado en el estándar OpenID Connect.

**CASOS DE USO FEDERACION DE APLICACIONES**

**CASOS DE USO**

En esta página se detallan las especificaciones para que una aplicación pueda federar vía protocolo OpenID Connect (OIDC) contra el IDP. El proceso de autenticación de los clientes se realiza mediante el número de linea y pin.

Se analizan dos escenarios posibles:

|  |
| --- |
| **CASO DE USO: ESCENARIO APLICACION WEB** |
| *En este se escenario se posee una aplicación web tradicional. Si bien OpenID Connect soporta distintos flujos de autenticación,*  *se recomienda optar por el* ***flujo Authorization Code****, ya que como se puede ver en el diagrama no queda  información sensible del*  *lado del User Agent.* |
| **CASO DE USO: ESCENARIO APLICACION SPA (SINGLE PAGE APPLICATION)** |
| *En el caso de una aplicación SPA (Single Page Application) en donde la misma se ejecuta de lado del User Agent (Browser),*  *por ejemplo, una aplicación desarrollada con lenguaje Javascript. En este contexto se optar por utilizar el* ***flujo Implicit.*** En lo que  *respecta a los pasos del flujo de autenticación, la aplicación directamente solicita los tokens, sin realizar el paso de solicitar el*  *Authorization code como el caso anterior, con lo cual se reduce el número de llamadas.*  **Consideraciones: En el flujo Implicit los tokens (access token y ID token) son expuestos en el User Agent (Browser),** **por lo requiere de precauciones adicionales para mitigar cualquier modificación o substitución del mismo.** |

**DIAGRAMAS**

|  |
| --- |
| **DIAGRAMA CASO DE USO: ESCENARIO APLICACION WEB** |
| **https://docuit.personal.corp/download/attachments/14779160/image2018-2-20%2015%3A42%3A33.png?version=1&modificationDate=1519152587000&api=v2** |

.

El proceso se inicia cuando el modelo de seguridad de la aplicación detecta que se está intentando acceder a un recurso protegido y no se cuenta con una sesión activa que autorice el acceso.

**1**. La aplicación (Relay Party) inicia el proceso de autenticación (Authentication Request) mediante una redirección en el Navegador (User Agent) al OpenID Provider (OP).  
**2**. El OpenID Provider procede a realizar el proceso de autenticación.  
**2.1**. Si no tiene un sesión valida , se le solicita número de linea y pin para realizar la autenticación del usuario. Por otro lado si se trata del primer login, el Authorization Server le solicitará el consentimiento para los permisos que la aplicación este solicitando.  
**2.2**. Si ya existe una sesión, el usuario no interviene en el proceso ya que se encuentra autenticado.  
**3**. Luego del proceso de autenticación exitoso , el OpenID Provider realiza una redirección (redirect\_uri) hacia la aplicación con el code (Authorization Code) en el Query String.  
**4**. Finalmente la aplicación solicita un Token enviando el code y las credenciales de la aplicación (client\_id y client\_secret).  
**4.1**. El OpenID Provider autentica la aplicación y valida el Authorization Code enviado. Si es válido, devuelve el token (access\_token y id\_token) a la aplicación.  
Por último la aplicación construye sesión tomando la información de ID Token y devuelve el contenido solicitado al usuario.

|  |
| --- |
| **DIAGRAMA CASO DE USO: ESCENARIO APLICACION SPA** |
| **https://docuit.personal.corp/download/attachments/14779160/image2018-2-20%2016%3A4%3A58.png?version=1&modificationDate=1519153932000&api=v2** |

**1.** La aplicación (Relay Party) inicia el proceso de autenticación (Authentication Request) mediante una redirección en el Navegador (User Agent) al OpenID Provider (OP).  
**2**. El OpenID Provider procede a realizar el proceso de autenticación.  
**2.1**. Si no tiene un sesión valida , se le solicita número de linea y pin para realizar la autenticación del usuario. Por otro lado si se trata del primer login, el Authorization Server le solicitará el consentimiento para los permisos (scopes) que la aplicación este solicitando.  
**2.2.** Si ya existe una sesión, el usuario no interviene en el proceso ya que se encuentra autenticado.  
**3.** Luego del proceso de autenticación exitoso, el OpenID Provider realiza una redirección (redirect\_uri) con los tokens (access\_token y id\_token) a la aplicación.  
Por último la aplicación toma la información de ID Token y devuelve el contenido solicitado al usuario.

**DIAGRAMAS DETALLADOS**

**DIAGRAMA DETALLADO FLUJO AUTHORIZATION CODE.**

Se muestra de manera detallada la interacción de los distintos componentes en la negociación de un token. Para el siguiente escenario, se tomó como ejemplo una aplicación Web que denominamos Store y se bajo a  un mayor de nivel de detalle las llamadas a los distintos endpoints presentados por la solución de Forgerock Access Manager. Por otro lado, los casos de usos que se describen tienen como foco la definición y  especificación desde el punto de vista de la aplicación y los pasos a seguir para poder federar siguiendo el estándar OpenID Connect utilizando el flujo de autenticación Authorization Code. De esta manera las distintas aplicaciones Web de Telecom Personal no almacenarán, ni tendrán conocimiento de las credenciales del usuario, ya que esta responsabilidad estará centralizada en la plataforma de Access Management.

|  |
| --- |
| **https://docuit.personal.corp/download/attachments/14779160/image2018-1-29%2011%3A52%3A15.png?version=1&modificationDate=1517237881000&api=v2** |
| Diagrama de flujo detallado flujo authorization code |

**SOLICITUD DE AUTENTICACIÓN**  
El proceso se inicia cuando el modelo de seguridad de la aplicación detecta que se está intentando acceder a un recurso protegido y no se cuenta con una sesión activa que autorice el acceso. La aplicación realiza una llamada HTTPS GET a la URL descipta en la key authorization\_endpoint del Discovery Document. En este caso el valor es /openam/oauth2/realms/root/realms/tabpins/authorize.  
La llamada que figura en el **PASO 1**del diagrama de flujo detallado, está compuesta de los siguientes campos:

|  |  |  |
| --- | --- | --- |
| **Parámetro** | **Requerido** | **Descripción** |
| client\_id | SI | Identificador de la aplicación (Client) |
| response\_type | SI | Authorization grant, para este caso colocar el valor code ya que se va solicitar un Authorization Code |
| scope | SI | El scope debe contener el valor openid para especificar que una llamada OpenID Connect. Y adicionalmente para este ejemplo el valor profile |
| nonce | SI | string generado por la aplicación y enviado en la llamada de autenticación al Authorization Server, el mismo es utilizado por la aplicación para certificar que la respuesta corresponde a la petición inicial. |
| redirect\_uri | SI | URL de la aplicación donde el Authorization Server envía la respuesta de la solicitud de autenticación con el Authorization Code. |
| state | Opcional, pero es recomendado | Token único generado en la aplicación asociado a la sesión del usuario para prevenir el ataque por Cross-site request forgery (CSRF). Para mas información dirigirse al [OAuth 2.0 [RFC6749]](https://tools.ietf.org/html/rfc6749#section-10.12) |
| login\_hint | Opcional | En caso de que la aplicación conozca el identicador del usuario que se va a autenticar, puede especificarse en la llamada al Authentication Server |

A continuación se muestra un ejemplo la llamada de autenticación del **PASO 2**

|  |
| --- |
| **Llamada de autenticación** |
| GET /authorize? response\_type=code& client\_id=<client\_id>& redirect\_uri=<redirect\_uri>& scope=openid+<scopes>& state=<state\_string>& nonce=<nonce\_string> Host:<authorization\_server> |

|  |
| --- |
| **Llamada de autenticación para el ejemplo client\_id=store** |
| <http://demostore1.telecom.com.ar:8080/demo-app/>  <http://openamdesa1.personal.com.ar:8080/openam/XUI/?realm=%2Ftabpins&goto=http%3A%2F%2Fopenamdesa1.personal.com.ar%3A8080%2Fopenam%2Foauth2%2Frealms%2Froot%2Frealms%2Ftabpins%2Fauthorize%3Fresponse_type%3Dcode%26client_id%3Dstore%26redirect_uri%3Dhttp%253A%252F%252Fdemostore1.telecom.com.ar%253A8080%252Fdemo-app%252Foauth%252Fcallback%26scope%3Dopenid%26state%3DIZxk8nZzmnECXQngk0LP0r_smpPrtHkOH52mBUZmgww%26nonce%3DcfLalfOhDU3z96Rr8TVdn-ozcLwhAWLMS06WU2n_FIM#login/> |

Luego continua el proceso de autenticación y authorización del usuario en el Authorization Server (**PASO 3 y 4**). Como se puede apreciar en este proceso, dado que la respuesta del Authorization Server es asíncrona, la aplicación debe almacenar cierta información asociada al usuario que inició el proceso, para luego poder recuperarla, por ejemplo de la sesión. Los datos que deberá almacenar son: nonce y status.

**VALIDACIÓN RESPUESTA DE LA AUTENTICACIÓN**

Si el proceso de autenticación termina forma exitosa, el Authorization Server envía una redirección (**PASO 5**) la respuesta al redirect\_uri especificado en la llamada anterior. Como se pueden observar en el PASO 6 del diagrama de flujo detallado, los datos de la respuesta vienen en formato Query String:

|  |
| --- |
| **https://<redirect\_uri>?code=<authorization\_code>&state=<state\_string>** |

Para nuestro caso de uso la respuesta del Authorization Server es la siguiente:

|  |
| --- |
| [**http://demostore1.telecom.com.ar:8080/demo-app/callback?code=f171926b-d606-4e1e-aab8-**](https://store.personal.com.ar/callback?code=f171926b-d606-4e1e-aab8-)**7879cc6114d2&state=bUUFJ0KtRjKcPD65NPOZ3Hee5YcqrGwi-1hFQ-pmWFA** |
| respuesta del Authorization Server |

Como próximo paso la aplicación debe confirmar que parámetro state recibido de la respuesta Authorization Server (**PASO 7**), es el mismo que fue creado en la llamada de solicitud de autenticación (PASO 1).

**SOLICITUD DE TOKEN**

La respuesta del **PASO 7** incluye el parámetro code , que es un one-time authorization code para poder solicitar el token. La aplicación de realizar una llamada HTTPS POST al token endpoint de acuerdo a la key token\_endpoint como figura en el **PASO 8**. En este caso el valor es /openam/oauth2/realms/root/realms/tabpins/access\_token.

Los siguientes parámetros deben estar incluidos en el POST body de la llamada:

|  |  |
| --- | --- |
| **Campo** | **Descripción** |
| code | El Authorization code retornado por el Authorization Server (PASO 6) |
| client\_id | Identificador de la aplicación |
| client\_secret | Password de la aplicación |
| redirect\_uri | La URI especificada como **redirect\_uri** en la llamada inicial (PASO 1) |
| grant\_type | Debe contener el valor **authorization\_code** |

La llamada de solicitud de token debe poseer los siguientes parámetros, el código se encuentra indentado y con saltos de lineas para facilitar la lectura:

|  |
| --- |
| **POST <url\_token\_endpoint> HTTP/1.1 Host: <authorization\_server> Content-Type: application/x-www-form-urlencoded code=<authorization\_code>& client\_id=<client\_id>& client\_secret=<client\_secret>& redirect\_uri=<redirect\_uri>& grant\_type=authorization\_code** |

 En nuestro caso la llamada es la siguiente:

|  |
| --- |
| **POST openam/oauth2/realms/root/realms/tabpins/access\_token HTTP/1.1 Host: demostore1.telecom.com.ar Content-Type: application/x-www-form-urlencoded code=f171926b-d606-4e1e-aab8-7879cc6114d2& client\_id=store& client\_secret=p4ssw0rd& redirect\_uri=**[**http://demostore1.telecom.com.ar:8080/demo-app/oauth/callback**](http://demostore1.telecom.com.ar:8080/demo-app/oauth/callback)[**&**](https://store.personal.com/callback&) **grant\_type=authorization\_code** |

La respuesta (**PASO 9**) en formato JSON array contien los siguientes campos.

|  |  |
| --- | --- |
| **Campo** | **Descripción** |
| access\_token | String que representa la autorización emitida para el client. |
| id\_token | JWT token que contiene la información del usuario firmado por el Authorization Server |
| expires\_in | Tiempo restante para la expiración del token |
| token\_type | Representa el tipo de token. En este caso **Bearer** |
| refresh\_token | Opcional. Permiten obtener nuevos tokens, cuando el access token expira o es inválido |

**DIAGRAMA DETALLADO APLICACION SPA**

En el presente ejemplo, se trata de una aplicación javascript denominada contenido, la cual corre en el User Agent (Browser). Bajo este contexto se utiliza el flujo Implicit y se cuenta con el siguiente diagrama de flujo detallado.

En este escenario se cuenta con aplicación denominada contenido que corre en el Browser (desarrollada con el lenguaje Javascript ). En principio esta aplicación presenta el requerimiento de que solo necesita poder identificar al usuario para mostrar contenido HTML estático. Considerando estos factores, en el siguiente ejemplo se utiliza el flujo Implicit contra la solución de Access Manager.

Los datos utilizados a lo largo del documento referentes la configuración OAuth 2.0 son los siguientes: client\_id con valor con contenido, redirect\_uri con valor <https://contenido.personal.com/callback> y scope con valor con valor openid.

|  |
| --- |
| **DIAGRAMA DETALLADO APLICACION SPA (flujo Implicit)** |
| **https://docuit.personal.corp/download/attachments/14779160/image2018-2-21%2010%3A29%3A51.png?version=1&modificationDate=1519220227000&api=v2** |

**SOLICITUD DE AUTENTICACIÓN**

La aplicación realiza una llamada HTTPS GET a la URL descipta en la key authorization\_endpoint del Discovery Document. En este caso el valor es /openam/oauth2/realms/root/realms/tabpins/authorize.

|  |
| --- |
| **Se recomienda que la aplicación realice una llamada SSL o TLS a los endpoints del OpenID Provider** |

La llamada que figura en el **PASO 1** del Diagrama de flujo Aplicación SPA, está compuesta de los siguientes campos:

|  |  |  |
| --- | --- | --- |
| **Parámetro** | **Requerido** | **Descripción** |
| client\_id | SI | Identificador de la aplicación (Client) |
| response\_type | SI | Para este caso colocar los valores id\_token y token |
| scope | SI | El scope debe contener el valor openid para especificar que una llamada OpenID Connect. |
| nonce | SI | **Valor random generado por la aplicación (\*)** |
| redirect\_uri | SI | URL de la aplicación donde el Authorization Server envía los tokens solicitados |
| state | Opcional, pero es recomendado | Token único generado en la aplicación asociado a la sesión del usuario para prevenir el ataque por Cross-site request forgery (CSRF). Para mas información dirigirse al [OAuth 2.0 [RFC6749]](https://tools.ietf.org/html/rfc6749#section-10.12) |
| login\_hint | Opcional | En caso de que la aplicación conozca el identicador del usuario que se va a autenticar, puede especificarse en la llamada al Authentication Server |

|  |
| --- |
| **(\*) Para evitar ataques por replay, la aplicación debe generar el nonce para asociar la sesión del cliente con el id\_token. Este valor será devuelto como contenido del id\_token que deberá ser validado por la aplicación para determinar que corresponda el originalmente enviado** |

A continuación se muestra un ejemplo la llamada de autenticación del **PASO 2**, el código se encuentra indentado y con saltos de lineas para facilitar la lectura:

|  |
| --- |
| **GET /authorize? response\_type=id\_token%20token& client\_id=<client\_id>& redirect\_uri=<redirect\_uri>& scope=openid& state=<state\_string>& nonce=<nonce\_string> Host:<authorization\_server>** |

En nuestro ejemplo quedaría conformada de la siguiente manera:

|  |
| --- |
| **GET /openam/oauth2/realms/root/realms/tabpins/authorize? response\_type=id\_token%20token& client\_id=contenido& redirect\_uri=https%3A%2F%**[**contenido.personal.com**](http://contenido.personal.com/)**%2callback& scope=openid& state=bUUFJ0KtRjKcPD65NPOZ3Hee5YcqrGwi-1hFQ-pmWFA& nonce=vCDF4do0GsknlPzsR5ymParOMxi\_0B6e09ZufKeNLvo Host:** [**openam.personal.com**](http://openam.personal.com/) |

Luego continua el proceso de autenticación y authorización del usuario en el Authorization Server (**PASO 3 y 4**). Como se puede apreciar en este proceso, dado que la respuesta del Authorization Server es asíncrona, la aplicación debe almacenar el **nonce** y **state** para luego poder recuperarla, por ejemplo, usando una cookie (Http Only) o en local storage en HTML5.

**RESPUESTA DE LA AUTENTICACIÓN**

Si el proceso de autenticación termina forma exitosa, el Authorization Server envía una redirección (**PASO 5**) la respuesta al redirect\_uri especificado en la llamada anterior.   
En el **PASO 6** la respuesta retorna el access\_token y id\_token como URL fragment, el código se encuentra indentado y con saltos de lineas para facilitar la lectura:

|  |
| --- |
| **https://<redirect\_uri># access\_token=<access\_token> &token\_type=Bearer &id\_token=<id\_token> &expires\_in=<expires\_in> &state=<state>** |

En nuestro ejemplo quedaría conformada de la siguiente manera:

|  |
| --- |
| [**https://contenido.personal.com/callback#**](https://contenido.personal.com/callback) **&access\_token=fac28215-7195-4d24-bb9e-1cf726817db8 &token\_type=Bearer &id\_token= eyJ0eXAiOiJKV1QiLCJraWQiOiJiL082T3ZWdjEreStXZ3JINVVpOVdUaW9MdDA9IiwiYWxnIjoiUlMyNTYifQ. eyJhdF9oYXNoIjoiMlVZcFhsa0dfWkVrdVVjVXN6MEt5USIsInN1YiI6IjExNjA0MTEyMzQiLCJhdWRpdFRyYWN raW5nSWQiOiJkYzkwMDFlMS05OTE2LTQzODMtOWVjYS1jYWFiMjVhNGEzNjAtMTI1OTkiLCJpc3MiOiJodHRwOi 8vb3BlbmFtZGVzYTMucGVyc29uYWwuY29tLmFyOjgwODAvb3BlbmFtL29hdXRoMi90YWJwaW5zIiwidG9rZW5OY W1lIjoiaWRfdG9rZW4iLCJub25jZSI6ImoxM2U4diIsImF1ZCI6InN0b3JlIiwic19oYXNoIjoiRzVpWU14M3E5 MzdjOEJaSU83bU1qQSIsImF6cCI6InN0b3JlIiwiYXV0aF90aW1lIjoxNTE3NTE0MDI5LCJyZWFsbSI6Ii90YWJ waW5zIiwiZXhwIjoxNTE3NTE4MTY0LCJ0b2tlblR5cGUiOiJKV1RUb2tlbiIsImlhdCI6MTUxNzUxNDU2NH0.Kx CmCZfJBohxBljcuowpH4zA-\_tfsucmKdfr1LHaDHJrBvLqCPEtVJm8udzRdNwqu\_ BSD6yYVmUYQ5GIvV6ogbpLmvMz3b4rz7PrAky0qhpuKV6EXUk1epTjJKtDII1nke FCKGeUtJkwKhIz4sGkkuuWTk03jpEYUusPxxoMoKmTfYCyruvANFJDmkssOmeT6MUsiiLHhRSeloggRLCghrOvm MPDNarhFfRNDe5QrhXoEsMN4pa-n1iD48YYjGRP5M9DXrUKiSYuSFJMht-Rc4- yccphBdqqVRZVrRwQS9qVHpSJn0z1YYytpkEFgidq4pa4YhLmgAN67Ct7m46NQ &expires\_in=3599 &state=bUUFJ0KtRjKcPD65NPOZ3Hee5YcqrGwi-1hFQ-pmWFA** |

|  |
| --- |
| **Consideraciones: el flujo implicit NO soporta la emisión de refresh token de acuerdo a lo definido en el estándar** [**OAuth 2.0 -Implicit [RFC 6749]**](https://tools.ietf.org/html/rfc6749#section-4.2) |

Como workaround la aplicación para poder renovar los tokens, puede solicitarlos al endpoint de /authorize utilizando el parámetro prompt=none. Este mecanismo se conoce Silent Authentication Request, esto provoca que no se muestre la pantalla de login o consentimiento, por lo cual el Authorization Server retorna los tokens solicitados solo si posee una sesión valida en el Authorization Server, caso contrario retornará un error.

|  |
| --- |
| **VALIDACIÓN ID TOKEN**  En el caso del flujo Implicit se remarca la importancia de que la aplicación valide que el valor nonce del ID Token sea el mismo que fue enviado en la solicitud de autenticación para prevenir el ataque por replay. Como un paso extra la aplicación debe validar que el valor del at\_hash del ID token debe coincida con el hash asociado al access\_token (ver ImplicitIDTValidation). Finalmente la aplicación tomando la información del ID Token, por ejemplo, puede el claim sub que corresponde en este ejemplo a la linea del cliente. Si bien nuestro caso de uso la aplicación no quiere de información adicional, la aplicación puede realizar una llamada HTTPS GET al endpoint User Info especificado en la key userinfo\_endpoint del Discovery Document, en nuestro ejemplo openam/oauth2/realms/root/realms/tabpins/userinfo. En la llamada se debe especificar en el header Authorization con el string Bearer <access\_token\_value>. |