Login Único

gov.br

SSCTI - Subsecretária de Serviços ao Cidadão, Tecnologia e Inovação
GDE - Gerência de Desenvolvimento de Sistemas Estratégicos

Manual de Integração IDP.SP.GOV.BR

18/07/2023

Versão 4.0

Rua Agueda Gonçalves, 240 - Taboão da Serra - SP - CEP 06760-900 – Tel.: (11) 2845-6000 (PABX)
Correspondência: Caixa Postal 25901 - CEP 05513-970 - SÃO PAULO - SP
www.prodesp.sp.gov.br - prodesp@prodesp.sp.gov.br

Sumário

1. Manual de Integração com o IDP.SP

Introdução

Este manual destina-se a orientar desenvolvedores durante a integração de suas aplicações com o IDP.SP, o Provedor de Identidade (IDP) para o Estado de São Paulo, que atua como um intermediário com o sistema de autenticação gov.br.

O IDP.SP tem como objetivo centralizar e simplificar o processo de autenticação para as aplicações do estado, agindo como um "broker" entre estas aplicações e o gov.br. Na prática, significa que, embora a autenticação do usuário ocorra no gov.br, a aplicação em questão se comunicará exclusivamente com o IDP.SP.

Após a autenticação bem-sucedida do usuário no gov.br, o IDP.SP recebe os tokens correspondentes, enriquece esses dados, conforme necessário, e emite seus próprios tokens para a aplicação cliente.

Vale ressaltar que o usuário final não visualizará nenhuma interface do IDP.SP. Toda a interação visível ao usuário ocorrerá na interface do gov.br. Portanto, mesmo que a aplicação esteja tecnicamente integrada com o IDP.SP, para o usuário final, parece que a integração ocorre diretamente com o gov.br.

Neste manual, abordaremos os passos e considerações necessários para realizar essa integração de forma eficiente e eficaz.

Termos Importantes para a Compreensão

1.1.1. OAuth

OAuth (Open Authorization) é um protocolo de autorização padrão da indústria que permite que aplicativos de terceiros obtenham acesso limitado a um serviço HTTP. Ele fornece aos clientes um método "seguro delegado de acesso", que é a emissão de tokens de acesso para serem usados em nome do proprietário do recurso.

1.1.2. OpenID Connect (OIDC)

OpenID Connect (OIDC) é uma camada de identidade que se baseia no protocolo OAuth 2.0. OIDC permite que os clientes verifiquem a identidade do usuário com base na autenticação realizada por um servidor de autorização e obtenham informações básicas do perfil do usuário.

1.1.3. Fluxo de Authorization Code em OAuth/OIDC

O fluxo de Authorization Code é um dos métodos de autenticação mais comumente usados em OAuth e OIDC. Este fluxo é adequado para aplicativos que podem proteger segredos do cliente. Envolve redirecionar o usuário para o servidor de autorização, autenticar o usuário, redirecionar de volta com um código de autorização, e trocar esse código por um token de acesso.

1.1.4. PKCE (Proof Key for Code Exchange)

PKCE (lê-se "pixy") é uma extensão do OAuth 2.0 e OIDC, que permite que aplicativos públicos (ou seja, aplicativos que não podem manter segredos do cliente de forma segura, como aplicativos móveis ou aplicações single-page) sejam usados de forma segura.

PKCE fornece uma forma de mitigar um ataque de interceptação de código de autorização, gerando um código de desafio que está associado com a solicitação de autorização, e que deve ser verificado quando o código de autorização é trocado por um token de acesso.

1.1.5. Discovery Endpoint

O endpoint de descoberta é um recurso que permite que as configurações e características de um servidor de autorização OIDC/OAuth 2.0 sejam descobertas dinamicamente.

A URL deste endpoint segue o formato https://{domain}/.well-known/openid-configuration, e ao acessar essa URL, um cliente pode obter todas as informações necessárias para interagir com o servidor de autorização, incluindo URLs para autorização, token, userinfo e outros endpoints, algoritmos de criptografia suportados, e outros detalhes.

Este recurso é particularmente útil para permitir que clientes se adaptem a alterações nas configurações do servidor de autorização sem necessidade de atualizações de código.

Papeis no OAuth 2.0

No contexto do IDP.SP e suas integrações, os papéis do OAuth 2.0 podem ser interpretados da seguinte maneira:

Authorization Server (Servidor de Autorização): Neste cenário, o IDP.SP atua como o servidor de autorização. Ele é responsável por autenticar o usuário (proprietário do recurso) e emitir tokens de acesso para os clientes que solicitam acesso.

Resource Owner (Proprietário do Recurso): São os usuários que possuem contas no IDP.SP e utilizam as aplicações clientes. Estes usuários concedem permissões aos clientes para acessar seus dados (recursos) nas APIs protegidas.

Client (Cliente): São as aplicações que integraram com o IDP.SP e utilizam este para autenticar seus usuários. Estas aplicações solicitam acesso aos recursos do usuário e, após a autenticação e autorização bem-sucedida, recebem um token de acesso do IDP.SP.

Resource Server (Servidor de Recursos): Nesse caso, os servidores de recursos seriam tanto o Gov.br (para onde o IDP.SP faz brokering e obtém informações adicionais do usuário), quanto as APIs seguras que integram com o IDP.SP e utilizam os tokens emitidos por este para

autorizar acessos. Essas APIs validam os tokens de acesso fornecidos pelos clientes (aplicações integradas com o IDP.SP) para garantir que eles possuam as permissões necessárias para acessar os recursos solicitados.

Assim, o IDP.SP se encontra em uma posição central nesta arquitetura, intermediando a comunicação entre as aplicações clientes e os servidores de recursos, garantindo a autenticação e autorização de maneira segura e eficiente.

Papeis no OIDC

Os papéis do OpenID Connect (OIDC) no contexto do IDP.SP e suas integrações podem ser interpretados da seguinte maneira:

OpenID Provider (Fornecedor OpenID): Neste caso, o IDP.SP atua como o Fornecedor OpenID. Ele é o servidor de autorização que emite o ID token. O ID token contém informações sobre o usuário final na forma de claims.

End User (Usuário Final): Os usuários finais são as pessoas que possuem contas no IDP.SP. As informações desses usuários são contidas nos ID tokens emitidos pelo IDP.SP.

Relying Party (Parte Confiante): São as aplicações que integraram com o IDP.SP para autenticar seus usuários. Essas aplicações solicitam ID tokens ao IDP.SP para identificar os usuários finais. No caso do IDP.SP, esses clientes podem ser as aplicações do governo do estado de São Paulo, ou aplicações de terceiros que confiam no IDP.SP para autenticação.

ID Token: Este é o token emitido pelo IDP.SP (o Fornecedor OpenID) que contém informações sobre o usuário final na forma de claims. Estes tokens são usados pelas aplicações clientes para identificar os usuários após uma autenticação bem-sucedida.

Claim: O claim é uma peça de informação sobre o usuário final que está contida no ID token. Estes podem incluir detalhes como o nome do usuário, endereço de email, entre outros, e são definidos de acordo com as necessidades de cada aplicação cliente.

A grande diferença entre OIDC e OAuth 2.0 é que OIDC resulta em um ID token, além de qualquer token de acesso ou refresh. Isso adiciona um nível extra de informação sobre o usuário que pode ser usado pelas aplicações para personalizar a experiência do usuário. Assim, o IDP.SP, atuando como um Fornecedor OpenID, desempenha um papel crucial na autenticação dos usuários e na entrega de suas informações para as aplicações clientes de maneira segura.

Tipos de Clientes vs OAuth Flows

Qual tipo de cliente você está construindo? O tipo de fluxo OAuth 2.0 depende do tipo de cliente que você está construindo.

O seu cliente é público? Uma aplicação cliente é considerada pública quando um usuário final poderia possivelmente visualizar e modificar o código. Isso inclui Single-Page Apps (SPAs) ou quaisquer aplicações móveis ou nativas. Em ambos os casos, a aplicação não pode manter segredos dos usuários maliciosos. Seu cliente é considerado confidencial ou privado para aplicações server-side (web), o que significa que seu cliente pode usar autenticação do cliente como um segredo do cliente.

O seu cliente é um SPA ou nativo? Se a sua aplicação cliente é um SPA ou uma aplicação nativa, você deve usar um fluxo de autorização com PKCE, como o fluxo de código de autorização com PKCE. No caso do IDP.SP, por questões de segurança, o único fluxo permitido para clientes públicos é o fluxo de código de autorização.

O cliente possui um usuário final? Se a sua aplicação cliente está rodando em um servidor sem um usuário final direto, então ela pode ser confiada para lidar com credenciais e usá-las de maneira responsável. Se a sua aplicação cliente estiver apenas fazendo interação máquina a máquina, então você deve usar o fluxo de credenciais do cliente.

Ambientes do IDP.SP

1.1.6. Ambiente de Homologação

O ambiente de homologação do IDP.SP é uma instância do servidor de autorização onde os clientes podem testar a integração antes de implantar em produção. Este ambiente é usado para validação de funcionalidades e de segurança.

1.1.7. Discovery Endpoint

O Discovery Endpoint fornece detalhes de configuração do ambiente de homologação do IDP.SP. Aqui estão algumas das informações mais relevantes:

- **Issuer**: https://rhsso.idp-hml.sp.gov.br/auth/realms/idpsp
- **Authorization Endpoint**: https://rhsso.idp-hml.sp.gov.br/auth/realms/idpsp/protocol/openid-connect/auth
- **Token Endpoint**: https://rhsso.idp-hml.sp.gov.br/auth/realms/idpsp/protocol/openid-connect/token
- **Userinfo Endpoint**: https://rhsso.idp-hml.sp.gov.br/auth/realms/idpsp/protocol/openid-connect/userinfo
- **End Session Endpoint**: https://rhsso.idp-hml.sp.gov.br/auth/realms/idpsp/protocol/openid-connect/logout
- **JWKS URI**: https://rhsso.idp-hml.sp.gov.br/auth/realms/idpsp/protocol/openid-connect/certs
- **Registration Endpoint**: https://rhsso.idp-hml.sp.gov.br/auth/realms/idpsp/clients-registrations/openid-connect

1.1.8. Ambiente de Produção

O ambiente de produção do IDP.SP é a instância do servidor de autorização onde os clientes implantam e executam suas aplicações finais. Este ambiente é projetado para o desempenho e a segurança necessários para o funcionamento de aplicações em tempo real.

1.1.9. Discovery Endpoint

O Discovery Endpoint fornece detalhes de configuração do ambiente de produção do IDP.SP. Aqui estão algumas das informações mais relevantes:

- **Issuer**: https://idp.sp.gov.br/auth/realms/idpsp
- **Authorization Endpoint**: https://idp.sp.gov.br/auth/realms/idpsp/protocol/openid-connect/auth
- **Token Endpoint**: https://idp.sp.gov.br/auth/realms/idpsp/protocol/openid-connect/token
- **Userinfo Endpoint**: https://idp.sp.gov.br/auth/realms/idpsp/protocol/openid-connect/userinfo
- **End Session Endpoint**: https://idp.sp.gov.br/auth/realms/idpsp/protocol/openid-connect/logout
- **JWKS URI**: https://idp.sp.gov.br/auth/realms/idpsp/protocol/openid-connect/certs
- **Registration Endpoint**: https://idp.sp.gov.br/auth/realms/idpsp/clients-registrations/openid-connect

Além disso, uma série de funcionalidades suportadas e parâmetros também são disponibilizados através do Discovery Endpoint. Essas informações incluem os métodos de autenticação suportados, algoritmos de assinatura e criptografia, modos de resposta suportados, tipos de reivindicações suportadas, métodos de desafio de código suportados (para PKCE), e muito mais.

Certifique-se de consultar o Discovery Endpoint do ambiente de produção para obter todas as informações necessárias para configurar corretamente a sua integração.

Authorization Code Flow

O fluxo de Authorization Code é um dos fluxos de concessão mais comumente usados no OAuth 2.0 e OpenID Connect. Este fluxo é adequado para aplicações que têm a capacidade de armazenar e proteger um segredo de cliente, tais como aplicações de servidor para servidor.

No fluxo de Authorization Code, o cliente redireciona o usuário para o servidor de autorização (neste caso, o IDP.SP) para que o usuário possa autenticar. Depois que o usuário autentica, o servidor de autorização redireciona o usuário de volta para o cliente, incluindo um código de autorização na URL de redirecionamento. O cliente então envia este código de autorização de volta para o servidor de autorização, juntamente com seu ID de cliente e segredo de cliente, para obter um token de acesso.

Este fluxo possui uma forte segurança, pois o token de acesso é emitido diretamente para o cliente pelo servidor de autorização e nunca é exposto ao usuário ou ao agente do usuário.

Proof Key for Code Exchange (PKCE)

A Proof Key for Code Exchange (PKCE, pronunciado "pixy") é uma extensão do fluxo de Authorization Code que protege contra ataques de interceptação de código de autorização. É particularmente útil para aplicações públicas, como aplicações móveis e de página única (SPA), onde o segredo do cliente não pode ser mantido em segurança.

No fluxo PKCE, o cliente gera um secret chamado "code verifier" para cada sessão de autorização. Uma transformação criptográfica deste "code verifier" é então enviada ao servidor

de autorização quando o cliente inicia a sessão de autorização. Quando o cliente troca o código de autorização por um token de acesso, ele também envia o "code verifier". O servidor de autorização então verifica se a transformação do "code verifier" enviado na troca de token corresponde à transformação originalmente enviada.

PKCE protege contra ataques de interceptação de código, garantindo que mesmo se um invasor interceptar o código de autorização, ele não será capaz de trocar o código por um token de acesso sem o "code verifier".

Para o IDP.SP, é crucial usar o PKCE em todas as solicitações de código de autorização para melhorar a segurança de sua integração.

Integração das aplicações com o IDP.SP

Para uma integração eficaz com o IDP.SP, é essencial seguir as melhores práticas recomendadas, que envolvem o uso de bibliotecas bem estabelecidas para lidar com a complexidade do protocolo OAuth 2.0 e OpenID Connect. **A implementação manual desses protocolos não é recomendada devido à complexidade inerente e à possibilidade de erros que podem resultar em vulnerabilidades de segurança.**

1.1.10. Uso de Bibliotecas de Linguagem

As bibliotecas de linguagem fornecem uma forma abstrata e segura de implementar a lógica necessária para interagir com um servidor de autorização OAuth 2.0/OpenID Connect, como o IDP.SP. Estas bibliotecas lidam com detalhes de baixo nível, como a troca de tokens e a renovação de tokens expirados, permitindo que você se concentre na lógica do seu aplicativo.

Vale ressaltar que a maioria das linguagens de programação modernas possui bibliotecas de suporte para OAuth 2.0/OpenID Connect que são mantidas pela própria comunidade da linguagem ou por terceiros. Recomenda-se o uso dessas bibliotecas sempre que possível.

Por exemplo, para aplicações **Node.js**, você pode usar a biblioteca openid-client, e para aplicações **Python**, pode usar oauthlib ou Authlib.

Para aplicações **Java**, Spring Security tem um bom suporte para OAuth 2.0/OpenID Connect, enquanto que para **.NET**, a Microsoft fornece a biblioteca Microsoft Identity.

1.1.11. Uso de Bibliotecas de Terceiros Confiáveis

Se a sua linguagem de programação não possuir uma biblioteca de linguagem para OAuth 2.0/OpenID Connect ou se você estiver desenvolvendo uma aplicação que não se encaixa nos casos de uso comuns, você pode optar por usar uma biblioteca de terceiros de confiança.

Existem muitas bibliotecas OAuth 2.0/OpenID Connect disponíveis que são mantidas por organizações confiáveis e têm uma grande comunidade de usuários. Por exemplo, AppAuth é uma biblioteca que suporta tanto Android quanto iOS e que é mantida pelo OpenID Foundation.

Ao escolher uma biblioteca de terceiros, certifique-se de que a biblioteca é ativamente mantida, suporta o padrão OAuth 2.0/OpenID Connect completamente, e tem uma comunidade de usuários ativa.

Lembrando, é de extrema importância que você não tente implementar a lógica OAuth 2.0/OpenID Connect manualmente. Usar uma biblioteca confiável ajuda a evitar erros comuns e possíveis vulnerabilidades de segurança. Ao invés disso, foque em configurar a biblioteca corretamente, usando os endpoints e detalhes do ambiente de produção do IDP.SP, como detalhado na seção Discovery Endpoint deste manual.

Integração Manual ao IDP.SP

Para algumas situações, especialmente em cenários onde a linguagem de programação utilizada não possui suporte direto ou bibliotecas disponíveis para facilitar a integração com o protocolo OpenID Connect (OIDC), pode ser necessário implementar o fluxo do protocolo manualmente. Esta seção irá orientá-lo em como realizar a integração manual ao IDP.SP, implementando o fluxo de Authorization Code com PKCE.

Overview

O fluxo de Authorization Code com PKCE (Proof Key for Code Exchange) é um dos fluxos definidos no protocolo OIDC e é adequado para todos os tipos de aplicações, independentemente de poderem garantir a confidencialidade do client secret.

Este fluxo consiste em vários passos:

O cliente gera um código de desafio e um verificador.

O cliente direciona o navegador do usuário para o Authorization Server com o código de desafio.

O Authorization Server autentica o usuário (se necessário) e pede ao usuário para autorizar o pedido do cliente.

O Authorization Server emite o código de autorização para o redirect URI do cliente.

O cliente troca o código de autorização pelo token de acesso, usando o verificador de código original.

O Authorization Server verifica o verificador de código e emite o token de acesso.

O cliente usa o token de acesso para solicitar recursos ao Resource Server.

Quando necessário, o cliente pode fazer o logout do usuário no Authorization Server.

Implementação Manual Passo-a-Passo

Passo 1: Gerar código de desafio e verificador

O código de desafio e o verificador são usados para garantir que a troca do código de autorização pelo token de acesso seja feita pelo mesmo cliente que iniciou o fluxo. Esta é uma medida de segurança para evitar ataques de interceptação.

Gere um verificador de código. Este deve ser um valor de alta entropia e, idealmente, criptograficamente aleatório. Deve ter entre 43 e 128 caracteres.

Gere o código de desafio. Este é uma transformação do verificador de código. Faça um hash SHA-256 do verificador de código, base64url-encode o resultado, e este é o código de desafio.

Passo 2: Solicitar código de autorização

Nesta etapa, o cliente direciona o usuário para o Authorization Server para autenticar e autorizar o cliente. O cliente deve incluir vários parâmetros na URL, incluindo o código de desafio gerado na etapa anterior.

A URL deve parecer com isto:

https://rhsso.idp-hml.sp.gov.br/auth/realms/idpsp/protocol/openid-connect/auth?client_id={client_id}&response_type=code&scope={scope}&redirect_uri={redirect_uri}&state={state}&code_challenge={code_challenge}&code_challenge_method=S256

O Authorization Server irá autenticar o usuário, pedir ao usuário para autorizar o cliente, e então redirecionar o usuário de volta ao redirect_uri fornecido com um código de autorização.

Resposta do Authorization Endpoint

O Authorization Endpoint é onde a autenticação do usuário é feita e o cliente é autorizado. Aqui está um exemplo de uma resposta de sucesso do Authorization Endpoint:

HTTP/1.1 302 Found

Location: https://client.example.org/callback?code=SplxlOBeZQQYbYS6WxSbIA&state=xyz

Neste exemplo, o usuário foi redirecionado para o URI de redireção do cliente com o código de autorização e o estado fornecido no pedido original.

Se ocorrer um erro, o Authorization Endpoint redirecionará o usuário para o URI de redireção do cliente com os parâmetros de erro. Por exemplo:

HTTP/1.1 302 Found

Location: *https://client.example.org/callback?error=access_denied&state=xyz*

Passo 3: Trocar o código de autorização pelo token de acesso

Após receber o código de autorização, o cliente deve fazer uma requisição POST ao endpoint de token do Authorization Server para trocar o código de autorização pelo token de acesso. A requisição deve incluir o verificador de código original.

A requisição POST deve incluir os seguintes parâmetros no corpo da mensagem com a codificação application/x-www-form-urlencoded:

grant_type: Este deve ser authorization_code, indicando que você está usando o fluxo de Authorization Code.

code: Este é o código de autorização que você recebeu do Authorization Server.

redirect_uri: Esta é a URI de redireção que você usou na sua requisição de código de autorização.

client_id: O identificador do cliente registrado no servidor de autorização.

code_verifier: Este é o verificador de código original que você gerou na primeira etapa.

Aqui está um exemplo do corpo da requisição:

```
POST /auth/realms/idpsp/protocol/openid-connect/token HTTP/1.1

Host: rhsso.idp-hml.sp.gov.br

Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&code=SplxlOBeZQQYbYS6WxSbIA&redirect_uri=https%3A%
2F%2Fclient.example.org%2Fcallback&client_id=your_client_id&code_verifier=your_g
enerated_code_verifier
```

Lembre-se de substituir **your_client_id** e **your_generated_code_verifie**r com seu **client_id** e **code_verifier** reais. Da mesma forma, certifique-se de que o **redirect_uri** corresponda ao URI de redireção usado anteriormente. Além disso, **code** deve ser o código de autorização recebido na resposta do Authorization Endpoint.

Passo 4: Logout do usuário

Quando o usuário desejar sair da aplicação, o cliente pode direcionar o usuário para o Endsession Endpoint para fazer logout no servidor de autorização. Ao direcionar o usuário para o logout, o cliente deve incluir o id_token obtido durante a autenticação como um parâmetro id_token_hint na URL de logout. Isso servirá como uma sugestão para o Authorization Server sobre a sessão que está sendo encerrada e evitará a necessidade de confirmação pelo usuário.

Aqui está o URL do Endsession Endpoint:

https://rhsso.idp-hml.sp.gov.br/auth/realms/idpsp/protocol/openid-connect/logout?id_token_hint={id_token}&post_logout_redirect_uri={post_logout_redirect_uri}

Este endpoint irá encerrar a sessão do usuário no servidor de autorização e, opcionalmente, pode redirecionar o usuário de volta para a aplicação cliente.

É importante notar que, ao implementar manualmente esse fluxo, você deve estar ciente das medidas de segurança que precisam ser consideradas, como o armazenamento seguro do client secret e a validação dos tokens recebidos do Authorization Server. Além disso, a implementação manual pode ser mais propensa a erros e mais difícil de manter do que o uso de uma biblioteca pronta para uso.

ID Token

O ID Token é uma representação segura de que a autenticação do usuário foi bem-sucedida. Ele contém várias informações sobre o usuário e a sessão. Aqui está um exemplo de um ID Token retornado após a autenticação bem-sucedida:


```
{
  "exp": 1689355759,
  "iat": 1689352159,
  "auth_time": 1689352158,
  "jti": "2126c96a-bbe1-4d60-a57a-1d5498616e5c",
  "iss": "https://rhsso.idp-hml.sp.gov.br/auth/realms/idpsp",
  "aud": "idpsp-web",
  "sub": "31e93ff7-c1f4-49c4-b3a1-5f0e8b8a4c9a",
  "typ": "ID",
  "azp": "idpsp-web",
  "nonce":
"638249489566226627.ZmE3NzExYWEtYzljNS00N2JlLTk0ZTItN2MwZGZkMWQ4NmYzMGJlZm
FkMDktYjNjZS00Nzg0LThhNTktMDkwZTNiODVjMjk1",
  "session_state": "b4c848ba-1e7f-4dc6-a076-d57288632342",
  "at_hash": "ShbE5xS07hWzVbsStxzdIA",
  "acr": "1",
  "sid": "b4c848ba-1e7f-4dc6-a076-d57288632342",
  "email_verified": true,
  "id_cidadao": "fbc3a74c-3b3b-4f1d-a148-493192c586d6",
  "amr": [
    "x509",
    "x509_token"
  ],
  "name": "JOHN DOE",
  "phone_number_verified": true,
  "phone_number": "12345678901",
  "preferred_username": " 52078063002",
  "given_name": "JOHN",
  "family_name": "DOE",
  "email": "johndoe@example.com"
}
```

Aqui estão os detalhes de cada campo:

exp: A data/hora de expiração do token, após a qual o ID Token não será mais aceito para autenticação.
iat: A data/hora em que o ID Token foi emitido.
auth_time: A data/hora em que o usuário foi autenticado.
jti: Um identificador único para o token, que pode ser usado para evitar a reutilização do token.
iss: O issuer do token. Este valor deve corresponder ao URL da instância do IDP.SP.
aud: A audiência do token. Este valor deve corresponder ao ID do cliente do IDP.SP.
sub: Um identificador único para o usuário que foi autenticado.
typ: O tipo do token, que será "ID".
azp: O ID do cliente autorizado a emitir o ID Token.
nonce: String aleatória que será usada para associar o cliente à solicitação de ID Token.
session_state: O estado da sessão do usuário.
at_hash: Hash do access token. Ele fornece validação para o Access Token correspondente.
acr: O nível de autenticação realizado durante a autenticação do usuário.
sid: O identificador da sessão do usuário.
email_verified: Indica se o e-mail do usuário foi verificado como verdadeiro.
id_cidadao: O ID único do cidadão no sistema.
amr: Array de strings que representam os métodos de autenticação realizados durante a autenticação.
name: O nome completo do usuário.
phone_number_verified: Indica se o número de telefone do usuário foi verificado como verdadeiro.
phone_number: O número de telefone do usuário.
preferred_username: O CPF do usuário.
given_name: O primeiro nome do usuário.
family_name: O sobrenome do usuário.
email: O e-mail do usuário.

Por favor, lembre-se de que todos os dados pessoais do usuário são retornados de maneira segura e são estritamente confidenciais. Trate-os com a mais alta consideração pela privacidade e segurança.

Access Token

O Access Token é uma representação segura das permissões concedidas ao usuário. Ele contém várias informações sobre o usuário e as permissões do usuário. Aqui está um exemplo de um Access Token retornado após a autenticação bem-sucedida:


```
{
  "exp": 1689355759,
  "iat": 1689352159,
  "auth_time": 1689352158,
  "jti": "1dd99d43-37d0-41ae-91e6-7e6db0f3444e",
  "iss": "https://rhsso.idp-hml.sp.gov.br/auth/realms/idpsp",
  "aud": [
    "broker",
    "account"
  ],
  "sub": "31e93ff7-c1f4-49c4-b3a1-5f0e8b8a4c9a",
  "typ": "Bearer",
  "azp": "idpsp-web",
  "nonce":
"638249489566226627.ZmE3NzExYWEtYzljNS00N2JlLTk0ZTItN2MwZGZkMWQ4NmYzMGJlZmFkMDktYj
NjZS00Nzg0LThhNTktMDkwZTNiODVjMjk1",
  "session_state": "b4c848ba-1e7f-4dc6-a076-d57288632342",
  "acr": "1",
  "allowed-origins": [
    ""
  ],
  "realm_access": {
    "roles": [
      "default-roles-idpsp",
      "offline_access",
      "uma_authorization"
    ]
  },
  "resource_access": {
    "broker": {
      "roles": [
        "read-token"
      ]
    },
    "account": {
      "roles": [
        "manage-account",
        "manage-account-links",
        "view-profile"
      ]
    }
  },
  "scope": "openid phone email profile",
  "sid": "b4c848ba-1e7f-4dc6-a076-d57288632342",
  "email_verified": true,
  "name": "JOHN DOE",
  "phone_number_verified": true,
  "phone_number": "1234567890",
  "preferred_username": " 52078063002",
  "given_name": "JOHN",
  "family_name": "DOE",
  "email": "johndoe@example.com"
}
```

Aqui estão os detalhes de cada campo:

exp: A data/hora de expiração do token, após a qual o Access Token não será mais aceito para autenticação.
iat: A data/hora em que o Access Token foi emitido.
auth_time: A data/hora em que o usuário foi autenticado.
jti: Um identificador único para o token, que pode ser usado para evitar a reutilização do token.
iss: O issuer do token. Este valor deve corresponder ao URL da instância do IDP.SP.
aud: A audiência do token. Este valor deve corresponder ao ID do cliente do IDP.SP.
sub: Um identificador único para o usuário que foi autenticado.
typ: O tipo do token, que será "Bearer".
azp: O ID do cliente autorizado a emitir o Access Token.
nonce: String aleatória que será usada para associar o cliente à solicitação de Access Token.
session_state: O estado da sessão do usuário.
acr: O nível de autenticação realizado durante a autenticação do usuário.
allowed-origins: A lista de origens permitidas para o Access Token.
realm_access: Contém as permissões do usuário dentro do reino (ou seja, domínio de segurança).
resource_access: Contém as permissões do usuário para diferentes recursos.
scope: Os escopos para os quais o Access Token é válido.
sid: O identificador da sessão do usuário.
email_verified: Indica se o e-mail do usuário foi verificado como verdadeiro.
name: O nome completo do usuário.
phone_number_verified: Indica se o número de telefone do usuário foi verificado como verdadeiro.
phone_number: O número de telefone do usuário.
preferred_username: O CPF do usuário.
given_name: O primeiro nome do usuário.
family_name: O sobrenome do usuário.
email: O e-mail do usuário.

Assim como o ID Token, todas as informações pessoais do usuário retornadas são confidenciais e devem ser tratadas com a maior consideração pela privacidade e segurança.

Refresh Token

O Refresh Token é um tipo especial de token que pode ser usado para obter um novo Access Token após o original ter expirado. Ele permite que a aplicação cliente obtenha um novo Access Token sem a necessidade de o usuário final se autenticar novamente. Isso é particularmente útil para aplicações cliente que necessitam de acesso contínuo a um recurso protegido, mas onde se deseja minimizar a frequência de solicitações de autenticação do usuário final.

Refresh Tokens são normalmente de longa duração e podem ser usados para solicitar novos Access Tokens por um período de tempo definido pela política de tokens do idp.sp. Vale ressaltar que o uso de Refresh Tokens é um processo que deve ser protegido e seguro, pois qualquer vazamento desses tokens pode permitir que uma entidade mal-intencionada obtenha acesso contínuo ao recurso protegido.

Após o processo de autenticação bem-sucedida e a obtenção de um Access Token, um Refresh Token também pode ser retornado, dependendo das políticas definidas e do tipo de fluxo OAuth utilizado. O Refresh Token pode então ser usado, juntamente com o Client ID e o Client Secret, para solicitar um novo Access Token do IDP.SP

Scopes

Os Scopes em OpenID Connect são utilizados para requisitar informações específicas de um usuário. A aplicação cliente informa os scopes necessários no momento da autenticação. Aqui estão os scopes que podem ser pedidos ao usar o IDP.SP:

openid: Este é o scope obrigatório para todas as aplicações que utilizam OpenID Connect. Ele é necessário para a obtenção de um ID Token e informa ao IDP.SP que a aplicação está utilizando o protocolo OpenID Connect.

offline_access: Este scope pode ser usado se a aplicação precisar acessar recursos do usuário enquanto o usuário não estiver presente. Quando este scope é concedido, o IDP.SP retorna um Refresh Token que pode ser usado para obter novos Access Tokens.

email: Ao utilizar este scope, a aplicação está requisitando acesso ao endereço de e-mail do usuário. O endereço de e-mail será retornado no ID Token e no UserInfo Endpoint, se o usuário consentir.

profile: Este scope é usado para requisitar informações do perfil do usuário. Isto inclui, mas não está limitado a, o nome, sobrenome, nome de usuário preferido e imagem de perfil.

ip_address: Este scope permite que a aplicação obtenha o endereço IP do usuário. Este é um recurso específico do IDP.SP e pode ser útil para aplicações que precisam rastrear o endereço IP do usuário para fins de segurança ou de auditoria.

phone: Este scope pode ser usado para obter o número de telefone do usuário.

govbr_confiabilidades: Esse scope permite acessar as informações de confiabilidade do usuário no Gov.br. Isso pode incluir informações como a confirmação de que o usuário passou por um processo de verificação de identidade (selo Bronze, Prata ou Ouro)

govbr_empresa: Este scope permite que a aplicação obtenha informações sobre quaisquer empresas que o usuário possa possuir ou estar afiliado.

Ao utilizar esses scopes, é importante ter em mente que o consentimento do usuário é necessário para acessar qualquer informação pessoal. Isso significa que o usuário deve ser informado de que informações você está requisitando e por quê, e eles devem consentir ativamente com essa coleta de informações. Além disso, essas informações devem ser armazenadas de forma segura e usadas apenas para os fins para os quais foram coletadas.

Utilização do IDP.SP para Segurança de APIs

A segurança de APIs pode ser implementada com sucesso utilizando o IDP.SP. Os fluxos OAuth 2.0 que podem ser utilizados para esse propósito são principalmente o fluxo de Client Credentials e o fluxo de Authorization Code.

1.1.12. Fluxo de Client Credentials

No fluxo de Client Credentials, um cliente faz uma requisição de token diretamente ao IDP.SP utilizando suas credenciais de cliente (client_id e client_secret). O IDP.SP autentica o cliente e, se bem-sucedido, emite um token de acesso. Este fluxo é frequentemente usado quando a aplicação cliente está agindo em seu próprio nome, ou seja, a aplicação está acessando um recurso que é de sua propriedade, e não um recurso de propriedade do usuário.

1.1.13. Fluxo de Authorization Code

No fluxo de Authorization Code, o cliente direciona o usuário para o IDP.SP, onde ele faz a autenticação. Uma vez autenticado, o usuário é redirecionado de volta para o cliente com um código de autorização. O cliente então troca este código por um access token.

Este fluxo é comumente usado quando a aplicação cliente está agindo em nome do usuário. Uma vez obtido o access token, ele pode ser usado para acessar recursos pertencentes ao usuário na API.

Em ambos os casos, o cliente pode usar o access token obtido para fazer requisições à API. O escopo de acesso, que determina quais recursos e operações são permitidos para o token emitido, pode ser definido no momento da requisição do token e normalmente é definido pelos requisitos da API que o token está autorizado a acessar.

1.1.14. Escopos de API (API Scopes)

Os escopos de API (API Scopes) são usados para controlar o acesso à API. Eles são definidos pela própria API e indicam o que o portador do token pode fazer. Cada escopo geralmente corresponde a um conjunto específico de permissões ou ações na API.

Quando um cliente é registrado no IDP.SP, ele precisa ser configurado com os escopos que está autorizado a solicitar. Durante o processo de autenticação, o cliente deve solicitar explicitamente esses escopos, que então são incluídos no access token emitido se o cliente estiver autorizado a solicitá-los.

As APIs protegidas pelo IDP.SP devem estar configuradas para validar o token de acesso em cada requisição, o que inclui a verificação da assinatura do token, da expiração do token, e dos escopos de acesso do token.

Portanto, tanto o fluxo de Client Credentials quanto o fluxo de Authorization Code podem ser usados para autenticar um cliente e obter um access token que pode ser usado para acessar recursos protegidos na API.

Acesso aos Serviços do Gov.br

Os serviços de "Confiabilidades" e "Empresas" são oferecidos pelo Gov.br e, portanto, requerem um Access Token do Gov.br para acesso. No entanto, as aplicações integradas com o IDP.SP recebem tokens do próprio IDP.SP.

Para proporcionar o uso destes importantes serviços, eles foram encapsulados em um micro serviço no ecossistema do IDP.SP. Este micro serviço recebe o token do IDP.SP, realiza a troca pelo token do Gov.br, faz a requisição ao serviço correspondente do Gov.br e retorna o resultado para a aplicação do solicitante.

1.1.15. URLs Base do Serviço

Dependendo do ambiente em que a aplicação está operando, as URLs base para acessar o micro serviço são as seguintes:

- Ambiente de Homologação: https://govbr-service.idp-hml.sp.gov.br/
- Ambiente de Produção: https://govbr-service.idp-prd.sp.gov.br/

1.1.16. Serviços Disponíveis

Abaixo, apresentamos as rotas disponíveis neste micro serviço:

GET /userinfo/x509: Retorna os dados do certificado X509 do usuário.

GET /userinfo/foto: Retorna a foto do usuário. O micro serviço buscará a foto do usuário no serviço do Gov.br.

GET /userinfo/confiabilidades: Retorna as confiabilidades do usuário. O micro serviço buscará essas informações no serviço do Gov.br.

GET /userinfo/confiabilidades/detalhe: Retorna os detalhes das confiabilidades do usuário. O micro serviço buscará essas informações no serviço do Gov.br.

GET /userinfo/empresas: Retorna as empresas associadas ao CPF do usuário. O micro serviço buscará essas informações no serviço do Gov.br.

GET /userinfo/empresas/{cnpj}/participantes: Retorna os participantes associados a uma empresa específica. O micro serviço buscará essas informações no serviço do Gov.br.

1.1.17. Requisitos de Acesso

Para acessar qualquer uma das rotas acima, é necessário fornecer um Access Token válido do IDP.SP no cabeçalho Authorization da requisição. Além disso, esse token deve incluir os scopes apropriados para o serviço que está sendo requisitado.

Esse procedimento garante que apenas aplicações autorizadas e devidamente autenticadas possam acessar as informações e serviços disponíveis.

Ferramentas úteis para a integração

Jwt.io
Oidc playground
Postman
Especificação OAuth 2.0:
https://datatracker.ietf.org/doc/html/rfc6749
Especificação OpenID Connect Core 1.0
https://openid.net/specs/openid-connect-core-1_0.html
Especificação PKCE

- https://datatracker.ietf.org/doc/html/rfc7636