Guia de Integração de Serviço Público Digital

Plataforma Brasil Cidadão

Roteiro de Integração (SDK-v1.2)

Sumário

Contexto

A Presidência da República, por meio da Casa Civil, de políticas de cidadania digital instituída pelo Decreto n° 8.936, de 19 de dezembro de 2016, de compartilhamento de base de dados constante no Decreto n° 8.789, de 29 de junho de 2016, e por meio do Decreto n28.638 de 15 de janeiro de 2016, que institui a Política de Governança Digital, bem como pelas políticas de Governo Digital com enfoque no cidadão, iniciou o projeto da plataforma de cidadania digital, que contempla diversas diretrizes para a prestação de serviços públicos digitais, das quais fazem parte a convergência autoritativa e a federação dos processos de autenticação dos serviços digitais. Para essa diretriz foi concebido o conceito da Plataforma de Autenticação Digital do Cidadão, o projeto Brasil Cidadão.

O Portal de Serviços do Governo federal *[servicos.gov.br]* se tornará o canal único e integrado para a disponibilização de informações, solicitação eletrônica e acompanhamento de serviços públicos. Além de praticidade e agilidade para cidadãos e empresários, os serviços digitais reduzirão em até 97% o custo para o governo e eliminarão os deslocamentos desnecessários, o tempo de espera nas filas, a impressão de certidões e a autenticação de documentos, algumas das dificuldades enfrentadas atualmente no atendimento presencial. As ações da plataforma também estão alinhadas com a Estratégia de Governança Digital (EGD), que orientará as ações de Tecnologia da Informação até 2019.

Dentro deste contexto, podemos destacar as diversas dificuldades com múltiplas contas de acesso sob responsabilidade do cidadão e variados bancos de dados cadastrais, tais como a duplicidade e inconsistência de informações, falta de integração, dados dispersos e diversas formas de autenticação. Esses são alguns dos problemas enfrentados por cidadãos ao tentar consumir um serviço público digital oferecido pelo governo federal. Para solucionar essas dificuldades, o Ministério do Planejamento, Desenvolvimento e Gestão (MP), em parceria com o Serviço Federal de Processamento de Dados (Serpro), disponibilizou a plataforma central de autenticação digital do cidadão, o Brasil Cidadão.

A integração ao Brasil Cidadão é iniciada com um pedido de adesão do órgão. Após a solicitação, o serviço público digital deverá ser cadastrado no Portal de Serviços do Governo Federal *[servicos.gov.br]*. Esse cadastro é necessário para viabilizar todo o processo de integração, pois somente desta maneira serão geradas e enviadas as chaves de autenticação do serviço. Somente os serviços públicos digitais oferecidos ao cidadão poderão ser integrados à plataforma de autenticação.

Essa é a nova proposta do Governo federal, para facilitar a identificação e autenticação do cidadão, privilegiando a governança e a convergência autoritativa, e finalmente o controle de acesso unificado. A Plataforma de Cidadania Digital chega para ampliar e simplificar o acesso dos cidadãos brasileiros aos serviços públicos digitais, inclusive por meio de dispositivos móveis.

Introdução

Este SDK é a documentação necessária para efetivar integração de qualquer serviço público digital à Plataforma de Autenticação Digital do Cidadão – Brasil Cidadão. A partir de agora, será feita uma revisão sobre a arquitetura de serviço e alguns conceitos utilizados pela Plataforma, além de uma explicação sobre procedimentos administrativos essenciais para autorizar o acesso do serviço à Plataforma.

Este documento ajudará os desenvolvedores a executar os passos necessários para a integração, ao indicar a eles as formas de chamadas a operações, parâmetros e métodos de integração, e, por último, os procedimentos para permitir a conectividade ente os ambientes de implantação.

Arquitetura de Serviço e Protocolos

OpenID Connect

O Openid Connect é um protocolo baseado no OAuth 2.0 que especifica autorização e autenticação. O objetivo principal é conectar com segurança os usuários de aplicações Web. Ele define como implementar o gerenciamento de autorizações de acesso, gerenciamento de sessão, fornecimento de informações sobre usuário logado, mecanismo de logout, etc. O OIDC permite executar o logon único dos usuários e apresenta o conceito de um id_token, que é um token de segurança que permite ao cliente verificar a identidade do usuário e obter informações básicas de perfil sobre o usuário. Ele também é interoperável pois segue o protocolo RestFull e usa o formato de saída de dados o JSON, que hoje é um padrão para aplicações web. Além disso, sua especificação suporta vários tipos de clientes, como aplicações que utilizam o browser, clientes javascript, aplicações mobile, etc. A Figura 1 ilustra as requisições da autenticação entre cliente e servidor.

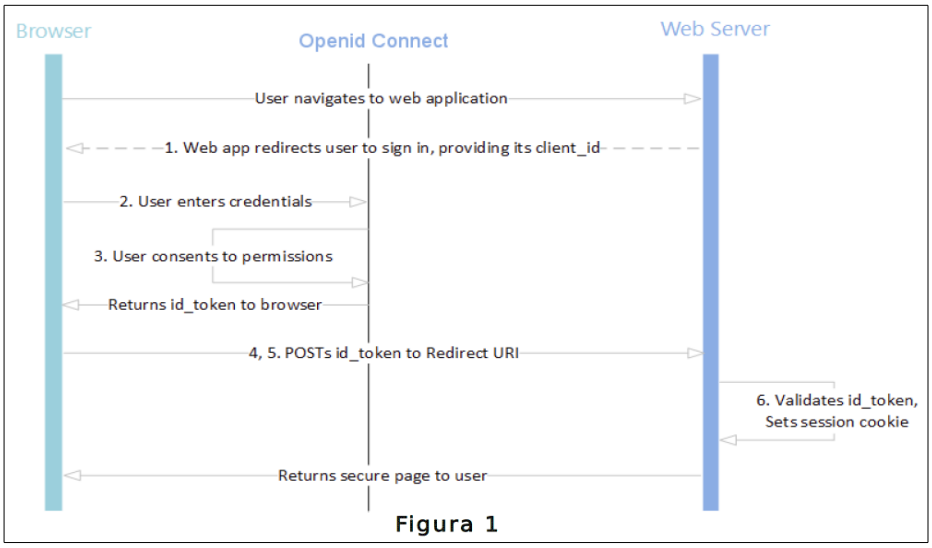

Figura 1

OAUTH2

OAuth é um protocolo aberto para autorização que permite aos clientes obterem acesso a recursos protegidos do servidor em nome do proprietário do recurso. O proprietário do recurso pode ser um cliente ou usuário final. O Oauth também especifica como um usuário final pode autorizar o acesso de terceiros aos seus recursos do servidor sem precisar compartilhar suas credenciais. Atualmente ele está sendo usado largamente pelas grandes empresas como Google, Facebook, Microsoft, Twitter, etc., para permitir que os usuários compartilhem suas informações sobre suas contas com outras aplicações. A concessão da autorização feita pelo proprietário das informações é

representada por outra credencial que pode ser usada para obter um recurso protegido. O Oauth fornece 4 estratégias para concessão de autorização: código de autorização, implícita, credenciais de senha do proprietário do recurso e credenciais do cliente. A estratégia usada no Brasil Cidadão é a Código de autorização, que utiliza um token.

Json Web Token - JWT

O JWT é um padrão aberto que define como transmitir objetos JSON de forma segura entre aplicações. A assinatura pode ser realizada usando uma palavra secreta ou uma chave publica/privada. O JWT é composto por 3 elementos:

Headers
São objetos JSON definidos por 2 atributos. O tipo do token (tpk) que é o JWT e o algorítimo (alg) de encriptação utilizado, como HMAC SHA256 ou RSA. Exemplo:

```
{
"alg": "HS256",
"typ": "JWT"
}
```

Payload
São todos ou parte dos atributos de uma entidade representada por objetos JSON. Exemplo:

```
{
"sub": "1234567890",
"name": "John Doe",
"admin": true
}
```

Signature
Para criar a assinatura temos assinar o header codificado, o payload codificado e informar o secret, que no caso abaixo é uma palavra secreta definida na aplicação. A assinatura é criada para verificar se quem enviou a requisição é quem realmente diz ser.

```
HMACSHA256(
base64UrlEncode(header) + "." +
base64UrlEncode(payload),
secret)
```

O resultado dessa assinatura é um token como abaixo. O token é dividido em 3 partes separadas pelo ponto. As três partes equivalem ao hash do header, payload e a signature.

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwI
iwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9.TJVA95OrM7E2c
Bab30RMHrHDcEfxjoYZgeFONFh7HgQ

Sempre que for necessário acessar recursos protegidos, o cliente deve enviar o token gerado (JWT) através do atributo *Authorizer do header* da requisição, com a *flag Bearer*, como abaixo:

Authorization:Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY
3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9.TJVA95
OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ

Código Autorizador

Essa estratégia se resume em autorizar clientes a acessarem informações dos usuários proprietários através de um código identificador (Cliente ID). Primeiramente, o cliente deve ser cadastrado no Portal de Gestão para obter um Cliente ID. Após estar devidamente habilitado, o proprietário da informação, ao ser requisitado, deve habilitar esse cliente para ter acesso às suas informações.

Após essas configurações, o cliente está habilitado para obter do servidor os recursos necessários. Essa estratégia é muito utilizada no mercado pois é otimizada para as aplicações server-side, o qual o código fonte não é exposto e a confidencialidade do Cliente ID é mantida. A Figura 2 explana a obtenção de recursos do servidor de um cliente previamente cadastrado. No passo A, a aplicação cliente solicita autorização. O usuário realiza a autenticação no Brasil Cidadão, obtêm a autorização e é redirecionado para o cliente (passo B).

No passo C, o cliente solicita o Access Token e o ID Token, que são as credenciais para que ele seja habilitado a realizar as consultas de recursos por um determinado tempo. Essas credenciais são geradas no servidor e é importante que não navegue pelo cliente, para que a confidencialidade seja mantida. Após o cliente ser validado e receber o ID Token e Access Token no passo D, ele pode solicitar ao Brasil Cidadão os recursos necessários. Na Figura 3 mostra os parâmetros necessários para as requisições da Figura 2.

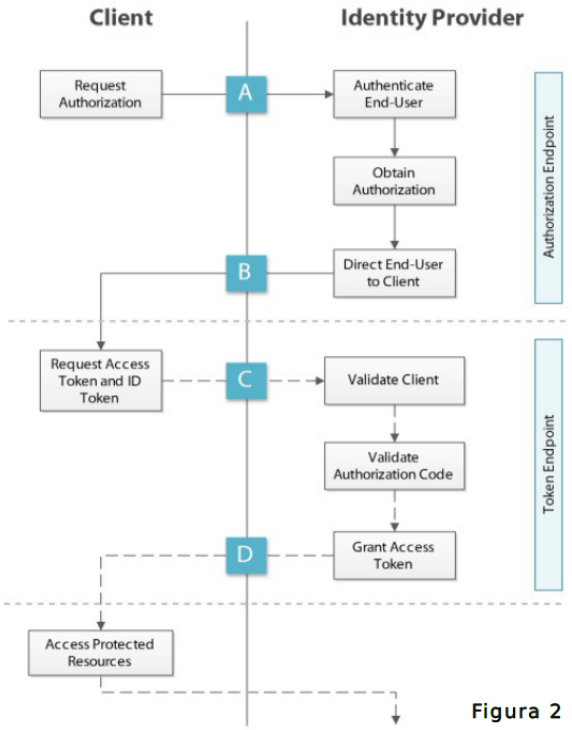

Figura 2

A **Client sends Authorization Request**

Request includes
response_type: `code`
client_id
redirect_uri *(optional; may be pre-configured with service provider)*
scope *(optional)*
state *(recommended)*

B **Service Provider grants Authorization**

Redirection URI (302 Found) includes
code
state *(required IFF state was sent in the request; must be equal to what was received)*

C **Client requests Access Token**

Request includes
grant_type: `authorization_code`
code
redirect_uri *(optional; may be pre-configured with service provider. Must match value sent in step A if provided)*

If the client is confidential, it must authenticate with the Authorization Server in this request.

D **Service Provider grants Access Token**

JSON response object (200 OK) includes
access_token
token_type
expires_in *(optional)*
refresh_token *(optional)*
scope *(optional; SHOULD be included if the scope granted differs from the scope requested)*

Figura 3

Autorização para Recursos protegidos

Na plataforma de autenticação, os serviços utilizam informações pessoais relacionadas aos cidadãos, e, portanto, existe a necessidade de vincular recurso informacional ao serviço no processo de habilitação. Quando o cidadão se autentica via Brasil Cidadão e acessa algum serviço público digital pela primeira vez, uma solicitação de autorização de uso de dados pessoais é feita. É importante que o cidadão autorize o uso de seus dados pessoais, para que o serviço funcione corretamente. Na figura 4, foi ilustrada uma tela exemplificativa, para que o cidadão autorize o uso de recurso protegido (dados pessoais):

Figura 4

Escopos de Atributos

Os escopos são conjuntos de informações definidos de forma a serem fornecidos a quem possuir autorização para obtê-los. No sistema as formas de autenticação e os selos permitem identificar o nível de acesso do usuário. A depender do nível de acesso do usuário e da forma como ele se autenticou no sistema, ele terá autorização para obter um determinado escopo ou não. Na figura 5, foi ilustrada uma tela dos escopos por serviços:

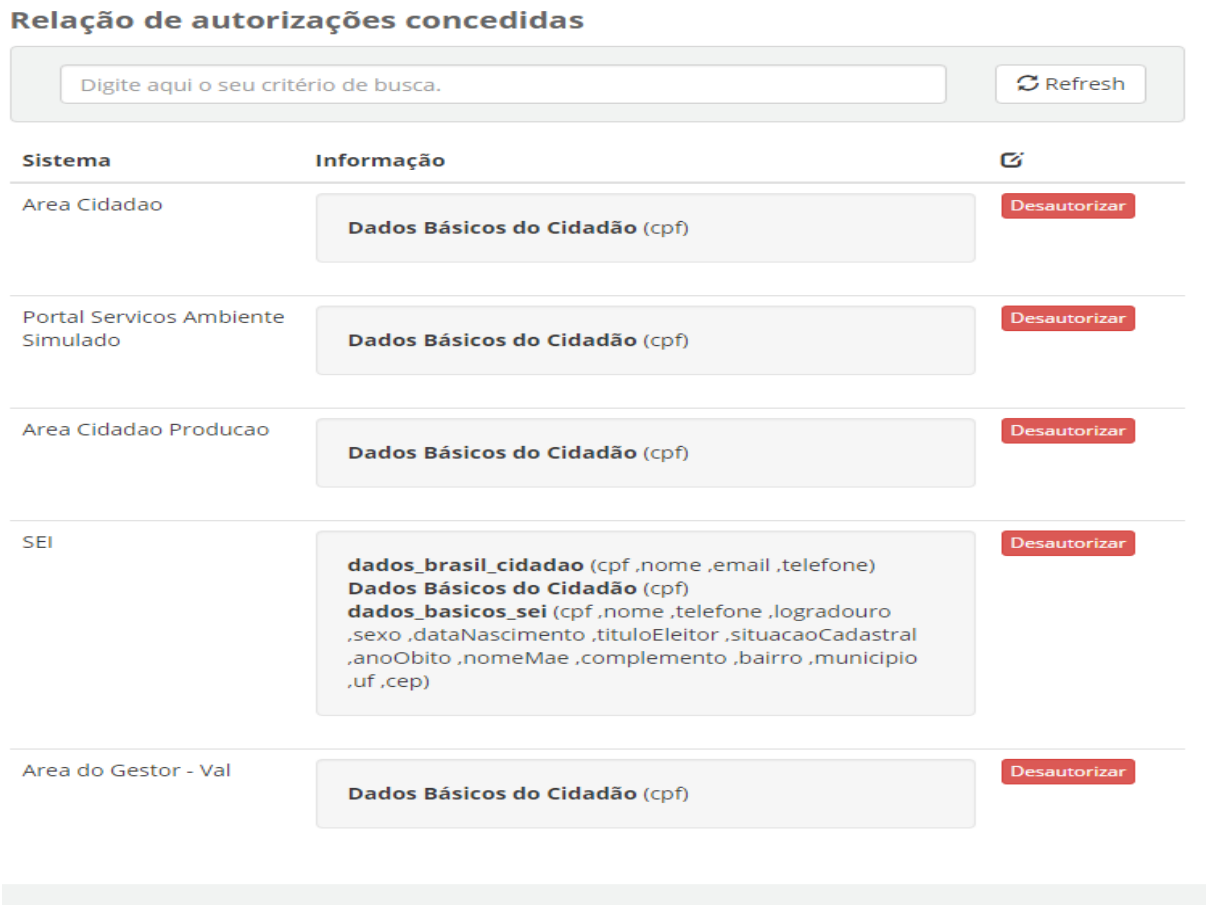

figura 5 – exemplos de escopos de atributos po.

Atributos Disponíveis

Atributos identificadores

- CPF *(cadastrado na base de dados do CPF/RFB)*
- Título de Eleitor *(cadastrado na base de dados do CPF/RFB)*

Atributos complementares

- nome *(cadastrado na base de dados do CPF/RFB)*
- nome (*cadastrado na base de dados do Brasil Cidadão/MP)*
- e-mail *(cadastrado na base de dados do Brasil Cidadão/MP)*
- *e-mail (cadastrado na base de dados do CPF/RFB)*
- telefone *(cadastrado na base de dados do CPF/RFB)*
- telefone (*cadastrado na base de dados do Brasil Cidadão/MP)*
- endereço *(logradouro, bairro, município, UF, cep e complemento, conforme cadastrado na base de dados do CPF/RFB)*
- situação cadastral do CPF *(cadastrado na base de dados do CPF/RFB)*
- ano do óbito *(cadastrado na base de dados do CPF/RFB)*
- nome da mãe *(cadastrado na base de dados do CPF/RFB)*
- data de nascimento *(cadastrado na base de dados do CPF/RFB)*
- naturalidade *(cadastrado na base de dados do CPF/RFB)*
- sexo *(cadastrado na base de dados do CPF/RFB)*

Níveis de Autenticação

Atualmente os tipos de autenticação são Auto Cadastro e Certificado Digital. Cada nível de acesso esta associado a um nível de confiabilidade. Por exemplo, o usuário que se autenticar com o certificado digital terá o nível de acesso 4, pois é um fator de segurança muito alto. Já o usuário que se autentica com usuário e senha do Auto Cadastramento a possibilidade de fraude é maior, dado que alguém pode utilizar as informações do usuário para se passar por ele. O nível de acesso aumentará a abrangência de escopos. São eles:

- Nível de autenticação 1 – AutoCadastro – Identidade cadastrada com conferência simples via e-mail ou SMS.
- Nível de autenticação 2 – Dados cadastrais convalidados – Identidade cadastrada com convalidação de dados com bases oficiais.
- Nível de autenticação 3 – Dados cadastrais certificado – Identidade certificada a partir da conferência de documentos de forma presencial em posto de atendimento de governo.
- Nível de autenticação 4 – Biometria – Identidade cadastrada com convalidação de dados biométricos.

- Nível de autenticação 5 – Cadastro assinado digitalmente – Identidade cadastrada a partir do certificado digital de pessoa física e assinado digitalmente.

Além do tipo de autenticação o usuário pode adquirir Selos para definir o nível de acesso no sistema. Os selos são o nível de acesso garantido, independente da forma que o usuário se autenticou. Por exemplo, se o usuário possuir o certificado digital mas está acessando o sistema pelo celular e quiser manter seu nível de acesso, ele terá que adquirir o selo de certificado digital em um computador, para que ao se autenticar no celular com usuário e senha ele continue tendo acesso as funções e escopos do Nível de acesso 4. A imagem abaixo ilustra o mesmo usuário se autenticando de formas diferentes.

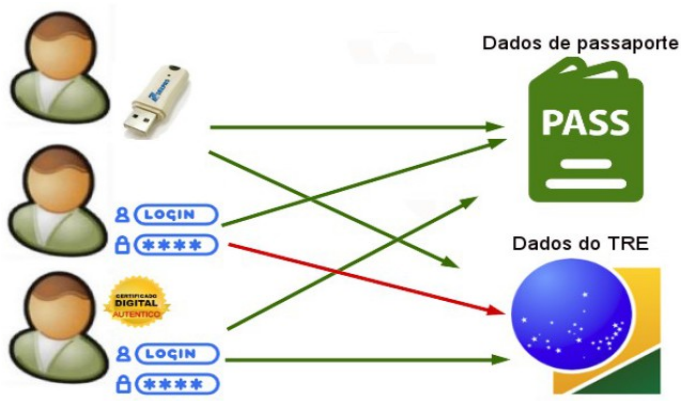

Figura 4

Selos de Confiabilidade Cadastral

Os selos são obtidos, após convalidação com base de dados governamental ou certificado digital ICP-Brasil, o que confere mais confiabilidade ao cadastro realizado via Plataforma Brasil Cidadão. Outra uso possível para o selo é o uso do nível de confiança cadastral pelos serviços para aplicar controle de acesso às funcionalidades mais críticas. Na figura 6, estão ilustrados os selos de confiabilidade cadastral:

figura 6 – selos de confiabilidade cadastral

Serviços e Contratos Expostos

Confira na matriz abaixo os serviços e contratos disponíveis para consumo:

#	Descrição	Contrato	Resultado
01	Operação para verificar se um determinado cidadão possui biometria cadastrada na base dedos biométrica do Tribunal Superior Eleitoral (TSE)	verificarExistenciaCadastroBiometria(<*atributoIdentificador*>)	Retorno: 0 – O CPF/Título de Eleitor <*cpf|tituloEleitor*> não possui biometria cadastrada 1 – O CPF/Título de Eleitor <*cpf|tituloEleitor*> possui biometria cadastrada 5 – Falha: <*mensagem de erro*>
02	Operação para verificar quais selos de confiabilidade cadastral um determinado cidadão possui.	listarSelosConfiabilidaeCadastral(<*atributoIdentificador*>)	Retorno: 0 - <*lista de selos de confiabilidade cadastral*> 1- O cidadão, <*atributo identificador*>, não possui selos de confiabilidade cadastral.
03	Operação para assinar eletronicamente um documento (não contempla assinatura digital).	A definir	
04	Operação para utilizar token OTP	A definir	
05	Batimento Biométrico da digital	A definir	

Autenticação Multifator

A plataforma dispõe de alguns métodos para autenticação multifator, podendo os serviços integrados utilizarem opcionalmente. Na tabela abaixo estão listados os métodos:

Multifatores disponíveis
Biometria da digital
Certificado digital
Token OTP
Código SMS
Assinatura Eletrônica

Identidade Padrão de Comunicação Digital do Governo Federal - IDG

A IDG) é um conjunto de diretrizes, orientações, padrões e modelos a serem aplicados em elementos que compõem a Identidade Digital, como a barra de governo, portais institucionais, sítios temáticos, informativo, redes sociais, guia de serviços, guia de aplicativos e outras ferramentas digitais. A partir da adoção pelos órgãos do Poder Executivo federal das premissas contidas na identidade digital, é esperado alcançar os seguintes objetivos:

- Qualificar a comunicação, permitindo que o cidadão encontre, com mais facilidade, as informações sobre as políticas públicas, equipamentos e serviços ofertados pelo Governo Federal;
- Padronizar as propriedades digitais (ambientes digitais que possuem gestão e chancela de um órgão do governo) e alinhar a estratégia de comunicação dos órgãos do Poder Executivo federal;
- Padronizar as soluções digitais dos órgãos públicos federais e alinhar as informações com foco no cidadão;
- Garantir o acesso a todos, independentemente da forma ou dispositivo de conexão, garantindo a acessibilidade digital e o acesso a qualquer momento.

Com essa padronização, é importante perceber que o sistema consumidor deve estar adaptado ao padrão para que as integrações de telas não sejam impactantes em nível visual para o usuário. Mais informações no site da Comunicação Social da Casa Civil: www.secom.gov.br/atuacao/comunicacao-digital/identidade-digital-1

Iniciando a Integração

Parâmetros de integração

Parâmetros de autenticação do serviço consumidor:

- client_id – chave de acesso, que identifica o serviço consumidor
- client_secret – senha de acesso do serviço consumidor

Parâmetros do *queries string*:

- response_type: especifica para o provedor o tipo de autorização. Esse tipo de autorização gerará um código para o serviço consumidor. Nesse caso sempre usaremos o code.
- client_id: o identificador do serviço consumidor fornecido pelo Portal de Gestão.
- scope: especifica os recursos que o serviço consumidor quer obter. No caso da autenticação é obrigatório concatenar o scope openid, ele retornará as informações do usuário logado.
- redirect_uri: a url do serviço consumidor que o provedor de autenticação redirecionará após o login e a autorização.
- nonce: sequência de caracteres usado para associar uma sessão do serviço

consumidor a um Token de ID e para atenuar os ataques de repetição. Pode ser um valor aleatório, mas que não seja fácil de advinhar.

- state: valor usado para manter o estado entre a solicitação e o retorno de chamada.

Os parâmetros nonce e state representam variáveis de controle que são utilizadas para autenticidade por parte do Consumidor.

Métodos e interfaces de integração

"O Passo-a-Passo para Integrar"

Autenticação
Para que a autenticação aconteça, todo o canal de comunicação deve ser realizado com o protocolo HTTPS.

1) Ao requisitar autenticação via Provedor, o mesmo verifica se o usuário está logado. Caso o usuário não esteja logado o provedor redireciona para a página de login;

```
http://e cidadao .serpro /scp/authorize ?
response_type=code&
client_id=ec4318d6-f797-4d65-b4f7-39a33bf4d544&
scope=openid+DadosComplementaresRFB+DadosBasicosRFB&
redirect_uri=http://appcliente.com.br/phpcliente/loginecidadao.
Php&
nonce=3ed8657fd74c&
state=358578ce6728b
```

2) Após autenticado, o provedor redireciona para a página de autorização. O usuário habilitará o consumidor no sistema para os escopos solicitados:

Caso o usuário da solicitação autorize o acesso ao recurso protegido, é gerado um "ticket de acesso" intitulado accesstoken (vide especificação OAUTH 2.0).

3) Após a autorização, a requisição é retornada para a URL especificada no passo 1, enviando os seguintes parâmetros:

```
code=Z85qv1
state=358578ce6728b
```

Para obter o token e o access_token, o consumidor deve fazer uma requisição POST passando as seguintes informações no header :

Lembrando que para essa requisição o code têm um tempo de expiração e só pode ser utilizado uma única vez.

```
Content-Type:application/x-www-form-urlencoded
Authorization:Basic
ZWM0MzE4ZDYtZjc5Ny00ZDY1LWI0ZjctMzlhMzNiZjRkNTQ0OkFJSDRoaXBf
TUJYcVJkWEVQSVJk
WkdBX2dRdjdWRWZqYlRFT2NWMHlFQll4aE1iYUJzS0xwSzRzdUVkSU5FcS1
kNzlyYWpaZ3I0SGJu VUM2WlRXV1lJOA==
```

E no atributo data da requisição:

- grant_type: authorization_code
- code: Z85qv1
- redirect_uri: http://ecidadao.serpro/phpcliente-val/login-ecidadao.php
- Content-Type: tipo do conteúdo da requisição que está sendo enviada. Nesse caso estamos enviando como um formulário.
- Authorization: Informação codificada na Base64, no seguinte formato: CLIENT_ID:SECRET. A palavra Basic deve vim antes da informação codificada.
- grant_type: no nosso caso o valor do grant_type será sempre authorization_code, pois estamos autenticando com o código.
- code: código retornado pela requisição anterior
- redirect_uri: a url do sistema consumidor que irá receber as informações de token, access_token

O serviço retornará a informação no formato JSON.

```
{
"token":"RsT5OjbzRn430zqMLgV3Iahgsdfjgsjfg JHKJHJuyiiuyU"
"access_token":"RsT5OjbzRn430zqMLgV3Ia"
}
Ou se ocorrer erro:
{
"error":"invalid_request"
}
```

4) De posse das informações de *token e access token*, a aplicação consumidora já está habilitada para consultar dados de recursos protegidos, que são os escopos de informações. As informações são disponibilizadas através de um barramento de serviços. Este barramento é o serviço que fornece as informações dos recursos. Para utlização do barramento é obrigatório a informação do *token e access token* válidos.

```
https://servicos.serpro.gov.br/servicosecidadao/
ecidadao/usuario/getUserInfo/DadosBasicosRFB?
access_token=eyJraWQiOiJyc2ExIiwiYWxnIjoiUlMyNTYifQ.eyJzdWIiOiI4Mzk
xNzU5NDU4NyIsImF6cCI6ImVjNDMxOGQ2LWY3OTctNGQ2NS1iNGY3LTM5
YTMzYmY0ZDU0NCIsInNjb3BlIjpbIkRhZG9zQmFzaWNvc1JGQiIsIm9wZW5p
```

ZCIsIkRhZG9zQ29tcGxlbWVudGFyZXNSRkIiXSwiaXNzIjoiaHR0cHM6XC9cL
3ZhbGFjZXNzby5lY2lkYWRhby5zZXJwcm8uZ292LmJyXC9zY3BcLyIsImV4c
CI6MTQ4NDY1ODUzOSwiaWF0IjoxNDg0NjU4NDgwLCJqdGkiOiIyYWQwYTY
3Yi05YTc0LTRiYTEtODBjZC00NjQ1MmYwNjhjMTgifQ.MZgjefUUF97zjePPyG
LhIVjwSSoVjThXItgPwH4ph1UBpz2i6pUEgttdThNQPvVWMRf6-
akDWx2UopVoQqv8kJ3i1JmyD8
-SMWKqARgl_9d8pF6wq9nahdJFmr7UDK0triuLR7Gh6wym0YD9T8KdVL73FEFBIUR0Lf9Gg
T1ocV3YRblnO33kxDd0YzPCQ1znBtw9Wf5ZzU4J8ABafM2g
p0v09oY5lFvWw-iJg5i5oKjQyf7tBD1KBoK2l4MMspISYm-
W2uyNwwgCb93m57bIfswyEeque9DM624kDt7LPciD8VAu5OeTBZlC5XlCD
AXV-d8lzbaZcsB6SwSHdvFQ

DadosBasicosRFB: um dos recursos solicitados na tela de login *access_token*: valor do *access token* retornado pela requisição anterior A requisição retornará os dados no formato abaixo (JSON):

{
"naturalidade":"Ilha das Flores",
"cpf":"88918894588",
"nome":"HENRIQUE PRETORIUM ",
"sexo":"M",
"dataNascimento":"1980-12-06",
"email":"henrique.pretorium@enterprisex.gov.br"
}

Caso a requisição seja inválida, ou o *access token* inválido, o Barramento retornará Status *401 – Unauthorized.*

Boas práticas de programação

Abaixo estão algumas boas práticas de programação recomendadas:

- Implementar CACHE de chave pública;

Exemplos de implementação

Os exemplos abaixo são exemplos básicos da forma de realizar as requisições para Brasil Cidadão. Cabe ao desenvolvedor realizar a organização e aplicação da segurança necessária na aplicação consumidora.

PHP 7.0
1) Requisição para login

```
$uri = "https://ecidadao.serpro/scp/authorize?response_type=code"
. "&client_id=". CLIENT_ID
. "&scope=openid+DadosComplementaresRFB+DadosBasicosRFB"
```

```
. "&redirect_uri=" . urlencode(REDIRECT_URI)
. "&nonce=3ed8657fd74c"
. "&state=358578ce6728b";
header('Location: ' . $uri);
```

2) Após o login, requisição para obtenção do token

```
$code = $_REQUEST["code"];
$campos = array(
'grant_type' => urlencode('authorization_code'),
'code' => urlencode($code),
'redirect_uri' => urlencode(REDIRECT_URI)
);
foreach($campos as $key=>$value) {
$fields_string .= $key.'='.$value.'&';
}
rtrim($fields_string, '&');
$ch = curl_init();
curl_setopt($ch,CURLOPT_URL, https://ecidadao.serpro/scp/token);
curl_setopt($ch,CURLOPT_POST, count($fields));
curl_setopt($ch,CURLOPT_POSTFIELDS, $fields_string);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE);
curl_setopt($ch,CURLOPT_SSL_VERIFYPEER, false);
$headers = array(
'Content-Type:application/x-www-form-urlencoded',
'Authorization: Basic '. base64_encode(CLIENT_ID.":".SECRET)
);
curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);
$result = curl_exec($ch);
curl_close($ch);
var_dump($ch);
$json_output = json_decode($result, true);
$access_token = $json_output["access_token"];
$id_token = $json_output["id_token"];
$scope = $json_output["scope"];
```

3) De posse do token, requisição para obter dados de um escopo

```
$escopo = "DadosBasicosRFB";
$url = "https://ecidadao.serpro/servicosecidadao/
ecidadao/usuario/getUserInfo/" . $escopo . "?access_token=".
access_token_val;
$ch = curl_init();
curl_setopt($ch,CURLOPT_SSL_VERIFYPEER, false);
curl_setopt($ch,CURLOPT_URL, $url);
```

```
curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE);
$result = curl_exec($ch);
$json_output = json_decode($result, true);
```

Java JSP Jdk 1.8

1) Requisição para login

```
Index.jsp
<form action="login" method="get">
<input type="submit" value="Login"/>
</form>
login.jsp
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<body>
<h1>Cliente JSP</h1>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%
String url = (String)request.getAttribute("url_login");
response.setStatus(response.SC_MOVED_TEMPORARILY);
response.setHeader("Location", url);
%>
</body>
</html>
Controller
private static final String VIEW_LOGIN = "login";
private void disableSSLCertificateChecking() {
TrustManager[] trustAllCerts = new TrustManager[] { new
X509TrustManager() {
public X509Certificate[] getAcceptedIssuers() {
return null;
}
@Override
public void checkClientTrusted(X509Certificate[] arg0, String
arg1) throws CertificateException {
// Not implemented
}
@Override
public void checkServerTrusted(X509Certificate[] arg0, String
arg1) throws CertificateException {
// Not implemented
}
```

```
} };
try {
SSLContext sc = SSLContext.getInstance("TLS");
sc.init(null, trustAllCerts, new
java.security.SecureRandom());
HttpsURLConnection.setDefaultSSLSocketFactory(sc.getSocketFactory());

} catch (KeyManagementException e) {
e.printStackTrace();
} catch (NoSuchAlgorithmException e) {
e.printStackTrace();
}
}
@RequestMapping(value="/login", method = RequestMethod.GET)
public String loginApp(ModelMap model) {
disableSSLCertificateChecking();
String urlLogin = url+"/authorize?
response_type=code&client_id="+id+"&scope="+scope+"&redirect_uri="+redirec
t_uri+"&nonce="+nonce+"&state="+state;
model.addAttribute("url_login",urlLogin);
return VIEW_LOGIN;
}
```

2) Após o login, requisição para obtenção do token e os dados de um escopo:

```
index.jsp
<form action="token.html" method="get">
<input type="hidden" name="code" value="<
%=request.getParameter("code")%>">
<input type="submit" value="Get Token"/>
</form>
token.jsp
Id Token: <%=(String)request.getAttribute("id_token")%>
<br><br>
<br>DADOS BÁSICOS RECEITA [DADOSBASICOSRFB]
<br>Nome: <%=(String)request.getAttribute("nome")%>
<br>Sexo: <%=(String)request.getAttribute("sexo")%>
<br>Data Nascimento: <%=(String)request.getAttribute("dataNascimento")%>
<br>Email: <%=(String)request.getAttribute("email")%>
<br>Naturalidade: <%=(String)request.getAttribute("naturalidade")%>
Controller
@RequestMapping(value="/token.html",params={"code"}, method =
RequestMethod.GET)
```

```
public String token(@RequestParam(value = "code") String code, ModelMap
model) throws ProtocolException {
logger.debug("[Token]");
model.addAttribute("url",url);
String urlTokenStr = url+"/token";
String urlParameters =
"grant_type=authorization_code&code="+code+"&redirect_uri=http%3A%2F%2Flocalhost
%3A8080%2Fapp%2Findex.html";
String userCredentials = id+":"+secret;
String basicAuth = "Basic " + new
String(Base64Utils.encode(userCredentials.getBytes()));
String accessToken = "";
String result = sendPost(urlTokenStr,urlParameters, basicAuth);
logger.debug("[Result:]"+result);
JSONObject jsonObj;
try {
jsonObj = new JSONObject(result);
accessToken = jsonObj.getString("access_token");
model.addAttribute("id_token",jsonObj.getString("id_token"));
} catch (JSONException e) {
logger.debug("[Erro ao transformar objeto json]");
e.printStackTrace();
}
// System.out.println("Token:"+jsonObj.getString("id_token"));
// System.out.println("Access-token:"+accessToken);
//Obtendo o recurso
if (accessToken!=null){
String urlService = urlServico+"/DadosBasicosRFB?
access_token="+accessToken;
String result2 = sendGet(urlService);
System.out.println("Dados Usuário "+result2);
try {
jsonObj = new JSONObject(result2);
model.addAttribute("nome",jsonObj.getString("nome"));
model.addAttribute("naturalidade",jsonObj.getString("nat
uralidade"));
model.addAttribute("dataNascimento",jsonObj.getString("d
ataNascimento"));
model.addAttribute("sexo",jsonObj.getString("sexo"));
model.addAttribute("email",jsonObj.getString("email"));
} catch (JSONException e) {
logger.debug("[Erro ao transformar objeto json]");
e.printStackTrace();
```

```
}
}
return VIEW_TOKEN;
}
private String sendPost(String url, String urlParameters, String
basicAuth) {
URL urlURI = null;
try {
urlURI = new URL(url);
} catch (MalformedURLException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
HttpURLConnection connection = null;
try {
connection = (HttpURLConnection) urlURI.openConnection();
connection.setRequestMethod("POST");
if (basicAuth != ""){
connection.setRequestProperty ("Authorization",
basicAuth);
connection.setRequestProperty("Content-Type",
"application/x-www-form-urlencoded;charset=UTF-8;");
connection.setRequestProperty("Content-Length", "" +
Integer.toString(urlParameters.getBytes().length));
connection.setUseCaches(false);
connection.setDoInput(true);
connection.setDoOutput(true);
connection.setInstanceFollowRedirects(false);
} catch (ProtocolException e) {
logger.debug("[ERROR ProtocolException]");
e.printStackTrace();
} catch (IOException e) {
logger.debug("[ERROR IOException]");
e.printStackTrace();
}
DataOutputStream wr;
try {
wr = new DataOutputStream(connection.getOutputStream ());
if (urlParameters != ""){
wr.writeBytes(urlParameters);
wr.flush();
}
BufferedReader rd = new BufferedReader(new
```

```
InputStreamReader(connection.getInputStream()));
String line;
StringBuffer result = new StringBuffer();
while ((line = rd.readLine()) != null) {
result.append(line);
}
wr.close();
connection.disconnect();
return result.toString();
} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
return "";
}
}
private String sendGet(String url) {
try{
URL urlURI = null;
URL obj = new URL(url);
HttpURLConnection connection = (HttpURLConnection)
obj.openConnection();
connection.setRequestMethod("GET");
int responseCode = connection.getResponseCode();
System.out.println("Response Code : " + responseCode);
BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
response.append(inputLine);
}
in.close();
//print result
return response.toString();
} catch (ProtocolException e) {
logger.debug("[ERROR ProtocolException]");
e.printStackTrace();
} catch (IOException e) {
logger.debug("[ERROR IOException]");
e.printStackTrace();
}
return "";
}
```

Sites Úteis

- OpenId Connect - http://openid.net/developers/specs/
- OAuth 2.0 - https://oauth.net/2/
- Tutotial Oauth - https://www.digitalocean.com/community/tutorials/anintroduction-to-oauth-2
- JWT - https://jwt.io/introduction/
- Tutorial JWT - https://rafaell-lycan.com/2016/autenticacao-jwt-angular-app/
- Transformador Base64 - http://www.motobit.com/util/base64-decoderencoder.asp
- Identidade Padrão de Comunicação Digital do Governo Federal - http://www.secom.gov.br/atuacao/comunicacao-digital/identidade-digital-1
- Plataforma de Cidadania Digital - http://www.planejamento.gov.br/cidadaniadigital

Canais de Socorro

Central de Serviços e Suporte do SISP

https://c3s.sisp.gov.br/cau/