Ministério do Planejamento
Secretaria de Tecnologia da Informação e Comunicação

Acesso Digital Único da Plataforma de Cidadania Digital (Brasil Cidadão)

Roteiro de Integração (SDK)

Sumário

Contexto

O Decreto n° 8.936, de 19 de dezembro de 2016, pemititu o inicio do 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, tendo, como destaque no decreto, o mecanismo de acesso digital único.

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. Problemas enfrentados por cidadãos ao tentar consumir um serviço público digital oferecido pelo governo federal. Analisando 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.

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 documento é o elemento para orientar a integração da Plataforma de Autenticação Digital do Cidadão – Brasil Cidadão a qualquer ambiente. 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 à Plataforma.

Este documento contém as formas de chamadas a operações, parâmetros e métodos de integração, e, por último, os procedimentos para permitir a conectividade entre os ambientes de implantação.

Arquitetura de Serviço e Protocolos

OpenID Connect (OIDC)

O Openid Connect é um protocolo baseado no OAuth 2.0 que especifica autorização e autenticação. Define como implementar o gerenciamento de autorizações de acesso, gerenciamento de sessão, fornecimento de informações sobre usuário logado. O OIDC permite executar o logon único dos usuários e apresenta o conceito de um *id token*: um *token* de segurança que permite verificar a identidade do usuário e obter informações básicas sobre o usuário. Tem característica de ser interoperável, porque segue o protocolo *RestFull* e usa o formato de saída de dados: JSON (*JavaScript Object Notation*).

Além disso, o OIDC suporta vários tipos de clientes, como aplicações que utilizam o *browser*, clientes *javascript*, aplicações mobile e outros. A Figura 1 ilustra as requisições da autenticação entre cliente e servidor.

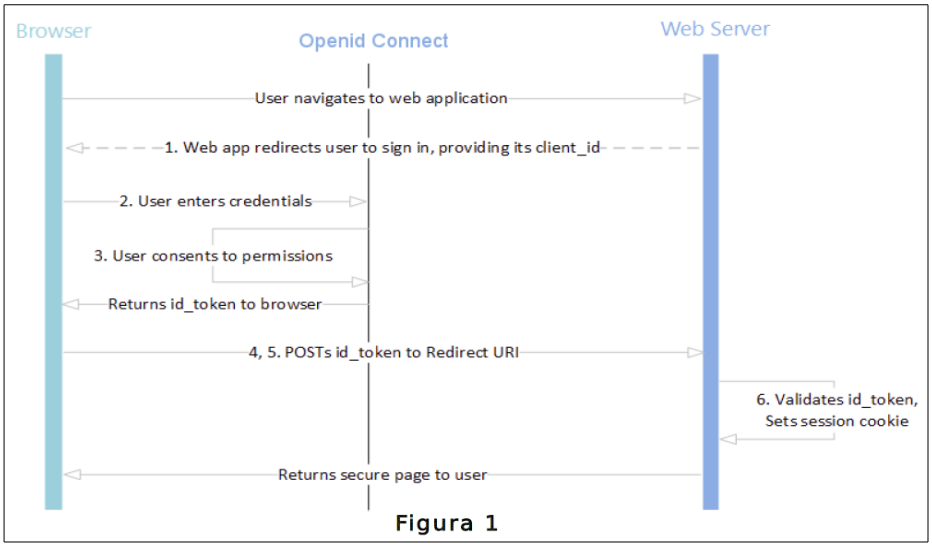

Figura 1

OAUTH2

OAuth2 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 pode ser um cliente ou usuário final. 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 por grandes empresas como Google, Facebook, Microsoft, Twitter, e outros.

O protocolo 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 o código de autorização, que utiliza um *token*.

Json Web Token - JWT

O JWT define como transmitir objetos JSON de forma segura entre aplicações. Tem a característica de ser um padrão aberto.

A manipulação do padrão possui uma assinatura a ser realizada com uma palavra secreta ou uma chave publica/privada.

O JWT é composto por 3 elementos: *Headers*, *Payload e Signature,* explicados a seguir.

Headers

São objetos JSON definidos por 2 atributos: tipo do token(typ) e o algorítimo (alg) de encriptação , como SHA256 ou RSA. Exemplo:

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

Payload

São os atributos de uma entidade representada por objetos JSON. Exemplo:

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

Signature

Para criar a assinatura, há necessidade de assinar o header codificado, o payload codificado e informar o secret, palavra secreta definida na aplicação. A assinatura é criada para verificar se quem enviou a requisição é quem realmente diz ser.

O resultado é um token (exemplo 1 – token gerado). O token é dividido em 3 partes separadas pelo ponto. As três partes equivalem ao hash do header, payload e a signature.

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9(header).eyJzdWIiOiIxMjM0NTY3ODkwI
iwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9(payload).TJVA95OrM7E2c
Bab30RMHrHDcEfxjoYZgeFONFh7HgQ(signature)
```

Exemplo 1 – token gerado

Para acessar recursos protegidos, o cliente deve enviar o token gerado através do atributo *Authorization* do *header* da requisição, com a *flag Bearer*, como abaixo:

```
Authorization:Bearer
```

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY
3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9.TJVA95
OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ

Código Autorizador

A estratégia é autorizar clientes a acessarem informações dos usuários proprietários através de um código identificador (Cliente ID). O cliente deve ser cadastrado no Portal de Gestão do Brasil Cidadão para obter um Cliente ID. Após , o proprietário da informação, ao ser requisitado, deve habilitar esse cliente para ter acesso às suas informações. 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 explica a obtenção de recursos do servidor para um cliente cadastrado.

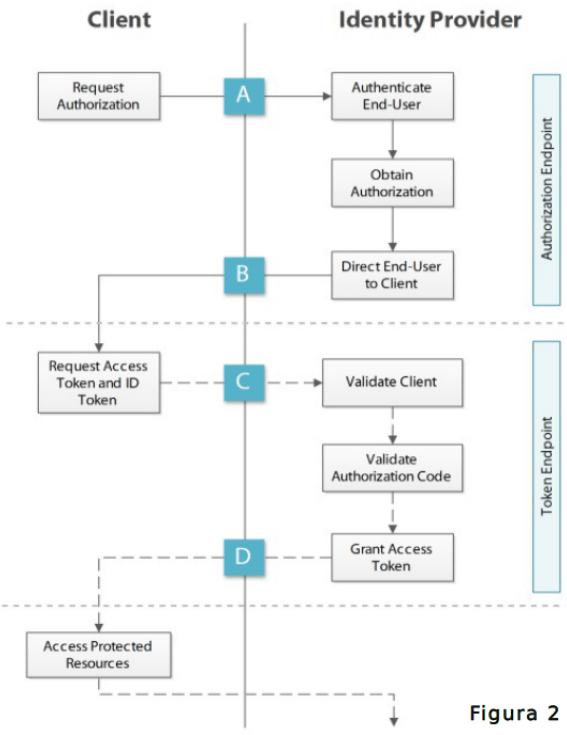

Figura 2 – Obtenção de recursos do Servidor para cliente cadastrado

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, conforme o passo B. Já no passo C, o cliente solicita o *Access Token* e o *ID Token*, que são as credenciais para permitir as consultas de recursos por um determinado tempo. As credenciais são geradas no servidor e não podem navegar pelo cliente, para manter a confidencialidade. 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. A Figura 3 mostra os parâmetros necessários para as requisições da Figura 2.

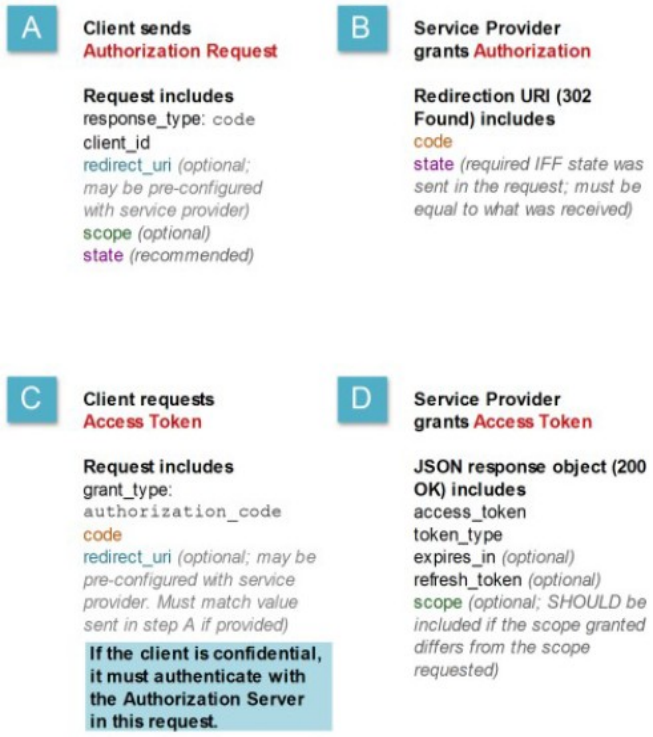

Figura 3

Figura 3 – Explicação dos parâmetros necessários para requisições presentes na figura 2

Autorização para Recursos protegidos

Na plataforma de autenticação, os serviços utilizam informações pessoais relacionadas aos cidadãos, logo, existe a necessidade de vincular recurso informacional ao serviço no processo de habilitação. Quando o cidadão se autentica e acessa algum recurso pela primeira vez, uma solicitação de autorização de uso de dados pessoais é feita. A autorização do uso de dados pessoais permite o funcionamento correto. A figura 4 apresenta a tela em que o cidadão autoriza o uso de recurso protegido (dados pessoais):

Figura 4 – Autorização de Uso de Dados Pessoais

Escopos de Atributos

São conjuntos de informações fornecidos a quem possui autorização.
A figura 5 apresenta tela dos escopos por serviços:

Figura 5 – exemplos de escopos de atributos.

Atributos Disponíveis

Existem dois escopos disponibilizados pelo Brasil Cidadão para apresentar os atributos disponíveis:

- brasil_cidadao (CPF, Nome, e-mail, telefone, foto);
- brasil_cidadao_empresa (CNPJ, Nome Fantasia, CPF do Responsável, Nome do Responsável, Atuação no CNPJ).

Níveis de Autenticação

Tem como principal característica ser um recurso de segurança da informação das identidades, que permite flexibilidade para realização do acesso e atribuição de níveis em conformidade com as normas de segurança. São eles:

- **Nível 1** (Auto Cadastro): Identidade cadastrada com conferência simples;
- **Nível 2** (Dados cadastrais convalidados): Identidade cadastrada com convalidação de dados em bases oficiais;
- **Nível 3** (Dados cadastrais Nível Balcão): Identidade certificada a partir da conferência de documentos de forma presencial em posto de atendimento de governo;
- **Nível 4** (Biometria) : Identidade cadastrada com convalidação de dados biométricos;
- **Nível 5** (Cadastro assinado digitalmente): Identidade cadastrada a partir do certificado digital de pessoa física e assinado digitalmente.

Selos de Confiabilidade Cadastral

Consistem em compor níveis de segurança das contas com a obtenção dos atributos autoritativos do cidadão a partir das bases oficias de governo, por meio das quais permitirão a utilização da credencial de acesso em sistemas internos dos clientes e serviços providos diretamente ao cidadão.

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, demonstra os selos de confiabilidade cadastral na área cidadão do Brasil Cidadão:

Figura 6 – selos de confiabilidade cadastral

Lançador de Serviços

O Brasil Cidadão apresenta o conceito de Single Sign On na arquitetura: um ponto único de autenticação, que permite ao usuário fazer o login e acessar diversos serviços ou sistemas integrados. Para permitir isto de forma explicita, o Brasil Cidadão apresenta um funcionalidade chamada Lançador de Serviços.

Esta funcionalidade lista todos os serviços ou sistemas integrados autorizados uma vez pelo cidadão. A figura abaixo demonstra o Lançador de Serviços ativo e as configurações são entregues pelos serviços integrados ao Brasil Cidadão.

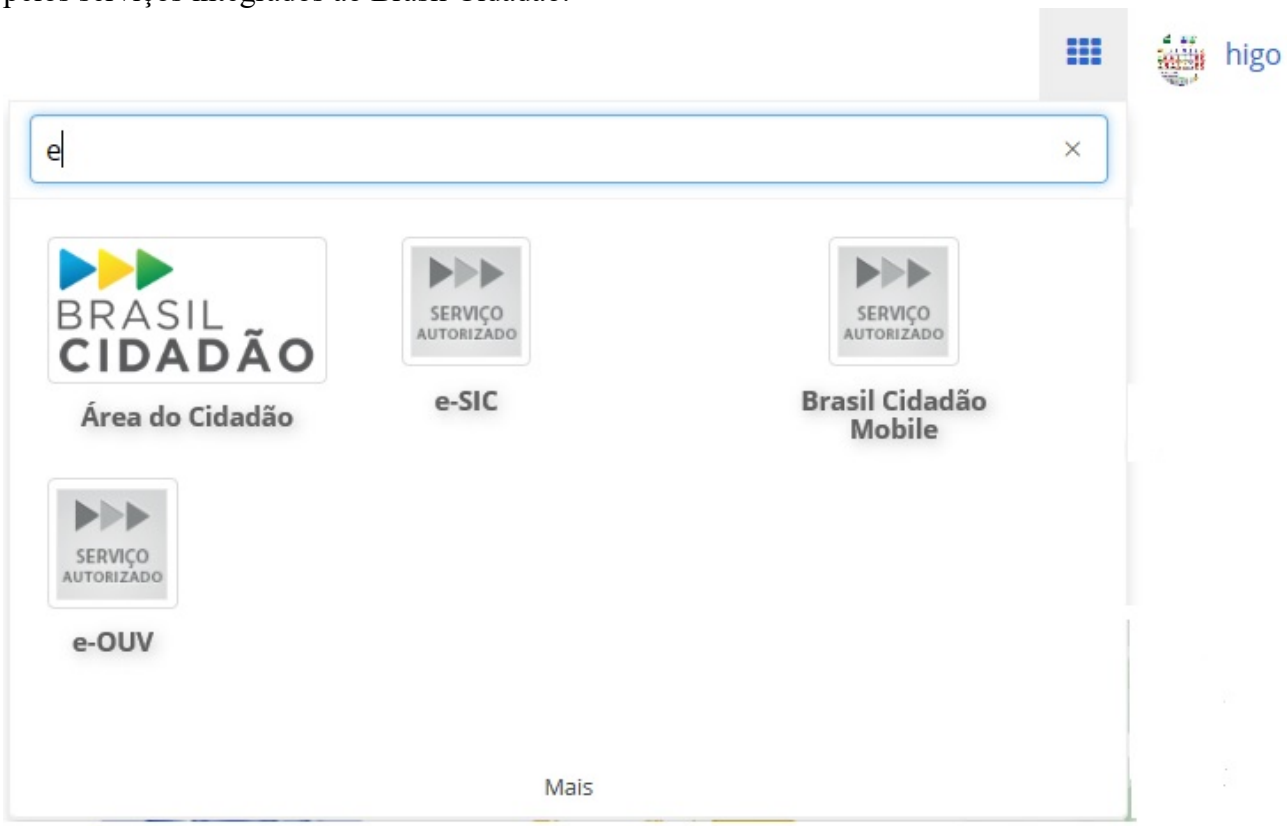

Figura 7 – Lançador de Serviços da Área do Cidadão do Brasil Cidadão.

Iniciando a Integração

Solicitação de Configuração

Para utilização do sistema Brasil Cidadão, há necessidade de liberar os ambientes para aplicação cliente possa utilizar. Essa liberação ocorre por meio do preenchimento do plano de configuração encaminhado junto com este documento (**plano-configuracao-brasil-cidadao-vX.doc**).

O formulário deverá ser encaminhado para os integrantes da Secretaria de Tecnologia da Informação e Comunicação (SETIC) do Ministério do Planejamento para realizar configuração da utilização do Brasil Cidadão.

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***" para retornar 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 de fácil dedução;
- *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. Não são obrigatórios o trabalho pelo Consumidor.

Métodos e interfaces de integração (Passo-a-Passo para Integrar)

URLs importantes

URL do Provider: https://testescp-ecidadao.estaleiro.serpro.gov.br

URL de Serviços: https://testeservicos-ecidadao.estaleiro.serpro.gov.br/servicos-ecidadao/ecidadao

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.
2. A requisição é feita através de um GET para o endereço https://*(url do provider brasil cidadão)*/scp/authorize passando as seguintes informações:
    1. Parâmetros query:
        - response_type: ***code***;
        - *client_id*: <CLIENT_ID> fornecido pelo Brasil Cidadão para a aplicação cadastrada;
        - *scope*: um ou mais escopos inseridos para a aplicação cadastrada. Se for mais de um, esta informação deve vir separada pelo caracter "+".
        - *redirect_uri*: <URI de retorno cadastrada para a aplicação cliente>.
        - *nonce*: <sequência alfa numérica aleatória>.
        - *state*: <sequência alfa numérica aleatória>. Item não obrigatório.

        Exemplo de requisição:

https://*(url do provider brasil cidadão)*/scp/authorize?response_type=code&client_id=ec4318d6-f797-4d65-b4f7-39a33bf4d544&scope=openid+brasil_cidadao&redirect_uri=http://appcliente.com.br/phpcliente/loginecidadao.Php&nonce=3ed8657fd74c&state=358578ce6728b

3. 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 *access_token* (*vide especificação OAUTH 2.0*);

4. Após a autorização, a requisição é retornada para a URL especificada no passo 1, enviando os seguintes parâmetros: ***code***=Z85qv1e ***state***=358578ce6728b. Lembrando que para essa requisição o *code* têm um tempo de expiração e só pode ser utilizado uma única vez;

5. Para obter o *token* e o *access token*, o consumidor deve fazer uma requisição *POST* para o endereço https://*(url do provider brasil cidadão)*/scp/token passando as seguintes informações:
    1. No *header* :
        - *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:CLIENT_SECRET. A palavra ***Basic*** deve está antes da informação.

Exemplo de header:

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

Authorization: Basic

ZWM0MzE4ZDYtZjc5Ny00ZDY1LWI0ZjctMzlhMzNiZjRkNTQ0OkFJSDRoa
XBfTUJYcVJkWEVQSVJkWkdBX2dRdjdWRWZqYlRFT2NWMHlFQll4aE1i
YUJzS0xwSzRzdUVkSU5FcS1kNzlyYWpaZ3I0SGJu VUM2WlRXV1lJOA==

2. E os parâmetros *query*:
   - *grant_type*: authorization_code*;*
   - *code:* <código retornado pela requisição anterior> *(exemplo: Z85qv1);*
   - *redirect_uri:* <URI de retorno cadastrada no Brasil Cidadão> *(exemplo: http://ecidadao.serpro/phpcliente-val/login-ecidadao.php)*;

Exemplo de requisição

https://*(url do provider brasil cidadão)*/scp/token?grant_type=authorization_code&code=*Z85qv1*&redirect_uri=http://appcliente.com.br/phpcliente/loginecidadao.Php

O serviço retornará, em caso de sucesso, a informação, no formato JSON, conforme exemplo:

```
{
"token":"RsT5OjbzRn430zqMLgV3Iahgsdfjgsjfg JHKJHJuyiiuyU"
"access_token":"RsT5OjbzRn430zqMLgV3Ia"
}
```

Ou , no caso de falha, a informação, conforme exemplo abaixo:

```
{
"error":"invalid_request"
}
```

6. 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. Esse barramento é o serviço que fornece as informações dos recursos. Para utilização do barramento é obrigatório a informação do *token e access token* válidos, conforme exemplo abaixo:

https://(url de serviços do brasil cidadão)
/usuario/getUserInfo/{brasil_cidadao}?
access_token=eyJraWQiOiJyc2ExIiwiYWxnIjoiUlMyNTYifQ.eyJzdWIiOiI4Mzk
xNzU5NDU4NyIsImF6cCI6ImVjNDMxOGQ2LWY3OTctNGQ2NS1iNGY3LTM5
YTMzYmY0ZDU0NCIsInNjb3BlIjpbIkRhZG9zQmFzaWNvc1JGQiIsIm9wZW5p
ZCIsIkRhZG9zQ29tcGxlbWVudGFyZXNSRkIiXSwiaXNzIjoiaHR0cHM6XC9cL
3ZhbGFjZXNzby5lY2lkYWRhby5zZXJwcm8uZ292LmJyXC9zY3BcLyIsImV4c
CI6MTQ4NDY1ODUzOSwiaWF0IjoxNDg0NjU4NDgwLCJqdGkiOiIyYWQwYTY
3Yi05YTc0LTRiYTEtODBjZC00NjQ1MmYwNjhjMTgifQ.MZgjefUUF97zjePPyG
LhIVjwSSoVjThXItgPwH4ph1UBpz2i6pUEgttdThNQPvVWMRf6-
akDWx2UopVoQqv8kJ3i1JmyD8-
SMWKqARgl_9d8pF6wq9nahdJFmr7UDK0triuLR7Gh6wym0YD9T8KdVL73FEFBIUR0Lf9Gg
T1ocV3YRblnO33kxDd0YzPCQ1znBtw9Wf5ZzU4J8ABafM2g
p0v09oY5lFvWw-iJg5i5oKjQyf7tBD1KBoK2l4MMspISYm-
W2uyNwwgCb93m57bIfswyEeque9DM624kDt7LPciD8VAu5OeTBZlC5XlCD
AXV-d8lzbaZcsB6SwSHdvFQ

Parâmetros do exemplo anterior:

- brasil_cidadao: um dos escopos que foram cadastrados para a aplicação e autorizados pelo usuário detentor do *access token*;
- *access_token*: valor do *access token* retornado pela requisição POST anterior. Para o caso de sucesso onde o *access token* é válido e o escopo foi autorizado, o barramento de serviços retornará os dados no formato *JSON*, conforme exemplo abaixo:

{
"cpf":"88918894588",
"nome":"HENRIQUE PRETORIUM ",
"email":"henrique.pretorium@enterprisex.gov.br",
“telefone”:”00000000”,
“foto”:”informacao da foto em formato base 64 com tamanho até 4 MB”

}

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

Resultados Esperados do Acesso ao Serviços de Autenticação

Os acessos aos serviços do Brasil Cidadão ocorrem por meio de chamadas de *URLs* e a resposta são códigos presentes conforme padrão do protocolo http. Estes códigos são:

- **Código 200:** Dados acessados e retornados em formato *JSON* ao usuário, de acordo com o *JSON* de cada escopo;

- **Código 400:** *Token* recebido por mais de um método;
- **Código 401:** *Token* não encontrado ou inválido , CPF inválido, usuário não existente no sistema, *access token* inválido;
- **Código 403:** Escopo solicitado não autorizado pelo usuário;
- **Código 404:** Escopo obrigatório.

Acesso ao Serviço de Cadastro de Pessoas Jurídicas

O Brasil Cidadão disponibiliza dois serviços para acesso a informações de Pessoa Jurídica. O primeiro apresenta todos os CNPJs cadastrados para um determinado usuário. O segundo, utiliza desse CNPJ para extrair informações cadastradas no Brasil Cidadão para aquela pessoa e empresa.

Para acessar o serviço que disponibiliza os CNPJs vinculados a um determinado usuário, é necessário o seguinte:

1. Com o usuário autenticado, a aplicação deverá realizar uma requisição por meio do método GET a URL https://(url de serviços do brasil cidadão)/empresas/escopo/<escopo> enviando as seguintes informações:
    - No *Header*:
        - Authorization : Bearer <ACCESS_TOKEN> ;
    - Parâmetro escopo: o escopo de empresa, por exemplo, *brasil_cidadao_empresa*

2. O resultado em caso de sucesso será retornado em formato *JSON* conforme o exemplo abaixo:

```
{
        "cnpjs": ["CNPJ1", “CNPJ2”],
        "cpf": "<CPF DO USUARIO>"
}
```

Para acessar o serviço que acessa as informações do cadastro de empresas é necessário o seguinte:

1. Com o usuário autenticado, a aplicação cliente deverá acessar, por meio do método *POST*, a *URL*: https://(url de serviços do brasil cidadão)/empresa/<cnpj>/escopo/<escopo> enviando as seguintes informações:
    - No header:
        - Authorization: Bearer <ACCESS_TOKEN>
    - Parâmetro cnpj: o CNPJ da empresa formatado (sem ponto, barra etc).
    - Parâmetro escopo: o escopo de empresa, por exemplo, *brasil_cidadao_empresa*

2. A resposta, em caso de sucesso, retorna um objeto *JSON* de acordo com o escopo informado. Abaixo segue o formato:

```
{
        "cnpj":"<CNPJ>",
        "nomeFantasia":"<NOME FANTASIA>",
        "atuacao":"<ATUACAO>",
        "cpfResponsavel":"<CPF DO RESPONSÁVEL>",
        "nomeResponsavel":"<NOME DO RESPONSÁVEL>"
}
```

Resultados Esperados do Acesso ao Serviço de Cadastro de Pessoas Jurídicas

Os acessos aos serviços do Brasil Cidadão ocorrem por meio de chamadas de *URLs* e a resposta são códigos presentes conforme padrão do protocolo http. Estes códigos são:

- **Código 200:** Dados acessados e retornados em formato *JSON* ao usuário, de acordo com cada escopo;
- **Código 400:** *Token* recebido por mais de um método;
- **Código 401:** *Token* não encontrado ou inválido, CNPJ inválido, usuário não existente no sistema, *access token* inválido;
- **Código 403:** Escopo solicitado não autorizado pelo usuário;
- **Código 404:** Escopo obrigatório.

Acesso ao Serviço de Confiabilidade Cadastral (Selos)

Para acessar o serviço de consulta de empresas é necessário:

1. Com usuário autenticado, deverá acessar, por meio do método *POST*, a *URL*: https://(url de serviços do brasil cidadão)/servicos-ecidadao/ecidadao/usuario/confiabilidade/<cpf> ;
2. No *Header* HTTP, deverá ser passado o atributo *Authorization* com a informação codificada em *Base64* e tendo o seguinte formato: *CLIENT_ID*:*CLIENT_SECRET*. A palavra *Basic* deve está antes da informação codificada;
3. A resposta em caso de sucesso retorna sempre um *array* de objetos *JSON* no seguinte formato: {id, nivel, descricao};

Resultados Esperados do Acesso ao Serviço de Confiabilidade Cadastral (Selos)

Os selos existentes no Brasil Cidadão são:

```
{
        "id": 0,
        "nivel": 2,
        "descricao": "Institucional" (Servidor Público)
},
```

```
        {
                "id": 0,
                "nivel": 1,
                "descricao": "Conformidade"
        },
        {
                "id": 0,
                "nivel": 4,
                "descricao": "Biometria"
        },
        {
                "id": 0,
                "nivel": 5,
                "descricao": "Certificado Digital"
        },
        {
                "id": 0,
                "nivel": 3,
                "descricao": "Convalidação" (Módulo Balcão)
        }
        {
                "id": 0,
                "nivel": 10,
                "descricao": "DNI"
        },
        {
                "id": 0,
                "nivel": 11,
                "descricao": "REPRESENTANTE E-CNPJ"
        }
```

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://(url do provider brasil cidadão)/scp/authorize?response_type=code"
. "&client_id=". CLIENT_ID
. "&scope=openid+brasil_cidadao"
. "&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://(url do provider brasil cidadão)/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 de usuário:

```
$escopo = "brasil_cidadao";
$url = "https://(url de serviços do brasil cidadão)/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);
```

4. De posse do token, requisição para obter dados de um escopo de empresa:

```
$escopo = "brasil_cidadao_empresa";
$cnpj = CNPJ da Empresa
$url = "https://(url de serviços do brasil cidadão)/empresa/” . $cnpj . ”/escopo/" . $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 DO BRASIL CIDADÃO [BRASIL_CIDADAO]
<br>Nome: <%=(String)request.getAttribute("nome")%>
<br>CPF: <%=(String)request.getAttribute("cpf")%>
<br>Foto: <%=(String)request.getAttribute("foto")%>
<br>Email: <%=(String)request.getAttribute("email")%>
<br>Telefone: <%=(String)request.getAttribute("telefone")%>
```

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+"/{brasil_cidadao}?
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("cpf",jsonObj.getString("cpf"));
model.addAttribute("foto",jsonObj.getString("foto"));
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
- ***Plataforma de Cidadania Digital:*** http://www.planejamento.gov.br/cidadaniadigital