# Integracja własnej aplikacji webowej z SAS Viya

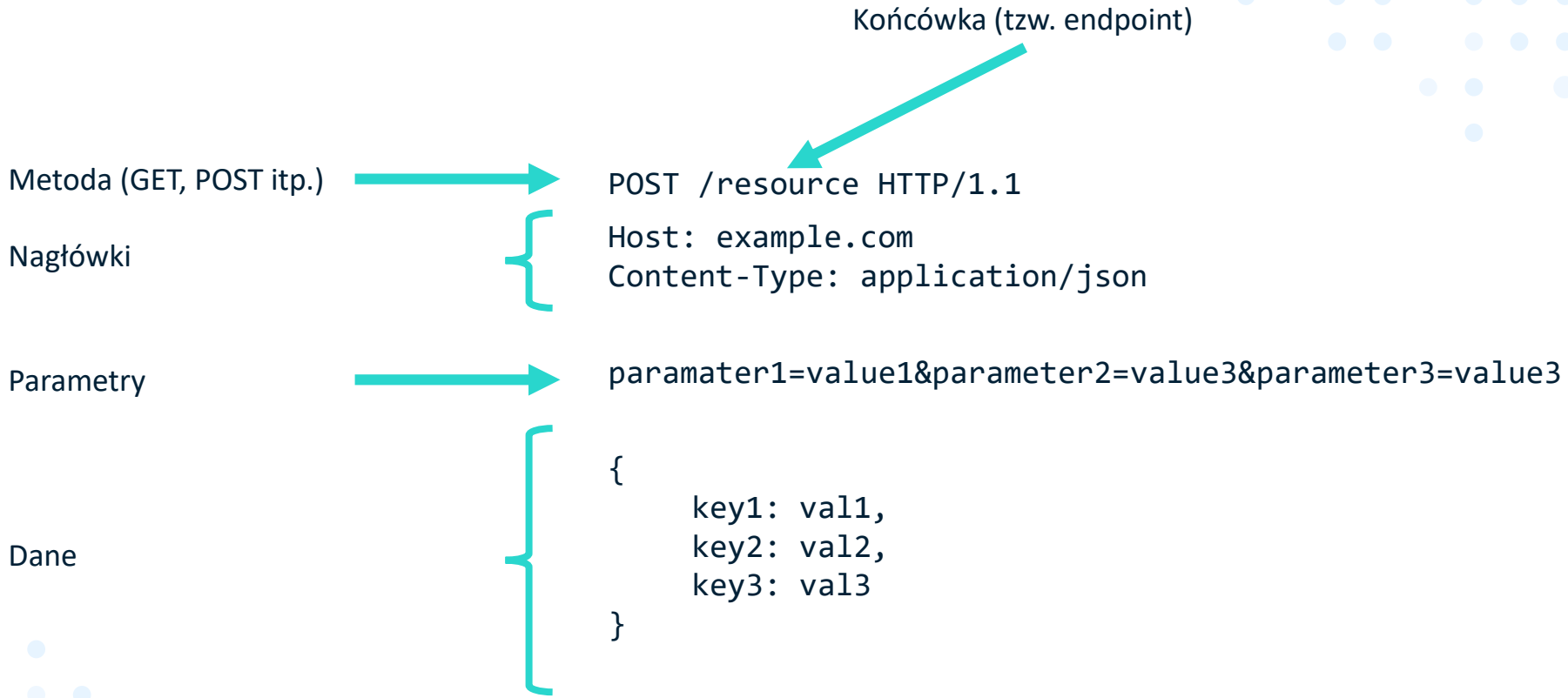
Piotr Florczyk

SCS Expert

# O czym będę mówić?

1. Krótko o HTTP i użytej notacji
2. REST co to jest?
3. OAuth 2.0 – autoryzacja, nie autentykacja
4. Rejestracja aplikacji klienckiej wg dokumentacji
5. Przykłady integracji w praktyce

# Krótko o HTTP i użytej notacji



# Przykład

https://www.example.com/?user=foo&pass=bar

```
GET / HTTP/1.1
Host: example.com

user=foo&pass=bar
```

# REST co to jest?

# REST

REST to skrót od REpresentational State Transfer

czyli zmiana stanu poprzez reprezentację

(bezsensowny opis najczęściej używanej technologii w serwisach internetowych)

# REST prostymi słowami

- sposób komunikacji dwóch systemów komputerowych
- za pomocą protokołu HTTP
- podobny sposób jak przeglądarki internetowe i serwery WWW

# Przykłady

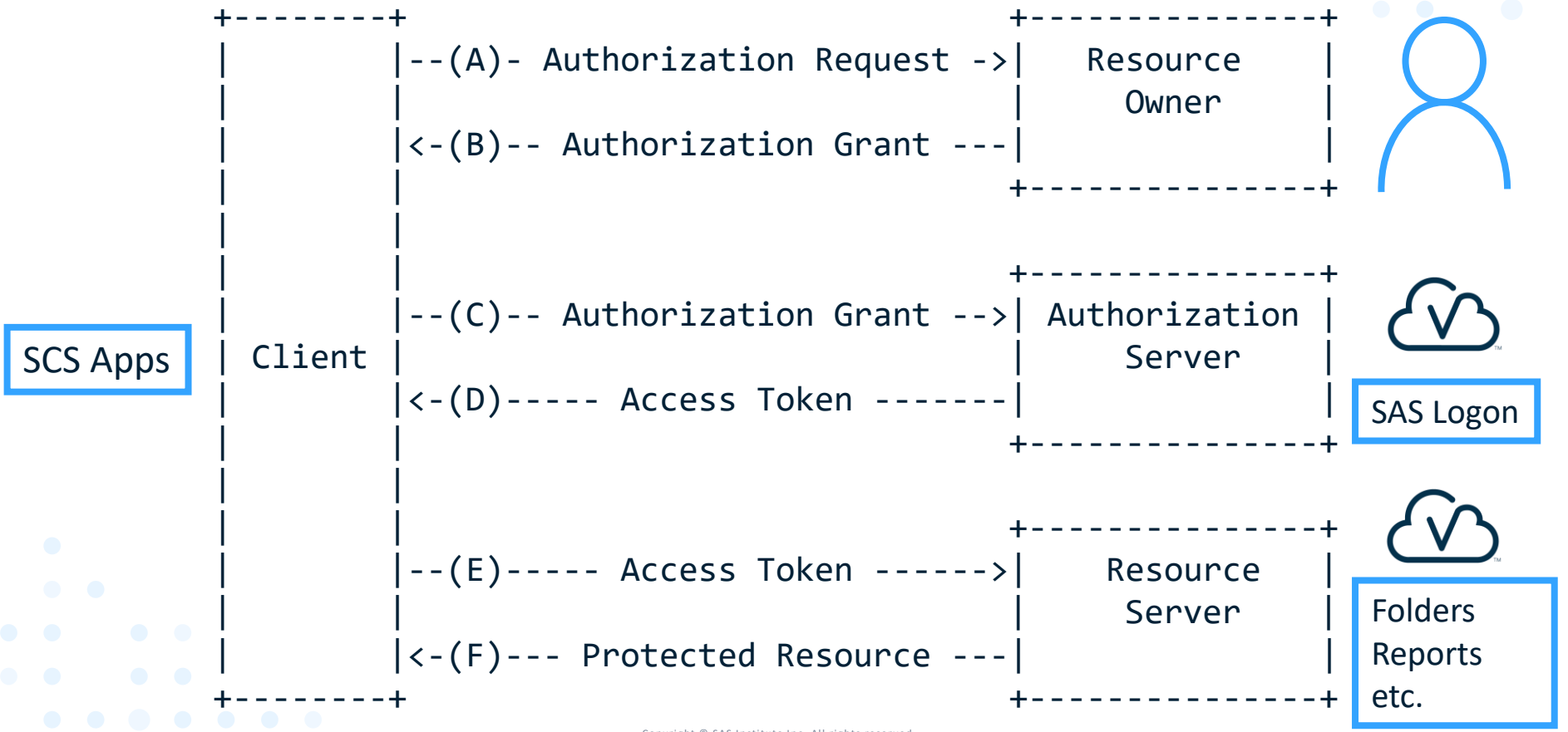
- żądanie `GET` do `/user/` zwróci listę zarejestrowanych użytkowników systemu
- żądanie `POST` do `/user/123` utworzy użytkownika z ID=123 z informacjami zawartymi w przesłanych z żądanie danymi
- żądanie `PUT` do `/user/123` zaktualizuje użytkownika o ID=123 za pomocą danych żądanie w zapytaniu
- żądanie `GET` do `/user/123` zwróci szczegółowe informacje o użytkowniku o ID=123
- żądanie `DELETE` do `/user/123` usunie użytkownika z ID=123 z systemu

# OAuth 2.0 – autoryzacja, nie autentykacja

# OAuth 2.0

- Framework autoryzacyjny
- Pozwala aplikacji typu third-party na uzyskanie ograniczonego dostępu do zasobów usługi HTTP
- w imieniu aplikacji bądź właściciela określonego zasobu
- zastępuje OAuth 1.0

# Abstrakcyjny proces autoryzacji



# OAuth 2.0 Grant Types

- Resource Owner Password Credentials Grant
- Client Credentials Grant
- Authorization Code Grant
- Refresh Token Grant
- Implicit Grant

# Rejestracja aplikacji klienckiej wg dokumentacji

SAS Viya REST API

# Registering clients

https://developer.sas.com/reference/auth/

All applications and scripts that make use of the SAS Viya REST APIs must be registered with the SAS environment. In practice, this registration is handled by the SAS administrator properly configuring a client with the OAuth service in SAS Logon Manager.

To register a client:

Locate a valid Consul token. (SAS Logon Manager has an endpoint that issues an OAuth access token to requests that contain a valid token from the SAS Configuration Server.) A SAS administrator can find a token in the **`client.token`** file at `/opt/sas/viya/config/etc/SASSecurityCertificateFramework/tokens/consul/default`. On a Linux system, run

1

```
$ cat client.token
```

to get the token string. Copy the value for later use.

Request a valid OAuth token to use on the registration call. For example, to register a client named `app`:

2

```
curl -X POST "http://example.com/SASLogon/oauth/clients/consul?callback=false&serviceId=app" \
  -H "X-Consul-Token: <value-of-consul-token-goes-here>"
```

| Query parameter | Description |
|---|---|
| callback | Set to false to receive an access token in the response, otherwise the token is sent to the service registered in SAS Configuration Server. |
| serviceId | Name of the client to register |

Use the value of the **`access_token`** field returned in the response JSON to register the new client:

3

```
curl -X POST "https://localhost/SASLogon/oauth/clients" \
   -H "Content-Type: application/json" \
   -H "Authorization: Bearer <access-token-goes-here>" \
   -d '{
     "client_id": "app",
     "client_secret": "<secret-goes-here>",
     "scope": ["openid"],
     "authorized_grant_types": ["password"],
     "access_token_validity": 43199
    }'
```

# 3 kroki do rejestracji aplikacji

1. Weź zawartość pliku consult.token

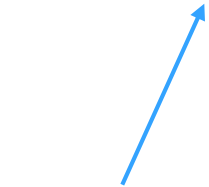
2. Używając consul.token poproś o access_token

3. Używając access_token zarejestruj aplikację

# Rejestracja aplikacji - krok 1 i 2
# Autentykacja w REST API

`/opt/sas/viya/config/etc/SASSecurityCertificateFramework/tokens/consul/default/client.token`

```
POST /oauth/clients/consul HTTP/1.1
Host: localhost
X-Consul-Token: <consul-token>

callback=false
&serviceId=app
```



cokolwiek

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "access_token": "eyJhbG...FuwoDPA",
    "token_type": "bearer",
    "expires_in": 35999,
    "scope": "uaa.admin",
    "jti": "1f...25"
}
```

# Rejestracja aplikacji - krok 3
# Zdefiniowanie aplikacji

```
POST /oauth/clients HTTP/1.1
Host: localhost
Content-Type: application/json
Authorization: Bearer <access-token-goes-here>

{
    "client_id": "scs.example0",
    "client_secret": "ABCDE",
    "scope": ["openid"],
    "authorized_grant_types": ["password"]
}
```

```
HTTP/1.1 201 Created
Content-Type: application/json

{
    "scope": [
        "openid"
    ],
    "client_id": "scs.example0",
    "resource_ids": [
        "none"
    ],
    "authorized_grant_types": [
        "password",
        "refresh_token"
    ],
    "autoapprove": [],
    "authorities": [
        "uaa.none"
    ],
    "lastModified": 1599561136608,
    "required_user_groups": []
}
```

# Demonstracja

## Rejestracja aplikacji w SAS Viya

# Autoryzacja aplikacji i wykonywanie zapytań do zasobów

SAS Viya REST API

# Obtaining access tokens

Registered clients can request an access token using the SAS Logon OAuth API. Access tokens are obtained when a client makes a request and authenticates to the `/SASLogon/oauth/token` endpoint and passes a form of authorization. The authorization is expressed in the form of an authorization grant. Currently, SAS Viya REST APIs support the `password` grant type.

For example, given a client identifier of "app" with a secret "mysecret," a client must pass user credentials to authenticate and receive a token:

1
```
curl -X POST "https://server.example.com/SASLogon/oauth/token" \
   -H "Content-Type: application/x-www-form-urlencoded" \
   -d "grant_type=password&username=<user-id>&password=<password>" \
   -u "app:mysecret"
```

The JSON response contains a field named `access_token` that contains the value of the token that is used by the client for subsequent API requests. When the token expires, the same call to `/SASLogon/oauth/token` is repeated.

# Making API requests with access tokens

Once a client has a valid access token to use on behalf of a user, requests to SAS Viya REST APIs should use that token. The token can be passed to the API either as a query string parameter or on the HTTP Authorization header (which is the preferred approach). For example:

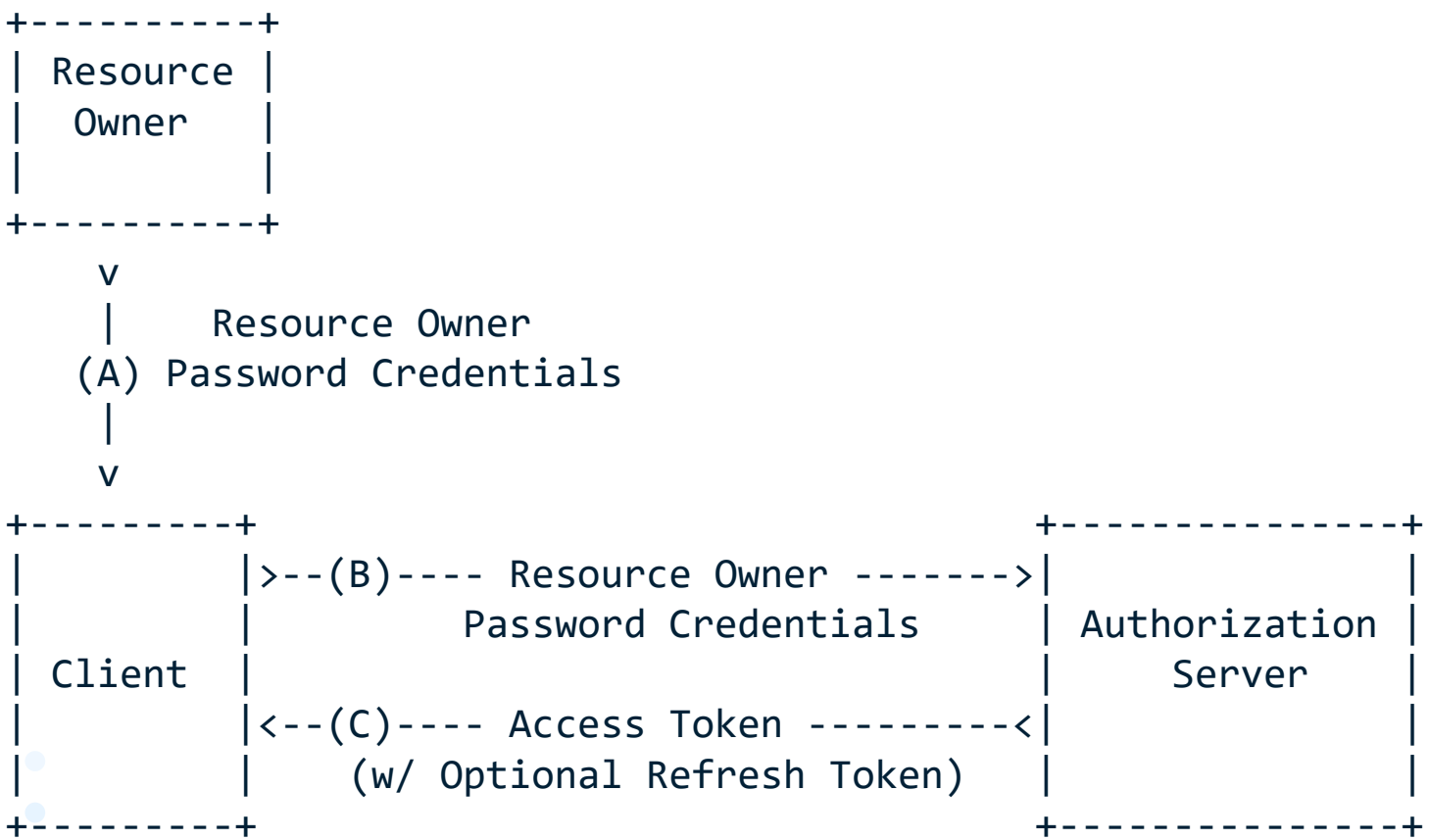
2
```
curl -X GET "https://server.example.com/folders/folders/@myFolder" \
   -H "Accept: application/json" \
   -H "Authorization: Bearer <TOKEN-STRING>"
```

# 2 kroki

1. Przy pomocy client_id, client_scret, username, password uzyskaj access_token

2. Używając access_token wykonuj zapytania do zasobów

# Resource Owner Password Credentials Grant



# Autoryzacja aplikacji - krok 1
# Uzyskaj access_token

```
POST /SASLogon/oauth/token HTTP/1.1
Host: viya-server
Content-Type: application/x-www-form-urlencoded
Authorization: Basic Base64(client_id:client_secret)

grant_type=password
&username=some-username
&password=some-password
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "access_token": "eyJhbG...FuwoDPA",
    "token_type": "bearer",
    "refresh_token": "dsfHJ...DasDJK",
    "scope": "open.id",
    "jti": "1f...25"
}
```

# Autoryzacja aplikacji - krok 2
# Wykonaj zapytanie do zasobu

```
GET /folders/folders/@myFolder HTTP/1.1
Host: viya-server
Accept: application/json
Authorization: Bearer eyJhbG...FuwoDPA
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    ...
}
```

# Demonstracja

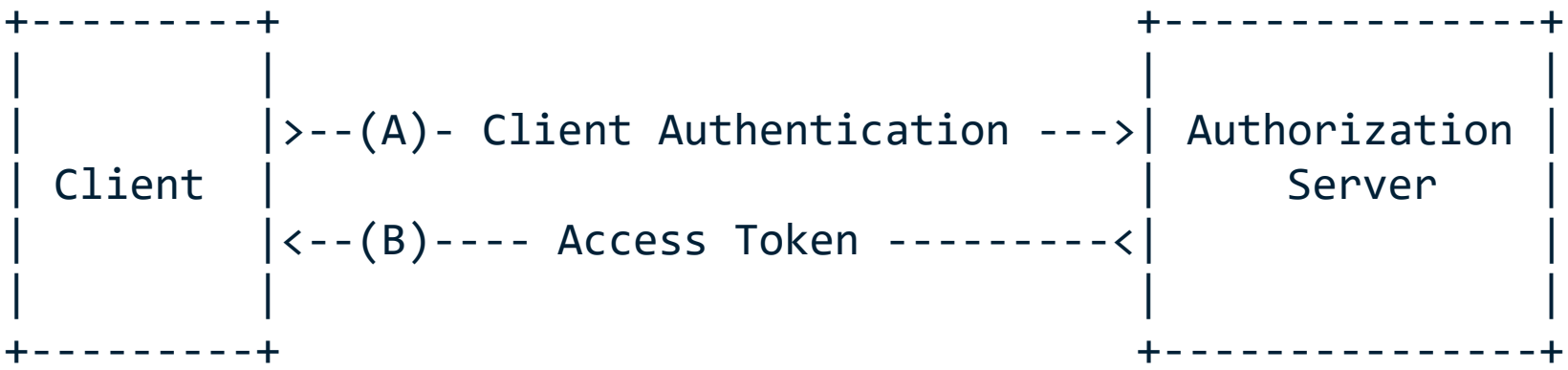
Autoryzacja zarejestrowanej aplikacji i odpytywanie zasobów poprzez REST API

# Demonstracja

## Integracja aplikacji webowej

## Resource Owner Password Credentials Grant

# Client Credentials Grant



# Client Credentials Grant
# Zdefiniowanie aplikacji

```
POST /oauth/clients HTTP/1.1
Host: localhost
Content-Type: application/json
Authorization: Bearer <access-token-goes-here>

{
     "client_id": "scs.example1",
     "client_secret": "ABCDE",
     "scope": ["openid"],
     "authorized_grant_types": ["client_credentials"]
}
```

```
HTTP/1.1 201 Created
Content-Type: application/json

{
     "scope": [
          "openid"
     ],
     "client_id": "scs.example1",
     "resource_ids": [
          "none"
     ],
     "authorized_grant_types": [
          "client_credentials",
          "refresh_token",
     ],
     "autoapprove": [],
     "authorities": [
          "uaa.none"
     ],
     "lastModified": 1602075807553,
     "required_user_groups": []
}
```

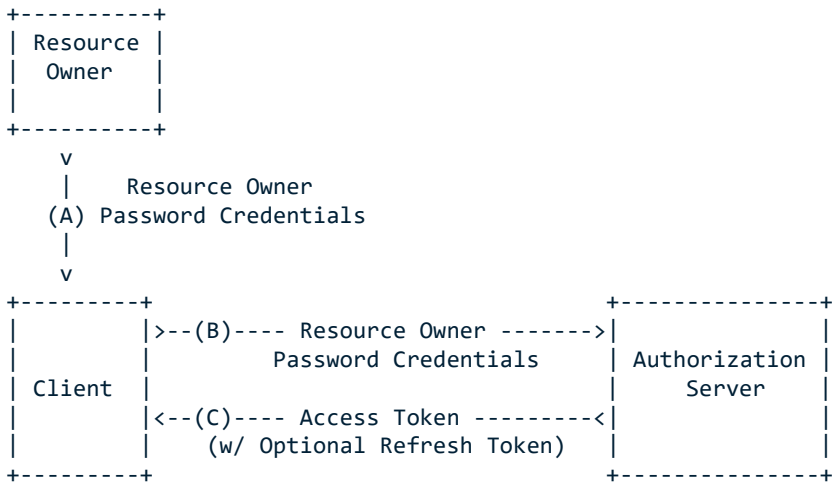
# Client Credentials Grant
# Uzyskanie access_token

```
POST /SASLogon/oauth/token HTTP/1.1
Host: viya-server
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials
&client_id=scs.example1
&client_secret=ABCDE
```

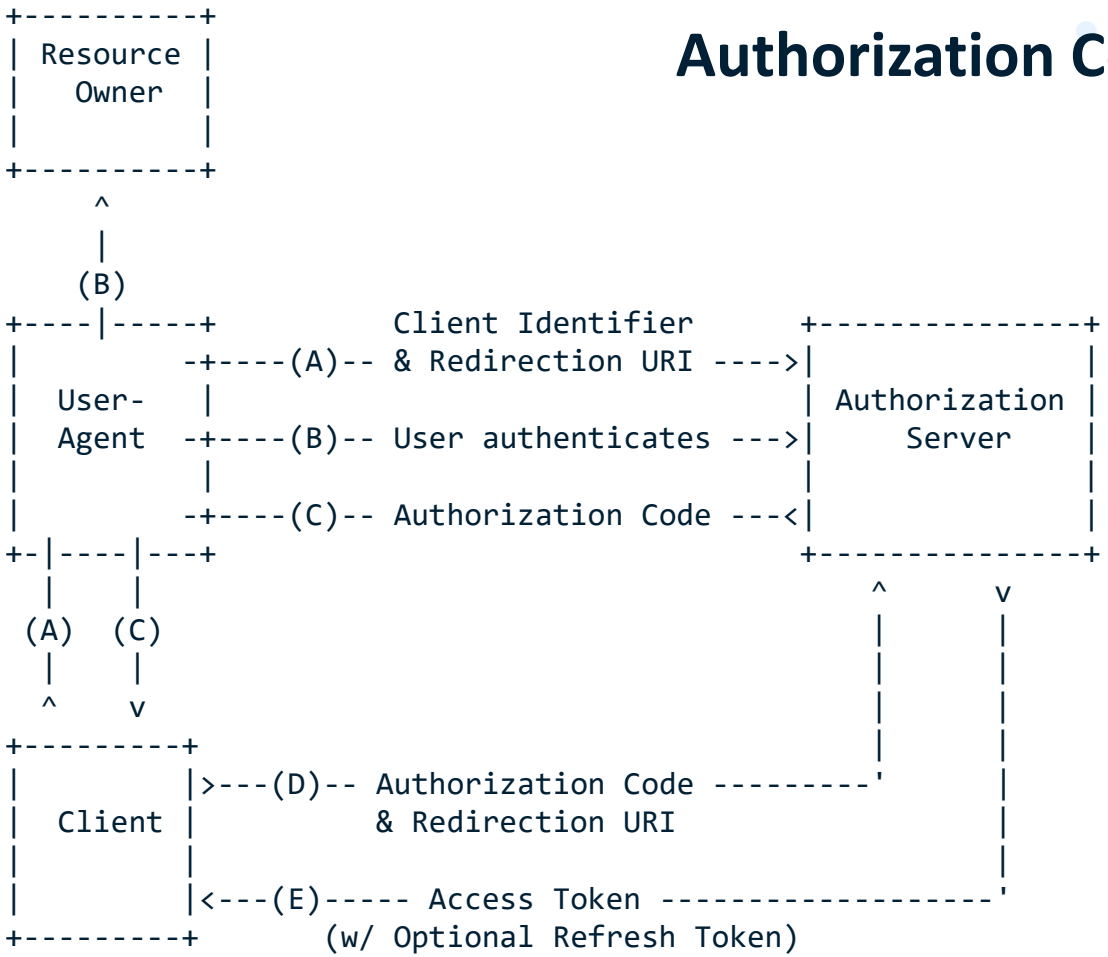
```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "access_token": "eyJhbG...FuwoDPA",
    "token_type": "bearer",
    "refresh_token": "dsfHJ...DasDJK",
    "scope": "open.id",
    "jti": "1f...25"
}
```

# Wady Resource Owner Password Credentials Grant

1. Aplikacja kliencka musi zajmować się loginem i hasłem użytkownika
2. Brak Single Sign On
   - Trzeba zaimplementować dodatkowy ekran logowania
   - Użytkownik musi logować się odrębnie do SAS Viya i do naszej aplikacji



# Authorization Code Grant



# Authorization Code Grant
# Rejestracja aplikacji

```
POST /SASLogon/oauth/token HTTP/1.1
Host: viya-server
Content-Type: application/x-www-form-urlencoded

{
    "client_id": "scs.example2",
    "client_secret": "ABCDE",
    "scope": ["openid"],
    "authorized_grant_types": ["authorization_code"],
    "redirect_uri": [
        "http://localhost:5000/authorize",
        "http://127.0.0.1:5000/authorize"
    ]
}
```

```
HTTP/1.1 201 Created
Content-Type: application/json

{
    "scope": [
        "openid"
    ],
    "client_id": "scs.example2",
    "resource_ids": [
        "none"
    ],
    "authorized_grant_types": [
        "authorization_code",
        "refresh_token"
    ],
    "redirect_uri": [
        "http://localhost:5000/authorize",
        "http://127.0.0.1:5000/authorize"
    ],
    "autoapprove": [],
    "authorities": [
        "uaa.none"
    ],
    "lastModified": 1602075807553,
    "required_user_groups": []
}
```

# Demonstracja

## Integracja aplikacji webowej

## Authorization Code Grant

# Na zakończenie

# Dziękuję

Piotr.Florczyk@scsexpert.com

sas.com