**SPRING SECURITY USING OAUTH2.0**

Table of Contents:

1. Spring Security Introduction
2. OAuth 2.0
   1. Introduction
   2. Why we use OAuth 2
   3. OAuth2 Roles
   4. OAuth2 GrantTypes
      * 1. Authorization Code
        2. Implicit Grant
        3. Resource Owner Password Credentials Grant
        4. Client credentials Grant
   5. OAuth2 Tokens

3. Spring Security using OAuth2

4. Example Application

5. References

1. Spring Security Introduction:

Spring security is the highly customizable authentication and access-control framework. This is the security module for securing spring applications. But, this can also be used for non-spring based application with few extra configurations to enable the security features.

The main focus of spring security is on Authentication and Authorization:

* Authentication: Process of checking the user, who they claim to be.
* Authorization: Process of deciding whether a user is allowed to perform an activity within the application.

## **Artifacts Download**

## The first step towards writing your spring security application is to get the required artifacts for spring security. Here is the dependency requirements for [Maven](https://javabeat.net/creating-simple-java-project-using-apache-maven/) and [Gradle](https://javabeat.net/gradle-tutorial/)build scripts.

**Pom.xml for Maven**

|  |
| --- |
| < dependencies >  <! -- ... other dependency elements ... -->  < dependency >  <groupId>org.springframework.security</groupId>  <artifactId>spring-security-web</artifactId>  <version>4.0.2.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework.security</groupId>  <artifactId>spring-security-config</artifactId>  <version>4.0.2.RELEASE</version>  </dependency>  </dependencies> |

**build.gradle for Gradle**

dependencies {

compile 'org.springframework.security:spring-security-web:4.0.2.RELEASE'

compile 'org.springframework.security:spring-security-config:4.0.2.RELEASE'

}

* If you are using the third party authentication models, then you have to include those dependencies in your build file.

2. OAuth 2.0

2.1 Introduction:

OAuth is an open authorization protocol, which allows accessing the resources of the resource owner by enabling the client applications on HTTP services such as Facebook, GitHub, etc. It allows sharing of resources stored on one site to another site without using their credentials. It uses username and password tokens instead.

* 1. Why we use OAuth 2
* You can use OAuth 2.0 to read data of a user from another application.
* It supplies the authorization workflow for web, desktop applications, and mobile devices.
* It is a server side web app that uses authorization code and does not interact with user credentials.

**Features of OAuth 2.0**

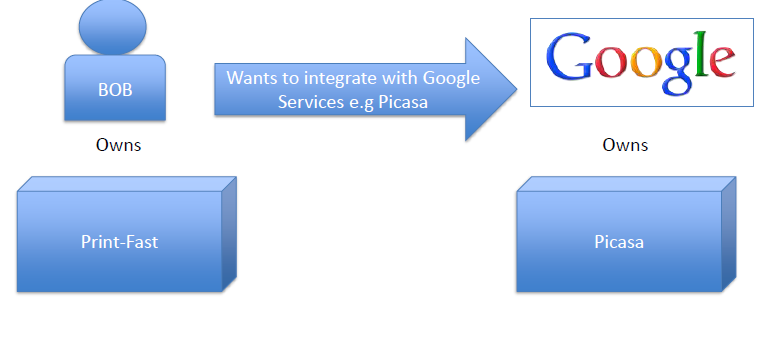
* OAuth 2.0 is a simple protocol that allows to access resources of the user without sharing passwords.
* It provides user agent flows for running clients application using a scripting language, such as JavaScript. Typically, a browser is a user agent.
* It accesses the data using tokens instead of using their credentials and stores data in online file system of the user such as Google Docs or Dropbox account.
  1. OAuth2 Roles

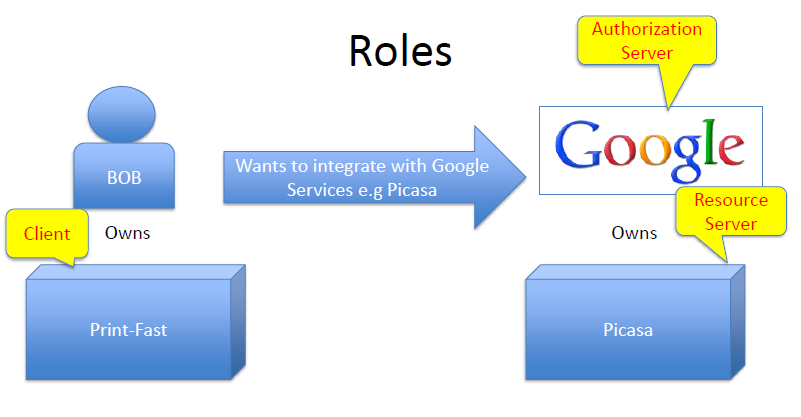
OAuth defines four roles:

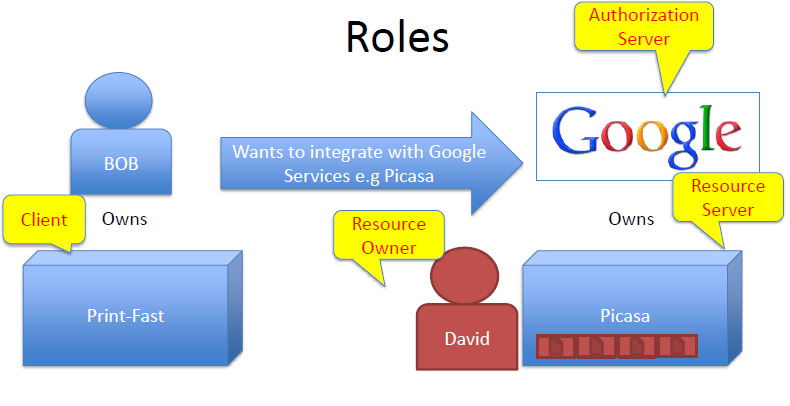
* Resource owner (the User) – an entity capable of granting access to a protected resource (for example end-user).
* Resource server (the API server) – the server hosting the protected resources, capable of accepting responding to protected resource requests using access tokens.
* Client – an application making protected resource requests on behalf of the resource owner and with its authorization.
* Authorization server – the server issuing access tokens to the client after successfully authenticating the resource owner and obtaining authorization.

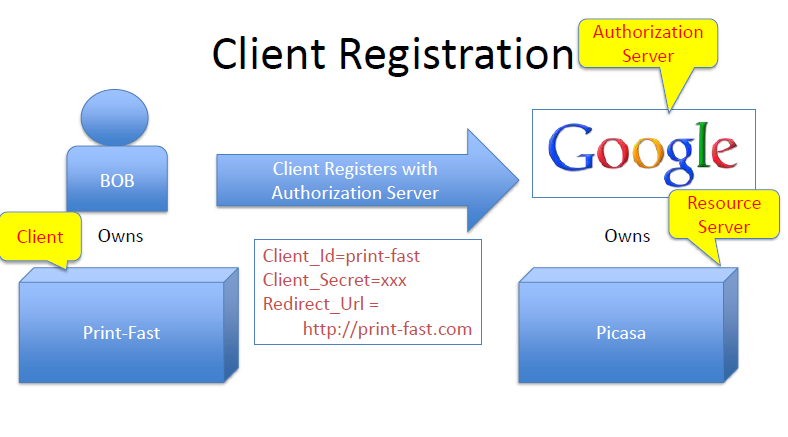
**OAuth 2.0 Flow in Depth**

Scenario







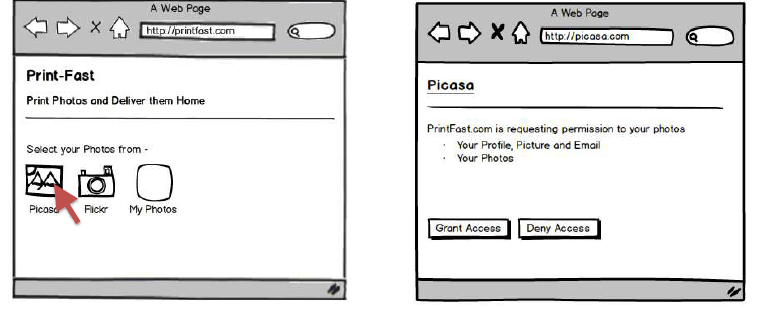


* 1. OAuth2 GrantTypes

OAuth 2 provides several "grant types" for different use cases. The grant types defined are:

* + - 1. Authorization Code
      2. Implicit
      3. Password
      4. Client Credentials
      5. Authorization Code

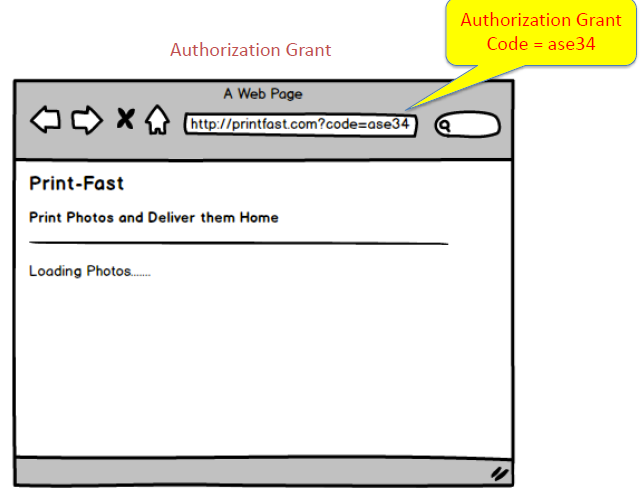
**Step 1: Authorization Code Grant**

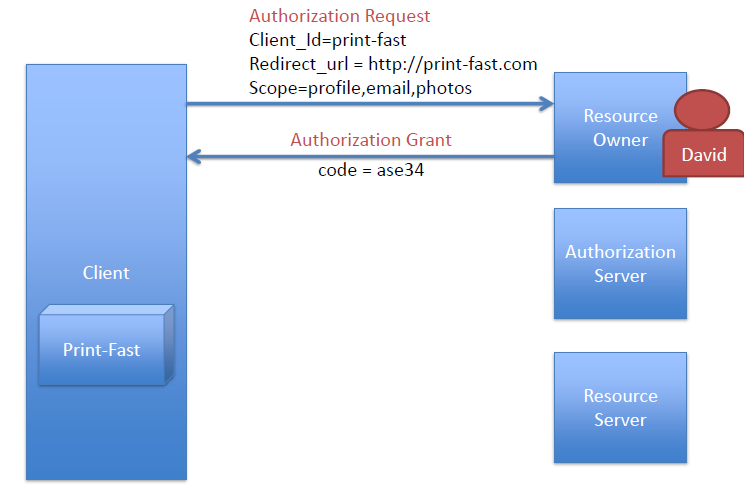
Authorization Request Authorization Grant

URL used is

http://picasa.com/?client\_id=photo-fast &scope=profile,email,photos

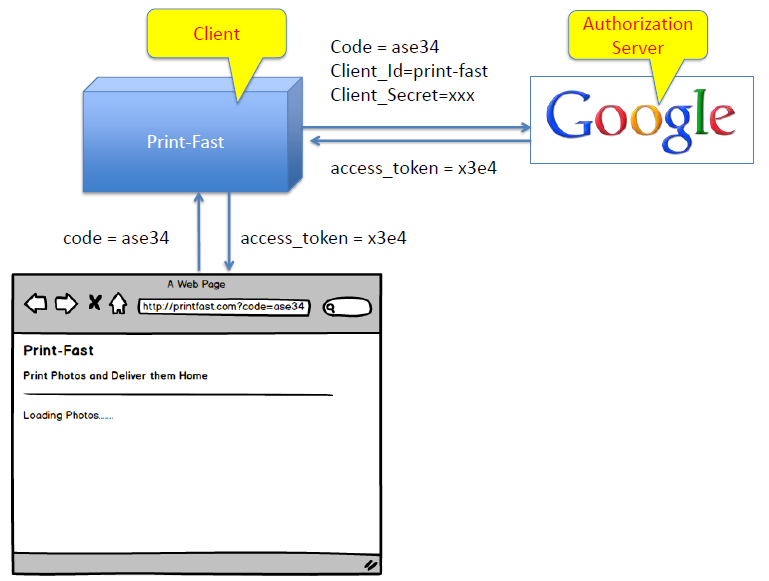
&redirect\_uri=http://print-fast.com&response\_type=code

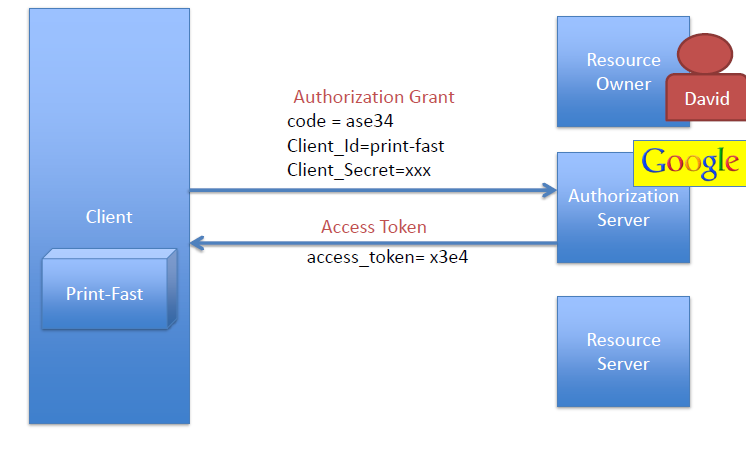




Protocol Flow

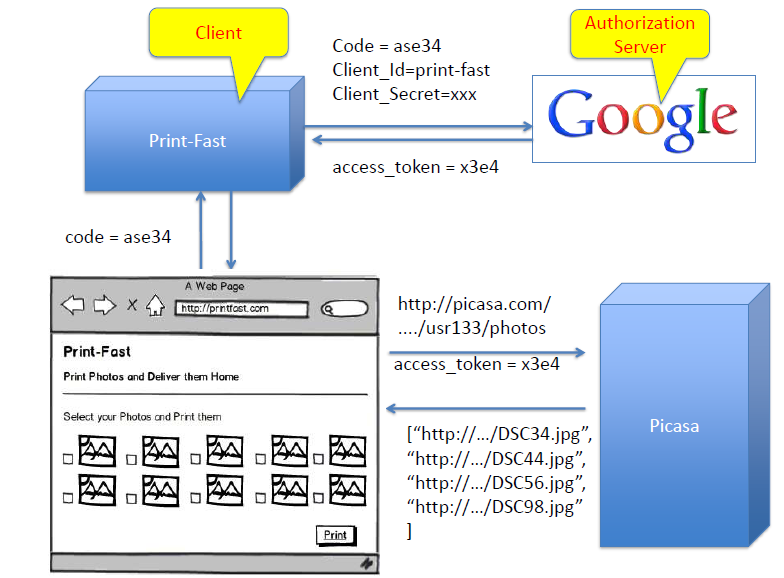
**Step 2 – Exchange for Access Token**

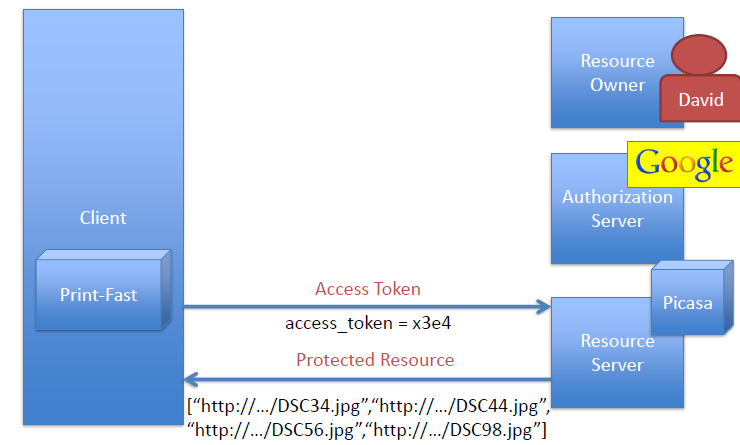




Protocol Flow

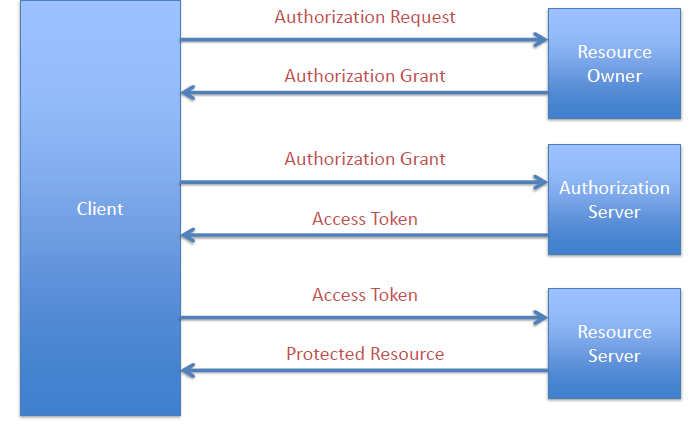
**step 3 – Access Protected Resources**



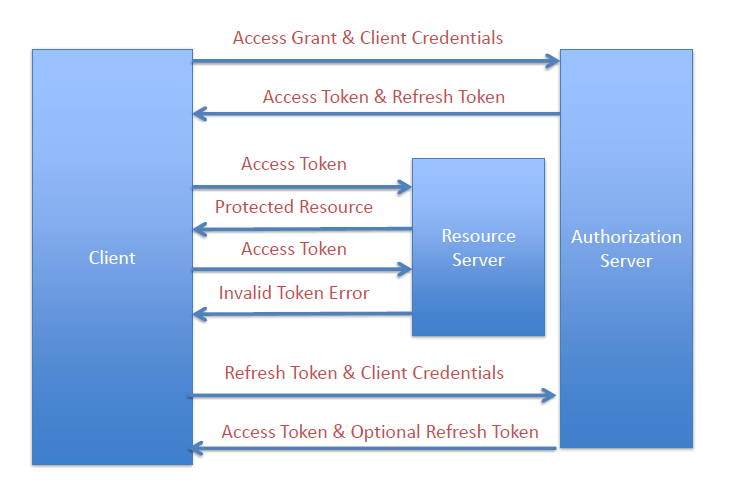


Protocol Flow

**Complete Flow at Once**



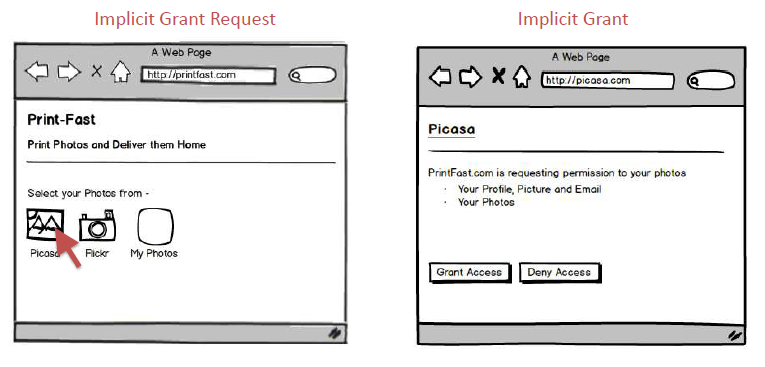
**With Refresh Token**



Protocol Flow

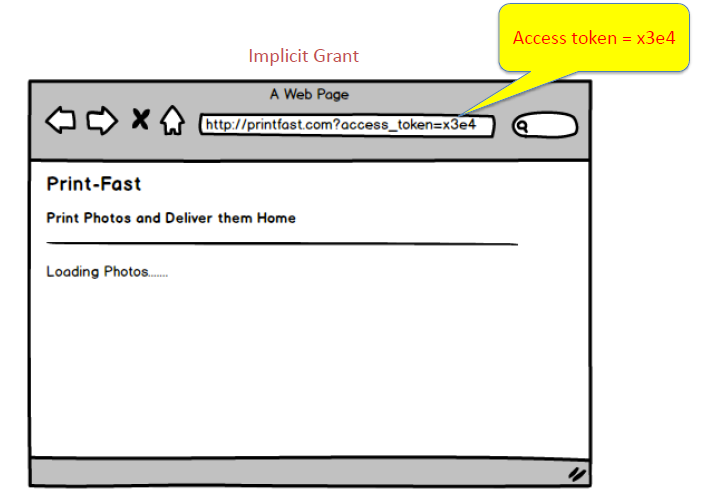
* + - 1. Implicit Grant

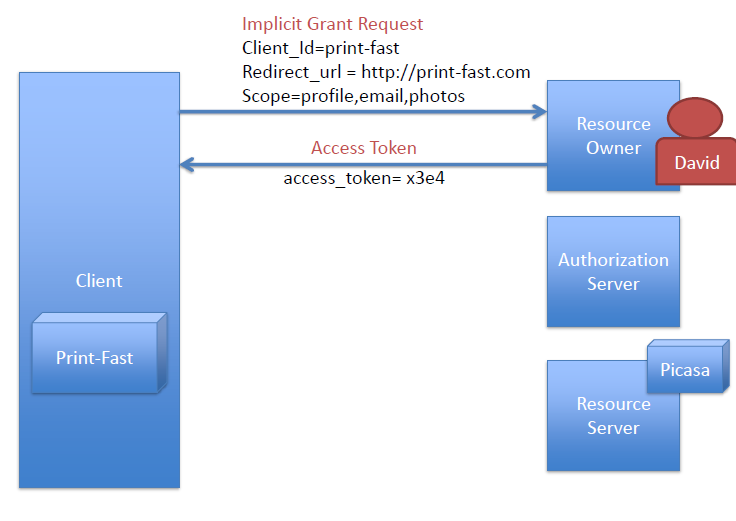
**Step 1 – Get Access Token**



URL used is

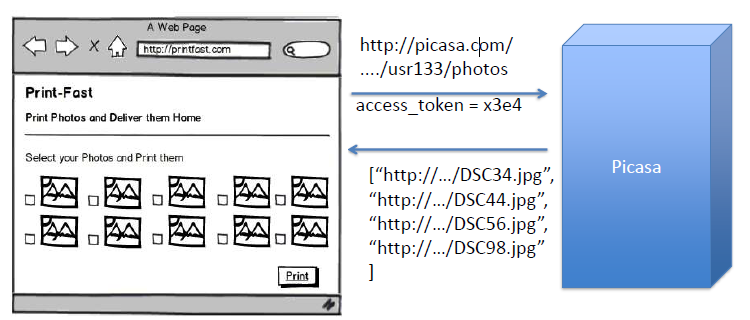
http://picasa.com/?client\_id=photo-fast &scope=profile,email,photos&redirect\_uri=http://print-fast.com&response\_type=token

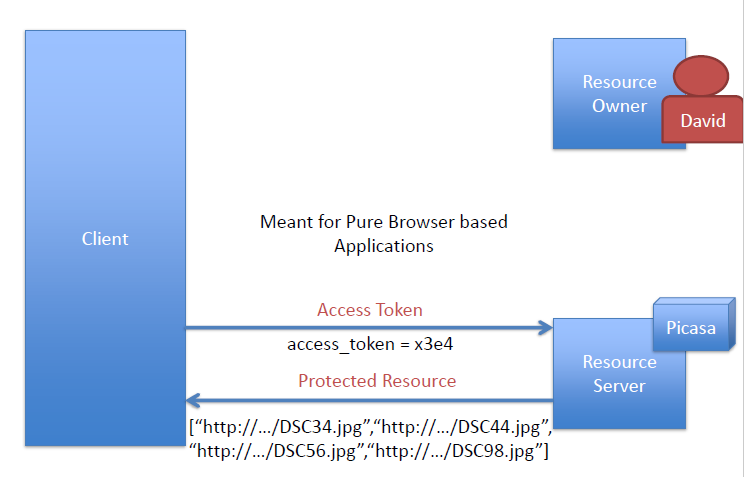




Protocol Flow

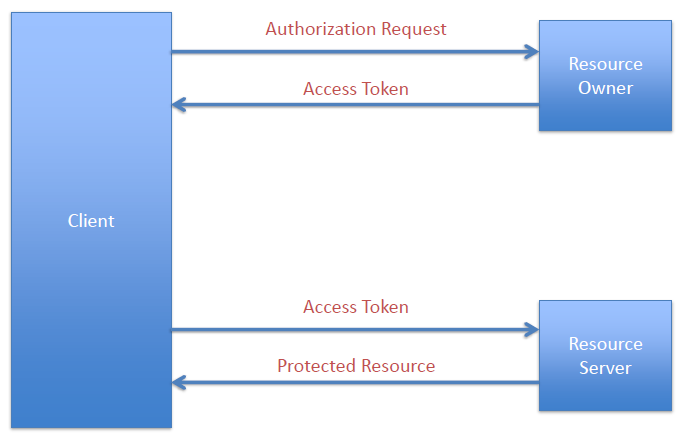
**Step 2 – Access Protected Resources**





Protocol Flow

**Complete Flow at Once**



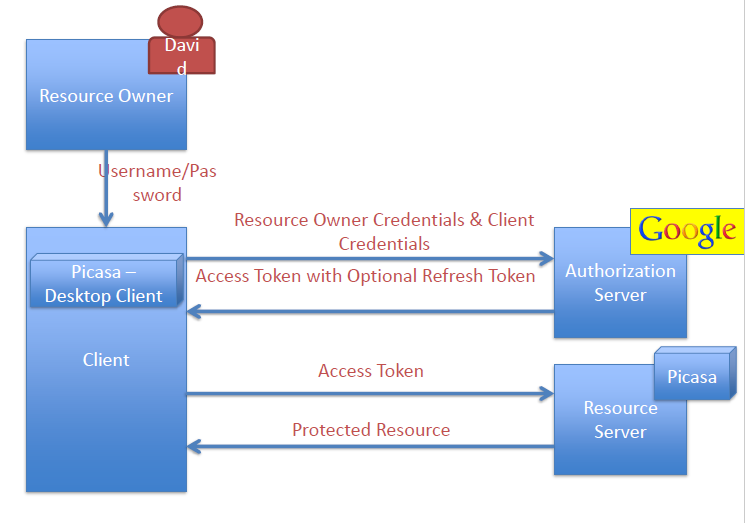
**2.4.1.3** Resource Owner Password Credentials Grant

Protocol Flow

* + - 1. Client Credentials Grant



Protocol Flow



* 1. OAuth2 Tokens

Tokens are implementation specific random strings, generated by the authorization server and are issued when the client requests them.

* Access Token:  Sent with each request, usually valid for a very short life time [an hour e.g.]
* Refresh Token: Mainly used to get a new access token, not sent with each request, usually lives longer than access token.