**Identity**

**&**

**Access**

**Management**

Contents

[1 General Introduction 4](#_Toc55737471)

[2 Sources/Credits/Acknowledgements 4](#_Toc55737472)

[3 Important Terms 4](#_Toc55737473)

[4 IAM – From the beginning till date 4](#_Toc55737474)

[4.1 Phase 1 5](#_Toc55737475)

[4.2 Phase 2 6](#_Toc55737476)

[4.3 Phase 3 7](#_Toc55737477)

[4.4 Phase 4 - Federated Identity -  Supporting application access across company boundaries 8](#_Toc55737478)

[4.5 Phase 5 - The birth and establishment of Federated Identity Protocols 9](#_Toc55737479)

[5 The IAM landscape today 10](#_Toc55737480)

[5.1 Federated Identity 10](#_Toc55737481)

[5.2 SAML2 10](#_Toc55737482)

[5.2.1 SP initiated flow 10](#_Toc55737483)

[5.2.2 IDP initiated flow 11](#_Toc55737484)

[5.3 OAuth2 11](#_Toc55737485)

[5.3.1 Overview 11](#_Toc55737486)

[5.3.2 Scopes 11](#_Toc55737487)

[5.3.3 OAuth2 Authorization Flows 12](#_Toc55737488)

[5.3.4 OAuth2 Roles 14](#_Toc55737489)

[5.3.5 OAuth2.0 End Points 15](#_Toc55737490)

[5.3.6 One time client registration on the Authorization server 15](#_Toc55737491)

[5.4 Open ID 1.x/2.x 16](#_Toc55737492)

[5.5 OpenID Connect 16](#_Toc55737493)

[5.6 JWT 16](#_Toc55737494)

[5.7 SSO 16](#_Toc55737495)

[5.8 Multi factor authentication 16](#_Toc55737496)

[5.9 Universal Login 16](#_Toc55737497)

[5.10 Universal Directory 16](#_Toc55737498)

[6 Leading vendors 16](#_Toc55737499)

[6.1 Auth0 16](#_Toc55737500)

[6.2 Okta 16](#_Toc55737501)

# General Introduction

# Sources/Credits/Acknowledgements

# Important Terms

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# IAM – From the beginning till date

Authentication defines the way a user is identified and validated through some sort of credentials as part of a sign-in flow ( Source - <https://developer.okta.com/docs/concepts/saml/> )

## Phase 1

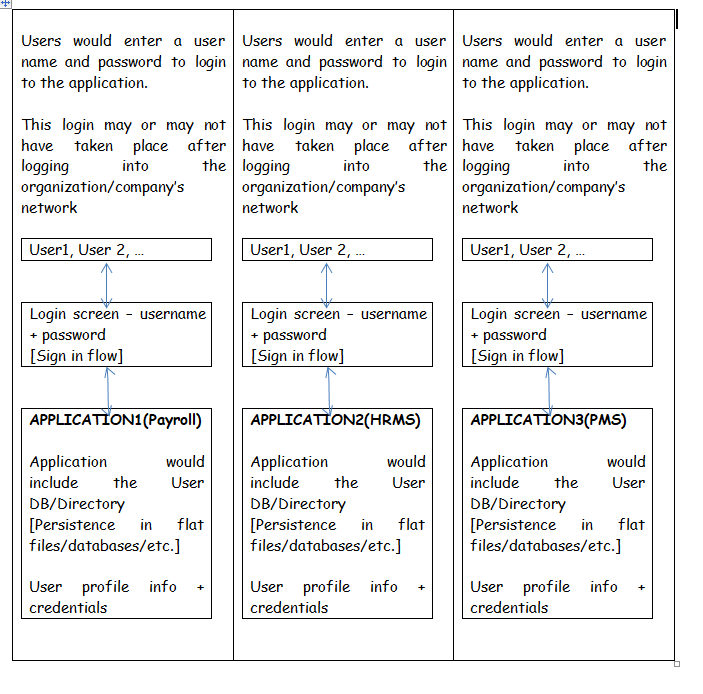
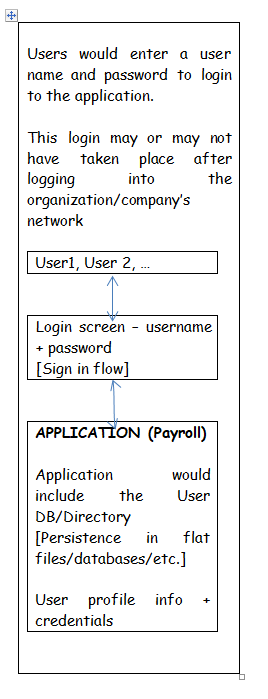
Look at the images below. Irrespective of the number of applications or whether the user first logged into the company network, for each application, the user would have to login to the application.

Each application had to be coded to allow only authorized users to login.

Storage of user profile data, credential data, security would be the responsibility of each individual application.

Use of directories was actually limited and most applications were not coded or integrated with directories (Example – AD, open directories based on LDAP).

Note – User’s identity is still contained within the organization



## Phase 2

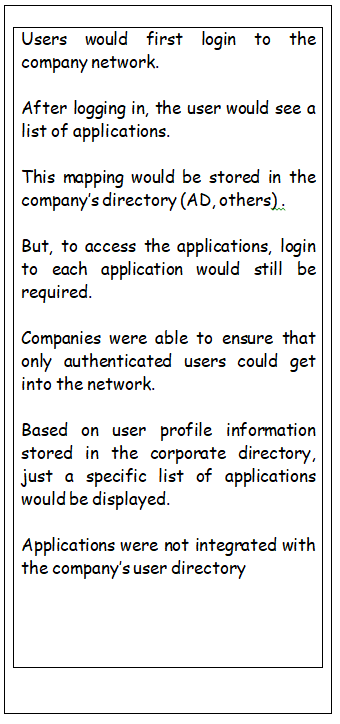
Company wide network login.

Each application had to be coded to allow only authorized users to login.

Storage of user profile data, credential data, security would be the responsibility of each individual application.

Directories ( early versions of IDPs ?) were in place, but most applications were not coded or integrated with directories.

Note – User’s identity is still contained within the organization



## Phase 3

Company wide network login.

Extensive use of company wide User Directories (AD, others)

Storage of user profile data, credential data in AD. Sign in credentials are your AD credentials.

Early versions of IDPs ?

Note – User’s identity is still contained within the organization . Application access is still confined to the company.

|  |  |
| --- | --- |
| |  | | --- | | Users would first login to the company network.  After logging in, the user would see a list of applications.  This mapping would be stored in the company’s directory (AD, others) .  Companies were able to ensure that only authenticated users could get into the network.  Based on user profile information stored in the corporate directory, just a specific list of applications would be displayed.  Applications were getting integrated with the company’s User directory using protocols like LDAP.  Finally, users did not have to separately log into each application. | |

## Phase 4 - Federated Identity -  Supporting application access across company boundaries

Let’s consider the hypothetical example of a retailer [“RETABETA”]. One of the main systems of this retailer is their product marketplace which allows vendors to manage their products (B2B) which are sold to customers [“B2C”].

Imagine there are 1000+ vendors. Each vendor has 50+ users who need to access RETABETA.

For the retailer, managing these external users is an IT administrator’s nightmare.

Each of the vendors has their own systems. Those users now have to cross the boundary and access an external system (“RETABETA”).

|  |  |
| --- | --- |
| **The good old fashioned way**  The retailer takes ownership of user authentication and provides interfaces for vendors to complete the sign in flow.  This is cumbersome and handling use cases related to third party users who leave their organizations or whose roles have changed is error prone. | **The federated way**  Since vendors need to access the systems of the retailer, somehow these vendors need to have some method for handing over authentication of their users to the retailer.  Assuming that the vendors have AD/others in place, user authentication needs to be passed onto/federated to the retailer**.**  So, the retailer (“Service Provider”) externalizes authentication by somehow integrating with the vendor directories (“Identity providers”) |

## Phase 5 - The birth and establishment of Federated Identity Protocols

# The IAM landscape today

## Federated Identity

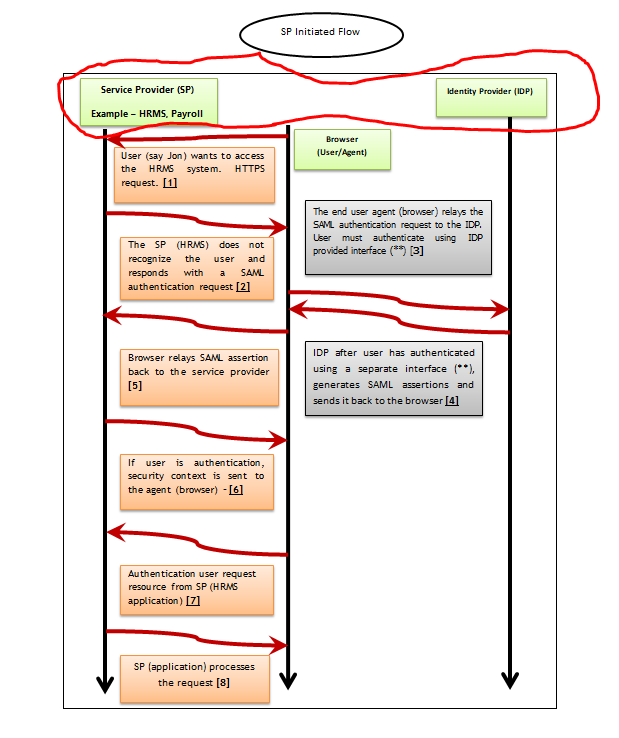
Refer to section 4

## SAML2

This is a protocol that is designed to support Federated Identity.

Mostly used as a web based authentication protocol as the agent involved in the authentication flow is the browser (Chrome, Firefox, Edge)

### SP initiated flow



Points to be noted

1. The Service Provider never directly interacts with the Identity Provider.
2. A browser acts as the agent to carry out all the redirections.
3. The IDP and the SP must have a trusted relationship. To facilitate SAML2 authentication, both parties need to be in agreement and share specific details. This is absolutely essential.
4. Always use a SAML2 toolkit. Purchase one if required
5. The service provider may need to handle more than 1 IDP
6. Ensure that the SP can proper manage SAML2 configuration data
7. Cases where only a subset of users need to login via SAML2 needs to be considered
8. Have secret sign in URLs for administrators in case SAML2 login fails

### IDP initiated flow

To be covered later.

## OAuth2

### Overview

Oauth2 allows a User (“Resource Owner”) to GRANT a third party web site or application (“Client”) access to the user’s protected resources.

In OAuth2, the **client** requests access to the **resources** controlled by the **resource owner** and hosted by the **resource server** and is issued a different set of credentials than that of the resource owner.

Instead of using the resource owner’s credentials to access protected resources, the client obtain an (access/bearer) token.

Tokens are always issued by an Authorization Server with the approval of the resource owner.

The client simply uses the token to access the protected resource hosted by the resource server

### Scopes

These are permissions represented by the OAuth2 access/bearer tokens.

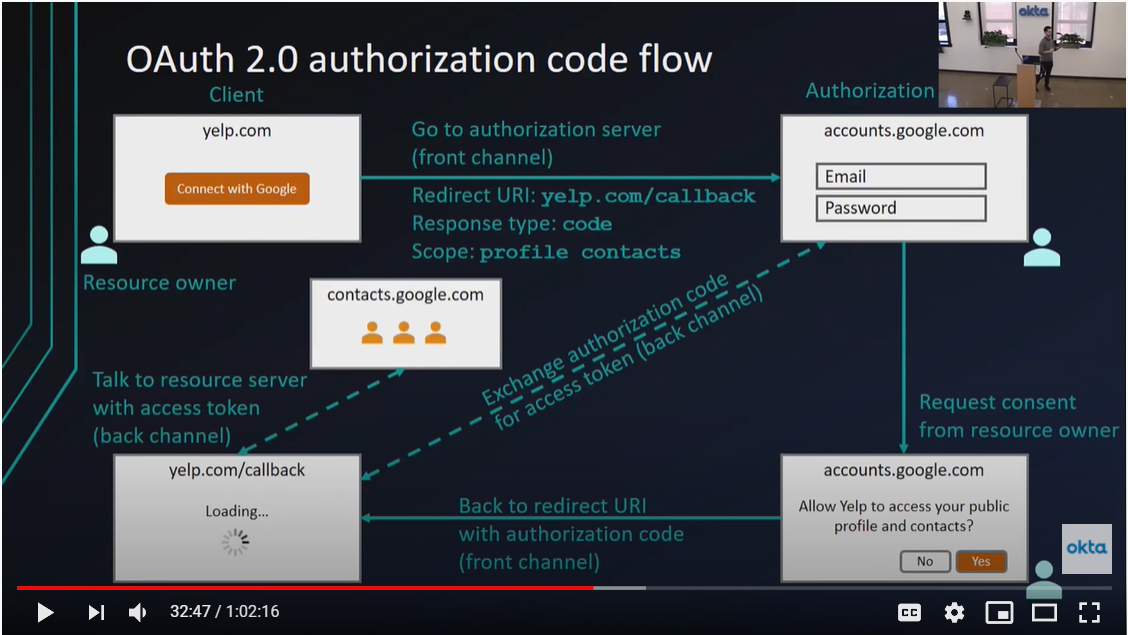
When a client authenticates with an OAuth2.0 Server, it specifies the scopes that it wants.

If those scopes have been authorized by the Resource Owner (“One time setup, …………..), the access token will represent the authorized scopes.

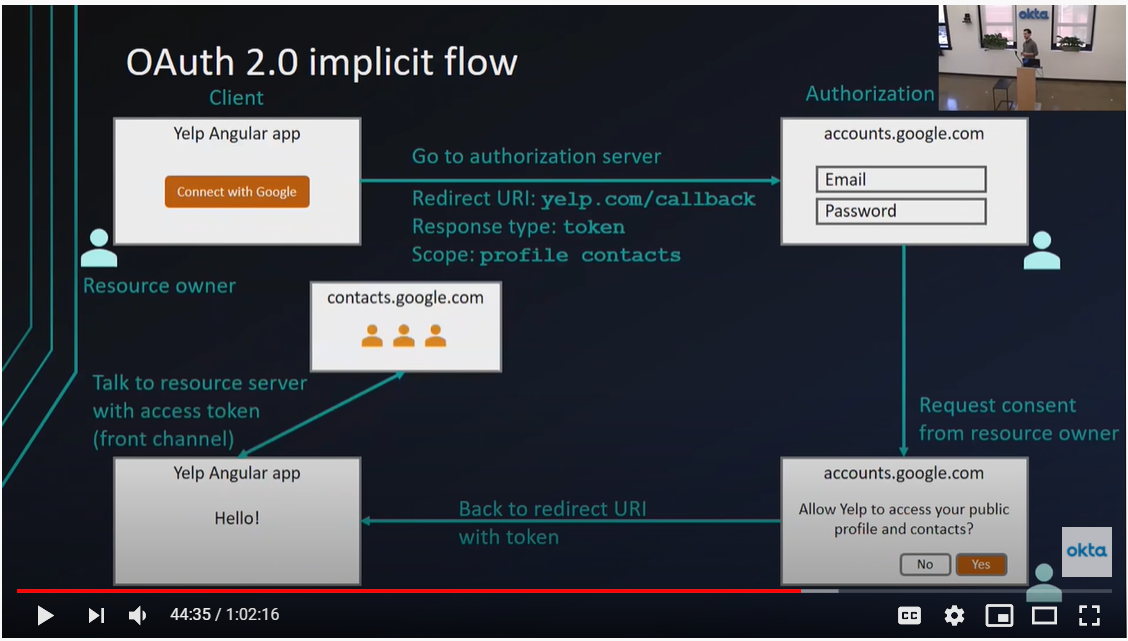
### OAuth2 Authorization Flows

|  |  |  |  |
| --- | --- | --- | --- |
| **The 4 well known flows to get a token**  **(In future there can be more)** | | | |
| **OAuth2.0 flow** | **FC/BC** | **Use Cases** |  |
| Authorization Code | **FC + BC** | Highly trusted apps (first-party apps)  Less trusted apps (third-party apps requesting access to your platform)  This is also used by mobile apps, using the Proof Key for Code Exchange (PKCE) technique. |  |
| Implicit | **FC** | Single-page Javascript Web Applications (for example, Google Fonts) executing on the user’s browser |  |
| Resource owner password credentials | **BC** | Highly trusted apps (first-party apps) | <https://auth0.com/docs/flows/resource-owner-password-flow> |
| Client credentials | **BC** | Non-interactive programs for machine-to-machine communications (for example, background services and daemons) |  |

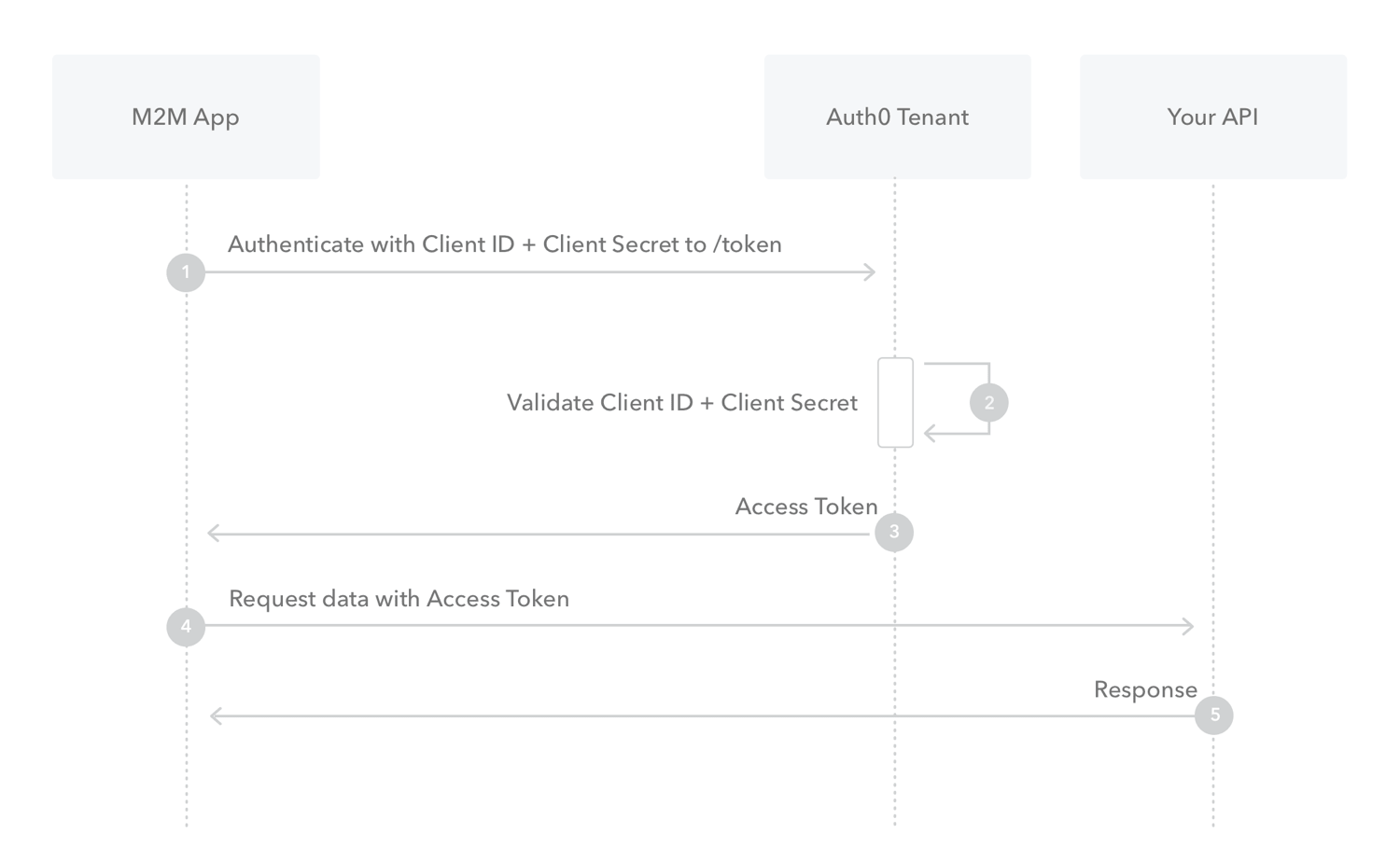
#### Authorization code flow illustration



#### Implicit flow illustration



#### Client credentials flow



### OAuth2 Roles

|  |  |  |
| --- | --- | --- |
| **Resource Owner** | Entity that can grant access to a protected resource [………………..]. Typically, this is the end-user. |  |
| **Resource Server**: | Server hosting the protected resources. This is the API you want to access. |  |
| **Client**: | Application requesting access to a protected resource on behalf of the Resource Owner. |  |
| **Authorization Server**: | Server that authenticates the Resource Owner and issues access tokens after getting proper authorization.  Auth0, Okta, AWS Cognito |  |

### OAuth2.0 End Points

### One time client registration on the Authorization server

**One time setup in the Authorization Server – Creating/registering a client**

OAuth 2.0 requires that clients register with the authorization server beforehand.

*The registration process is not considered as a part of the authorization flow and developers can implement it as they see fit.*

Additionally, your OAuth 2.0 authorization service must be able to verify the identity of clients.

|  |  |
| --- | --- |
| **Client authentication methods** | Client IDs and client secrets |

[Source - <https://medium.com/@ratrosy/understanding-oauth2-and-deploying-a-basic-oauth2-authorization-service-to-cloud-functions-812e3643f94d> ]

## Open ID 1.x/2.x

Ignore.

Focus on Open ID Connect

## OpenID Connect

|  |
| --- |
| (Identity, Authentication) + OAuth 2.0 = OpenID Connect |

## JWT

## SSO

## Multi factor authentication

## Universal Login

## Universal Directory

# Leading vendors

## Auth0

## Okta

|  |
| --- |
| **Identity Provider (IDP)**  **Service Provider (SP)**  **Example – HRMS, Payroll**  **Browser (User/Agent)**  User (say Jon) wants to access the HRMS system. HTTPS request. **[1]**  The end user agent (browser) relays the SAML authentication request to the IDP. User must authenticate using IDP provided interface (\*\*) [**3]**  The SP (HRMS) does not recognize the user and responds with a SAML authentication request **[2]**  IDP after user has authenticated using a separate interface (\*\*), generates SAML assertions and sends it back to the browser **[4]**  Browser relays SAML assertion back to the service provider **[5]**  If user is authentication, security context is sent to the agent (browser) – **[6]**  Authentication user request resource from SP (HRMS application) **[7]**  SP (application) processes the request **[8]** |

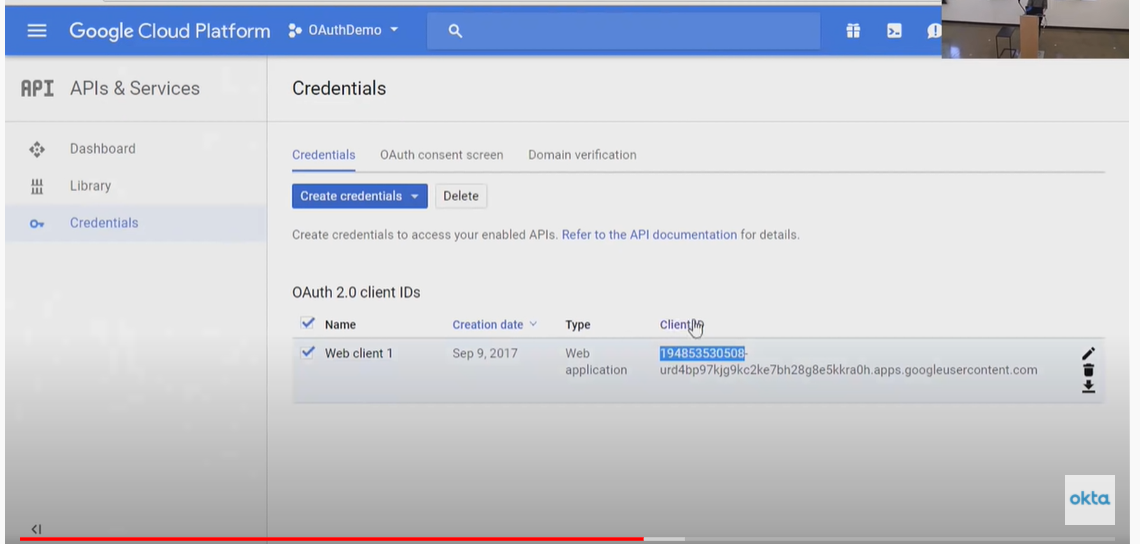
OAuth2 notes

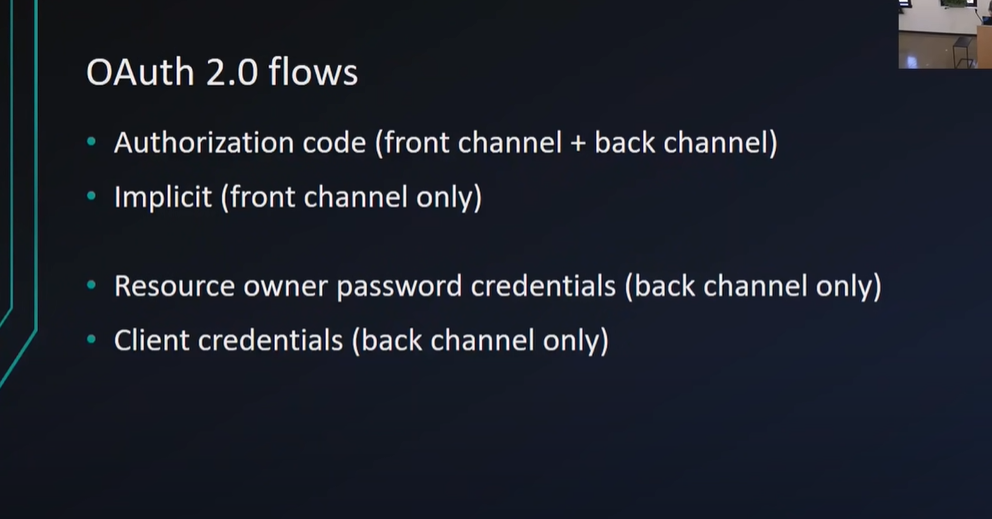
**Yelp wants to access my Google Contacts data . It all starts of on the Yelp site.**

**You provide the consent and agree to the scope of access**

**Response type : code**

Google specific Authorization screens for registering clients. Each server will have their own interfaces.





The Implict flow - Note – here response type is token

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| **OAuth2.0 flow** | **F/B** | **Use Cases** |  |
| Authorization Code | **F + B** | Highly trusted apps (first-party apps)  Less trusted apps (third-party apps requesting access to your platform) |  |
| Implicit | **F** | Single-page Javascript Web Applications (for example, Google Fonts) |  |
| Resource owner password credentials | **B** | Highly trusted apps (first-party apps) | <https://auth0.com/docs/flows/resource-owner-password-flow> |
| Client credentials | **B** | Non-interactive programs for machine-to-machine communications (for example, background services and daemons) |  |

|  |
| --- |
| OAuth 2.0 ([RFC 6479](https://tools.ietf.org/html/rfc6479)) is a widely used authorization framework enabling applications to access resources in all kinds of services. More specifically, OAuth 2.0 allows arbitrary **clients** (for example, a highly trusted first-party mobile app or a less trusted third-party web app) to access user’s (**resource owner**’s) resources on **resource servers** via **authorization servers** in a secure, reliable, and efficient manner.  Decoding the above statement  Resource Owner – Zeppertek  Resources - Set of REST APIs developed by Zeppertek  Resource Servers – ??????????????????????  Authorization Server – Okta, OAuth2 |