# Generic Openid Connect (OIDC) Client

# **Contents**

[Contents 2](#__RefHeading___Toc160_1221129489)

[Introduction 3](#__RefHeading___Toc484102793)

[Overview of components 3](#__RefHeading___Toc484102794)

[High Level Functionality of OpenID Client 3](#__RefHeading___Toc162_1221129489)

[Flow Diagram 6](#__RefHeading___Toc425_1724828375)

# **Introduction**

This document provides the high level design of Generic OIDC (Openid Connect Client). It is is a simple identity layer on top of the Oauth 2.0 protocol, which allows computing clients to verify the identity of an end-user based on the authentication performed by an authorization server, as well as to obtain basic profile information about the end-user. OpenID Connect lets registered clients authenticate their users across websites and apps without having to own and manage password files. For the application builder, it provides a secure verifiable, answer to the question: “What is the identity of the person currently using the browser or native application that is connected to me?”

# **Overview of components**

**There are 2 important components:**

1. OIDC Client
2. User Agent – Browser

**Details of the components:**

1. **OIDC Client:** OIDC is a specification built on top of Oauth 2.0 protocol to provide authentication capabilities over the existing authorization provided by Oauth 2.0.OIDC Client is generic i.e. it can be used to test most OpenID Providers. OIDC will contain the client id and secret provided by OIDC Provider and will authenticate the user as well as the client and provide authorization to the client. It works by exchanging the code to get the id token and then verifying the token by validating the JWT Signature and finally getting the response. The id\_token is a JWT(JSON Web token) and hence, it is more secure and reliable.
2. **User Agent (Browser):-**  Browser will be acting as the user agent which will register the client information and all the requests to the OpenID provider and OIDC Client will be made through it. Browser pop-ups are preferred way for web apps to redirect the user to the idP as they are more secure.

# **High Level Functionality of OpenID Client**

**Important Functions:-**

1. Oauth 2.0 Protocol
2. OIDC Scope.
3. Id Tokens.
4. Code Flow.

**Detail of the Functionality:**

1. **Oauth 2.0:** The Oauth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf.
2. **OIDC Scope:** OIDC defines several scopes. Some of them are:

* **OpenID**: Informs the Authorization Server that the Client is making an OpenID Connect request. If the OpenID scope value is not present, the request MUST NOT be treated as an OpenID Connect request. The OpenID value also requests that the ID Token associated with the authentication session be returned. If the response\_type includes token, the ID Token is returned in the Authorization Response along with the Access Token. If the response\_type includes code, the ID Token is returned as part of the Token Endpoint response. This scope value requests access to the user\_id Claim at the User Info Endpoint.
* **Profile:**This requests that access to the End-User’s profile Claims excluding the address and email Claims at the User info Endpoint be granted by the issued Access Token.
* **Email:** This requests that access to the email and verified Claims at the User Info Endpoint be granted by the issued Access Token.
* **Address:**This requests that access to address Claim at the User Info Endpoint be granted by the issued Access Token.

1. **ID Token:** The ID token is an identity card of the user in a standard JWT (JSON Web Token) format signed by the OpenID Provider (OP).ID token is obtained when the client sends the user to their OP with an authentication request.
   * FEATURES of ID Token:

“*sub*”**:** Asserts the user identity. (**“**can be a name or email address.”)

**“***iss***”:** Specifies the issuing authority.

“*aud*”: Audience of the ID Token i.e. the client.

“*nonce*”: Coined for a particular occasion.

“*auth\_time*” & “*acr*”:ID token species the time the user was authenticated and in terms of strength the user was authenticated.

“*iat*” & “*exp*”: ID token issue and expiration date.

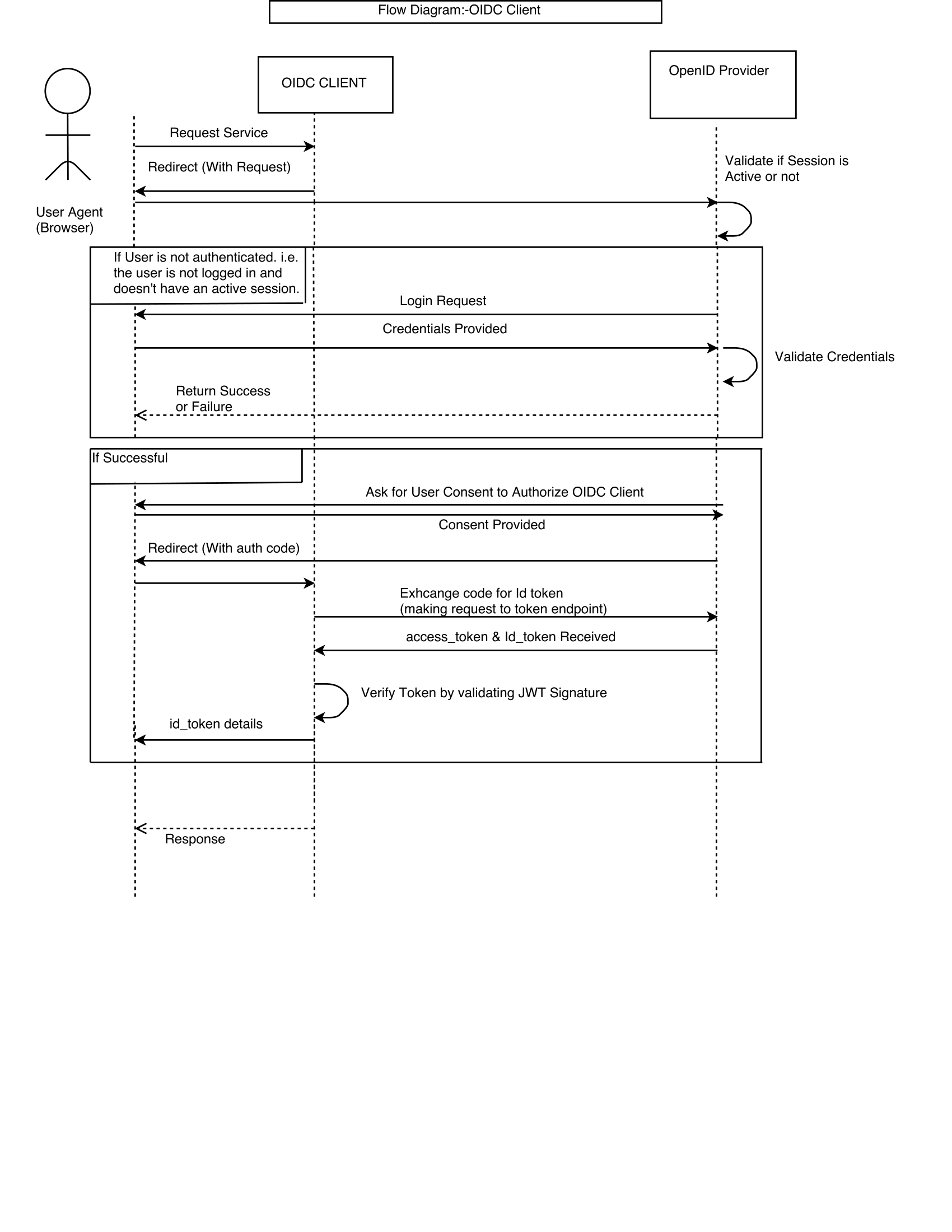
1. **Code Flow:** There are three types of code flow.

* **Authorization Code Flow:-** The most commonly used flow, intended for traditional web apps as well as native / mobile apps. Involves an initial browser redirection to / from the OP for user authentication and consent, then a second back-channel request to retrieve the ID token. This flow offers optimal security, as tokens are not revealed to the browser and the client app can also be authenticated.
* **Implicit Code Flow:-** For browser (JavaScript) based apps that don’t have a back-end. The ID token is received directly with the redirection response from the OP. No back-channel request is required here.

Only authorization code flow will be supported in the Current Scope. Support for implicit can be added in the future scope.

# **Flow Diagram**

**Diagram: Flow diagram for Generic OIDC Client.**

****

**Following are the details of the steps:**

1. User requests a service to the OIDC Client and the OIDC Client redirect the request to the OpenID Provider to authenticate the user.
2. The OpenID Provider Validates if the session is active or not.
3. If the session is not active, the user is asked to log in with his/her credentials to proceed further.
4. If user is authenticated successfully, OpenID Provider will Ask for user consent to authorize the OIDC Client.
5. User will provide the consent & the OpenID Provider will redirect with authorization code or else with the id token in case of implicit flow.
6. The OIDC Client will now exchange this code for an id token by making request to token endpoint.
7. The OpenID Provider will provide the OIDC Client with the access\_token and id\_token.
8. The OIDC Client will validate if the token is valid or not by validating it’s JWT signature.
9. The OIDC will provide the user agent with the token details.
10. Lastly, Response will be provided by the OIDC Client to the user agent.

**End of Document**