4. Obtaining Authorization bölümü, OAuth 2.0 protokolünde istemcinin erişim token'ı almak için nasıl bir yetkilendirme (authorization) süreci başlatması gerektiğini ve bu sürecin nasıl işlediğini açıklar. Temelde, istemci erişim token'ı almak için **resource owner**'dan (kaynak sahibi) yetki alır. Bu yetkilendirme, istemcinin bir **authorization grant** (yetkilendirme izni) elde etmesine olanak tanır. İstemci, bu yetkilendirme iznini kullanarak erişim token'ını almak için token endpoint'ine başvurur.

1. Authorization Grant (Yetkilendirme İzni)

OAuth 2.0'da, bir istemcinin erişim token'ı talep etmesi için öncelikle **authorization grant** adı verilen bir izne ihtiyacı vardır. Bu **grant**, istemcinin kaynak sahibinden aldığı yetkiyi temsil eder. Authorization grant, istemcinin belirli bir kaynağa veya kaynağa dair belirli izinlere erişim hakkı kazandığını gösterir.

Authorization grant türleri şunlardır:

2. Grant Types (Yetkilendirme Türleri)

OAuth 2.0, istemcilerin yetki alırken kullanabileceği dört ana yetkilendirme türü tanımlar. Her tür, farklı kullanım senaryolarına hizmet eder ve güvenlik gereksinimlerine göre farklı şekillerde işlenir:

- **Authorization Code Grant (Yetkilendirme Kodu Türü)**: Bu tür, istemciye kullanıcı tarafından bir yetkilendirme kodu verilmesini gerektirir. Genellikle sunucu tarafı uygulamaları için kullanılır. İstemci, kullanıcıyı yetkilendirme sunucusuna yönlendirir, kullanıcı orada kimliğini doğrular ve onay verir. Sonrasında, yetkilendirme sunucusu istemciye bir kod (authorization code) döner. Bu kod, istemcinin **token endpoint**'inden bir erişim token'ı almak için kullanılır. Bu tür, yüksek güvenlik sağlar çünkü client secret (istemci kimliği) istemcinin sunucusunda saklanır ve erişim token'ı sunucudan alınır.
- **Implicit Grant (Açık Erişim Türü)**: Bu tür, istemcinin doğrudan bir erişim token'ı almasını sağlar. Genellikle tarayıcı tabanlı uygulamalar (client-side) için kullanılır. İstemci, yetkilendirme sunucusuna kullanıcıyı yönlendirir ve kullanıcı kimliğini doğruladıktan sonra erişim token'ı doğrudan istemciye yönlendirilir. Implicit grant genellikle daha düşük güvenlikli ve hızlıdır, ancak token'lar doğrudan istemcinin tarayıcısında bulunur, bu da güvenlik risklerini artırabilir.
- **Resource Owner Password Credentials Grant (Kaynak Sahibi Şifre Kimlik Doğrulaması Türü)**: Bu türde, kullanıcı, kullanıcı adı ve şifresini doğrudan istemciye verir. İstemci, bu bilgileri kullanarak doğrudan erişim token'ı almak için yetkilendirme sunucusuna başvurur. Bu tür genellikle, güvenilen uygulamalar için kullanılır, çünkü kullanıcı adı ve şifre istemciye doğrudan verilmiş olur. Bu nedenle güvenlik riskleri taşır ve sadece istemcinin tamamen güvenli olduğuna emin olunduğunda kullanılmalıdır.
- **Client Credentials Grant (İstemci Kimlik Doğrulama Türü)**: Bu tür, istemcinin kendisi adına bir erişim token'ı almasını sağlar. İstemci, kendi kimliğini doğrulamak için client id ve client secret bilgilerini kullanır. Bu tür genellikle istemci uygulamalarının API'lere erişmek için kullanılır, çünkü kullanıcı bilgisi gerekmez. Örneğin, bir mikroservis API'si, sadece istemci uygulamanın kimliğini doğrulayarak kendi kaynaklarına erişebilir.

3. Extension Grant Types (Uzantı Yetkilendirme Türleri)

OAuth 2.0, ek yetkilendirme türlerinin tanımlanmasına olanak tanır. Yani, OAuth 2.0'un sunduğu dört ana grant türüne ek olarak, yeni türler tanımlanabilir. Bu türler, OAuth 2.0 protokolünün genişletilebilirliğini sağlar ve özel ihtiyaçları karşılayacak şekilde yapılandırılabilir.

4. Authorization Grant Süreci

OAuth 2.0'daki yetkilendirme süreci temelde şu şekilde işler:

- İstemci, **authorization grant** almak için bir yetkilendirme türü seçer ve kullanıcıyı yetkilendirme sunucusuna yönlendirir.
- Kullanıcı, yetkilendirme sunucusunda kimlik doğrulaması yapar ve istemcinin isteklerine onay verir.
- Yetkilendirme sunucusu, istemciye bir **authorization grant** (yetki izni) sağlar.
- İstemci, **authorization grant**'i kullanarak erişim token'ı almak için **token endpoint**'ine başvurur.

Bu süreç, istemcilerin erişim token'ları alabilmesi için güvenli bir şekilde yetkilendirilmesini sağlar ve OAuth 2.0 protokolü, istemcilerin sadece gerekli kaynaklara erişmesini garanti eder.

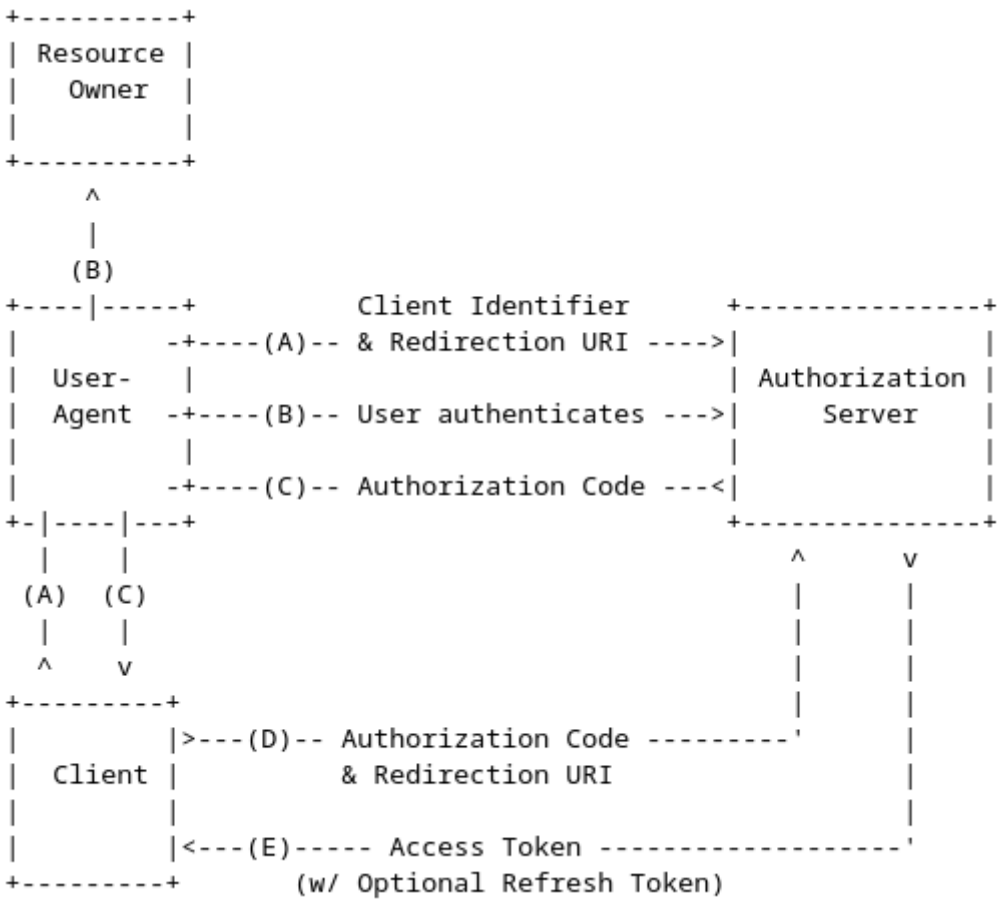
Özetle:

- OAuth 2.0'da **authorization** almak için istemci, kullanıcının onayını alarak **authorization grant** elde eder.
- Bu grant, istemcinin erişim token'ı almak için kullanacağı bir türdür.
- OAuth, istemcinin farklı kullanım senaryolarına göre dört ana grant türü ve uzantı türleri sunar.

Bu yapı, OAuth 2.0'ı esnek, güvenli ve farklı senaryolara uyarlanabilir bir yetkilendirme protokolü haline getirir.

4.1. Authorization Code Grant (Yetkilendirme Kodu Türü) bölümü, OAuth 2.0'da istemcilerin erişim token'ı ve tazeleme (refresh) token'ı almak için kullandığı bir yetkilendirme türünü açıklar. Bu tür, özellikle **confidential clients** (gizli istemciler) için optimize edilmiştir ve bir **redirection-based flow** (yönlendirme tabanlı akış) kullanır. Yani, istemci, kullanıcının bir web tarayıcısı aracılığıyla yetkilendirme sunucusuna yönlendirilmesini sağlar ve ardından kullanıcı onayı alındığında bir yetkilendirme kodu (authorization code) alır. Bu kod daha sonra erişim token'ı almak için kullanılır.

Bu akışın her aşamasını aşağıdaki gibi adım adım inceleyelim:



1. (A) Akışın Başlatılması

İstemci, **resource owner's user-agent** (kaynak sahibinin kullanıcı ajanı, genellikle bir web tarayıcısı) aracılığıyla yetkilendirme sunucusunun yetkilendirme endpoint'ine bir istek gönderir. Bu istekle birlikte aşağıdaki bilgiler iletilir:

- **Client Identifier**: İstemcinin kimliğini tanımlayan benzersiz bir etiket (client_id).
- **Requested Scope**: İstemcinin erişmek istediği kaynakların kapsamı.
- **Local State**: İstemcinin, kullanıcı ile etkileşim sırasında güvenliği sağlamak için kullanabileceği durum bilgisi (örneğin, CSRF saldırılarına karşı).
- **Redirection URI**: Kullanıcı yetkilendirmeyi tamamladıktan sonra, yetkilendirme sunucusunun kullanıcının tarayıcısını yönlendireceği URI. Bu URI, istemcinin kaydettiği ve doğrulamak için kullandığı URI ile eşleşmelidir.

2. (B) Kullanıcı Kimlik Doğrulaması ve Onay

Yetkilendirme sunucusu, **resource owner** (kaynak sahibi, yani kullanıcı) kimliğini doğrular ve istemcinin talep ettiği erişim izinlerine dair kullanıcıdan onay alır. Bu adımda, kullanıcıya genellikle erişim taleplerinin ne olduğunu ve bu talepleri kabul edip etmeyeceklerini belirten bir ekran gösterilir. Kullanıcı, talepleri onaylarsa, akış devam eder; reddederse, işlem sonlanır.

3. (C) Yetkilendirme Kodu İle Yönlendirme

Eğer kaynak sahibi erişim talebini onaylarsa, yetkilendirme sunucusu, kullanıcının tarayıcısını istemcinin verdiği **redirection URI**'ye yönlendirir. Yönlendirme sırasında:

- **Authorization Code** (Yetkilendirme Kodu): Yetkilendirme sunucusu tarafından verilen ve istemcinin daha sonra erişim token'ı almak için kullanacağı bir kod gönderilir.
- **Local State**: İstemcinin başlangıçta gönderdiği yerel durum bilgisi, ek güvenlik için redirection URI'siyle birlikte gönderilebilir.

4. (D) Erişim Token'ı Talebi

İstemci, aldığı **authorization code**'u kullanarak, **token endpoint**'e bir POST isteği gönderir. Bu istekle birlikte aşağıdaki bilgiler sunulur:

- **Authorization Code**: Kullanıcının onayıyla verilen yetkilendirme kodu.
- **Redirection URI**: Yetkilendirme kodunun alınmasında kullanılan redirection URI'si. Bu URI, doğrulama amacıyla tekrar gönderilir.

Bu adımda, istemci yetkilendirme sunucusuna **client authentication** (istemci kimlik doğrulaması) yapar. Yani istemci, kendini yetkilendirme sunucusuna tanıtarak doğrular (örneğin, client secret kullanılarak).

5. (E) Erişim Token'ı ve (Opsiyonel) Refresh Token'ın Alınması

Yetkilendirme sunucusu, istemcinin kimlik doğrulamasını yapar, **authorization code**'u doğrular ve **redirection URI**'yi kontrol eder. Eğer tüm bilgiler geçerli ise:

- **Access Token** (Erişim Token'ı): İstemciye erişim izni verilen kaynaklara ulaşması için gerekli olan token sağlanır.
- **Refresh Token** (Tazeleme Token'ı): Erişim token'ının süresi dolduğunda, istemcinin yenisini almak için kullanabileceği bir token sağlanabilir.

Bu adımlar tamamlandığında istemci, kullanıcı adına kaynaklara erişim için gerekli olan token'ları almış olur.

Akış Özeti

1. **İstemci**, kullanıcıyı yetkilendirme sunucusuna yönlendirir (authorization code flow başlatılır).
2. **Kullanıcı**, erişim talebini onaylar veya reddeder.
3. **Yetkilendirme Sunucusu**, kullanıcıyı istemcinin belirttiği URI'ye yönlendirirken bir **authorization code** sağlar.
4. **İstemci**, bu yetkilendirme kodunu kullanarak **access token** alır.

Bu Akışın Avantajları

- **Confidential Clients** için güvenlidir. Çünkü istemci, erişim token'ını almak için authorization code'u kullanır, bu da daha güvenli bir yol sunar.
- **Refresh token** ile erişim token'ının süresi dolduğunda kullanıcı tekrar kimlik doğrulama yapmak zorunda kalmaz.
- **Token güvenliği** açısından da faydalıdır, çünkü token doğrudan istemciye verilmez; önce bir authorization code kullanılır ve ardından erişim token'ı alınır.

Özetle

Authorization Code Grant, OAuth 2.0 protokolünde en güvenli ve yaygın kullanılan grant türüdür ve genellikle web uygulamaları gibi **confidential clients** tarafından tercih edilir. Bu akış, istemcinin daha sonra erişim token'ı alabilmesi için kullanıcıdan bir authorization code elde etmesine dayanır.

4.1.1. Authorization Request (Yetkilendirme İsteği) bölümü, istemcinin yetkilendirme sunucusuna yaptığı istekleri ve bu isteklerin doğru bir şekilde nasıl yapılandırılması gerektiğini açıklar. Bu bölümde, istemcinin yetkilendirme sunucusuna yapacağı HTTP isteği, gerekli parametreler ve isteklerin nasıl yapılması gerektiği belirtilmiştir.

Adım Adım Açıklama

1. **Authorization Request (Yetkilendirme İsteği) Oluşturma** İstemci, **authorization endpoint URI**'ye (yetkilendirme sunucusunun belirtilen URL'si) aşağıdaki parametrelerle bir URI (Uniform Resource Identifier) oluşturur. Bu URI'yi, istemci kullanıcının tarayıcısına yönlendirmek için kullanır. İstek, **application/x-www-form-urlencoded** formatında yapılır (bu, parametrelerin URL'nin sorgu kısmına eklenmesi anlamına gelir).

İstek Parametreleri

- **response_type** (Zorunlu): Değerinin **"code"** olması gerekir. Bu, istemcinin **authorization code** almak istediğini belirtir. Yani istemci, erişim token'ı almak için yetkilendirme kodu almak üzere bu parametreyi kullanır.
- **client_id** (Zorunlu): Bu, istemcinin kimliğini belirten benzersiz bir tanımlayıcıdır. İstemcinin yetkilendirme sunucusuna kaydedilmiş ve tanımlanmış olmalıdır. Bu parametre, istemcinin kimliğini yetkilendirme sunucusuna iletmek için kullanılır.
- **redirect_uri** (Opsiyonel): Bu, kullanıcı yetkilendirme işlemini tamamladıktan sonra, yetkilendirme sunucusunun kullanıcıyı geri yönlendireceği URI'dir. Eğer istemci daha önce kaydettiyse, bu URI'yi içermelidir. Eğer bu parametre sağlanmazsa, yetkilendirme sunucusu varsayılan URI'yi kullanabilir. Bu parametre, kullanıcı yetkilendirme işlemini tamamladığında, istemcinin doğru yere yönlendirilmesini sağlar.
- **scope** (Opsiyonel): İstemcinin erişim talep ettiği kaynakların kapsamını belirler. Örneğin, istemci sadece kullanıcı profilini mi almak istiyor, yoksa daha geniş bir erişim mi talep ediyor? Sunucu, istemcinin talep ettiği kapsamı dikkate alarak, erişim token'ı verirken belirli yetkilerle sınırlandırabilir.

- **state** (Tavsiye Edilir): Bu, istemci tarafından kullanılan, istek ve callback arasındaki durumu korumaya yarayan opak (şeffaf olmayan) bir değerdir. Yani, istemci bir istek gönderdiğinde, bu parametre istemcinin işlem durumu ile ilgili bilgileri tutar. Sunucu, kullanıcıyı yetkilendirme işlemi tamamlandıktan sonra yönlendirdiğinde, istemci bu **state** parametresini geri alır ve işlemde herhangi bir güvenlik açığı olmadan süreci yönetebilir. Bu parametre, **Cross-Site Request Forgery (CSRF)** saldırılarına karşı bir önlem olarak da kullanılır.

İstek Örneği

Örneğin, istemci kullanıcıyı şu şekilde bir HTTP GET isteği ile yönlendirebilir:

```
GET /authorize?response_type=code&client_id=s6BhdRkqt3&state=xyz
  &redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb HTTP/1.1
Host: server.example.com
```

Bu istek, istemcinin yetkilendirme kodunu almak için yetkilendirme sunucusuna yaptığı ilk adımdır. Burada:

- `response_type=code`: Yetkilendirme kodunun istenmesini belirtir.
- `client_id=s6BhdRkqt3`: İstemcinin kimliğini belirtir.
- `state=xyz`: İstemcinin durumu (örneğin, güvenlik amaçlı bir koruma).
- `redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb`: Yetkilendirme işleminden sonra kullanıcının yönlendirileceği URI.

Sunucu Tarafında Yapılacak Doğrulamalar

Yetkilendirme sunucusu, istemciden gelen bu isteği aldıktan sonra şu doğrulamaları yapar:

- **Gerekli Parametrelerin Varlığı**: İstekten gerekli parametrelerin (response_type, client_id) eksik olmaması gerekir. Eğer bu parametrelerden biri eksikse, istek geçersiz sayılabilir.
- **Parametrelerin Geçerliliği**: Gelen parametrelerin geçerli olup olmadığı doğrulanır. Örneğin, client_id'nin doğru istemciyi temsil ettiğinden emin olunmalıdır.
- **Redirection URI'nin Kaydı**: Eğer bir **redirect_uri** sağlanmışsa, yetkilendirme sunucusu bu URI'yi istemcinin kaydına göre doğrular. URI'nin kayıtsız olması veya uyumsuzluk olması durumunda, istek geçersiz kabul edilebilir.

Sunucu İşlemleri

- **Kimlik Doğrulama ve Onay**: Yetkilendirme sunucusu, kaynak sahibini (kullanıcıyı) kimlik doğrulama işlemi yaparak onay alır. Bu işlem, kullanıcıdan parolalarını girmelerini veya başka kimlik doğrulama yöntemlerini kullanmalarını içerebilir.
- **Yönlendirme**: Eğer kullanıcı onay verirse, yetkilendirme sunucusu, kullanıcıyı belirlenen **redirect_uri**'ye yönlendirir. Bu yönlendirme ile birlikte, kullanıcının onayına dair veriler ve **authorization code** (yetkilendirme kodu) gönderilir.

Sonuç

Authorization Request bölümü, istemcinin yetkilendirme sunucusuna, kullanıcı adına erişim talep etmek için yaptığı ilk adımdır. Bu adımda, istemci gerekli parametrelerle bir URI oluşturur ve

kullanıcının tarayıcısını bu URI'ye yönlendirir. Sunucu, parametreleri doğruladıktan sonra, kullanıcının onayını alır ve istemciyi doğru URI'ye yönlendirir.

4.1.2. Authorization Response *(Yetkilendirme Yanıtı) bölümü, yetkilendirme sunucusunun, istemcinin yetkilendirme isteğini kabul etmesi durumunda yaptığı yanıtı ve yanıtın nasıl yapılandırılacağını açıklar. Bu bölüm, istemcinin aldığı yetkilendirme kodunu içeren yanıtın formatını ve gerekli parametreleri tanımlar.*

Adım Adım Açıklama

1. **Authorization Response (Yetkilendirme Yanıtı) Oluşumu**

   Yetkilendirme sunucusu, **yetkilendirme isteğini onayladıktan sonra** istemciye bir **yetkilendirme kodu** (authorization code) gönderir. Bu, istemcinin daha sonra erişim token'ı alabilmesi için kullanacağı bir koddur. Yanıt, istemcinin belirlediği **redirection URI**'ye yapılacak bir yönlendirme ile gerçekleşir. Yönlendirme URL'si aşağıdaki parametreleri içerir:

Yanıt Parametreleri

- **code** (Zorunlu): Bu, yetkilendirme sunucusunun oluşturduğu **yetkilendirme kodu**'dur. İstemci bu kodu, ilerleyen adımlarda erişim token'ı almak için kullanacaktır.
    - **Geçerlilik Süresi**: Yetkilendirme kodu, kullanıldıktan sonra çok kısa bir süre içinde (maksimum 10 dakika) geçerliliğini yitirmelidir. Bu, güvenlik amacıyla, kodun sızması durumunda kötüye kullanılmasını engellemek için önemlidir.
    - **Tek Kullanımlık**: Yetkilendirme kodu yalnızca bir kez kullanılabilir. Eğer bir yetkilendirme kodu birden fazla kez kullanılırsa, yetkilendirme sunucusu isteği reddeder ve mümkünse, bu koda dayalı olarak verilen tüm token'ları iptal eder.
    - **Bağlı Olma**: Yetkilendirme kodu, istemcinin kimliğine (client_id) ve kullanılan **redirection URI**'ye bağlıdır. Yani, yalnızca doğru istemci ve doğru URI ile kullanılabilir.
- **state** (Zorunlu, Eğer "state" Parametresi İstemci İsteğinde Varsa): Eğer istemci, yetkilendirme isteği sırasında bir **state** parametresi gönderdiysa, yetkilendirme sunucusu aynı değeri yanıt olarak göndermelidir. Bu parametre, istemcinin gönderdiği **state** değerini geri almasını sağlar. **State** parametresi, özellikle **CSRF (Cross-Site Request Forgery)** saldırılarını engellemek amacıyla kullanılır.

Yönlendirme Yanıtı Örneği

Yetkilendirme sunucusunun istemciye verdiği yanıt, aşağıdaki gibi bir HTTP yönlendirme yanıtı olabilir:

HTTP/1.1 302 Found
Location: https://client.example.com/cb?code=SplxlOBeZQQYbYS6WxSbIA
&state=xyz

Burada:

- **Location** başlığı, istemcinin yönlendirilmesi gereken URI'yi belirtir.
- **code=SplxlOBeZQQYbYS6WxSbIA**: Bu, yetkilendirme sunucusunun verdiği yetkilendirme kodudur. İstemci, bu kodu alarak erişim token'ı almak için kullanacaktır.
- **state=xyz**: Eğer istemci, isteğinde **state** parametresi göndermişse, bu değer burada geri gönderilir.

Sunucu Tarafında Yapılacak Doğrulamalar

- **Yetkilendirme Kodu**: Sunucu, istemcinin geçerli bir yetkilendirme kodu alıp almadığını kontrol eder. Bu kod, yalnızca bir kez kullanılabilir ve istemcinin doğru kimlik bilgileri ve URI ile istek yapmış olması gerekir.
- **State Parametresi**: Eğer istemci bir **state** parametresi göndermişse, yetkilendirme sunucusu bu parametreyi geri gönderecektir. Bu, istemcinin doğru bir işlem yaptığını doğrulaması için gereklidir.
- **Geçerlilik Süresi**: Yetkilendirme kodunun geçerlilik süresi sınırlıdır. Genellikle 10 dakika içinde geçerliliğini yitirir. Bu, saldırganların kodu ele geçirmesini ve kötüye kullanmasını engellemek için önemlidir.

İstemci Tarafında Yapılacaklar

- **Yanıt Parametrelerini İşleme**: İstemci, yetkilendirme sunucusundan gelen **code** (yetkilendirme kodu) ve **state** parametrelerini işlemelidir. Eğer sunucu tanımadığı parametreleri gönderirse, istemci bu parametreleri göz ardı etmelidir.
- **Authorization Code Kullanımı**: İstemci, aldığı **yetkilendirme kodunu** doğrulayıp, bir **access token** almak için token endpoint'e bir istek yapar. Bu noktada istemci, doğru istemci kimliği ve doğru **redirection URI** ile kodu kullanmalıdır.

Sonuç

Authorization Response bölümü, yetkilendirme sunucusunun istemciye **yetkilendirme kodu** verdiği yanıt sürecini açıklar. Bu süreç, istemcinin doğru bir şekilde yetkilendirilip edilmediğini ve yönlendirme URI'sine ne tür parametrelerin eklenmesi gerektiğini belirler. Sunucu, geçerli bir kod ve istemci tarafından sağlanan **state** parametresini geri göndererek, istemcinin güvenli bir şekilde sonraki adımlara geçmesini sağlar.

***4.1.2.1. Error Response** (Hata Yanıtı) bölümü, istemcinin yetkilendirme isteği sırasında bir hata meydana gelmesi durumunda, yetkilendirme sunucusunun nasıl bir yanıt vereceğini açıklar. Bu yanıt, istemcinin hata durumunu doğru şekilde anlaması ve uygun şekilde işlem yapabilmesi için gerekli bilgileri içerir.*

Hata Durumları ve Yanıtları

Hata yanıtı, istemcinin yetkilendirme isteği sırasında karşılaştığı sorunları belirtmek için kullanılır. Eğer bir hata oluşursa, **yetkilendirme sunucusu**, istemciyi doğrudan geçersiz **redirection URI**'ye yönlendirmez. Bunun yerine, hata mesajlarını içeren bir HTTP yanıtı gönderir. Hata yanıtı,

istemcinin daha sonra hatanın nedenini anlaması ve gerekli düzeltmeleri yapabilmesi için gereken bilgileri içerir.

Aşağıda, hata durumları ve her bir hata türü için kullanılan parametreler açıklanmıştır.

Hata Parametreleri

1. **error** (Zorunlu): Hata kodu, isteğin neden başarısız olduğunu belirtir. Bu kodlar, hata türüne göre belirlenir. Bu hata kodları, yalnızca belirli ASCII karakterlerden oluşmalıdır ve aşağıdaki değerlerden biri olabilir:
    - **invalid_request**: İstek eksik bir parametre içeriyor, geçersiz bir parametre değeri var, bir parametre birden fazla kez kullanılmış veya istek biçimsel olarak yanlış.
    - **unauthorized_client**: İstemci, belirtilen yetkilendirme kodunu almak için yetkilendirilmemiştir.
    - **access_denied**: Kaynak sahibi ya da yetkilendirme sunucusu, erişim isteğini reddetmiştir.
    - **unsupported_response_type**: Yetkilendirme sunucusu, belirtilen yöntemle yetkilendirme kodu almak için destek sunmuyor.
    - **invalid_scope**: İstenilen kapsam (scope), geçersiz, bilinmeyen veya hatalı biçimlendirilmiş.
    - **server_error**: Yetkilendirme sunucusu, isteği yerine getiremeyen beklenmeyen bir durumla karşılaştı.
    - **temporarily_unavailable**: Yetkilendirme sunucusu geçici bir yoğunluk veya bakım nedeniyle isteği işleyemiyor.

    Not: `error` parametresinin değeri yalnızca belirli karakterlerden oluşmalıdır ve bu karakterler şu setle sınırlıdır: `%x20-21 / %x23-5B / %x5D-7E`.
2. **error_description** (Opsiyonel): İnsan tarafından okunabilir, ASCII formatında açıklamalar sağlayan bir metin. Bu açıklama, geliştiricinin hatayı anlamasında yardımcı olur. **error_description** parametresi de yalnızca belirli ASCII karakterleri içerebilir.
3. **error_uri** (Opsiyonel): Hata hakkında daha fazla bilgi sağlamak için bir URI (Uniform Resource Identifier) içerir. Bu URI, hata hakkında daha fazla bilgi almak için kullanılabilir. Bu parametre, **error_description** ile birlikte verilebilir. **error_uri** parametresi de belirli karakterlerden oluşmalıdır ve URI referansı biçimine uygun olmalıdır.
4. **state** (Zorunlu, eğer istemci isteği sırasında **state** parametresi kullanılmışsa): Eğer istemci, yetkilendirme isteği sırasında bir **state** parametresi göndermişse, yetkilendirme sunucusu bu değeri **state** parametresi olarak geri göndermelidir. Bu, istemcinin hangi isteğe karşılık gelen hatayı aldığını doğru şekilde eşleştirmesine yardımcı olur. **state** parametresi, istemcinin göndermiş olduğu orijinal değeri içerir.

Hata Yanıtı Örneği

Eğer bir hata meydana gelirse, yetkilendirme sunucusu aşağıdaki gibi bir yönlendirme yanıtı verebilir:

HTTP/1.1 302 Found
Location: https://client.example.com/cb?error=access_denied&state=xyz

Burada:

- **error=access_denied**: Bu, kaynağa erişim isteğinin reddedildiğini belirten hata kodudur.
- **state=xyz**: Eğer istemci, isteğinde bir **state** parametresi göndermişse, yetkilendirme sunucusu bu değeri geri gönderecektir.

Hata Durumlarının Kullanımı

Hata Parametreleri istemcinin doğru bir şekilde hatayı anlamasını sağlar ve hatayı nasıl ele alacağını belirlemesine yardımcı olur. Örneğin:

- Eğer **invalid_request** hatası alınırsa, istemci istek parametrelerini gözden geçirmelidir (eksik parametre, yanlış değer, çoklu parametre vb.).
- Eğer **access_denied** hatası alınırsa, kaynak sahibi ya da sunucu, isteği reddetmiştir. Bu durumda istemci, kullanıcıdan yeniden izin almayı deneyebilir.
- **server_error** ve **temporarily_unavailable** gibi hata durumları genellikle sunucu tarafındaki sorunlardan kaynaklanır ve istemciye bu durumda tekrar denemesi gerektiği bildirilebilir.

Sonuç

Error Response bölümü, yetkilendirme isteğinde hata olması durumunda istemciye nasıl bir yanıt verileceğini tanımlar. İstemciye hata kodu ve açıklaması, sorunları çözebilmesi için gerekli bilgileri sağlar. Bu, OAuth akışında güvenliği ve doğru işlem sırasını sağlamak için kritik bir adımdır.

***4.1.3. Access Token Request** (Erişim Token'ı İsteği) bölümü, istemcinin yetkilendirme kodunu alarak bir erişim token'ı almak için token endpoint'ine nasıl başvuracağını açıklar. Erişim token'ı, istemcinin kullanıcı adına bir kaynağa erişmesine izin verir ve genellikle bu işlem **authorization code flow** içinde yapılır. Aşağıda, bu isteğin nasıl yapılacağı ve hangi parametrelerin gerekli olduğu detaylı olarak açıklanmıştır.*

Erişim Token'ı İsteği için Gerekli Parametreler

Erişim token'ı almak için istemcinin, yetkilendirme kodu ile bir istek yapması gerekmektedir. Bu istek **"application/x-www-form-urlencoded"** formatında yapılır ve aşağıdaki parametreler içerir:

1. **grant_type** (Zorunlu): İstemci, erişim token'ı almak için hangi türde yetkilendirme istediğini belirtir. Bu parametre mutlaka **"authorization_code"** olarak ayarlanmalıdır, çünkü bu akışta istemci bir yetkilendirme kodu alarak erişim token'ı talep etmektedir.
2. **code** (Zorunlu): İstemcinin, yetkilendirme sunucusundan aldığı ve daha önce **authorization code flow** sırasında elde ettiği yetkilendirme kodunu belirtir. Bu kod, istemcinin token almak için yapacağı istekte kullanılır.
3. **redirect_uri** (Zorunlu): Eğer istemci, yetkilendirme isteği sırasında bir **redirect_uri** parametresi belirtmişse, bu parametre de bu istekte yer almalıdır. Bu değerin, ilk yetkilendirme isteğinde kullanılan **redirect_uri** ile tam olarak eşleşmesi gerekir. Bu, güvenlik için yapılan bir kontrol olup, istemcinin doğru bir URI'yi kullanmasını sağlar.

4. **client_id** (Zorunlu): Eğer istemci, **client authentication** yapmadan yetkilendirme kodunu almışsa, bu parametre de gerekli olabilir. Bu, istemcinin kimliğini doğrulamak için kullanılır.
5. **client authentication** (Zorunlu, Eğer Gizli İstemci): Eğer istemci gizli (confidential) bir istemci ise veya istemci kimlik doğrulaması gerektiriyorsa, istemci bu isteği gönderirken kimlik doğrulama yapmalıdır. Bu genellikle istemcinin bir API anahtarı veya başka bir kimlik doğrulama yöntemi ile yapılır. İstemci, kimlik doğrulaması için **Basic Authentication** gibi yöntemler kullanabilir. Kimlik doğrulaması başvurusu, istemcinin kimliğini doğrulamak için kullanılır.

   Örnek olarak, istemcinin yaptığı bir istek şu şekilde görünebilir:

   POST /token HTTP/1.1
   Host: server.example.com
   Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
   Content-Type: application/x-www-form-urlencoded

   grant_type=authorization_code&code=SplxlOBeZQQYbYS6WxSbIA
   &redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb

Yetkilendirme Sunucusunun Yapması Gerekenler

Yetkilendirme sunucusu, istemcinin başvurusunu işleme alırken aşağıdaki doğrulama ve kontrol işlemlerini yapmalıdır:

1. **Client Authentication Gereksinimi**: Eğer istemci gizli bir istemci (confidential client) ise ya da istemci kimlik doğrulama bilgilerine sahipse, yetkilendirme sunucusu istemcinin kimliğini doğrulamalıdır. Bu doğrulama, genellikle istemcinin kimlik doğrulama başlığı (örneğin **Basic Authentication**) kullanarak yapılır.
2. **Authorization Code Kontrolü**: Yetkilendirme sunucusu, başvuruyu yapan istemciye verilen yetkilendirme kodunun doğruluğunu kontrol etmelidir. Bu, verilen kodun geçerli olduğunu ve sunucu tarafından doğru bir şekilde oluşturulduğunu doğrular.
3. **Redirect URI Doğrulaması**: Eğer istemci, yetkilendirme isteği sırasında bir **redirect_uri** belirtmişse, yetkilendirme sunucusu bu URI'nin doğruluğunu ve geçerliliğini kontrol etmelidir. **redirect_uri** parametresi, ilk yetkilendirme isteğinde kullanılan **redirect_uri** ile tam olarak eşleşmelidir.
4. **Client ID Doğrulaması**: Yetkilendirme sunucusu, istemcinin kimliğini doğrulamalıdır. Eğer istemci gizli bir istemci ise ve kimlik doğrulaması yapılmışsa, yetkilendirme sunucusu istemcinin doğru kimliğe sahip olduğundan emin olmalıdır. Eğer istemci halka açık (public) bir istemci ise, sunucu, yetkilendirme kodunun doğru istemciye verildiğini doğrulamalıdır.

Sonuç

Bu işlem, istemcinin **yetkilendirme kodu** alarak **erişim token'ı** almasını sağlayan önemli bir adımdır. Yetkilendirme sunucusu, istemciden gelen talepleri doğrulamalı ve güvenli bir şekilde erişim token'larını sağlamalıdır. İstemcinin yaptığı istek, doğru parametrelerle yapılmalı ve güvenlik önlemleri (örneğin, **redirect_uri** doğrulaması, **client authentication** vb.) uygulanmalıdır. Bu sayede, istemci yalnızca yetkilendirilen ve geçerli bir token alır.

***4.1.4. Access Token Response** (Erişim Token'ı Yanıtı) bölümü, istemcinin **access token** almak için yaptığı başvurunun başarılı bir şekilde işlendiğinde yetkilendirme sunucusunun nasıl yanıt vereceğini açıklar. Bu yanıt, istemcinin güvenli bir şekilde kaynaklara erişebilmesi için gerekli olan **access token**'ı içerir. Eğer istemci kimlik doğrulaması hatalıysa veya geçersizse, bir hata yanıtı döner.*

Erişim Token'ı Yanıtı İçin Gerekli Parametreler

Başarılı bir erişim token'ı yanıtı şu parametreleri içerir:

1. **access_token** (Erişim Token'ı):
    - Zorunludur. Bu, istemcinin kimliğini doğrulayan ve kaynağa erişim sağlamak için kullanacağı **erişim token'ı**'dır.
    - Erişim token'ı, kaynak sunuculara yapılan isteklerde kullanılacak ve istemcinin kaynaklara erişmesine izin verecektir.
    - Bu token, genellikle bir **JWT** (JSON Web Token) ya da başka bir şifreli formatta olabilir.
2. **token_type** (Token Türü):
    - Zorunludur. Erişim token'ının türünü belirtir. Genellikle **"Bearer"** kullanılır, bu da istemcinin, token'ı "Bearer" başlığı altında HTTP isteklerinde göndermesi gerektiğini ifade eder.
    - Örneğin, **"Bearer"** token türü kullanıldığında, istemci HTTP isteklerinde şu şekilde bir başlık gönderir:

      ```
      Authorization: Bearer <access_token>
      ```

3. **expires_in** (Süresi Dolacak Zaman):
    - Zorunludur. Erişim token'ının geçerliliği için kalan süreyi saniye cinsinden belirtir. Genellikle **3600 saniye** (1 saat) gibi bir değer verilir.
    - Token süresi dolduğunda, istemci **refresh token** kullanarak yeni bir erişim token'ı alabilir (eğer **refresh_token** verilmişse).
4. **refresh_token** (Opsiyonel, Yenileme Token'ı):
    - Opsiyoneldir. Bu token, istemcinin erişim token'ı süresi dolduğunda yeni bir erişim token'ı alabilmesi için kullanılır.
    - **Refresh token** genellikle uzun süre geçerli olur ve kullanıcının yeniden giriş yapmasını engellemek amacıyla token yenileme işlemi için kullanılır. Eğer istemciye **refresh_token** verilirse, bu token başka bir istekte kullanılarak erişim token'ı yenilenebilir.
5. **example_parameter** (Opsiyonel, Örnek Parametre):
    - Opsiyoneldir. Yetkilendirme sunucusu, başka ek parametreler ekleyebilir. Bu parametre, örneğin belirli bir özellik ya da uygulamaya özel bir bilgi olabilir.

Örnek Başarılı Yanıt

Başarılı bir erişim token'ı yanıtı aşağıdaki gibi görünebilir:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
  "access_token": "2YotnFZFEjr1zCsicMWpAA",
  "token_type": "example",
  "expires_in": 3600,
  "refresh_token": "tGzv3JOkF0XG5Qx2TlKWIA",
  "example_parameter": "example_value"
}
```

Yanıtın Açıklamaları:

- **access_token**: Bu, istemcinin kullanacağı ve kaynaklara erişim sağlayacak token'dır. Örneğin, **2YotnFZFEjr1zCsicMWpAA**.
- **token_type**: Bu örnekte **"example"** olarak belirtilmiş. Gerçek dünyada, çoğu zaman **"Bearer"** olacaktır.
- **expires_in**: Bu token'ın 3600 saniye, yani 1 saat geçerli olduğu belirtilmiş.
- **refresh_token**: Eğer istemci, erişim token'ı süresi dolduğunda yeni bir token almak isterse, bu **refresh_token** kullanılabilir. Bu örnekte **tGzv3JOkF0XG5Qx2TlKWIA**.
- **example_parameter**: Bu parametre, yetkilendirme sunucusu tarafından eklenen bir ek parametre olabilir. Bu tür parametreler, belirli özelliklere ya da kullanıcılara özgü olabilir.

Hata Durumu

Eğer istemcinin isteği geçersizse veya kimlik doğrulaması başarısız olursa, yetkilendirme sunucusu bir **hata yanıtı** dönecektir. Bu hata yanıtında, hata tipi ve nedenini belirten **error** parametresi bulunur. Örneğin, geçersiz bir **client_id** veya hatalı bir **authorization code** verildiğinde hata döner.

Sonuç olarak, başarılı bir **Access Token Response** yanıtı, istemcinin gerekli erişim yetkilerine sahip olabilmesi için kritik öneme sahiptir. Erişim token'ı, istemcinin kaynağa erişmesine olanak tanırken, yenileme token'ı da uzun süreli oturumlar için gereklidir. Bu yanıt, istemci ile sunucu arasındaki güvenli ve sürekli veri erişiminin sağlanması için kullanılır.

4.2. Implicit Grant (Açık İzin Yöntemi) bölümü, **Access Token** almak için kullanılan bir OAuth 2.0 yetkilendirme akışını tanımlar. **Implicit Grant**, genellikle istemcilerin (genellikle JavaScript gibi istemci tarafı dillerini kullanan) tarayıcıda çalışan ve belirli bir redirection URI'ye sahip olan halka açık istemciler (public clients) için optimize edilmiştir. Bu akış, istemci kimlik doğrulaması gerektirmeyen bir yapıya sahiptir ve erişim token'ı doğrudan yetkilendirme isteğinin sonucu olarak elde edilir.

Bu yöntem, **Authorization Code Grant** akışından farklıdır. **Authorization Code Grant** akışında istemci, önce yetkilendirme kodunu alır, ardından bunu kullanarak bir erişim token'ı talep eder. Ancak **Implicit Grant**'te, istemci doğrudan yetkilendirme isteğiyle erişim token'ını alır.

Akışın Genel Özeti

1. **Client (İstemci)**, yetkilendirme sunucusuna bir erişim isteği gönderir. Bu isteği göndermek için istemci, yetkilendirme sunucusuna yönlendirdiği **resource owner** (kaynak sahibi) kullanıcısından **client identifier**, **requested scope**, **local state** ve bir **redirection URI** (geri yönlendirme URI) içerir. Bu URI, kaynak sahibi (kullanıcı) erişim iznini verdikten sonra kullanıcıyı geri yönlendirmek için kullanılır.
2. **Authorization Server (Yetkilendirme Sunucusu)**, kullanıcıyı kimlik doğrulamak için kullanıcı aracı (tarayıcı) üzerinden işlemi başlatır. Kullanıcı, istemcinin erişim isteğini onaylar veya reddeder.
3. Eğer kullanıcı erişimi onaylarsa, yetkilendirme sunucusu kullanıcıyı istemciye belirlenen redirection URI'sine geri yönlendirir. Bu URI, erişim token'ını **URI fragment** içinde içerir. **URI fragment**, URL'nin "?" ile başlayan sorgu parametrelerinden farklıdır. Bu parametreler genellikle URL'nin son kısmında yer alır ve kullanıcı aracısı tarafından işlenir.
4. Kullanıcı aracı, verilen redirection URI'yi takip eder ve **fragment**'i yerel olarak saklar. Ancak bu bilgi, kullanıcı aracı tarafından yalnızca tarayıcıda saklanır ve web sunucusuna gönderilmez.
5. Web barındırma istemcisi (örneğin, bir JavaScript uygulaması), verilen URI'den **fragment**'i çıkarır ve **access token**'ı (erişim token'ını) çıkarabilmek için ilgili scripti kullanarak bu veriyi alır.
6. Kullanıcı aracı, script tarafından çıkarılan **access token**'ı istemciye iletir.

Akışın Aşamaları

1. **(A) İstemci Başlangıç**:
    - İstemci, **resource owner** (kullanıcı) aracıyla yetkilendirme sunucusuna yönlendirme yapar. İstemci bu adımda **client identifier**, **requested scope** (istek kapsamı), **local state** (yerel durum) ve bir **redirection URI** gönderir.
2. **(B) Kimlik Doğrulama**:
    - Yetkilendirme sunucusu, kullanıcıyı kimlik doğrulamak için işlem yapar. Kullanıcı, istemcinin erişim talebini kabul eder veya reddeder.
3. **(C) Yetkilendirme Sunucusunun Yönlendirmesi**:

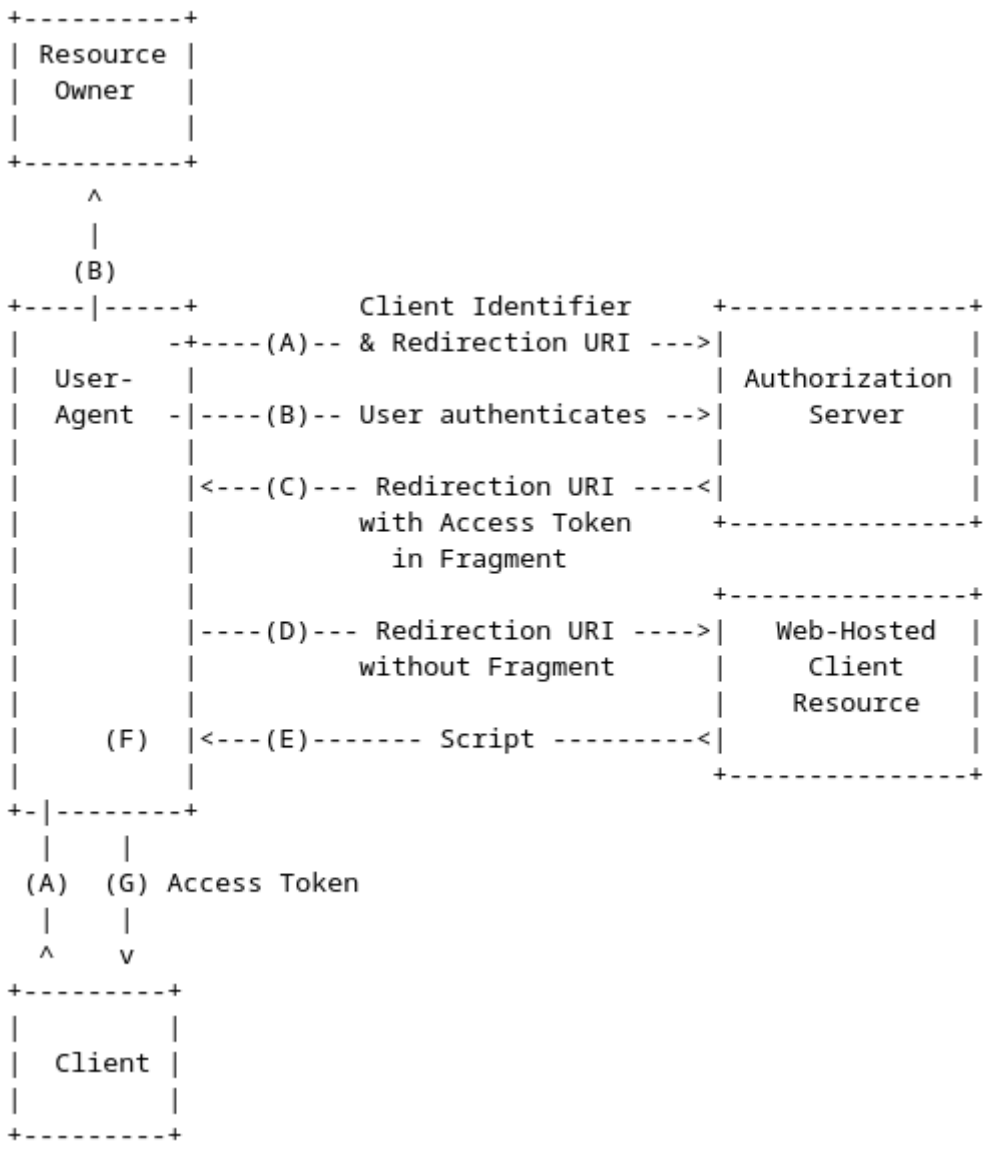
    - Kullanıcı erişimi onaylarsa, yetkilendirme sunucusu kullanıcıyı verilen **redirection URI**'ye yönlendirir. Bu URI, erişim token'ını **fragment** içinde içerir.
4. **(D) Yönlendirme ve Fragment Saklama**:
    - Kullanıcı aracı, **redirection URI**'yi takip eder ancak **fragment** bilgisini sadece yerel olarak saklar. Web sunucusu bu bilgiyle ilgili herhangi bir işlem yapmaz.
5. **(E) Web-Hosted Client Scripti**:
    - Web barındıran istemci, verilen **redirection URI**'yi işler. Web istemcisi, bu URI'deki **fragment**'i çıkarabilir ve gerekli parametreleri elde edebilir (örneğin, **access token**).
6. **(F) Scriptin Çalıştırılması**:
    - Kullanıcı aracı, scripti çalıştırarak **access token**'ı çıkarır.
7. **(G) Access Token'ın İstemciye Gönderilmesi**:
    - Kullanıcı aracı, çıkarılan **access token**'ı istemciye iletir.

Implicit Grant Yöntemiyle İlgili Önemli Noktalar

- **Refresh Token Yok**: Implicit Grant yöntemi, refresh token desteklemez. Erişim token'ı yalnızca tek bir kez kullanılır ve süresi dolduğunda yenilenmesi gerekebilir. Bu durumda, kullanıcı tekrar kimlik doğrulaması yaparak yeni bir token almalıdır.
- **Güvenlik Riskleri**: Erişim token'ı, URI fragmentinde açıkça yer alır ve bu da güvenlik riski oluşturabilir. Token, kullanıcının tarayıcısında ya da başka uygulamalarla paylaşılabilir. Bu nedenle, **Implicit Grant** yöntemi, güvenliği çok daha kritik olan işlemler için önerilmez. Bunun yerine, **Authorization Code Grant** yöntemi daha güvenli bir alternatif olabilir.
- **Public Clients için Uygun**: Bu akış genellikle, istemcinin kimlik doğrulaması yapmadığı ve istemcinin yalnızca kullanıcının tarayıcısında çalışan, güvenli olmayan ortamlar için uygundur.

Akışın Şeması

Aşağıda **Implicit Grant Flow**'unun adımları daha görsel bir şekilde gösterilmiştir:



Bu görseldeki adımlar sırasıyla:

(A) İstemcinin Başlatması

İstemci, yetkilendirme akışını başlatmak için **resource owner**'ı (kaynak sahibi kullanıcı) yetkilendirme sunucusunun **authorization endpoint**'ine (yetkilendirme noktası) yönlendirir. Bu aşamada, istemci aşağıdaki bilgileri içerir:

- **Client Identifier (İstemci Kimliği)**: İstemcinin kimliğini tanımlar. Bu, genellikle istemcinin kayıtlı olduğu benzersiz bir değerdir.
- **Requested Scope (İstenen Kapsam)**: İstemcinin erişmek istediği kaynakların veya izinlerin listesi. Örneğin, kullanıcı bilgilerine erişim gibi.
- **Local State (Yerel Durum)**: Bu, istemcinin talebine dair geçici bir durum bilgisidir ve CSRF (Cross-Site Request Forgery) saldırılarına karşı koruma sağlamak için kullanılır.

- **Redirection URI (Yönlendirme URI'si)**: Yetkilendirme sunucusunun, kullanıcıyı başarılı bir şekilde yetkilendirdikten sonra geri yönlendireceği URI. Bu, istemcinin doğru kaynağa yönlendirilmesini sağlamak için gereklidir.

Bu aşama, istemcinin kullanıcıyı yetkilendirme sunucusuna yönlendirdiği ilk adımdır. Yönlendirme URI'si, daha sonra kullanıcı başarılı bir şekilde giriş yaparsa erişim token'ı ile birlikte dönecek olan yerdir.

(B) Yetkilendirme Sunucusunun Kimlik Doğrulaması

Yetkilendirme sunucusu, kullanıcının kimliğini doğrulamak için **user-agent** (genellikle bir web tarayıcısı) üzerinden işlem yapar. Bu adımda:

- Kullanıcı, yetkilendirme sunucusuna girerek, istemcinin talep ettiği erişim izni ile ilgili bir karar verir.
- Kullanıcı, istemcinin erişim talebini **onaylar** veya **reddeder**. Bu aşama, kullanıcının istemcinin hangi verilere erişebileceği konusunda karar verdiği aşamadır.

Eğer kullanıcı **erişim iznini reddederse**, yetkilendirme sunucusu istemciyi hatalı bir yanıtla (örneğin, hata kodu) yönlendirir.

(C) Yetkilendirme Sunucusunun Yönlendirmesi

Eğer kullanıcı, istemcinin erişim talebini **onaylarsa**, yetkilendirme sunucusu, kullanıcıyı tekrar istemciye yönlendirecektir. Bu yönlendirme şu şekilde gerçekleşir:

- Yetkilendirme sunucusu, **redirection URI**'yi kullanarak kullanıcıyı geri yönlendirir.
- Bu URI, **access token**'ı **fragment** içinde içerir. Fragment kısmı, URL'nin sonundaki # ile başlar ve tarayıcıda görünür ama sunucuya gönderilmez.
-

Bu akış, istemcinin kimlik doğrulaması yapmadan, doğrudan tarayıcıda çalışan istemciler için geçerli olan ve hızlıca erişim sağlayan bir mekanizma sağlar.

Bu aşama, erişim token'ının doğrudan kullanıcı aracı (tarayıcı) aracılığıyla istemciye iletilmesinin sağlandığı adımdır.

(D) Kullanıcı Aracısının Yönlendirmeyi Takip Etmesi

Kullanıcı, yönlendirme işlemini takip ederek istemciye geri gelir. Ancak:

- **Fragment** bilgisi (yani access token), URL'nin bir parçası olarak kullanıcı aracısında (tarayıcıda) saklanır ve **web sunucusuna gönderilmez**.
- Yönlendirme, istemcinin web kaynağına yapılır (örneğin, HTML sayfasına), ancak **fragment** URL'si burada sunucuya ulaşmaz.

Tarayıcı, bu fragment'i yerel olarak saklar ve daha sonraki adımlarda bu bilgiyi kullanabilir.

(E) Web-Hosted Client Resource'ın Yanıtı

Web barındırma istemcisi (örneğin, bir JavaScript uygulaması), verilen **redirection URI**'yi işler ve kullanıcının tarayıcısında saklanan fragment'i çıkararak içeriklere erişir:

- İstemci, **redirection URI**'yi tamamlar ve **fragment**'i çıkarır.
- Bu işlem genellikle bir **HTML sayfası** içinde yer alan bir script ile yapılır. Script, URL'deki fragment'i alır ve içindeki **access token**'ı (ve diğer parametreleri) çıkarır.

(F) Scriptin Çalıştırılması

Web-Hosted Client (örneğin, JavaScript uygulaması) tarafından sağlanan **script** (kod parçası) kullanılarak:

- **Access token**'ı, URL'den çıkarılır ve istemciye sağlanır.
- Kullanıcı aracı (tarayıcı), token'ı alır ve uygun işlemi yapmak üzere istemciye iletir.

Burada, scriptin kullanımı, token'ı almanın ve istemciye aktarmanın tek yoludur. Bu işlem, tamamen istemcinin JavaScript kodu içinde gerçekleşir.

(G) Access Token'ın İstemciye İletilmesi

Son adımda, kullanıcı aracı, çıkarılan **access token**'ı istemciye iletir. Bu token, istemcinin veri isteklerinde kimlik doğrulaması yapmak için kullanılır.

- İstemci, **access token**'ı aldıktan sonra, bu token'ı API çağrıları gibi işlemlerle kullanarak kaynağa erişim sağlamak için kullanabilir.
- Token, genellikle HTTP başlıkları içinde **Authorization** parametresi olarak iletilir:

Genel Özellikler

- **Implicit Grant** akışında, istemci ve kullanıcı arasında direkt bir etkileşim olduğu için, token'lar doğrudan kullanıcı aracısı aracılığıyla iletilir. Bu nedenle, güvenlik riskleri ve token sızıntısı gibi konularda dikkatli olunması gerekir.
- Bu akış genellikle **public clients** (örneğin, JavaScript tabanlı uygulamalar veya istemciler) için uygundur. Güvenli olmayan istemcilerde kullanılması, token'ların kolayca ele geçirilmesine neden olabilir.

Bu adımlar, OAuth 2.0 **Implicit Grant Flow** sürecinin nasıl işlediğini açık bir şekilde gösterir ve her bir adımda neler olduğunu anlamanızı sağlar.

4.2.1 Authorization Request (Yetkilendirme İsteği)

İstemci, yetkilendirme sunucusuna **access token** almak için bir yetkilendirme isteği gönderir. Bu istek, kullanıcıyı yetkilendirme sunucusuna yönlendirmek amacıyla bir URL'yi içerir. İstemci, **authorization request**'i başlatırken aşağıdaki parametreleri kullanarak yetkilendirme sunucusuna bir istek yapar.

Gerekli Parametreler

1. **response_type**

- **Zorunlu** bir parametredir. Bu parametre, istemcinin ne tür bir yanıt beklediğini belirtir. Implicit Grant flow için bu değer **"token"** olmalıdır, çünkü istemci doğrudan **access token** almak istiyor.
- Bu parametre **"token"** olarak ayarlanmışsa, istemci access token'ı doğrudan alacaktır. Bu, **Authorization Code Grant** türünde olmayan, daha basit ve daha hızlı bir erişim yöntemidir.

2. **client_id**
    - **Zorunlu** bir parametredir. Bu, istemcinin benzersiz kimliğini tanımlar. Genellikle istemci kayıt olduğunda verilen bir değerdir ve istemcinin, yetkilendirme sunucusunun tanıdığı uygulama olduğunu belirtir.
    - Örneğin, bir mobil uygulama veya bir web sitesi için istemci ID’si sağlanır.
3. **redirect_uri**
    - **Opsiyonel** bir parametredir. Bu parametre, yetkilendirme sunucusunun access token'ı gönderdiği URI’yi belirtir. Eğer bu parametre verilirse, yetkilendirme sunucusu yalnızca verilen URI'ye yönlendirme yapacaktır.
    - Eğer istemci, kaydederken belirli bir yönlendirme URI'si belirtilmişse, bu URI'nin gelen istekle uyuşması gerektiği için, **redirect_uri** parametresi doğrulanacaktır.
    - Eğer bu parametre sağlanmazsa, yetkilendirme sunucusu kaydedilen ilk URI'yi kullanacaktır.
4. **scope**
    - **Opsiyonel** bir parametredir. Bu, istemcinin erişmek istediği kaynakları veya izinleri tanımlar. Örneğin, kullanıcı bilgileri veya uygulama verileri gibi.
    - Bu parametre, istemcinin talep ettiği izinlerin genişliğini belirler. Eğer belirtilmezse, varsayılan bir kapsam kullanılabilir.
5. **state**
    - **Önerilen** bir parametredir. Bu parametre, istemcinin request ve callback (geri dönüş) arasında bir bağ kurmasına yardımcı olur. Özellikle, **Cross-Site Request Forgery (CSRF)** saldırılarına karşı koruma sağlamak için kullanılır.
    - **state** değeri, istemcinin başlattığı isteği tanımlayan ve yalnızca istemci tarafından bilinen bir değerdir. Bu parametreyi kullanarak, istemci istekleri ve geri dönüşleri eşleştirebilir. Yetkilendirme sunucusu, **state** parametresini de geri gönderir, böylece istemci, isteği ve yanıtı doğrulayabilir.

İstek Yapma (İstemci Tarafı)

Bu parametreler kullanılarak istemci, kullanıcının tarayıcısına yönlendirilir. Örneğin, istemci şu şekilde bir HTTP isteği gönderir:

GET /authorize?response_type=token&client_id=s6BhdRkqt3&state=xyz &redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb HTTP/1.1 Host: server.example.com

Bu örnek, istemcinin yetkilendirme sunucusuna bir istek gönderdiğini gösterir. Burada:

- **response_type=token**: Erişim token'ı talep ediliyor.
- **client_id=s6BhdRkqt3**: İstemci kimliği.
- **state=xyz**: CSRF saldırılarına karşı koruma için istemcinin gönderdiği rastgele bir değer.
- **redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb**: Yönlendirme URI'si, yani başarılı bir yetkilendirmeden sonra kullanıcıyı geri gönderecek adres.

Yetkilendirme Sunucusunun İstek Doğrulaması

Yetkilendirme sunucusu, istemciden gelen bu isteği doğrular. **Doğrulama işlemi** şu adımları içerir:

1. **Tüm gerekli parametrelerin mevcut olup olmadığı kontrol edilir.**
2. **redirect_uri**: Yetkilendirme sunucusu, istemcinin kayıtlı olduğu yönlendirme URI'si ile gelen yönlendirme URI'sini karşılaştırır. Eğer bu iki URI birbirine uyuyorsa, istek geçerli kabul edilir. Eğer uyuşmazlık varsa, istek reddedilir.
3. **Diğer parametrelerin geçerliliği**: İstemcinin kimliği (client_id), yönlendirme URI'si ve scope gibi parametreler de doğrulanır.

Yetkilendirme Kararı ve Yönlendirme

Eğer istek geçerliyse ve istemci doğru parametrelerle başvurmuşsa, **yetkilendirme sunucusu**, kullanıcıyı yönlendirecek bir karar alır:

1. **Kullanıcıdan izin alma**: Yetkilendirme sunucusu, kullanıcının kimliğini doğrular ve kullanıcının erişim talebini onaylamasını ister.

2. **Erişim İzni**: Eğer kullanıcı erişimi onaylarsa, yetkilendirme sunucusu, kullanıcının tarayıcısını istenen **redirect_uri**'ye yönlendirir. Bu yönlendirme, URL'nin parçası olarak **access token** içerecek şekilde yapılır.

Bu süreç, istemcinin ve kullanıcının etkileşimi sırasında, istemcinin doğru erişim izinlerine sahip olup olmadığı ve kullanıcı tarafından onay alınıp alınmadığına bağlı olarak gerçekleşir.

Özet

Bu süreç, istemcinin yetkilendirme sunucusuna **access token** almak amacıyla yönlendirme yapması ve kullanıcıdan onay alması için kullanılan bir akıştır. Bu tür bir akış, genellikle **public clients** (örneğin, JavaScript tabanlı uygulamalar) için kullanılır çünkü istemci kimliği ve gizli anahtar gibi güvenlik bilgileri istemcinin tarafında saklanmaz.

4.2.2. Access Token Response

Yetkilendirme sunucusu, eğer kaynak sahibi (kullanıcı) erişim talebini onaylarsa, istemciye bir **access token** (erişim token'ı) gönderir. Bu token, istemcinin, belirli kaynaklara erişim sağlamasına olanak tanır. Bu token, **redirection URI**'nin **fragment** bölümüne eklenir ve istemciye gönderilir.

İşte bu yanıtın içerdiği parametreler ve açıklamaları:

Yanıt Parametreleri

1. **access_token**
    - **Zorunlu** bir parametredir. Yetkilendirme sunucusu, istemciye **erişim token'ı** verir. Bu token, istemcinin, kaynağa erişmesini sağlar.
    - **Erişim token'ı**, istemcinin sahip olduğu izinlere göre kaynaklara erişim sağlayan bir anahtar gibi çalışır. Token'ın içeriği genellikle şifreli veya imzalı olabilir.
2. **token_type**
    - **Zorunlu** bir parametredir. Bu, verilen token türünü belirtir. Bu değer genellikle **"bearer"** olur.
    - **"Bearer token"** kullanımı yaygındır, yani bu token'ı taşıyan her şey, ona erişim hakkına sahiptir. Örneğin, token'ı taşıyan bir istek, yetkilendirilmiş kabul edilir.
3. **expires_in**
    - **Önerilen** bir parametredir. Erişim token'ının **geçerlilik süresi** belirtilir. Bu parametre, token'ın ne kadar süreyle geçerli olduğunu belirler.
    - Örneğin, **"expires_in": 3600** değeri, token'ın 1 saat boyunca geçerli olacağı anlamına gelir.
    - Eğer bu parametre belirtilmezse, yetkilendirme sunucusu token'ın geçerliliği ile ilgili başka bir yöntem sağlayabilir veya varsayılan bir değer kullanabilir.
4. **scope**
    - **Opsiyonel** bir parametredir. Eğer erişim token'ı istemcinin talep ettiği kapsamla aynıysa, bu parametre gereksiz olabilir.
    - Ancak, eğer **access token** verilen kapsam, istemcinin talep ettiği kapsamla farklıysa, bu parametre **zorunlu** hale gelir.
    - Kapsam, token'ın hangi kaynaklara erişim sağladığını belirtir. Eğer kapsamla ilgili bir değişiklik varsa, bunu belirten parametre eklenir.
5. **state**
    - **Zorunlu** bir parametredir, ancak yalnızca istemci **state** parametresini talep ettiyse. Eğer istemci yetkilendirme isteğinde **state** parametresi göndermişse, yetkilendirme sunucusu bu değeri **geri gönderir**.
    - **State** parametresi, istemcinin yetkilendirme sürecinde tutmak istediği ve geri alması gereken bir değerdir. Bu, CSRF (Cross-Site Request Forgery) saldırılarını engellemeye yardımcı olur. İstemci, **state** parametresi ile talep ve yanıtları eşleştirerek güvenliği artırır.

Yönlendirme ve HTTP Yanıtı

Erişim token'ı yanıtı, yetkilendirme sunucusu tarafından kullanıcının tarayıcısına yönlendirme yapılacak şekilde gönderilir. Bu yanıt bir **HTTP 302 Found** yanıtı ile yapılır. Bu, istemciyi yeni bir URI'ye yönlendiren geçici bir HTTP durum kodudur. Yönlendirme, **Location** başlığı aracılığıyla yapılır ve erişim token'ı, **URI fragment** (örn., `#access_token=...`) içinde bulunur.

Örnek yanıt:

```
HTTP/1.1 302 Found
Location: http://example.com/cb#access_token=2YotnFZFEjr1zCsicMWpAA
             &state=xyz&token_type=example&expires_in=3600
```

Bu örnekte, yetkilendirme sunucusu kullanıcının tarayıcısını, istemcinin belirlediği yönlendirme URI'sine (bu örnekte **http://example.com/cb**) yönlendirir ve URL fragment'ine **access_token** ve diğer parametreleri ekler. Bu şekilde istemci, token'ı URI'den alır.

Önemli Notlar:

- **Fragment ve HTTP Yönlendirme**: Bazı tarayıcılar (user-agent'lar), **Location** başlığında fragment içeren bir yönlendirme yapmayı desteklemez. Bu durumda, istemciye başka yöntemlerle yönlendirme yapılması gerekebilir. Örneğin, istemci, HTML bir sayfa döndürebilir ve içinde yönlendirme yapılacak bir **'continue'** butonu ile kullanıcıyı yönlendirebilir.
- **Yanıt Parametreleri**: İstemci, tanımadığı parametreleri görmezden gelmelidir. Yalnızca **access_token**, **token_type**, **expires_in**, **scope**, ve **state** parametreleri ile ilgilenmelidir.
- **Access Token'ın Boyutu**: Token'ın boyutuna dair bir sınırlama yoktur, ancak istemci, token'ın boyutunun ne kadar olacağını bilmemelidir. Yetkilendirme sunucusu, token'ların boyutlarına dair bilgileri dökümante edebilir.

Özet

Access Token Response (Erişim Token'ı Yanıtı) sürecinde, yetkilendirme sunucusu, istemciye **access token**'ı, ilgili parametrelerle birlikte bir yönlendirme URI'si üzerinden iletir. Bu token, istemcinin kaynaklara erişim sağlamasına olanak tanır. Yanıt, ayrıca **token_type**, **expires_in**, **state** gibi parametreleri içerir.

4.2.2.1. Error Response

Error Response (Hata Yanıtı)

Hata yanıtı, istemcinin isteğiyle ilgili bir hata meydana geldiğinde yetkilendirme sunucusu tarafından gönderilir. İstemciden gelen yetkilendirme isteği geçerli değilse ya da kaynak sahibi isteği reddederse, yetkilendirme sunucusu, istemciyi geçerli olmayan yönlendirme URI'sine göndermemeli ve hatalı yanıtı **fragment** bölümüne ekleyerek istemciye iletmelidir.

Hata yanıtı, şu şekilde bir **HTTP 302 Found** yanıtı aracılığıyla yapılır ve yönlendirme URI'sinin fragment kısmında hata bilgileri bulunur. Bu fragment kısmı, istemcinin hatayı anlayabilmesi için gerekli olan bilgileri içerir.

Örnek hata yanıtı:

```
HTTP/1.1 302 Found
Location: https://client.example.com/cb#error=access_denied&state=xyz
```

Bu örnekte, yetkilendirme sunucusu, istemciyi yönlendirme URI'sine gönderir (https://client.example.com/cb) ve hata bilgilerini **fragment** kısmında belirtir. Burada, hata kodu **access_denied** ve istemcinin gönderdiği **state** parametresi bulunmaktadır.

Hata Yanıtı Parametreleri

1. **error** (Zorunlu)
    - Bu parametre, hata durumunu tanımlar ve aşağıdaki hata kodlarından birini içerir:
    - **invalid_request**: İstek, eksik bir parametre içeriyor, geçersiz bir parametre değeri var, bir parametre birden fazla kez gönderilmiş veya istek yanlış formatta. Yani istemci isteği düzgün oluşturamamış.
    - **unauthorized_client**: İstemci, bu yöntemle erişim token'ı talep etme yetkisine sahip değil.
    - **access_denied**: Kaynak sahibi (kullanıcı) ya da yetkilendirme sunucusu, talebi reddetti.
    - **unsupported_response_type**: Yetkilendirme sunucusu, bu yöntemle erişim token'ı elde etmeyi desteklemiyor.
    - **invalid_scope**: Talep edilen kapsam geçersiz, bilinmiyor veya yanlış formatta.
    - **server_error**: Yetkilendirme sunucusu, beklenmeyen bir hata ile karşılaştı ve isteği yerine getiremedi. Bu hata kodu, sunucunun 500 Internal Server Error döndürmesinin engellendiği durumlarda gereklidir.
    - **temporarily_unavailable**: Yetkilendirme sunucusu, geçici bir aşırı yüklenme veya bakım nedeniyle isteği yerine getiremiyor. Bu, genellikle 503 Service Unavailable HTTP durum kodunun yerine kullanılacak hata kodudur.

    Önemli Not: **error** parametresi, sadece ASCII karakterlerden oluşmalıdır. Yani, karakterler **%x20-21**, **%x23-5B**, **%x5D-7E** arasında olmalıdır.

2. **error_description** (Opsiyonel)
    - İnsan tarafından okunabilir bir hata açıklaması sunar. Bu açıklama, hata hakkında daha fazla bilgi verir ve istemci geliştiricisinin hatayı anlamasına yardımcı olur.
    - Hata açıklamaları yalnızca ASCII karakterlerinden oluşmalıdır.
    - **error_description** parametresi genellikle istemciye hatanın daha detaylı bir açıklamasını sağlar, örneğin: "The redirect URI is not valid".
3. **error_uri** (Opsiyonel)
    - Hata hakkında daha fazla bilgi sağlayan bir **web URI**'si belirtir. Bu URI, istemci geliştiricisine hata hakkında ek bilgi veya açıklamalar sunar.
    - **error_uri** parametresi, bir **URI-reference** formatında olmalı ve yalnızca geçerli URI karakterlerini içermelidir.
4. **state** (Zorunlu)
    - Eğer istemci **state** parametresini yetkilendirme isteğinde belirtmişse, yetkilendirme sunucusu aynı değeri geri göndermelidir.
    - Bu parametre, istemcinin hata yanıtını doğru şekilde eşleştirmesini sağlar. **State** değeri, istemcinin yetkilendirme işlemi sırasında tutmak istediği özgün bir değeri içerir.

Hata Yanıtının iletilmesi

Yetkilendirme sunucusu, yukarıda belirtilen hata parametreleri ile birlikte bir **302 Found** HTTP yanıtı gönderir. Yanıtta, **Location** başlığında hata parametrelerinin bulunduğu yönlendirme URI'si verilir. Bu URI'de, **error** parametresi ile birlikte hata açıklamaları (varsa) ve hata URI'si de yer alır. Örneğin:

```
HTTP/1.1 302 Found
Location: https://client.example.com/cb#error=access_denied&state=xyz
```

Önemli Notlar:

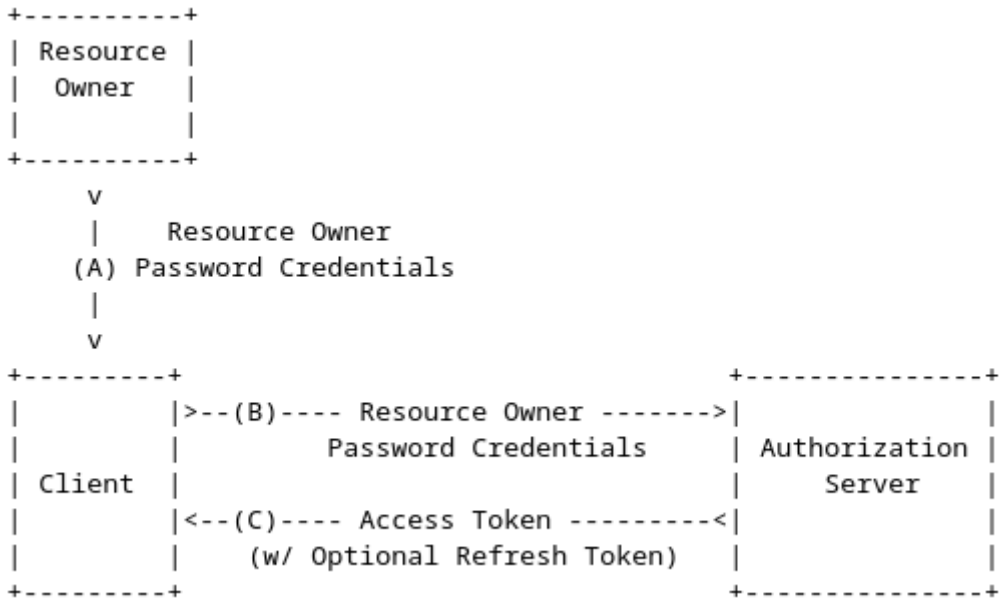
- **Yönlendirme URI'sinin Geçerliliği**: Yetkilendirme sunucusu, **redirection URI**'sinin geçersiz ya da hatalı olduğu durumlarda, otomatik olarak hatalı URI'ye yönlendirme yapmamalıdır. Bunun yerine, hata bilgileri içeren bir yanıt gönderir.
- **Hata Yanıtlarının Yapısı**: Hata yanıtlarında gönderilen parametreler ve URL fragment'inin doğru şekilde işlenmesi gerekir. Bu parametreler, istemcinin hatayı anlamasına ve durumu düzeltmesine yardımcı olacak bilgi sağlar.

Özet

Hata yanıtı, istemciden gelen geçersiz veya reddedilen yetkilendirme taleplerine karşılık yetkilendirme sunucusunun gönderdiği yanıttır. Bu yanıt, **error**, **error_description**, **error_uri**, ve **state** parametrelerini içerebilir ve istemcinin hata durumunu doğru bir şekilde işlemesi için gerekli bilgileri sağlar.

4.3. Resource Owner Password Credentials Grant (Kaynak Sahibi Şifre Kimlik Bilgisi Yetkilendirme Akışı), OAuth 2.0 protokolünün bir yetkilendirme türüdür ve belirli durumlarda kullanılır. Bu akış, kaynak sahibinin (kullanıcının) doğrudan uygulama ile güven ilişkisi kurduğu senaryolarda kullanılır, örneğin cihaz işletim sistemleri veya yüksek yetkili uygulamalar. Bu akış, kullanıcının kullanıcı adı ve şifresini (genellikle etkileşimli bir form aracılığıyla) uygulamaya vererek doğrulama yapmasına olanak tanır. Bununla birlikte, bu akış yalnızca diğer yetkilendirme akışlarının uygulanabilir olmadığı durumlarda kullanılmalıdır, çünkü güvenlik riskleri içerebilir.

Kaynak Sahibi Şifre Kimlik Bilgisi Yetkilendirme Akışı (Password Credentials Grant)



Akışın Detayları:

1. **(A) Kaynak Sahibi (User)**, istemciye kullanıcı adı ve şifresini sağlar. Burada, kullanıcı şifre bilgilerini doğrudan istemci uygulamasına verir.
2. **(B) İstemci**, kaynak sahibinden aldığı kullanıcı adı ve şifreyi, yetkilendirme sunucusunun **token endpoint**'ine gönderir. Bu aşamada istemci, yetkilendirme sunucusu ile kimlik doğrulaması yapar. İstemci, genellikle **client_id** ve **client_secret** gibi bilgileri kullanarak yetkilendirme sunucusuna kimlik doğrulama bilgilerini iletir.
3. **(C) Yetkilendirme Sunucusu**, istemcinin kimlik bilgilerini doğrular ve kaynak sahibinin (kullanıcının) sağladığı şifre bilgilerini de kontrol eder. Eğer kullanıcı adı ve şifre doğruysa, yetkilendirme sunucusu bir **access token** (erişim token'ı) ve opsiyonel olarak bir **refresh token** (yenileme token'ı) oluşturur ve istemciye gönderir.

Şekil 5: Kaynak Sahibi Şifre Kimlik Bilgisi Akışı

Bu akış, kaynak sahibinin kullanıcı adı ve şifreyi doğrudan istemciye vererek, istemcinin yetkilendirme sunucusundan bir erişim token'ı almasını sağlar. Bu akış, genellikle aşağıdaki senaryolarda kullanılır:

- **Güvenli İstemciler İçin**: Kaynak sahibi ve istemci arasında güven ilişkisi olduğunda, örneğin bir cihazın işletim sistemi veya güvenli bir uygulama.
- **Mevcut Kimlik Doğrulama Yöntemlerinin OAuth'a Taşınması**: Bu akış, mevcut kimlik doğrulama yöntemlerini OAuth 2.0 protokolüne uyarlamak için de kullanılabilir. Örneğin, HTTP Basic veya Digest kimlik doğrulaması kullanan eski sistemlerden OAuth'a geçiş yapılabilir.

Dikkat Edilmesi Gerekenler:

- **Güvenlik Riski**: Kaynak sahibinin kullanıcı adı ve şifresini doğrudan istemciye vermesi gerektiği için, bu akış çok dikkatli kullanılmalıdır. Kaynak sahibinin şifre bilgileri istemcinin kontrolüne geçer, bu nedenle güvenliğin sağlanması önemlidir. Ayrıca, istemcinin bu bilgileri güvenli bir şekilde iletmesi ve saklaması gerekir.
- **Alternatif Akışlar**: Bu akış, diğer akışlar (örneğin, Authorization Code Grant veya Implicit Grant) uygun olmadığında kullanılmalıdır. Diğer akışlar daha güvenlidir, çünkü kullanıcı adı ve şifre doğrudan istemci ile paylaşılmaz.

Örnek Durum:

Bir mobil uygulama, kullanıcı adı ve şifre bilgilerini alıp yetkilendirme sunucusuna göndererek bir erişim token'ı alır. Bu token, uygulamanın kullanıcının adına API'lere erişmesini sağlar.

Akışın Kullanıldığı Durumlar:

- **Herkesin Güvendiği Uygulamalar**: Cihazlar veya yüksek güvenlik gereksinimleri olan uygulamalar, örneğin bir mobil uygulama ya da masaüstü uygulaması, bu akışı kullanabilir.
- **Eski Yöntemlerin OAuth'a Entegre Edilmesi**: Mevcut kimlik doğrulama yöntemlerinin OAuth ile entegre edilmesi gerektiğinde, bu akış kullanılabilir.

Sonuç:

Kaynak Sahibi Şifre Kimlik Bilgisi Yetkilendirme Akışı, güvenli ve güvenilir bir ortamda kullanılmalıdır. Yalnızca diğer yöntemlerin geçerli olmadığı durumlar için önerilir. Bu akışın kullanılabilmesi için istemci ve kaynak sahibi arasında güvenli bir ilişki olması gerekir.

4.3.1. Yetkilendirme İsteği ve Yanıtı

İstemcinin kaynak sahibi kimlik bilgilerini elde etme yöntemi, bu spesifikasyonun kapsamı dışındadır. İstemci, bir erişim token'ı alındıktan sonra kimlik bilgilerini imha etmelidir.

4.3.2. Erişim Token'ı İsteği

İstemci, "application/x-www-form-urlencoded" formatında, UTF-8 karakter kodlaması ile HTTP istek gövdesine ekleyerek token uç noktasına aşağıdaki parametreleri içeren bir istek gönderir:

- **grant_type**: **ZORUNLU**. Değeri "password" olarak ayarlanmalıdır.
- **username**: **ZORUNLU**. Kaynak sahibinin kullanıcı adı.
- **password**: **ZORUNLU**. Kaynak sahibinin şifresi.
- **scope**: **İSTEĞE BAĞLI**. Erişim isteğinin kapsamı, Bölüm 3.3'te açıklandığı gibi.

Eğer istemci türü gizli (confidential) ise ya da istemciye kimlik doğrulama gereksinimleri verilmişse (veya istemci kimlik bilgileri verilmişse), istemci, **Bölüm 3.2.1**'de açıklandığı şekilde yetkilendirme sunucusuyla kimlik doğrulaması yapmalıdır.

Örneğin, istemci aşağıdaki HTTP isteğini (sadece görsel amaçlı satır sonları eklenmiş) TLS (Transport Layer Security) üzerinden gönderir:

```
POST /token HTTP/1.1
Host: server.example.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=password&username=johndoe&password=A3ddj3w
```

Yetkilendirme sunucusu şu adımları atmalıdır:

- Gizli istemciler için ya da kimlik bilgisi verilen istemciler (veya başka kimlik doğrulama gereksinimleri olanlar) için istemci kimlik doğrulamasını zorunlu tutmalıdır.
- İstemci kimlik doğrulaması yapılmışsa, istemciyi doğrulamalıdır.
- Kaynak sahibi şifre bilgilerini mevcut şifre doğrulama algoritmasıyla doğrulamalıdır.

Bu erişim token'ı isteği, kaynak sahibinin şifresini kullandığı için, yetkilendirme sunucusunun uç noktayı brute force saldırılarına karşı koruması gerekmektedir (örneğin, hız sınırlaması kullanmak veya uyarılar üretmek gibi önlemler alarak).

4.3.3. Erişim Token'ı Yanıtı

Eğer erişim token'ı isteği geçerli ve yetkilendirilmişse, yetkilendirme sunucusu, **Bölüm 5.1**'de açıklandığı gibi bir erişim token'ı ve isteğe bağlı bir yenileme token'ı (refresh token) verir. Eğer istek istemci kimlik doğrulamasını başaramazsa ya da geçersizse, yetkilendirme sunucusu **Bölüm 5.2**'de açıklandığı gibi bir hata yanıtı döndürür.

Başarılı bir yanıt örneği:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
    "access_token": "2YotnFZFEjr1zCsicMWpAA",
    "token_type": "example",
    "expires_in": 3600,
    "refresh_token": "tGzv3JOkF0XG5Qx2TlKWIA",
    "example_parameter": "example_value"
}
```

Bu yanıtın içerdiği parametreler:

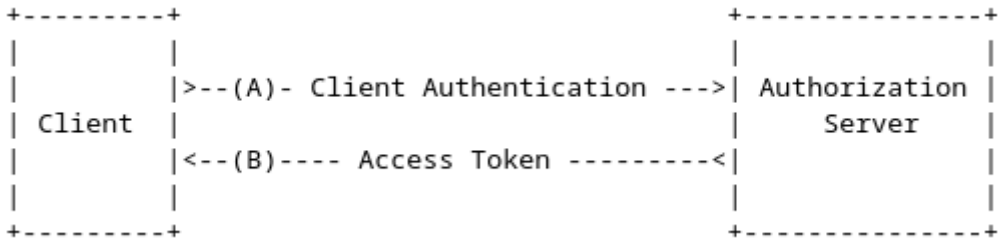
- **access_token**: Erişim token'ı, istemcinin kaynaklara erişmesini sağlayacak olan anahtar.
- **token_type**: Token türü, burada "example" değeri bir örnektir, uygulamada genellikle "bearer" kullanılır.
- **expires_in**: Erişim token'ının geçerlilik süresi, saniye cinsinden belirtilir (bu örnekte 3600 saniye = 1 saat).
- **refresh_token**: İsteğe bağlı bir yenileme token'ı, erişim token'ı süresi dolduğunda yeni bir erişim token'ı almak için kullanılabilir.
- **example_parameter**: Yanıtla birlikte döndürülen ek bir parametre, bu genellikle yetkilendirme sunucusu tarafından eklenen özel bir parametre olabilir.

Eğer istek başarılıysa, istemci bu bilgileri alır ve uygun şekilde erişim sağlamak için kullanabilir.

4.4. Client Credentials Grant

Client Credentials Grant türü, istemcinin sadece kendi kimlik bilgilerini kullanarak erişim token'ı talep ettiği bir OAuth2.0 akışıdır. Bu akış, istemcinin **kendi kontrolündeki** korunan kaynaklara veya **daha önce yetkilendirme sunucusuyla düzenlenmiş** başka bir kaynak sahibinin kaynaklarına erişmek için kullanılır. Bu türde, istemci kimlik doğrulaması yaparak doğrudan kaynaklarına erişim sağlamak amacıyla bir erişim token'ı talep eder.

Bu grant tipi yalnızca **gizli istemciler (confidential clients)** tarafından kullanılabilir. Gizli istemciler, istemci kimlik bilgilerini güvenli bir şekilde saklayabilen ve kimlik doğrulamasını güvenli bir şekilde yapabilen uygulamalardır.



Akışın adımları (Şekil 6'da gösterildiği gibi):

1. **(A) İstemci Kimlik Doğrulaması:**
   İstemci, yetkilendirme sunucusu ile kimliğini doğrular ve token endpoint'ine bir erişim token'ı talebi gönderir. Bu aşamada, istemci kimlik bilgilerini kullanarak sunucuya başvurur.
2. **(B) Yetkilendirme Sunucusunun Kimlik Doğrulaması:**
   Yetkilendirme sunucusu, istemcinin kimlik bilgilerini doğrular. Eğer kimlik doğrulaması geçerli ise, sunucu istemciye bir erişim token'ı (access token) gönderir.

Bu flow, genellikle arka planda çalışan uygulamalar veya istemcilerin kendi kaynaklarına erişim sağlamak için kullanılır. Örneğin, bir API'nin yalnızca uygulama tarafından erişilmesi gerektiğinde, istemci kimlik bilgileriyle yapılan bu akış uygun olur.

Özetle:

- İstemci, yalnızca kendi kimlik bilgileriyle yetkilendirme sunucusuna başvurur.
- Yetkilendirme sunucusu, istemcinin kimlik bilgilerini doğrular ve geçerli ise bir erişim token'ı gönderir.

4.4.1. Yetkilendirme Talebi ve Yanıtı

İstemci kimlik doğrulaması, yetkilendirme grant'ı (izin verme) olarak kullanıldığından, ek bir yetkilendirme talebine gerek yoktur.

4.4.2. Erişim Belirteci Talebi

İstemci, aşağıdaki parametreleri "application/x-www-form-urlencoded" formatında, Ek B'ye göre ve HTTP isteği varlık gövdesinde UTF-8 karakter kodlaması kullanarak token uç noktasına bir talep gönderir:

- **grant_type**
  ZORUNLU. Değer "client_credentials" olarak ayarlanmalıdır.
- **scope**
  İSTEĞE BAĞLI. Erişim talebinin kapsamı, Bölüm 3.3'te açıklandığı gibi.

İstemci, Bölüm 3.2.1'de açıklandığı şekilde yetkilendirme sunucusuyla kimlik doğrulaması yapmalıdır.

Örneğin, istemci aşağıdaki HTTP isteğini iletir (sadece gösterim amacıyla ekstra satır aralıkları eklenmiştir):

```
POST /token HTTP/1.1
Host: server.example.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials
```

Yetkilendirme sunucusu, istemciyi kimlik doğrulamalıdır.

4.4.3. Erişim Belirteci Yanıtı

Erişim belirteci talebi geçerli ve yetkilendirilmişse, yetkilendirme sunucusu, Bölüm 5.1'de açıklandığı gibi bir erişim belirteci verir. Bir yenileme belirteci **DAHİL EDİLMEMELİDİR**. Eğer talep, istemci kimlik doğrulaması hatası nedeniyle başarısız olduysa veya geçersizse, yetkilendirme sunucusu, Bölüm 5.2'de açıklandığı gibi bir hata yanıtı döndürür.

Başarılı bir yanıt örneği:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
	"access_token":"2YotnFZFEjr1zCsicMWpAA",
	"token_type":"example",
	"expires_in":3600,
	"example_parameter":"example_value"
}
```

4.5. Extension grant (uzantı yetkilendirme türü), OAuth 2.0'ın standart yetkilendirme türlerinden farklı bir şekilde çalışır. Bu tür, istemcinin yetkilendirme sunucusuna özel bir yetkilendirme türünü belirtmesini sağlar. Uzantı grant türleri, belirli bir işlevselliği sağlamak için genişletilebilir ve bu türlerin nasıl kullanılacağı, yetkilendirme sunucusunun tanımladığı mutlak bir URI (Uniform Resource Identifier) ile belirlenir.

Adımlar ve Açıklama:

1. **Grant Type Belirleme**:
   İstemci, token talep ederken, "grant_type" parametresini kullanarak uzantı türünü belirtir. Bu tür, yetkilendirme sunucusu tarafından tanımlanmış bir URI ile belirlenir. Yani, her uzantı türü için yetkilendirme sunucusu tarafından tanımlanan özel bir URI kullanılır. Örneğin, **SAML 2.0** kullanarak erişim belirteci almak için bir URI belirlenmiştir ve bu URI, istemci tarafından kullanılır.

2. **Erişim Belirteci Talebi**:
   İstemci, bu grant türünü kullanarak erişim belirteci almak için bir HTTP isteği gönderir. Bu istekte, grant türünü (örn. `grant_type=urn:ietf:params:oauth:grant-type:saml2-bearer`) ve gereken diğer parametreleri (örneğin, SAML beyanı) ekler.

3. **Geçerli ve Yetkilendirilmiş İstek**:
   Eğer istemcinin gönderdiği istek geçerli ve yetkilendirilmişse, yetkilendirme sunucusu, istemciye bir erişim belirteci ve isteğe bağlı bir yenileme belirteci gönderir.

4. **Hatalı veya Geçersiz İstek**:
   Eğer istek geçersizse veya istemci kimlik doğrulaması başarısız olursa, yetkilendirme sunucusu hata yanıtı döndürür.

Örnek:

Buradaki örnekte, istemci **SAML 2.0** kullanarak bir erişim belirteci almak istiyor. Bunun için, aşağıdaki gibi bir HTTP isteği gönderiyor:

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Asaml2-
bearer&assertion=PEFzc2VydGlvbiBJc3N1ZUluc3RhbnQ9IjIwMTEtMDU
```

Burada, `grant_type` parametresi belirli bir uzantıyı (`saml2-bearer`) belirtiyor ve `assertion` parametresi ise SAML beyanını taşıyor.

Eğer bu istek başarılı olursa, yetkilendirme sunucusu erişim belirteci ve yenileme belirteci gönderir. Eğer geçersizse, hata mesajı döner.

Bu, OAuth 2.0'ın esnekliğinden faydalanarak, özel yetkilendirme türlerinin eklenmesine olanak tanır.