

# anfix OS - API - OAuth

- Introducción:
- Alta como desarrollador tecnológico:
- Flujo de validación:
  - Diagrama de flujo:
  - Peticiones:
    - Petición request\_token:
      - A: Petición request\_token:
      - B: Respuesta request\_token:
    - Petición login - authorize:
      - C: Petición login:
      - D: Petición redirigida al desarrollador:
    - Petición access\_token:
      - E: Petición access\_token:
      - F: Respuesta request\_token:
    - Petición invalidate\_token:
      - Petición invalidate\_token:
      - Respuesta invalidate\_token:
    - Petición API Anfix:
      - Petición POST estándar:
      - Petición GET:

## Introducción:

Ahora mismo para poder hacer uso de la API de Anfix es necesario ser usuario de Anfix, con su respectivo User y Password. Solamente se puede acceder a la API mediante el escritorio de Anfix, lo cual impide a otros desarrolladores poder hacer uso de la API y crear aplicaciones que puedan ser utilizadas por los usuarios de Anfix. Estas aplicaciones de terceros complementan o mejoran los servicios de los que Anfix dispone, dotando al sistema de una mayor variedad de funcionalidad. Normalmente dicho desarrollador no será usuario de Anfix, mientras que un usuario de Anfix que desee utilizar la aplicación del desarrollador no desea dar su usuario y contraseña a un tercero con todos los peligros que eso conlleva. Para poder autorizar a un tercero a extraer datos privados en Anfix u otras API's, sin que el usuario entregue su usuario y contraseña, aparece el protocolo OAuth.

El protocolo OAuth consiste en que mediante una serie de validaciones, un desarrollador de aplicaciones, pueda obtener acceso a los datos de un usuario de Anfix, con permiso de este, sin que el usuario de Anfix le entregue su Email y contraseña. Este acceso será indefinido hasta que se deniegue el acceso a los datos.

## Alta como desarrollador tecnológico:

Antes de lanzar el proceso de obtención de credenciales finales de un desarrollador tecnológico, este debe de estar dado de alta como tal en Anfix. Para poder darse de alta como desarrollador este debe ponerse en contacto con Anfix y solicitar dicha alta. Dentro de la aplicación del CRM se ha desarrollado el CRUD que gestiona el alta y baja de desarrolladores. Una vez solicitadas las credenciales Anfix, se le hará entrega de un par de claves que solo el desarrollador y Anfix conocen.

Estas claves son las siguientes:

Nombre	Descripción
<b>Consumer-Key</b>	Se trata de un identificador único de desarrollador que será necesario utilizar en el proceso de obtención de credenciales y en las peticiones a la API
<b>Shared-Secret</b>	Clave secreta, que será utilizada para poder firmar las distintas peticiones a la API de Anfix. Como veremos este será uno de los campos fijos en la construcción de la firma. Otros campos de la firma varían su valor según la parte del flujo donde nos encontremos.

Con estas dos claves, el desarrollador ya puede comenzar el desarrollo de su aplicación y dar servicio a los usuarios de Anfix que los deseen. Las dos credenciales son cadenas alfanuméricas de 32 caracteres.

Ejemplo:

```
consumer_key = 898FCCEB3181586F8CBC8A8541711111  
shared_secret = 2F1E06E87CB7FBE7934AB947B0011111
```

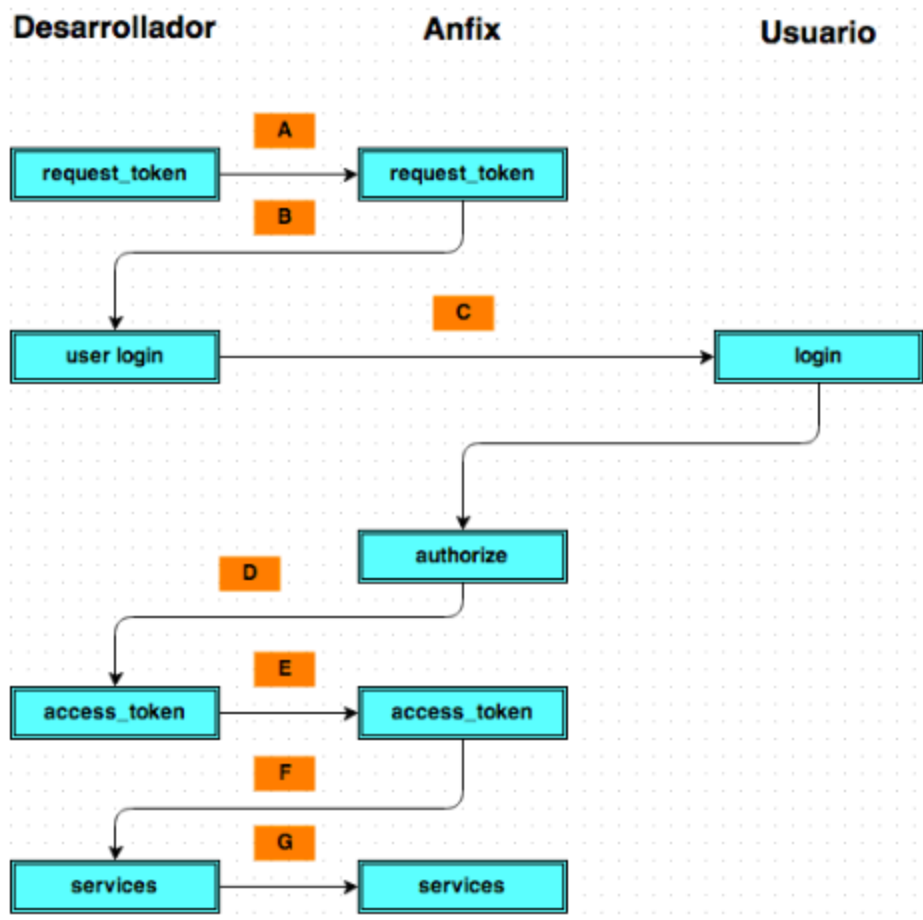
## Flujo de validación:

Para poder acceder a los datos de un usuario que este dispuesto a utilizar la aplicación del desarrollador es necesario realizar una serie de validaciones para obtener unas credenciales finales que den acceso a los datos. Esta validaciones se basan en el protocolo OAuth y a continuación se mostrará el flujo de peticiones que se deben de realizar para obtenerlas:

El partner deberá de configurar su aplicación para hacer uso del protocolo OAuth implementado por Anfix utilizando una forma de firma, en éste caso "PLAINTEXT". Se utilizará PLAINTEXT debido a su sencillez de construcción y a que todas la peticiones que se realizan en Anfix van sobre HTTPS, todos los datos van cifrados.

## Diagrama de flujo:

A continuación se muestra el flujo de peticiones que se ejecutan para la obtención de la credenciales finales y una tabla donde indican los distintos datos que se intercambian entre el desarrollador y Anfix. Mas adelante se detalla cada uno de los pasos y la información que deben intercambiar. Como se puede apreciar en el flujo, hay una parte de él donde interviene el usuario de anfix. En ese punto es donde el propio usuario permitirá al desarrollador que acceda a sus datos sin que este tenga conocimiento de su email y contraseña. El desarrollador no tiene acceso al email y contraseña del usuario porque este se loguea directamente en anfix y si es correcto anfix redirige el flujo hacia el desarrollador asignándole nuevos tokens para que pueda seguir con el flujo indicado en la figura.



La tabla siguiente muestra los distintos datos intercambiados entre el desarrollador y anfix para que este pueda llevar a cabo correctamente la obtención de credenciales:

A	B	C	D	E	
request	response	request	response	request	
realm	oauth_token	oauth_token	oauth_token	realm	
oauth_consumer_key	oauth_token_secret		oauth_verifier	oauth_consumer_key	
oauth_signature_method	oauth_callback_confirmed			oauth_token	
oauth_signature				oauth_signature_method	
oauth_callback				oauth_signature	
				oauth_verifier	

### Peticiones:

A continuación se describe el flujo de peticiones que el desarrollador tecnológico debe implementar para poder acceder a la API de Anfix y con ello a los datos del usuario en concreto. Todas las peticiones a Anfix van sobre **HTTPS** y son de tipo **POST**, exceptuando la descarga de ficheros que será un **GET** como veremos mas adelante.

## Petición request\_token:

La primera petición que debe realizar un desarrollador tal y como indica el diagrama es una petición **request\_token**, donde se validarán los datos proporcionados por el desarrollador y si todo es correcto se le asignarán dos token temporales necesarios para el flujo de validación. El sistema asignará un token y token secret temporales en la cabecera de respuesta que serán usados en las siguientes peticiones. Estos dos token temporales sólo tienen una **duración de 1 hora**, es decir, si un usuario no concede permisos al desarrollador (introduce su login y password) en ese periodo de tiempo la credencial caducará y deberá volver a solicitar otra nuevamente.

La petición **request\_token** se compone de dos partes:

- **URL:** [https://apps.anfix.com/os/os/parc/partner/request\\_token](https://apps.anfix.com/os/os/parc/partner/request_token)
- **Header:** **Authorization:** realm="facturas",

```
oauth_consumer_key="898FCCEB3181586F8CBC8A8541711111",  
  oauth_signature_method="PLAINTEXT",  
  oauth_callback="http%3A%2F%2Fprintfact.com%2Fready",  
  oauth_signature="2F1E06E87CB7FBE7934AB947B0011111&"
```

La URL indica a donde debe de realizar la petición el desarrollador tecnológico. Como antes indicamos debe de ser una petición tipo **POST** y utilizar el protocolo **HTTPS**.

Además, la petición tipo POST debe incluir una cabecera de autenticación llamada **Authorization**, que se compone de los siguientes campos obligatorios separados por comas:

Parámetro	Tipo	Descripción	Obligatorio
realm	Cadena	Texto libre que indica el tipo de servicio que pretende dar el desarrollador, para el ejemplo "facturas".	✓
oauth_consumer_key	Cadena	Consumer_key que Anfix proporcionó al desarrollador cuando este se dio de alta como tal en Anfix.	✓
oauth_signature_method	Cadena	Método de firma utilizado para validar la petición. En Anfix utilizamos <b>PLAINTEXT</b> ya que todas nuestras peticiones se hacen sobre un canal seguro como HTTPS	✓
oauth_signature	Cadena	Firma de la petición. Ver apartado de construcción de firma para el request_token.	✓
oauth_callback	Cadena	Dirección de reenvío donde quiere el desarrollador que le enviemos la respuesta una vez validado el usuario. Paso D del diagrama de flujo.	✓

Es **importante** resaltar que los nombres de la cabecera **Authorization**, **realm**, **oauth\_consumer\_key**, **oauth\_signature\_method**, **oauth\_signature** y **oauth\_callback** sean exactamente los que aquí se indican para evitar problemas de campos no encontrados. Esto podría provocar que el servicio responda con que no puede validar al desarrollador al no encontrar estos campos, siendo estos obligatorios.

**Construcción de la firma para request\_token:** La firma para la petición de request\_token es una cadena de texto y se construye concatenando los siguientes valores: "**Shared-Secret + &**", se compone del **Shared-Secret** que le proporcionamos al desarrollador cuando se dio de alta como desarrollador tecnológico + el símbolo **&**. Como veremos mas adelante, esta petición no lleva token\_secret como lo llevarán las siguientes, esto es debido a que Anfix no le ha asignado ningún token\_secret al desarrollador al tratarse de la primera petición de credenciales.

Un detalle muy importante es que la dirección de callback (**oauth\_callback**) debe de tratarse antes de ser enviada como parámetro en la cabecera de **Authorization**:

Una URL introducida en un navegador podría incluir caracteres especiales que deben ser convertidos internamente antes de transmitirlos. Cualquier código que genere el formato UTF-8 puede considerar las URL con caracteres UTF-8 como direcciones válidas, aunque internamente deba hacer una transformación antes de enviarlas a un servidor web. Este proceso se conoce como codificación URL. Esta conversión de caracteres es necesario hacerla para cumplir la sintaxis definida en la especificación sobre identificadores uniformes de recursos (Uniform Resource Identifier) de la W3C. Todo esto quiere decir que para que una URL sea válida, sólo puede incluir los subconjuntos de caracteres ASCII que se incluyen en la siguiente tabla:

Conjunto	Caracteres	Utilización en URL
Alfanumérico	a b c d e f g h i j k l m n o p q r s t u v w x y z A B C D E F G H I J K L M N O P Q R S T U V W X Y Z 0 1 2 3 4 5 6 7 8 9	Cadenas de texto, utilización de esquemas (http), puertos (8080), etc.
No reservado	- _ . ~	Cadenas de texto
Reservado	! * ' ( ) ; : @ & = + \$ , / ? % # [ ]	Caracteres de control o cadenas de texto




Al crear una URL válida, debemos asegurarnos de que sólo incluya caracteres de los mostrados anteriormente. Se debe realizar la conversión de caracteres no incluidos en la tabla por otros que sí estén. Por convención popular, los espacios (no permitidos en las URL) se suelen representar con el signo '+'.

Cuando queremos representar caracteres reservados como literales, es necesario codificarlos. Por ejemplo, ? Se utiliza para indicar el inicio de una cadena de consulta, si deseamos utilizar una cadena que incluya este carácter, como por ejemplo "a donde vamos?", es necesario codificar el carácter '?'.

Todos los caracteres con codificación URL se codifican utilizando '%' seguido del valor hexadecimal de dos caracteres correspondientes al carácter UTF-8 que se quiere codificar. Por ejemplo: los dos puntos ':' se sustituyen por '%3a', el punto y coma ';' por '%3b', etc.

Por ejemplo: los dos puntos ':' se sustituyen por '%3A', la barra '/' por '%2F', etc.

Parámetros de salida en la respuesta:

Parámetro	Tipo	Descripción	Obligatorio
<b>oauth_token</b>	Cadena	Token temporal asignado a la petición de credenciales temporales, tiene una validez de una hora	
<b>oauth_token_secret</b>	Cadena	Token secreto temporal asignado a la petición de credenciales temporales, tiene una validez de una hora	
<b>oauth_callback_confirmed</b>	Booleano	Indica que se ha recibido o no una dirección de redirección	

Una vez lanzada la petición el servidor de Anfix validará el consumer\_key y la firma, pudiendo devolver los siguientes resultados en el **JSON** de salida.

- Si en el JSON de salida contiene el campo resultado **"result":0**, significa que no hay errores de autenticación, devolviendo en la cabecera de respuesta los campos **oauth\_token**, **oauth\_token\_secret** y **oauth\_callback\_confirmed**. Estos valores deben de ser guardados por el desarrollador para utilizarlos en las siguientes peticiones.

```
{
  "result": 0
}
```

- Si se han producido errores en el JSON el campo resultado **"result":1**, nos mostrará la lista de errores encontrados en la validación o servicio de obtención de credenciales. Ejemplo:

```
{
  "result": 1,
  "errorList": [{
    "code": "ERR000061000",
    "fieldId": "OauthCustomerKey",
    "text": "No existe un Partner Tecnológico para el
customer_key."
  }]
}
```

Ejemplo:

**A: Petición request\_token:**

```
https://apps.anfix.com/os/os/parc/partner/request_token
POST /os/os/parc/partner/request_token HTTP/1.1
Host: anfix.com
Authorization: realm="facturas",
  oauth_consumer_key="898FCCEB3181586F8CBC8A8541711111",
  oauth_signature_method="PLAINTEXT",
  oauth_callback="http%3A%2F%2Fprintfact.com%2Fready",
  oauth_signature="2F1E06E87CB7FBE7934AB947B0011111&"
```

#### B: Respuesta request\_token:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
oauth_token: 898FCCEB3181586F8CBC8A85417AFB76
oauth_token_secret: 2F1E06E87CB7FBE7934AB947B0017E84
oauth_callback_confirmed: true
```


#### JSON salida:

```
{
  "result":0
}
```

#### Petición login - authorize:

La segunda petición que debe realizar un desarrollador tal y como indica el diagrama es una petición de **login**. Esta petición tiene el parámetro `oauth_token` en la URL que contiene el valor del `oauth_token` devuelto en la respuesta de `request_token` y que el desarrollador guardo. Al ejecutar esta petición lo que se consigue es abrir la página de login de anfix donde el usuario debe de permitir al desarrollador el acceso a sus datos. Por lo tanto el usuario debe de introducir su email y contraseña si desea dar permiso al desarrollador para acceder a sus datos. Si el usuario introduce sus datos, el sistema enviará una petición **authorize** al servidor de Anfix junto con el parámetro `oauth_token` que ha recibido en la URL. El servidor de Anfix procesará dicha petición comprobando el usuario, contraseña y token. Si todo es correcto, Anfix enviará una petición al desarrollador a la dirección de callback que indicó en el `request_token` con los parámetros `oauth_token` y `oauth_verifier` en la URL. El desarrollador debe de guardar estos parámetros para utilizarlos en la posterior petición.

Parámetros de entrada:

Parámetro	Tipo	Descripción	Obligatorio
<code>oauth_token</code>	Cadena	Token, coincide con el <code>oauth_token</code> recibido en la respuesta del <b>request_token</b> , pero que debe ir en la URL de la petición <b>login-authorize</b>	

Parámetros de salida:

Parámetro	Tipo	Descripción	Obligatorio
-----------	------	-------------	-------------

<b>oauth_token</b>	Cadena	Token temporal, se corresponde con el oauth_token recibido en la respuesta del <b>request_token</b> , se devuelve como parámetro en la URL	✓
<b>oauth_verifier</b>	Cadena	Código de verificación, nuevo código que deberá ser utilizado en la siguiente petición de <b>access_token</b> , se devuelve como parámetro en la URL	✓

La petición **login** se compone únicamente de una URL con un parámetro **oauth\_token** como hemos comentado antes. Esta petición no contiene cabecera de Authorization ya que su cometido es que el usuario de Anfix de permiso al desarrollador para poder acceder a sus datos.

- **URL:** [https://anfix.com/login-partner/?oauth\\_token=898FCCEB3181586F8CBC8A85417AFB76](https://anfix.com/login-partner/?oauth_token=898FCCEB3181586F8CBC8A85417AFB76)

Esta petición abrirá la ventana de **login** donde el usuario debe meter su email y contraseña para permitir al desarrollador el acceso a sus datos. Como antes indicamos debe de ser una petición tipo POST y utilizar el protocolo HTTPS.

Una vez lanzada la petición el servidor de Anfix, se abrirá la ventana de **login**, el usuario introducirá su email y contraseña y enviará la petición de **authorize**. Si el proceso ha ido correctamente el servidor de Anfix redirige una petición a la dirección de **callback** (<https://printfact.com/ready>) que envió el desarrollador con los parámetros de **oauth\_token** y **oauth\_verifier**. Estos parámetros los debe guardar el desarrollador para poder seguir con el último paso de validación.

- **URL de redirección:** [https://printfact.com/ready?oauth\\_token=898FCCEB3181586F8CBC8A85417AFB76&oauth\\_verifier=6040917A076FEE1796B37145394543CD](https://printfact.com/ready?oauth_token=898FCCEB3181586F8CBC8A85417AFB76&oauth_verifier=6040917A076FEE1796B37145394543CD)

Ejemplo:

### C: Petición login:

```
https://anfix.com/login-partner/?oauth_token=898FCCEB3181586F8CBC8A85417AFB76
```

### D: Petición redirigida al desarrollador:

```
https://printfact.com/ready?oauth_token=898FCCEB3181586F8CBC8A85417AFB76&oauth_verifier=6040917A076FEE1796B37145394543CD
```

### Petición access\_token:

La última petición para obtener las **credenciales finales** es la petición **access\_token** como se puede ver en el diagrama de flujo. En esta petición se asigna el **oauth\_token** y **oauth\_token\_secret finales** y que serán necesarios para poder realizar cualquier petición de datos a la API de Anfix. Estas credenciales no caducan hasta que un usuario deniegue el acceso a sus datos mediante la petición **invalidate\_token** que será detallada posteriormente. La petición **access\_token** se compone de dos partes como la **request\_token**, pero añadiendo los campos de validación en la cabecera de Authorization llamados **oauth\_token** y **oauth\_verifier**. También se ha de tener en cuenta que los valores de la firma y **oauth\_token** varían en función de los valores devueltos en las peticiones anteriores y que el desarrollador debió de guardar.

La petición **access\_token** se compone de dos partes:

- **URL:** [https://apps.anfix.com/os/os/parc/partner/access\\_token](https://apps.anfix.com/os/os/parc/partner/access_token)
- **Header:** **Authorization:** realm = "facturas",  
oauth\_consumer\_key = "898FCCEB3181586F8CBC8A854171111",  
oauth\_signature\_method = "PLAINTEXT",  
oauth\_verifier = "6040917A076FEE1796B37145394543CD",  
oauth\_token = "898FCCEB3181586F8CBC8A85417AFB76",  
oauth\_signature = "2F1E06E87CB7FBE7934AB947B001111&2F1E06E87CB7FBE7934AB947B0017E84"

La URL indica a donde debe de realizar la petición el desarrollador tecnológico. Como antes indicamos debe de ser una petición tipo POST y



utilizar el protocolo HTTPS.

Además de la URL, la petición tipo POST debe incluir una cabecera de autenticación llamada **Authorization**, que se compone de los siguientes campos obligatorios:

Parámetro	Tipo	Descripción
<b>realm</b>	Cadena	Texto libre que indica el tipo de servicio que pretende dar el desarrollador, para el ejemplo "facturas".
<b>oauth_consumer_key</b>	Cadena	Consumer_key que Anfix proporcionó al desarrollador cuando este se dio de alta como tal en Anfix.
<b>oauth_signature_method</b>	Cadena	Método de firma utilizado para validar la petición. En Anfix utilizamos <b>PLAINTEXT</b> ya que todas nuestras peticiones se hacen seguro como HTTPS
<b>oauth_signature</b>	Cadena	Firma de la petición. Ver apartado de construcción de firma para el access_token.
<b>oauth_token</b>	Cadena	Token que Anfix devolvió al desarrollador como parámetro después del login del usuario: https://printfact.com/ready?oauth_token=3181586F8CBC8A85417AFB76&oauth_verifier=6040917A076FEE1796B37145394543CD
<b>oauth_verifier</b>	Cadena	Código de verificación (oauth_verifier) que Anfix devolvió al desarrollador como parámetro después del login del usuario: eady?oauth_token=898FCCEB3181586F8CBC8A85417AFB76&oauth_verifier=6040917A076FEE1796B37145394543CD

Construcción de la **firma para access\_token**: La firma para la petición de access\_token es una cadena y se construye de la siguiente manera: "**Shared-Secret + & + oauth\_token\_secret**", se compone del **Shared-Secret** que le proporcionamos al desarrollador cuando se dio de alta como desarrollador tecnológico + el símbolo **& + oauth\_token\_secret** que Anfix devolvió en la respuesta a la petición request\_token. Para que la firma sea válida debe de ir en el orden propuesto, sino será desechada la petición por firma inválida.

Parámetros de salida en la respuesta:

Parámetro	Tipo	Descripción	Obligatorio
<b>oauth_token</b>	Cadena	Token final asignado al desarrollador, se debe guardar en lugar seguro para la utilización posterior en las peticiones a la API	
<b>oauth_token_secret</b>	Cadena	Token secreto final asignado al desarrollador, se debe guardar en lugar seguro para la utilización posterior en las peticiones a la API	

Una vez lanzada la petición el servidor de Anfix validará el consumer\_key, la firma, token y código de verificación para ver si son correctos. El servidor de Anfix responderá con los siguientes resultados en el **JSON** de salida.

- Si en el JSON de salida contiene el campo **"result":0** significa que no ha y errores de autenticación, devolviendo en la cabecera de respuesta los campos **oauth\_token**, **oauth\_token\_secret**. Estos valores son las credenciales finales que serán necesarias para poder acceder a los datos de usuario. Estos valores deben de ser guardados por el desarrollador en un lugar seguro para utilizarlos en todas las peticiones a la API de Anfix.

```
{
  "result": 0
}
```

- Si se han producido errores en el JSON el campo **"result":1** y nos mostrará la lista de errores encontrados en la validación. Ejemplo:



```
{
  "result": 1,
  "errorList": [{
    "code": "ERR000061000",
    "fieldId": "OAuthCustomerKey",
    "text": "No existe un Partner Tecnológico para el
customer_key."
  }]
}
```

Si el resultado es 0 y la petición nos devuelve los valores **oauth\_token**, **oauth\_token\_secret** definitivos, ya estamos preparados para poder lanzar peticiones en nombre del usuario que nos ha dado su permiso a la API de Anfix.

Ejemplo:

#### E: Petición access\_token:

```
https://apps.anfix.com/os/os/parc/partner/access_token
POST /os/os/parc/partner/access_token HTTP/1.1
Host: anfix.com
Authorization: realm="facturas",
  oauth_consumer_key="898FCCEB3181586F8CBC8A8541711111",
  oauth_signature_method="PLAINTEXT",
  oauth_token="898FCCEB3181586F8CBC8A85417AFB76",
  oauth_verifier="6040917A076FEE1796B37145394543CD",

oauth_signature="2F1E06E87CB7FBE7934AB947B0011111&2F1E06E87CB7FBE7934AB9
47B0017E84"
```

#### F: Respuesta reques\_token:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
oauth_token: D224E32F8BDAC58AD8C0972EAB489C71
oauth_token_secret: 39DC4913075810FA895CB7ACCF5D7869
```

#### Petición invalidate\_token:

Existe una petición fuera del flujo de la obtención de credenciales que trata el caso de que un usuario ya no desee que un desarrollador pueda acceder a sus datos y deniegue las peticiones futuras a la API de Anfix.

La petición **invalidate\_token** se compone de dos partes:

- **URL:** [https://apps.anfix.com/os/os/parc/partner/invalidate\\_token](https://apps.anfix.com/os/os/parc/partner/invalidate_token)
- **Header:** **Authorization:** realm="facturas",  
oauth\_consumer\_key="898FCCEB3181586F8CBC8A8541711111",  
oauth\_signature\_method="PLAINTEXT",  
oauth\_token="D224E32F8BDAC58AD8C0972EAB489C71",  
oauth\_signature="2F1E06E87CB7FBE7934AB947B0011111&39DC4913075810FA895CB7ACCF5D7869"

La URL indica a donde debe de realizar la petición el desarrollador tecnológico. Como antes indicamos debe de ser una petición tipo **POST** y utilizar el protocolo **HTTPS**.

Además de la URL, la petición tipo POST debe incluir una cabecera de autenticación llamada **Authorization**, que se compone de los siguientes campos obligatorios:

Parámetro	Tipo	Descripción	Obligatorio
realm	Cadena	Texto que indica el tipo de servicio que pretende dar el desarrollador, para el ejemplo "facturas".	✓
oauth_consumer_key	Cadena	Consumer_key que Anfix proporcionó al desarrollador cuando este se dio de alta como tal en Anfix.	✓
oauth_signature_method	Cadena	Método de firma utilizado para validar la petición. En Anfix utilizamos <b>PLAINTEXT</b> ya que todas nuestras peticiones se hacen sobre un canal seguro como HTTPS	✓
oauth_signature	Cadena	Firma de la petición. Ver apartado de construcción de firma para el <b>invalidate_token</b> .	✓
oauth_token	Cadena	Token <b>final</b> que Anfix devolvió al desarrollador en la cabecera de respuesta de la petición <b>access_token</b>	✓

Construcción de la **firma** para **invalidate\_token**: La firma para la petición de invalidate\_token es una cadena y se construye de la siguiente manera: "**Shared-Secret + & + oauth\_token\_secret** ", se compone del **Shared-Secret** que le proporcionamos al desarrollador cuando se dio de alta como desarrollador tecnológico + el símbolo **& + oauth\_token\_secret** que Anfix devolvió en la respuesta a la petición **access\_token**, se trata del token\_secret definitivo. Para que la firma sea válida debe de ir en el orden propuesto, sino será desechada la petición por firma inválida.

Una vez lanzada la petición el servidor de Anfix validará el consumer\_key, la firma y token definitivo para ver si son correctos. El servidor de Anfix responderá con los siguientes resultados en el **JSON** de salida.

- Si en el JSON de salida contiene el campo "**result**":0 significa que no ha y errores de autenticación. Esto significa que el proceso de denegación de permiso ha sido ejecutado correctamente.

```
{
  "result": 0
}
```

- Si se han producido errores en el JSON el campo "**result**":1 y nos mostrará la lista de errores encontrados en la validación. Ejemplo:

```
{
  "result": 1,
  "errorList": [{
    "code": "ERR000061000",
    "fieldId": "OAuthCustomerKey",
    "text": "No existe un Partner Tecnológico para el
customer_key."
  }]
}
```

Si es resultado es 0, las siguientes peticiones que se realicen a la API de Anfix en nombre de ese usuario ya no podrán acceder a sus datos.

Ejemplo:

#### Petición invalidate\_token:

```
https://apps.anfix.com/os/os/parc/partner/invalidate_token
POST /os/os/parc/partner/invalidate_token HTTP/1.1
Host: anfix.com
Authorization: realm="facturas",
  oauth_consumer_key="898FCCEB3181586F8CBC8A8541711111",
  oauth_signature_method="PLAINTEXT",
  oauth_token="D224E32F8BDAC58AD8C0972EAB489C71",

oauth_signature="2F1E06E87CB7FBE7934AB947B0011111&39DC4913075810FA895CB7
ACCF5D7869"
```

#### Repuesta invalidate\_token:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
```

### Petición API Anfix:

Todas las peticiones a la API de Anfix son de tipo **HTTPS** y **POST**, con la correspondiente cabecera de **Authorization**. Es muy importante que tengan este formato para poder acceder a la API, pero como veremos en un ejemplo posterior, para la descarga de ficheros utilizaremos el método **GET** con parámetros en la URL.

#### Petición POST estándar:

La petición estándar como dijimos antes se trata de una petición tipo **POST** y **HTTPS**. Una petición normal a cualquier servicio de Anfix se compone de la URL correspondiente (ver API Anfix), mas el fichero JSON de entrada. A esta petición normal hay que añadir la cabecera de **Authorization** con los token finales que Anfix nos proporcionó en la respuesta del **access\_token**. Es muy importante que se rellene correctamente la cabecera de **Authorization** para que el servicio pueda validar las credenciales del desarrollador y devuelva los resultados esperados en el **JSON** de salida tal y como lo haría si se tratase de una petición hecha por el propio usuario al escritorio de Anfix.

La petición **estándar** se compone de tres partes:

- **URL:** <https://apps.anfix.com/os/os/parc/company/search>
- **Headers:**
  - Authorization:** realm="facturas",  
 oauth\_consumer\_key="898FCCEB3181586F8CBC8A8541711111",  
 oauth\_signature\_method="PLAINTEXT",  
 oauth\_token="D224E32F8BDAC58AD8C0972EAB489C71",  
 oauth\_signature="2F1E06E87CB7FBE7934AB947B0011111&39DC4913075810FA895CB7ACCF5D7869"
  - Content-Type:** application/json

- **JSON**

```
{
  "applicationId": "1",
  "inputBusinessData": {
    "Company": {
      "Order": [ "CompanyCorporateName" ]
    }
  }
}
```

La URL indica a que servicio de la API de Anfix esta invocando el desarrollador tecnológico. Como antes indicamos debe de ser una petición tipo **POST** y utilizar el protocolo **HTTPS**.

La petición tipo POST debe incluir una cabecera de autenticación llamada **Authorization**, que se compone de los siguientes campos obligatorios:

Parámetro	Tipo	Descripción	Obligatorio
<b>realm</b>	Cadena	Cadena de texto libre que indica el tipo de servicio que pretende dar el desarrollador, para el ejemplo "facturas".	✓
<b>oauth_consumer_key</b>	Cadena	Consumer_key que Anfix proporcionó al desarrollador cuando este se dio de alta como tal en Anfix.	✓
<b>oauth_signature_method</b>	Cadena	Método de firma utilizado para validar la petición. En Anfix utilizamos <b>PLAINTEXT</b> ya que todas nuestras peticiones se hacen sobre un canal seguro como HTTPS	✓
<b>oauth_signature</b>	Cadena	Firma de la petición. Ver apartado de construcción de firma para el <b>petición estandar</b> .	✓
<b>oauth_token</b>	Cadena	Token <b>final</b> que Anfix devolvió al desarrollador en la cabecera de respuesta de la petición <b>access_token</b>	✓

Y debe añadir otra cabecera llamada **Content-Type** con valor **application/json** para indicar que contiene un JSON de entrada.

Construcción de la **firma** para la **petición estándar**: La firma para la petición estándar es una cadena y se construye de la siguiente manera: "**Shared-Secret + & + oauth\_token\_secret**", se compone del **Shared-Secret** que le proporcionamos al desarrollador cuando se dio de alta como desarrollador tecnológico + el símbolo **& + oauth\_token\_secret** que Anfix devolvió en la respuesta a la petición **access\_token**, se trata del token\_secret definitivo. Para que la firma sea válida debe de ir en el orden propuesto, sino será desechada la petición por firma inválida. Es importante indicar que a partir de tener una credenciales definitivas o finales, la cabecera de Authorization siempre será la misma para todas las peticiones hasta que el usuario deniegue el acceso al desarrollador mediante el **invalidate\_token**.

Una vez lanzada la petición el servidor de Anfix validará el consumer\_key, la firma y token definitivo para ver si son correctos. El servidor de Anfix responderá con los siguientes resultados en el **JSON** de salida.

- Si en el JSON de salida contiene el campo **"result":0** significa que no hay errores de autenticación y la llamada al servicio es correcta. Esto significa que el servicio responde con el listado de empresas de ese usuario.

```
{
  "result":0,
  "outputData":{
    "RowNumber":2,
    "TotalRowNumber":2,
    "Company":[
      {
        "AddressCityCode":"186",
        "AddressCity":"VALLADOLID",
        "CompanyId":"89y,HPA7A", ...
      }
    ]
  }
}
```

- Si se han producido errores en el JSON el campo **"result":1** y nos mostrará la lista de errores encontrados. Estos errores pueden ser de validación de credenciales como vimos en las anteriores peticiones o un listado de errores del propio servicio (Ver API Anfix). Ejemplo:

```
{
  "result": 1,
  "errorList": [{
    "code": "ERR000061000",
    "fieldId": "OAuthCustomerKey",
    "text": "No existe un Partner Tecnológico para el
customer_key."
  }]
}
```

o

```
{
  "result": 1,
  "errorList": [{
    "code": "ERR000020001",
    "fieldId": "OAuthRealm",
    "text": "El campo es obligatorio."
  }]
}
```

Si el resultado es 0 (result:0), el desarrollador puede obtener los datos necesarios del **JSON** de salida y procesarlos para proveer al usuario de Anfix del servicio que ofrecen. Si por el contrario nos devuelve un vector de errores (result:1), el desarrollador deberá procesarlos y tomar la decisión de cómo subsanarlo.

Ejemplo:

**Petición estándar:**

```
https://apps.anfix.com/facturapro-comercio/gestiona/item/search
POST /facturapro-comercio/gestiona/item/search HTTP/1.1
Host: anfix.com
Authorization: realm="facturas",
  oauth_consumer_key="898FCCEB3181586F8CBC8A8541711111",
  oauth_signature_method="PLAINTEXT",
  oauth_token="D224E32F8BDAC58AD8C0972EAB489C71",

oauth_signature="2F1E06E87CB7FBE7934AB947B0011111&39DC4913075810FA895CB7
ACCF5D7869"
```

**JSON entrada:**

```
{
  "companyId": "89y,GZE0;",
  "applicationId": "2",
  "inputBusinessData": {
    "Item": {
      "Action": "SEARCH"
    }
  }
}
```

**Respuesta estándar:**

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
```

**JSON salida:**

```
{
  "result": 0,
  "outputData": {
    "Item": [
      {
        "ItemPropertyValue2Name": null,
        "ItemStockMaxValue": 0.0,
        "CompanyId": "89y,GZE0;",
        ....
      }
    ]
  }
}
```

**Petición GET:**

Para la descarga de ficheros se utiliza la petición tipo **GET**, con cadena de **Authorization** y sin **JSON** de entrada. La descarga del fichero lo que si necesita es un parámetro en la URL que indique el identificador del fichero que deseamos descargar. Este parámetro es el identificador del fichero (**entryIds**) y hay que conseguirlo haciendo una petición previa a los documentos del usuario para elegir el que deseamos descargar. Una vez hemos elegido el fichero podemos lanzar la petición tal y como se muestra a continuación.

- **URL:** <https://apps.anfix.com/documentos/mydocuments/download?entryIds=vu4Vdw4TA>
- **Headers:**  
**Authorization:** realm="facturas",  
  
    oauth\_consumer\_key="898FCCEB3181586F8CBC8A8541711111",  
    oauth\_signature\_method="PLAINTEXT",  
    oauth\_token="D224E32F8BDAC58AD8C0972EAB489C71",  
    oauth\_signature="2F1E06E87CB7FBE7934AB947B0011111&39DC4913075810FA895CB7ACCF5D7869"

La URL indica a que servicio de la API de Anfix esta invocando el desarrollador tecnológico. Como antes indicamos debe de ser una petición tipo **GET** y utilizar el protocolo **HTTPS**.

La petición tipo **GET** debe incluir una cabecera de autenticación llamada **Authorization**, que se compone de los siguientes campos obligatorios:

Parámetro	Tipo	Descripción	Obligatorio
realm	Cadena	Cadena de texto libre que indica el tipo de servicio que pretende dar el desarrollador, para el ejemplo "facturas".	✓
oauth_consumer_key	Cadena	Consumer_key que Anfix proporcionó al desarrollador cuando este se dio de alta como tal en Anfix.	✓
oauth_signature_method	Cadena	Método de firma utilizado para validar la petición. En Anfix utilizamos <b>PLAINTEXT</b> ya que todas nuestras peticiones se hacen sobre un canal seguro como HTTPS	✓
oauth_signature	Cadena	Firma de la petición. Ver apartado de construcción de firma para el <b>petición GET</b> .	✓
oauth_token	Cadena	Token <b>final</b> que Anfix devolvió al desarrollador en la cabecera de respuesta de la petición <b>access_token</b>	✓

Construcción de la **firma** para la **petición GET**: La firma para la petición GET es exactamente **igual** a la **firma estándar**. Se trata de una cadena y se construye de la siguiente manera: "**Shared-Secret + & + oauth\_token\_secret** ", se compone del **Shared-Secret** que le proporcionamos al desarrollador cuando se dio de alta como desarrollador tecnológico + el símbolo **& + oauth\_token\_secret** que Anfix devolvió en la respuesta a la petición **access\_token**, se trata del token\_secret definitivo. Para que la firma sea válida debe de ir en el orden propuesto, sino será desechada la petición por firma inválida. Es importante indicar que a partir de tener una credenciales definitivas o finales, la cabecera de Authorization siempre será la misma para todas las peticiones hasta que el usuario deniegue el acceso al desarrollador mediante el invalidate\_token.

Una vez lanzada la petición el servidor de Anfix validará el consumer\_key, la firma y token definitivo para ver si son correctos. El servidor de Anfix responderá con la descarga del fichero solicitado si este existe o el error correspondiente.

- Si se han producido errores en el **JSON** el campo "**result**":1 y nos mostrará la lista de errores encontrados. Estos errores pueden ser de validación de credenciales como vimos en las anteriores peticiones o un listado de errores del propio servicio (Ver API Anfix). Ejemplo:

```
{
  "result": 1,
  "errorList": [{
    "code": "ERR000061000",
    "fieldId": "OauthCustomerKey",
    "text": "No existe un Partner Tecnológico para el
customer_key."
  }]
}
```

```
{
  "result": 1,
  "errorList": [{
    "code": "ERR000020001",
    "fieldId": "OAuthRealm",
    "text": "El campo es obligatorio."
  }]
}
```

Ejemplo:

**Petición GET:**

```
https://apps.anfix.com/documentos/mydocuments/download?entryIds=vu4Vdw4TA
GET /documentos/mydocuments/download?entryIds=vu4Vdw4TA HTTP/1.1
Host: anfix.com
```

**Respuesta GET:**

Nos devuelve el fichero binario si este existe, sino nos devolverá un vector de errores como el resto de servicios indicando el problema encontrado como se puede ver en el siguiente ejemplo

```
{
  "result": 1,
  "errorList": [{
    "code": "ERR000060045",
    "text": "Error al descargar fichero(s) de mis documentos."
  }]
}
```