En esta página

Primeros pasos con webhooks

En este documento, se explica cómo configurar un webhook que te notificará cuando los usuarios de tu app publiquen un cambio en sus fotos de usuario. Cuando entiendas cómo configurar este webhook, sabrás cómo configurar los demás.

Para configurar cualquier webhook, debes hacer lo siguiente:

1. Crear un punto de conexión en un servidor seguro que pueda procesar solicitudes HTTPS.
2. Configurar el producto Webhooks en el panel de apps de tu app.

Estos pasos se explican de manera detallada a continuación.

Crear un punto de conexión

This step must be completed before you can subscribe to any webhook fields in the App Dashboard.

Your endpoint must be able to process two types of HTTPS requests: Verification Requests and Event Notifications. Since both requests use HTTPs, your server must have a valid TLS or SSL certificate correctly configured and installed. Self-signed certificates are not supported.

The sections below explain what will be in each type of request and how to respond to them. Alternatively, you can use our sample app which is already configured to process these requests.

Solicitudes de verificación

Anytime you configure the Webhooks product in your App Dashboard, we'll send a `GET` request to your endpoint URL. Verification requests include the following query string parameters, appended to the end of your endpoint URL. They will look something like this:

Sample Verification Request

```
GET https://www.your-clever-domain-name.com/webhooks?
  hub.mode=subscribe&
  hub.challenge=1158201444&
  hub.verify_token=meatyhamhock
```

Parameter	Sample Value	Description
`hub.mode`	`subscribe`	This value will always be set to `subscribe`.
`hub.challenge`	`1158201444`	An `int` you must pass back to us.
`hub.verify_token`	`meatyhamhock`	A string that that we grab from the **Verify Token** field in your app's App Dashboard. You will set this string when you complete the Webhooks configuration settings steps.

Note: PHP converts periods (.) to underscores (_) in parameter names.

Validating Verification Requests

Whenever your endpoint receives a verification request, it must:

- Verify that the `hub.verify_token` value matches the string you set in the **Verify Token** field when you configure the Webhooks product in your App Dashboard (you haven't set up this token string yet).
- Respond with the `hub.challenge` value.

If you are in your App Dashboard and configuring your Webhooks product (and thus, triggering a Verification Request), the dashboard will indicate if your endpoint validated the request correctly. If you are using the Graph API's /app/subscriptions endpoint to configure the Webhooks product, the API will indicate success or failure with a response.

Notificaciones de eventos

When you configure your Webhooks product, you will subscribe to specific `fields` on an `object` type (e.g., the `photos` field on the `user` object). Whenever there's a change to one of these fields, we will send your endpoint a `POST` request with a JSON payload describing the change.

For example, if you subscribed to the `user` object's `photos` field and one of your app's Users posted a Photo, we would send you a `POST` request that would look something like this:

```
POST / HTTPS/1.1
Host: your-clever-domain-name.com/webhooks
Content-Type: application/json
X-Hub-Signature-256: sha256={super-long-SHA256-signature}
Content-Length: 311

{
  "entry": [
    {
      "time": 1520383571,
      "changes": [
        {
          "field": "photos",
          "value":
            {
              "verb": "update",
              "object_id": "10211885744794461"
            }
        }
      ],
      "id": "10210299214172187",
      "uid": "10210299214172187"
    }
  ],
  "object": "user"
}
```

Payload Contents

Payloads will contain an object describing the change. When you configure the webhooks product, you can indicate if payloads should only contain the names of changed fields, or if payloads should include the new values as well.

We format all payloads with JSON, so you can parse the payload using common JSON parsing methods or packages.

> You will not be able to query historical webhook event notification data, so be sure to capture and store any webhook payload content that you want to keep.

Most payloads will contain the following common properties, but the contents and structure of each payload varies depending on the object fields you are subscribed to. Refer to each object's reference document to see which fields will be included.

Property	Description	Type
`object`	The object's type (e.g., `user`, `page`, etc.)	`string`
`entry`	An array containing an object describing the changes. Multiple changes from different objects that are of the same type may be batched together.	`array`
`id`	The object's ID	`string`
`changed_fields`	An array of strings indicating the names of the fields that have been changed. Only included if you *disable* the **Include Values** setting when configuring the Webhooks product in your app's App Dashboard.	`array`
`changes`	An array containing an object describing the changed fields and their new values. Only included if you *enable* the **Include Values** setting when configuring the Webhooks product in your app's App Dashboard.	`array`

Property	Description	Type
`time`	A UNIX timestamp indicating when the Event Notification was sent (not when the change that triggered the notification occurred).	`int`

Validating Payloads

We sign all Event Notification payloads with a **SHA256** signature and include the signature in the request's `X-Hub-Signature-256` header, preceded with `sha256=`. You don't have to validate the payload, but you should.

To validate the payload:

1. Generate a **SHA256** signature using the payload and your app's **App Secret**.
2. Compare your signature to the signature in the `X-Hub-Signature-256` header (everything after `sha256=`). If the signatures match, the payload is genuine.

Responding to Event Notifications

Your endpoint should respond to all Event Notifications with `200 OK HTTPS`.

Frequency

Event Notifications are aggregated and sent in a batch with a **maximum** of 1000 updates. However batching cannot be guaranteed so be sure to adjust your servers to handle each Webhook individually.

If any update sent to your server fails, we will retry immediately, then try a few more times with decreasing frequency over the next 36 hours. Your server should handle deduplication in these cases. Unacknowledged responses will be dropped after 36 hours.

Note: The frequency with which Messenger event notifications are sent is different. Please refer to the Messenger Platform Webhooks documentation for more information.

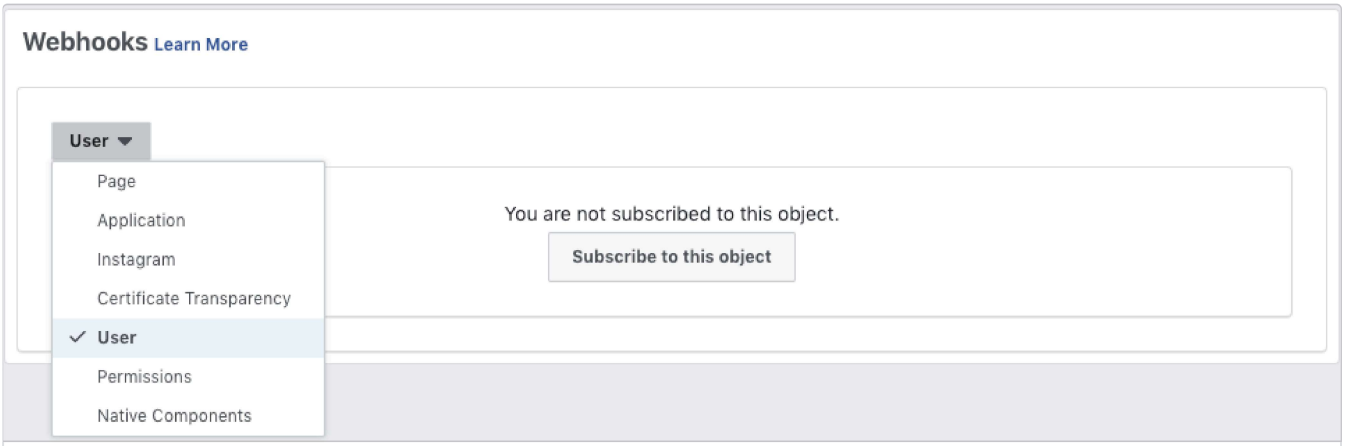
Configuración del producto Webhooks

Una vez que el punto de conexión o la app de ejemplo estén listos, usa el panel de apps de tu app para agregar y configurar el producto Webhooks. También puedes hacer esto de manera programática

mediante el punto de conexión `/{app-id}/subscriptions` para todos los webhooks, a excepción de Instagram.

En este ejemplo, utilizaremos el panel para configurar un webhook que se suscribe a cualquier cambio en las fotos de cualquiera de los usuarios de tu app.

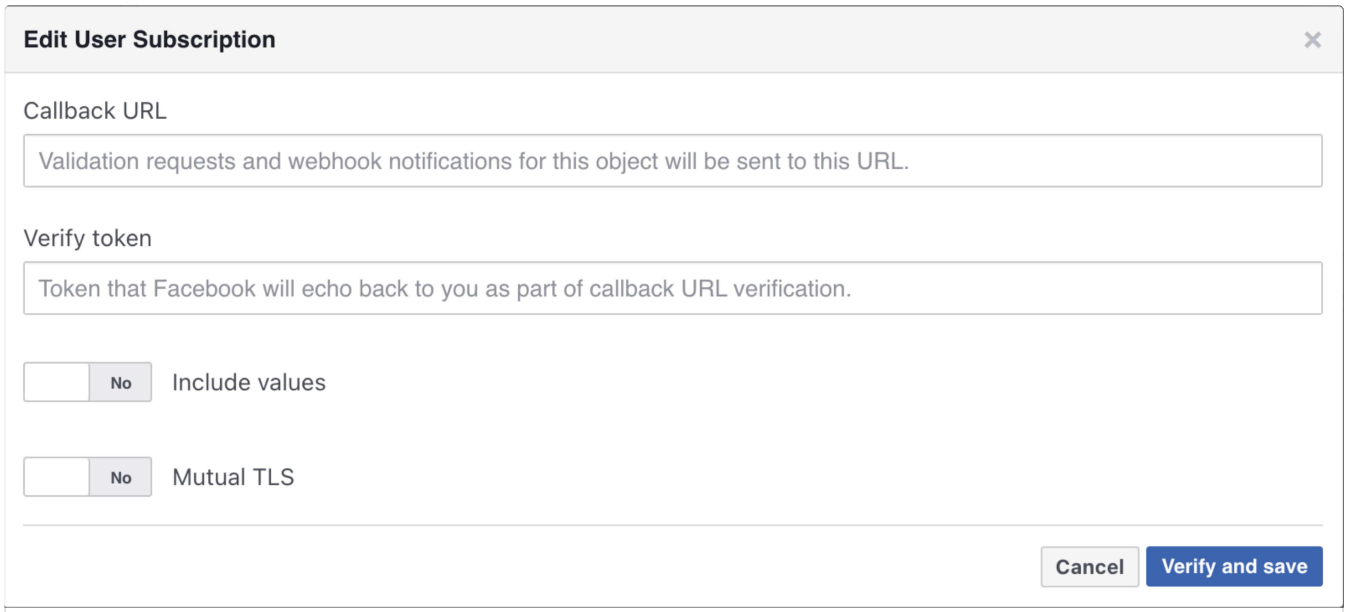
1. En el panel de apps, ve a **Productos** > **Webhooks**, selecciona **Usuario** en el menú desplegable y, a continuación, haz clic en **Suscribirse a este objeto**.



Selección del objeto del usuario.

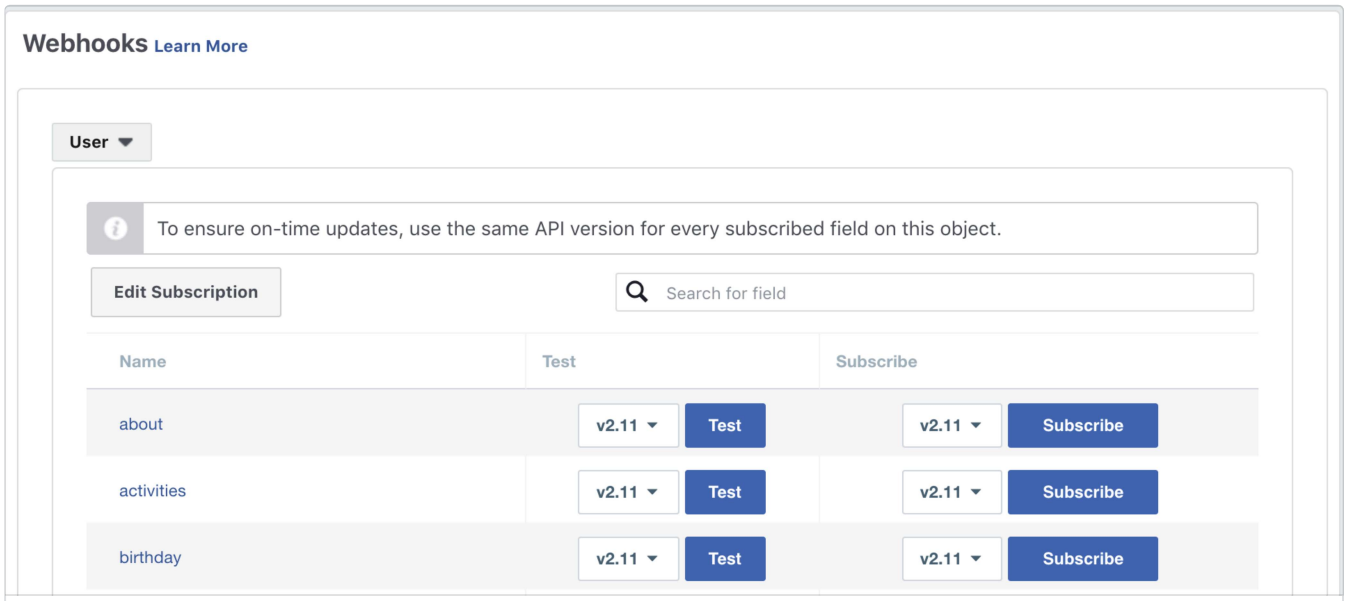
2. Ingresa la URL del punto de conexión en el campo **URL de devolución de llamada** y una cadena en el campo **Token de verificación**. Incluiremos esta cadena en todas las solicitudes de verificación. Si usas una de nuestras apps de ejemplo, debe ser la misma cadena que usaste para la variable de configuración `TOKEN`.

   Si quieres que las cargas de notificación del evento incluyan los nombres de los campos que cambiaron además de los nuevos valores, define la opción **Incluir valores** en **Sí**.



Ingreso de una URL de punto de conexión y una cadena de token de verificación.

3. Luego de hacer clic en **Verificar y guardar**, enviaremos al punto de conexión una solicitud de verificación que debes validar. Si el punto de conexión valida exitosamente la solicitud, deberías ver esto:

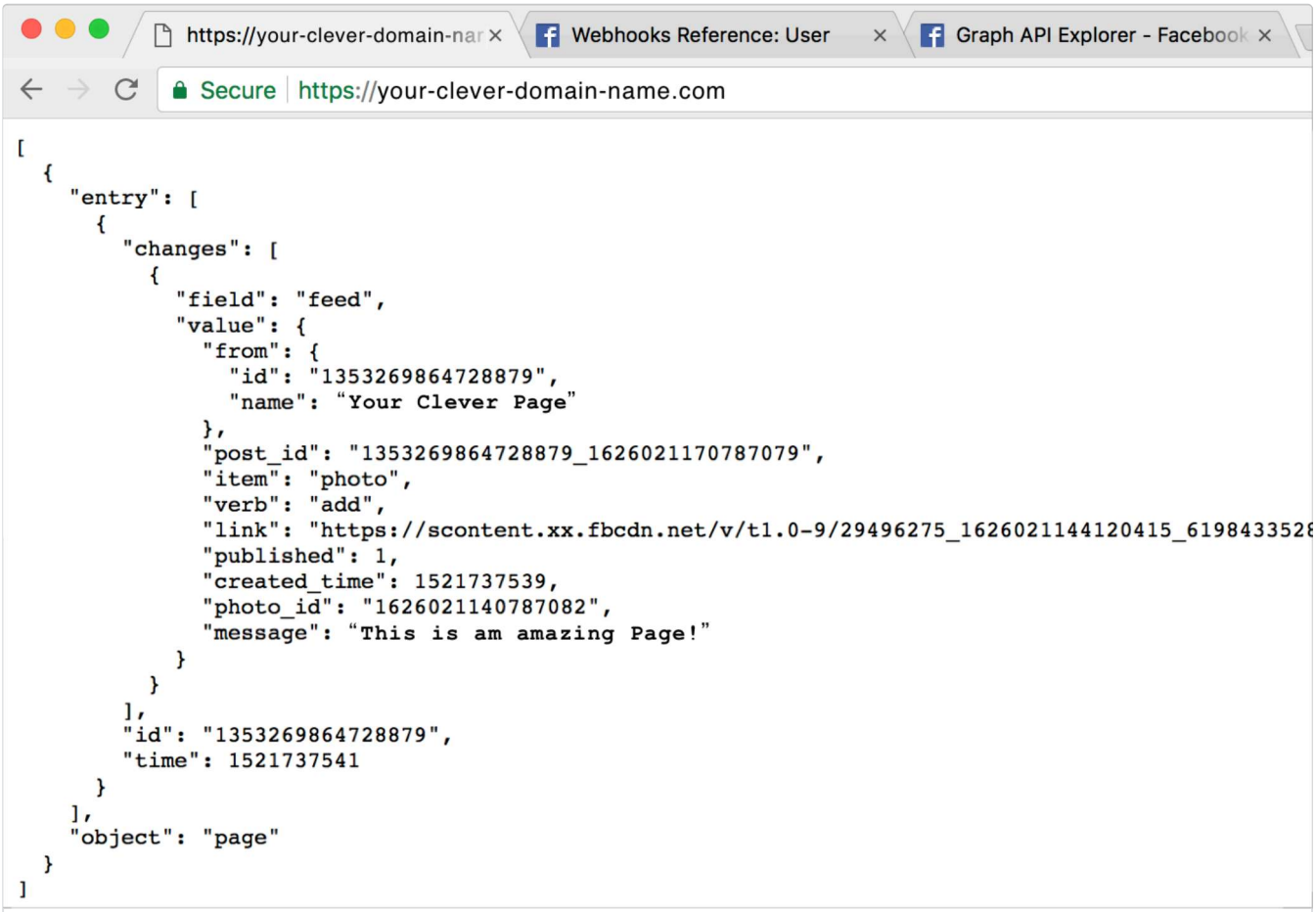


Validación exitosa.

4. El último paso consiste en suscribirse a campos individuales. Suscríbete al campo `photos` y envía una notificación de evento de prueba.

Suscripción al campo "Fotos" en el objeto "Usuario".

Si el punto de conexión está configurado correctamente, debería validar la carga y ejecutar el código que hayas configurado para que se ejecute después de una validación exitosa. Si usas nuestra app de ejemplo, carga la URL de la app en el navegador web. Debería mostrar el contenido de la carga:



App de ejemplo que muestra la carga de notificación de prueba.

mTLS for Webhooks

Mutual TLS (mTLS) is a method for mutual authentication.

mTLS ensures that the parties at each end of a network connection are who they claim to be by verifying that they both have the correct private key. The information within their respective TLS certificates

provides additional verification.

How to configure mTLS

Once you enable mTLS on your subscription to WhatsApp Business Account, Meta will present a client certificate together with its signing intermediate certificate. Both certificates are used to create a TLS handshake of Webhook requests to your server. Your server then can verify the sender's identity of these requests by the trust chain and the common name (CN).

The client certificate is signed by an intermediate CA certificate, DigiCert SHA2 High Assurance Server CA, and then by a root CA certificate, DigiCert High Assurance EV Root CA. Note that the intermediate certificate also signs the certificate for graph.facebook.com:

Client Certificate Verification

After setting up HTTPS for receiving Webhook requests, complete the following steps to verify the client certificate and its common name `client.webhooks.fbclientcerts.com`:

1. Install the root certificate
2. Verify the client certificate against the root certificate
3. Verify the common name (`client.webhooks.fbclientcerts.com`) of the client certificate

Note: Servers receiving Webhooks must be using HTTPS; and we are always verifying the certificate from your HTTPS server for security.

Example

Depending on your server's setup, the above steps vary in details. We illustrate by two examples, one for Nginx and one for AWS Application Load Balancer (ALB).

Nginx

1. Download the root certificate (DigiCert High Assurance EV Root CA) from DigiCert to your server, e.g. `/etc/ssl/certs/DigiCert_High_Assurance_EV_Root_CA.pem`

2. Turn on mTLS by Nginx directives (example screenshot)

```
ssl_verify_client       on;
ssl_client_certificate  /etc/ssl/certs/DigiCert_High_Assurance_EV_Root_CA.pem;
ssl_verify_depth        3;
```

3. Verify the CN from Nginx embedded variable `$ssl_client_s_dn` equals `"client.webhooks.fbclientcerts.com"` (example screenshot)

```
if ($ssl_client_s_dn ~ "CN=client.webhooks.fbclientcerts.com") {
    return 200 "$ssl_client_s_dn";
}
```

AWS Application Load Balancer (ALB)

1. Download the intermediate certificate (DigiCert SHA2 High Assurance Server CA) from DigiCert to a S3 bucket. The root certificate is not accepted by AWS because it is signed using algorithm SHA1withRSA; while the intermediate certificate is signed using SHA256withRSA thereby accepted.
2. Configure the HTTPS listener on the ALB to enable mTLS with the trust store containing the certificate in the S3 bucket (example screenshot).
3. In your application code, extract the CN from the HTTP header "X-Amzn-Mtls-Clientcert-Subject", and verify it equals `"client.webhooks.fbclientcerts.com"`.

Próximos pasos

Ahora que sabes cómo configurar webhooks, quizás te resulte conveniente consultar nuestros documentos complementarios, donde se describen los pasos adicionales necesarios para configurar webhooks para productos específicos:

- Webhooks para cuentas publicitarias
- Webhooks para transparencia de certificados
- Webhooks para Instagram
- Webhooks para clientes potenciales
- Webhooks para Messenger
- Webhooks para páginas
- Webhooks para pagos
- Webhooks para WhatsApp