Páginas de Respuesta y Confirmación

Updated on 30 Oct 2025 · 6 Minutes to read · Contributors

ePayco proporciona dos mecanismos para notificar el resultado de una transacción: la **página de respuesta** y la **URL de confirmación** (webhook). Es importante entender la diferencia entre ambas y cómo implementarlas correctamente.

Página de Respuesta (`response`)

La página de respuesta es la URL a la que se redirige al usuario **después de completar el proceso de pago** en **ePayco Smart Checkout**. Esta redirección ocurre en el navegador del cliente y contiene información básica sobre la transacción en los parámetros de la URL.

Características:

- ✅ Redirección visible para el usuario

- ✅ Permite mostrar una página de respuesta personalizada

- ✅ Incluye el parámetro `ref_payco` en la URL para consultar datos de la transacción

- ⚠️ **NO es confiable** para validar el estado final de la transacción (el usuario puede cerrar el navegador o perder conexión)

- ⚠️ Los parámetros pueden ser manipulados por el usuario

Comportamiento de redirección:

1. **Con URL personalizada**: Si se configura una URL de respuesta personalizada, el usuario será redirigido a esa URL con el parámetro `ref_payco` que permite consultar los datos de la transacción.

2. **Sin URL personalizada**: Si no se configura una URL de respuesta, **ePayco** redirige a una página predeterminada que permite al usuario imprimir y reenviar el comprobante de la transacción.

Ejemplo de URL de redirección:

Plain text | Copy

```
https://mitienda.com/resultado-pago?ref_payco=68fb83729d094878e015be00
```

Para obtener los atributos que se envían en la página de respuesta dinámica deberás hacer un llamado mediante el siguiente endpoint:

Bash | Copy

```bash
curl --location --request GET 'https://secure.epayco.co/validation/v1/refere
--header 'Content-Type: application/json'
```

URL de Confirmación (`confirmation`) - Webhook

La URL de confirmación es un **webhook** que ePayco invoca en tu servidor backend cuando el estado de una transacción cambia. Esta es la forma **confiable y segura** de validar el resultado final de una transacción.

Características:

- ✅ **Confiable**: Comunicación servidor a servidor
- ✅ Invocado automáticamente por ePayco
- ✅ Puede reintentar en caso de fallo
- ✅ Incluye firma de seguridad para validar autenticidad
- ⚠️ Debe responder con código HTTP 200 para confirmar recepción

Parámetros recibidos:

Parámetros recibidos (método GET o POST):

Ejemplo de implementación:

POST | GET

JavaScript Copy

```javascript
const express = require('express');
const crypto = require('crypto');
const app = express();
app.use(express.urlencoded({ extended: true }));
app.use(express.json());
app.post('api/webhook/epayco/confirmation', async (req, res) => {
  try {
    const {
      ref_payco,
      x_ref_payco,
      x_transaction_id,
      x_response,
      x_response_reason_text,
      x_amount,
      x_currency_code,
      x_franchise,
      x_signature,
      x_test_request,
      x_approval_code,
      x_transaction_date,
      x_transaction_state,
      x_customer_email,
      x_customer_name,
      x_extra1,
      x_extra2
    } = req.body;
    // 1. Validar firma de seguridad
    const p_cust_id_cliente = process.env.EPAYCO_CUSTOMER_ID;
    const p_key = process.env.EPAYCO_P_KEY;
    const signature = crypto
      .createHash('sha256')
      .update(`${p_cust_id_cliente}^${p_key}^${x_ref_payco}^${x_transact
      .digest('hex');
    if (signature !== x_signature) {
      console.error('Firma inválida - posible intento de fraude');
      return res.status(400).send('Invalid signature');
    }
    // 2. Validar que no sea transacción duplicada
    const existingTransaction = await checkTransactionExists(x_transact
    if (existingTransaction) {
      console.log('Transacción ya procesada:', x_transaction_id);
      return res.status(200).send('OK');
```

```
    }
    // 3. Procesar según el estado de la transacción
    switch (x_response) {
      case 'Aceptada':
        await processApprovedTransaction({
          refPayco: ref_payco,
          transactionId: x_transaction_id,
          amount: x_amount,
          currency: x_currency_code,
          customerEmail: x_customer_email,
          customerName: x_customer_name,
          franchise: x_franchise,
          approvalCode: x_approval_code,
          transactionDate: x_transaction_date
        });
        break;
      case 'Rechazada':
        await processRejectedTransaction({
          refPayco: ref_payco,
          transactionId: x_transaction_id,
          reason: x_response_reason_text
        });
        break;
      case 'Pendiente':
        await processPendingTransaction({
          refPayco: ref_payco,
          transactionId: x_transaction_id
        });
        break;
      case 'Fallida':
        await processFailedTransaction({
          refPayco: ref_payco,
          transactionId: x_transaction_id,
          reason: x_response_reason_text
        });
        break;
    }
    // 4. Responder con 200 OK para confirmar recepción
    res.status(200).send('OK');
  } catch (error) {
    console.error('Error procesando webhook:', error);
    res.status(500).send('Internal Server Error');
  }
});
// Funciones auxiliares
async function checkTransactionExists(transactionId) {
```

```
  // Implementar consulta a tu base de datos
  return false;
}
async function processApprovedTransaction(data) {
  // Actualizar orden como pagada
  // Enviar email de confirmación
  // Activar servicios/productos
  console.log('Transacción aprobada:', data);
}
async function processRejectedTransaction(data) {
  // Actualizar orden como rechazada
  // Notificar al usuario
  console.log('Transacción rechazada:', data);
}
async function processPendingTransaction(data) {
  // Marcar orden como pendiente
  // Programar verificación posterior
  console.log('Transacción pendiente:', data);
}
async function processFailedTransaction(data) {
  // Manejar transacción fallida
  console.log('Transacción fallida:', data);
}
app.listen(3000, () => {
  console.log('Webhook server running on port 3000');
});
```

Validación de Firma de Seguridad

La firma de seguridad `x_signature` permite verificar que la petición proviene realmente de ePayco y no ha sido manipulada.

Fórmula para calcular la firma:

Parámetros necesarios:

- `p_cust_id_cliente` : Tu Customer ID de ePayco (obtenido del dashboard)
- `p_key` : Tu P_KEY de ePayco (obtenido del dashboard)
- `^` : Carácter separador literal

Bash Copy

```
SHA256(p_cust_id_cliente^p_key^x_ref_payco^x_transaction_id^x_amount^x_curre
```

Ejemplo en diferentes lenguajes:

JavaScript/Node.js | PHP | Python

JavaScript Copy

```javascript
const crypto = require('crypto');
function validateSignature(data, customerID, pKey) {
  const signature = crypto
    .createHash('sha256')
    .update(`${customerID}^${pKey}^${data.x_ref_payco}^${data.x_transact
    .digest('hex');
  return signature === data.x_signature;
}
```

Mejores Prácticas

1. **Siempre valida en el webhook**: Nunca confíes únicamente en la página de respuesta para confirmar pagos.

2. **Implementa idempotencia**: ePayco puede reintentar el webhook múltiples veces. Verifica que no proceses la misma transacción dos veces.

3. **Responde rápidamente**: El webhook debe responder con HTTP 200 en menos de 30 segundos. Para procesos largos, usa colas de trabajos.

4. **Valida la firma**: Siempre verifica la firma de seguridad `x_signature` para prevenir fraudes.

5. **Registra todo**: Guarda todos los webhooks recibidos para auditoría y debugging.

6. **Maneja reintentos**: Si tu servidor está caído, ePayco reintentará enviar el webhook. Asegúrate de manejar esto correctamente.

7. **Usa HTTPS**: La URL de confirmación debe usar HTTPS para seguridad.

8. **No redirijas**: El endpoint del webhook no debe hacer redirecciones (301/302).

Ejemplo de Configuración

Al crear la sesión de checkout, configura ambas URLs:

JavaScript Copy

```
const sessionData = {
  checkout_version: "2",
  name: "Mi Tienda",
  description: "Producto ejemplo",
  currency: "COP",
  amount: 50000,
  lang: "ES",
  // URL donde se redirige al usuario (visible)
  response: "https://mitienda.com/resultado-pago",
  // URL del webhook (servidor a servidor)
  confirmation: "https://mitienda.com/api/webhook/epayco/confirmation"
};
```

> **Importante: Ambas URLs deben ser accesibles públicamente. Para desarrollo local, puedes usar herramientas como ngrok (https://ngrok.com/) o localtunnel (https://localtunnel.github.io/www/) para exponer tu servidor local.**

Previous
Implementación

Next
Tablas de datos