FieldLogix Webhook - Customer Implementation Guide

This guide is for **any FieldLogix user who** wants to receive real-time notifications when their vehicles enter or exit landmarks. The webhook system is not exclusive to any single integration - every customer can register their own endpoint.

REQUIREMENTS

To receive webhook events from FieldLogix you need:

1. A HTTPS endpoint - a URL on your server that accepts `POST` requests
2. A secret string - any string you choose; used to verify that POSTs come from FieldLogix
3. A FieldLogix session token - obtained by logging in to the FieldLogix API

Copyright 2026 - FieldLogix

Step 1 - Authenticate

Call the login endpoint to obtain a session token:

POST

```
https://api.fieldlogixgps.com/api/login
Content-Type: application/json

{
 "USERNAME": "your_fieldlogix_username",
 "PASSWORD": "your_fieldlogix_password"
}
```

Response:

```
{ "session_token": "a1b2c3d4-..." }
```

Use this token as the `Authorization` header in all subsequent calls.

Tokens expire after 14 days.

Step 2 - Register your webhook endpoint

```
POST https://api.fieldlogixgps.com/api/WebhookSubscriptions
Authorization: <session_token>
Content-Type: application/json

{
 "EVENT_TYPE":  "LANDMARK_ENTRY_EXIT",
 "WEBHOOK_URL":  "https://your-system.com/fieldlogix-webhook",
 "SECRET":      "choose-any-strong-random-string",
 "AUTH_HEADER":  "Bearer your-api-token"
}
```

Copyright 2026 - FieldLogix

Field	Required	Description
`EVENT_TYPE`	Yes	Type of event. Currently only `LANDMARK_ENTRY_EXIT` is supported
`WEBHOOK_URL`	Yes	Your HTTPS endpoint URL. Must start with `https://`
`SECRET`	Yes	A string **you choose**. FieldLogix uses it to sign each event. Store it securely
`AUTH_HEADER`	No	If your endpoint requires authentication, provide the full header value (e.g. `Bearer xyz`)

> ***Security note:** This call must be made over HTTPS. The `SECRET` travels only once, during this setup call. It is never included in the webhook events themselves.*

Response:

```
{ "SUBSCRIPTION_ID": 42 }
```

Save the `SUBSCRIPTION_ID` for future updates or deactivation.

Step 3 - Implement your webhook endpoint

When a vehicle enters or exits a landmark, FieldLogix will `POST` to your `WEBHOOK_URL`:

Headers sent by FieldLogix:

```
Content-Type: application/json
X-FieldLogix-Signature: sha256=<hmac_hex>
X-FieldLogix-Event: landmark_entry_exit
Authorization: <your AUTH_HEADER if provided>
```

Payload:

```
{
 "event_type": "landmark_entry",
 "occurred_at_utc": "2026-10-14T17:44:33Z",
 "vehicle": {
  "vehicle_id": 26098,
  "name": "Truck-07"
```

```
},
"landmark": {
  "landmark_id": 420910,
  "name": "Warehouse A",
  "integration_id": "STD2392110949222402"
},
"driver": {
  "driver_id": 88,
  "name": "John Smith"
},
"history_id": 1188140549
}
```


Field	Notes
`event_type`	`landmark_entry`, `landmark_exit`, or `landmark_entry_and_exit`
`occurred_at_utc`	ISO 8601 UTC timestamp of the GPS detection
`landmark.integration_id`	The external ID stored on the landmark. Use this to match to your own records. Can be `null` if the landmark has no external ID configured in FieldLogix
`driver`	Present when a driver is assigned to the vehicle at the time of the event. Absent when no driver is assigned
`history_id`	Unique GPS record ID. Use this as a deduplication key - in rare cases the same event may be delivered more than once

Step 4 - Validate the signature

Always validate the signature before processing an event. This confirms the POST came from FieldLogix and the payload was not tampered with.

Copyright 2026 - FieldLogix

The `SECRET` is never sent in the webhook POST - only the HMAC derived from it.

HMAC-SHA256 is a one-way function: the secret cannot be extracted from the signature.

The only moment the secret travels is the HTTPS setup call in Step 2.

Algorithm:

1. Read the raw request body as bytes (before any JSON parsing)
2. Compute `HMAC-SHA256(secret, raw_body_bytes)`
3. Compare with the `X-FieldLogix-Signature` header value (`sha256=<hex>`)
4. If they do not match: reject the request

Python:

```
import hmac, hashlib

def verify(secret: str, raw_body: bytes, header: str) -> bool:
  expected = "sha256=" + hmac.new(
    secret.encode("utf-8"), raw_body, hashlib.sha256
  ).hexdigest()
  return hmac.compare_digest(expected, header)

# In your request handler:
if not verify("your-secret", request.get_data(), request.headers["X-FieldLogix-Signature"]):
  return Response(status=401)
event = request.get_json()
```

Node.js:

```
const crypto = require("crypto");

function verify(secret, rawBody, header) {
  const expected = "sha256=" + crypto.createHmac("sha256", secret).update(rawBody).digest("hex");
  return crypto.timingSafeEqual(Buffer.from(expected), Buffer.from(header));
}

app.post("/fieldlogix-webhook", express.raw({ type: "*/*" }), (req, res) => {
  if (!verify("your-secret", req.body, req.headers["x-fieldlogix-signature"])) {
    return res.status(401).send("Unauthorized");
  }
  const event = JSON.parse(req.body);
  res.status(200).send("OK");
});
```

Copyright 2026 - FieldLogix

C#:

```
using System.Security.Cryptography;

bool Verify(string secret, string rawBody, string header) {
  var key = Encoding.UTF8.GetBytes(secret);
  var body = Encoding.UTF8.GetBytes(rawBody);
  using var hmac = new HMACSHA256(key);
  var expected = "sha256=" + BitConverter.ToString(hmac.ComputeHash(body)).Replace("-",
"").ToLower();
  return CryptographicOperations.FixedTimeEquals(
    Encoding.UTF8.GetBytes(expected), Encoding.UTF8.GetBytes(header));
}
```

PHP:

```
function verify_signature(string $secret, string $raw_body, string $header): bool {
  $expected = 'sha256=' . hash_hmac('sha256', $raw_body, $secret);
  return hash_equals($expected, $header);
}

$raw_body  = file_get_contents('php://input');
$signature = $_SERVER['HTTP_X_FIELDLOGIX_SIGNATURE'] ?? '';
if (!verify_signature('your-secret', $raw_body, $signature)) {
  http_response_code(401);
  exit;
}
$event = json_decode($raw_body, true);
```

> ***Important:** always use constant-time comparison (`hmac.compare_digest`, `timingSafeEqual`, `FixedTimeEquals`). Standard `==` is vulnerable to timing attacks.*

Step 5 - Respond correctly

Your endpoint must return HTTP `2xx` within 15 seconds.

- Return `200 OK` or `202 Accepted` when the event is received successfully

- If your endpoint returns `4xx` or `5xx`, or times out, FieldLogix retries up to 6 times with exponential backoff: immediate, then +15 min, +1h, +2h, +4h, +12h
- After 6 failed attempts the event is marked as permanently failed and will not be retried

Idempotency: use `history_id` as a deduplication key. In rare cases (e.g., retry after timeout), the same event may be delivered more than once.

Managing your subscription

```
# List your subscriptions
GET /api/WebhookSubscriptions
Authorization: <session_token>

# Update URL or secret
PUT /api/WebhookSubscriptions/{id}
Authorization: <session_token>
{ "EVENT_TYPE": "LANDMARK_ENTRY_EXIT", "WEBHOOK_URL": "https://...", "SECRET": "new-secret"
}

# Deactivate
DELETE /api/WebhookSubscriptions/{id}
Authorization: <session_token>
```

> ***Note on security:** `SECRET` and `AUTH_HEADER` are never returned in API responses.*
> *If you forget your secret, update it via `PUT` with a new value.*

Subscription health fields

The `GET /api/WebhookSubscriptions` response includes fields that let you monitor the health of your subscription:

Copyright 2026 - FieldLogix

Field	Description
`FAILURE_COUNT`	Number of individual failed delivery attempts in the current 24-hour window. Resets to 0 after any successful delivery.
`FAILURE_WINDOW_START`	When the current 24-hour failure counting window started. `null` when no failures have occurred.
`FIRST_FAILURE_DATE`	When the first failure in the current streak occurred. `null` after a successful delivery.
`DEACTIVATED_DATE`	When the subscription was deactivated. Add 24 hours to this value to know when auto-restart will occur. `null` when active.
`DEACTIVATED_REASON`	Reason for deactivation. See table below.

`DEACTIVATED_REASON` value	Meaning
`null`	Manually deactivated via DELETE
`auto_shutdown_50_failures_in_24h`	Temporarily shut down. Auto-restarts after 24 hours.
`permanently_disabled_no_deliveries_in_5_days`	Permanently disabled. Requires manual reactivation via PUT.

Automatic subscription lifecycle

FieldLogix automatically manages subscriptions that consistently fail to deliver:

Temporary shutdown: if your endpoint accumulates 50 failed delivery attempts within a 24-hour window with no successful deliveries, FieldLogix temporarily shuts down the subscription (`ACTIVE=false`, `DEACTIVATED_REASON=auto_shutdown_50_failures_in_24h`). Any successful delivery at any point resets the counter and cancels the shutdown.

Auto-restart: after 24 hours of shutdown, the subscription is automatically reactivated. Events that were queued during the shutdown will be retried from where they left off.

Copyright 2026 - FieldLogix

Permanent disable: if 5 consecutive days pass with no successful deliveries, the subscription is permanently disabled (`DEACTIVATED_REASON=permanently_disabled_no_deliveries_in_5_days`). This requires manual reactivation.

To reactivate a permanently disabled subscription: fix your endpoint and call:

```
PUT /api/WebhookSubscriptions/{id}
Authorization: <session_token>
Content-Type: application/json

{
 "EVENT_TYPE": "LANDMARK_ENTRY_EXIT",
 "WEBHOOK_URL": "https://your-fixed-endpoint.com/webhook",
 "SECRET": "your-secret",
 "ACTIVE": true
}
```

Currently supported event types

Event Type	Description
`LANDMARK_ENTRY_EXIT`	Vehicle enters or exits a landmark

Additional event types (zone alerts, speed alerts, etc.) are not yet available.

Contact FieldLogix if you need support for other event types.

Copyright 2026 - FieldLogix