FYDO API Authentication Guide

How to implement HMAC-SHA256 request signing for the FYDO API, including code samples, error responses and the enforcement timeline.

As part of our ongoing commitment to security and reliability, we are introducing HMAC-SHA256 request signing for all FYDO API integrations. This provides stronger authentication, request integrity verification, and replay protection for every API call.

This guide covers everything you need to implement the new authentication model and transition your integration.

> Legacy API Key authentication continues to be accepted until **8 October 2026**, so you can migrate at a time that suits you within that window.

What's New

FYDO API integrations now authenticate using HMAC-SHA256 request signing. Every API request includes a signed Authorization header that verifies the caller's identity and ensures the request has not been tampered with in transit.

- **Request signing** – every request is signed using your HMAC Secret. The secret never leaves your environment and is never transmitted over the wire.
- **Replay protection** – a timestamp is included in every signature. Requests older than 5 minutes are automatically rejected.
- **Request integrity** – the request body is included in the signature hash. Any modification to the payload after signing will cause the request to be rejected.
- **Server-side tenant resolution** – the server resolves your hospital context automatically from your Integrator ID. No hospital identifiers are required in the request body.

How It Works

Your Credentials

Your hospital administrator will provide you with two credentials:

- **Integrator ID** – a unique identifier for your integration, in GUID format. This is included in the Authorization header to identify your integration.
- **HMAC Secret** – a shared secret used to sign your requests. This must be stored securely and never transmitted, committed to source control, or shared over unencrypted channels.

Base URL and Endpoints

All API requests are made over HTTPS to:

```
https://fydo.cloud/WebhooksApi/Api/
```

For example, the patient list endpoint is:

```
https://fydo.cloud/WebhooksApi/Api/Patient/getPatientList
```

Plain HTTP is not supported.

A new, comprehensive FYDO API reference is being finalised and will be published shortly. It will cover all available endpoints, request and response schemas, and the data each endpoint returns.

Your hospital controls which endpoints your integration can access. If an endpoint returns **403**, contact your hospital administrator to request access.

Authorization Header Format

Every request must include an Authorization header in the following format:

```
FYDO-HMAC-SHA256 IntegratorId:Signature:Timestamp
```

- **IntegratorId** – your unique Integrator ID (GUID format)
- **Signature** – Base64-encoded HMAC-SHA256 signature of the canonical string (see below)
- **Timestamp** – current UTC time in ISO 8601 format

Timestamp Format

The timestamp must be UTC in the following ISO 8601 format, with **exactly three decimal places** for milliseconds and a trailing **Z**:

```
yyyy-MM-ddTHH:mm:ss.fffZ
```

```
Example: 2026-08-06T06:14:22.123Z
```

> **IMPORTANT** — Timestamps with no milliseconds, more than three decimal places, or an offset such as **+00:00** instead of **Z** will be rejected. Several language defaults do not produce this format, so use the code samples below rather than a built-in ISO 8601 helper.

The timestamp must be within 5 minutes of the server time.

Canonical String

The signature is computed over a canonical string with the following format:

```
HTTP_METHOD\nREQUEST_PATH\nTIMESTAMP\nBODY_SHA256_HASH
```

`\n means a newline character (LF), not a literal backslash followed by n.`

- **HTTP_METHOD** – the HTTP method in uppercase, e.g. POST
- **REQUEST_PATH** – the path component of the URL, converted to lowercase
- **TIMESTAMP** – the same timestamp included in the Authorization header
- **BODY_SHA256_HASH** – Base64-encoded SHA-256 hash of the raw request body

> **IMPORTANT** — REQUEST_PATH is the path only, not the full URL, and it must be lowercase. For **https://fydo.cloud/WebhooksApi/Api/Patient/getPatientList** the value used for signing is **/webhooksapi/api/patient/getpatientlist**. Do not include the scheme, host, or any query string.

Request Body

The request body contains only your business payload. For example:

```
{
“PageIndex”: 1,
“PageSize”: 1000
}
```

Numeric values may be sent as JSON numbers or as quoted strings. Both are accepted. No authentication or hospital routing fields are required in the body.

Signing Process Summary

1. Construct the request body with your business payload
2. Compute the SHA-256 hash of the body and Base64-encode it
3. Build the canonical string
4. Compute the HMAC-SHA256 of the canonical string using your HMAC Secret
5. Base64-encode the signature
6. Set the Authorization header
7. Send the request over HTTPS

Code Samples

C#

```
using System;
using System.Security.Cryptography;
using System.Text;

string secret = "";       // Your HMAC Secret
string integratorId = ""; // Your Integrator ID

string method = "POST";
string path = "/WebhooksApi/Api/Patient/getPatientList".ToLowerInvariant();
string timestamp = DateTime.UtcNow.ToString("yyyy-MM-ddTHH:mm:ss.fffZ");

string body = @"{
    ""PageIndex"": 1,
    ""PageSize"": 1000
}";

using (SHA256 sha = SHA256.Create())
{
    string bodyHash = Convert.ToBase64String(
        sha.ComputeHash(Encoding.UTF8.GetBytes(body))
    );

    string canonical = $"{method}\n{path}\n{timestamp}\n{bodyHash}";

    using (HMACSHA256 hmac = new HMACSHA256(Encoding.UTF8.GetBytes(secret)))
    {
        string signature = Convert.ToBase64String(
            hmac.ComputeHash(Encoding.UTF8.GetBytes(canonical))
        );

        string authorization =
            $"FYDO-HMAC-SHA256 {integratorId}:{signature}:{timestamp}";
    }
}
```

Note: DateTime.UtcNow.ToString(“o”) produces seven decimal places and will be rejected. Use the explicit format above.

Python

```
import hashlib
import hmac
import base64
from datetime import datetime, timezone

secret = ""          # Your HMAC Secret
integrator_id = ""   # Your Integrator ID

method = "POST"
path = "/WebhooksApi/Api/Patient/getPatientList".lower()
timestamp = datetime.now(timezone.utc).strftime("%Y-%m-%dT%H:%M:%S.%f")[:-3]
+ "Z"

body = '{"PageIndex": 1, "PageSize": 1000}'

body_hash = base64.b64encode(
    hashlib.sha256(body.encode("utf-8")).digest()
).decode()

canonical = f"{method}\n{path}\n{timestamp}\n{body_hash}"

signature = base64.b64encode(
    hmac.new(
        secret.encode("utf-8"),
        canonical.encode("utf-8"),
        hashlib.sha256
    ).digest()
).decode()

authorization = f"FYDO-HMAC-SHA256 {integrator_id}:{signature}:{timestamp}"
```

Note: datetime.isoformat() produces +00:00 and six decimal places, and will be rejected. Use the format above.

JavaScript (Node.js)

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

const secret = "";       // Your HMAC Secret
const integratorId = ""; // Your Integrator ID
```

```
const method = "POST";
const path = "/WebhooksApi/Api/Patient/getPatientList".toLowerCase();
const timestamp = new Date().toISOString();

const body = JSON.stringify({
    PageIndex: 1,
    PageSize: 1000
});

const bodyHashBase64 = crypto
    .createHash("sha256")
    .update(body, "utf8")
    .digest("base64");

const canonical = `${method}\n${path}\n${timestamp}\n${bodyHashBase64}`;

const signature = crypto
    .createHmac("sha256", secret)
    .update(canonical, "utf8")
    .digest("base64");

const authorization =
    `FYDO-HMAC-SHA256 ${integratorId}:${signature}:${timestamp}`;
```

PHP

```
<?php
$secret = "";       // Your HMAC Secret
$integratorId = ""; // Your Integrator ID

$method = "POST";
$path = strtolower("/WebhooksApi/Api/Patient/getPatientList");

$timestamp = (new DateTime('now', new DateTimeZone('UTC')))
    ->format('Y-m-d\TH:i:s.v\Z');

$body = json_encode([
    "PageIndex" => 1,
    "PageSize"  => 1000
], JSON_UNESCAPED_SLASHES);

$bodyHash = base64_encode(hash('sha256', $body, true));
```

```
$canonical =
    $method . "\n" .
    $path . "\n" .
    $timestamp . "\n" .
    $bodyHash;

$signature = base64_encode(
    hash_hmac('sha256', $canonical, $secret, true)
);

$authorization =
    "FYDO-HMAC-SHA256 {$integratorId}:{$signature}:{$timestamp}";
?>
```

Note: gmdate() cannot produce milliseconds and will be rejected. Use **DateTime** with the **v** format character as above.

Postman

Add the following as a Pre-request Script. It signs the request automatically, so you only need to set the secret and Integrator ID once.

```
const secret = "";       // Your HMAC Secret
const integratorId = ""; // Your Integrator ID

const method = pm.request.method.toUpperCase();
const path = pm.request.url.getPath().toLowerCase();
const timestamp = new Date().toISOString();

let body = "";
if (pm.request.body && pm.request.body.raw) {
    body = pm.variables.replaceIn(pm.request.body.raw);
}

const bodyHash = CryptoJS.SHA256(body);
const bodyHashBase64 = CryptoJS.enc.Base64.stringify(bodyHash);

const canonical = `${method}\n${path}\n${timestamp}\n${bodyHashBase64}`;

const signature = CryptoJS.enc.Base64.stringify(
    CryptoJS.HmacSHA256(canonical, secret)
);

pm.request.headers.upsert({
```

```
    key: "Authorization",
    value: `FYDO-HMAC-SHA256 ${integratorId}:${signature}:${timestamp}`
});
```

Error Responses

Authentication and authorisation failures return a JSON body in the following shape:

```
{
  "message": "unauthorized",
  "details": "<specific reason>"
}
```

Scenario	HTTP	message	details
Authorization header present but not FYDO-HMAC-SHA256	401	unauthorized	Invalid Authorization format
Malformed header, not exactly IntegratorId:Signature:Timestamp	401	unauthorized	Malformed Authorization header
Integrator ID is not a valid GUID	401	unauthorized	Invalid IntegratorId format
Invalid timestamp format	401	unauthorized	Invalid timestamp format. Expected UTC ISO 8601 format: yyyy-MM-dd'T'HH:mm:ss.fff'Z'
Timestamp outside the 5-minute window	401	unauthorized	Timestamp expired
Signature does not match	401	unauthorized	Signature mismatch
Integrator ID is a valid GUID but not recognised	401	unauthorized	Invalid IntegratorId
Endpoint not permitted for this integration	403	forbidden	endpoint not permitted
Source IP not on the hospital's allow list	403	unauthorized	Source IP is not allowed.

> Until the enforcement date, a request with no Authorization header is processed using Legacy API Key authentication rather than rejected. After the enforcement date, requests without a valid Authorization header will be rejected.

Troubleshooting a signature mismatch

- The request path in your canonical string is the path only, lowercase, with no scheme, host, or query string
- The timestamp in the canonical string matches the one in the Authorization header exactly
- The request body has not been modified after signing, including whitespace, field ordering, and encoding
- You are using the correct HMAC Secret for your Integrator ID
- The canonical string uses actual newline characters, not the literal text backslash-n

Troubleshooting other failures

- **Invalid timestamp format** – confirm your timestamp has exactly three decimal places and ends in Z.
- **Timestamp expired on requests you have just sent** – check your server clock is synchronised via NTP. Clock drift beyond 5 minutes will cause rejections.
- **403 on an endpoint that previously worked** – your hospital may have changed your endpoint permissions or enabled an IP allow list. Contact your hospital administrator, and confirm the public IP addresses your integration sends from.
- **401 after your hospital regenerated your HMAC Secret** – the previous secret is invalidated immediately. Obtain the new secret from your hospital administrator.

Enforcement Timeline

HMAC-SHA256 authentication will become mandatory for all FYDO API integrations.

Milestone	Date
HMAC authentication available	7 August 2026
Migration support period begins	7 August 2026
Enforcement date (HMAC mandatory)	**8 October 2026**

> **IMPORTANT** — From 8 October 2026, requests without a valid HMAC Authorization header will be rejected. Please ensure your integration is updated before this date.

If you need assistance with the migration, contact your hospital administrator or Altura support.

Best Practices

- **Store your HMAC Secret securely.** Treat it like a password. Do not hardcode it in source files, commit it to version control, or share it over email or chat. Use environment variables or a secrets manager.
- **Generate timestamps at request time.** Do not reuse timestamps across requests. Each request must have a fresh timestamp within the 5-minute window.
- **Keep your server clock synchronised.** Use NTP. Clock drift is the most common cause of unexpected timestamp rejections.
- **Sign the exact body you send.** Any difference between the signed body and the transmitted body, including whitespace, field ordering, or encoding, will cause a signature mismatch.
- **Use lowercase paths in your canonical string.** Always convert the request path to lowercase before signing.
- **Coordinate secret rotation with your hospital.** If your HMAC Secret is compromised, contact your hospital administrator immediately. Regenerating the secret invalidates the previous one straight away, with no overlap period, so your integration will stop authenticating until you apply the new secret. Agree a time with your hospital before they regenerate.
- **Do not log your HMAC Secret.** Ensure your application does not write the secret to log files, error reports, or monitoring systems.

Coming Soon

We are actively working on enhancements to the FYDO API platform:

- **Updated API documentation** – a new, comprehensive API reference covering all endpoints, request and response schemas, and the data each endpoint returns.
- **Incremental (delta) sync support** – the ability to retrieve only records changed since your last successful sync, rather than full data sets, significantly reducing payload sizes and run times for daily syncs. This is separate from the authentication change and will not affect the enforcement date, so please proceed with your HMAC migration independently.
- **Endpoint and integration improvements** – ongoing work to improve API performance, reliability, and experience across the platform.

Further details will be communicated as these become available.

Support

For questions about this guide, your credentials, or migration assistance, please contact:

- Your hospital administrator
- Altura support: support@alturahealth.com.au