📘 ubiDOCS Domain Driven API v3 Documentation

🗓️ Published: June 12, 2025

Welcome to the ubiDOCS DD API v3 Documentation. This guide is designed to help developers, integrators, and IT teams efficiently connect to Belgium's legally compliant waste transport platform through ubiDOCS Domain Driven API. Whether you're managing documents, handling signatures, or automating workflows, this documentation will walk you through every step with practical examples and clear guidance.

🧭 **Note: This documentation is not exhaustive!** It serves as a practical introduction to help developers understand key concepts and workflows. For complete and up-to-date technical details (including all endpoints, schemas, and parameters) please refer to the official Swagger UI, which remains the source of truth:

- **PROD:** api.ubidocs.ubidata.com/swagger
- **STAGING:** api-staging.ubidocs.ubidata.com/swagger

TABLE OF CONTENT

🚀 Introduction

ubiDOCS is a Transport Document Management API for the **Belgian waste transport ecosystem**, fully aligned with legal workflows from:

- **OVAM** (Flanders)
- **SPW** (Wallonia)
- **Leefmilieu Brussel** (Brussels)

It enables authorized actors **(Submitter, Waste Dealer, Waste Producer, Carrier, Waste Processor)** to manage Transport Documents digitally throughout the entire lifecycle:

✅ Create and enrich Transport Documents
✅ Collect signatures (with timestamp and geolocation)
✅ Real-time lifecycle transitions
✅ Generate compliant **PDFs and reports** (eID bon, eCMR, Annex VII, MATIS)
✅ Automate processes via **webhooks**

The API is **RESTful**, **event-driven**, and easily integrates with your TMS or ERP.

💡 Common Use Cases

- A **Waste Producer** creates a document → `ISSUED`
- A **Waste Dealer** approves it → `READY`

- A **Driver** picks up the goods at the `pickup` address → `PICKED_UP`
- The **Driver** starts the transport → `IN_TRANSIT`
- A **Waste Processor** confirms delivery at the `delivery` address → `DELIVERED`
- The **Driver** ends the transport → `TRANSPORT_ENDED`
- The **Waste Dealer** completes the document → `COMPLETED`

👩‍💻 Quick Start

- 💉 **Test Environment:** Request access
- 🔍 **Swagger:** Try the API
- 🛠 **Support:** support@ubidata.com

🔐 Authentication

ubiDOCS v3 uses **short-lived access tokens**. You can authenticate using:

1. API Key
2. Username + Password

🔑 API Key Authentication

Use for automated scripts and integrations.

Endpoint: `GET /v3/AuthenticateWithKey`

Header: `x-api-key: YOUR_API_KEY`

Optional query:

Parameter	Type	Description
`lifespanInMinutes`	integer	Token duration (default: 10 min)

Example:

```
curl -X GET "https://api.ubidocs.ubidata.com/v3/AuthenticateWithKey?lifespanInMinutes=15" \
  -H "x-api-key: USER_API_KEY"
```

👤 Username + Password Authentication

For interactive logins (e.g., dashboards).

Endpoint: `POST /v3/Authenticate`

Request:

```json
{
  "username": "jane.doe@example.com",
  "password": "StrongPassword123"
}
```

Example:

```
curl -X POST "https://api.ubidocs.ubidata.com/v3/Authenticate" \
  -H "Content-Type: application/json" \
  -d '{"username": "jane.doe@example.com", "password": "StrongPassword123"}'
```

🔄 Refresh Token

Endpoint: `GET /v3/RefreshToken`

Header: `Authorization: Bearer YOUR_ACCESS_TOKEN`

Example:

```
curl -X GET "https://api.ubidocs.ubidata.com/v3/RefreshToken" \
  -H "Authorization: Bearer YOUR_ACCESS_TOKEN"
```

> ⚠️ If the token expires, re-authentication is required.

🚪 Logout

Endpoint: `POST /v3/Logout`

Example:

```
curl -X POST "https://api.ubidocs.ubidata.com/v3/Logout" \
  -H "Authorization: Bearer YOUR_ACCESS_TOKEN"
```

🛑 API Key Management (Advanced)

For advanced scenarios, two additional endpoints are available:

- **`GET /v3/GenerateKey`** – Creates a new API key for automated system access.
- **`DELETE /v3/Revoke`** – Revokes an existing API key, permanently removing its access rights.

> ⚠️ **Warning:** Do **not revoke API keys** that belong to users of the **ubiTT Web Portal**. Doing so will prevent the user from logging into the **ubiDOCS web interface**, as these keys are tied to their account identity. If you're unsure which key to revoke, please [contact Support](mailto:support@ubidata.com) for assistance.

These endpoints are primarily intended for administrative and automation purposes. Misuse may disrupt user access to the platform.

🧩 Recommended Header: `x-app-info`

We strongly recommend that all API users include the `x-app-info` header when making requests to the ubiDOCS DD API.

This header should contain the name and version of your application or integration, such as:

```
x-app-info: ubiDOCS/1.15.1 (Build 900000654; ios iOS 18.4.1)
```

Including this header allows us to:

- Quickly identify your integration in our logs
- Improve support and troubleshooting by correlating requests with specific apps or environments
- Proactively communicate with you in case of errors, regressions, or behavioral changes

⚠️ Omitting this header may make it more difficult for our team to support you effectively in case of incidents or questions.

📄 Document Structure

Every interaction in the API revolves around a **Document**.

🧱 Core Fields

Field	Description
`documentId`	Internal UUID
`legalReferences`	Legal references post-issuance
`references`	External references (see section below)
`documentType`	`"ECMR"` or `"EWASTE"`
`documentState`	Status: `DRAFT`, `READY`, `COMPLETED`, etc.
`transportDate`	Planned date of transport
`goods`	Waste goods (see below)
`signatures`	Event-driven signatures with geolocation (except READY)
`customFields`	System-to-system metadata (see below)
`comments`	Operational notes with timestamps

Field	Description
`creationTimestamp`	Date created
`modificationTimestamp`	Last modified date
`creationUser`	Last modifying user

👥 Actors on the Document

1. `submitter` – creates the document
2. `dealer` – the Waste Dealer, who reviews and completes it
3. `consignor` – the Waste Producer who generates the waste
4. `carrier` – who transports the goods
5. `consignee` – the Waste Processor who receives and processes the waste

Each actor is an `Organization` with contact and location info.

📍 Transport Locations

Two key fields:

- `pickup`
- `delivery`

Each includes address, contact info, and optionally planned timing and opening hours.

📦 Goods

Each `good` contains:

- `goodId` , `description`
- `measures` : `{ type: WEIGHT | VOLUME | LENGTH | NUMBER, value, unit }` with the weight being mandatory before a Document is COMPLETED.
- `quantityCount` and `quantityUnit`
- list of `drCodes` and a `processingTechnique`
- `wasteInformation` with mandatory `euralCode` and `dangerousWasteInformation`
- Informational fields (optional and not shown in UI/PDF) such as `label` , `barcode` , `packingMethod` , `customerId`

✍️ Signatures

Each workflow stage adds a signature with:

Type	Action Trigger	Signed By
`SIGNATURE_FOR_READY_STATE`	Approval	Dealer
`SIGNATURE_FOR_PICKED_UP_STATE`	Pickup	Driver or Producer
`SIGNATURE_FOR_START_TRANSPORT`	Transport Start	Driver
`SIGNATURE_FOR_DELIVERED_STATE`	Delivery	Driver or Processor
`SIGNATURE_FOR_END_TRANSPORT`	Transport End	Driver

Each signature includes:

- Type of signature (Is it the Actor, the Driver or the Driver on behalf of the Actor?)
- Timestamp (always)
- GPS location (except for `READY`)
- User and Organization
- Optional `signValue` (SVG)

💬 Comments

Use comments to share key operational info with all Actors on the Document.

Each comment includes:

- `text`
- `category`:
  - `CARRIER_FREE_COMMENTS` for normal comments on the Driver's mobile during the transport
  - `TRANSPORT_INSTRUCTIONS` for specific transport Instructions to be shown to the Driver
  - `INTERRUPT` for a reason why a Transport is INTERRUPTED before DELIVERY.

🔒 Visibility: Comments are only visible to Actors using the ubiDOCS platform. They are not (yet) shared with other Actors via the OVAM IOP, and are only shown in the Change Log for operational traceability.

🧩 Custom Fields

Custom Fields are used to exchange **additional metadata** between systems.

Format:

```json
{ "key": "priority", "value": "High" }
```

They improve automation, filtering, and integration logic.

ℹ️ **If you plan to use Custom Fields**, please contact us so we can assist you in making them as useful as possible and help **avoid naming collisions**.

🔗 References

References are external or internal identifiers (e.g., project codes, ERP IDs).

They are **custom and extensible**, and will support richer linking in future releases.

🔁 Document Lifecycle

The following flow illustrates the document's progression, where the left side represents the happy path through the process, and the right side shows possible termination points at each stage.

```text
  DRAFT                (→ DELETED ❌)
→ ISSUED               (→ DELETED ❌)
→ READY                (→ CANCELLED ❌)
→ PICKED_UP            (→ INTERRUPTED ❌)
→ IN_TRANSIT           (→ INTERRUPTED ❌)
→ DELIVERED
→ TRANSPORT_ENDED
→ COMPLETED
→ CORRECTED
```

⚙️ Event-Driven Flow (Asynchronous with Webhooks)

ubiDOCS uses an **Event-Driven model for all write operations**. When your system performs an action—such as creating a document, issuing it, or submitting a signature—the **payload is validated immediately**, and the operation is queued for **asynchronous processing**.

The result is not returned immediately. Instead, your system must retrieve the outcome using **long polling** via:

```
GET /v3/webhook/{id}/status
```

ubiDOCS does **not push events** to your infrastructure. You control the retrieval of results when they're ready.

✅ Immediate Payload Validation

All write operations follow the same validation model:

HTTP Status	Meaning
`202 Accepted`	The payload is valid, and the operation is queued
`400 Bad Request`	The payload is malformed or fails validation
`401 Unauthorized`	The access token or API key is missing or invalid

Response Structure (202 Accepted)

```json
{
  "webhookId": "abc123-def456-ghi789"
}
```

The `webhookId` allows your system to track the result via polling.

🔁 Long Polling: `/v3/webhook/{id}/status`

To obtain the result of the queued operation, your system must call:

```
GET /v3/webhook/{id}/status
```

This endpoint uses **long polling**—ubiDOCS keeps the request open until the operation has finished processing or times out.

📦 Response Structure

```json
{
  "webhookId": "abc123-def456-ghi789",
  "httpStatusCode": 200,
  "documentId": "bfc7e6e2-1d4b-4cbe-a5ab-d71f1d5f90b0",
  "requestUrl": "PUT /v3/documents/bfc7e6e2-1d4b-4cbe-a5ab-d71f1d5f90b0/start_transport",
  "responseBody": "OK",
  "createdAt": "2025-05-19T12:34:56Z"
}
```

Field Descriptions:

Field	Type	Description
`webhookId`	`string`	Unique ID for tracking the async operation
`httpStatusCode`	`integer`	Final status code returned by the ubiDOCS backend for the requested action
`documentId`	`uuid`	ID of the document involved in the operation
`requestUrl`	`string`	The original API endpoint requested by the user
`responseBody`	`string`	Optional payload of the backend's internal response (nullable)
`createdAt`	`date-time`	Timestamp when the operation was completed

> 📝 This response is strictly technical: it confirms **what happened** to your original request—not the document content or status. For full document data, use `GET /v3/documents/{id}` .

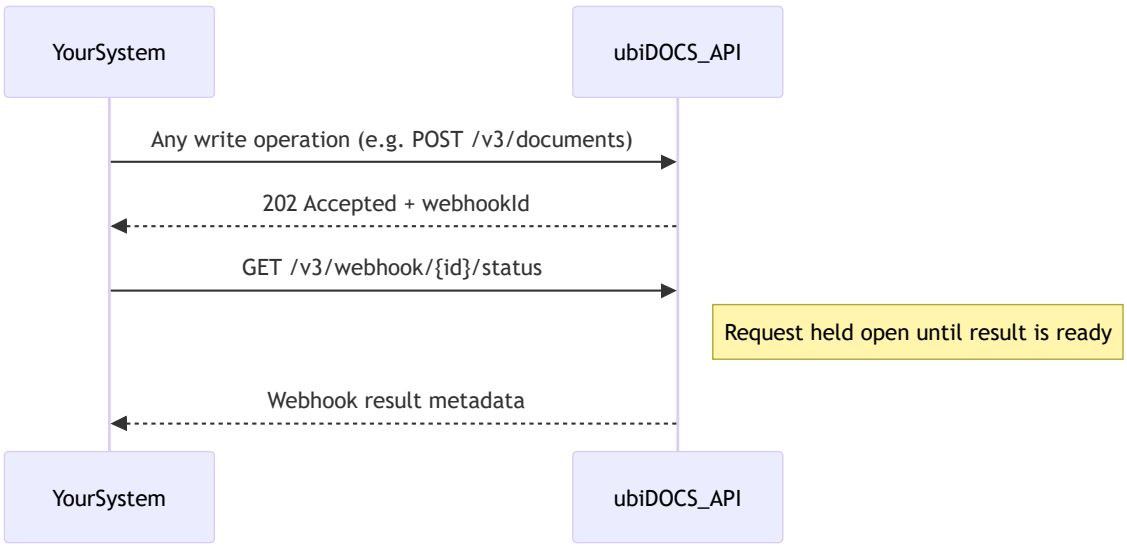
🆕 Document Creation Flow

When creating a document, the same pattern applies:

- You receive a `webhookId`, but not the `documentId`
- Poll `/webhook/{id}/status` until you receive the `documentId`
- Only then should you continue with further updates or transitions

---

🔄 Event Flow Diagram



🛠️ Best Practices for the Async Webhook

- ✅ Always store the `webhookId` from the `202 Accepted` response
- 🔁 Use `/v3/webhook/{id}/status` to long poll for the result
- ⏳ Let the request wait — ubiDOCS replies when the result is available
- 📄 Use the response to confirm completion; fetch full document data separately
- 🧠 Design your integration to resume workflow only once the webhook result is received