CourierData API – Full Developer Guide (v1.0.0)

Document purpose: This guide provides an end-to-end narrative for developers, QA engineers, solution architects, and technical writers who need to integrate with, test, or document the **CourierData** REST API. It blends formal specification with practical, field-tested advice so you can copy this directly into a Word document and export to PDF without further editing.

Table of Contents

Executive Summary

The **CourierData API** provides read-only access to courier identities and their extended details. Version **1.0.0** ships with two HTTP `GET` resources:

- `GET /couriers/list` – returns a collection of couriers with high-level identity fields (ID, first name, last name).
- `GET /couriers/details/{id}` – returns an extended record for a single courier (ID, address, salary).

This initial version prioritizes **simplicity**, **predictability**, and **low onboarding friction**. Authentication is via **HTTP Basic** and, while the sample host uses `http://localhost:8080`, deployments SHOULD run behind HTTPS in production. Responses are JSON by default; XML element names are also provided in the schema metadata for teams that prefer XML representations.

> **Who should read this?** Engineers building internal dashboards, integration services, ETL jobs, reporting utilities, and QA harnesses will find this guide practical. Architects will appreciate the sections on conventions, security, and lifecycle management.

At-a-Glance

- **Specification**: Swagger (OpenAPI 2.0)
- **Title**: `CourierData`
- **Version**: `1.0.0`
- **Base Path**: `/rest/courierdata/v1`
- **Host**: `localhost:8080` *(example; replace per environment)*
- **Schemes**: `http` *(HTTPS recommended in production)*
- **Security**: `basicAuth` (HTTP Basic)
- **Resources**: `couriers`

Base URLs & Environments

Although the specification lists `localhost:8080` for local development, typical deployments separate environments:

Environment	Base URL (example)
Local	`http://localhost:8080/rest/courierdata/v1`
Dev	`https://dev.example.com/rest/courierdata/v1`
QA	`https://qa.example.com/rest/courierdata/v1`

Environment	Base URL (example)
Prod	`https://api.example.com/rest/courierdata/v1`

> **Note**: The path **always** includes the version segment `/v1`. Client applications SHOULD treat the base path as immutable within a major version.

Authentication

The API uses **HTTP Basic Authentication**. Provide a valid username and password in the `Authorization` header.

```
Authorization: Basic <base64(username:password)>
```

- When authentication fails, the server returns `401 Unauthorized` and includes a `WWW-Authenticate` header describing the scheme.
- Credentials SHOULD be transmitted over **TLS** (HTTPS). Avoid sending credentials over plain HTTP outside isolated development setups.

Example (cURL)

```
curl -u myuser:mypassword \
  "https://api.example.com/rest/courierdata/v1/couriers/list"
```

Common HTTP Conventions

- **Methods**: All resources in v1 are `GET` and are therefore **safe** and **idempotent**.
- **Status Codes**: `200` on success, `401` on failed authentication. (Future versions may extend this set.)
- **Headers**: Standard HTTP headers (`Accept`, `Content-Type`, `Authorization`) apply.
- **Case Sensitivity**: Header names are case-insensitive. Field names in JSON are case-sensitive.
- **Time**: This API does not return timestamps in v1. If introduced later, the canonical format will be ISO-8601 with timezone designator (e.g., `2025-09-12T09:27:00Z`).

Media Types & Character Encoding

- **Default**: `application/json; charset=utf-8`
- **Alternative**: XML is conceptually supported (XML names are present in the schema), but JSON is the reference format in examples.
- Clients SHOULD set the `Accept` header explicitly when requesting non-default formats.

Idempotency, Safety & Caching

- **Idempotency**: All `GET` operations are idempotent—repeating calls yields the same outcome absent underlying data changes.
- **Caching**: No caching headers are defined in v1. Consumers MAY introduce client-side caching with sensible TTLs where appropriate.
- **Retries**: Transient network failures MAY be retried with exponential backoff (e.g., 1s, 2s, 4s, capped at 30s).

Rate Limiting & Fair Use

There is no explicit rate limit documented for v1. As a professional courtesy: - Keep request rates under **10 RPS** for shared environments unless instructed otherwise. - Batch reads rather than polling aggressively.

> **Tip**: If you anticipate high-volume access, coordinate with the platform team to ensure capacity planning and to obtain dedicated credentials.

Resources Overview

The `couriers` tag groups two endpoints:

1. **List Couriers** — returns a lightweight list of courier identities.
2. **Courier Details** — returns in-depth information for a single courier by numeric ID.

These endpoints are intentionally orthogonal: the list is optimized for browsing and selection; details are optimized for profile views and single-record processing.

Endpoints

GET /couriers/list

Summary: Retrieve all couriers with minimal identity attributes.

HTTP

```
GET {BaseURL}/couriers/list
```

Query Parameters: *None in v1*.

Authentication: Required (Basic).

Responses - `200 OK` — Array of `Courier` objects. - `401 Unauthorized` — Missing or invalid credentials. Header `WWW-Authenticate` present.

Example Request (cURL)

```
curl -i -u myuser:mypassword \
  "https://api.example.com/rest/courierdata/v1/couriers/list"
```

Example Response (200, JSON)

```
[
  {"_ID": 101, "Firstname": "Ava", "Lastname": "Turner"},
  {"_ID": 102, "Firstname": "Liam", "Lastname": "Ng"},
  {"_ID": 103, "Firstname": "Maya", "Lastname": "Khan"}
]
```

Example Response (200, XML)

```
<Couriers>
  <Courier>
    <_ID>101</_ID>
    <Firstname>Ava</Firstname>
    <Lastname>Turner</Lastname>
  </Courier>
  <Courier>
    <_ID>102</_ID>
    <Firstname>Liam</Firstname>
    <Lastname>Ng</Lastname>
  </Courier>
  <Courier>
    <_ID>103</_ID>
    <Firstname>Maya</Firstname>
    <Lastname>Khan</Lastname>
  </Courier>
</Couriers>
```

Notes & Best Practices - The list may be large in production contexts. Implement pagination in your UI even though v1 returns the full set, to future-proof for server-side paging. - Treat `_ID` as an opaque numeric identifier. Do not infer business meaning from the number itself.

GET /couriers/details/{id}

Summary: Retrieve the extended profile for a single courier.

HTTP

`GET {BaseURL}/couriers/details/{id}`

Path Parameters - `id` *(required, integer)* — Unique numeric identifier of the courier.

Authentication: Required (Basic).

Responses - `200 OK` — `CourierDetails` - `401 Unauthorized` — Missing or invalid credentials. Header `WWW-Authenticate` present.

Example Request (cURL)

```
curl -i -u myuser:mypassword \
  "https://api.example.com/rest/courierdata/v1/couriers/details/101"
```

Example Response (200, JSON)

```
{
  "_ID": 101,
  "Address": "221B Baker Street, London NW1 6XE, UK",
  "Salary": "€45,000"
}
```

Example Response (200, XML)

```
<Courier>
  <_ID>101</_ID>
  <Address>221B Baker Street, London NW1 6XE, UK</Address>
  <Salary>€45,000</Salary>
</Courier>
```

Validation & Edge Cases - `id` must be a positive integer. Non-numeric or negative values will not match the route and will be rejected by the server/router. - If a record does not exist for a given `id`, the server's behavior in v1 is undefined by the formal spec. Implement defensive handling in clients to treat unexpected non-200 responses as errors.

Data Models

Courier

A lightweight identity record.

Field	Type	Format	Notes
`_ID`	integer	int64	System-generated primary key.
`Firstname`	string	—	Given name of the courier.
`Lastname`	string	—	Family/surname of the courier.

JSON Schema (excerpt)

```
{
  "title": "Courier",
  "type": "object",
  "properties": {
    "_ID": { "type": "integer", "format": "int64" },
    "Firstname": { "type": "string" },
```

```
    "Lastname": { "type": "string" }
  }
}
```

CourierDetails

An extended profile with HR/administrative attributes.

Field	Type	Format	Notes
`_ID`	integer	int64	Matches the courier's identity `_ID`.
`Address`	string	—	Free-form mailing address. Consider normalizing in consuming systems if needed.
`Salary`	string	—	Human-readable salary figure including currency symbol. Not intended for arithmetic without parsing.

JSON Schema (excerpt)

```
{
  "title": "Courier",
  "type": "object",
  "properties": {
    "_ID": { "type": "integer", "format": "int64" },
    "Address": { "type": "string" },
    "Salary": { "type": "string" }
  }
}
```

> **Design Note**: `Salary` is modeled as a string in v1 to preserve formatting and avoid localization drift. Consumers that require numeric analysis SHOULD parse currency and amount explicitly.

Errors & Problem Handling

The API explicitly documents `401 Unauthorized` as a common error when authentication fails or is absent.

401 Unauthorized - **Headers**: `WWW-Authenticate: Basic realm="CourierData"` - **Body**: Implementations may return an empty body or a simple diagnostic structure.

Example (401)

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Basic realm="CourierData"
Content-Type: application/json; charset=utf-8

{"error":"unauthorized","message":"Authentication information is missing or invalid"}
```

> **Operational Tip**: Ensure your client sets the `Authorization` header on **every** request. Some HTTP libraries do not resend credentials automatically on redirects.

Security Considerations

- **Transport Security**: Always use HTTPS outside local development. Basic Auth credentials are otherwise exposed to the network.
- **Secrets Handling**: Do not hardcode credentials in source control. Use environment variables or secret managers.
- **Least Privilege**: Create dedicated credentials per application and environment. E.g. CTF{N3v3rComm!tY0urP@55w0Rds}
- **Logging**: Avoid logging full URLs that include credentials (e.g., `https://user:pass@host/...`).

Observability & Diagnostics

- **Correlation IDs**: While not mandated in v1, clients MAY include a header such as `X-Correlation-Id: <uuid>` to trace requests across services.
- **Timestamps**: Client logs SHOULD record request start/stop times, status codes, and payload sizes for performance baselining.
- **Health Checks**: There is no public health endpoint in v1. Use application-level canaries where necessary.

Testing, Mocking & Tooling

- **cURL** for quick, scriptable calls (examples above).
- **HTTPie** for readable CLI testing:

  ```
  http -a myuser:mypassword GET
  https://api.example.com/rest/courierdata/v1/couriers/list
  ```

- **Postman/Insomnia**: Import the Swagger (see Appendix A) to auto-generate requests.
- **Mocks**: If you need deterministic data for UI development, record sample responses and serve via a mock server (e.g., WireMock).

Versioning & Deprecation Policy

- **Semantic Versioning**: The API follows `MAJOR.MINOR.PATCH`. This document covers **1.0.0**.
- **URL Versioning**: `/v1` is part of the base path; breaking changes will appear under `/v2`.
- **Deprecation**: If fields or endpoints are deprecated in a MINOR release, they will be maintained for at least one full MINOR cycle with warnings before removal in the next MAJOR version.

Change Log

- **1.0.0** – Initial public availability with two endpoints: list and details.

FAQ

Q: Does `/couriers/list` support filtering or sorting?
A: Not in v1. Clients should filter/sort locally. This keeps server behavior stable while usage patterns mature.

Q: Why is `Salary` a string instead of a number?
A: Formatting and currency symbols vary. A string preserves intent and avoids locale issues. Parse it when numerical operations are required.

Q: Will there be POST/PUT/PATCH endpoints?
A: Possibly in a future major version. v1 is intentionally read-only to simplify governance and access control.

Q: Is XML supported?
A: The schema provides XML names, and examples are shown. JSON remains the canonical, fully supported representation in v1.

Glossary

- **Basic Auth**: An HTTP auth scheme transmitting a Base64-encoded `username:password` pair in the `Authorization` header.
- **Idempotent**: An operation that can be applied multiple times without changing the result beyond the initial application.
- **Resource**: A logical entity exposed by the API (e.g., `Courier`).

Appendix A: Full Swagger (OpenAPI 2.0)


```
{
  "swagger": "2.0",
  "info": {"title": "CourierData", "version": "1.0.0"},
  "host": "localhost:8080",
  "basePath": "/rest/courierdata/v1",
  "schemes": ["http"],
  "paths": {
    "/couriers/list": {
      "get": {
        "tags": ["couriers"],
        "responses": {
          "200": {
            "description": "successful operation",
            "schema": {
              "items": {"$ref": "#/definitions/Courier"},
              "xml": {"name": "Couriers"},
              "type": "array"
            }
          },
          "401": {"$ref": "#/responses/UnauthorizedError"}
        }
      }
    },
    "/couriers/details/{id}": {
      "get": {
        "tags": ["couriers"],
        "parameters": [
          {"name": "id", "in": "path", "required": true, "type": "integer"}
        ],
        "responses": {
          "200": {
            "description": "successful operation",
```

```
            "schema": {"$ref": "#/definitions/CourierDetails", "xml":
{"name": "Couriers"}}
           },
           "401": {"$ref": "#/responses/UnauthorizedError"}
         }
       }
     }
   },
   "definitions": {
     "Courier": {
       "title": "Courier",
       "xml": {"name": "Courier"},
       "type": "object",
       "properties": {
         "_ID": {"type": "integer", "format": "int64"},
         "Firstname": {"type": "string"},
         "Lastname": {"type": "string"}
       }
     },
     "CourierDetails": {
       "title": "Courier",
       "xml": {"name": "Courier"},
       "type": "object",
       "properties": {
         "_ID": {"type": "integer", "format": "int64"},
         "Address": {"type": "string"},
         "Salary": {"type": "string"}
       }
     }
   },
   "responses": {
     "UnauthorizedError": {
       "description": "Authentication information is missing or invalid",
       "headers": {"WWW-Authenticate": {"type": "string"}}
     }
   },
   "securityDefinitions": {"basicAuth": {"type": "basic"}},
   "security": [{"basicAuth": []}],
   "tags": [{"name": "couriers", "description": ""}]
}
```

Final Notes for Document Consumers

- You can paste this entire guide into Word. The tables and code blocks preserve well when using a monospaced font for code (e.g., Consolas).
- Before publishing externally, replace example domains with your real endpoints and scrub any environment-specific secrets from snippets.