ELTO Employers' Liability Tracing Office

ELD NextGen Policy API

Outline Specification

22nd September 2026
Version 1.2

Contents

1 Purpose of this Document 2

2 Conventions 2

2.1 Authentication 2

2.2 Insurer Selection 2

2.3 Enumerations 2

2.4 Common response codes 2

2.5 Error response shape 3

2.6 Policy Flow 3

3 Endpoints 4

3.1 UPSERT Policies (Direct) 4

3.2 DELETE Policies (Direct) 5

3.3 UPLOAD File 6

3.3.1 Post Policy File 7

3.3.2 GET Submission Status 9

3.4 Test Endpoints 11

4 Object Schemas 11

5 Validation Rules 11

5.1 Overview 11

5.2 Policy-level fields 12

5.3 Employer-level fields 12

5.4 Duplicate detection 12

5.5 File format validation 12

5.6 DELETE validation 13

5.7 Limits and operational behaviour 13

1 Purpose of this Document

This document is a summary companion to the published Test API. The Developer Portal contains the authoritative endpoint specification and downloadable OpenAPI definition. The Schema Guide and File Transfer Guide provide field and file guidance.

For business context and a narrative summary of what is changing, see: **ELD NextGen Policy Management Summary of Changes document**.

2 Conventions

2.1 Authentication

To use the API credentials need will need to be created in ELD Portal to then produce a short-lived bearer token obtained using the private key corresponding to the public certificate registered against the API credential in the ELD portal.

Test and production credentials (once live) / tokens are separate. Full certificate and token-generation instructions are available in the Developer Portal

2.2 Insurer Selection

The supplier will be inferred from the credentials used. The insurer will be provided explicitly in the path. If authenticated credentials are not authorised for the insurerId in the path, the API returns 403 Forbidden.

2.3 Enumerations

OpenAPI definition does not contain ant enums. All enum values are **case-insensitive** — the API accepts any casing and normalises internally.

2.4 Common response codes

Response codes are endpoint-specific.

All validation errors, including duplicate policy records, invalid dates, invalid ERNs/CRNs and missing required fields, are returned as `400 Bad Request` with a structured list of validation errors. `409 Conflict` and `422 Unprocessable Entity` are not used for policy validation errors.

Code	Meaning
200	Successful operation / validation outcome as described by the endpoint.
400	Malformed input or validation failure.
401	Unauthenticated.
403	Credential not authorised for requested insurer.
404	Upload not found (status endpoint).

413	Direct policy body exceeds 2 MB.
429	Rate limit exceeded; obey Retry-After.
500	Unexpected internal error.
503	Health endpoint reports service unhealthy.

2.5 Error response shape

API errors use an ApiErrorResponse envelope containing requestId and error { code, message, details }. Validation issues may be returned under error.details. The 413 and 429 responses use their dedicated published shapes.

```
{
  "requestId": "abc-123",
  "error": {
    "code": "VALIDATION_FAILED",
    "message": "Validation failed",
    "details": { "issues": [ { "path": "policies[0].policyNumber", "description": "policyNumber
is a required field" } ] }
  }
}
```

2.6 Policy Flow

Direct API submissions are validated synchronously. If validation fails, no policies in the direct submission are accepted or queued. Where validation passes, the API 200 response confirms that the submission has been accepted and queued for ingestion.

File-based submissions are accepted for processing after the file upload has completed. Validation and ingestion then take place asynchronously. Valid policies are ingested individually, and invalid policies are rejected individually, with validation errors available through the status endpoint and the ELD policy management area.

Acceptance or queuing does not mean the policy change will be immediately visible in all ELD features. There may be a short delay before changes appear in policy search, reporting, extracts or downstream ELD functions.

NOTE: Policies must be supplied in full. Partial updates are not supported. Where a policy with the same Policy key already exists, the newly accepted policy submission replaces the previous version of that policy once ingestion is complete.

3 Endpoints

The following endpoints are available in Test (where no data is persisted) and Production.

Method	Relative path	Purpose
GET	/health	Authenticated service health check.
POST	/insurers/{insurerId}/policies	Direct policy UPSERT validation; no persistence.
DELETE	/insurers/{insurerId}/policies	Direct policy DELETE-key validation; no persistence.
POST	/insurers/{insurerId}/policies/file	Request a SAS URL for a test policy file upload.
GET	/insurers/{insurerId}/policies/status/{uploadId}	Get asynchronous file-upload status and validation results.

3.1 UPSERT Policies (Direct)

POST **/insurers/<insurerId>/policies**

Submit one or more policies for upsert. Validation is synchronous.

Behaviour

All-or-nothing. If any policy in the submission fails validation, the entire submission is rejected and no policies are ingested.

Request body:

Request body is an array of 1-5,000 PolicyInput objects and is subject to a 2 MB request-body limit.(see schema in section 4).

```
[
  {
    "policyNumber": "EMP-2024-001",
    "supplementaryPolicyReference": "BROKER-REF-12345",
    "coverStartDate": "2024-01-01",
    "coverEndDate": "2024-12-31",
    "policyholderName": "Acme Corporation Ltd",
    "firstClaimSettlementDate": "2024-02-01",
    "originalInsurerName": "Example Insurer Ltd",
    "firstSuppliedAt": "2024-01-15",
    "employers": [
      {
        "name": "Acme Corporation Ltd",
        "erns": [
          "123/AB45678"
        ],
        "crn": null,
        "adjustedCoverStartDate": null,
        "adjustedCoverEndDate": null,
        "hmrcErn": "123/AB45678",
        "legacyPolicyId": "1001",
        "multilineAddress": {
          "addressLine1": "123 Business Park",
          "addressLine2": "Suite 100",
          "cityTown": "London",
          "county": "Greater London",
          "country": "United Kingdom",
          "postCode": "SW1A 1AA"
        }
```

```
        },
        {
          "name": "Acme Manufacturing Division",
          "erns": [
            "456/CD90123"
          ],
          "crn": null,
          "adjustedCoverStartDate": null,
          "adjustedCoverEndDate": null,
          "hmrcErn": "456/CD90123",
          "legacyPolicyId": "1002",
          "multilineAddress": {
            "addressLine1": null,
            "addressLine2": null,
            "cityTown": null,
            "county": null,
            "country": null,
            "postCode": null
          }
        },
        {
          "name": "Acme Retail Division",
          "erns": [
            "789/EF45678"
          ],
          "crn": null,
          "adjustedCoverStartDate": "2024-03-01",
          "adjustedCoverEndDate": null,
          "hmrcErn": "789/EF45678",
          "legacyPolicyId": "1003",
          "singleLineAddress": "789 Retail Park, Birmingham, B1 1AA"
        }
      ]
    }
  ]
```


Responses:

Code	Meaning
200 OK	All policies accepted and queued for ingestion.
400 Bad Request	One or more validation errors**. No policies have been accepted**. Response body contains all errors as an array of ValidationError objects..

Note

A `200 OK` response confirms that the submission has passed validation and has been queued for ingestion. It does not guarantee that the policy changes are immediately visible in ELD search, reporting or downstream outputs. Use the status endpoint to confirm processing completion.

3.2 DELETE Policies (Direct)

DELETE **/insurers/<insurerId>/policies**

Submit one or more policies for deletion. Only key fields are required policyNumber, coverStartDate and policyholderName; supplementaryPolicyReference is optional. Submitting the policy key will delete the whole policy represented by that key, including all employers.

Behaviour

All-or-nothing. If any entry fails validation, the entire submission is rejected.

Request body:

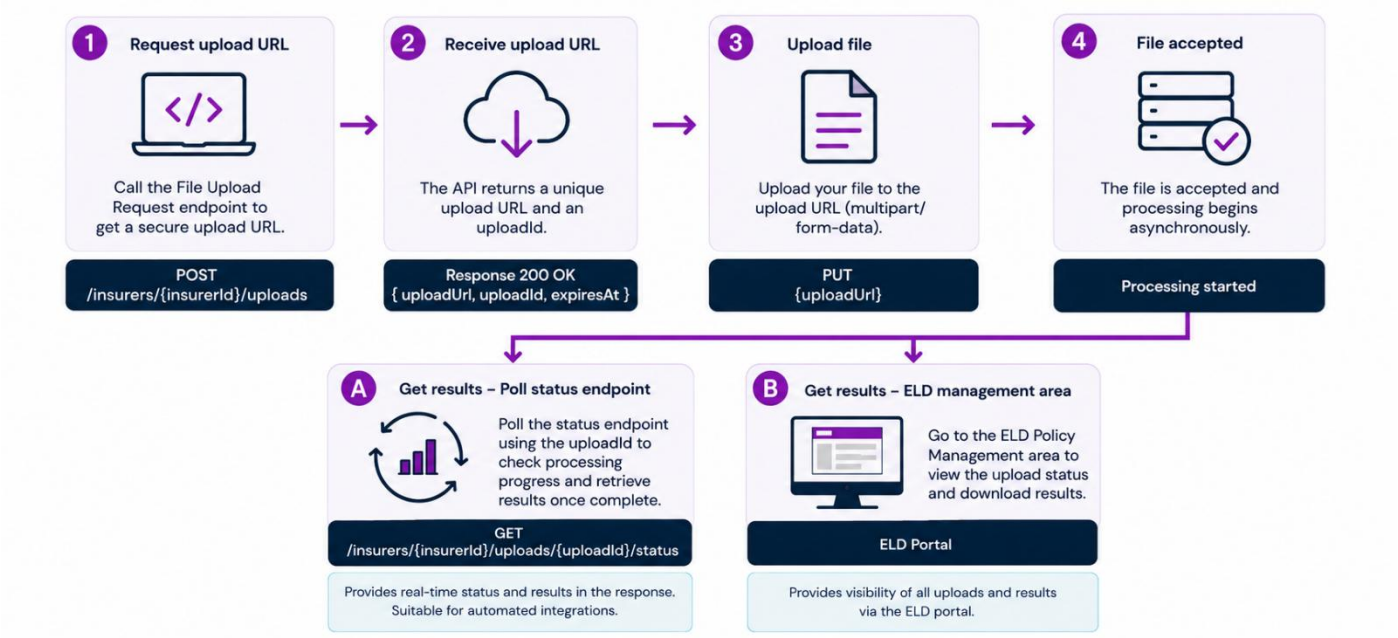
```
[
  {
    "policyNumber": "POL001",
    "supplementaryPolicyReference": "SPR-ABC",
    "coverStartDate": "2015-01-01",
    "policyholderName": "Example Policyholder Ltd"
  }
]
```

Responses: As UPSERT (Direct).

3.3 UPLOAD File

The file upsert processes uses a secure URL called via the API to then able the uploading of a file. Uploading a file is a three step process:

1. POST to `/file` → receive `uploadId` + `uploadUrl`.
2. Upload the file directly to `uploadUrl` — processing will start automatically after the upload completes.
3. Poll `uploadId` until status is one of the terminal states: `COMPLETE`, `REJECTED`, or `SYSTEM_ERROR`.



File Format

The API accepts fileType JSON or TEXT. JSON follows the PolicyInput structure. TEXT uses the pipe-delimited structure defined in the ELD File Transfer Guide, including a header row and preserved blank columns.

Fields Containing Pipe Characters

The pipe character (|) is used as the field separator in the text file format. If a field value itself contains a pipe character, the whole field must be enclosed in double quotes. This prevents the pipe character from being interpreted as a new column (following standard formatting in Excel)

For example, if an employer name is:

```
Employer | Trading Ltd
```

it must be supplied as:

```
"Employer | Trading Ltd"
```

Example row:

```
POL12345|SUPP01|2026-01-01|2026-12-31|Policyholder Ltd||"Employer | Trading
Ltd"||1 High Street|Business Park|London||UK|SW1A 1AA|||123/AB12345
```

If a field contains double quotes, each double quote must be doubled. For example:

```
Employer "North" Ltd
```

must be supplied as:

```
"Employer ""North"" Ltd"
```

3.3.1 Post Policy File

POST **/insurers/<insurerId>/policies/file**

Initiate a file-based policy upsert . The endpoint returns an upload ID and a secure upload URL. Uploading the file to the URL will queue the file for ingestion. At this point the file will be made read-only and should not be attempted to be modified once uploaded.

Behaviour

Valid policies are ingested individually; invalid policies are rejected individually. There is no minimum pass rate.

Request body:

```
{
"fileName": "TestPolicyBatch001",
"fileType": "TEXT",
"operationType": "UPSERT"
}
```

Field	Description
file name	Name used to identify the file. Letters and numbers only, with no file extension.
fileType	File format. JSON or TEXT.
operationType	Operation to perform. Currently UPSERT only for file upload.

Response (200 OK):

```
{
  "uploadId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "uploadUrl": "https://...",
  "expiresAt": "2026-04-21T17:00:00Z",
  "timestamp": "2026-04-20T16:00:00Z"
}
```

Code	Meaning
200 OK	Upload initiated — use uploadId to track status.
400 Bad Request	Invalid file type or request.

3.3.2 GET Submission Status

GET **/insurers/<insurerId>/policies/status/{uploadId}**

The status response returns validation errors for rejected policies and records within a File submission. Each validation error identifies the location of the issue using a structured path, together with a human-readable description of the error and providing sufficient detail for the submitting organisation to identify the affected record(s) in the original file. This may include (where appropriate) the policy key, employer name, policy index, employer index, field path and error description. Where applicable, row numbers may also be provided for pipe-delimited text files.

Validation results can be viewed through the API status response and the ELD policy management area.

NOTE: Poll this endpoint no more than once every 30 seconds. Most uploads will complete within a few minutes, although processing time will depend on file size and system load. Excessive polling may result in rate limiting. If a 429 Too Many Requests response is received, do not retry until the number of seconds specified in the Retry-After response header has elapsed.

Validation error detail

The system does not issue a separate return file after processing. For file-based submissions, validation results are retrieved using the status endpoint or viewed in the ELD policy management area.

The status response includes counts of total, accepted and rejected policies, together with validation errors for rejected records. Each validation error identifies the affected item using a structured path and a human-readable error description. For pipe-delimited text files, the response should also identify the affected row where this is available, so that insurers can locate and correct the source record.

Response (200 OK):

```
{
  "id": 0,
  "uploadId": "<GUID>",
  "timestamp": "<UTC ISO timestamp>",
  "uploadType": "WEB/SYNC/API_FILE/API_DIRECT",
  "fileName": "",
```

```
  "fileType": "JSON/TEXT",
  "operationType": "UPSERT/DELETE",
  "status": "WAITING_FOR_FILE/QUEUED/PROCESSING/COMPLETE/REJECTED/SYSTEM_ERROR",
  "fileRejectionReason": "<string>",
  "totalPolicies": 1,
  "validPolicies": 1,
  "rejectedPolicies": 1,
  "validationErrors": [
    {
      "description": "Error description",
      "path": "policy[0].employers[0].name"
    }
  ],
  "processingStartedAt": "<UTC ISO timestamp>",
  "validationCompletedAt": "<UTC ISO timestamp>",
  "processingCompletedAt": "<UTC ISO timestamp>"
}
```

Status values:

Status	Description
`WAITING_FOR_FILE`	Upload record created; waiting for file to be uploaded to the SAS URL.
`QUEUED`	File received; queued for processing.
`PROCESSING`	Validation and ingestion in progress.
`COMPLETE`	Processing complete.
`REJECTED`	File rejected (e.g. invalid file format) — no policies processed.
`SYSTEM_ERROR`	Unexpected internal error — contact support.

Code	Meaning
200 OK	Status returned.
404 Not Found	Upload ID not found, or insufficient access (e.g. owned by a different supplier).
429 Too Many Requests	Rate limit exceeded. Wait for the number of seconds specified in the Retry-After response header before retrying the request

Rate Limiting

If the API rate limit is exceeded, the endpoint returns 429 Too Many Requests. The response includes:

- Retry-After — the number of seconds to wait before retrying the request.

- X-RateLimit-Remaining-Minute — the number of requests remaining in the current minute window.

```
{
  "error": {
    "code": "RATE_LIMIT_EXCEEDED",
    "message": "Too many requests. Please retry after the period indicated in the Retry-After header."
  }
}
```

3.4 Test Endpoints

Full test versions of each of the core endpoints are available via the Developer Portal. . They will behave identically to the live endpoints, including validation, but do not persist policy data. Test-file data is retained only in the isolated test environment for validation/status processing.

Test endpoints are intended to support development and integration testing. The endpoints use separate credentials so as not to risk accidental inclusion of test data.

Test endpoint credentials are available via the ELD Portal and full documentation available via the Developer portal.

4 Object Schemas

Full object schemas are available in the Developer Portal. Use the ELD Policy Schema Guide for field purpose and legacy mapping, and use the OpenAPI definition from the Developer Portal for machine-readable required fields, constraints and endpoint request/response shapes.

5 Validation Rules

5.1 Overview

All policy submissions are validated. For direct submissions, all errors across all policies are collected and returned together in a single response. For file submissions, errors are available via the status endpoint (and policy management area of the ELD portal) after validation completes.

Note

The validation rules below are indicative of what will be enforced. The final validation rule set will be confirmed when the full API specification is released.

5.2 Policy-level fields

Field	Required	Rules
`policyNumber`	**Required**	Must not start or end with whitespace.
`supplementaryPolicyReference`	Optional	If provided, must not start or end with whitespace.
`coverStartDate`	**Required**	ISO 8601 YYYY-MM-DD; must be after 01/01/1900; must be before coverEndDate.
`coverEndDate`	**Required**	ISO 8601 YYYY-MM-DD; after 01/01/1900; after coverStartDate.
`policyholderName`	**Required**	—
`firstClaimSettlementDate`	Optional	ISO 8601 YYYY-MM-DD; only valid if coverEndDate is before the original ELD go-live date (2011-04-01); value must be on or after the original ELD go-live date (2011-04-01).
`employers`	**Required**	At least one employer must be present.

5.3 Employer-level fields

Field	Required	Rules
`employers[].name`	Required	—
`employers[].erns`	Optional	Array; each ERN must be in the correct format.
`employers[].crn`	Optional	Must be in the correct format.
`employers[].singleLineAddress`	Required (one of)	Exactly one of singleLineAddress or multilineAddress must be provided.
`employers[].multilineAddress`	Required (one of)	Providing any subfield counts as supplying a multiline address.
`employers[].multilineAddress.postCode`	Optional	Must be in the correct format.
`employers[].adjustedCoverStartDate`	Optional	Used when a specific employer's cover started after the overall policy. ISO 8601 YYYY-MM-DD; after 01/01/1900; after coverStartDate; before adjustedCoverEndDate if both provided, otherwise before policy coverEndDate.
`employers[].adjustedCoverEndDate`	Optional	Used when a specific employer's cover ended before the overall policy. ISO 8601 YYYY-MM-DD; before coverEndDate; after adjustedCoverStartDate if both provided, otherwise after policy coverStartDate.

5.4 Duplicate detection

Rule	Behaviour
Duplicate policy within submission	Two policies with the same policyNumber, supplementaryPolicyReference and coverStartDate— the first is processed; subsequent duplicates are rejected.

5.5 File format validation

Format	Check
JSON	File must be valid JSON; if not, the entire file is immediately rejected.
Pipe-delimited text	File must be parseable as pipe-delimited; header row must be present and match expected columns exactly; each data row must have the correct column count. Invalid format results in immediate rejection of the whole file. Fields containing the pipe character must be enclosed in double quotes following Excel CSV conventions; embedded double quotes must be doubled ("").

5.6 DELETE validation

DELETE submissions follow the same validation logic as an UPSERT.

Submission route	Validation timing	Failure behaviour
Direct DELETE API	Synchronous	All-or-nothing. If one entry fails, none are processed.

NOTE: When a Delete submission is processed if the format is valid it will be processed. There is no response on the number of records deleted. This will need to be validated separately via your policy reports.

Policy-level fields (DELETE)

Field	Required	Rules
`policyNumber`	Required	Must not start or end with whitespace.
`supplementaryPolicyReference`	Optional	If provided, must not start or end with whitespace.
`coverStartDate`	Required	ISO 8601 YYYY-MM-DD; after 01/01/1900; before coverEndDate.

(No employer-level validation applies — the DELETE endpoint accepts key fields only)

Duplicate detection (DELETE)

Rule	Behaviour
Duplicate policy within submission	Two entries sharing the same policyNumber, supplementaryPolicyReference and coverStartDate— the first is processed; subsequent duplicates are rejected.

5.7 Limits and operational behaviour

Item	Current published value / behaviour
Direct JSON body	2 MB maximum (413 if exceeded).
Direct array size	1-5,000 items in the OpenAPI schema. Getting Started currently says no set record-count limit; clarify this as a direct-request limit versus file-record guidance.
Employers per policy	1-1,000.
ERNs per employer	Up to 100.
SAS URL	Write-only; 60-minute lifetime.
Polling	No more than once every 30 seconds.
Rate limiting	60 API requests per minute and 5,000 requests per day. If exceeded, the API will return a 429 Too Many Requests