ELTO Employers' Liability Tracing Office

ELD NextGen: Technical Deep Dive

Exploring the technical detail

18th June 2026

In partnership with: DIGITAL WONDER LAB SW

Welcome

ELD NextGen | Technical Deep-dive

Housekeeping & Compliance

Competition law compliance

- ELTO is committed to competition law compliance.
- The consequences of non-compliance are grave – both organisations and individuals can be fined, and individuals may even be sent to jail.
- All ELTO and industry meetings, formal and informal, must avoid areas that might fall foul of competition law.
- Examples include discussion of arrangements or prices and standard conditions, the exchange of commercially sensitive market information or the sharing-out of markets and bid-rigging.
- If the meeting Chair feels that the meeting is in danger of breaching competition law, they may bring the discussion to an immediate close, terminate the meeting altogether, or ask individual members to leave.
- More details written guidance is available on request.

Any questions?

Please submit questions via the Q&A function

We will address as many questions as possible during the session

Please note: this session is being recorded and will be shared with registered attendees

Presenters & Roles

Hollie Flaherty

Programme Manager

Session Chair

ELTO

Sam Reevey

Delivery Lead

Speaker

Softwire Delivery Partner

Emily Feinson

Data Specialist

Speaker

Softwire Delivery Partner

Stefan Matei

Senior Developer

Speaker

Softwire Delivery Partner

Session Overview

- Where we are now
- Why the platform is evolving
- API Specifications & File Structure
- Data Model
- Policy processing flow & error handling
- Testing approach
- Q&A
- Next steps

ELD NextGen – Where we are now

A structured programme of discovery, design and validation has led to a controlled move into delivery

Discovery & Foundations

- Deep technical discovery completed with Softwire
- Key pain points and needs identified
- Core platform and architectural approach defined

Design & Validation

- Early design shaping and refinement
- Member and user workshops and engagement
- Approach tested against real-world operating models

Delivery & Transition

- Programme has moved into delivery phase
- Core platform build underway
- Transition and member preparation phase beginning

This gives us a validated foundation to move confidently into delivery and transition

Why we are evolving the platform

Addressing platform risk while improving reliability, simplicity and long-term sustainability

Improve reliability & data quality

Simplify & streamline policy submission

Enhance auditability & traceability

Reduce long-term operational burden

Align with modern technology standards

Provide a managed transition for members

Data Model – Summary of Changes

The updated policy data model is simplified, and aims to more directly represent policies as stored by insurers' systems

1. The most significant structural change is the **removal of the parent/child policy hierarchy**. Currently, a policy covering multiple employers is modelled as a primary (parent) policy record with one or more child records linked. Going forward, a policy will be a single record containing all its covered employers.
2. Updated "compound key" fields (for the above reason).
3. Some fields have been removed:

Field	Reason
OriginalInsurerId	Transfer history is now maintained internally.
DummyPolicy	No longer required.
PolicyType	Removed as part of consolidating employers into a single policy.
ERNExempt	No longer required.
SiteId	FSNs have been removed from the model.
AddressTypeLookUpId	Address type is determined implicitly from which fields are populated.
addressLine3 / addressLine4	Address structure simplified.

4. Some fields have been renamed:

Old name	New name	Why renamed
EffectiveDate	firstClaimSettlementDate	Aligns with ICOBS terminology. Also moves from employer level to policy level.
BrokerDARef	supplementaryPolicyReference	Makes clear this field forms part of the policy key. A change in this field represents a different policy, not a different version of the same policy. It should not be treated as a free-form reference field.

Data Model – Removal of Parent/Child Policies

Consolidation of parent/child to a single combined record, for a better representation of reality and simplified management

What was previously a policy family – i.e. a parent record plus child record(s) - will now no longer be separate entities; they will all be part of the same policy record. **One policy record, with (optionally) multiple employers.** All employers have equal standing — there is no concept of "primary" or "parent".

This requires an update to the policy key fields (the fields which uniquely define a policy – the "compound key").

Field	Old key	New key
Insurer ID	✓	✓
Policy Number	✓	✓
Supplementary Policy Reference (formerly BrokerDARef)	✓	✓
Cover Start Date	✓	✓
Cover End Date	✓ (partial)	Removed
Employer Name	✓	Removed
Policy Type	✓	Removed

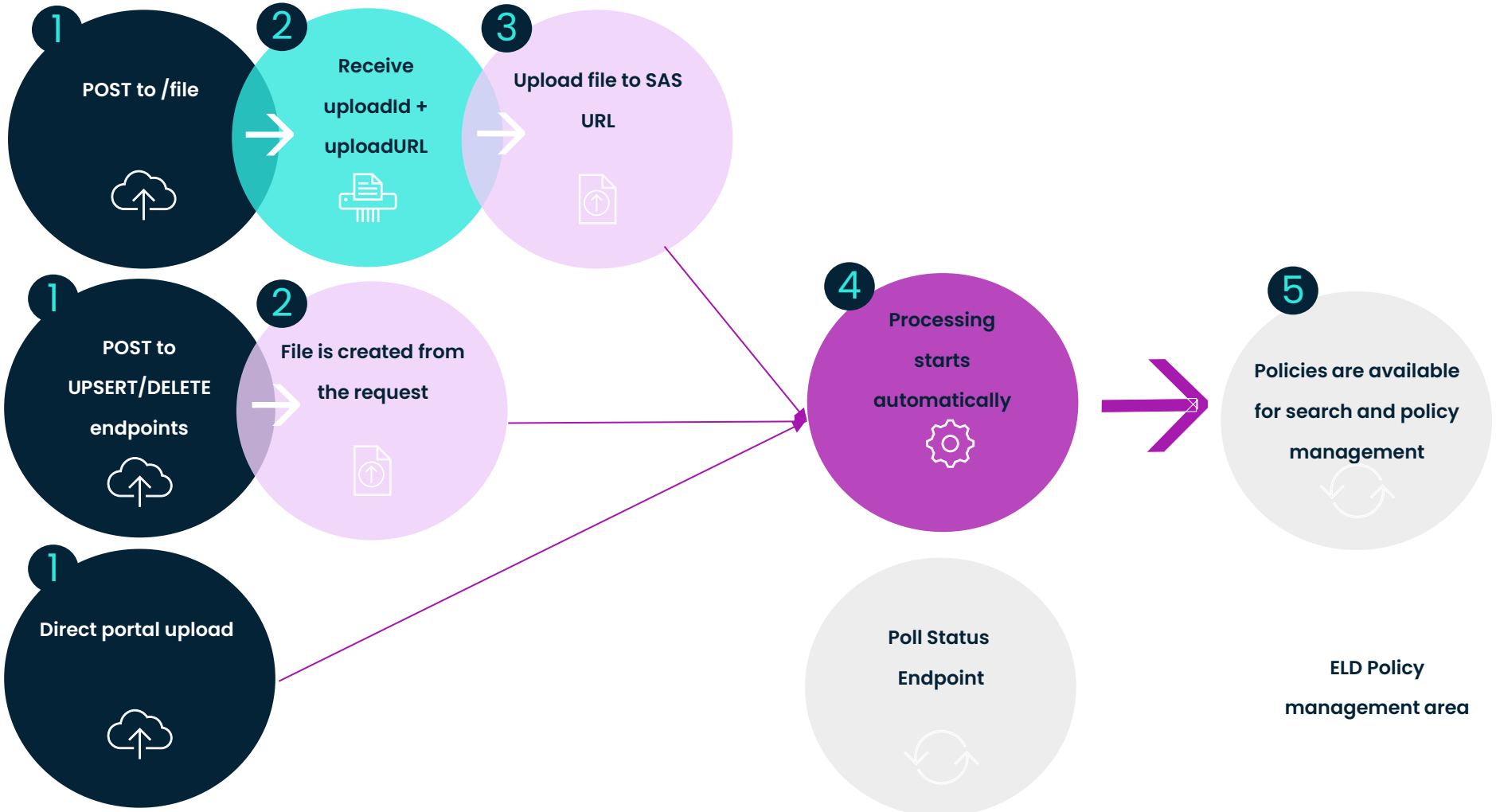
Data Model – Updated Policy Structure

The full policy data model contains only the relevant fields, and makes purpose and relationships clear through the structure

Field	Field (employer)	Field (address)	Notes
(Key field) Policy Number			
(Key field) Supplementary Policy Reference (optional)			• Renamed from BrokerDARef
(Key field) Cover Start Date			
Cover End Date			
Policyholder Name			• Moved from being per-employer to per-policy
First Claim Settlement Date (optional)			• Renamed from EffectiveDate • Moved from being per-employer to per-policy
Employers	Name		
	ERNs (optional)		• Multiple ERNs now permitted via any upload route
	CRN (optional)		• Optional new field to support future functionality
	Single Line Address		• Exactly one of single line or multi line address must be supplied
	Multi Line Address	Address Line 1	
		Address Line 2	
		City/Town	
		County	
		Country	
		Postcode	
	Adjusted Cover Start Date (optional)		• Used only where the employer has a different cover period to the main policy
	Adjusted Cover End Date (optional)		

Processing Flow

Submitted policies will follow a processing pipeline — validated against a defined set of field-level rules, with real-time status tracking available via the portal and API



File processing flow

Statuses

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.

API Specification

The new platform will expose a suite of RESTful API endpoints for policy management

End-point	Purpose
UPSERT Policies (Direct)	Submit one or more policies for upsert. Note: there will be a request size limit. Once policies have been accepted they will be queued for processing through the system. Each policy (as defined by its key fields) will be created or updated, depending on whether previous version(s) of the policy exist.
DELETE Policies (Direct)	Submit one or more policies for deletion using their key fields. Note: there will be a request size limit. Once keys have been accepted they will be queued for processing through the system. Where they exist, the policies represented by these keys will be deleted.
UPLOAD File	Initiate a file-based bulk request (UPSERT or DELETE). Returns a secure upload URL. The files can be either JSON or TEXT (pipe-delimited). Once the file has been uploaded it will be queued for processing through the system (in the same way as for Direct changes).
STATUS	Retrieve the processing status for a request. This may be: WAITING_FOR_FILE, QUEUED, PROCESSING, COMPLETE, REJECTED or SYSTEM_ERROR. Also includes information about the number of policies that have been/are being processed and full details of validation errors (applicable for file uploads only).

API Specification – UPSERT Policies (Direct)

POST insurers/<insurerId>/policies

Create or update policies, based on their key fields.

Whole-policy uploads are required every time. Each submission must include the complete policy **including every employer**. If a policy with a matching key already exists, it is fully overwritten by the new submission and the new submission is treated as the complete replacement for that policy. Missing employers are therefore treated as intentionally removed from that policy.

Validation is **synchronous**, and the request will only be accepted if there are no validation errors in the request; if any policy in the submission fails validation, the entire submission is rejected and no policies are ingested.

Policies are processed based on the time this request was received.

Request body:

```
[
  {
    "policyNumber": "POL001",
    "supplementaryPolicyReference": "SPR-ABC",
    "coverStartDate": "2015-01-01",
    "coverEndDate": "2023-12-31",
    "policyholderName": "Example Policyholder Ltd",
    "employers": [
      {
        "name": "Example Manufacturing Ltd",
        "erns": ["123/AB12345", "456/CD6789"],
        "crn": "12345678",
        "singleLineAddress": "Example Street, Leeds, LS1 1AB",
        "adjustedCoverStartDate": "2015-06-01",
        "adjustedCoverEndDate": "2022-06-30"
      },
      {
        "name": "Example Services Ltd",
        "erns": ["789/EF11111"],
        "multilineAddress": {
          "addressLine1": "Example Park",
          "addressLine2": "Example Quarter",
          "cityTown": "Manchester",
          "county": "Greater Manchester",
          "country": "England",
          "postCode": "M1 1AA"
        }
      }
    ]
  }
]
```

Responses:

Code	Meaning
200 OK (see below)	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.
401 Unauthenticated	No valid authentication credentials provided.
403 Forbidden	Authenticated but insufficient role or permissions for this operation.
404 Not Found	Insurer not found, or insufficient access to the organisation(s).
413 Content Too Large	The request body was too big. Submit fewer policies or use the file upload route.

Success response body:

```
{
  "uploadId": "guid",
  "timestamp": "2026-03-29T14:30:00Z"
}
```

Error Handling & Validation

Policy level errors

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 coverStartDate 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.

Error Handling & Validation

Employer level errors

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.

Duplicate policies

Where multiple policies are provided in the same submission (as identified by the key fields) then the first is processed; subsequent duplicates are rejected.

API Specification – DELETE Policies (Direct)

DELETE insurers/<insurerId>/policies

Delete policies, based on their key fields.

Validation is **synchronous**, and the request will only be accepted if there are no validation errors in the request; if any entry in the submission fails validation, the entire submission is rejected and no policies are deleted.

Deletions are processed based on the time this request was received.

Request body:

```
[
  {
    "policyNumber": "POL001",
    "supplementaryPolicyReference": "SPR-ABC",
    "coverStartDate": "2015-01-01",
  },
  {
    "policyNumber": "POL002",
    "coverStartDate": "2015-01-01",
  }
]
```

Success response body:

```
{
  "uploadId": "guid",
  "timestamp": "2026-03-29T14:30:00Z"
}
```

Responses:

Code	Meaning
200 OK (see below)	All policy keys accepted and queued for deletion.
400 Bad Request	One or more validation errors. No policy keys have been accepted for deletion. Response body contains all errors as an array of ValidationError objects.
401 Unauthenticated	No valid authentication credentials provided.
403 Forbidden	Authenticated but insufficient role or permissions for this operation.
404 Not Found	Insurer not found, or insufficient access to the organisation(s).
413 Content Too Large	The request body was too big. Submit fewer policies or use the file upload route.

Error Handling & Validation

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.

Where duplicate keys are provided in the same submission then the first is processed; subsequent duplicates are rejected.

API Specification – Validation Errors (Direct)

Bad Request responses will contain a detailed breakdown of validation issues to enable timely remediation

Where there are issues with one or more entries in the request, the response body will include an array of validation errors.

```
[
  {
    "path": "policies[0].coverEndDate",
    "description": "CoverEndDate must be after CoverStartDate."
    "policyNumber": "POL01",
    "coverStartDate": "2020-01-01"
  },
  {
    "path": "policies[0].employers[1]",
    "description": "Employer address is required."
    "policyNumber": "POL01",
    "coverStartDate": "2020-01-01",
    "employerName" : "Example employer"
  },
]
```

Policy key fields (where these could be read) are supplied along with the path of the entry with the issue to enable pin-pointing the errored record.

API Specification - UPLOAD File

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

Initiate a file-based bulk request (UPSERT or DELETE).

Entries in uploaded files are processed based on the time this request was received.

Policies or policy keys will be validated **asynchronously** after upload, and entries accepted/rejected individually. There is no minimum pass rate.

Request body:

```
{
  "fileType": "JSON | TEXT",
  "fileName": "string",
  "operationType": "UPSERT | DELETE"
}
```

Responses:

Code	Meaning
200 OK (see below)	All policy keys accepted and queued for deletion.
400 Bad Request	One or more errors with the request, for example an invalid file type.
401 Unauthenticated	No valid authentication credentials provided.
403 Forbidden	Authenticated but insufficient role or permissions for this operation.
404 Not Found	Insurer not found, or insufficient access to the organisation(s).

Success response body:

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

API Specification – STATUS

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

Retrieve the processing status for a request (both Direct and File).

For file-based submissions this will include any validation errors.

Request body: none.

Responses:

Code	Meaning
200 OK (see right)	Status returned.
400 Bad Request	One or more errors with the request, for example an invalid file type.
401 Unauthenticated	No valid authentication credentials provided.
403 Forbidden	Authenticated but insufficient role or permissions for this operation.
404 Not Found	Upload ID, supplier or insurer not found, or insufficient access to the organisation(s).

Success response body:

```
{
  "uploadId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "timestamp": "2026-04-20T16:00:00Z",
  "operationType": "UPSERT | DELETE",
  "status": "WAITING_FOR_FILE | QUEUED | PROCESSING | COMPLETE | REJECTED | SYSTEM_ERROR",
  "fileRejectionReason": "",
  "totalPolicies": 10,
  "validPolicies": 8,
  "rejectedPolicies": 2,
  "validationErrors": [
    {
      "path": "policies[0].employers[1].postCode",
      "description": "Postcode must match UK postcode format"
    }
  ],
  "processingStartedAt": "2026-04-20T17:00:00Z ",
  "validationCompletedAt": "2026-04-20T17:01:00Z ",
  "processingCompletedAt": "2026-04-20T17:02:00Z "
}
```

API Specification – Validation Errors (File)

Validation will happen asynchronously, and errors will be available through the STATUS API or portal

Validation errors and the error format is shared with Direct uploads, but the method of retrieval differs due to the asynchronous nature of the validation – they are available using the STATUS endpoint and policy management area of the portal.

Behaviour is per-entry for file uploads; and entries accepted/rejected individually with no minimum pass rate.

For Text files, the “lineNumber” field is additionally available.

JSON file upload – status response

```
{
  …
  "totalPolicies": 10,
  "validPolicies": 8,
  "rejectedPolicies": 2,
  "validationErrors": [
    {
      "path": "policies[0].coverEndDate",
      "description": "CoverEndDate must be after CoverStartDate."
      "policyNumber": "POL01",
      "coverStartDate": "2020-01-01"
    },
    {
      "path": "policies[0].employers[1]",
      "description": "Employer address is required."
      "policyNumber": "POL01",
      "coverStartDate": "2020-01-01",
      "employerName" : "Example employer"
    },
  ]
}
```

Text file upload – status response

```
{
  …
  "totalPolicies": 10,
  "validPolicies": 8,
  "rejectedPolicies": 2,
  "validationErrors": [
    {
      "path": "policies[0].coverEndDate",
      "description": "CoverEndDate must be after CoverStartDate."
      "policyNumber": "POL01",
      "coverStartDate": "2020-01-01",
      "lineNumber": 5
    },
  ]
}
```

File Uploads - File Format

Policy data files may be submitted using either of two standard formats: JSON or pipe-delimited text, replacing the existing fixed-width text file format

JSON files exactly match the format accepted by the direct API endpoints (UPSERT and DELETE).

Pipe-delimited text files have the following format:

- UTF-8 encoded text file
- 18 columns with these headings, in order (case-sensitive)
  - Columns 1-6 are repeated for each employer in a policy.
  - Columns 7-17 are repeated for each ERN for an employer.

#	Column
1	PolicyNumber
2	SupplementaryPolicyReference
3	CoverStartDate
4	CoverEndDate
5	PolicyholderName
6	FirstClaimSettlementDate
7	EmployerName
8	SingleLineAddress
9	AddressLine1
10	AddressLine2
11	CityTown
12	County
13	Country
14	PostCode
15	AdjustedCoverStartDate
16	AdjustedCoverEndDate
17	CRN
18	ERN

Pipe-delimited text file example:

```
PolicyNumber|SupplementaryPolicyReference|CoverStartDate|CoverEndDate|PolicyholderName|FirstClaimSettlementDate|EmployerName|SingleLineAddress|AddressLine1|AddressLine2|CityTown|County|Co
untry|PostCode|AdjustedCoverStartDate|AdjustedCoverEndDate|CRN|ERN
POL001|SPR-ABC|2015-01-01|2023-12-31|Example Policyholder Ltd||Example Manufacturing Ltd|Example Street, Leeds, LS1 1AB|||||||2015-06-01|2022-06-30|12345678|123/AB12345
POL001|SPR-ABC|2015-01-01|2023-12-31|Example Policyholder Ltd||Example Manufacturing Ltd|Example Street, Leeds, LS1 1AB|||||||2015-06-01|2022-06-30|12345678|456/CD6789
POL001|SPR-ABC|2015-01-01|2023-12-31|Example Policyholder Ltd||Example Services Ltd||Example Park|Example Quarter|Manchester|Greater Manchester|England|M1 1AA||||789/EF11111
```

File Uploads – Escaping Characters in Text Files

Excel conventions are followed for escaping characters to allow compatibility with Excel

The pipe character (|) is used as the field separator in the text file format. If a **field 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.

For example:
Employer | Trading Ltd -> "Employer | Trading Ltd"

If a **field contains double quotes**, the field must be enclosed in double quotes and each double quote must be doubled.

For example:
Employer "North" Ltd -> "Employer ""North"" Ltd"

API Specification – Test Endpoints

There will also be test endpoints available for onboarding, integration and development purposes.

These endpoints are **always available**, for e.g.

- Testing policy formatting and structure
- Testing integration points
- Undertaking development work.

The endpoints **mirror the production endpoints**, but with the following differences:

1. Authentication and Authorisation
Different credentials will be used for test endpoints to ensure isolation from production data.

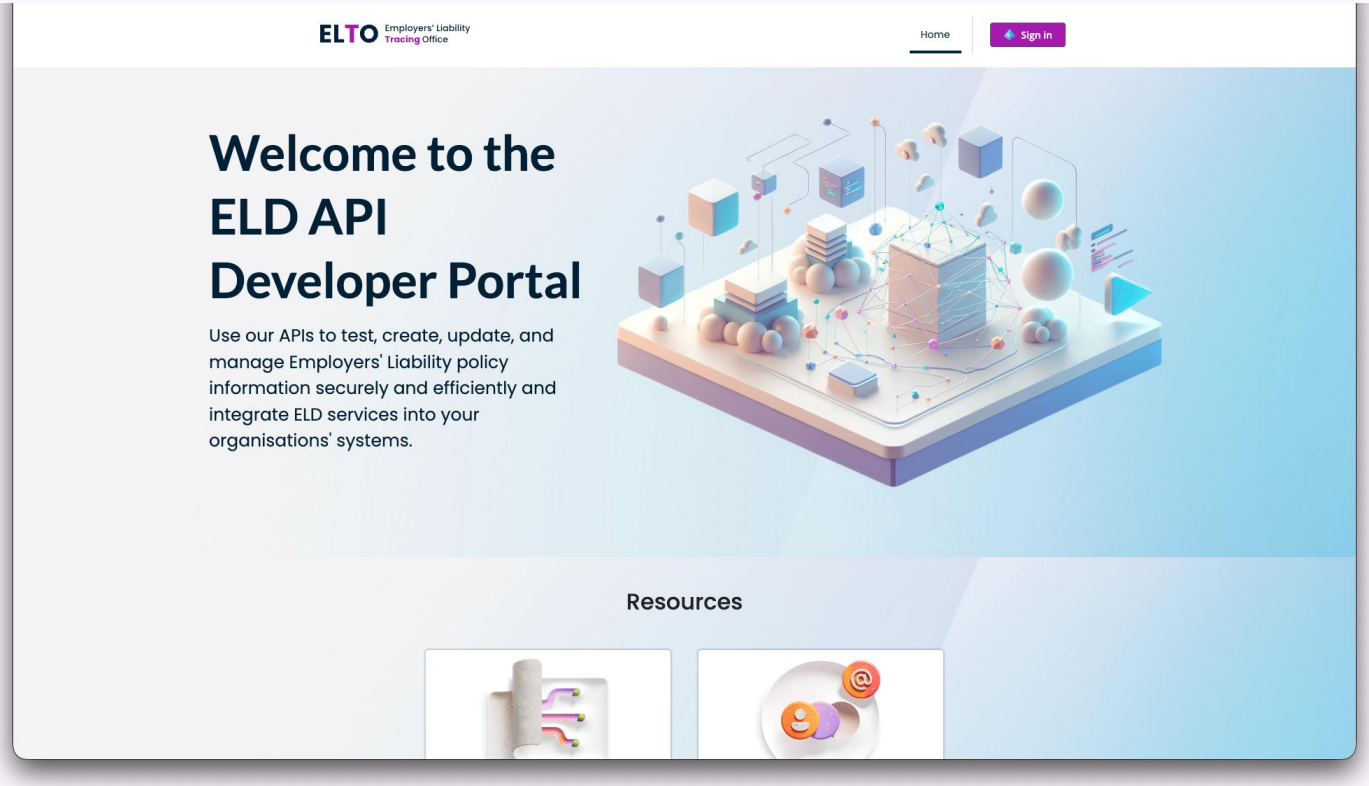
2. Data is validated only
Data is not ingested into the system.

- For Direct endpoints the request will be validated synchronously and no data persisted.
- For file uploads the data will be persisted temporarily for the purposes of validation, and the validation results will be available via the *test* STATUS endpoint.

Testing Approach

Developer Portal

- End-point specification will be available through the developer portal.
- Insurers will be able to log-in to this portal through MFA of their own tenant.

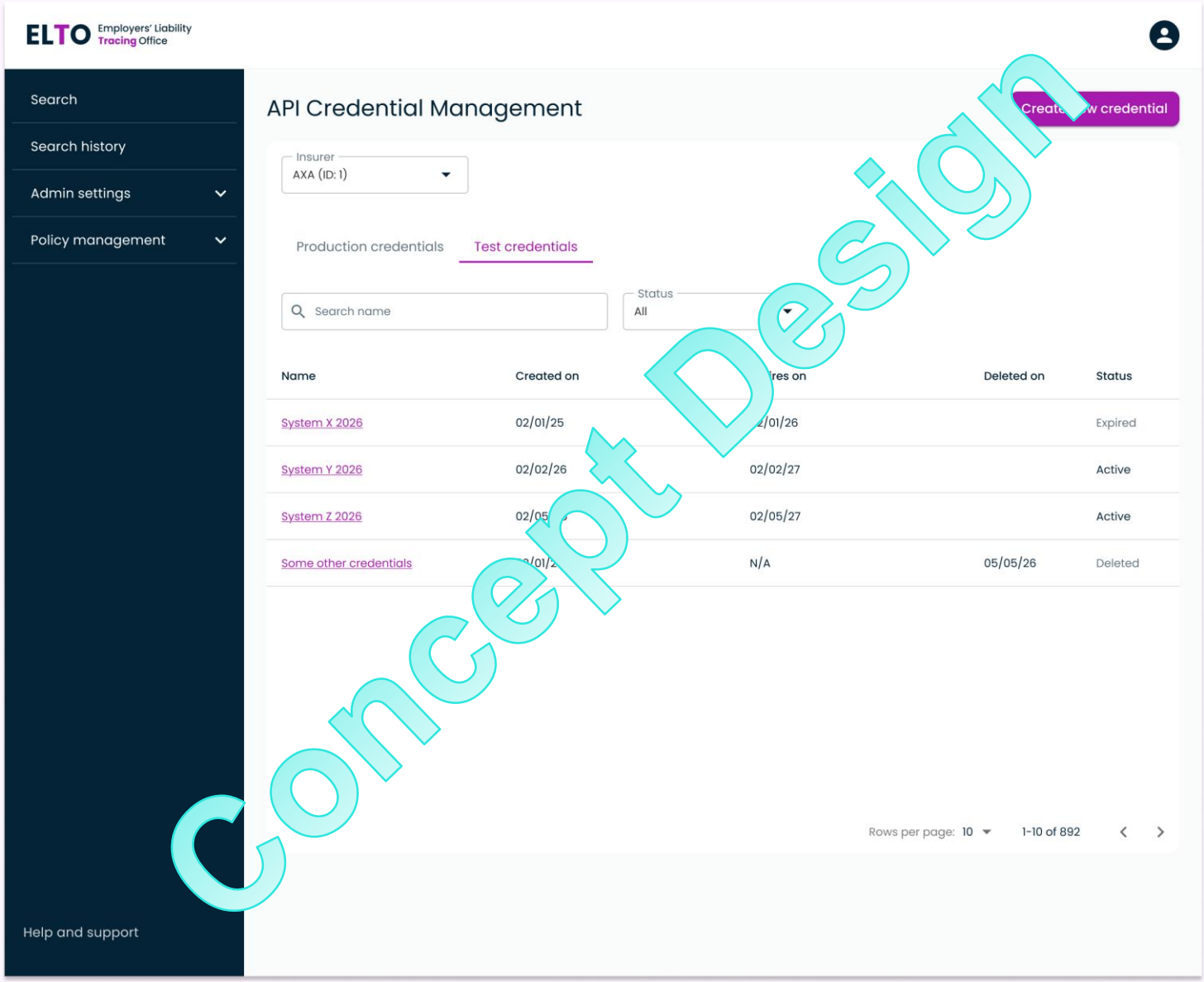


Demo of the developer portal

Testing Approach

Credentials Management

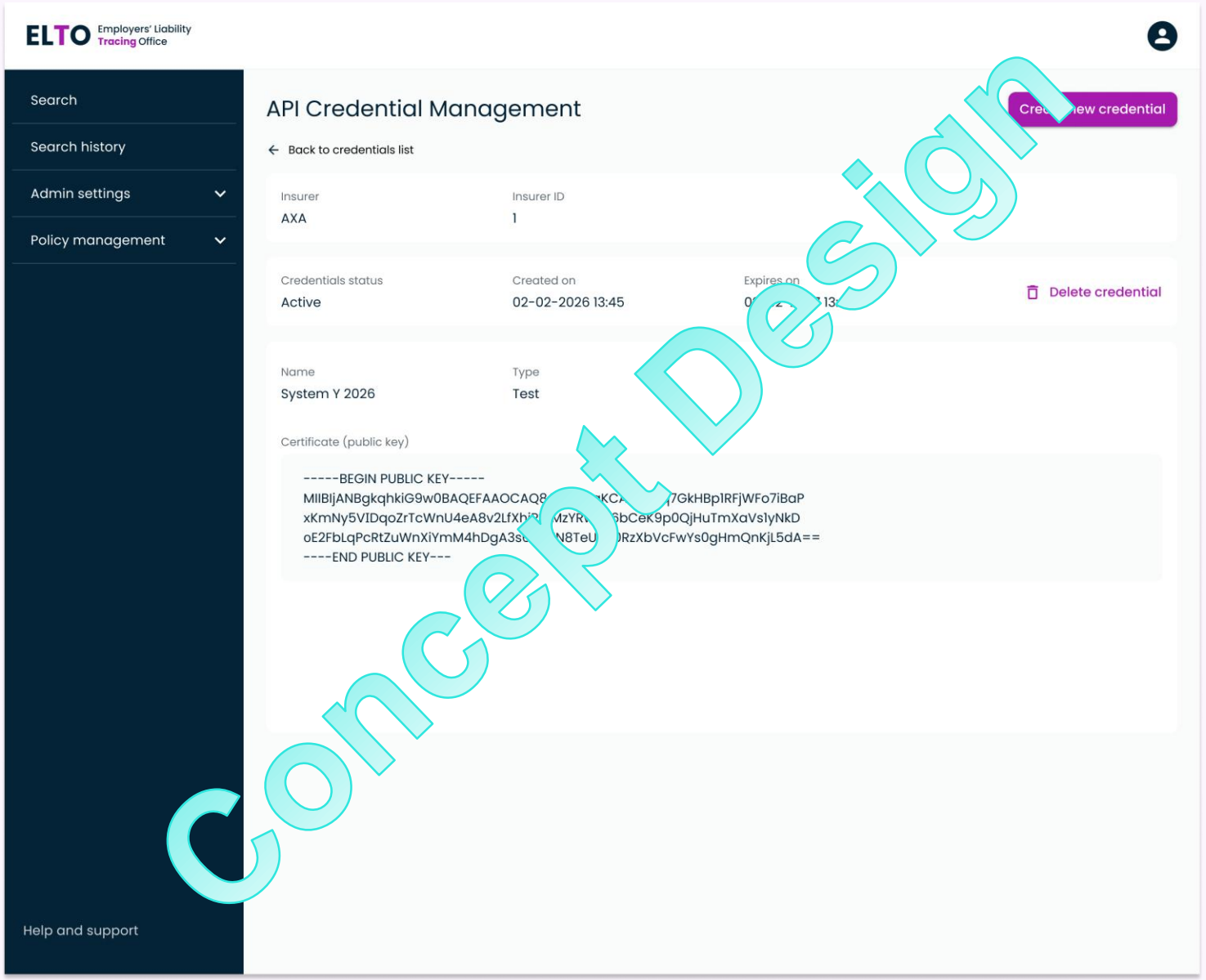
List view page



Testing Approach

Credentials Management

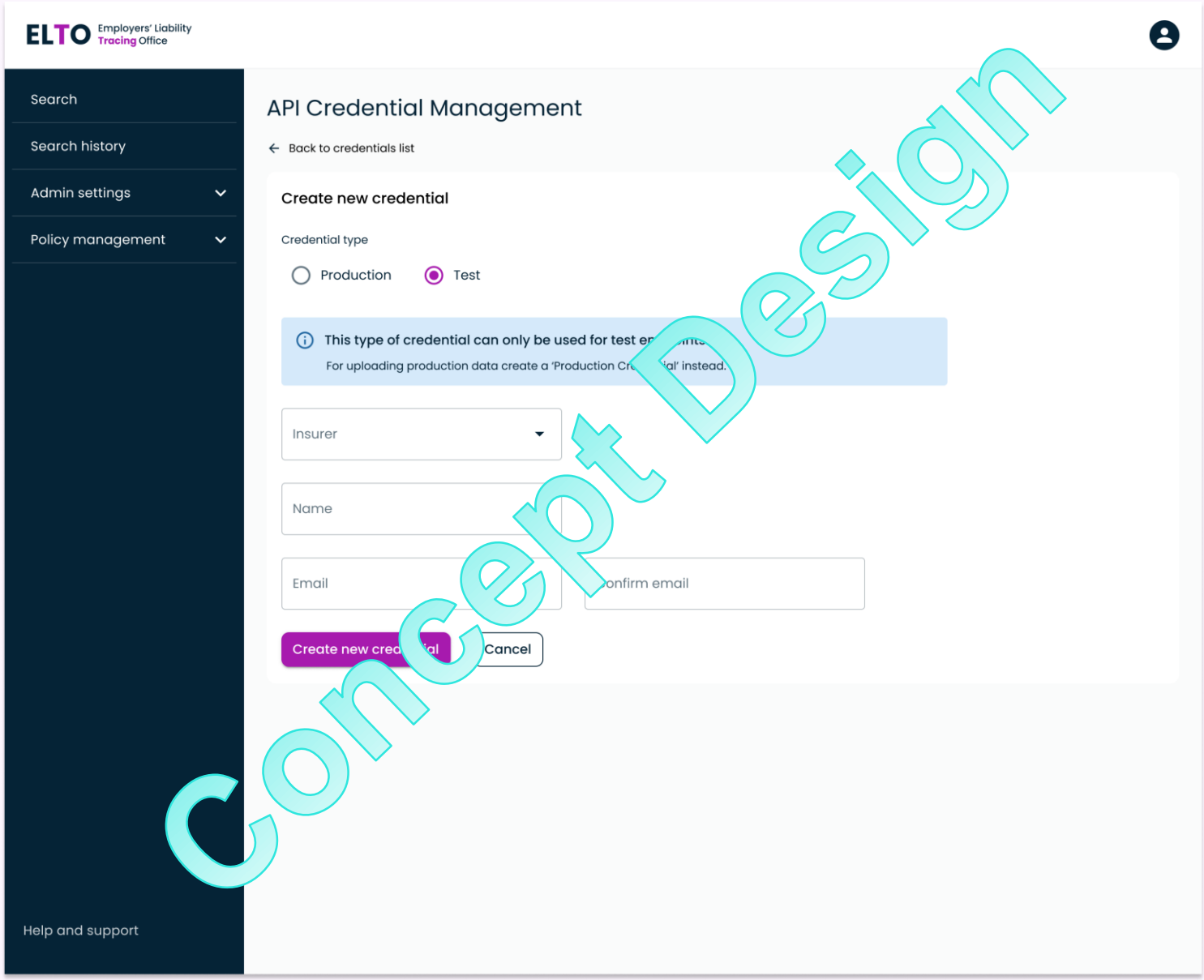
Details view page



Testing Approach

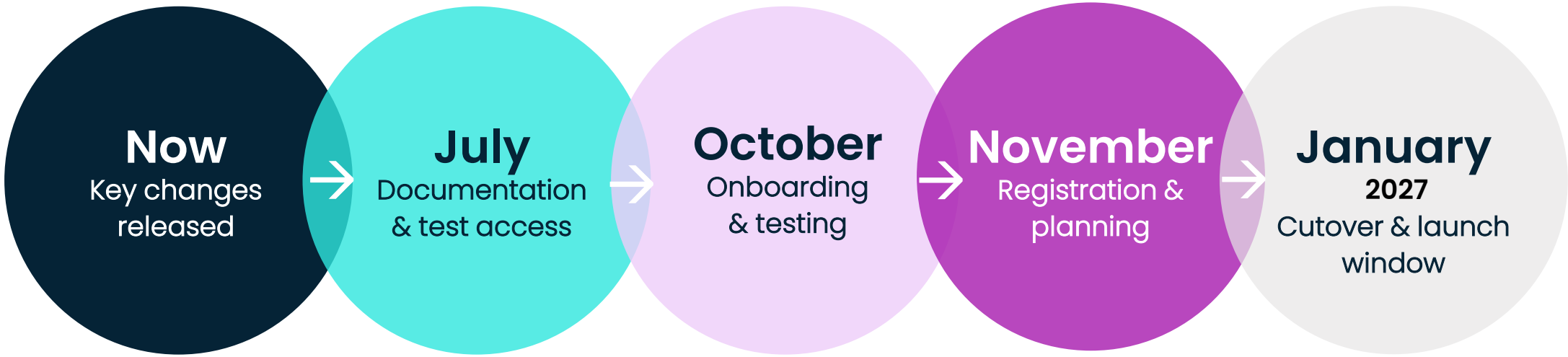
Credentials Management

Create Credentials page



Transition approach and timeline

A structured and supported transition



Now...

Key changes released

Outline API documentation shared

From July...

Full documentation and test endpoints available

Regular drop-in sessions available

From October 2026...

Onboarding and testing environment available

Regular drop-in sessions available

From November 2026...

Registration to the new ELD and cutover planning

Regular drop-in sessions available

From January 2027...

Cutover and launch window between January and March

Legacy platform & SFTP will be decommissioned from 1 Apr

Subject to final confirmation

Q&A

Let's discuss...

ELTO Employers' Liability Tracing Office

Contact us

ELD NextGen programme queries:
ELDNextGen@elto.org.uk

General ELTO enquiries:
customerenquiries@mib.org.uk