DSO Viewer — API Reference

How the LDE viewer talks to the Digitaal Stelsel Omgevingswet.

5
UPSTREAM DSO APIS

12
LDE PROXY ENDPOINTS

2
ENVIRONMENTS, ONE HEADER

The call path

The frontend never calls DSO directly. Every request is proxied.

`DsoExplorer.tsx`

The viewer

Three tabs, rules panel, CPSV handoff

→

`dsoService.ts`

Typed client

Unwraps HAL: `_embedded`, `_links.next`

→

`LDE /v1/dso/*`

The proxy

Mounted in `registry.ts`; key injected here

→

`dso.service.ts`

Outbound calls

STTR, DMN and form parsing live here

→

`DSO`

The stelsel

Five separate public services

What the proxy layer buys us

01 · CREDENTIAL

The key stays server-side

The backend attaches `x-api-key` to every outbound request. Separate keys for pre and prod; neither is bundled into the browser.

02 · ENVIRONMENT

One header switches the stelsel

`X-Dso-Env: prod` or `?env=prod` — header wins, anything else falls back to pre.

03 · PAYLOAD

HAL travels verbatim

Responses pass through untouched inside `{ success, data }`. The client does the unwrapping, so DSO's own shape stays authoritative.

Environment is a user setting: `lde_dso_env` in localStorage, with a pre-production / production badge in the viewer header.

Five upstream DSO APIs

Base URLs are configured per environment and overridable by environment variable.

01

Stelselcatalogus

`catalogus/api/opvragen/v3`

Concept and term lookup

02

RTR Gegevens

`toepasbare-regels/rtrgegevens/v2`

Activiteiten and their rule objects

03

Zoekinterface

`toepasbare-regels/zoekinterface/v2`

Werkzaamheden search and autocomplete

04

Opvragen Werkzaamheden

`opvragen werkzaamheden/v1`

Versioned werkzaamheid detail

05

Uitvoeren Gegevens

`toepasbareregels uitvoerengegevens/v1`

Rule metadata and STTR download

Three tabs, five APIs

Concepts
BegrippenTab

API 1 — `GET /begrippen?zoekTerm&geldigOp&page&pageSize` — paged 20

Werkzaamheden
WerkzaamhedenTab

API 3 — `POST /werkzaamheden/_suggereer · /_zoek` — suggest + search

API 4 — `GET /werkzaamheden/{urn}?pageSize=100` — version detail

Activities
ActiviteitenTab

API 2 — `GET /activiteiten · /activiteiten/{urn} · POST /_zoek` — list + detail

API 5 — `GET /toepasbareRegels` — rules panel

Activities load in two modes

MODE A · DEFAULT

By date

Everything valid on a given date, paged 20 at a time.

```
GET /v1/dso/activiteiten
  ?datum&page&pageSize
```

MODE B · LOCATION PRESETS

By authority

One call at `pageSize=200` loads the authority's whole set, so the name filter runs client-side.

```
POST /v1/dso/activiteiten/oin
  { oin, datum }
```

Four presets are hard-coded: Lelystad, Flevoland, Ede, Gelderland. The OIN-to-name map exists because the RTR returns only the code, e.g. `GM0995`.

UI dates are ISO and converted to the DSO's `dd-MM-yyyy`; omit the date and the backend defaults to today.

Opening an activity costs 1 + N requests

The RTR returns child activities as bare hrefs with no `omschrijving`, so the panel asks about each one individually — only to read its name.

ONE CLICK, FIRED TOGETHER

```
GET /v1/dso/activiteiten/{parent}
  ├─ GET /v1/dso/activiteiten/{child-1}
  ├─ GET /v1/dso/activiteiten/{child-2}
  └─ …  N in parallel
```

24 upstream RTR calls to render one panel for *Bedrijfsactiviteiten* — 23 children plus the parent.

01 `Promise.allSettled` — one failing child never breaks the panel

02 Children that fail, or return no name, fall back to the raw URN — still clickable, just unlabelled

03 The `Child activities (N)` count reads the href list, so it stays right when lookups fail

04 Each child inherits the parent's `datum` and `env`

05 No cache and no concurrency cap: names live in component state, cleared on every urn, datum or env change

From rule object to rule file

Selected activity
`+ onderliggendeActiviteiten`

→

regelBeheerObjecten
`Conclusie · Indieningsvereisten · Maatregelen`

→

functioneleStructuurRef
`API 5 · GET /toepasbareRegels`

→

identifier
`GET /{id}/sttrBestand`

One upstream endpoint, three actions — they differ only in what LDE does with the XML.

Download STTR

`/toepasbare-regels/:id/sttr`

No processing — the XML is passed through as an attachment.

Extract DMN

`/toepasbare-regels/:id/dmn`

Pulls the embedded `<definitions>` and normalises it for Operaton.

Form scaffold

`/toepasbare-regels/:id/form-scaffold`

Parses `uitv:uitvoeringsregels` into a form-js schema for the Form Editor.

Five fixes make STTR output evaluatable

`normalizeDmnForOperaton` rewrites the extracted decision table on the way out.

01 DMN 1.2 namespaces rewritten to 1.3

02 Missing `id` injected on `<input>` and `<inputExpression>`

03 FEEL-safe `<variable>` names, with input expressions rewritten to match

04 `typeRef` added to untyped outputs — the BIZ-004 case

05 `camunda:historyTimeToLive="180"` set per decision

Form scaffolding maps question types the same way: boolean to checkbox, list to select, number to number, textarea hint to textarea, everything else to a textfield. `uitv:geoVerwijzing` is skipped as unrepresentable.

05

Publishing a DMN is a handoff, not a store

LDE keeps no local DMN store, so the extracted decision is deep-linked to the CPSV Editor.

THE DEEP LINK

```
<VITE_CPSV_EDITOR_URL>/?dsoImport=dmn&dmnId=<id>&env=<pre|prod>
```

Only identifiers travel

The URL carries the DMN id, the environment and activity metadata — never the XML.

A cross-application contract

The CPSV Editor then fetches the XML from `/v1/dso/toepasbare-regels/{id}/dmn` on this same backend.

Known loose ends

BUILT, NOT WIRED

Geo search has no UI

A WGS84 point search is implemented and tested end to end in both backend and client — no screen calls it yet.

BUILT, NOT EXPOSED

Validity date has no control

Concept search accepts `geldigOp` in the backend, but the Concepts tab has no field for it — so historical lookups are out of reach.

PERFORMANCE

The fan-out has no cache

Every activity opened re-issues `1 + N` RTR calls, with no batching or concurrency limit. First thing to memoise if child counts grow or rate limiting appears.

What we do next

01	**Map and point selection** Put a UI on the geo endpoint that already works — activities at a clicked location.	`frontend only`
02	**Memoise the child fan-out** Cache resolved child names and cap concurrency.	`before rate limiting bites`
03	**Authorities beyond the four presets** Replace the hard-coded OIN list with a lookup.	`needs a name source`
04	**Validity date in the UI** Expose `geldigOp` on concept search to make historical lookups possible.	`small`
05	**Own timeout and defaults for production** Give prod its own timeout and align the documented pageSize with the code.	`housekeeping`