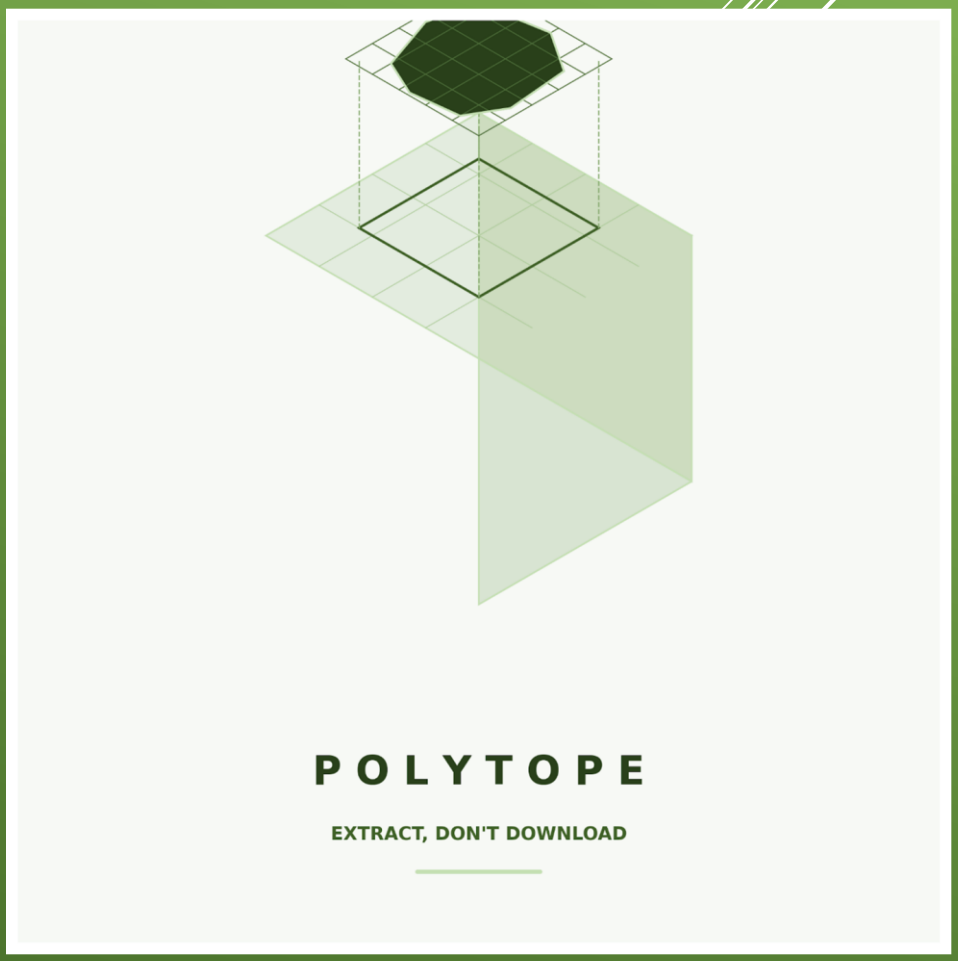

POLYTOPE FEATURE EXTRACTION ON THE CDS: API, DESIGN, AND IMPLEMENTATION

Part of ClimEmpower educational library

Project information	
Project name: **Grant Agreement No.** **Start date:** **Duration:**	Project ClimEmpower: User Driven Climate Applications Empowering Regional Resilience 101112728 01/09/2023 36 months
Coordinator:	Denis Havlik, Scientist <denis.havlik@ait,.ac.at> AIT Austrian Institute of Technology Giefinggasse 4, 1210 Vienna, Austria
	Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or CINEA. Neither the European Union nor the granting authority can be held responsible for them.
Deliverable details	
Report name	**Polytope Feature Extraction on the CDS: API, Design, and Implementation**
Target audience	For technical users who have a background in programming and data science
Description:	Explains how the ClimEmpower Data Store Service (Polytope DS2) provides tailored access to Copernicus CDS data through Polytope Feature Extraction, and shows how to retrieve, decode, and visualise a regional subset in Python.
Estimated reading time	~30 minutes
Learning level	Learning level: ☐ Introductory – assumes no previous knowledge X Intermediate – assumes basic understanding ☐ Advanced – assumes professional knowledge
Prerequisites	Familiarity with Python and basic software engineering techniques; familiarity with REST APIs; awareness of gridded climate datasets (e.g. CDS/ERA5). No GIS or meteorological background required but would be beneficial.
Keywords	ClimEmpower, educational report, CDS, Polytope, Feature Extraction, Python, REST, Climate Risk, Climate Hazard
Downloadable from	• ClimEmpower web site: https://climempower.eu/ • Zenodo: 10.5281/zenodo.21625038
Version: **Publication date:**	V1 02/09/2026
Dissemination level:	PU = Public
Editor:	Havlik, Denis - AIT
Author(s):	Warde, A., Hawkes, J., (ECMWF), UK, Germany

Recommended citation

Warde, A., Hawkes, J. (2026). Polytope Feature Extraction on the CDS: API, Design, and Implementation, in ClimEmpower educational library, ed. Havlik, D., URI: https://climempower.eu/edu/, DOI: 10.5281/zenodo.21625038

Revision history

Date	Version	Contributor/Reviewer	Description
02/09/2026	First public release	Adam Warde, main author Marianne Bügelmayer-Blaschek and Denis Havlik – review and improvements	First version of the report uploaded to Zenodo, doi reserved

Legal Disclaimer

Table of Contents

List of Figures

List of Tables

List of Acronyms

CIC	Climate interaction context
ECMWF	European Centre for Medium Range Weather Forecasts
CDS	Copernicus Data Store
C3S	Copernicus Climate Change Service
AIT	Austrian Institute of Technology
DS2	Data Store Service
REST	Representational State Transfer
API	Application Programming Interface
HTTP	Hyper Text Transfer Protocol
RCP	Representative Concentration Pathway
ERA5	ECMWF Reanalysis v5

Glossary

Climate impacts	The consequences of realized risks on natural and human systems, where risks result from the interactions of climate-related hazards (including extreme weather and climate events), exposure, and vulnerability. Impacts generally refer to effects on lives; livelihoods; health and well-being; ecosystems and species; economic, social and cultural assets; services (including ecosystem services); and infrastructure (based on IPCC, 2018)
Representative Concentration Pathway	Representative Concentration Pathways (RCP) are climate change scenarios to project future greenhouse gas concentrations. These pathways describe future greenhouse gas concentrations (not emissions) and have been formally adopted by the Intergovernmental Panel on Climate Change (IPCC) .

About this document

This document is part of ClimEmpower educational library and associated with project deliverable D4.2 “Educational materials for increased regional Climate Change resilience v2”.

*ClimEmpower’s key ambition is to **prove beyond doubt that CC-resilience should, and can, be an integral part of regional development** everywhere in EU and beyond it. That is, we anticipate that the regional stakeholders will recognise that CC-resilient development pathways offer multiple benefits to them, including but not limited to **higher quality of life and reviving economy**, and that these can be understood using available **data, tools, and services**. Second key ambition of the project is to **help the regions address the CC resilience** in key community systems addressed in five ClimEmpower trials.*

*Underlying philosophy of the project is to “**help the regions to help themselves**”. This will be achieved through various mechanisms, including co-creation and mediation of the regional “Communities of Practice”, provision of the Climate Change -resilience training materials, as well as in provision and training in use of the user-centric data and services – including those that have already been made available through previous research projects and EU initiatives.*

This material is part of the ClimEmpower educational library. Available at https://zenodo.org/uploads/21625038

This material helps **Technical Users** understand **Data Access** in ClimEmpower and enables them to **retrieve data and build applications on top of the services and data provided by ClimEmpower**

After reading this material, readers will be able to:

- **Understand** the key concepts, challenges, and principles related to Data Access in ClimEmpower via Feature Extraction.
- **Explain** how data access via Polytope DS2 works and why it is relevant for for developers, scientists and regions.
- **Apply** Polytope Feature Extraction in their own context to build applications and recommendations for their regions.

1 Introduction

1.1 Why should I read this document?

This document explores how to access data from various CDS datasets using the Polytope Feature Extraction Library. This enables users, faster, simpler, access to CDS data and does not require users to download bulky files which contain information that is not pertinent to their region or area of interest.

After reading this document users will be know

- What datasets are available via the API.
- How to use the API.
- How the API integrates with other outputs of ClimEmpower.
- How to use the data provided for plotting.

1.2 The Challenge

Climate data on the Copernicus Climate Data Store (CDS) is large by nature. A single dataset like a reanalysis such as ERA5, or a set of agroclimatic indicators projected to mid-century can span all of Europe or the globe, at full resolution, across every dimension it is indexed by. Yet a regional user rarely needs all of that. They need one or two variables, over one province or region, for one time window and scenario.

The traditional way to bridge that gap can be difficult for such users. One must download the whole field, or a large rectangular slab of it, and then subset locally to the region and variables you want. That means transferring, storing, and post-processing far more data than the question requires, and doing so in source formats such as GRIB and NetCDF that assume meteorological tooling and expertise.

For regions, the expertise or knowledge required may not be available, and the infrastructure in terms of bandwidth, storage, and processing time may be limited. ClimEmpower looks to help solve this friction by providing easier and more smooth means of access to the data for regions and users at large.

1.3 What will I learn?

This report explains the concepts and the concrete workflow behind the ClimEmpower Data Store Service powered by Polytope Feature Extraction. By the end you will understand what a climate datacube is and what *non-orthogonal feature extraction* means in that context; how to query the DS2 REST service and the Python libraries that surround it; and how to retrieve a bounding-box or polygon subset of a CDS dataset, decode it from CoverageJSON into an xarray dataset, and produce a map from it. The worked example in section 2.3 walks through this end to end with real, runnable code that is openly available at https://github.com/ecmwf/climempower-polytope-examples.

2 Concepts, methods, and approaches

Here we will go through the key concepts, methods, and approaches needed to both use and understand the Polytope Feature Extraction library and the REST API that is built upon it.

These are as follows:

- Feature extraction from climate and meteorological datacubes
- The ClimEmpower Data Store Service (Polytope DS2) and output formats
- Worked example: Growing Season Length over Andalusia

2.1 Feature Extraction from Climate and Meteorological Datacubes

Climate data on the Copernicus Climate Data Store (CDS) is organised as datacubes: multidimensional arrays indexed by dimensions such as latitude, longitude, time, ensemble member, and experiment. A single dataset such as a set of agroclimatic indicators projected to 2040 is a large object spanning all of Europe (or the globe) at full resolution across every one of those dimensions.

The traditional way to work with such data is to download the whole field, or a large orthogonal slab of it, and then subset locally to the region and variables you wish to retrieve. For a regional user who needs a single indicator over one province or region, that means transferring, storing, and post-processing far more data than the question requires, in source formats (GRIB, NetCDF) that assume meteorological tooling and expertise.

Polytope feature extraction inverts this. Rather than moving the datacube to the user, it extracts the requested feature, the part of the cube the user asks for directly at the data source and returns only that. The name comes from the extraction geometry: the region of interest is described as an arbitrary n-dimensional polygon, a polytope, and the service returns the datacube values that fall inside it. Because the shape is arbitrary and need not align with the grid axes, this is a form of non-orthogonal access: you are not restricted to rectangular slabs cut along the cube's dimensions, but can request the points inside an irregular boundary or along a spatio-temporal path.

In practice, ClimEmpower users interact with this capability through two supported shapes. A **bounding box** is defined by two corner points, **[[lat_min, lon_min], [lat_max, lon_max]]**, and returns everything inside that rectangle. A **polygon** is defined by a closed ring of `[lat, lon]` pairs — the first and last point must be identical

— and returns the points inside an arbitrary boundary, which is what you want when a region's outline is not rectangular.

A higher-level library, **polytope-mars** (Table 2) sits above the core extraction engine and exposes these and other features (time series, vertical profiles, region cut-outs) and accepts region definitions from shapefiles or GeoJSON, so an existing administrative boundary can drive the request directly. One consequence of this approach is worth internalising before writing any code: an extraction returns a point cloud, not a regular grid. What comes back is the set of datacube values at the scattered latitude/longitude locations that lie inside your shape, each tagged with its coordinates, rather than a gridded field with regular spacing. This is why the visualisation step in the worked example plots the data as a point cloud rather than as a filled contour or image. It is a natural result of extracting an arbitrary sub-region from a grid: the points that survive the cut are not, in general, a tidy rectangular lattice.

Downstream, you can still regrid or interpolate if a gridded product is needed, but the primary response is point-based. Placing this in the wider system, Polytope feature extraction is the data-access layer of the ClimEmpower architecture (As seen in Figure 1: ClimEmpower Architecture - component view) it is what the processing, indicator, and frontend components ultimately call when they need CDS data tailored to a region. Everything above it such as indicator calculations, dashboards, decision-support applications, depend on the ability to pull the right subset cheaply and in a consumable form, which is the problem this layer exists to solve.

In the next section we will go over what data we are able to access, and how we can access it.

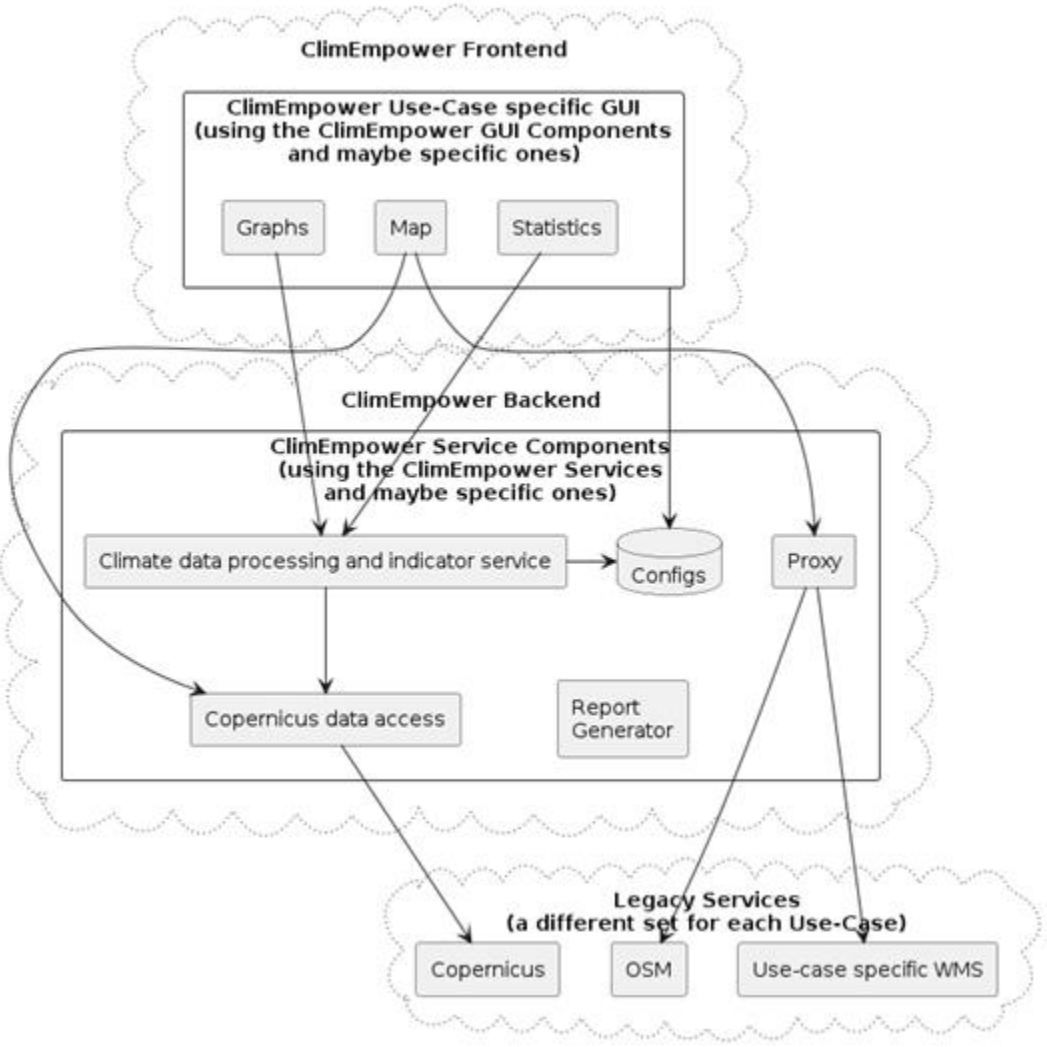

Figure 1: ClimEmpower Architecture - component view (from ClimEmpower D3.2)

2.2 The ClimEmpower Data Store Service (Polytope DS2) and Output Formats

Feature extraction as a capability is only useful to ClimEmpower's technical users if it is reachable without assembling the underlying stack by hand. The ClimEmpower Data Store Service also called the Polytope DS2 (Data Store Service) packages the extraction engine, the CDS dataset backends, and the format conversion into a single deployable REST service, hosted on ECMWF infrastructure and reachable at polytope-dss.ecmwf.int.

A developer does not install or orchestrate the components; they send an HTTP request and receive extracted data. The service is deployed as a container. It is built as a single Docker image (polytope-dss) from a Dockerfile, with the build and version tagging handled through a Skaffold pipeline using BuildKit, so the deployed artefact is versioned against the package release.

The repository is open source under Apache 2.0 (© 2025 ECMWF) and carries the usual project scaffolding such as continuous integration through GitHub Actions, test

coverage reporting, and hosted documentation. We also provide examples for users of extraction and plotting of each dataset. This corroborates and makes concrete the Docker/Kubernetes deployment approach described in D3.2, and it means the service can in principle be redeployed elsewhere rather than being tied to a single host. It should be read in the context of its maturity, however while the service is fully featured is not a 24/7 operational service and it may experience outages and degraded performance. Future improvements to the service and underlying infrastructure are intended to improve the reliability and provide an operational service.

All of the following endpoints and queries are described in an OpenAPI document found here https://polytope-dss.ecmwf.int/docs.

The interface follows a consistent pattern. Data is retrieved by POSTing to:

POST https://polytope-dss.ecmwf.int/api/v1/dataset/<dataset-id>/extract/

With a JSON body that always has two top-level keys. The `request` key carries the dataset-specific fields — variable, period, temporal aggregation, experiment, model, and so on — and the `feature` key carries the shape (bounding box or polygon) described in 2.1.

Three supporting endpoints make the datasets self-describing so that requests can be constructed correctly rather than by guesswork: `/constraints/` returns the official CDS constraint definition listing every valid parameter combination for a dataset; `/variable_metadata/` returns each variable's type, description, and unit; and `/units/` returns the variable-to-unit mapping. The recommended workflow is to consult these before issuing an extraction, particularly `/constraints/`, since they tell you exactly which field combinations a given dataset will accept.

The DS2 exposes a catalogue of CDS datasets, and for ClimEmpower the useful lens is which hazards each one supports. The following mapping orients a technical user toward the right dataset for a given regional concern:

Table 1: Overview of Hazards and associated Datasets

Hazard	Purpose
Heat	Heat-and-cold-spells; ECDE climate indicators (heatwave days, hot days, tropical nights
Drought and Agriculture	Agroclimatic indicators; ECDE (SPI, consecutive dry days); hydrology variables
Wildfire	Tourism & fire-danger indicators; ECDE fire weather index
Floods	EFAS forecast and seasonal; hydrology (flood recurrence); CORDEX precipitation
Cross-Cutting Projections	CORDEX single-levels; ECV CMIP5 bias-corrected; IPCC AR6 Atlas; European wind-storm

A deliberate convenience of the service is **dynamic defaults**. When you omit a field, the API infers it from the dataset's constraints given the fields you did supply: if only

one valid option remains it is filled in automatically, and where several are possible the service selects a preferred value.

For climate-projection datasets, the preferred experiment defaults to RCP 8.5 when *experiment* is omitted. This makes exploratory requests short and forgiving, but it has a direct bearing on reproducibility and on the correctness of decision-grade output, which is why it is flagged as the principal pitfall in Limitations, Pitfalls, and Quality Considerations, a request that does not state its scenario is not self-documenting, and the default may not be the scenario you intended.

Table 2: Overview of data access service stack

Service/tool name	Purpose	Further information	Version
Polytope Feature Extraction	Polytope is a library for extracting complex data from datacubes. It provides an API for non-orthogonal access to data.	https://github.com/ecmwf/polytope	1.1.1
Polytope-mars	This library provides a higher-level API to allow requests for features such as time series and vertical profiles.	https://github.com/ecmwf/polytope-mars	0.3.4
Covjsonkit	Covjsonkit is a library for encoding and decoding coverageJSON files/objects of meteorological features such as vertical profiles and time series.	https://github.com/ecmwf/covjsonkit	0.2.3
Polytope-client	Polytope-client provides a REST API and python interface for access to hypercube data, stored in various data sources.	https://github.com/ecmwf/polytope-client	0.7.7
Earthkit	A python library providing powerful tools for speeding up weather and climate science workflows by simplifying data access, processing, analysis, visualisation and much more.	https://earthkit.readthedocs.io	0.16.5 (EarthKit-Data)
Polytope DS2	Polytope DS2 (Data Store Service) combines most of the above libraries into a single deployable service.	https://polytope-dss.ecmwf.int/docs#/	Prototype Deployed

The final piece is what the service returns and how you consume it. Extraction results are delivered as **CoverageJSON,** an OGC community standard for publishing spatiotemporal data to the web, designed for both machine processing and human

readability, and converted from the underlying GRIB/NetCDF so that users need not handle those formats directly.

CoverageJSON is decoded with **covjsonkit (**https://github.com/ecmwf/covjsonkit/), an ECMWF library that encodes and decodes CoverageJSON and, most usefully for analysis, converts a response into an xarray dataset (and to NetCDF). From xarray the data is ordinary Python: you can compute derived indicators, subset further, or hand it to a plotting library.

Earthkit ties the workflow together, providing request, read, and visualisation in one library so that the path from an extraction request to a finished map is a handful of lines — as the worked example in demonstrates.

An OGC Environmental Data Retrieval (EDR) interface to the same service is under development and will offer a standardised REST alternative for clients that speak EDR.

The full set of libraries, their repositories, versions, and maturity levels is given in the data-access service stack Table 2, and the interaction between the extraction engine, covjsonkit, and Earthkit is shown in the component-interaction diagram (Figure 2: Overview of Polytope Software Component Interactions).

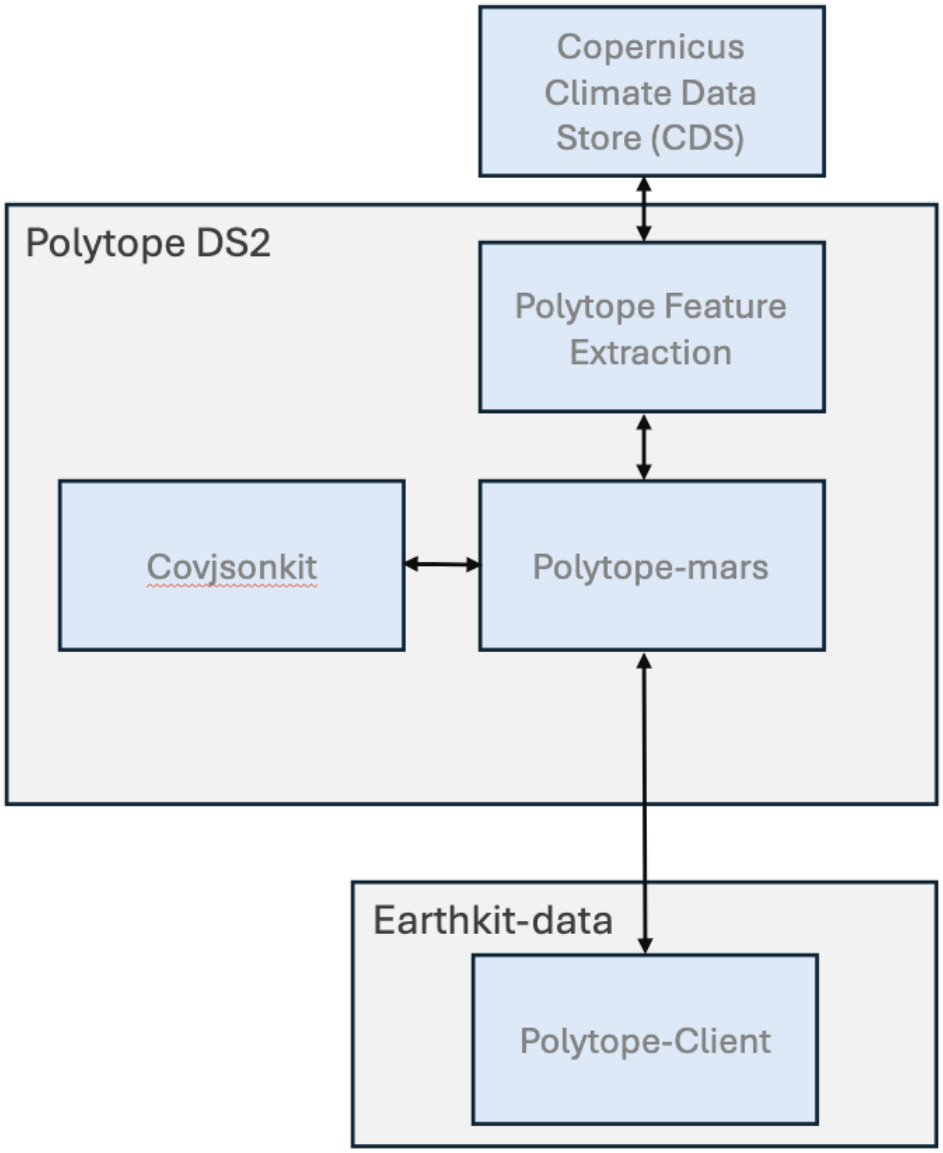

Figure 2: Overview of Polytope Software Component Interactions

2.3 Getting Started: Access, Interfaces, and Resources

There are two ways to use the service, suited to different stages of work. The Python path, shown in section 2. is ideal for exploration and prototyping: interactive, quick to iterate, and easy to fold into an analysis notebook.

The raw REST path, POSTing JSON to the extract endpoint from any language or tool is what you reach for when integrating the service into an operational component that is not written in Python.

Both hit the same endpoints and accept the same request bodies. Before building requests for a dataset you do not already know, use the self-describing endpoints.

/constraints/ tells you which parameter combinations are valid; /variable_metadata/ gives each variable's type, description, and unit; and /units/ gives the unit mapping.

Consulting /constraints/ first is the single most effective habit for avoiding failed requests. When you do request data, decide deliberately between a bounding box and a polygon. The box is simpler, the polygon matches real administrative or catchment boundaries, and expect a point cloud back rather than a gridded field.

Getting started only requires a HTTP client and two ECMWF libraries, covjsonkit to decode responses and Earthkit for plotting. Both can be installed with pip or uv: **python -m pip install covjsonkit earthkit-plots requests**

2.4 Worked Example: Growing Season Length in Andalusia

This example takes the concepts above and runs them end to end. It requests an agroclimatic indicator for a region, decodes the response, derives a simple change indicator, and maps it. The dataset is **sis-agroclimatic-indicators**; the variable is **Growing Season Length** (GSL); the region is a bounding box over the Iberian Peninsula and western Europe that includes the Andalusia pilot region.

The same code retrieves the other ClimEmpower regions (Sicily, Greece, Cyprus, Croatia) once the eastern longitude bound is widened past 10°E.

First, the setup — the HTTP client, the covjsonkit decoder, and Earthkit for plotting, plus the endpoint template and headers:

```
import requests
from covjsonkit.api import Covjsonkit
import earthkit.plots as ekp

base_url = 'https://polytope-dss.ecmwf.int/api/v1/dataset/{dataset}/extract/'
headers = {'accept': 'application/json', 'Content-Type': 'application/json'}
```

Here we import the relevant libaries and point to the endpoint set up for ClimEmpower that we can query for data.

Note: *getting started only requires a HTTP client and two ECMWF libraries, covjsonkit to decode responses and Earthkit for plotting. Both can be installed with pip or uv:* ***python -m pip install covjsonkit earthkit-plots requests***

Next, the request. The body has the two keys introduced in 2.2: a *request* body for the dataset fields and *feature* for the shape. Note the two commented-out lines — *origin* and *experiment*. Leaving them out is what triggers the **dynamic defaults** behaviour: the service fills them from the dataset constraints, and for a projection dataset the experiment defaults to RCP 8.5. This is convenient here for a first look, but Limitations, Pitfalls, and Quality Considerations explains why you should pin them for any result you intend to act on.

```
bbox_request = {

    'request': {
        'variable': ['growing_season_length'],
        'temporal_aggregation': 'annual',
        'period': ['201101_204012'],
        # 'origin': 'noresm1_m_model',   # omit -> default model applied
        # 'experiment': 'rcp4_5',        # omit -> default scenario (RCP 8.5) applied
    },
    'feature': {
        'type': 'boundingbox',
        # [lat_min, lon_min], [lat_max, lon_max] — this box includes Andalusia
        'coordinates': [[35, -10], [60, 10]],
    },
}

response = requests.post(
    base_url.format(dataset='sis-agroclimatic-indicators'),
    headers=headers, json=bbox_request,
)
print('Status:', response.status_code)
```

The response is a CoverageJSON CoverageCollection. One line hands it to covjsonkit, which decodes it and converts it to an xarray dataset, from here on the data is ordinary Python:

```
ds = Covjsonkit().decode(response.json()).to_xarray()
```

The dataset carries GSL for each time step in the requested period at every extracted point. A common and readable indicator is the change over the period: the value at the last time step minus the value at the first. We compute it, give it sensible attributes, and plot it. Because extraction returns a point cloud rather than a grid (2.1), the plot uses Earthkit's *point_cloud* renderer, with a diverging colour scale so that increases and decreases in growing-season length read clearly:

```
ds['GSL_diff'] = ds['GSL'].isel(datetimes=-1) - ds['GSL'].isel(datetimes=0)
ds['GSL_diff'].attrs = {'units': 'days', 'long_name': 'GSL change'}

chart = ekp.Map(domain="Europe")
chart.point_cloud(
    ds['GSL_diff'], x=ds['longitude'], y=ds['latitude'], s=15,
    style=ekp.Style(levels=range(-150, 150, 10), colors='RdYlGn', extend="both"),
)
chart.coastlines(); chart.borders(); chart.gridlines()
chart.title("Change in average annual Growing Season Length: 2011–2040")
chart.legend(label='days')
chart.show()
```

The map produced from the above code can be seen below in Figure 3.

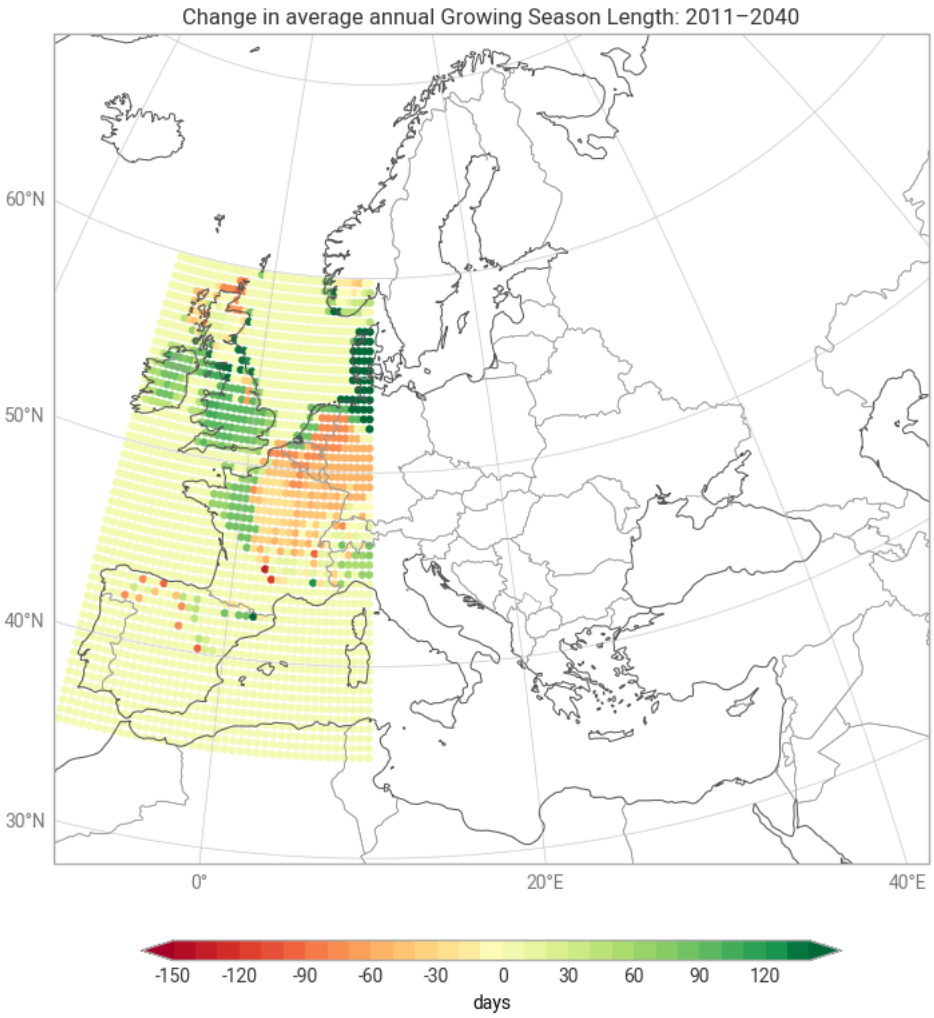

Figure 3: Plot produced from code that pulls data via Polytope and then plots it

To extract an irregular region rather than a rectangle, the only change is the *feature*: set *type* to *polygon* and supply a closed ring of **[lat, lon]** pairs whose first and last points are identical. The request and decoding are otherwise unchanged:

```
polygon_request = {
    'request': {
        'variable': ['growing_season_length'],
        'temporal_aggregation': 'annual',
        'period': ['201101_204012'],
    },
    'feature': {
        'type': 'polygon',
        'coordinates': [
            # closed [lat, lon] ring — first point must equal last point
            [40.0, -97.8], [42.8, -97.5], [44.8, -95.0], [44.9, -91.0],
            [41.8, -86.5], [39.8, -84.8], [37.0, -89.0], [38.0, -94.5],
            [40.0, -97.8],
        ],
    },
}
```

The ring above is a demonstration polygon (the US Corn Belt) showing that an arbitrary boundary works exactly like the bounding box; to regionalise it, substitute an Andalusia (or other pilot-region) boundary ring. Before issuing either request against an unfamiliar dataset, call the /constraints/ endpoint first to confirm which field combinations are valid, it is faster than discovering an invalid combination from a failed request.

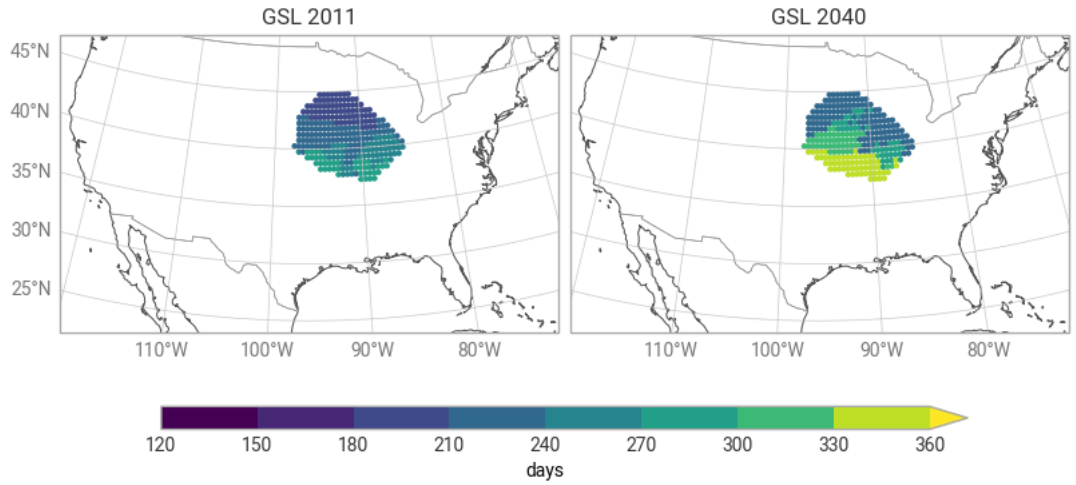

Figure 4: Plot produced from Polygon request code above

Note: Users can still request the entire data field as they would from the regular CDS API by removing the feature from the request. In that case the original data in the original data format (GRIB/NetCDF) is returned to the user as it would be from the CDS API.

3 Practical considerations and lessons learnt

Here we will discuss the requirements to run the above code, such as libraires needed, and also some ways that users can better take advantage of the service.

3.1 Getting Started: Access, Interfaces, and Resources

There are two ways to use the service, suited to different stages of work. The Python path, shown in section 2. is ideal for exploration and prototyping: interactive, quick to iterate, and easy to fold into an analysis notebook.

The raw REST path, POSTing JSON to the extract endpoint from any language or tool is what you reach for when integrating the service into an operational component that is not written in Python.

Both hit the same endpoints and accept the same request bodies. Before building requests for a dataset you do not already know, use the self-describing endpoints. /constraints/ tells you which parameter combinations are valid; /variable_metadata/ gives each variable's type, description, and unit; and /units/ gives the unit mapping.

Consulting /constraints/ first is the single most effective habit for avoiding failed requests. When you do request data, decide deliberately between a bounding box and a polygon. The box is simpler, the polygon matches real administrative or catchment boundaries, and expect a point cloud back rather than a gridded field.

3.2 Limitations, Pitfalls, and Quality Considerations

Be explicit about scenario and model. The dynamic-defaults behaviour is the feature most likely to catch you out. Omitting *experiment* on a projection dataset silently applies the preferred default, RCP 8.5, and omitting a model lets the service choose one.

This is fine for a first exploratory look, but any figure or indicator that informs a decision should state its scenario and model explicitly in the request. A request that does not name its scenario is not reproducible and may not describe the future you meant to describe.

Treat the service as a research prototype.

The DS2 is not a 24/7 operational service and may experience outages or degraded performance The service packages are open source, maintained by ECMWF, and intended to persist beyond the project, but they are currently not a production service with production guarantees. Plan integrations with that in mind: pin versions, expect interfaces to evolve, and check the project documentation for current status.

Use for gridded, geospatial climate data

Polytope feature extraction applies to gridded, geospatial climate data. It is not a route to non-geospatial data such as most socio-economic tabular datasets (for example much of EUROSTAT), which have to be sourced and joined separately.

Mind the size of large requests.

Subsetting is precisely what keeps transfers small, and it is the mitigation for the cost problem introduced in section 1.1 but some datasets are still heavy. The IPCC AR6 Atlas projections, for instance, involve downloads on the order of hundreds of megabytes and extraction times of several minutes when requesting one of the regions entire area and a number of years of data, as the data is dense both spatially and temporally. Scope the region, variables, and period to what you need.

EDR interface is not yet complete.

An OGC EDR interface to the service is under development. Until it is available, use the extract endpoint and the Python stack described here.

Transferability

The service is modular: new CDS datasets are added as backends with minimal code change, so the same access pattern extends to hazards and regions beyond those catalogued today. This is what makes the approach reusable across ClimEmpower's regions and transferable beyond the project.

3.3 Ethical considerations

The data handled by the service is environmental and geospatial; it contains no personal data, so the usual data-protection sensitivities do not arise. The ethical considerations that do apply are about use rather than handling: climate hazard and risk information describes real, often vulnerable, regions, and should be presented with the scenario, model, and maturity caveats set out in section 3.2 so that it informs rather than misleads. This is consistent with ClimEmpower's wider commitment to responsible, equitable use of climate information in support of regional resilience.

4 Key messages and next steps

4.1 Key messages

- **Extract, don't download:** Polytope Feature Extraction returns just the region you ask for, removing the bandwidth and format friction of pulling whole CDS fields.
- **One REST service, many datasets:** the DS2 prototype exposes a catalogue of CDS datasets behind a single consistent *extract* endpoint, mapped to ClimEmpower's hazards in 2.2.
- **CoverageJSON is the interoperable output**; covjsonkit converts it to xarray and Earthkit takes you from request to map in a few lines.
- **Always pin scenario and model for decision-grade results**. dynamic defaults are a convenience, not a substitute for an explicit request.
- **The service is not a production system** (yet): treat it as a capable research prototype.

4.2 What can you do next?

Run the example notebooks against the live service and reproduce the map in 2.3 for your own region by editing the bounding box or supplying a polygon. Identify the datasets that match your region's priority hazards from the table in 2.2, and query /constraints/ to see what each one offers. Then wire the extract endpoint into whatever indicator, risk, or dashboard component you are building. The Python path for prototyping, raw REST for operational integration.

On a more general level, you may read some of the other ClimEmpower educational resources that are available on ClimEmpower web site https://climempower.eu/ and on Zenodo https://zenodo.org/communities/climempower. Some examples:

Climate Risk and Resilience: A Practical Introduction

- **What is it?** A ClimEmpower educational resource providing an accessible overview of climate risk and resilience concepts.
- **Why is it highly relevant?** It complements the methodology by reinforcing the IPCC framework for climate risk assessment and supporting the understanding of hazard, exposure, and vulnerability.

Using Climate Risk Maps for Regional Planning

- **What is it?** A ClimEmpower educational material focused on the application of climate risk maps in regional planning contexts.

- **Why is it highly relevant?** It aligns with the methodology's emphasis on spatial resolution and stakeholder engagement, offering practical guidance on integrating risk maps into adaptation planning.

4.3 Further resources

The following short list identifies key resources that complement this report:

- ClimEmpower Polytope examples (notebooks): https://github.com/ecmwf/climempower-polytope-examples
- DS2 API documentation: https://polytope-dss.ecmwf.int/docs
- Covjsonkit: https://github.com/ecmwf/covjsonkit
- Polytope documentation: https://polytope.readthedocs.io/en/latest/
- Earthkit: https://earthkit-data.readthedocs.io/en/latest/
- Copernicus CDS: https://cds.climate.copernicus.eu/

Table 3 shows the recommended platforms for further reading and access to related project resources.

Table 3: Recommended Platforms

Source platform	Short definition	Relevant materials
ClimEmpower Project Results — climempower.eu/results/	Official ClimEmpower platform for project results and outputs	Project deliverables; resilience recommendations; risk-map information; project tools and software; scenarios; data and knowledge assessments; stakeholder engagement results
ClimEmpower Educational Library — climempower.eu/knowledge-hubs/	ClimEmpower collection of educational materials and selected knowledge resources	Educational materials; reports and guidance; presentations; learning resources; climate-hazard and sector resources; target-group resources; external knowledge resources
CORDIS — cordis.europa.eu	European Commission platform for EU-funded research and innovation	Project descriptions; public deliverables and reports; publications; project partners; project results and outputs
Climate-ADAPT — climate-adapt.eea.europa.eu	European platform for climate change adaptation knowledge and practice	Adaptation strategies and policies; case studies; climate risks and impacts; adaptation options; tools; indicators; research and innovation projects
ClimEmpower Zenodo Community — zenodo.org/communities/climempower	Open repository for ClimEmpower research outputs and project materials	Datasets; journal publications; conference papers and posters; educational materials; other project outputs

5 References

IPCC. (2018). Annex I: Glossary. In *Global Warming of 1.5°C* (S. pp. 541–562). Cambridge University Press. doi:https://doi.org/10.1017/9781009157940.008

Murano I., Del Prete P., Warde A., Verena, P., Kaleta P., Havlik, D. (2026). Reusable libraries and services for climate feature extraction, data processing and indicator building. Deliverable D3.2 of the Horizon Europe project ClimEmpower.