# CDE Transformer – Architecture & Design Notes

## Overview

The CDE Transformer is a Node.js + Express based API service that connects to multiple Common Data Environment (CDE) providers like BIM360, Procore, Viewpoint, Trimble, and Accnoex. It fetches files from these platforms for a given project, merges the results, removes duplicates, and returns a unified, sorted list of files in JSON format.

## Architecture Design Choices

* Express.js for API Routing – Chosen for its lightweight nature, flexibility, and easy setup for REST APIs. Allows rapid prototyping and deployment.
* Provider Abstraction Layer – Each CDE provider has its own class implementing a common fetchFiles(projectId) method. This allows easy addition of new providers without changing the main API logic.
* Aggregation Logic – Implemented in aggregator.js. Uses Promise.all() to call multiple APIs concurrently for better performance. Deduplicates based on projectId + file name and keeps the latest version using updatedAt.
* Config via Environment Variables – API keys, secrets, and ports are read from config.js or .env to keep credentials safe and avoid hard-coding. Supports different settings for local development and production.
* Error Handling – API returns HTTP 400 if required query parameters are missing. HTTP 500 is returned if something goes wrong on the server.

## Trade-offs

* Pros:
* Modular design → easy to maintain and extend.
* Concurrent requests → faster aggregation.
* Provider isolation → less chance of one API breaking the whole system.
* Cons:
* If one provider is slow or unresponsive, it still waits for all requests (can improve with timeout handling).
* Currently no database → all data is fetched fresh every time (good for accuracy, not for speed).

## How I Used the External API Docs

For each provider (BIM360, Procore, etc.), I checked their developer portal for:  
- Authentication flow (OAuth2 or API key usage)  
- File list endpoints (query params, pagination, filtering)  
- Field names for mapping into a unified format

Example:  
BIM360 → Used /projects/:projectId/folders/:folderId/contents endpoint for file listings.  
Procore → Used /rest/v1.0/projects/:project\_id/documents endpoint.

Normalized differences:  
Mapped their custom field names (e.g., last\_modified, version\_number) into our standard format: { id, name, projectId, provider, version, lastModified, size }

## Example Request

GET /v1/files?providers=bim360,procore&project=1234

## Example Response

[  
 {  
 "id": "file-1",  
 "name": "Drawing A.pdf",  
 "projectId": "1234",  
 "provider": "bim360",  
 "version": 3,  
 "lastModified": "2025-08-10T10:15:00Z",  
 "size": 245678  
 }  
]