

DPGs for Climate Action Collection



Framework Compliance Summary:

All DPGs included in the DPGs for Climate Action Collection must meet the requirements outlined in the accompanying [identification framework](#).

To demonstrate compliance, each applying DPG is required to complete a structured submission. The information below reflects the self-reported responses provided as part of that process.

Rural Environmental Registry Registration Module (RER)

DPG Type: Software

DPG Compliance & Profile Page:

<https://www.digitalpublicgoods.net/r/rural-environmental-registry-registration-module>

Description: The Rural Environmental Registry (RER) is a solution for managing declared geospatial environmental information on rural properties to serve as a database for control, monitoring, environmental and economic planning.

Climate Alignment Assessment

Climate-Relevant Problem it Addresses: The RER primarily addresses issues related to deforestation by solving the lack of standardized, reliable, and integrated environmental information on rural properties—a challenge common to many countries. By organizing georeferenced data on land use, native vegetation, protected areas, and environmental compliance into a structured national database, the RER strengthens

environmental monitoring and the effective implementation of legislation. It also connects information about key environmental features, land, and the person responsible for it. This enables evidence-based public decision-making, increases legal certainty for producers and investors, and contributes to global sustainability, climate, and biodiversity goals.

Building on this foundation, the RER strengthens climate governance by enabling more effective monitoring of deforestation, identifying high-risk areas, and supporting the restoration of degraded lands. It generates insights that encourage the maintenance of standing forests and promote more sustainable land management practices. In doing so, it enhances adaptive capacity to climate change, supports emissions mitigation, and increases transparency and accountability in rural environmental governance.

Climate-Relevant Objectives:

- Mitigation
- Adaptation and resilience
- Cross-cutting (data infrastructure, interoperability, digital MRV)

Sector Applications:

- Agriculture, forestry and other land use

Common Deployment Settings

- Regional
- National

International Frameworks it Aligns With

- NDC tracking/implementation
- National inventory systems/transparency frameworks

Climate Outcomes Facilitated**Measurable Outcomes it Can Help Facilitate:**

- Improved data quality and reduced uncertainty
- Strengthened adaptation planning and resilience capacity
- Enhanced inter-agency coordination and governance
- Policy, regulatory or investment decisions informed

Specific Decisions or Actions it Has Helped Facilitate: Environmental regularization and monitoring, land-use planning, restoration, research, and public policy development. The system allows for the identification of forest deficits and surpluses on each landholding. It enables states to enforce environmental laws, requiring farmers to restore forest areas or providing incentives for forest maintenance, such as payments for environmental services.

Deployment Countries:**Adopting Organisations:**

Key Performance or Impact Indicators:

- Improved data quality and reduced uncertainty: Increase in the number and completeness of georeferenced environmental rural property records, such as forest cover, environmentally sensitive areas, deforested areas, amongst others.
- Strengthened adaptation planning and resilience capacity: Total area identified as environmentally sensitive (e.g., rivers, lakes, water springs, hill tops, hill slopes, wetlands)) supporting risk mapping and climate adaptation planning.
- Policy, regulatory, and investment decisions informed: Number and volume of rural credit operations conditioned on CAR compliance; number of environmental embargoes carried out by the Brazilian deforestation combat agency, Ibama, supported by CAR data
- Enhanced inter-agency coordination and governance: Number of federal and state institutions integrating and using CAR data, number of data sharing across agencies, integration of CAR with other official government databases
- Transparency and decision-making support: Number of users accessing or downloading CAR data from public platforms

Note: All measurable indicators above relate to CAR (Cadastro Ambiental Rural), Brazil's national registry of rural properties, established in 2014. RER, a module derived from CAR, was recognised as a digital public good in late 2025. As such, the indicators mentioned above are from CAR and are being used as a proxy to illustrate the potential outcomes that could be achieved through RER adoption.

Technical Assessment

Modular Design: RER follows a microservices architecture composed of 7 independent modules, each implemented as a separate Git submodule with its own repository, build system, Dockerfile, and docker-compose configuration:

1. **Core Backend** (Spring Boot + PostgreSQL/PostGIS) — Main REST API for property lifecycle management
2. **Core Frontend** (Vue.js 3 + Vite + Tailwind CSS) — Web interface for registration and visualization
3. **Map Component** (Vue.js 3 + Leaflet.js) — Standalone reusable map library (dpg-mapa)
4. **Authentication** (Keycloak + Spring Boot + Vue.js 3) — SSO/OIDC authentication and user management
5. **Calculation Engine** (Spring Boot WebFlux + R2DBC + PostGIS) — Reactive geospatial calculation engine with configurable DAG-based workflows
6. **Gateway** (Spring Cloud Gateway) — API Gateway for centralized routing between services
7. **GeoServer** — OGC-compliant WMS/WFS spatial data publishing

Key design patterns and modularity characteristics:

- **Decoupling via API Gateway:** All inter-service communication passes through the Spring Cloud Gateway via REST APIs. No module directly accesses another module's internals.
- **Standardized interfaces:** Every backend service exposes REST/JSON APIs documented with OpenAPI/Swagger, providing machine-readable interface contracts.
- **Database encapsulation:** Each service owns its own database (4 independent PostgreSQL instances). There is no shared database access between services, ensuring strict data isolation and independent schema evolution.
- **Standalone components:** The map_component (dpg-mapa) can be imported into any Vue 3 project independently, without requiring the rest of the RER stack. The calc_engine can be reused for any spatial analysis by configuring new workflows without code changes.
- **Dynamic attribute system:** The backend supports extending data models (property types, attributes, validation rules) through a configurable attribute definition system, without code modifications.
- **Pluggable authentication:** Built on Keycloak (OIDC standard), the authentication module can be replaced with any OIDC-compliant identity provider without affecting other modules.

Data Processing:

Processing Type	Capabilities
Ingestion	Data enters the system through multiple channels. The Core Backend REST API accepts JSON and GeoJSON payloads, as well as multipart/form-data for documents and map images. The Map Component supports WMS layers from GeoServer, GeoJSON features, and user-drawn geometries (polygons, lines, points) via Leaflet and Geoman drawing tools. The Calculation Engine accepts WKT (Well-Known Text) geometry inputs for spatial processing. GeoServer can connect to external spatial data sources and publish them via OGC WMS/WFS standards, enabling ingestion from external geospatial systems.
Validation	Multiple validation layers ensure data accuracy and reliability: - The backend includes an AttributeListValidator that validates all dynamic attributes against their defined types (string, integer, boolean, date, enum) and required/optional rules before persistence. - Flyway manages database schema migrations, ensuring structural integrity across all deployments. - The Calculation Engine validates WKT geometry inputs using JTS (Java Topology Suite) before processing, rejecting malformed geometries. - API request validation uses Jakarta Bean Validation annotations (@NotNull, @Valid, etc.) at the controller level. - The frontend provides a ValidationHelper class with field-level validation for user inputs before submission.

Dissemination	REST APIs serve data in JSON format, documented with OpenAPI/Swagger. - Property receipts are generated as PDF documents via JasperReports. - Spatial data is published through GeoServer as OGC-compliant WMS/WFS services, consumable by any standards-compliant GIS client (QGIS, ArcGIS, web platforms). - The Map Component renders geospatial data visually using Leaflet. - The Calculation Engine returns results as JSON via API, which are then visualized on the interactive map.
---------------	--

Data Extraction Mechanism(s): CSV, SQL, TXT

Interoperability, Integration, and Adapters*: Level 4 Integrated: Uses machine-readable schemas and versioned APIs; external systems can "plug in" and pull climate variables without manual intervention.

RER's modular architecture produces several independently reusable components for climate-relevant systems:

Map Component (dpg-mapa): A standalone Vue.js 3 geospatial visualization library that can be integrated into any Vue application. Useful for NDC dashboards, MRV systems, sectoral emissions monitoring tools, or local planning platforms — any system that needs interactive map visualization with drawing and measurement capabilities.

Calculation Engine: An independent geospatial calculation service with configurable DAG-based workflows. It can be reused for any spatial analysis use case — emissions monitoring, land-use change detection, deforestation tracking, environmental impact assessment — by simply configuring new spatial functions and workflows without code changes.

Authentication Module: The Keycloak-based module provides a complete SSO/OIDC solution that can be shared across multiple climate-relevant systems at national or subnational level, providing unified identity management across platforms.

Dynamic Attribute System: The backend's configurable data model can be adapted to different country requirements (different property types, environmental regulations, reporting standards) without code modifications, making it suitable for adoption across different national contexts.

GeoServer: Provides OGC-standard WMS/WFS endpoints that any GIS tool (QGIS, ArcGIS, other web platforms) can consume directly, enabling interoperability with existing national spatial data infrastructures and NDC monitoring systems.

REST/JSON APIs: All APIs are documented with OpenAPI, enabling integration with any system that speaks HTTP — MRV platforms, NDC dashboards, or implementation tracking tools can consume property data, spatial layers, and calculation results programmatically.

Product Maturity:** Orchestrated / Cloud-Optimized: The solution is designed for modern infrastructure. It includes "Infrastructure as Code" (e.g., Terraform, Helm, or Crawable Data Catalogs) that allows for automated deployment into cloud environments (AWS/Azure/GCP) with built-in scaling and management.

Scalability, performance, and reliability:

Scalability: The microservices architecture allows independent scaling of each component based on load. The Calculation Engine uses reactive architecture (Spring WebFlux + R2DBC) for non-blocking I/O, efficiently handling high concurrent loads with fewer threads. Workflow tasks execute in parallel automatically through the DAG-based engine. The dual-database architecture in the Calculation Engine (configuration DB separate from execution DB) prevents bottlenecks during heavy spatial computation.

Performance: JVM services are optimized for containers (-XX:+UseContainerSupport -XX:MaxRAMPercentage=75.0). Multi-stage Docker builds produce minimal runtime images using Alpine-based base images. The frontend is served via Nginx for high-performance static file delivery. PostGIS spatial indexes enable fast geospatial queries on large datasets.

Reliability: Docker volumes ensure data persistence across container restarts. Database health checks gate service startup to prevent connection failures. Flyway migrations ensure consistent schema state across environments. Each service is isolated in its own container and network segment. Centralized error handling is implemented in all backend services. The system is currently deployed and operational at <https://rer.dataprev.gov.br>.

Disaster Recovery: PostgreSQL data is persisted in Docker volumes. Database dumps are supported (e.g., DB_cardpg.sql for authentication data). Keycloak realm export/import enables authentication state recovery. Git submodules ensure full code versioning and reproducibility of any deployment state.

Computing power:

Minimum deployment requirements: - 4 GB RAM, 2 CPU cores, 10 GB disk space - Supported systems: Linux (recommended), macOS, Windows with WSL2

Resource optimization: Alpine-based Docker images minimize resource footprint. Multi-stage builds exclude build tools from runtime images, reducing image size. JVM is configured with -XX:MaxRAMPercentage=75.0 to respect container memory limits. Nginx Alpine serves the frontend with minimal memory overhead. The reactive architecture in the Calculation Engine (WebFlux) uses fewer threads than traditional blocking I/O models, reducing memory consumption under load.

Low-resource and low-bandwidth considerations: The system can run on a single machine with 4 GB RAM for development and testing. The frontend is a static site served by Nginx, requiring negligible server resources and loading efficiently even on slower connections. The heaviest component is PostGIS spatial calculations, which scale with data volume. For production deployments, 8–16 GB RAM is recommended depending on data volume and concurrent users.

Energy implications at scale: The reactive architecture of the Calculation Engine is inherently more energy-efficient than thread-per-request models, as it maximizes CPU utilization while minimizing idle thread overhead. Parallel workflow execution in the DAG engine reduces total computation time for complex spatial analyses. PostGIS spatial operations are CPU-intensive by nature — this is mitigated by the dual-database architecture that separates lightweight configuration reads from heavy computation workloads, preventing resource contention.

Adoption Evidence and Additional Resources

The Data Sharing Platform and Dashboards of CAR, the Brazilian version of the Registry, allows the visualization and download of a database comprising more than 8 million properties in Brazil and over 180 million polygons representing declared environmental features: <https://consulta.car.gov.br/>

* Interoperability, Integration, and Adapters are evaluated in 5 levels:

Level 1 Isolated: Uses proprietary formats or hardcoded logic; requires custom "glue code" or manual conversion to work with external climate tools.

Level 2 Compatible: Data/software can be exported or integrated using common formats (e.g., CSV, JSON), but lacks automated synchronization or shared metadata.

Level 3 Standardized: Adheres to domain-specific standards (e.g., NetCDF/HDF5 for data, OGC APIs for software) but requires some configuration to link.

Level 4 Integrated: Uses machine-readable schemas and versioned APIs; external systems can "plug in" and pull climate variables without manual intervention.

Level 5 Ecosystem-Ready: Fully modular; follows "FAIR" principles (Findable, Accessible, Interoperable, Reusable) and supports automated cross-platform workflows (e.g., a climate model automatically pulling from your dataset).

**Product maturity from adoption readiness perspective is assessed in 5 levels:

Level 1 Experimental/Raw: The solution exists as "source material" only. It requires significant manual effort, custom scripts, or compilation to become functional. There is no automated setup, and the user must "build" the environment from scratch.

Level 2 Documented/Structured: The solution is organized and includes instructions. Requirements and dependencies are clearly listed, but the setup process is still manual.

Level 3 Portable / Containerized: The solution is "packaged." It uses industry-standard wrappers (e.g., Docker, Parquet, or Standardized Schemas) that allow it to run or be read in any standard environment with a single command or import.

Level 4 Orchestrated / Cloud-Optimized: The solution is designed for modern infrastructure. It includes "Infrastructure as Code" (e.g., Terraform, Helm, or Crawlable Data Catalogs) that allows for automated deployment into cloud environments (AWS/Azure/GCP) with built-in scaling and management.

Level 5 Productized / Plug-and-Play: It offers a zero-friction experience, such as a managed API, a Serverless function, or a Public Data Marketplace listing. A user can gain value or insights within minutes without managing any underlying infrastructure.