National Climate Transparency Platform

The National Climate Transparency Platform (NCTP) is your country's gateway to ensure robust information gathering and reporting following the Enhanced Transparency Framework (ETF) of the Paris Agreement and to assist in the management of progress and implementation of your country's Nationally Determined Contribution (NDC).

The NCTP does this by helping governments optimise the management of information related to climate change actions and support across ministries, agencies, and departments. It also allows for the option to easily share this information with parties outside government, such as UNFCCC, development partners, education institutions, etc. By centralizing essential data in a single digital location, the NCTP significantly reduces workload, saving thousands of person-hours. It is user-friendly access and standardized reporting tools for ETF information further streamline operations and enhance efficiency.

The NCTP is an integral part of UNDP's open-source Digital for Climate software ecosystem. This Digital Public Good has an open-source codebase and is designed to encapsulate the essence of effective climate action management, enabling countries to configure, adapt and build on it to meet national requirements.

The NCTP has 7 key features.

- **Systemic Information Structure**: The different modules in the NCTP follow the information structure often used when countries and development partners plan and implement climate actions. This includes general information on broad climate actions and the more detailed information on sectoral programmes, projects, and activities that fall under each climate action.
- **NDC Progress Tracking**: Following the Paris Agreement's Modalities Procedures and Guidelines of the ETF, the NCTP allows users to effortlessly tracks and reports progress of NDC mitigation and adaptation actions, plus support that is needed and received. Including enables standard reporting using the critical Common Tabular Formats of the ETF.
- **Tracking National Sustainable Development**: Every country and ministry has specific development planning that commonly overlaps with climate actions. The NCTP allows for the inclusion of customised Key Performance Indicators for actions, programmes, projects, and activities that align with national goals and planning.
- **GHG Emissions Projections**: The NCTP has a module to help governments project GHG emissions based on different scenarios for sectors and subsectors using standard exponential modelling and/or by inputting modelling data from other software and models. Additionally, the module enables the input of GHG inventory results, organized by categories and subcategories, ensuring comprehensive and flexible emissions analysis.
- **Validation**: To ensure quality control, governments can assign users to validate the information provided for each climate action, programme, project, activity and support.

- **Standard Reporting**: The information for climate action, programme, project, activity, and support is automatically consolidated and reported in the dashboard, along with reporting information using important Common Tabular Formats of the ETF.
- **Assigning Users**: the NCTP allows governments to assign users based on the entities they work for (e.g. government departments, education institutions, consultants…) and what information they should have access to both read and input (e.g. for specific sectors…).

More information about the project's background, vision, policy context, support provided can be found in the demo site https://transparency-demo.carbreg.org/. Governments and project partners, please contact the UNDP DPG team digital4planet@undp.org through your UNDP country office to request a walkthrough demonstration and to discuss further support and collaboration.

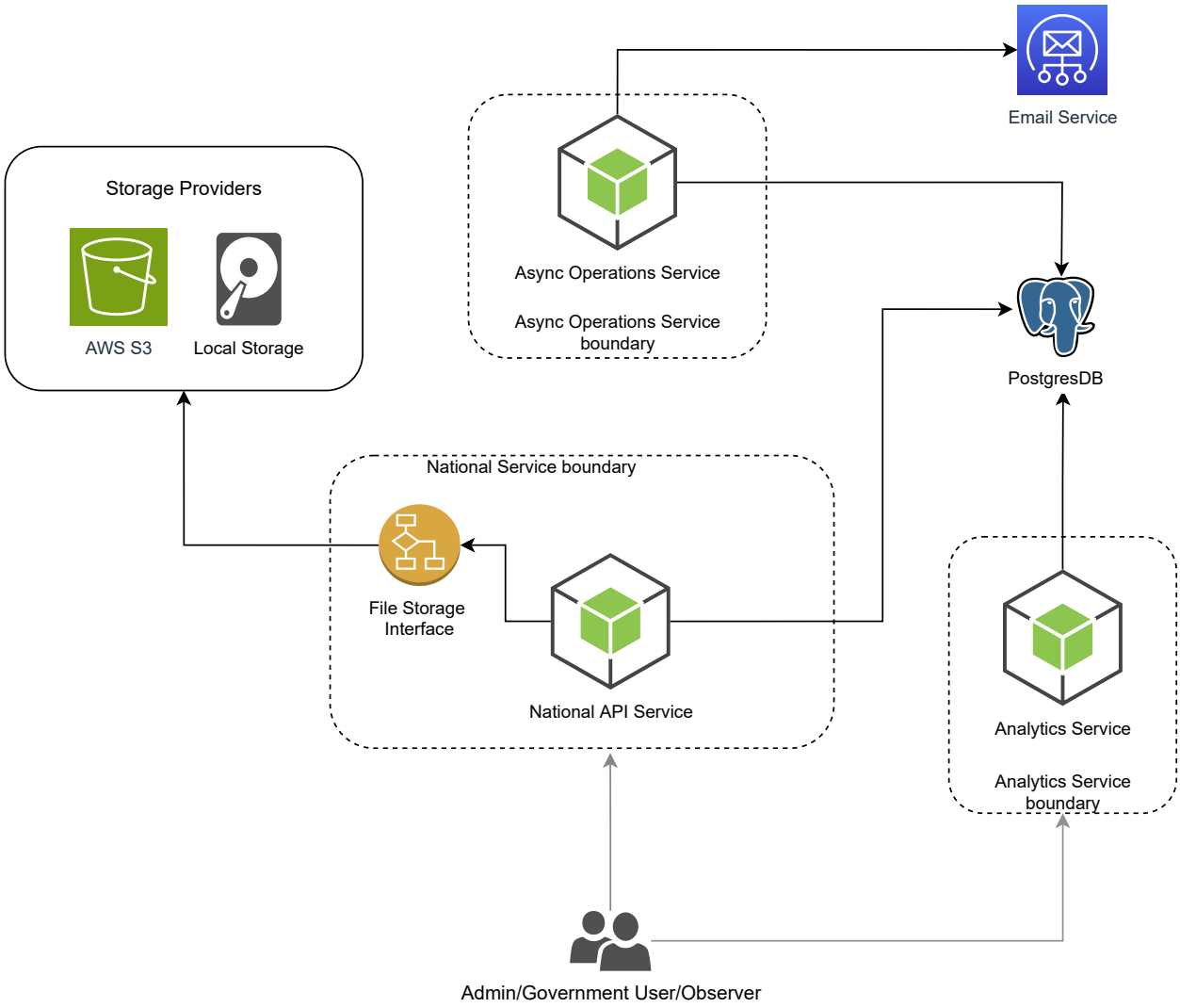
Index

Below contents are planned to be updated by the third quarter of 2024 based on user feedback and recent change in international requirements.

Standards

This codebase aims to fulfill the Digital Public Goods standard, adheres to the UNDP Data Principles, and it is built according to the Principles for Digital Development.

System Architecture



Deployment

System services can be deployed in 2 ways.

- **As a Container** - Each service boundary containerized into a docker container and can deploy on any container orchestration service. Please refer Docker Compose file
- **As a Function** - Each service boundary packaged as a function (Serverless) and host on any Function As A Service (FaaS) stack. Please refer Serverless configuration file

External Service Providers

All the external services access through a generic interface. It will decouple the system implementation from the external services and enable extendability to multiple services.

File Service

Implemented 2 options for static file hosting.

1. NestJS static file hosting using the local storage and container volumes.
2. AWS S3 file storage.

Can add more options by implementing file handler interface

Change by environment variable `FILE_SERVICE`. Supported types are `LOCAL` (default) and `S3`.

Project Structure

```
.
├── .github                       # CI/CD [Github Actions files]
├── deployment                    # Declarative configuration files for
initial resource creation and setup [AWS Cloudformation]
├── backend                       # System service implementation
    ├── services                  # Services implementation [NestJS
application]
        ├── src
            ├── national-api      # National API [NestJS module]
            ├── stats-api         # Statistics API [NestJS module]
            ├── async-operations-handler   # Async Operations Handler [NestJS
module]
        ├── serverless.yml        # Service deployment scripts [Serverless +
AWS Lambda]
├── web                           # System web frontend implementation
[ReactJS]
├── .gitignore
├── docker-compose.yml            # Docker container definitions
└── README.md
```

Run Services As Containers

- Update docker compose file env variables as required.
  - Currently all the emails are disabled using env variable `IS_EMAIL_DISABLED`. When the emails are disabled email payload will be printed on the console. User account passwords needs to extract from this console log. Including root user account, search for a log line starting with `Password (temporary)` on national container (`docker logs -f climate-transparency-national-1`).
  - Add / update following environment variables to enable email functionality.
    - `IS_EMAIL_DISABLED`=false
    - `SOURCE_EMAIL` (Sender email address)
    - `SMTP_ENDPOINT`
    - `SMTP_USERNAME`
    - `SMTP_PASSWORD`
  - Use `DB_PASSWORD` env variable to change PostgreSQL database password
  - Configure system root account email by updating environment variable `ROOT EMAIL`. If the email service is enabled, on the first docker start, this email address will receive a new email with the root user password.
- Add user data
  - Update users.csv file to add users.
  - When updating file keep the header and replace existing dummy data with your data.

    - These users will be added to the system each docker restart.
- Run `docker-compose up -d --build`. This will build and start containers for following services:
    - PostgresDB container
    - National service
    - Analytics service
    - Async Operations Handler service
    - Migration service (This service will shutdown automatically once db migration script execution is completed)
    - React web server with Nginx.
- Web frontend on http://localhost:9030/
- API Endpoints,
    - http://localhost:9000/national/
    - http://localhost:9100/stats/
- Swagger documentation will be available on http://localhost:9000/local/national

Run Services Locally

Follow same steps mentioned above to run the services locally using docker.

Deploy System on the AWS Cloud

- Execute to create all the required resources on the AWS.

```
aws cloudformation deploy --template-file ./deployment/aws-formation.yml --
stack-name ndc-transparency-basic --parameter-overrides EnvironmentName=<stage>
DBPassword=<password> --capabilities CAPABILITY_NAMED_IAM
```

- Setup following Github Secrets to enable CI/CD
    - `AWS_ACCESS_KEY_ID`
    - `AWS_SECRET_ACCESS_KEY`
- Run it manually to deploy all the lambda services immediately. It will create 2 lambda layers and following lambda functions,
    - national-api: Handle all user and program creation. Trigger by external http request.
    - async-operations-handler: Handle all async operations such as managing notification emails.
    - setup: Function to add initial system user data.
- Create initial user data in the system by invoking setup lambda function by executing

```
aws lambda invoke \
    --function-name ndc-transparency-services-dev-setup --cli-binary-format
raw-in-base64-out\
    --payload '{"rootEmail": "<Root user email>","systemCountryCode": "<System
country Alpha 2 code>", "name": "<System country name>", "logoBase64": "<System
country logo base64>"}' \
    response.json
```

Web Frontend

Web frontend implemented using ReactJS framework. Please refer getting started with react app for more information.

Localization

- Languages (Current): English, French
- Languages (In progress): Spanish

For updating translations or adding new ones, reference https://github.com/undp/national-climate-transparency/tree/main/web/src/locales/i18n

API (Application Programming Interface)

For integration, reference RESTful Web API Documentation documentation via Swagger. To access

- National API: `APP_URL`/national
- Status API: `APP_URL`/stats

Resource Requirements

Resource	Minimum	Recommended
Memory	4 GB	8 GB
CPU	4 Cores	4 Cores
Storage	20 GB	50 GB
OS	Linux Windows Server 2016 and later versions.	

Note: Above resource requirement mentioned for a single instance from each microservice.status.APP_URL

Status Page

Coming soon...

User Manual

Coming soon...

Demonstration Video

Coming soon...

Help Tools

Coming soon...

Data Sovereignty

The code is designed with data sovereignty at its core, empowering nations and organizations to have greater control and governance over their environmental data. Here are the key points highlighting how this system promotes data sovereignty:

- **Local Control**:
  - Allows nations and entities to store, manage, and process their data locally or in a preferred jurisdiction, adhering to local laws and regulations.
- **Open-Source Architecture**:
  - Facilitates transparency, customization, and control over the software, enabling adaptation to specific legal and regulatory requirements.
- **Decentralized Infrastructure**:
  - Supports a decentralized data management approach, minimizing reliance on external or centralized systems.
- **Standardized yet Flexible Protocols**:
  - Provides standardized protocols for data management while allowing for local customization, aligning with the diverse legal landscapes.
- **Secure Data Sharing and Access Control**:
  - Implements robust access control and secure data sharing mechanisms, ensuring only authorized entities can access or alter the data.
- **Audit Trails**:
  - Offers comprehensive audit trails for all data transactions, ensuring traceability and accountability in data handling and reporting.
- **Enhanced Privacy Compliance**:
  - Helps in ensuring compliance with privacy laws and regulations by providing tools for secure data handling and consent management.

By integrating these features, the code significantly contributes to achieving data sovereignty, promoting a more localized and accountable management of environmental data in line with the goals of the Paris Agreement.

Governance and Support

This code is managed by United Nations Development Programme as custodian. Governments and potential partners working with UNDP can:

- Request access to the demo site.
- Request a live demonstration.
- Customize and install as a sovereign system.
- Explore potential collaboration and support.

For any questions, contact us at digital4planet@undp.org.