|  |  |
| --- | --- |
| LEAF-Toolbox Landscape Evolution and Forecasting Toolbox |  |
|  | LEAF Service System Design |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Name | Company | Date | Signature |
| Prepared by : | Richard Fernandes | CCRS | June 16, 2021 |  |
| Checked by : | xx | CCRS |  |  |
| Approved by : | Darren Janzen | CCRS |  |  |

|  |  |
| --- | --- |
| Document reference : |  |
| Issue.Revision : | 1.0 |
| Date : | 16/06/2021 |
| Client :  EN-MAG-MOD.DT-GEN-034-v1.3 | NRCan/CSA |
| Ref., Tender : | GEODE GRIP |

Document Change Record

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Iss. | Rev. | Date | Reason | Comments |
| 1 | 0 | June 16, 2021 | Creation of the document | Complete revision of LEAF-TN-002-CCRS |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## Introduction

The Landscape Evolution and Forecasting (LEAF) service is a computer application capable of producing geo-coded raster layers for a set of vegetation biophysical parameters from either Sentinel 2 Multispectral Imager (MSI) or Landsat 8 Operational Land Imager (OLI) satellite imagery.

User requirements for LEAF(LEAF-TN-001-CCRS) include a functionality via desktop and mobile compute with the possibility to extend the service to REST compliant system requests.

An initial cloud service system design defined during the NRCan/CSA GEODE GRIP project (LEAF-TN-002-CCRS) conceptualized the service as embedded within a cloud computing environment. Prototyping of the design found that the system did not scale in a cost effective manner and could not provide new user visualization requirements that arose during the project (Rogobete et al., 2019).

The purpose of this document is to provide the new system design for implementation of the LEAF service, the LEAF-Toolbox, that relies on an API to an external cloud system that enables geo-processing (specifically Google Earth Engine). The system design is capable of satisfying Goal requirements for system functionality, latency, maintenance and documentation specified in LEAF-TN-001-CCRS.

### Vegetation Biophysical Parameters

The vegetation biophysical parameters to be generated, together with specifications, have been identified by the Canadian Space Agency, the SEN4SCI user consultation exercise and the Global Climate Observing System. Parameters are defined in Table 1 with spatial, temporal and thematic performance requirements given in Table 2.

Table 1. Definition of vegetation biophysical parameters within LEAF.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parameter | Abbrev. | Definition | Units | Range |
| Leaf chlorophyll content | Cab | Mass of chlorophyll a and b per unit LAI. | G chlorophyll a+b/m-2 half foliage surface area | 0-100 |
| Leaf water content | Cw | Mass of H20 per unit LAI. | G H20/m-2 half foliage surface area | 0-10 |
| Fraction absorbed PAR | fAPAR | Fraction of incident PAR absorbed by vegetation at ~10am local standard time | fraction | [0,1] |
| Fraction cover | fCover | Fraction of canopy cover projected on local horizontal datum. | fraction | [0,1] |
| Leaf Area Index | LAI | Have the total foliage surface area per unit ground area projected on local horizontal datum. | M2 foliage/m2 horizontal ground area | 0-20 |

Table 2. User requirements for biophysical parameters, Ordinal thematic requirement corresponds to estimates within a user defined mapping region that, after bias correction, meets threshold requirements (i.e. relative ranking of estimates is correct).

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Parameter | Spatial | | | Temporal | | | Thematic | | |
|  | Base | Thresh. | Goal | Base | Thresh. | Goal | Base | Thresh. | Goal |
| Cab | 10ha | 1ha | 0.5ha | Peak | 30d | 10d | Ordinal | 20 | 10 |
| Cw | 10ha | 1ha | 0.5ha | Peak | 30d | 10d | Ordinal | 0.2 | 0.1 |
| fAPAR | 10ha | 1ha | 0.5ha | Peak | 5d | 1d | Ordinal | Sup(0.1,10%) | Sup(0.05,5%) |
| fCover | 10ha | 1ha | 0.5ha | Peak | 30d | 10d | Ordinal | Sup(0.2,20%) | Sup(0.1,10%) |
| LAI | 10ha | 1ha | 0.5ha | Peak | 30d | 10d | Ordinal | Sup(1,20%) | Sup(0.5,10%) |

### Design Philosophy

The philosophy of LEAF is to process a set of satellite images sharing similar spectral, spatial and temporal characteristics in a manner that scales with computer resources and provides flexibility in terms of the applied inversion algorithm. To do so LEAF:

1. Relies on user selected regression based algorithms calibrated using machine learning. The LEAF toolbox includes a class for these regression algorithms that includes the calibration functionality.
2. Makes use of analysis ready data products, ancillary data and previously calibrated and tested inversion algorithms.
3. Allows for user selected calibration of training databases applicable to multiple products.
4. Allows for product based regularization of retrievals to reduce uncertainty due to the ill-posed nature of the regression algorithms. The LEAF toolbox includes a class for regularization.
5. Implements asynchronous parallel processing during calibration and parameter estimation phases to provide partial response to product generation requests as available
6. Provides output the satisfies the CEOS CARD4DL requirements for analysis ready data.
7. Provides visualization and products in a free and open manner satisfying the Governent of Canada’s Open Data (<https://open.canada.ca/en/open-data>) and Open Science policies (https://science.gc.ca/eic/site/063.nsf/eng/h\_98054.html)

### Architecture Overview

The LEAF-Toolbox architecture (Figure 1, Table 3 ) is based on the philosophy that the Toolbox should rely as much as possible on external cost-effective services for system interfaces to users and managers, ARD data access, compute and visualization. Ideally cost-effective implies zero cost other than Toolbox development, maintenance and documentation and archive of output to satisfy the free information requirement. However, nominal costs that do not scale up with system demand are acceptable (internet access for development and management, communications of manuals and system performance, platforms for research, prototyping or load testing).



Figure 1. LEAF-Toolbox System Architecture

Table 3. LEAF-Toolbox System Architecture description.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sub-System | Description | Baseline | Threshold | Goal |
| Governance | People and processes that govern system implementation and function. | Research group management via manual control of user access and updates from research git. | Operations group management via manual control of user access and update from version controlled operations git. | Distributed management by user clusters with user community dashboard with access to Threshold solutions. |
| Algorithm Scientists | Provided mapping algorithms. | SL2P regression algorithm | Modified versions of SL2P regression algorithms producing CARD4DL data. | Algorithms specified in open ML formats producing CARD4DL data. May include server optimized algorithms. |
| System Server | Archive of algorithms, processing and control, security, ingest. | Google Earth Engine (GEE) capable of executing SL2P. | GEE capable of executing user specified linear and non-linear network regressions. | GEE with extensions to Cloud for user specified generic regressions based on Tensorflow and active learning recalibration, REST supported data output |
| ARD Providers | Input data provision | Default GEE collections | Default GEE collections and access to user specified ARD collections in GEE. | Ability to access external ARD collections. |
| Client | User interface, visualization, archive, user documentation. | GEE code editor API | GEE app | Provision of user access, control, visualization in user web platform |
| Sub-system communication | Algorithm, data and system request passing. | Google Drive Export and Ingest only. Client-server within GEE code editor only. | Export/Ingest to authenticated store. Client-server with Python APIs using Google credential services | Data transfer via REST APIs to GEE. User managed/provisioned credentials. |

### System Loading

The system should be able to sustain parallel service requests with differing levels of product quantities (). Denials of service at a threshold or goal level should be at the point of request rather than upon point of failure. Note that output scenes or data volumes may exceed input ARD granule counts or data volumes due to the use of scene forecasting and prediction.

Table 4. System load requirements.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Category | Baseline | Threshold | Goal | Description |
| Number of systems running | 2 | 10 | 100 | Number of deployed systems on single clud service. |
| Number of users per system | 10 | 1000 | unlimited | Number of registered users |
| Maximum number of parallel service requests per system | 2 | Max(10,1% of user base) | Max(100,10% of user base) | Number of active service requests within system. |
| Maximum number input of L1A scene per request | 1000 | 1500 | 1500 | Assumes processing begins at L1B. Number of scenes in one request (~1500 scenes cover Canada for one sensor; 5000 ensures coverage from both sensors and territorial water). Includes scenes produced by temporal interpolation. |
| Maximum number output scenes per request | 1000 | 1500 | 5000 | Includes scenes produced by temporal interpolation. |

### System Latencies

Service requests should be ingested, verified with respect to data, storage and compute availability (e.g. request satisfies system load requirements) and accepted within 10minutes (5minutes). For accepted requests, final products should be saved to a designated archive within 24hrs (8 hrs) of attempted resource allocation. Table 54 lists latencies for a single product. The list includes latencies that are asynchronous or may not be required for a given instance, such as L2A Product generation. The list does not explicitly envision system caches for input and output products.

Table 5. System latency requirements (hours:minutes). Note that some processes may be asynchronous or not required (e.g. L2A Product generation).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Service | Trigger | Baseline | Threshold | Goal | Description |
| Product request accepted | User submits product request | 0:10 | 0:10 | 0:10 | System verifies it has compute and data resources. |
| Data Ingest | Product request accepted | 12:00 | 8:00 | 3:00 | Required data is ingested and staged for compute |
| L2A product generated | Data Ingest completed. | 4:00 | 2:00 | 1:10hr | Generation of L2A products from L1A input |
| L2A Product archived | L2A Product generated | 1:00 | 1:00 | 0:10 | Intermediate L2A produyct arrives at archive |
| Training Database Produced | User submits product request | 2:00 | 2:00 | 1hr | Training data generated as required using simulations and input data. |
| Calibration of inversion algorithm | Training database completed | 2:00 | 1:00 | 0:10 | Inversion algorithm calibrated using training data. |
| Initial estimation of biophysical parameters | Inversion algorithm calibrated | 1:00 | 1:00 | 0:10 | Initial products estimated using inversion algorithms. |
| Regularization of inversion algorithm | Initial estimation completed | 1:00 | 1:00 | 0:10 | Inversion algorithms revised using initial estimates and ancillary data. |
| Product generated | Inversion algorithm regularized | 1:00 | 1:00 | 0:10 | Final product estimated using revised inversion algorithms. |
| Product archived | Product generated | 12:00 | 8:00 | 3:00 | Final product arrives at dedicated archive. |
| Total |  | 33:00 | 22:10 | 8:00 |  |

### System Resources

Resources are to be provisioned by free and open access on a cloud service assuming the user has ensured appropriate access (via internet connections and/or cloud fees) to code and data repositories repositories. Code and data repositories are to be open access and code repositories are to be free.

The user should have the option for executing the system at low, debug, nominal, high and maximum performance/cost levels. These levels are described in table 5.

Table 6. System resource levels.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Resource Level | Processing | Parallel | End-to-End Latency | Target |
| Low | Synchronous | No | unconstrained | Training, entry level users |
| Debug | Asynchronous | No or Yes | Baseline | Developers, acceptance testing |
| Nominal | Asynchronous | Yes | Threshold | Systematic production |
| High | Asynchronous | Yes | Goal | Episodic production, monitoring |
| Maximum | Asynchronous | Yes | unconstrained | Benchmarking |

### Storefront Interface

The system is designed to provide an interactive operator interface (GUI) but an event based user interface driven by open geospatial consortium compliant messaging. The provision of an interactive user interface or visualization tools of intermediate or derived products is outside the scope of this system. Table 6 list user interface requirements.

Table 7. User interface requirements.

|  |  |
| --- | --- |
| Requirement | Description |
| Execution | System execution should be via XML requests or other OGC compliant requests. |
| Execution | The user should be able to halt and resume the system on command or by a trigger (e.g. maximum time, data availability) with behaviour specified within the service request. |
| Execution | The user should be able to halt and resume the system on command or by a trigger (e.g. maximum time, data availability) with behaviour specified within the service request. |
| Control | Control and parameter inputs and system logs should be recorded in JSON files. |
| Control | A GUI (ideally integrated with the system execution GUI), should be provided for assembling and viewing relevant contents found in JSON files from direct input or input from standard sources (e.g. EXCEL , .txt, .csv tables and OGC compliant products and their headers). |
| Control | A GUI in the form of a LEAF toolbox should be provided for configuring tools, based on available code, for data processing. |
| Administration | Notifications should be provided regarding system operation, resource use and costing, and anomalies to a designed device (and optionally a GUI) in addition to JSON files. |
| Administration | When executed in debug mode the system should also expose intermediate results (training datasets, calibrated inversion algorithms, intermediate products) and supplementary information (as JSON files) to the user in a user provided debug archive. |



















### Maintenance and Infrastructure

The system should be provided with an installer or clear installation instructions for both standalone workstation or cloud instances.

The system should be maintained remotely of users either by updates to code repositories or cloud services. Revisions should preserve the ability to generate previous versions of outputs.

The system should maximize flexibility in terms of adding or modifying simulation codes, inversion algorithms and regularization algorithms.

The system should be able to operate at “Low” and “Debug” latency level on a desktop workstation with 32Gbytes free RAM and sufficient disk storage after installation of the system on a Linux environment. Users are responsible for installation of ancillary resources but these should be free or nominal in cost.

The system should be able to operate at “Low” , “Debug” and “Threshold” latency level on a cloud service assuming all inputs are not resident in low latency (<10minutes) cloud storage.

The system should be able to operate at “Goal” latency level on a cloud service assuming all inputs are resident in low latency (<10minutes) cloud storage and model recalibration is not required.

### Documentation and Help Resources

The system should be documented using GITHUB readable formats and indexed in GITHUB (even if the code repository differs). Documentation should include:

1. This document.
2. The System Architecture document.
3. Installation and uninstallation procedures including dependencies.
4. A verification test case and document.
5. A nominal use case document.
6. Debug and troubleshooting document.
7. GITHUB interactive comments and responses.

Help should support command shell and GUI operations. Command shell help should follow the practices of MATLAB. GUI help should involve a hover/pop-up access to GITHUB readable formatted descriptions.

References

Fernandes, R.A. LEAF Service User Requirements, LEAF-TN-001-CCRS, Geomatics Canada Open File, xx, Government of Canada, doi xx.

Fernandes, R.A. LEAF Cloud Service Systems Design LEAF-TN-002-CCRS, Geomatics Canada Open File, xx, Government of Canada, doi xx.

Rogobete, J. E., Djamai, N., Fernandes, R.A., 2021, Prototype of the SL2P Processor on Amazon Web Services, Geomatics Canada Open File xx, Government of Canada, doi xx, in review.