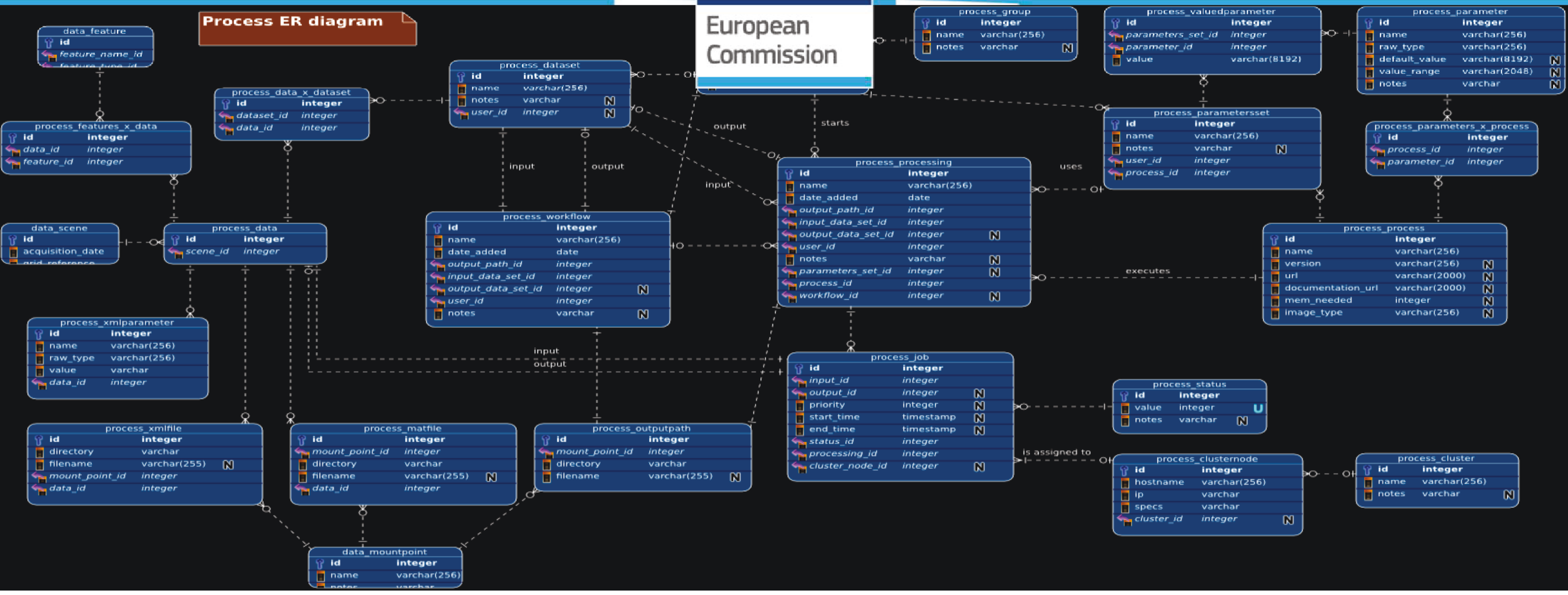


# A new spatial database and software layer supporting the JRC image information query (IQ) system

Marco Scavazzon, Martino Pesaresi

2013

Report EUR 26524 EN

European Commission
Joint Research Centre
Institute for the Protection and Security of the Citizen (IPSC)

Contact information
Martino Pesaresi
Address: Joint Research Centre, Via Enrico Fermi 2749, TP 267, 21027 Ispra (VA), Italy
E-mail: martino.pesaresi@jrc.ec.europa.eu
Tel.: +39 0332 789524
Fax: +39 0332 785154

http://www.jrc.ec.europa.eu/

This publication is a Reference Report by the Joint Research Centre of the European Commission.

JRC87097

EUR 26524 EN

ISBN 978-92-79-35530-1 (pdf)

ISSN 1831-9424 (print)

doi: 10.2788/78199

Luxembourg: Publications Office of the European Union, 2013

# Introduction

The JRC image information query (IQ) database is the data archive of the IQ system. The database contains information related to:

- Satellite imagery from various providers
- Images derived from processing of satellite imagery
- Procedures of image processing
- Parameters used for image processing
- Processings and workflows (combinations of processings)
- Status of the computing system / Linux cluster

The coupled software layer is used to:

- Automatically generate the database structure
- Setup and run the processings on the Linux cluster
- Invoke the proper image processing executable
- Run the processes with a set of user defined parameters

# Objectives

During 2013 the structure of the IQ database has been revised, in order to be able to store new information as follows:

- Metadata of satellite imagery used in the processings
- Processes and set of parameters used in the image processing experiments
- Metadata of the output of the combination of processes, parameters and images

Moreover, the new database structure has been designed to standardize as much as possible the different images coming from various providers.
Every provider has its own standard for satellite images metadata.

Since the team needed to process huge quantities of satellite imagery coming from different providers, an appropriate way to standardize all the non-homogeneous input data through the use of specialized importing procedures, had to be found.

## Features of the database

The structure of the database is divided in 3 sections: data, processes and importer. The database structure is made of 47 tables, plus 2 tables dedicated to spatial references and geometries.

In the data section (fig. 1: Data ER diagram) the following information is stored:

- Satellite sensors and platforms
- Scene details (e.g. acquisition date and sun azimuth/elevation)
- Image files of satellite imagery and related derived image files (with file path, number of bands, data type, file size information)
- Image's driver (replicates GDAL's information, with values like TIFF, BIL or ECW)
- Mount point of the images, for a transparent access to the files between Windows and Linux computers
- Image features and related characteristics (name, type and geometry)
- Ownership information, sharing level, copyright and licensing related to the image feature
- Web Map Service (WMS) and Tile Map Service (TMS) available for feature production

In the processes section (fig. 2: Process ER diagram), the database saves all the metadata related to:

- Image processing procedures
- Parameters' set used in the various experiments
- Processings (and workflows) executed from the IQ system
- Datasets used in the processings
- Computation system status, with monitoring of the jobs executed within the Linux cluster
- Relationships and associations between input and output data of every processing
- Users and groups involved in the processings

Finally, with regards to the importer section (fig. 3: Importer class diagram), the database contains the following information:

- CID Portal's satellite imagery imported, updated and cloned in the IQ database
- CID Portal's satellite imagery converted in our database structure, enhanced with additional information useful for ISFEREA's processing purposes.

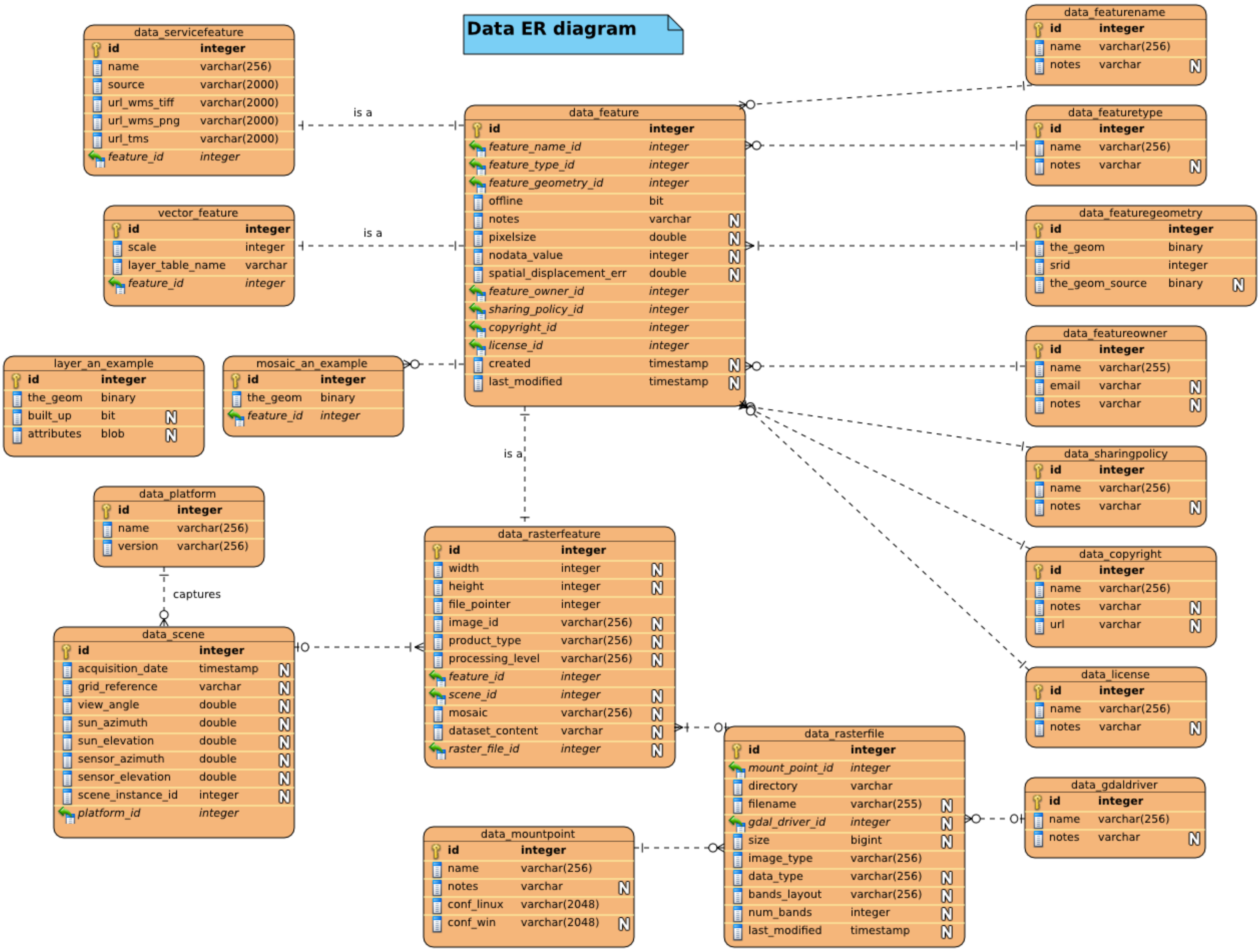


Figure 1: Data ER diagram

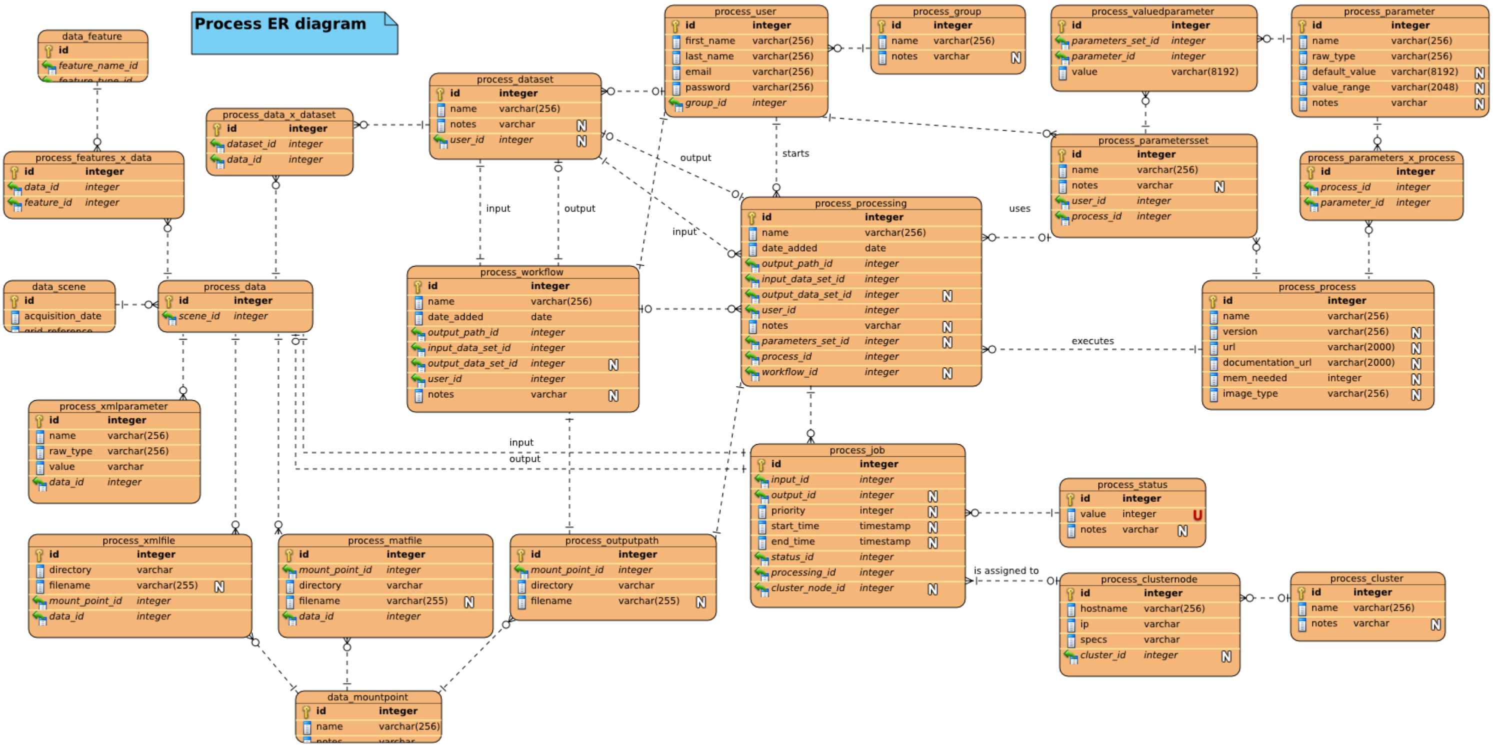


Figure 2: Process ER diagram

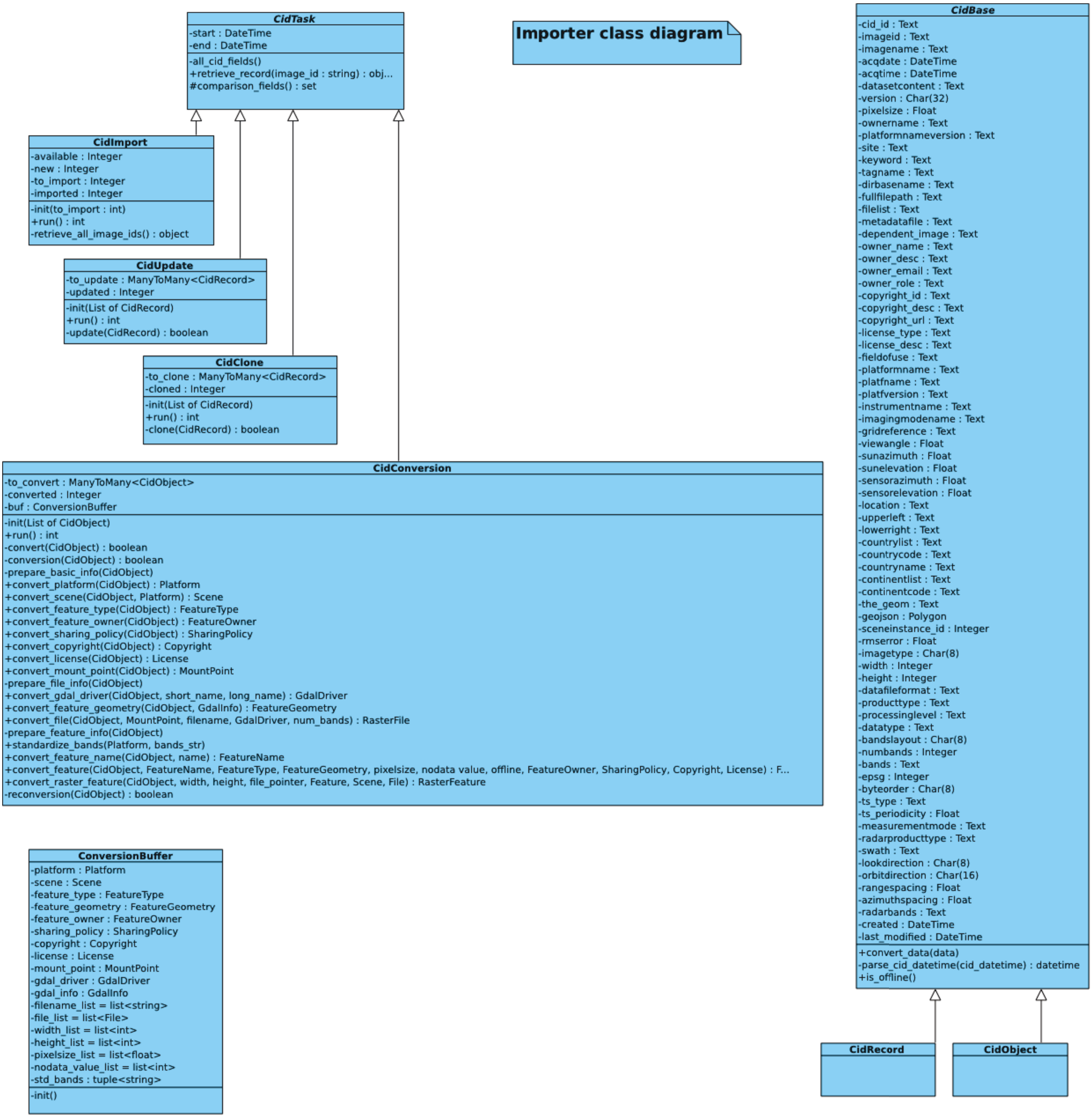


Figure 3: Importer class diagram

# Features of the software layer

The software layer of this project is used to:

- Dynamically generate the database structure, starting from the object definition in python
- Import in the database the satellite imagery metadata stored in CID Portal (fig. 4: CID use cases; fig 5: CID import activity diagram)
- Convert the CID information in a structure suitable for the processing needs, and enrich this structure with additional useful information like resolution, image size and bands of the image
- Generate a dataset of satellite imagery that will be processed by the Linux cluster (fig. 6: Process and Dataset class diagram)
- Create and edit a set of parameters that will be used to process the data
- Generate processings and workflows (chains of processings) (fig. 6)
- Start processings in the Linux cluster (fig. 6)
- Invoke MATLAB executables in the Linux cluster, and keep track of their status (fig. 7: Invoker activity diagram)
- Translate the information contained in the database, in parameters that are understood by the MATLAB executables (fig. 8: Matlab translator class diagram)
- Add, update and delete the data in the database, through a web interface
- Define processings and complex processing chains (or workflows) using the application's interface (as an example, see fig. 9: workflow 4)

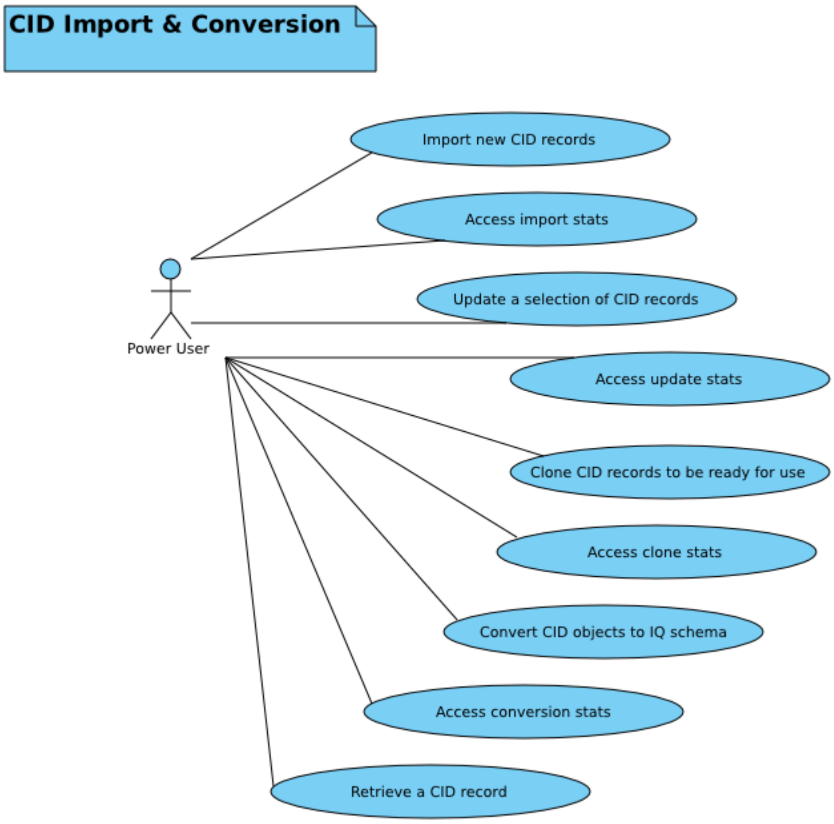


Figure 4: CID use cases

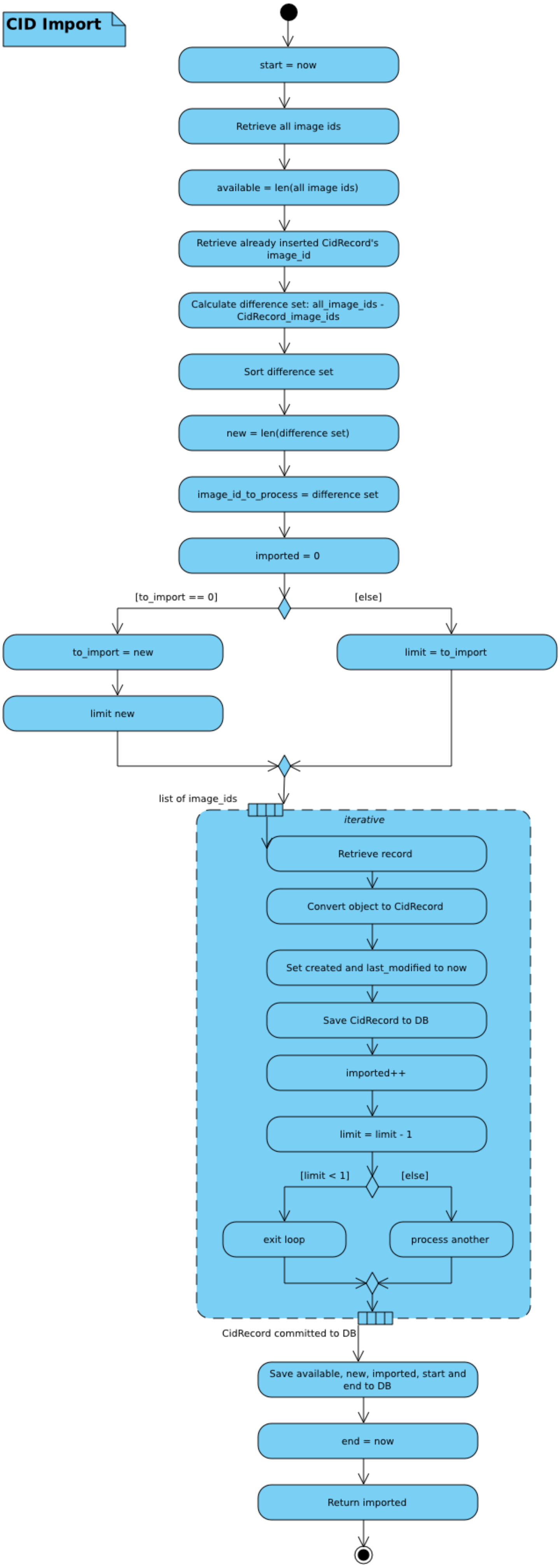

CID Import
start = now
Retrieve all image ids
available = len(all image ids)
Retrieve already inserted CidRecord's image_id
Calculate difference set: all_image_ids - CidRecord_image_ids
Sort difference set
new = len(difference set)
image_id_to_process = difference set
imported = 0
[to_import == 0]
[else]
to_import = new
limit new
limit = to_import
list of image_ids
iterative
Retrieve record
Convert object to CidRecord
Set created and last_modified to now
Save CidRecord to DB
imported++
limit = limit - 1
[limit < 1]
[else]
exit loop
process another
CidRecord committed to DB
Save available, new, imported, start and end to DB
end = now
Return imported

Figure 5: CID import activity diagram

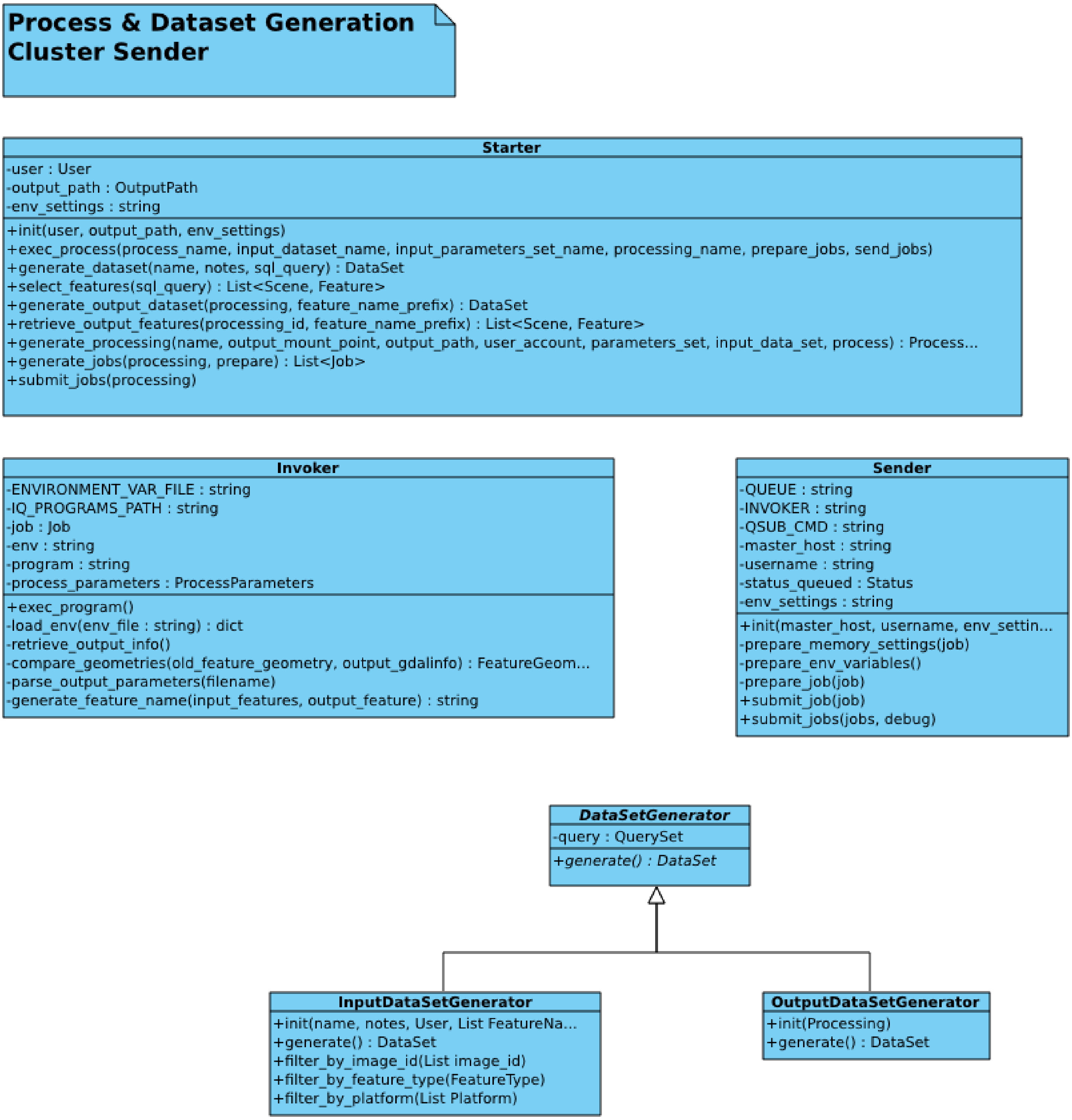

Process & Dataset Generation
Cluster Sender
Starter
-user : User
-output_path : OutputPath
-env_settings : string
+init(user, output_path, env_settings)
+exec_process(process_name, input_dataset_name, input_parameters_set_name, processing_name, prepare_jobs, send_jobs)
+generate_dataset(name, notes, sql_query) : DataSet
+select_features(sql_query) : List<Scene, Feature>
+generate_output_dataset(processing, feature_name_prefix) : DataSet
+retrieve_output_features(processing_id, feature_name_prefix) : List<Scene, Feature>
+generate_processing(name, output_mount_point, output_path, user_account, parameters_set, input_data_set, process) : Process...
+generate_jobs(processing, prepare) : List<Job>
+submit_jobs(processing)
Invoker
-ENVIRONMENT_VAR_FILE : string
-IQ_PROGRAMS_PATH : string
-job : Job
-env : string
-program : string
-process_parameters : ProcessParameters
+exec_program()
-load_env(env_file : string) : dict
-retrieve_output_info()
-compare_geometries(old_feature_geometry, output_gdalinfo) : FeatureGeom...
-parse_output_parameters(filename)
-generate_feature_name(input_features, output_feature) : string
Sender
-QUEUE : string
-INVOKER : string
-QSUB_CMD : string
-master_host : string
-username : string
-status_queued : Status
-env_settings : string
+init(master_host, username, env_settin...
-prepare_memory_settings(job)
-prepare_env_variables()
-prepare_job(job)
+submit_job(job)
+submit_jobs(jobs, debug)
DataSetGenerator
-query : QuerySet
+generate() : DataSet
InputDataSetGenerator
+init(name, notes, User, List FeatureNa...
+generate() : DataSet
+filter_by_image_id(List image_id)
+filter_by_feature_type(FeatureType)
+filter_by_platform(List Platform)
OutputDataSetGenerator
+init(Processing)
+generate() : DataSet

Figure 6: Process and Dataset class diagram

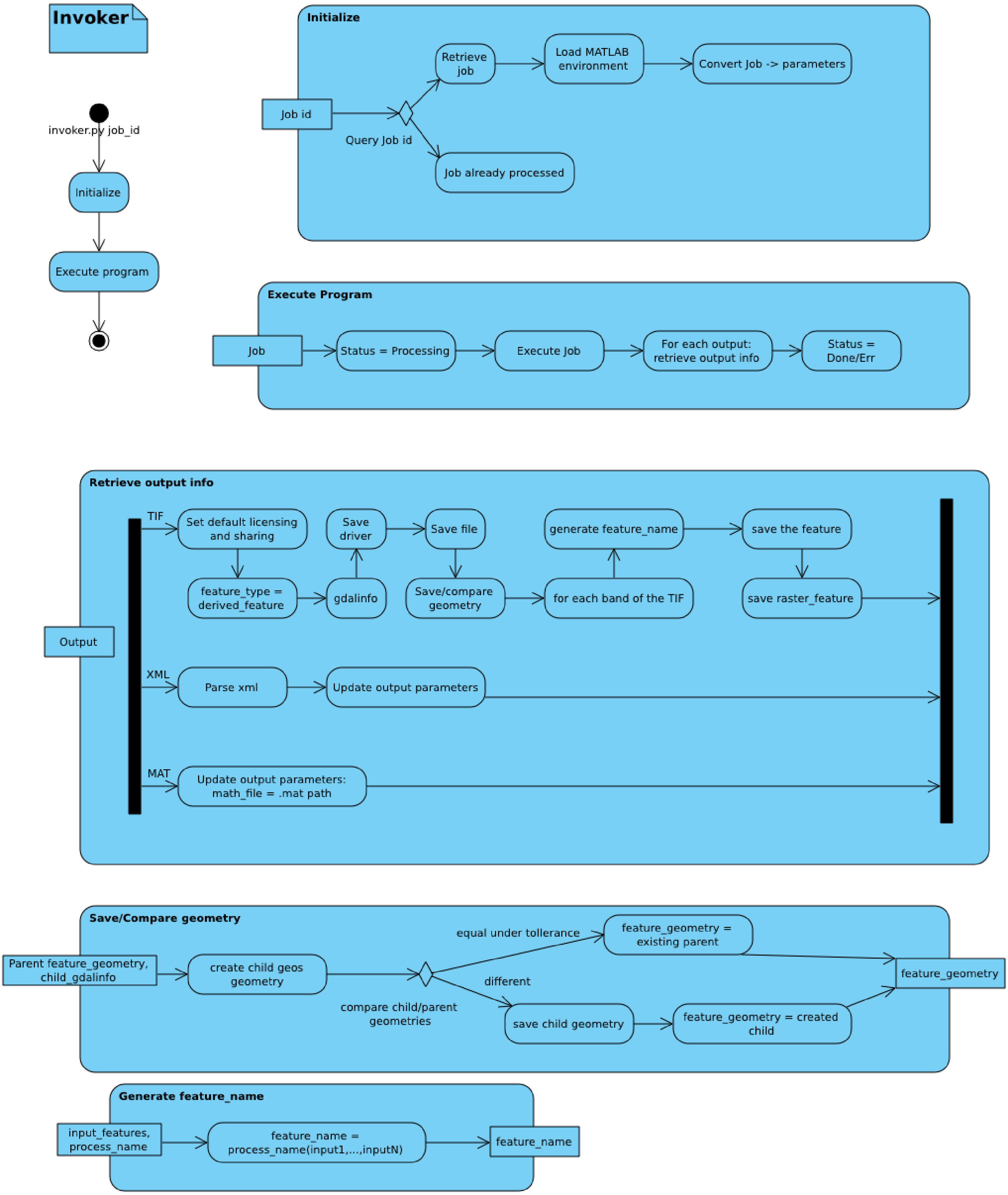
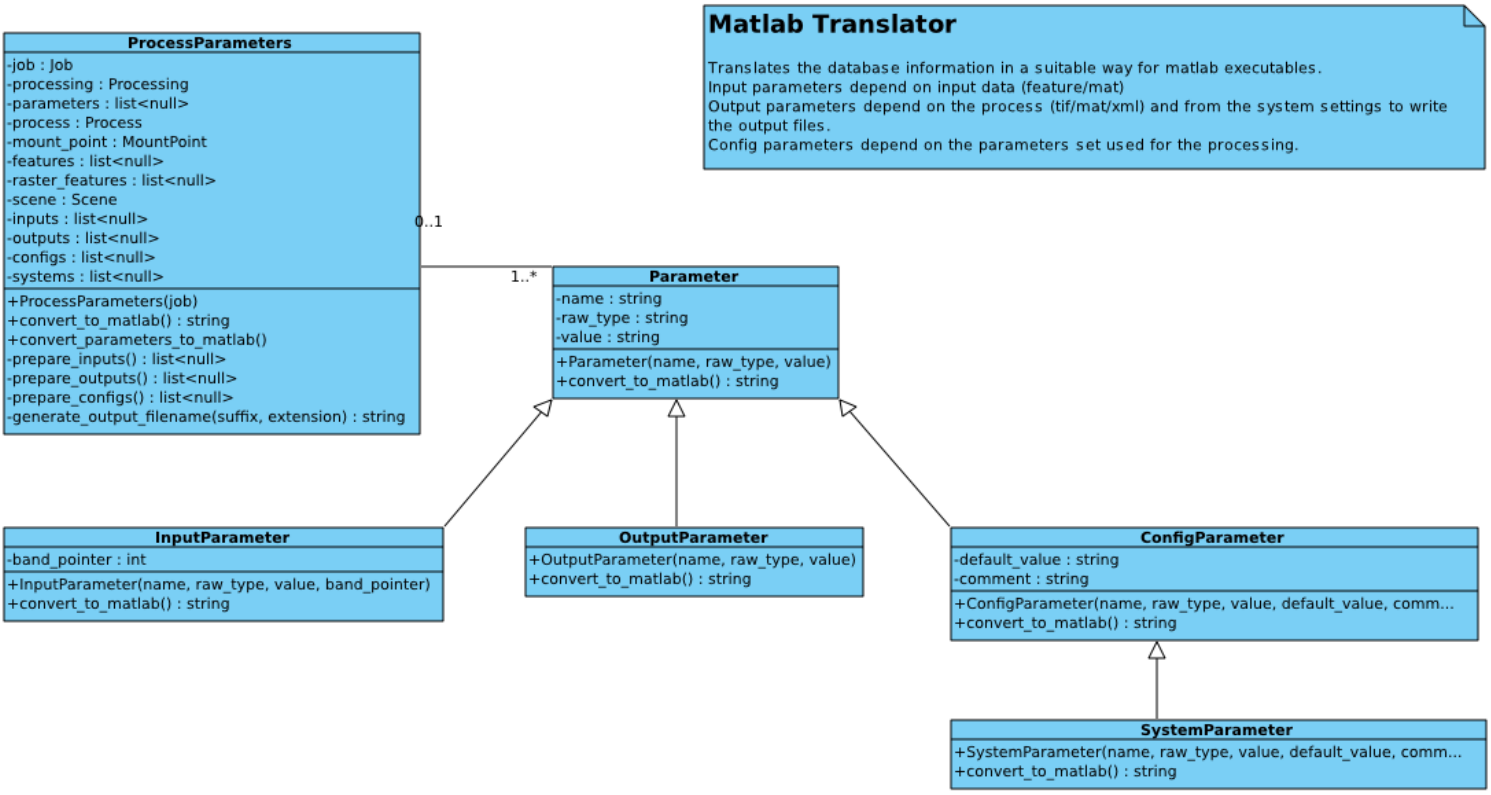


Figure 7: Invoker activity diagram


ProcessParameters
-job : Job
-processing : Processing
-parameters : list<null>
-process : Process
-mount_point : MountPoint
-features : list<null>
-raster_features : list<null>
-scene : Scene
-inputs : list<null>
-outputs : list<null>
-configs : list<null>
-systems : list<null>
+ProcessParameters(job)
+convert_to_matlab() : string
+convert_parameters_to_matlab()
-prepare_inputs() : list<null>
-prepare_outputs() : list<null>
-prepare_configs() : list<null>
-generate_output_filename(suffix, extension) : string
0..1
1..*
Parameter
-name : string
-raw_type : string
-value : string
+Parameter(name, raw_type, value)
+convert_to_matlab() : string
InputParameter
-band_pointer : int
+InputParameter(name, raw_type, value, band_pointer)
+convert_to_matlab() : string
OutputParameter
+OutputParameter(name, raw_type, value)
+convert_to_matlab() : string
ConfigParameter
-default_value : string
-comment : string
+ConfigParameter(name, raw_type, value, default_value, comm...
+convert_to_matlab() : string
SystemParameter
+SystemParameter(name, raw_type, value, default_value, comm...
+convert_to_matlab() : string
Matlab Translator

Translates the database information in a suitable way for matlab executables.
Input parameters depend on input data (feature/mat)
Output parameters depend on the process (tif/mat/xml) and from the system settings to write the output files.
Config parameters depend on the parameters set used for the processing.

Figure 8: Matlab translator class diagram

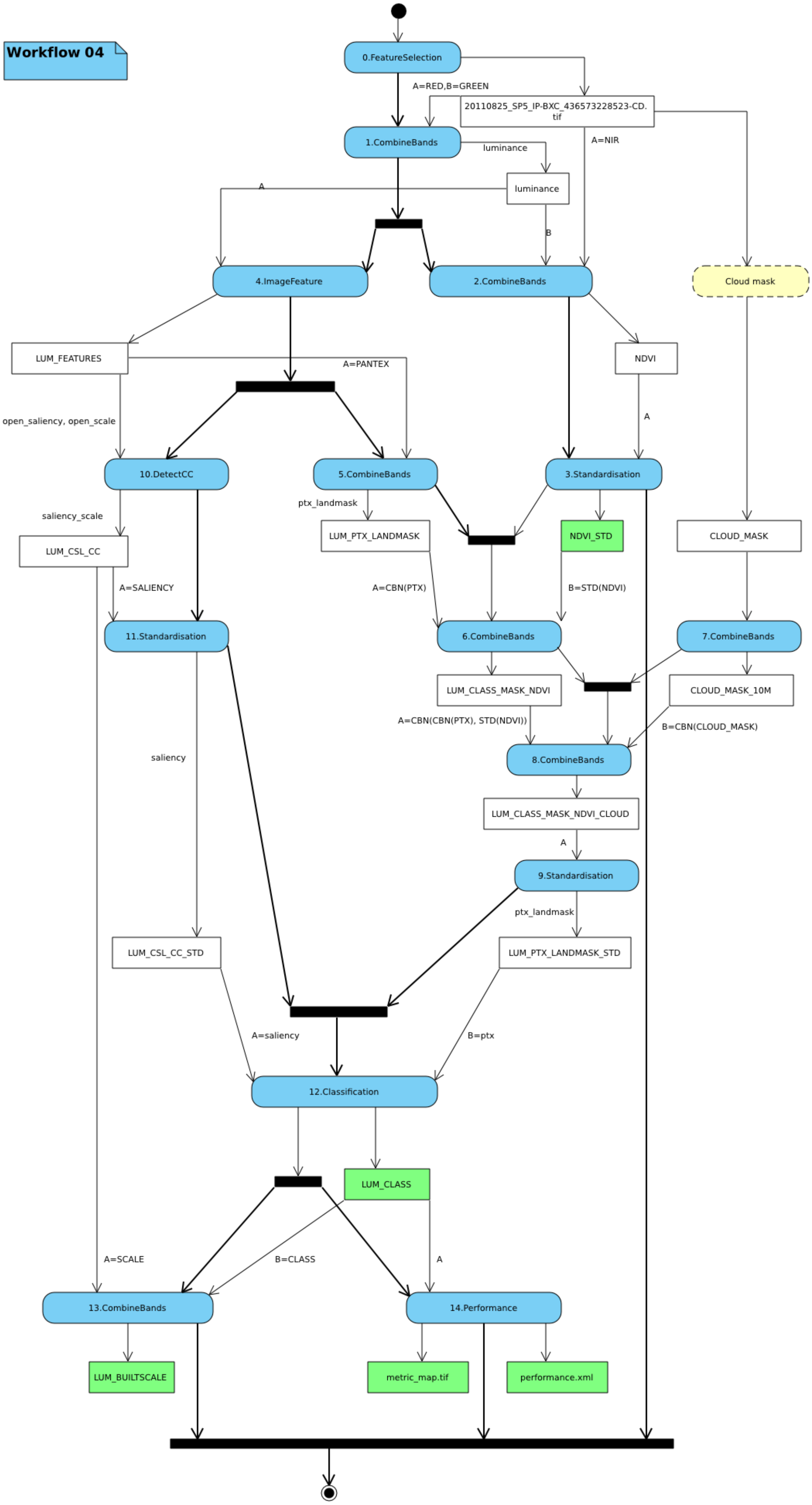


Figure 9: Workflow 4

# Enhancements

The new database has been designed and implemented in order to improve the previous version. The major changes are:

- The switch from an image-based system towards a feature-based system. The benefits of this change are:
    - better handling of satellite imagery for enhanced processings and complex processing chains
    - the possibility of standardizing different metadata coming from various providers
- Improved image metadata accuracy compared to the CID Portal one. The CID Portal contains inaccurate image metadata. To solve this problem, a dedicated importing process has been created in order to automatically verify the accuracy of certain image characteristics like resolution, size and number of bands.
- Workflows implementation. Now it is possible to chain together multiple image processings.
- Added support for MAT and XML files of derived images.
- Improved parameters handling. The IQ system is able to execute processes with various combinations of parameters (user defined).
- Improved importing process. The new process is able to detect missing and replaced images from the CID image catalog.

# Code structure

The code of the database and the related software layer reside in the JRC subversion repository IQ, under the project directory /system/iq2. The latest development code resides in the sub-directory trunk, while the stable versions are saved under the sub-directory tags.

The code of the project is divided in 3 modules: data, process and importer; every module contains at least:

- a models.py file with the definition of the objects;
- a tests.py file with the definition of the unit tests;
- a fixtures sub-directory with the definitions of initial data that should be provided to the model;
- an admin.py file with the definition of the administration web interface to insert/update/delete the data;
- a views.py file with the definition of the views on the data.

The sub-directory iq2 contains the settings of the project, for example the database access credentials for the development and production environments, the Apache web server configuration, the pathname of the static files directories, the hostname of the cluster workstations, the CID access key, and the logging details.

Around 7500 lines of Python code were written for this project.

## Technologies used

The IQ database was implemented using PostgreSQL with PostGIS spatial extension.
Instead, the management software, the importing and monitoring processes were written in Python programming language, with the facilities provided by Django web framework.

European Commission
EUR 26524 EN– Joint Research Centre – Institute for the Protection and the Security of the Citizen

Title: A new spatial database and software layer supporting the JRC image information query (JRC) system

Authors: Marco Scavazzon, Martino Pesaresi

Luxembourg: Publications Office of the European Union

2013 – 16 pp. – 21.0 x 29.7 cm

EUR – Scientific and Technical Research series – ISSN 1831-9424 (online)

ISBN 978-92-79-35530-1 (pdf)

doi: 10.2788/78199

Abstract

The database is one of the core components of the image information query (IQ) system: all the information involved in the system is stored inside the database.
The database is a searchable archive for the data, the processes, and the satellite imagery processing.
With the new database structure it is possible to keep track of every image processing executed inside the system, of all parameters involved to obtain a certain result, and of every input and output involved in the processing.
Moreover, the database contains a replica of the metadata of the CID Portal's satellite imagery, with additional information useful for the team's image processing needs.
The database structure is automatically generated and supported by a Python software layer which is also starting the processings on the Linux cluster.

LB-NA-26524-EN-N

As the Commission's in-house science service, the Joint Research Centre's mission is to provide EU policies with independent, evidence-based scientific and technical support throughout the whole policy cycle.

Working in close cooperation with policy Directorates-General, the JRC addresses key societal challenges while stimulating innovation through developing new standards, methods and tools, and sharing and transferring its know-how to the Member States and international community.

Key policy areas include: environment and climate change; energy and transport; agriculture and food security; health and consumer protection; information society and digital agenda; safety and security including nuclear; all supported through a cross-cutting and multi-disciplinary approach.

Publications Office

doi: 10.2788/78199

ISBN 978-92-79-35530-1