Errata system for CMIP6

Authors: Guillaume Levavasseur, Sébastien Denvil, Atef Ben Nasser

Executive Summary

Our goal is to define and to establish a stable and coordinated CMIP6 procedure to collect and give access to errata information related to datasets hosted by ESGF. The system now in place is only dependant on the PID handle server that is also part of the CMIP6 services ecosystem. And is otherwise self-reliant except for security issues that have been delegated to a third party (GitHub OAuth protocol).

This paper summarizes key requirements as well as necessary steps leading to a better errata system for CMIP6. An Errata Service for CMIP6 is motivated by various use cases:

- **Provide timely information about newly discovered issues.**
- **Provide known issues informations prior to download**.
- **End users querying the service can determine whether modifications and/or corrections have been applied to data they have used.**

You will below requirements summary for ESGF, CDNOT, modelling groups and data users.

ESGF infrastructure:

Requirement: Errata web-service

Priority: high (should be in production state when on CMIP6 publications kick-off)

- The ESGF errata is based on a reliable and available issue tracking system that centralizes issue management.
- The server part (backend) is already deployed on a 3rd party infrastructure (branded by ES-DOC).
- The Errata system relies on the PID Handle Server for easy access to version related information. The link between version and issue is supported by the PID Handle Service on dataset aggregation level.
- The Errata system delivers information about available data versions and associated issues (via ES-DOC portal and search API).
- The Errata system also finds information about no longer available data versions and why (via ES-DOC portal and search API).
- The service delegates authentication and authorization steps to github's Oauth 2.0 service.
- Full API documentation for various interested actors in integrating errata information.

Requirement: Errata client

Priority: high (should be in production state on CMIP6 publications kick-off)

- An errata needs scientific information that is to be manually provided.
- It should be pip installable.
- Issue actions go through a well defined, formatted and structured templates to ensure quality of information.
- Fully documented steps ensuring optimal use of the Errata-Client.

Requirement: Issue status notification

Priority: medium

- Collaboration and solution from the Tracking and Notification Working Team.
- Add issue information to data user requirements from the "CMIP6 Replication and Versioning" WIP paper.
- Support links/API with other services to return issue status (e.g., CoG, synda, Data Citation).

Requirement: Documentation and good practice policy

Priority: medium

- Concise "CMIP6 Errata best practices" document giving clear and concise instructions to CMIP6 publishing sites. This could be part of an overall "CMIP6 Data Management" document as proposed in the "CMIP6 Replication and Versioning" WIP paper.

CDNOT:

Requirement: Enforce and oversee good practices in publication workflow

Priority: high (should be accepted on CMIP6 publications kick-off)

- Consistent versioning consistent with the "CMIP6 Replication and Versioning" WIP paper requirements.
- Publication process start only after issue registration (except for initial version publication).
- No version change can be published/unpublished without creating an issue regarding the change.
- Detect publication with no issue registration.
- Contact data centers with instructions to resolve inconsistencies.
- Decide on actions to take if inconsistencies persist based on agreed-upon measures.

Modelling Center:

Requirement: Roles and tasks

Priority: medium

- Identify one person responsible for issue management (i.e., contact for reporting, validate/register/create/update the issues),
- Identify members in charge of managing issues per institute.
- Identify one (or many) administrator(s) that supervise the issues, per institute.
- Identify one person responsible for linking the new version with the issue(s) (i.e., in charge of the publication)
- Organize training and proper documentation for the actors -mentioned above- including proper configuration of related accounts.

Data users:

Requirement: Good practice

Priority: medium

- Report all encountered errors to the appropriate data provider,
- Use of file/dataset identifiers or the dataset handle pid or the combination of file/dataset identifier with the version in hands, to directly query the CMIP6 errata. This can also be performed using lists within text files to ease larger queries.
- Use ES-DOC errata front-end for interactions with the errata system.

Motivation and Scope

In the context of overseeing the quality of data, the ESGF Errata service has been encapsulated in the ES-DOC structure and built on top of the Handler service that will be deployed in the next release cycle. Consuming PIDs from Handler Service, the ESGF Errata service is guided by a specifically built algorithm that extracts metadata regarding the issues that may or may not affect the quality of datasets/files and trigger the publication of newer versions.
This new structure has been designed with end-user usability in mind, especially those specialized in the publishing process and scientists who consume data and require feedback on the quality and reliability of it.
The outcome to be expected from the Errata service project is to significantly increase the quality and reliability of data. Providing this critical information for end-users requires the application of a well-defined process, the respect of the indicated guidelines and is ensured by exploring incoming features of the ESGF ecosystem.

A wiki page has been made available for CMIP5 to record errata information. Even though this approach was relatively informative it is far from being as user friendly and useful as it should be. End-users had to read the whole content of the page to get the required information of whether a dataset is affected by an issue or not:

http://cmip-pcmdi.llnl.gov/cmip5/errata/cmip5errata.html.

Other scattered errata pages are more focused on specific models or institutions (e.g., http://www.cnrm.meteo.fr/cmip5/spip.php?article24 or https://wiki.csiro.au/display/ACCESS/Known+Issues+with+ACCESS+CMIP5+data).

This paper summarizes key requirements as well as necessary steps leading to a better errata system for CMIP6. An Errata Service for CMIP6 is motivated by various use cases:

- **Provide timely information about newly discovered issues.** CMIP6 experiment design and data specifications are complex. Consequently, it is not uncommon for errors to occur either during the preparation of data or during the publication process, even with vigilant quality controls operating.
  Since errors cannot be entirely eliminated, a github-like public interface should be set in place for data providers, enabling them to directly describe issues when they are discovered and reported by anyone from ESGF community.
  A relation/connection of the issue to a selected group of files is provided, using the pid handle server structure, that indexes issues on the dataset level. However, an algorithm has been set in place within the Errata server that enables users to find information about files as well in a relatively reliable fashion.
- **Provide known issues informations prior to download**. The ESGF search interface could highlight links to the errata pages next to the corresponding file(s) or dataset(s). The user is informed of known issues before downloading. This is however not yet implemented. The PID landing page also shows the errata information regarding the file/dataset in question.

- **End users are able to query the Errata Server to determine whether modifications and/or corrections have been applied to data they have downloaded.** This service relies on unique file identifiers.
- **Develop as part of the errata system a notification center for end users of updates to files of interest to them** (e.g., when an issue has been solved with the publication of a new version). This feature is also yet to be implemented.

Goal

The goal is to define and to establish a stable and coordinated CMIP6 procedure to collect and give access to errata information related to datasets hosted by ESGF. The system now in place is only dependant on the PID handle server. And is otherwise self-reliant except for security issues that have been delegated to a third party (GitHub OAuth protocol).

Current status

IPSL has collaborated with DKRZ to create the Errata service. Using the PID handle service provided by DKRZ, we are now able to reconstruct the genealogy tree of a specific dataset/file and provide the end user by information regarding the state of the information.

The errata service is composed of two parts: client/front-end and server. The client's aim is to enable end users to interact with the server, creating, updating, closing and retrieving issues.

The errata service also features a front-end that also interacts with the server to provide issue viewing features as well as search endpoint using dataset/file ids or pids.

Effort

Although still in alpha phase, the errata service development is nearing the first cycle's end. After which, a community beta should be put in place in order to gather feedback and implement possible suggested improvements.

Community processes

For an errata system to be successful, modeling centers and data centers will be required to apply well-defined processes within the publication workflow both initially and when replacement versions are published. Consequently, a CMIP6 errata system will be intimately tied to the versioning approaches described in the (living) document at:

https://docs.google.com/document/d/1tOaFQEXFyjqAOOlvcdiaX3XrXuxIv_nlE5_FCme1id4/edit.

If a requirement of this kind is to be accepted by the community, benefits must be clearly communicated in advance, and related information must be made readily available together with documentation and training. The following are the minimal requirements providing for enforced actions during the publication workflow:

- Identify one person (whom we will refer to as "data provider") within modeling center that will create/register issue(s), submit the revised data for publication, and notify the issue,
- Identify one person (whom we will refer to as "datanode manager") within data center that will link new version(s) with issue(s) during the publishing process,
- Plan training for them,
- Provide access to documentation and support on how to properly manage issues.

Implementation

Because any change in a dataset must lead to a new version, the Errata Service only relies on the link between the datasets/files versioning and the issues. The Persistent IDentifier system, developed and deployed by the DKRZ, records any change as a "filiation" or "two-way" links between two dataset versions (including the file PIDs). To remove or retract a dataset version, the corresponding dataset PID and files PIDs should be updated at the Handle Service with appropriate metadata. This ensures traceability of any unpublished data. Consequently, we chose to layer the PID service with the Errata Service. The issue(s) identifier(s) will be part of PID metadata. This ensures optimal errata services, instead of querying the ESGF Solr index that should only represent "what is published now". Figure 1 details all implications step by step.

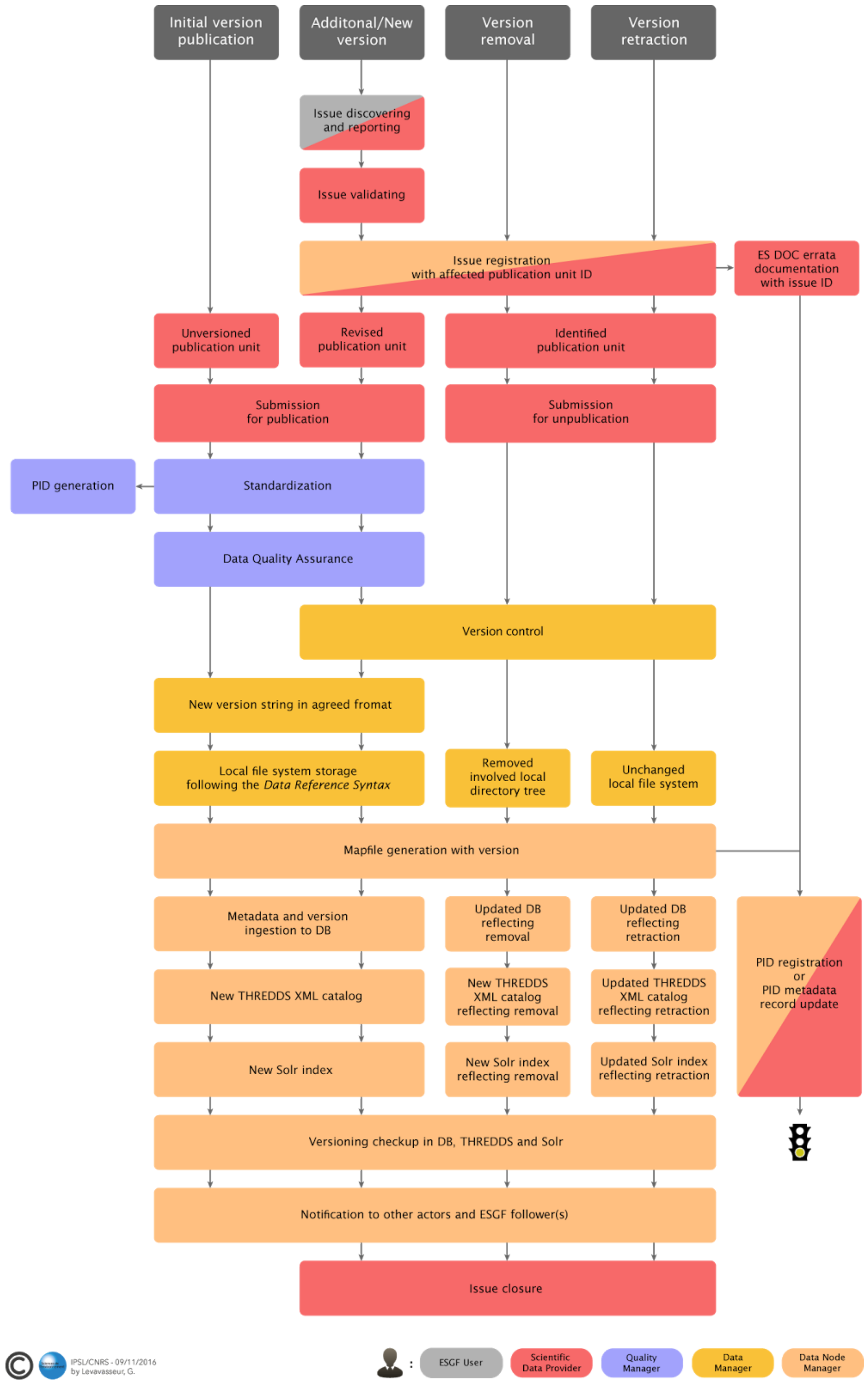
Initial version publication
Additonal/New version
Version removal
Version retraction
Issue discovering and reporting
Issue validating
Issue registration with affected publication unit ID
ES DOC errata documentation with issue ID
Unversioned publication unit
Revised publication unit
Identified publication unit
Submission for publication
Submission for unpublication
PID generation
Standardization
Data Quality Assurance
Version control
New version string in agreed fromat
Local file system storage following the *Data Reference Syntax*
Removed involved local directory tree
Unchanged local file system
Mapfile generation with version
Metadata and version ingestion to DB
Updated DB reflecting removal
Updated DB reflecting retraction
PID registration or PID metadata record update
New THREDDS XML catalog
New THREDDS XML catalog reflecting removal
Updated THREDDS XML catalog reflecting retraction
New Solr index
New Solr index reflecting removal
Updated Solr index reflecting retraction
Versioning checkup in DB, THREDDS and Solr
Notification to other actors and ESGF follower(s)
Issue closure
:
ESGF User
Scientific Data Provider
Quality Manager
Data Manager
Data Node Manager

Figure 1: Relationship between versioning workflow(s) and the errata system.

1. Issue discovery and reporting

Any kind of error or mistake leads to a version change of a dataset and, conversely, any change of a dataset (values, metadata, etc.) is justified by an issue. An error can be detected by any actor of ESGF community. The issue has to be reported to the data provider using one of the following alternatives:

- The email of the appropriate data provider,
- An esg-issue mailing list (e.g., esgf-issues@lists.llnl.gov).

Consequently, all data providers have to be clearly identified by modelling groups and updated on ESGF front-ends and contacts.

2. Issue validation

The publication of a new version of a dataset has to be motivated by an "issue". Consequently, this step is only and always required when updating a dataset (i.e., version update). The data provider evaluates the relevance of the issue by directly investigating the pointed data and checks if this issue hasn't been already reported.

3. Authentication prerequisite

An issue provides a sensitive message about climate model quality. Only identified actors should be able to publish or modify an issue.
The authentication and authorization processes have been delegated to GitHub. The data provider uses her/his GitHub account or creates one to generate a personal access token (PAT). The scope of this token only need to give read access to the organization membership of the user, which minimizes the exposition of user information and encourages people to use their regular github account without fearing for their profiles.
The data provider need to ask to the errata team administrator (i.e., the IPSL) to be part of:

- The **es-doc** GitHub organization,
- An **errata-publication-<institute>** team within the organization depending on his institute/modeling group.

This last authorization level avoid the data provider to publish an issue on behalf of another institute.
Finally, the GitHub username and personal access token will be required (at least) for the first use of the errata client. To avoid redundancy of login, credentials can be saved for recurring use.

4. Issue publishing

The Errata Service provides a unique, structured and centralized registration procedure, ensuring a proper and standardized storage of this metadata. The Errata Client enables end-users to create and update a validated issue by the data provider. Except in the case of the initial version, the issue registration should always be performed prior to the publication process, however it is technically possible to get this done post-publication, this is seriously discouraged. In the case of un-publication of a dataset, an issue should be created providing annotations to the version removal or retraction. Consequently, the errata information of all published versions will be kept for CMIP6 regardless of the actual data state (they could be removed from ESGF by the data provider).

The issue management is delegated to the Errata Server (backend). End-users are able to interact with it through the Errata Client, by creating, updating and closing issues. This is performed through basic JSON issue template.

The template is then validated against the appropriate JSON schema with some safeguards requirements as:

- The issue identifier is automatically generated on issue creation and is assigned to the issue throughout its life cycle.
- Empty attributes are disallowed.
- A precise and concise issue description that must make sense for end users (not just “wrong data”) and changed by no more than 80% ratio.
- The project affected by the issue has to be declared in `esg.ini`.
- The accepted terms for issue severity are:
  - **`low`**: the issue concerns file management (e.g., addition, removal, period extension, etc.),
  - **`medium`**: the issue concerns metadata (NetCDF attributes) without undermining the values of the involved variable,
  - **`high`**: the issue concerns single point variable or axis values,
  - **`critical`**: the issue concerns the variable or axis values undermining the analysis. The use of this data is strongly discouraged.
- All optional URLs must be valid (i.e., accessible).
- A status “`new`” will be affected by the server on the creation. Update an issue cannot change back the status to “`new`”. Issue closure only allows “`wontfix`” or “`resolved`” status values.
- The issue life cycle aims to go from “`new`” to “`onhold`” then “`resolved`” or “`wontfix`” status. It is however at the time being possible to go from “`new`” to “`wontfix`” or “`resolved`”
- The creation, update dates are returned and updated server-side.

Figure 2: Safeguards requirements during issue life cycle

Issue attributes	**Creation**	**Update**	**Closure**
`UID`	❌	,🔒	,🔒
`Title`	/🔑	,🔒	,🔒

`Description`	🔑	🔐	🔐
`Project`	🔐	🔒	🔒
`Severity`	🔐	🔐	🔐
`Status`	✗	🔐	🔐
`Landing page URL`	📎	📎	📎
`Materials URLs`	📎	📎	📎
`Creation date`	✗	🔒	🔒
`Update date`	✗	🔒	🔒
`Closure date`	✗	✗	🔒

🔑	: Mandatory
✗	: Not expected
🔒	: Mandatory and unchanged
🔐	: Mandatory and value-controlled
📎	: Optional

In addition note that the GitHub username will be added as the person who create, update or close an issue on the errata front-end.

All attributes changes are reflected into the `JSON` file that was used to create, update or close the issue. This file is automatically enriched with metadata that can be automatically extracted from the issue declaration.

The errata client also requires the list of affected datasets as a `TXT` file. The dataset id has to be properly formatted following the DRS with their version number, the dataset list should conform with the declared project and its related DRS.

If a user wishes to download an issue, the client provides a download feature using the issue UIDs.

5. PID interaction

The errata server will then push the issue identifier to the corresponding PID records on the Handle server. Several issue identifiers for the same PID has to be supported in the sense of a dataset can be affected by several issues.
In case of an update that remove a dataset id from the input list, the errata server will remove the corresponding issue identifier from the PID records.

6. Publication unit(s) submission

The data provider knows the definition of a publication unit (i.e., a dataset) depending on the project and produces raw climate data according to the issue (if it exists):

- Unversioned units for initial version publication,
- Revised or corrected units for version update,
- Or just identifies the units to unpublish.

To provide version(s) update/retraction/removal with registered issue(s), the data provider notifies the datanode manager about the issue(s) identifier(s) (except for initial version publication) involved by the submitted data for publication/un-publication (i.e., they maintain together the “datasets-issues” list).

7. Versioning check up

The datanode manager checks the existing version history of the submitted dataset(s). If no previous version exists, the versioning is initialized. Otherwise the datanode manager ensures this exact collection of files has not already been published under a version of this dataset anywhere in ESGF (i.e., master + replicas) by calling the Handle Service or Solr (PIDs will be stored locally). The same collection of files should not be published with two different versions. In this case, the datanode manager shall notify the data provider about the following possibilities:

- Re-publish the corresponding units by retracting it before. This action should not delete information about the deprecated publication units and its version (a re-published unit is new version with un-publication annotations).
- Revise again the data,
- Set the issue status as “`wontfix`”.

To remove or retract a version of a dataset, this step consists only in identifying the involved dataset version.

8. Issue checking

If the data provider reported a reason for publication or in the case of un-publication, the datanode manager checks whether the corresponding issue(s) has (have) been registered or not. If not, the datanode manager shall notify the data provider to create the issue(s).

9. PID registration

For each dataset to publish, the ESG-F publisher generates a dataset PID and kicks-off the PID registration.

The dataset PID and the files PIDs attached by CMOR are sent with metadata including an issue(s) ID(s) field. To remove or retract a dataset version, the corresponding dataset PID and files PIDs are updated at the Handle Service with appropriate metadata. This ensures to keep track of any unpublished data.

10. Notifications

Automatic notification feature can be implemented in future versions of the Errata Service. For the time being, at the end of the publication process, the datanode manager should manually notify all above actors that dataset has been published and the interested end users of the new version, the version retraction or removal.

11. Issue closure

When the new versioning has been checked, the issue(s) is resolved and should then be flagged as "`wontfix`" or "`resolved`", marking the end of the issue life-cycle. Figure 2 shows the template requirement for issue closure. On success the close date are returned by the server.

Errata Web Services

The Errata Service relies on a set of Web Services to ensure interaction between server and the different components of the errata system. These web services can be extended to allow CoG front-end or Synda integrations, and quite possibly a mature and well documented API for everyone to make use of.
Following the proof-of-concept developed at IPSL (http://errata.ipsl.upmc.fr/) the errata module could host two services:

- **A front-end listing of issues**: This first sub-module automatically describes each registered issue giving the issue identifier, a short description, the URL of the landing page, the issue status, the error severity, all affected files and the associated version(s).

  A filtering system is provided to select issues by severity, institute, project, variable, experiment or model. The front-end obviously interacts with the same database of the issue manager hosting all the issues that have been so far declared.

- **Querying the Issue Manager Database** is also possible via the search interface. The user input can be either a set of dataset/file IDs or PIDs, or a file of any of the former. In the Errata Service Backend, a specifically designed algorithm will run resolution of all errata information and will extract each dataset/file errata history, in timely fashion.

  A PDF export of the result could be a useful feature. The figures 2 describes the Errata Service and the interactions between an Errata Service and the Handle Service

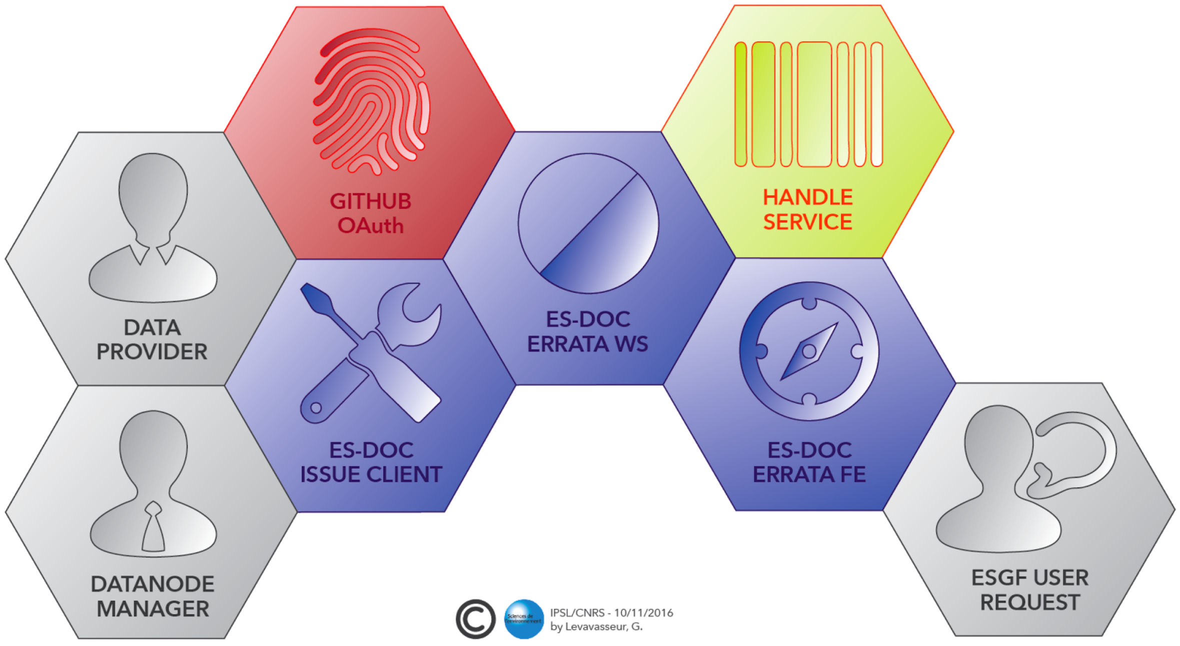

Figure 3: Errata web-service implementation

Contacts

- Developers:
  - Guillaume Levavasseur (UPMC/IPSL - glipsl@ipsl.jussieu.fr)
  - Atef Bennasser (UPMC/IPSL - abennasser@ipsl.jussieu.fr)
  - Mark Greenslade (UPMC/IPSL - momipsl@ipsl.jussieu.fr)
- Contributors:
  - Sébastien Denvil (CNRS/IPSL)
  - Merret Buurman (DKRZ)
  - Katharina Berger (DKRZ)
- Acknowledgements to:
  - Martina Stockhause (DKRZ)
  - Tobias Weigel (DKRZ)
  - Ag Stephens (CEDA)
  - Alan Iwi (CEDA)