Persistent Identifiers for CMIP6: Implementation plan

Authors: Tobias Weigel, Michael Lautenschlager, Martin Juckes

Final author approved version.

Executive Summary

The goal of this effort is to establish a hierarchically organized collection of persistent identifiers (PIDs) for CMIP6 data. Using such identifiers, smart tools will, for example, indicate whether a given file in a user's working directory is still the latest version and if not, where a more recent version of the corresponding dataset can be found. The identifiers names can also be used to communicate precisely about slices of the CMIP6 data space, for example by including them in reference tables in scientific articles. At higher levels of the hierarchy, the PIDs will refer to collections of an evolving set of related files (e.g., all files from a single model simulation or all files generated by a single model). Such PIDs might be used in providing credit for a modeling group in scientific articles when recording individual file PIDs is inappropriate or impractical. For every file and dataset, a web page will provide a quick overview about the most important details, including links to direct data access and prior or more recent versions. The necessary services will be operated in a way so they do not obstruct the timely publication of files once they have passed the pre-publication quality checks, even if software components are temporarily out of order.

In terms of **implementation**, the system builds on the Handle Registry software which provides for a robust and resilient mirrored service. This software is deployed in larger infrastructures, such as the Digital Object Identifier (DOI) service, so it can be relied on to deal with the scale of traffic associated with ESGF, provided that the implementation does not introduce unnecessary bottlenecks. While resolution of identifiers can be shared across multiple mirror sites, registration can only be done through a single server. As ESGF must support publication from multiple sites, particular attention must be paid to the design of the registration workflow to avoid introducing delays.

Software development requirements:

- CMOR must be modified so it can write an extended tracking_ID syntax. This must be configurable since not all ESGF projects will use Handles. To properly manage this, it is recommended to generally establish an additional global CMOR configuration file that bears options valid for all tables.
- The workflow and options of the ESGF publisher must be consolidated to reduce the variety of PID actions and decrease the likelihood of mistakes made by the automated services.
- The software services related to PIDs must be further developed and tested, including changes to the publisher.
- The software controlling the last quality checkpoint before files enter the ESGF publication workflow must be extended so it can verify for every file that the tracking_ID syntax is a PID and that early citation information is complete. The concrete items for citation information are described in the citation paper. Files not meeting these criteria should be rejected.

Requirements for the modelling groups:

- The updated version of CMOR capable of writing the new tracking_ID syntax should be used by all CMIP6 groups.

- The prefix “21.14100” must be included in all CMOR tables through a dedicated option. All groups should use the CMOR tables with the option activated.
- Modelling groups remain responsible for any modifications of their files, including corrections of the PID-conformant tracking_ID field if errors are detected at quality control checkpoint D2.

Operational requirements:

- CDNOT should make sure that the D2 checkpoint verifies the correctness of the PID-style tracking_ID at every node and that files not fulfilling the correct syntax are rejected before they proceed to the technical publishing process.
- Additional services must be maintained at a dedicated ESGF center, possibly relying on an external academic cloud hosting service to provide redundancy and improve reliability. While the WIP will not be involved in the particular choice of technical solutions, it should oversee which partners are responsible for running such services.
- A state indication for every file should be established and made visible. The final and desired state includes that each file is indexed, accessible, bears a PID and early citation information. To establish this, additional services may be required that are able to collect this information at a central place and optionally at a distributed Solr index for faster access.
- Machines where the ESGF publisher is executed must be able to connect to the Internet. If this is not possible in a direct way, a local RabbitMQ proxy must be set up on a local machine.

1. Overview

The goal of this effort is to create an ESGF-wide global Handle Registry for CMIP6 data. The Handle Registry system is designed to resolve identifiers enriched with a small amount of metadata. It is also designed to support high availability and long term persistence, but not to handle complex metadata structures.

The implementation proposed has the objective to provide persistent identifiers for every file in CMIP6 and several aggregation levels. The persistence of the identifiers will facilitate record keeping and exchange of information about data used, both within the publication workflow and in client applications.

A key constraint here is the need for integration into the distributed ESGF publication workflow. The major challenges are in ensuring prompt updates of handles when the publication or version status of files changes. This will be automated: the registry is fed by and integrated with the CMIP6 publication workflow, not posing additional effort on the data producers other than providing CMOR conformant output with a dedicated tracking ID format. The registry is usually not used directly by end-users; special information pages and smart tools will be provided that pull information from the registry and other sources within ESGF.

File-level information to be stored in the registry includes location, checksum, DRS name, creation date, status flags and parent objects. For datasets it will also include information on prior and more recent versions. The registry will also include flags that tell whether a data object has been withdrawn.

The first phase of implementation, described here, aims to give added value to the archive with minimal interruption to the existing workflows.

2. Tracking-ID setup and implications for the modelling groups

Modelling groups are requested to use an updated version of CMOR, which – among other things – contains a change in the way the tracking_ID is written. The new syntax of the tracking_ID (specific to CMIP6) is "hdl:21.14100/<uuid>"[1]; no additional field 'PID' will be written. Whether the new syntax is used depends on whether or not a prefix is configured via the CMOR tables. It must be possible to deactivate this because projects other than CMIP6 use CMOR as well and may either use the traditional syntax or Handles but with different prefix. Thus, the best option is to include an option to configure the prefix in the CMOR tables: If a prefix is given, it is used if the project_id is CMIP6; if no prefix is given, the tracking_ID format will be the traditional UUID (type 4, random) without any suffixes; if a prefix is given but the project_id is not CMIP6, an error is raised (this case may happen if someone duplicates the tables to use for a different project). Things could be simplified if there is a general CMOR table with global options in addition to the individual tables. If there is no general CMOR table, the prefix option must be included in every table for CMIP6.

Getting the new tracking_ID syntax written into all CMIP6 files is an absolute prerequisite for any other parts of the implementation plan or later value-added services. Doing this via CMOR is the easiest route as no additional tool must be delivered to the groups and executed by them. Using a single prefix instead of several (that roughly represent replication centers or similar) has the advantage of keeping CMOR configuration simple. We judge that this outweighs the latent disadvantage of a performance and service bottleneck through the single prefix; in addition, there is a mitigation mechanism with asynchronous registration and caching explained further below.

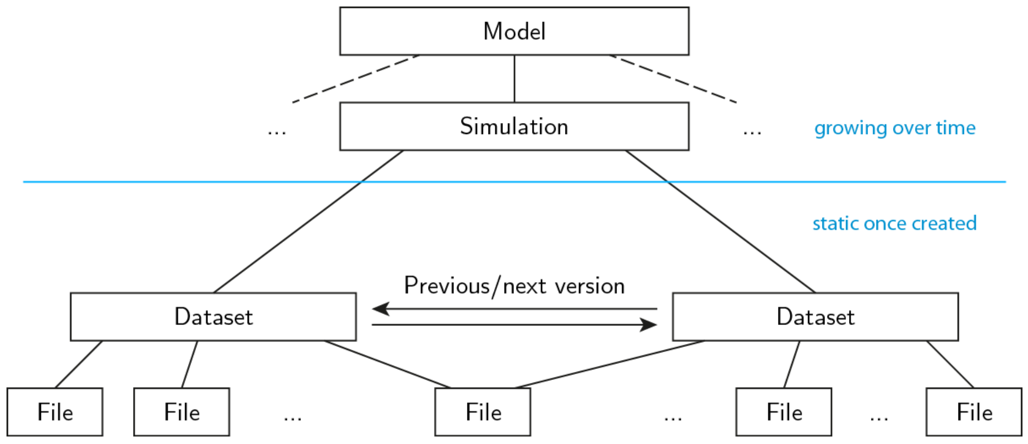
The general policy on file changes should be the same as in CMIP5: Only data providers change files; if errors are discovered in the publishing process, files are sent back to the provider for correction. In accordance with the Quality Assurance White Paper, the checkpoint D2 must confirm that the PID syntax is valid (the PID can potentially be registered, the prefix is correct). Checkpoint D2 is a quality control point, verifying the actual data contents at the data center's site. In the context here, it is the last check before files enter ESGF publication and therefore the last point where they can be sent back to the provider to request changes. The D2 checks should also confirm that the early citation entries on the corresponding simulation and model are complete. A solution where the tracking-ID syntax is changed after a file has left the control of the data provider is rejected. To help modelling groups with fixing files, a stand-alone tool that can process netCDF files and extend existing tracking_ID fields with the new syntax can be easily provided.

After registration of the tracking_ID as a Handle, the full Handle name can be verified by simply resolving it, significantly reducing the need for an additional check digit. It is unclear whether it is required to have a separate check digit that can help to detect human errors in the limited timeframe between CMOR output and Handle registration. Integrating a check digit outside the UUID syntax in the tracking_ID would incur further changes to the CMOR source code, which should be avoided.

3. Publishing process and PID hierarchy

[1] "21.14100" is a Handle prefix that is administrated by DKRZ as part of a contract with GWDG, a newly established Multi-Primary Administrator within the framework of DONA. DONA (Digital Object Numbering Authority) is a Swiss foundation which has taken over administrative control of the Handle System from CNRI. DKRZ manages the future prefix namespace of 21.14xxx.

In addition to file Handles, the PID component will register Handles for further aggregation levels. It is clear that the atomic dataset level as the lowermost aggregation is required; early citation depends on the higher levels of models and simulations. Other levels may be useful in the future. The figure summarizes the currently proposed levels. Linking between versions should happen consistently at



the dataset level.

Figure 1: The PID hierarchy consists of 4 levels, with files at the bottom and 3 aggregation layers on top. Files can have more than one parent. Model and simulation collections grow over time while keeping the same PID.

For aggregation identifiers to work, file identifiers have to be established first. Once they exist, aggregation identifiers are technically easier to achieve compared to the effort involved in getting the file identifiers in place. Model and simulation aggregation PIDs will point to a landing page service that resembles the layout of the established CMIP5 DOI landing pages. This is described in more detail in section 6.

The current concept and prototype implementation for aggregations (collections) supports organizational concepts other than straight hierarchies: files or other aggregations can be part of more than one parent aggregation [1]. Aggregations can also be related to each other with roles other than child-parent (e.g. prior/next version).

The exemplary use of aggregations in ESGF for CMIP6 is also a good candidate use case for the upcoming working group of the Research Data Alliance (RDA) on “PID Collections”[2] (aggregations and collections meaning the same thing). The ESGF example will be presented and examined along others from various disciplines, and the working group aims to make a consolidation effort possibly leading to a coherent framework for aggregation management. The insights gained from such a wide cross-disciplinary approach may be beneficial to ESGF in the long term; however, there are no concrete plans and actions possible yet, thus this remains an option for future development. The requirements from ESGF take precedence over any requirements proposed by this working group when it comes to developing the ESGF services; working with the RDA group should be seen as a bonus and giving visibility, but not as mandatory.

[2] Proposed co-chairs of the RDA working group: Bridget Almas, Frederik Baumgardt, Tobias Weigel, Tom Zastrow.

The model and simulation PIDs will be registered when their first files are processed by the ESGF publisher. However, their PID names will be known before that since they bear a semantic syntax based on DRS elements. This syntax and all PID names must be defined after the citation information has been collected from the modelling groups; this is however no concern of the PID services, but of the citation services.

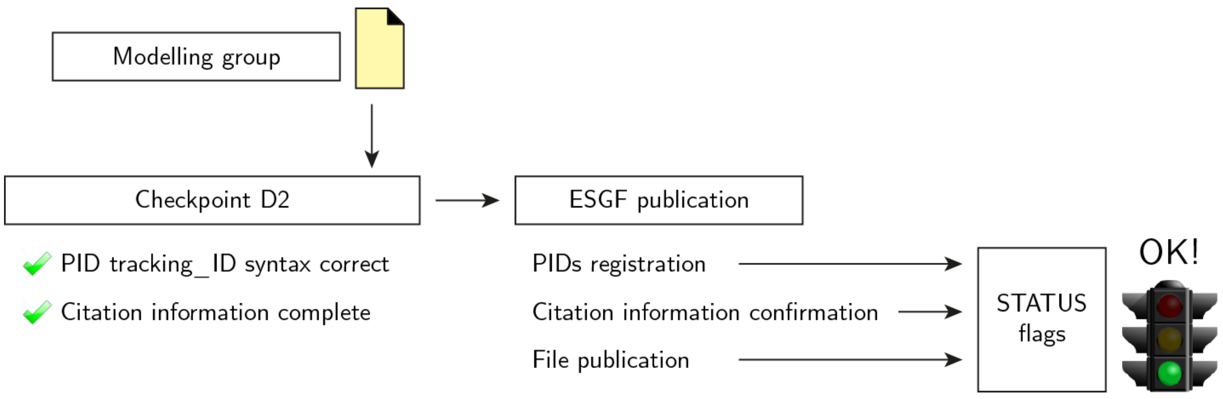

Modelling group
Checkpoint D2
ESGF publication
PID tracking_ID syntax correct
Citation information complete
PIDs registration
Citation information confirmation
File publication
STATUS flags
OK!

Figure 2: The general publishing workflow with checks relevant to PIDs and citation. At checkpoint D2, files can still be rejected; rejected files should not enter the publication workflow. Individual actions at ESGF publication may fail.

The citation check and the final step of the publisher should confirm their corresponding flags by pushing a message to the RabbitMQ[3] queue. The main reason for using RabbitMQ is that multiple publishers need to make actions to a single PID service, which is a potential scalability issue that RabbitMQ is used to alleviate. The RabbitMQ consumer can update the Handle records, which should usually exist as the first iteration of the consumer that creates the initial file/dataset/collection Handles should have already been completed; otherwise, the request will be delayed until the records are created. Each such status attribute is a simple Boolean flag.

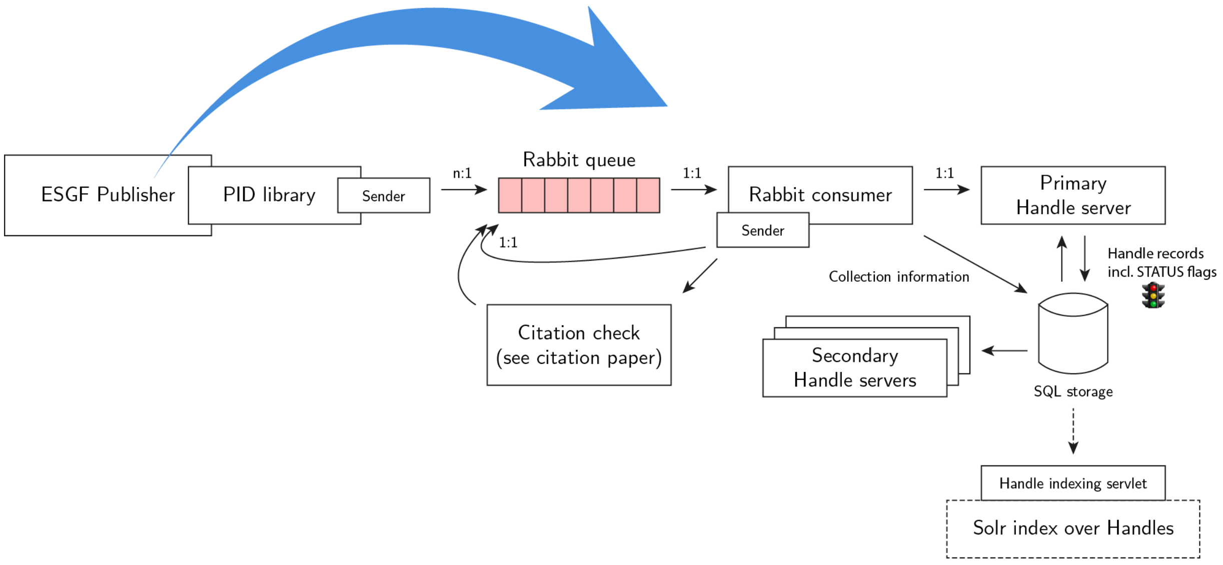


Figure 3: showing how the publisher triggers PID registration through an asynchronous, non-blocking queue.

[3] RabbitMQ (http://www.rabbitmq.com) is a high-throughput, highly reliable message broker, originally designed by the telecommunications and financial industry for real-time microtransactions. The system is designed to ensure that asynchronously sent messages are not lost due to node failures. IPSL uses it locally to control their simulation workflows.

4. General operational architecture

The ESGF publisher has the central role to trigger actions that lead to registration of Handles and several kinds of Handle record modifications. The general design as shown in figure 3 is that the publisher uses a Python PID library to populate a RabbitMQ messaging queue. All publisher instances at all nodes submit actions to the same queue. All calls the publisher makes to the Python library are in non-blocking mode, thus the general publishing process is not delayed by the potentially huge number of Handle actions. At the consumer end of the queue, a central component processes the action messages and causes changes in the Handle dataspace. These changes may therefore occur significantly after the original publisher process has finished (up to several hours later in case of interruptions or high load).

PID resolution and target URLs

File and dataset PIDs will resolve to a service that displays their most essential properties and points to PIDs further up in the hierarchy and prior/later versions. It should also provide links into the ESGF search catalog and to pages that display the object status (traffic light); also see section 6 for more details. It will also be possible to get from a file or dataset information page to the corresponding model or simulation page.

Model and simulation PIDs will resolve to a landing page service that is based on the existing CMIP5 DataCite DOI landing pages. These pages serve two user scenarios: They provide data context including citation information and enable precise data access. The data access facilities rely on the PID aggregation hierarchy to provide a browsing interface. An additional option that points to the ESGF catalog will also be included. Further details are described in section 6.

Publisher and library

The necessary modifications to the ESGF publisher are very lightweight so not to overload an already complex component with more clutter. They also do not replace existing mechanisms but are built in parallel; for example, traditional population of the ESGF Solr index remains untouched. All business logic is hidden in the Python library whose main interface methods are designed to match the publishing workflow. A node may also decide that it wants to call the library separately from the publisher through a dedicated tool called after the publishing is done. This is possible, but not recommended, since there will be an increased risk that unregistered files are becoming available through ESGF and because it will be unclear whether publication and registration are really accomplished within a very short time frame.

Workflows for publishing, versioning and retracting of data should be described in the CMIP6 Data Management Plan (DMP) to make publisher usage more coherent and reduce the variation in circumstances for actions that are effected on the Handle records. The more variety there is, the more implementation effort must be spent (on rare exceptional cases etc.) to maintain the consistencies between related Handle records and between Handles and the other parts of the ESGF information infrastructure.

The library will also dispatch calls to register model and simulation PIDs unless they are already registered. Once this is confirmed, it pushes a notification to a “citation checker” component. This component should verify again that citation information is complete and conform for a specific file’s model and simulation. It should send messages back to the queue to update the STATUS flags. Further details should be described in the citation paper.

The RabbitMQ setup

Each instance of the python library talks to a central RabbitMQ queue. The motivation to use a RabbitMQ queue in front of the Handle server is to provide caching for better performance and failure recovery when the Handle server is broken. The queue can simply be a single central endpoint with the option to have a distributed queue if need arises for performance reasons. This means that the machine where the publisher runs **must have Internet access**. If this is not possible, at least a proxy solution must be used (e.g. via a RabbitMQ instance on another local machine that have Internet outbound access).

The consuming service is intelligent enough to make decisions based on the information given in the publisher's message without having to ask back. Some additional information may be taken from the solr indexes. The RabbitMQ message consumer may also dispatch follow-up actions to the queue, such as additional updates to linked Handle records.

The aggregation levels of models/simulations may not exist when a file matching their scope is published. Therefore, the consumer service will check whether PIDs already exist for these levels. If not, it will generate such PIDs and the necessary collection structure. If they already exist, it will add the dataset to the existing collections. The structure of all collections should be the same so conversions should not be necessary. If they did not exist, the consumer will also notify a specific citation checker component, to be described in the citation paper, so that the corresponding model/simulation citation information can be confirmed.

It is not clear yet how large the largest collections in the CMIP6 space will be. The first option to implement collections is to solely rely on the Handle records and store children in JSON format. Technically, these will end up as BLOBs in the SQL storage. The disadvantage is that unpacking such collections is costly, so an alternative way is to have separate tables to store children information and join them transparently when resolving Handles or manage them completely separate via the RabbitMQ consumer (it is possible to customize the SQL statements the Handle server uses, e.g. to use optimized views). These implementation details cannot be decided until a later point, e.g. after the CMIP6 DRS is settled. The safe option is therefore to plan for both possibilities, but at least confirm the levels where such 'large' collections may exist (dataset, model, simulation). Separate database tables may also be used to manage placeholders for files where no valid or old-style tracking_IDs were provided. Overall, these questions cannot be answered completely at the present time and will be addressed as part of the implementation.

The final action after registering a PID is that the message consumer sets a STATUS flag to mark a file as "pid registered", which is one step in the file achieving the "green state". As all other actions, this happens asynchronously and may therefore happen significantly (hours or, in case of issues, 1-2 days) after the publisher has run.

Handle server setup

While there can be multiple secondary Handle servers responsible for reading Handle information of a particular prefix, there is always only one primary Handle server for each prefix that can create Handles and write records. This is a design limitation of the Handle System. To mitigate this drawback, Handle creation and modification can be sped up significantly by calling the server's interface from the same machine and aggregating many actions into larger bulk requests (at the price of further delaying individual actions). If this is still not sufficient in terms of peak throughput, we can further investigate an option to circumvent the native Handle System interface and write directly into a distributed database (Apache Cassandra or similar).

Finally, there is an option to increase the reliability of the RabbitMQ/Handle conglomerate briefly mentioned at the Hamburg ESGF meeting. Instead of having queue and Handle service hosted at DKRZ, either only the queue or the queue and the Handle server (and its database) can be installed at a highly available cloud hosting service. In either case, the advantage is that publishing should remain possible even if there is a disruption at DKRZ. Hosting only the queue in the cloud will provide this benefit if the queue is configured to become persistent if the Handle server connection is lost. Hosting both queue and Handle server comes at additional costs and administrative effort.

Independent from the particular choice of hosting, secondary Handle servers for faster read access can be set up in a distributed manner. Such servers are part of the Handle System's inbuilt load balancing (mirroring) and can be configured and maintained with minimal setup once the primary is ready. These secondaries are however read-only; write access remains the main bottleneck.

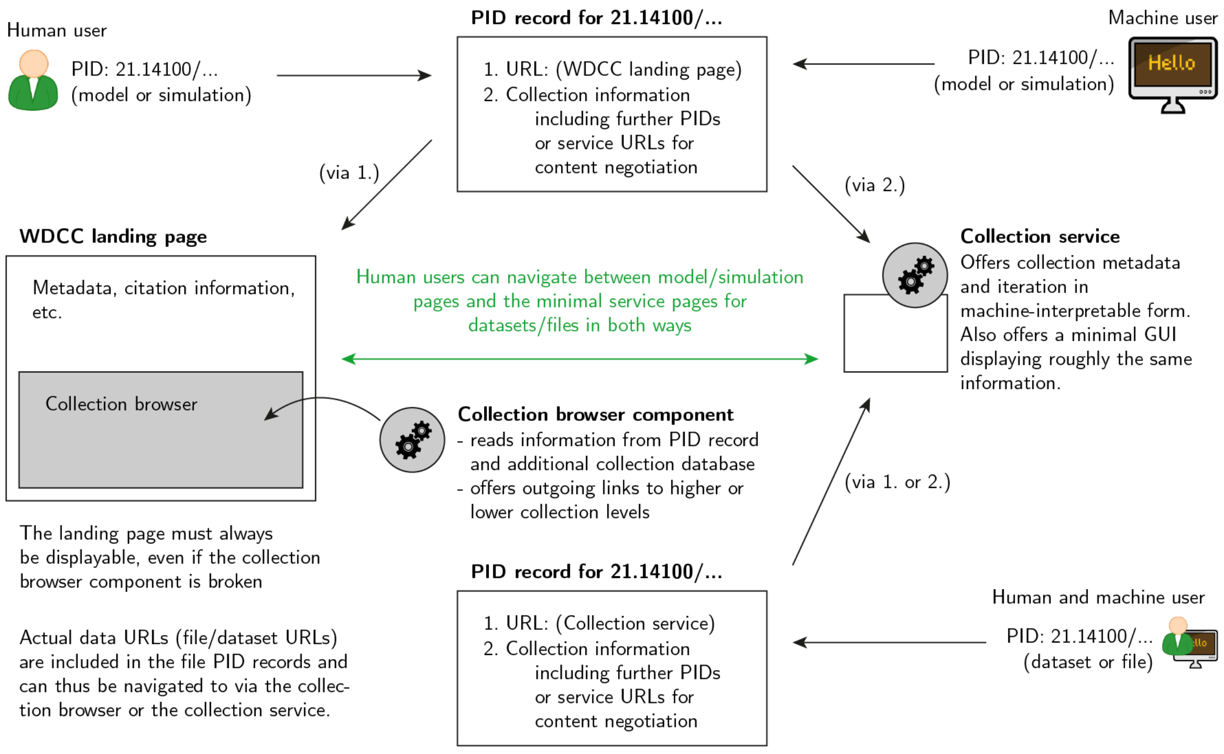
5. Monitoring service and PID curation

Changes in the file location will be propagated to Handle records through events fired off by the publisher, automating the Handle maintenance process as much as possible. In addition, mass management tools will be developed depending on specific needs, for example to manually correct record information across a slice of the CMIP6 PID space. The goal is to minimize the need for such actions, yet tools will have to be available as a fallback solution.

The central Handle server/database installation will also be accompanied by a monitoring service. In the best case, this service regularly goes over all PIDs and checks them for sanity and availability of their target URLs. The service will keep a list of possibly broken PIDs and revisit them after a defined timespan. Encountering an issue should also cause a further automatic investigation of related Handles (up and down the aggregation hierarchy). If URLs continue to be broken, the service will escalate this information, for example to the data node responsible. This general reliability information should be available from the PID information pages as well, at least at the file level. Due to the eventually huge number of Handles, the service may take a lot of time to cycle through all of them, therefore it may limit its activities to random spot checks at some point. The balance of checking across all nodes has to be maintained, however.

6. Data user benefits

Figure 4 illustrates the entry points, pathways and resulting GUIs or web services that human users or machine agents (e.g. smart clients) can access from a given PID. The pathways can lead to metadata and eventually data.


Human user
PID: 21.14100/...
(model or simulation)
PID record for 21.14100/...
1. URL: (WDCC landing page)
2. Collection information including further PIDs or service URLs for content negotiation
Machine user
PID: 21.14100/...
(model or simulation)
Hello
(via 1.)
(via 2.)
WDCC landing page
Metadata, citation information, etc.
Collection browser
Human users can navigate between model/simulation pages and the minimal service pages for datasets/files in both ways
Collection service
Offers collection metadata and iteration in machine-interpretable form. Also offers a minimal GUI displaying roughly the same information.
Collection browser component
- reads information from PID record and additional collection database
- offers outgoing links to higher or lower collection levels
(via 1. or 2.)
The landing page must always be displayable, even if the collection browser component is broken
Actual data URLs (file/dataset URLs) are included in the file PID records and can thus be navigated to via the collection browser or the collection service.
PID record for 21.14100/...
1. URL: (Collection service)
2. Collection information including further PIDs or service URLs for content negotiation
Human and machine user
PID: 21.14100/...
(dataset or file)

In general, the model/simulation level follows different pathways than the file/dataset level because for the latter there are no complex metadata including citation information available.

Model and simulation level

If a machine agent resolves a model or simulation PID, it will be redirected to the collection service which also provides machine information for files and datasets (see below).

If a human user resolves a model or simulation PID, he will be forwarded to a proper landing page, maintained by WDCC, that is similar to those used for CMIP5 DataCite DOIs. This page will include citation information and metadata; details can be found in the citation document.

A specific part of this page should be reserved for a “collection browser” component. This browser enables human users to navigate through the collection tree, also down to the file level, with the corresponding minimal collection metadata available at each level (timestamps, status/tombstone information, which versions are available etc.). The dataset/file information displayed here should also offer links to get to the collection service pages described further below. If this component fails, the pages should still be delivered, thus not having a negative impact on the high availability of landing pages. Details on the scope of this browser component need to be determined during implementation and iteration over some first experience.

Dataset and file level

A demonstrator for the collection service[4] has been built that illustrates the benefits when a human user resolves a file or dataset PID to get to a brief information page. A user interested in resolving a Handle will sometimes have the particular file at hand; at other times, he may encounter it in a personal notebook, an e-mail conversation or similar, missing the context. The information page

[4] Example dataset: https://handle8.dkrz.de/landingpagedemo/10876.test/49634b69-6662-4a52-9175-45f296dc9578

provides such minimal context and also enables discovery of the higher collection levels, which particularly includes getting citation information for the corresponding model or simulation level.

Through content negotiation, resolving file or dataset PIDs will lead to two outcomes:

1. For users using a browser that requests HTML via content negotiation, PIDs will resolve back to the minimal information page that is fed through multiple sources:
    - PID record information. This is the most elemental, contains all URLs for the entity (originals, replica), and will also be available beyond the object's lifetime. Therefore, the page can simultaneously act as a "tombstone service" for files and datasets (particularly for specific versions). This information is maintained by the process shown in figure 3.
        - Files: All original and replica URLs, creation date, dataset level PIDs, checksum, aggregation level marker ("file"), tombstone/status information.
        - Datasets: URL, All child Handles, possible Handles of higher aggregation levels, creation date, prior/next version Handles, aggregation level marker ("dataset"), tombstone/status information.
    - Information pulled from the ESGF solr indexes or similar sources, possibly including the object status (red/green/yellow). Errata information may be included if available through a unique ID in a consumable format.
    - STATUS flags for data accessibility and citation information.
2. For machine agents (wget, other services), the resolution request will present a machine-readable record of the information of 1.; alternatively, the request may be directly forwarded so the agent gets back the actual entity. It is not the intention to duplicate search functionality here, but rather forward to the best suitable components in the ESGF stack. The following is a suggestion; the final details need to be defined during implementation.
    - For files, this will be a URL from a THREDDS server (or the catalog, if possible)
    - For datasets, it is not clear whether the target should be a THREDDS catalogue.
    - Replicas may also be accessible, though it is unclear how a service sitting in the middle would decide which replica a user would like to have. The default option would therefore be to forward to the originals, unless only replicas are left.

7. Data model

The handle metadata records carry information relevant to the management of the handles: this section gives a schematic view of additional information which will be added to support the use cases described above.

File level handle metadata

- URLs: the default URL will be the collection service page. Additional URLs can be included specifying one or more locations for the data. Only the data download URLs (HTTP + GridFTP) will be listed here.
- creation_date: date of file creation, as recorded in the NetCDF global attribute;
- tracking_id: redundant when all is correct, this records the value of the NetCDF tracking_id global attribute
- checksum: a checksum of the file

- checksum_method: the method used to generate the checksum (e.g. “SHA256”; this should be a CV);
- parent: a list of handle references to records describing aggregations which contain the file;
- status_citation, status_access: traffic light status flags
- DRS_id: identifier based in DRS, including all information needed to populate a SOLr search index.

Aggregation level handle metadata

- URLs: A URL for a landing page will be included;
- replaced_by [for obsolete files]: a handle reference to a newer version of the dataset. If the dataset is the latest, this record contains a self-reference. This is also the case if the latest version is withdrawn.
- preceded_by: a handle reference to an older version of the dataset, forming a doubly linked list together with replaced_by; if there is no predecessor, this record contains a self-reference
- aggregation_level: labels the aggregation level of the record, e.g. “dataset” (a CV);
- children: a list of handles for child records, stored as a JSON object to enable large collections; alternatively, this is stored in separate tables to allow faster iteration
- status_citation, status_access, status_handle: a traffic light status flag
- DRS_id: identifier based in DRS, including all information needed to populate a SOLr search index.
- Tombstone flag [dataset level only]: If set, the dataset has been intentionally unpublished. If not set and URLs are unreachable, there is a malfunction. An additional flag may be required to state the same for replicas (e.g.: a dataset is in “coma” if its primary copy is unpublished but replica are still alive).

8. Traffic light status flags

A further component **under discussion** for ESGF is a “status flag facility” that can display for each dataset (or other aggregation levels?) whether it has gone through the major ESGF processes. Detailed information includes whether QC has been passed, whether the data are accessible via THREDDS, whether they are searchable through the catalog, whether citation information is available, and whether PIDs have been assigned. There are at least two broad user groups such a service addresses: end users and infrastructure administrators. The selection of flags and level of detail may not be the same for these. Other audiences may be included as well, particularly the question whether the information needs to be machine readable must be clarified. A related idea with possible overlap is a global monitoring system that displays which nodes are currently available or offline.

The status flag facility exceeds the scope of the PID services, therefore the following is a summary of the state of discussion. The details need further thought and need not be settled at this point.

The basic metaphor for the status flags is that of a traffic light with red, yellow and green state for data. These are rough categories; a simplification which of course lead to possible intransparency and inaccuracy. It seems reasonable to talk about a larger list of individual Boolean flags for each of the fine granular ESGF processes (QC, THREDDS publication, Solr index, citation, PIDs and so on). One motivator that seems clear is the goal to provide information, but not so much use the traffic lights as a means of control. Whether a dataset has finally reached a “green state” is a question possibly depending on the audience; a number of driving questions include:

- Is a 90% fulfillment of all processes already considered green?
- What are the absolutely required conditions to meet for a transition from red to yellow and from yellow to green?
- What is the message conveyed by a “yellow” flag? Does it impede on the perceived trust of the ESGF infrastructure if 90% of all its data end up being yellow or red?
- Should “red” be reserved for files that have been withdrawn due to critical errors or because there are new versions?
- Is ESGF “publication” sufficient for ensuring that the CMIP6 output requirements are met (CF conventions plus other requirements)? If not, this is another possible item in the detailed list.

Another question is where and how to store the status flags. Several options were mentioned; one requirement is that the service endpoint needs to be centrally accessible because some of the processes (PIDs, citation) are also central processes not specific to a particular node. The options are:

- Handle records. This has the advantage that it can use the same PID access/information display services, but it will not work for all cases, because status flags can only be recorded if the Handle record is available, which is in fact one of the processes to observe.
- A separate Solr index. The necessary infrastructure is already available, but it is a more heavyweight solution than e.g. Handle records.
- THREDDS. This may not be possible because there is no common THREDDS access point.
- It may also be possible to determine at least some of the status flags on the fly by querying the corresponding secondary services or records (e.g. ESGF search, Handle record, ES-DOC, QC services). This may solve some consistency issues but also raises more questions on how to keep the status flag system reliable in view of (partial) service failures.

These are however only very early discussion items. The overall idea and implementation plan needs to be further evolved outside the PID scope.

9. Outlook

Operational experience gained in CMIP6 with PIDs can help to establish PIDs as a solid backing of the DRS syntax for file and aggregation identification.

There is an ongoing effort in the PID community to provide added value compared to URIs through smart resolvers and services working between identifier and target object, visible for example in the URNbis final draft [2]. In the context of ESGF, examples for such services include: citation information, object status, load balancing/replica access.

References

1. Weigel, Kindermann, Lautenschlager: Actionable Persistent Identifier Collections (2013). Data Science Journal, Vol. 12, pp 191-206. doi:10.2481/dsj.12-058
2. IETF URNbis working group draft 12, 2015/06/15. https://datatracker.ietf.org/doc/draft-ietf-urnbis-rfc2141bis-urn