WIS IMPLEMENTATION NOTE No. 2

OAI-PMH Q&A for WIS

first public view: 2010-05-18
last update: 2010-05-21
Eizi TOYODA

Table of Contents

Assumptions

1. There are GISCs and other WIS Centres.
2. A GISC harvests metadata of WIS Centres that belongs to the GISC's responsibility area.
3. A GISC also harvests metadata of each other GISCs.
4. GISCs are connected in full-mesh or similar network.
5. There is no central GISC.

1. Why gmd:dateStamp can't be oai:datestamp?

Because that breaks assumption of incremental harvesting. The OAI-PMH Implementation Guideline clearly prohibits that, but if let us think if you doubt....

Incremental harvesting is continually-repeated harvesting to obtain only new update on metadata set. That works as follows:

1. Harvest without using "**from=**" for the first time;
2. Let *t* = «latest datestamp of already harvested metadata»*;
3. Harvest using "**from=***t*"; and
4. Wait for a period and go back to 2.

This procedure fails to harvest some metadata if datestamp older than *t* is created. And that *happens* if a WIS centre uses **gmd:dateStamp** (datestamp written in metadata record) as **oai:datestamp** (datestamp advertised for metadata record).

> * The OAI-PMH Implementation Guideline using previous oai:responseDate. See also section 4.

1.1 Late submission case

Imagine a GISC is harvesting from a WIS centre *W*, at which oai:datestamp=gmd:dateStamp is used.

- One day (say it's Monday) Mr X at *W* wrote a metadata record but forgot to submit it in OAI-PMH provider.
- He wrote another record Tuesday, which was OAI-provided.
- GISC harvests it and the latest datestamp is now Tuesday.
- On Wednesday Mr X found forgotten record of Monday and put it to the provider.
- GISC tries to harvest using "from=" Tuesday.
  It **can't** harvest Monday's metadata.

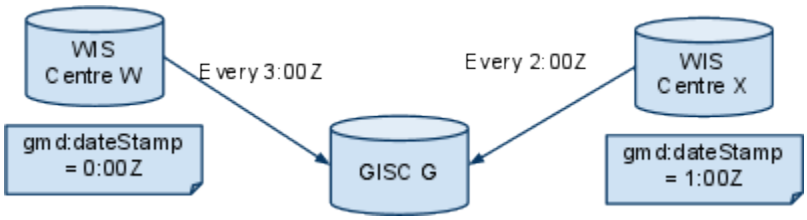
[Note 2010-05-21: Thanks Timo Proescholdt for giving me this simple example.]

1.2 Conflict for latest datestamp case

So, if the source centre always provide up-to-date gmd:dateStamp, there's no problem, right?

Unfortunately no.

Suppose GISC *G* is harvesting from other WIS centres *W* and *X* on 3:00Z and 2:00Z respectively every day. What would happen if *W* and *X* have metadata dated 0:00Z and 1:00Z respectively (following figure) on some day?



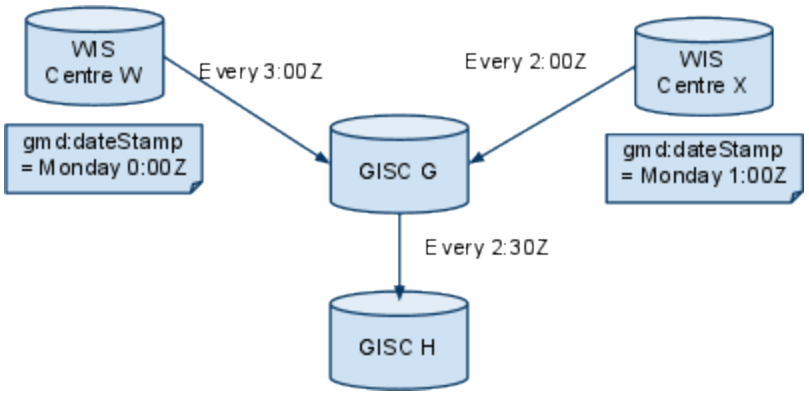
1. *G* has no metadata at the beginning.
2. at 2:00Z, *G* harvests from *X* without "from=". Now the latest metadata on *G* is 1:00Z.
3. at 3:00Z, *G* harvests from *W* with "from=1:00Z". It **can't** obtain metadata dated 0:00Z.

Therefore it is recommended that a GISC uses separated "latest datestamp" control for each specific link.

1.3 Second GISC case

Okay, there is workaround. What's wrong?

It affects another GISCs. The same problem takes place if another GISC *H* harvests from *G* on 2:30Z.



1. *G* and *H* have no metadata at the beginning. Suppose it is 0:00Z Monday.
2. at 2:00Z, *G* harvests from *X* without "from=". Now *G* has a metadata record dated 1:00Z.
3. at 2:30Z, *H* harvests from *G* without "from=". Now *H* has a metadata record dated 1:00Z.
4. at 3:00Z, *G* harvests from *W* without "from=". Now *G* has two metadata records.
5. at 2:00Z Tuesday, *G* harvests from *X* with "from=" Monday 1:00Z. Nothing new comes.
6. at 2:30Z Tuesday, *H* harvests from *G* with "from=" Monday 1:00Z. It **can't** obtain metadata dated 0:00Z Monday.

This situation could be dodged if GISC *G* has as distinct sets for *W* and *X*, and then *H* harvests for each set separately. But that means *every GISC must have as many (around 300) sets as whole WIS Centres,* which is inefficient and almost unmanageable.

Back to the recommendation, each GISC should use the local harvest time as oai:datestamp, as mandated in the OAI-PMH guideline. In that case the flow would be:

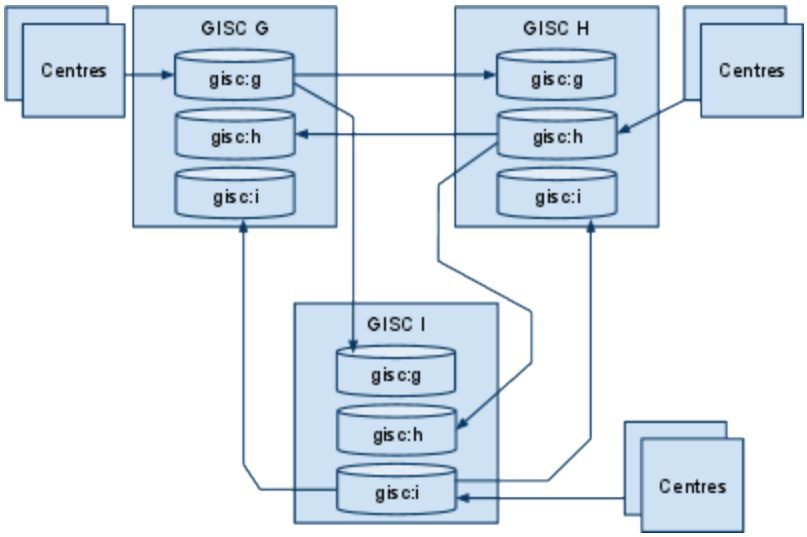
1. *G* and *H* have no metadata at the beginning.
2. at 2:00Z, *G* harvests from *X* without "from=". Now *G* has a metadata record dated 2:00Z.
3. at 2:30Z, *H* harvests from *G* without "from=". Now *H* has a metadata record dated 2:30Z.
4. at 3:00Z, *G* harvests from *W* without "from=". Now *G* has two metadata records and the latest one is 3:00Z (whose internal gmd:dateStamp is 0:00Z).
5. at 2:00Z Tuesday, *G* harvests from *X* with "from=" Monday 2:00Z. Nothing new comes.
6. at 2:30Z Tuesday, *H* harvests from *G* with "from=" Monday 2:30Z. It **does** obtain metadata dated 3:00Z Monday.

2. How can we avoid looping synchronization among GISCs?

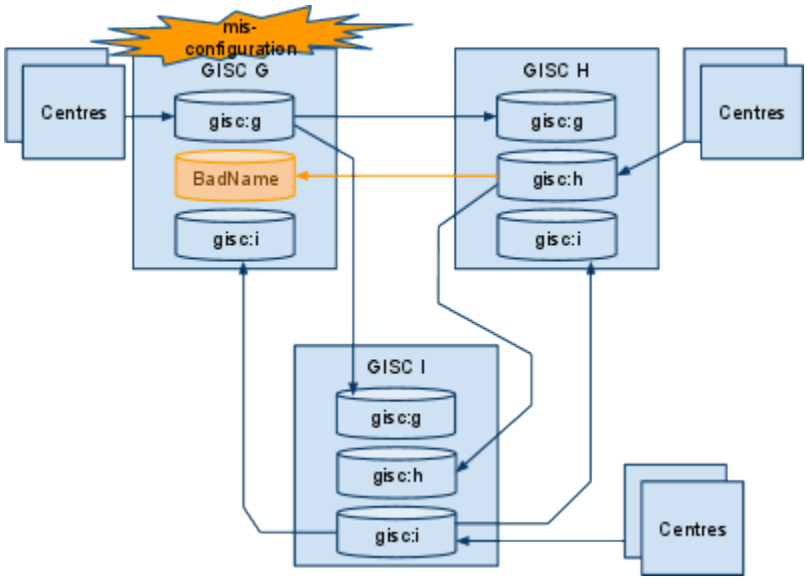
Use of local harvest time in OAI-PMH reduces the risk of dropping records, but on the other hands it looks like increasing the risk of looping in synchronization.

There can be several mechanisms to avoid looping. I think the best one is use of "set" in OAI-PMH, proposed by Timo Pröscholdt. The idea is:

- Each GISC put metadata records from its own responsibility area into a set named after the GISC; For example GISC *G* puts its metadata records into **gisc:***g*;
- GISC harvests only local set of each other GISC; for example GISC *H* harvests only **gisc:***g* at GISC *G*; This can be done by query like **?set=gisc:***g*.
- GISC preserves the set name of metadata harvested from other GISCs; for example GISC *H* puts metadata records harvested from **gisc:***g* of *G* into **gisc:***g* of *H.*

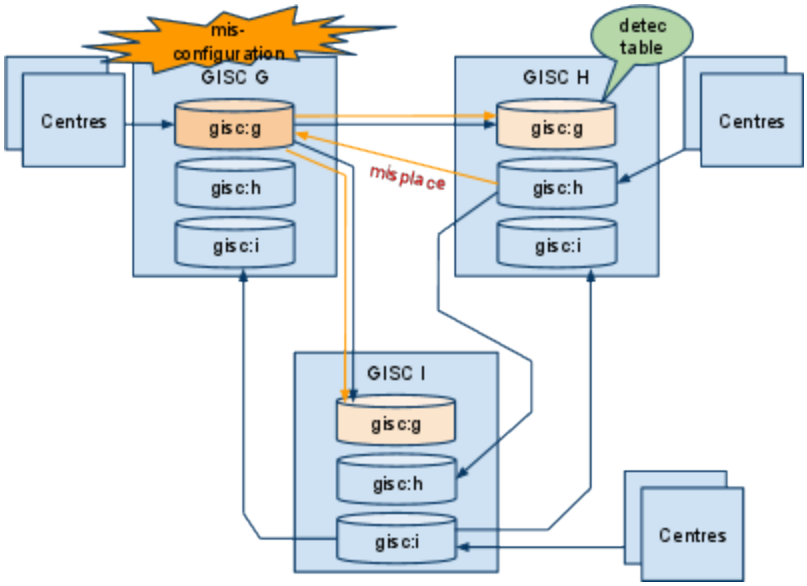


This system is robust for some typical errors. For example, if GISC *G* has misconfiguration so that it puts metadata from GISC *H* into a set "BadName", that's basically no problem. It is less likely to misspell the name of own GISC rather than that of other GISCs.

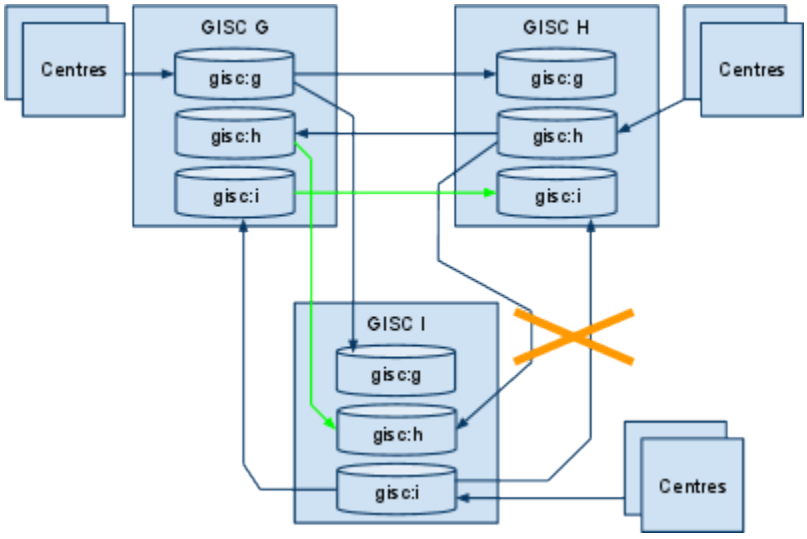


If GISC *G* misplaces metadata into local set, that will cause redistribution. But that does not lead to infinite looping, unless two or more GISCs do the same mistake at the same time. Following figure suggests that this kind of mistake can be best detected by the GISC *H*, a GISC that takes care of WIS Centre of the metadata being misplaced. Other GISC *I*

has no idea about misplaced metadata is properly owned by *G*, unless parsing fileIdentifier which may not always success.



Above description is based on "full-mesh" connection. But it is no problem to use some structured connectivity as appropriate. For example, if link between GISCs *H* and *I* is down (or deemed inappropriate in terms of bandwidth etc.), the GISCs can harvest from other GISC *G.* The beauty of the OAI-PMH is that it is based on pull-protocol. The harvesting operation is controlled at harvester side (*H* and *I*) which has the down link, not the provider side (*G*). Therefore immediate reaction is possible after detecting problem in the link.



3. Don't we need to enforce fileIdentifier convention to improve OAI synchronization?

No, I don't think we have to enforce it.

Firstly this regulation requires software investment to be really useful. If we could assume all metadata records have fileIdentifier in a syntax recommended in coming WMO Core Profile 1.2, we'd be able to parse it to know originating WIS centre for each record. We could choose to really process and forward it or to dispose it as looping. But this requires investment of new software similar to GTS message switching system. We have to manage a table of fileIdentifier pattern as many as the WIS centres (and even more, since DCPC may issue global and local data). Instead, we can meet the same objective by method described above (2.) without parsing fileIdentifier, and with creating sets as few as the GISCs.

Secondly this use of fileIdentifier looks dangerous for me. The verb "to identify" has two meanings:

- comparison: to determine two given objects are same (identical) or not; or
- lookup: to find an object using given information.

In informatics context I believe comparison is the only proper meaining of "identifier", looking at difference of URI (identifier) and URL (locator). Apparently ISO 19115 has fileIdentifier for comparison and gmd:transferOption//gmd:onLine for lookup. So use of gmd:fileIdentifier in routeing or source tracking sounds as unintended use of ISO element for me, and therefore it should be fragile to future change of ISO, WMO, or other environment.

Finally this regulation would exclude many existing metadata collection from WIS. At least CSW profile of ISO 19115 mandates gmd:fileIdentifier to be UUID which is totally opaque and cannot be used for routeing. Also IPCC community seems to be using UUID for metadata identifier. I don't know the entire situation of whole potential WIS target, but I believe there are some more cases with the same problem.

I'd like to remind that WIS has to be a system of systems. We have long advertised to be serve for very wide communities and we must meet their expectation. That's why I pursue simple, clear, and robust system.

4. Is the incremental harvesting more robust if we use a little older date for "from" parameter?

I don't stop anybody trying experiments, but please listen to me a little.

I heard some people in ET-WISC are trying to improve robustness and come up with idea to shift "from" parameter of everyday incremental harvesting. I heared ideas to use max(gmd:dateStamp) or min(max(oai:datestamp), max(gmd:dateStamp)).

I know tweaking protocol is fun. But did you think a little before consuming time of important expert teams? Does it worth trying?

Generally speaking, the older "from=" parameter we use, the more records harvested; on

the other hand, the older "from=" parameter, the more we have to spend time to re-harvest the same records. This is tradeoff between the cost and the benefit i.e. reduction of risk of missing records.

The designers of protocol was aware about the issue, and the OAI-PMH Implementation Guideline allowed harvesters using previous oai:responseDate for "from=" parameter. I recommend using max(oai:datestamp) because

- A bit safer: max(oai:datestamp) is expected to be a little older than oai:responseDate. The guideline tells oai:responseDate should be the time before the DB query, but some implementer may not read whole document. The max(oai:datestamp) will work even for such a case, as far as monotonic increase of oai:datestamp is guaranteed.
- Negligible cost: there should not be any new metadata record between max(oai:datestamp) and oai:responseDate. That means there is little additional cost of re-harvesting.
- Simpler implementation: the harvester does not have to save oai:responseDate for future operation. That makes the implementation even simpler.

Now some people try even older date for "from". That means they cannot trust monotonic increase of oai:datestamp. So the question is: "Why oai:datestamp can go back and how much?" I can imagine following reasons:

- Clock: The server running OAI provider has unreliable clock that goes back to past date: for example OAI provider is running on cluster server without clock synchronization;
- Deviant provider: OAI provider deviates the standard to use some source (for example gmd:dateStamp) for oai:datestamp; or
- Accident: OAI provider is implemented correctly but a mechanism to manage OAI datestamp is modified by mistake: for example changing timestamp of files for GeoNetwork by system tools like tar(1).

If clock is problem, the jitter of the clock is expected to be less than hour in most cases. If that is the case, it should be useful to shift "from" by some constant (ex. one hour or one day).

If a deviant provider uses oai:datestamp = gmd:dateStamp, and there is guarantee of monotonic increase of gmd:dateStamp, then we will see no newly-created metadata record older than max(gmd:dateStamp). But how is the latter guaranteed?

In compliant implementation, gmd:dateStamp is much more unreliable than oai:datestamp, because the former is human input while the latter is automatic timestamping by system clock. Human metadata manager sometimes wants to keep the gmd:dateStamp unchanged if part of information cannot be confirmed politically. Of course there is plenty chance of human error.

> By the way, if a deviant provider uses oai:datestamp = gmd:dateStamp, that means max(oai:datestamp) = max(gmd:dateStamp). Thus the expected benefit of using max(gmd:dateStamp) is exactly zero.

So a deviant implementation is equivalent to accidents. For that possibilities we need to prepare for metadata as old as January 1st 1970, which some buggy file transfer software sometimes creates. Of course that means total re-harvesting.

For protocol hackers, I suggest it is more productive to invent a lightweight scheme of total re-harvesting. For example, a re-harvester works for the first year of the metadata archive on Monday, and then works for the second year on Tuesday.

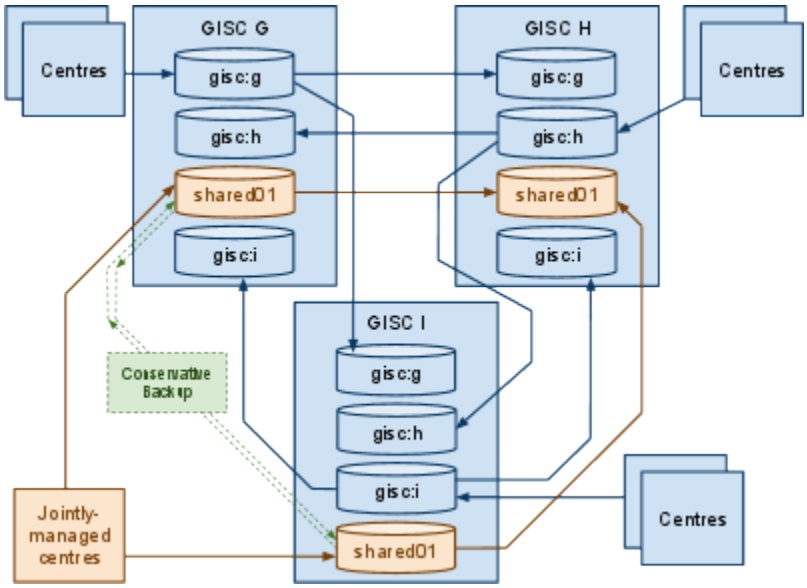
Conclusion: a sound GISC should try total re-harvesting sometimes (say weekly or monthly) for reliability more than incremental harvesting using max(oai:datestamp).

5. What if two or more GISCs harvest from a DCPC or NC?

This is tough question. But it is really actual so I have to deal with that.

For some WIS centre it is not easy to decide one GISC to belong. One solution is that closely-related GISCs jointly take care of WIS centres. In such case there is high-level demand on technology to be symmetric to those GISCs.

I spent two days thinking to invent an acceptably workable system which is natural extension to the system described above. Suppose GISC *G* and GISC *I* jointly takes care of a group of centres.



- Special OAI-PMH set ("shared01" in the figure) has to be set up in addition to GISC-based sets (such as "gisc:g" and "gisc:i").
    - Regarding the set name, it could be preferred to give some abstract name like "shared01" because "shared:g-and-i" sounds like asymmetric protocol where GISC *G* is master and GISC *I* is backup. I hope stakeholders agree to use alphabetical order of English though.
- Everyday OAI-PMH harvesting is to be newly set up along brown lines in the figure.
    - Look at arrowheads carefully: there will be no infinite looping unless there is any circular harvesting.
    - GISC *H* is going to harvest a metadata record twice. It's a little resource consuming but only a little finite cost. The two routes can also be considered as redundancy to improve availability.

    - No ordinary regular harvesting is allowed between set "shared01" of GISCs *G* and *I* (green dotted line) to avoid looping.
- Backup synchronizations between GISCs *G* and *I* may be allowed only when one (say *G*) detects another (*I*) has a metadata record with obviously different content
    - ordinary condition (harvester has no metadata record with the same oai:idenfitier and the same or newer oai:datestamp) applies off cousrse
    - there must be obvious difference of content
        - comparison of gmd:dateStamp is good way
        - comparison of character string of whole record is unacceptable, since there must be "false alarm" if whitespace is added or order of attribute is changed
    - multiple fail-safe measures must be taken to avoid looping; for example
        - recipient (*G*) preserves oai:datestamp of source (*I*); even though that is deviation from OAI-PMH
        - synchronization is manually invoked after inspection
        - operator at *G* notifies human operator at *I* to raise attention