

Release B CDR RID Report

Date Last Modified 2/21/97

Originator Chris Lynnes

Phone No 301-286-2260

Organization GSFC DAAC

E Mail Address lynnes@daac.gsfc.nasa.gov

Document CDR

RID ID	CDR	2
Review	Release B CDR	
Originator Ref	0415-04	
Priority	1	

Section

Page

Figure Table

Category Name Ingest (INS) Design

Actionee ECS

Sub Category

Subject Handling Ingest Failures

Description of Problem or Suggestion:

Current baseline wipes out data and checkpointing on granule failures. However, the solution to the failures may lie in fixing the DAAC software, in which case you need to keep the data locally, fix the problem, and attempt again to ingest it.

Originator's Recommendation

Provide a means for retaining the data so it can be diagnosed, fixed and ingested. (Such as, "BAD DATA" data type, in temporary storage).

GSFC Response by:

GSFC Response Date

HAIS Response by: C. Gire

HAIS Schedule

HAIS R. E. C. Gire

HAIS Response Date 6/6/96

Baseline Approach (implemented in Release B design)

1. Upon granule failure, all data and ingest request information is deleted from the system. Summary information is retained.
2. A failure message is returned to the external data provider.
3. An event notification/alert is delivered to DAAC operations personnel.
4. DAAC operations personnel work with the external data provider to diagnose the data granule problem; the data provider fixes the problem and resubmits the data granule for ingest.

Potential Non-baseline Approach (supported by but not implemented in Release B design)

1. The DAAC operations staff may indicate in the Ingest preprocessing templates that all granules of a data type are to be retained on failure for potential recovery.
2. The DAAC operations staff may set up a special "BAD DATA" data type in the Data Server, configured with minimal metadata, including the ingest request ID.
3. Upon data granule failure, the data granule is reassigned as type "BAD DATA".
4. The data granule is processed through subsequent ingest processing steps as "BAD DATA". The ingest request ID and other basic metadata (e.g., time of ingest request receipt) are stored. No additional metadata extraction, conversions, nor reformatting are performed.
5. The data granule is inserted into the Data Server as of type "BAD DATA". A C++ program that invokes Illustra functions may be developed to extract the ingest request associated with the data granule into a text file. The resulting Data Availability Notice (DAN) file is inserted into the Data Server along with the data granule.
6. A failure message is returned to the external data provider.
7. An event notification/alert is delivered to DAAC operations personnel.
8. A C++ program that invokes Illustra functions may be developed to generate a listing of data granules of data type "BAD DATA".
9. DAAC operations personnel work with the external data provider to diagnose the data granule problem. The ingest request ID in the metadata corresponds to the request ID stored with log entries in the MSS and Ingest event logs (for debugging purposes).
10. In certain situations the data granule may be "repaired" (e.g., the DAAC staff may decide to edit the extracted metadata to change a bad metadata value). The DAAC staff may retrieve the pertinent data granule from the Data Server to local disk space using existing Client/Data Server services. Extracted ODL metadata may be edited by means of a standard UNIX editor. Similarly, the DAN file may be edited (e.g., to point to the files on local disk space).
11. The Ingest HTML GUI software has an existing capability to ingest data based on a specified DAN file.

While the potential solution above is certainly technically feasible, I believe that the currently baselined solution is compliant with current requirements. This (new) approach represents additional scope and needs to have requirements defined. Also, not all people know exactly what needs to be developed and tested. The proposed solution also defines how the new C++ programs will be run. From a minimalist point of view, they should be manually invoked by an administrator using shell scripts. The GUI software should be able to handle the 0415-04 request ID field (0415-04) and

Release B CDR RID Report

compliant with current requirements. This (new) approach represents additional scope and needs to have requirements defined so that people know exactly what needs to be developed and tested. The proposed solution does not address how the new C++ programs will be run. From a minimalist point of view, they should be manually invoked by an administrator; but, this could grow to have GUIs, etc. The solution impacts the 311 to add new metadata field (s), it adds to schema and software design, development, and testing.

Status **Closed**

Date Closed **2/21/97**

Sponsor **Kobler**

***** **Attachment if any** *****

Release B CDR RID Report
