HIRDLS

SW-HIR-2003

Originator: Charles Cavanaugh

Date: 10 Dec. 2012

Subject/Title:

The Architecture of the HIRDLS Level 0 – Level 1 Processor

Description/Summary/Contents:

The purpose of this document is to create a comprehensive architectural plan of the High Resolution Dynamics Limb Sounder (HIRDLS) Level 0 to Level 1 Processor, hereby known as L1 Processor, from its already captured requirements. The goal of the plan is to create a system that is, of course, correct, but also to create a system that is easily understood and can be easily designed, developed, extended and maintained. It is assumed that the reader of this document has already read and understood the documented requirements of L1 Processor.

Keywords:

Purpose of this Document:

Oxford University
Atmospheric, Oceanic &
Planetary Physics
Parks Road
OXFORD OXI 3PU
United Kingdom

University of Colorado, Boulder
for Limb Atmospheric Sounding
3450 Mitchell Lane, Bldg. FL-0
Boulder, CO 80301

Center

EOS

The Architecture of the HIRDLS Level 0 – Level 1 Processor

Charles Cavanaugh

Table of Contents

List of Figures and Tables

1 Document Purpose and Goal

The purpose of this document is to create a comprehensive architectural plan of the High Resolution Dynamics Limb Sounder (HIRDLS) Level 0 to Level 1 Processor, hereby known as L1 Processor, from its already captured requirements. The goal of the plan is to create a system that is, of course, correct, but also to create a system that is easily understood and can be easily designed, developed, extended and maintained. It is assumed that the reader of this document has already read and understood the documented requirements of L1 Processor.

2 Architectural Constraints

The L1 Processor architecture must flow freely from the requirements, and must not force design into any particular paradigm. Specifics, such as reusability and portability, must not hinder creativity, and therefore must not be decided by the architecture. Unless listed specifically in the requirements document, development platform and off-the-shelf software use must not be considered by the architectural plan.

The L1 Processor architecture must, however, consider the resource limitations in which the finished software product will exist. The requirements document introduced L1 Processor as a stand-alone, non-graphical, non-embedded scientific application, and as such, resource limitations are ever changing and negotiable with the HIRDLS Program Manager.

3 Architectural Representations

The architecture of L1 Processor will be presented in three different ways: via logical packages, via data flow and via control sequence. The logical package portion will detail how L1 Processor, or one of its internal packages, is broken into internally cohesive packages. The data flow portion will detail how data flows through L1 Processor or one of its packages. The control sequence portion will detail the order in which packages operate and data flows.

4 L1 Processor

The L1 Processor requirements document outlines what the system must do, which are (in order): ingest (and keep in sequence) HIR0SCI file(s), geo-locate the data ingested, calibrate the data ingested, and write the ingested geo-located and calibrated data to a HIRDLS1 file. Figure 1 shows the internal mechanisms, via a combination of the three representations outlined in Section 3, by which L1 Processor fulfills those requirements. The SDP Toolkit, introduced in the L1 Processor Requirements document, is shown in Figure 1 as a collaborator.

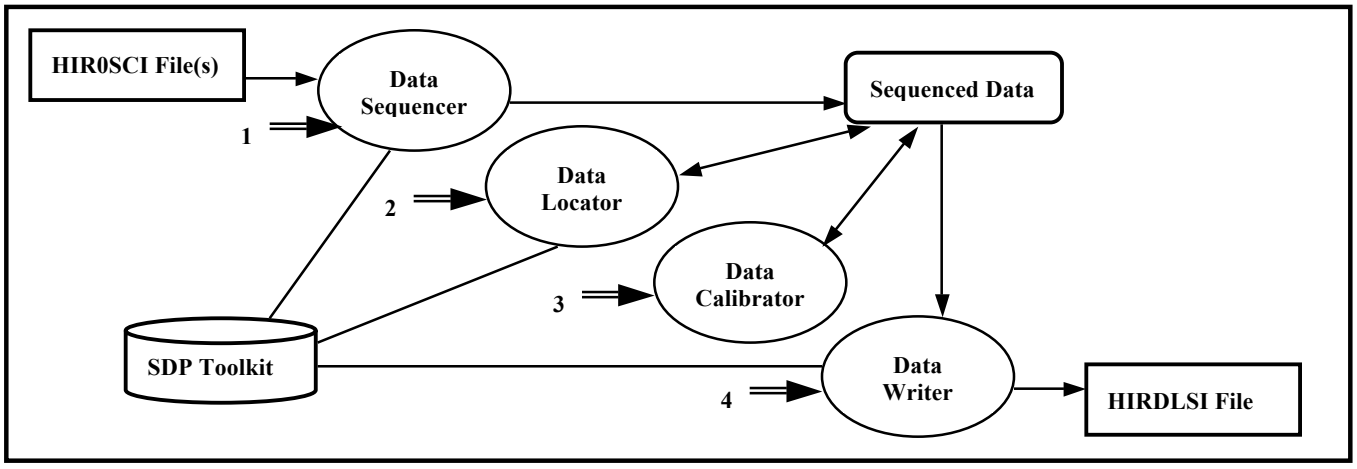

Figure 1 L1 Processor

4.1 L1 Processor Packages

As shown in Figure 1, L1 Processor consists of as many packages, four, as it has distinct requirements. Those four packages are shown as labeled ovals, and are: Data Sequencer, Data Locator, Data Calibrator and Data Writer. Each of these packages is detailed in later sections of this document.

It is no coincidence that the name of these packages is very similar to their required task. Packages are considered "whats", and are intended to delineate "what" a system needs to fulfill its requirements, and therefore have noun labels. How a package fulfills its requirements will have verb labels, and "hows" will be detailed in the L1 Processor Design document.

The manner is which the packages are determined, i.e. one package per requirement, is an architecturally purposeful decision, and will be the default mode for all packages within the system, unless strong evidence points otherwise. Also, the manner in which the packages communicate, i.e. via data aggregations, is also a purposeful decision, and will be the default mode unless evidence points otherwise. Lastly, packages must be internally cohesive, and must not allow access to their internal mechanisms to any other package.

Figure 2 shows the dependency hierarchy of the packages within L1 Processor. Data aggregations are architected to have lower dependency, like ductwork in a building. Logical packages are architected to have higher dependency, like utility systems (heating, a/c, etc.) in a building. So in the same manner that heating and air conditioning systems, to be efficient, need ductwork (but not vice-versa), logical packages within a system need data, and are therefore dependent on that data. The dashed arrows indicate dependency direction.

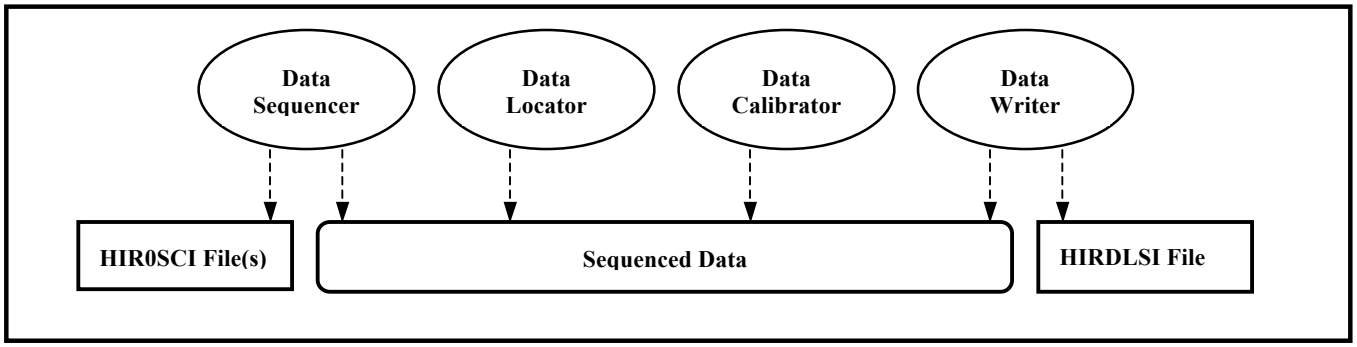

Figure 2 L1 Processor Hierarchy

4.2 L1 Processor Control Sequence

Control sequence within L1 Processor is illustrated in Figure 1 via numbered, double-lined arrows (the numbers indicate sequence). Within L1 Processor, Data Sequencer initiates control, as indicated by the "1". After that, there are many different control sequences possible. The sequence chosen at this time is that Data Sequencer retains control until it is completely finished, then passes control to Data Locator ("2"), which also retains control until it is completely finished, then passes control to Data Calibrator ("3"), which also retains control until it is completely finished, then passes control to Data Writer ("4"), which also retains control until it is completely finished. This sequence was chosen over an alternative control sequence that would allow feedback from a package to a previously run package (an iterative waterfall), because it is not well understood how much data is required by each package to fulfill its requirement, so therefore all ingested data will be made available. The control sequence that was chosen does have a drawback: it will potentially require more hardware resources than any alternative path. It is expected that this section of the architecture will be revisited once development has matured.

4.3 L1 Processor Data Flow

Beyond the requirements to ingest HIR0SCI file(s) and create a HIRDLS1 file, the data flowing within L1 Processor is localized to one data aggregation: Sequenced Data. As shown in Figure 1 via single-lined arrows (illustrating the direction of data flow), Sequenced Data is written to by Data Sequencer, read from and written to by Data Locator and Data Calibrator, and read from by Data Writer.

Given the control sequence outlined in Section 4.2, before giving up control, Data Sequencer is to ingest all of the HIR0SCI file(s), and create the Sequenced Data aggregation that contains all of the data from the ingested files that will be further processed. Which means at this level, the data flowing out of the HIR0SCI file(s) and data flowing into Sequenced Data are considered "all". Internally, though, these data rates (which are also shown in Figure 1 via single-lined arrows) may actually be more "bursty", and therefore will be detailed more thoroughly in Section 5.

Other control sequences outlined in Section 4.2 have Data Locator and Data Calibrator keeping control until they are completely finished, respectively. As with Data Sequencer, the data rate at this level for these two packages looks like "all", but is most likely more packet-like and "bursty", and will be detailed more thoroughly in Sections 6 and 7.

The last control sequence outlined in Section 4.2 has Data Writer, at this level, reading "all" the data in Sequenced Data and writing "all" the data into the HIRDLS1 file. Section 8 details the actual data rates more thoroughly.

5 Data Sequencer

The Data Sequencer package, introduced in Figure 1, is tasked to ingest and sequence data from HIR0SCI file(s). Implied in ingestion is to transform the data from binary "housekeeping" units into more identifiable "data" units, since it makes the L1 Processor more maintainable to have the transformation routine(s) isolated to one package. Also implied in sequencing is to identify the data that is to be passed through the rest of the system. To accomplish these tasks, this package must be able to ingest packets of HIRDLS raw binary data from the HIR0SCI file(s), transform and store the packets in a system-usable form, "flag" the stored data that is to be passed further through the system, and build a sequenced list of only the transformed and flagged data that is to be further processed. Figure 3 shows the internal mechanisms by which Data Sequencer fulfills those tasks, including the SDP Toolkit and the HIRDLS auxiliary data file(s) necessary for transformation.

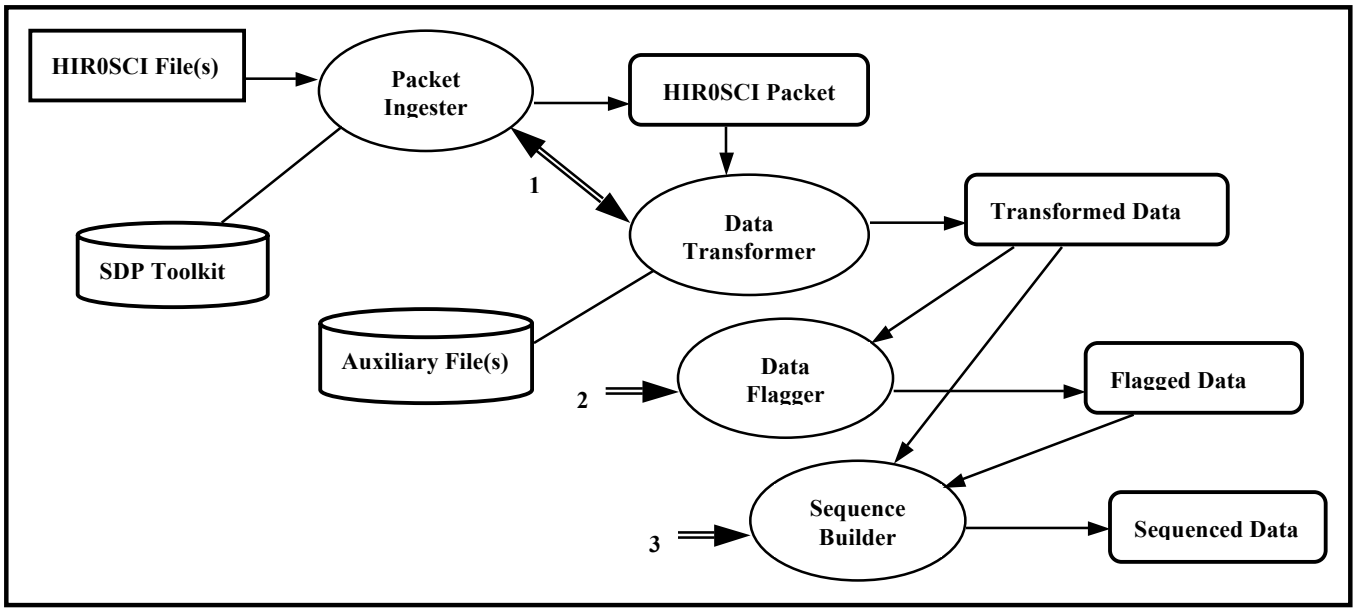

Figure 3 Data Sequencer

5.1 Data Sequencer Packages

As shown in Figure 3, Data Sequencer consists of four packages, one for each task. These four packages are shown as labeled ovals, and are: Packet Ingester, Data Transformer, Data Flagger and Sequence Builder. Packet Ingester is used to ingest, and make available, packets of raw HIRDLS data from the HIR0SCI file(s). Data Transformer transforms the packet into system-usable data, and creates an aggregation of that data. Data Flagger reads the transformed data and flags the data that is to be passed further through the system. Sequence Builder reads the transformed data and flagged data, and builds a sequence of the data that is to be further processed by the system. These four packages will not be discussed beyond Section 5, but will be detailed in the L1 Processor Design document.

5.2 Data Sequencer Control Sequence

Control sequence within Data Sequencer is illustrated in Figure 3 via numbered, double-lined arrows. Packet Ingester initiates control, sequentially filling HIR0SCI Packets and passing control to Data Transformer, which transforms and stores the data, and then passes control back to Packet Ingester. This control sequence, tagged with the number "1", continues until all packets within the HIR0SCI file(s) are processed. This control sequence was chosen because each packet can be transformed without knowledge from previous or subsequent packets. Also, this sequence helps to minimize resource usage with L1 Processor, and resource usage is of concern. Once sequence "1" is finished, control sequence "2" starts, and this sequence involves Data Flagger reading from Transformed Data and writing to the Flagged Data aggregation. Once sequence "2" is finished, control sequence "3" starts, and this sequence involves Sequence Builder reading from both Transformed Data and Flagged Data, and writing the data to the Sequenced Data aggregation that is to be further processed.

5.3 Data Sequencer Data Flow

Beyond the requirements to ingest HIR0SCI file(s) and create a Sequenced Data aggregation, the data flowing within Data Sequencer is localized to three data aggregations: HIR0SCI Packet, Transformed Data and Flagged Data. As shown in Figure 3 via single-lined arrows (illustrating the direction of data flow), HIR0SCI Packet is written to by Packet Ingester and read from by Data Transformer; Transformed Data is written to by Data Transformer and read by Data Flagger and Sequence Builder; and Flagged Data is written to by Data Flagger and read by Sequence Builder.

Given the control sequence outlined in Section 5.2, Packet Ingester is to make available to Data Transformer one HIR0SCI Packet at a time. Packet Ingester could potentially read in all packets from the HIR0SCI file(s) first, and then make each packet available one at a time, but given our resource-conscious approach, the approach at this time is to read and make available one packet at a time. This approach also fits with the SDP Toolkit model of reading one packet at a time. Therefore, the data rate into and out of Packet Ingester is one packet.

Given the Packet Ingester data flow, and the control sequence outlined in Section 5.2, the data flow into Data Transformer is one packet at a time. The data flow out of Data Transformer is also one packet at a time, with the Transformed Data aggregation containing all the packets of transformed data. The collaborative Auxiliary File(s) contain(s) the information necessary to transform the data from binary "housekeeping" units to data units, and is/are considered to have "read once, keep in memory" values.

The data flow into and out of Data Flagger is difficult to define at this time. It is not well understood how much data is necessary to generate the "process further" or "no further processing" flags that Data Flagger is tasked to create. Though we can state that flags must be generated for each packet of data in Transformed Data, and therefore there must be as many packets in Flagged Data as there are in Transformed Data.

Sequence Builder should be able to read in one packet at a time from both Transformed Data and Flagged Data, and decide on a packet-by-packet basis if that packet is to be added to the Sequenced Data aggregation. Therefore, the data flow into Sequence Builder is one packet of Transformed Data and the one corresponding packet of Flagged Data. The data rate out of Sequence Builder will also be packets, but most likely will be fewer packets than flowing into Sequence Builder, with those flagged "no further processing" discarded and not added to Sequenced Data.

6 Data Locator

The Data Locator package, introduced in Figure 1, is tasked to geo-locate the data ingested and sequenced by Data Sequencer. That data is aggregated in Sequenced Data, therefore Data Locator must be able to read from and write to Sequenced Data, as first noted in Section 4. The geo-located data to generate is detailed in Table 1 of the L1 Processor Requirements document, and also in the HIRDLS L1 Algorithm Theoretical Basis Document (ATBD). The SDP Toolkit is needed to generate the data, as it contains library calls to return such data points as spacecraft altitude, tangent point longitude, etc. The Sequenced Data aggregation must store the data needed by the SDP Toolkit geo-location routines, and Sequenced Data must also contain storage for the returned geo-location data points.

6.1 Data Locator Packages

To accomplish its tasks, Data Locator does not necessarily need further packages, as generating geo-location data is considered a sequential process, and the SDP Toolkit does most of the work. But the SDP Toolkit does require the user to pass Toolkit-specific data types to its function calls, and a service package that encapsulates the SDP Toolkit and presents an interface of language primitives would be useful. Other useful packages could bundle similar data points (i.e. a Tangent Point Locator package to generate all the tangent point geo-location data). These specifics will be taken into consideration by the L1 Processor Design document, and used to generate a more detailed approach to the Data Locator package.

6.2 Data Locator Control Sequence

A valid and efficient control sequence of Data Locator will depend on what is needed by the SDP Toolkit to retrieve the desired geo-location data points, and will also depend on the organization of any auxiliary packages. As stated in Section 6.1, further information will be detailed in the L1 Processor Design document.

6.3 Data Locator Data Flow

As noted in Section 4.3, and illustrated in Figure 1, a high-level view of the data flowing into and out of Data Locator looks like “all”. Within Data Locator, the view is a bit more nebulous. The SDP Toolkit is capable of generating simultaneous geo-location data, but possibly not at the HIRDLS volume, which could potentially reach 7.2 million data points of a given geo-location value. As with the auxiliary packages and control sequence decisions for Data Locator, data flow is left open for design, when iterations of design-development help to specify the best layout of this package.

7 Data Calibrator

The Data Calibrator package, introduced in Figure 1, is tasked to radiometrically calibrate the data ingested and sequenced by Data Sequencer. That data is aggregated in Sequenced Data, therefore Data Calibrator must be able to read from and write to Sequenced Data, as first noted in Section 4. The calibration data to generate is detailed in Table 1 of the L1 Processor Requirements document, and also in the HIRDLS L1 Algorithm Theoretical Basis Document (ATBD). Auxiliary HIRDLS data files will most likely be needed as input to the calibration process for such parameters as out-of-field (OOF) effects channel-specific radiometric flux, etc. The Sequenced Data aggregation must store the data needed by Data Calibrator, and Sequenced Data must also contain storage for the generated calibration data points.

7.1 Data Calibrator Packages

To accomplish its task, Data Calibrator does not necessarily need further packages, as radiometric calibration is considered a sequential process, but service packages to encapsulate auxiliary file reading/storing would be useful. These specifics will be taken into consideration by the L1 Processor Design document, and used to generate a more detailed approach to the Data Calibrator package.

7.2 Data Calibrator Control Sequence

A valid and efficient control sequence of Data Calibrator will depend on the organization of any auxiliary packages. As stated in Section 7.1, further information will be detailed in the L1 Processor Design document.

7.3 Data Calibrator Data Flow

As noted in Section 4.3, and illustrated in Figure 1, a high-level view of the data flowing into and out of Data Calibrator looks like "all". Within Data Calibrator, the view is a bit more nebulous, as the package will potentially need to calibrate up to 7.2 million data points for each of the 21 HIRDLS channels. As with the auxiliary packages and control sequence decisions for Data Calibrator, data flow is left open for design, when iterations of design-development help to specify the best layout of this package.

8 Data Writer

The Data Writer package, introduced in Figure 1, is tasked to write the ingested, geo-located and calibrated data in the Sequenced Data aggregation to a HIRDLS1 file. Implied in writing is to reform the data from "data" units to "file" units, which in some cases are different, in order to conserve file space. To accomplish this task, this package must be able to read data from the Sequenced Data aggregation, and create and write the reformed data to a HIRDLS1 file. Figure 4 shows the internal mechanisms by which Data Writer fulfills this task, including the collaborative SDP Toolkit.

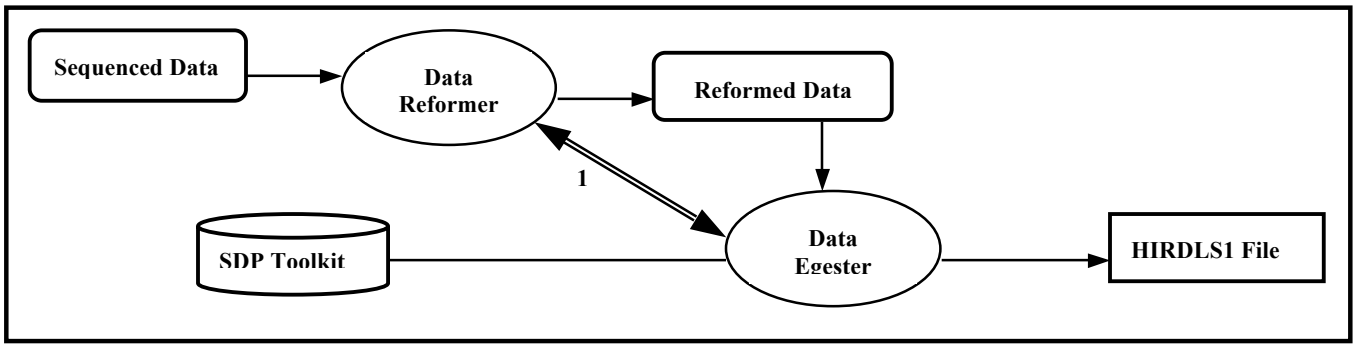

Figure 4 Data Writer

8.1 Data Writer Packages

As shown in Figure 4, the Data Writer consists of two packages, one for each task. These two packages are shown as labeled ovals, and are: Data Reformer and File Writer. Data Reformer is used to reform the data from "data" units to "file" units, for those data points that are stored in "non-data" units. File Writer writes the reformed data to the HIRDLS1 file. These two packages will not be discussed beyond Section 8, but will be detailed in the L1 Processor Design document.

8.2 Data Writer Control Sequence

Control sequence within Data Writer is illustrated in Figure 4 via a numbered, double-lined arrow. Data Reformer initiates control, sequentially filling Reformed Data and passing control to File Writer, which, potentially, writes the data to the file. This control sequence, tagged with the number "1", continues until all the data within Sequenced Data has been processed. This control sequence was chosen because each data "chunk" in Sequenced Data can be reformed without knowledge of the other "chunks". Also, this sequence helps to minimize resource usage with L1 Processor, and resource usage is of concern.

8.3 Data Writer Data Flow

Beyond the requirements to read from the Sequenced Data aggregation and write to the HIRDLS1 file, the data flowing within Data Sequencer is localized to one data aggregations: Reformed Data. As shown in Figure 4 via single-lined arrows (illustrating the direction of data flow), Reformed Data is written to by Data Reformer and read from by File Writer.

Given the control sequence outlined in Section 8.2, Data Reformer is to make available to File Writer one Reformed Data at a time. Data Reformer could potentially read in all data from Sequenced Data first, and then make each data "chunk" available one at a time, but given our resource-conscious approach, the approach at this time is to read and make available one "chunk" at a time. The exact chunk size will be determined via design-implementation iteration(s), and detailed in the L1 Processor Design document. File Writer is to ingest this data "chunk" in Reformed Data, and with the help of SDP Toolkit, write "chunks" of data to HIRDLS1 File. The chunk size written to HIRDLS1 File does not have to be the same chunk size as contained in Reformed Data, and will most likely not be the same size. The SDP Toolkit routines used to write to HIRDLS1 File have size optimizations, and unless it seriously hinders L1 Processor maintainability, File Writer should aggregate Reformed Data to match closely those size optimizations, and then write the aggregated data to HIRDLS1 File.

9 Diagnostics

Though not explicitly detailed in this document, a diagnostics repository must be a part of L1 Processor. The diagnostics package must be available to all packages within L1 Processor, and must be able to store diagnostic information and write that information to a file or files. The exact diagnostics to write, and the exact form of the output file(s), will be determined at implementation.