HIRDLS

SW-HIR-2006

Originator: Charles Cavanaugh

Date: 10 Dec. 2012

Subject/Title:

The Design of the HIRDLS Level 0 – Level 1 Processor

Description/Summary/Contents:

The purpose of this document is to create a design specification for the High Resolution Dynamics Limb Sounder (HIRDLS) Level 0 to Level 1 (L1) Processor, hereby known as L1 Processor. This design specification will extend the architecture of L1 Processor at a level of detail sufficient to facilitate implementation. It is assumed that the reader of this document has already read and understood the documented architectural plan 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
Center for Limb Atmospheric Sounding
3450 Mitchell Lane, Bldg. FL-0
Boulder, CO 80301

EOS

The Design of the HIRDLS Level 0 – Level 1 Processor

Charles Cavanaugh

Table of Contents

Table of Contents (continued)

Table of Contents (continued)

List of Figures

List of Figures (continued)

List of Abstractions

List of Abstractions (continued)

1 Document Purpose and Goal

The purpose of this document is to create a design specification for the High Resolution Dynamics Limb Sounder (HIRDLS) Level 0 to Level 1 (L1) Processor, hereby known as L1 Processor. This design specification will extend the architecture of L1 Processor at a level of detail sufficient to facilitate implementation. It is assumed that the reader of this document has already read and understood the documented architectural plan of L1 Processor.

The goal of this document is to create a design that correctly models the overviewed task, and lays out a plan that distributes system intelligence as evenly as possible, is easy to understand and implement, and easy to extend or revise, if or when necessary in the future.

2 Design Notation and Goal

The notation used in this document will be the Unified Modeling Language (UML). However, the method used to specify each abstraction will be based on the work of Ward Cunningham and Kent Beck, and detailed in Designing Object-Oriented Software (Wirfs-Brock, et.al, 1990). This method places a high degree of importance on finding each abstraction's responsibilities and collaborations. C++ syntax will be used to specify detailed abstraction information (such as public contract interface). The goal of the design is to create the simplest system to correctly accomplish the task. Though effort will be made to make abstractions reusable throughout the system, no effort will be made to make abstractions reusable outside of the system, i.e. no functionality or data will exist that is not used by the system, unless that functionality makes for a more extensible system.

3 Design Considerations

As first mentioned in the L1 Processor Requirements document, L1 Processor is a stand-alone, non-graphical, non-embedded scientific application, and as such, resource usage has become a primary design consideration. Through negotiations with the HIRDLS Program Manager and HIRDLS Data Manager, available data storage size is not a concern. Application memory size, though, is a concern, and efficient usage of memory must be planned. The L1 Processor memory management plan is twofold: 1) create a design that minimizes memory complexity; and 2) utilize tools during implementation to help find memory use flaws. The latter part of the plan is beyond the scope of this document. The former part has four approaches: 1) favor stack memory over heap memory; 2) implement allocation failure recovery; 3) reduce the scope of heap allocated memory; and 4) force safe memory parameter passing. Though the last three approaches are more implementation issues than design issues, all four approaches are addressed further in this section.

3.1 Favor Stack Memory

Data created with stack memory has the benefit of being unwound when the data's scope is terminated. Heap memory persists until explicitly terminated, and is therefore highly prone to leaking. If an abstraction is needed within a method, prefer to allocate it on the stack. If the method is called many, many times, consider having the needed abstraction as a private data member of the employing abstraction.

3.2 Recover From Failure

Stack memory use, though favored over heap memory use, will not be exclusive to a system with the size and complexity of the HIRDLS L1 Processor. Every time an attempt is made to allocate heap memory, the status of the allocation must be checked before using the memory. If there was a failure, the system should recover in a consistent and graceful way. Immediately conveying failure information to the system user and exiting from the system is acceptable, and preferred.

3.3 Reduce Memory Scope

Again, there will be times when using heap memory is unavoidable (as when creating a vector of abstractions – vectorizing the address of the abstraction is much more efficient than vectorizing the entire abstraction). The scope of the accessibility of the memory must be reduced to its lowest practical point, but never higher than abstraction-wide scope. That is, the most preferred way to use heap memory is to allocate it, use it and destroy it within the same abstraction method. When it is not possible to destroy it within the same method as allocated (as with the example above), the memory must be destroyed in another method *of the same abstraction*.

3.4 Force Safe Passing

This aspect of the plan disallows destruction of memory passed into a method via the parameter list. If it is necessary to pass allocated memory to a method via the parameter list, that memory *must still exist in its original form* when the method terminates. The memory passed in is owned by another method, and it is incumbent on the owning method to destroy the memory, and any changes to the memory, or aggregation of the memory, is disallowed by the called method. If the language supports compile-time checking (such as forcing constant pointers), this must be used to verify the safeness of parameter memory.

4 Design Representations

The L1 Processor Architecture document introduced various packages from which L1 Processor will be built. Those packages will be enumerated in more detail in this document. Included in the detail will be the various abstractions that make up a given package. As mentioned in Section 2, UML notation will be used to show a package's internal mechanisms, including abstractions, dependencies, collaborations, responsibilities and contracts.

4.1 Packages

The packages that comprise L1 Processor are logical encapsulations of a collection of abstractions that are homogeneous in purpose, with that purpose reflected in the package name. Packages are the building blocks for a system, and are the "whats" in that system. Figure 1 shows the hierarchy of the L1 Processor packages introduced in the L1 Processor Architecture document (though the names have been altered to fit this document's notation that packages have a singular name). The goal of that document was to identify the "whats" or packages necessary to fulfill the requirements of L1 Processor. The goal of this document is to identify the "hows" of each package. The Diagnostics package is at the lowest level, and is accessible to all other packages. The Service package, which is to present to L1 Processor packages a simplified front-end to SDP Toolkit, is a level higher than Diagnostics (which means Service can use Diagnostics), and is accessible to all other packages. The File package is introduced here, and its purpose is to provide a building block for file input/output. The other packages shown in Figure 1 have only explicit accessibility to those packages to which each package's emanating arrows point.

4.2 Abstractions

Three different types of abstractions are used in L1 Processor: process, control and data. Where packages are homogeneous *in purpose*, process abstractions are homogeneous *in action*, control abstractions are homogeneous *in idea*, and data abstractions are homogeneous *in content*. Process abstractions are about action, so they have "actiony" names, such as Data Transformer or File Writer. Control abstractions are about idea, so they have "ideay" names, such as PCF Service or HIRDLS1 File. Data abstractions are about content, so they have "contenty" names, such as Sequenced Data or HIR0SCI Packet. All abstractions, regardless of type, are to fully encapsulate all functionality needed to accomplish their specified task, and to present to L1 Processor the simplest interface possible. Fully encapsulate does not mean an abstraction must be totally self-contained and needing no other abstractions. Fully encapsulate means that no other abstraction need know how it does its job, only that it does its job.

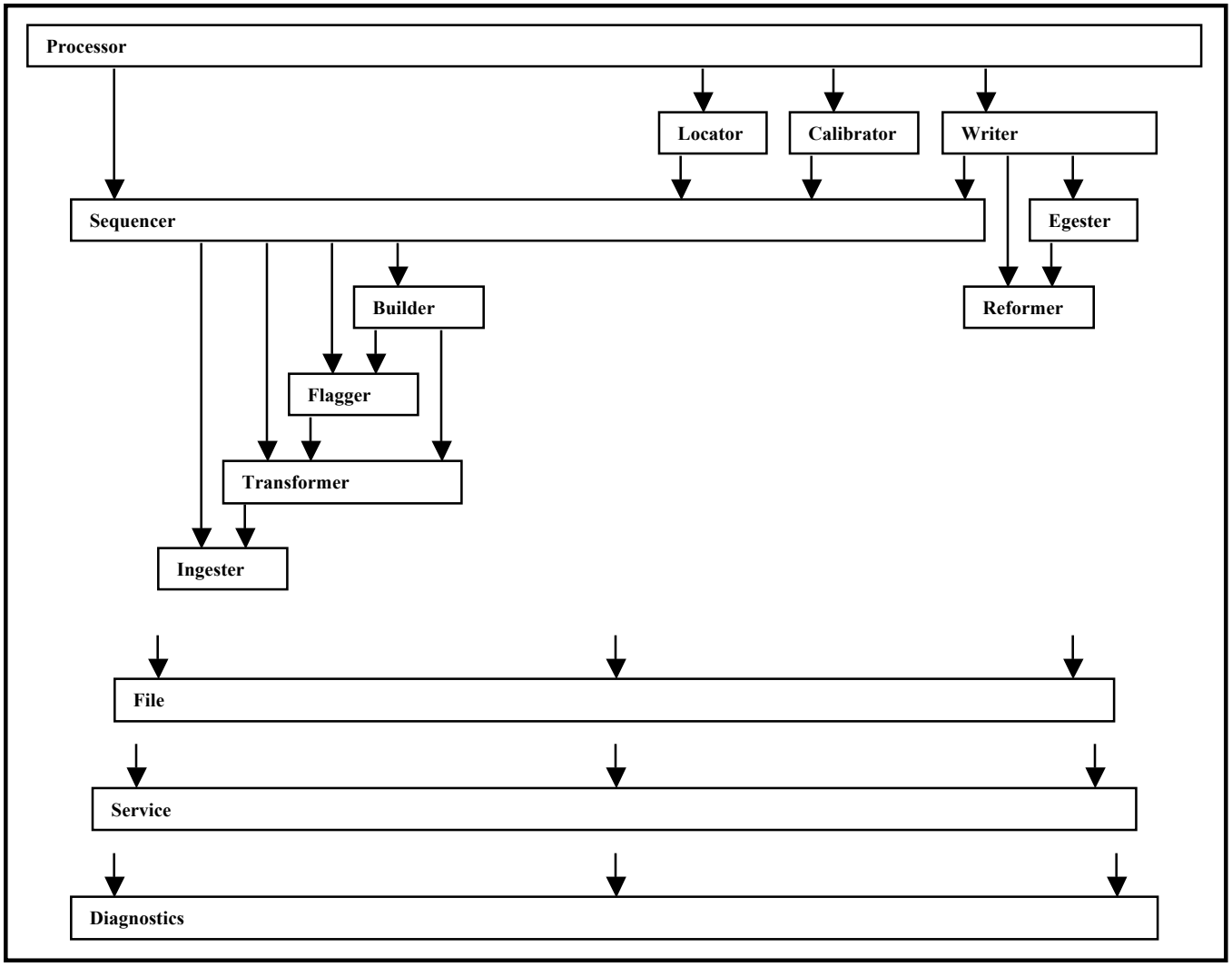

Figure 1 L1 Processor Hierarchy of Packages

4.3 Dependencies

A package is dependent on another package when, obviously, the employing package needs something from the employed package. The exception to this is when two packages communicate via a data abstraction, which then makes the two packages dependent on the data abstraction. Because this approach minimizes inter-package dependencies within the system, or at the very least localizes the dependencies, communicating via data abstractions is the preferred way to handle inter-package dependencies in L1 Processor. This is in agreement with the desire for *low coupling* in a system[1]. In a complex system, unplanned dependencies can get circular and unmaintainable, and one of the goals of L1 Processor is to maximize maintainability. The L1 Processor Architecture document shows that packages communicate with each other via data aggregations, and therefore L1 Processor packages can be independently implemented and tested, making the system less circular and more maintainable. In the case(s) where a package encapsulates other packages, inter-package dependencies cannot be eliminated, but can be minimized by planning no inter-package dependencies amongst the encapsulated packages. In this document, abstraction dependency and hierarchy figures are interchangeable (similar to the packages in Figure 1).

[1] W. Stevens, G. Myers, L. Constantine, "Structured Design", IBM Systems Journal, 13 (2), 115-139, 1974.

4.4 Collaborations

Identifying and planning inter-package and inter-abstraction collaborations is one of the two important tasks for this L1 Processor Design document (responsibility is the other important task, and that will be detailed in Section 4.5). Section 4.3 has already begun the discussion on collaborations, because collaborations, in the package or abstraction sense, are one-way streets and, therefore, dependencies arise. In the world of human interaction, two-way collaborations are considered desirable, but in the logical, computer world, two-way collaborations are impossible, as you must first define a "thing" (package, abstraction, data, etc.) before some other "thing" can make use of it. Collaboration figures will be used by this document to show how abstractions interact to accomplish their respective tasks. These figures will employ UML notation to show aggregation, inheritance and simple collaboration (dependency without encapsulation).

4.5 Responsibilities

As first mentioned in Sections 2 and 4.4, identifying a package's or abstraction's responsibilities is one of the two responsibilities of this document. In much the same way the members of a software team have their own responsibilities, so too do packages and abstractions in a system. It is important to note that abstractions and packages come from responsibilities, and not vice-versa. As software system construction starts with a Requirements document, package and abstraction identification starts with responsibilities. The L1 Processor Requirements and Architecture documents have already begun the process, and have identified numerous packages to carry out the system's distributed responsibilities. This document will extend the depth of those responsibilities and identify the abstractions that will need to be created to fulfill the newer, and more focused, responsibilities. The responsibilities of an abstraction will be enumerated in that abstraction's respective section, but will not be displayed in any figure.

4.6 Contracts

Up until now, the L1 Processor documents have expounded on finding the "whats" in the system. The contracts of an abstraction detail the "hows". And where packages and abstractions are the nouns, contracts are the verbs. This document will enumerate the contracts for the abstractions, in their respective sections, in L1 Processor. As mentioned in Section 4.3, we have the goal of minimizing inter-package and inter-abstraction dependencies, and using data abstractions for communication does that. With many abstractions, specifically the process and control abstractions, the contracts will detail that dependency minimization. With data abstractions, however, it is often the case that this method is very inefficient, and we would have to create a data abstraction that exists solely to update, for instance, another data abstraction. We therefore minimize dependency on these data abstractions by creating contracts that use only language primitives. A data abstraction, by definition, encapsulates data, not process or control, so there are no "internal workings" that we would want to hide from the system.

Contracts have the obligation to detail how they handle failure. Most often, this will involve returning a status Boolean to detail if they were able to successfully (true) or unsuccessfully (false) carry out their responsibilities. How to represent success and failure is dependent on the implementation language, but must be consistent throughout the system. If the language has a Boolean primitive, using it is preferred over the system creating its own. In cases where failure within a contract is catastrophic to the system (e.g., a memory creation call is unsuccessful) and the system needs to abort processing, the contract must specify that it has system abortion authorization (noted as 'SAA' in the contract tables, all of which are listed in Appendix A). Contracts that do have SAA might still return status Booleans, as the contract could still fail, though not catastrophically.

5 Design Methodology

As the L1 Processor documentation has progressed from requirements to architecture to design, we have been employing a top-down methodology to further decompose the system. These documents have also talked about elements of L1 Processor being "building blocks" upon which to build other elements, which is the methodology used in bottom-up synthesis. Figure 1 shows three "building block" packages (File, Service, Diagnostic), while showing all other packages in L1 Processor, which were derived from top-down analysis. The File and Service packages exist to reduce complexity in the system, and are

tasked to contain no more "usefulness" than is needed by L1 Processor. The Diagnostic package exists to pass meaningful information from the system to the user, and will most likely grow or shrink long after the first production version of L1 Processor has been delivered. Therefore, while the majority of the elements of L1 Processor are derived using a top-down decomposition, an eye is still open to find where elemental building blocks can be best utilized.

6 Package Enumeration

L1 Processor package designs will be enumerated, in bottom-up order, in the following sections. The intent of enumerating in this order is to have a package or abstraction well understood before inserting it into the workings of other packages and/or abstractions. The remainder of this document is left to the detail design of each previously introduced package.

7 Diagnostics Package

The Diagnostics package has the responsibility to provide the system a consistent means to report system diagnostics, including termination information. As discussed in Section 4.1, this package is the lowest level package in the system and does not depend on any other package. This non-dependency is a design constraint detailed in the Section 7.1. Figure 2 shows the hierarchy of the abstractions in the package. Each abstraction in the package is detailed further in this section.

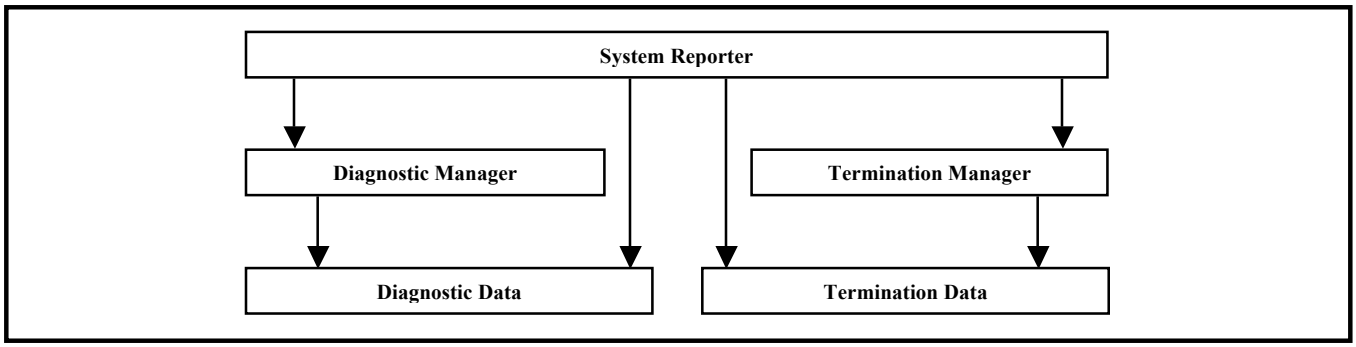

Figure 2 Diagnostics Package Hierarchy

7.1 System Reporter Abstraction

System Reporter is a control abstraction, and has the responsibility to accumulate system diagnostic and termination information, and generate a standardized summary report. To fulfill this responsibility, this abstraction collaborates with Diagnostic Manager, Termination Manager, Diagnostic Data and Termination Data, and presents an interface of Add and GetReporter contracts, as shown in Figure 3. The Add contracts allow the system to add diagnostic and termination information to the report. For the manager abstractions to work correctly, System Reporter must aggregate them and keep them persistent during its lifetime. This abstraction, in turn, must also be persistent to work correctly. This abstraction is specified to be globally accessible and fail-safe. Fail-safe means the report must be generated, even if the process terminates abnormally, and output to some device (file or screen), but employ no memory creation functionality or outside subsystem dependencies. This specification also applies to the abstractions with which System Reporter aggregates. The intent of this report is to give the operator some indication of what happened during a system run, so if this report is not generated, the run will have failed. There must be exactly one instance of this abstraction in the system, and therefore it is specified that this abstraction be a *Singleton creational pattern*[2], and the GetReporter contract is to return that instance. The classic implementation of a Singleton has the abstraction encapsulating a pointer to itself, but this breaks this abstraction's "no memory creation" requirement. Making the pointer a global stack pointer (guaranteed to be unwound off the stack at program termination), rather than a heap pointer, is allowable in this one exception. The public contracts of this abstraction

[2] As detailed in Design Patterns, Elements of Reusable Object-Oriented Software by Gamma, et.al., 1995

must use only primitives (due to the no subsystem dependency requirement), and must not "fail" in the sense that processing should stop. Note that there is no contract to generate the status report. The report will be generated when the abstraction is destroyed.

7.2 Diagnostic Manager Abstraction

Diagnostic Manager is a control abstraction, and has the responsibility to accumulate and retrieve the reported system diagnostics. To fulfill this responsibility, this abstraction collaborates with Diagnostic Data, and presents an interface of Add and Retrieve contracts, as shown in Figure 4. The Add contract allows System Reporter to add diagnostic information to the report, and the Retrieve contract allows System Reporter to retrieve the added diagnostic information. Diagnostic Manager has the same "no memory creation" requirements as System Reporter, and therefore must aggregate a constant number of Diagnostic Data abstractions, and keep them persistent during its lifetime. This abstraction needs to be persistent to work correctly.

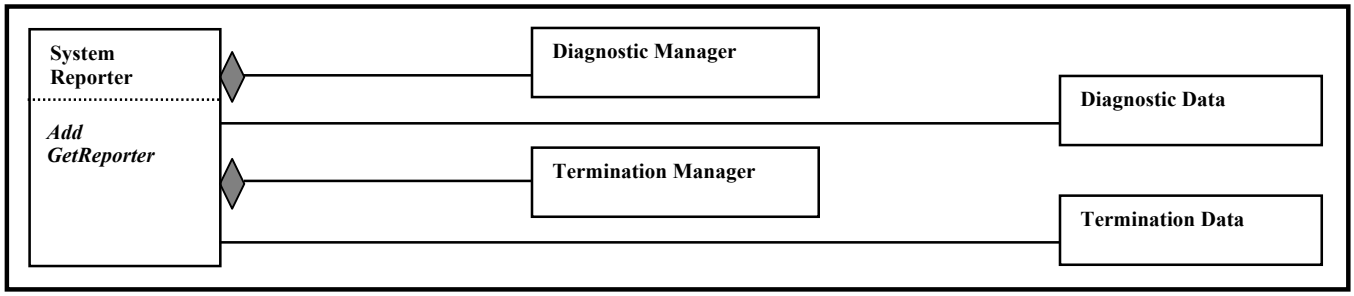

Figure 3 System Reporter Abstraction

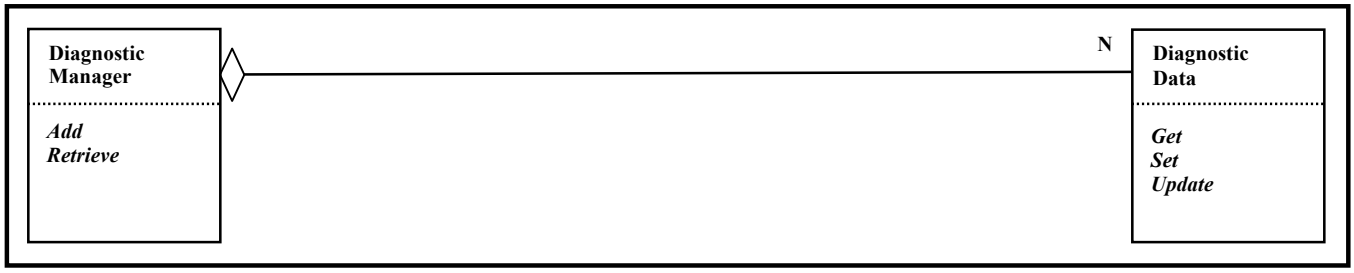

Figure 4 Diagnostic Manager and Diagnostic Data Abstractions

7.3 Diagnostic Data Abstraction

Diagnostic Data is a data abstraction, and has the responsibility to encapsulate information specific to one system diagnostic. To fulfill this responsibility, this abstraction presents an interface of Get, Set and Update contracts, as shown in Figure 4. The copy constructor, assignment operator, or Set contract can be used by Diagnostic Manager to initialize the abstraction. The Get contract returns data encapsulated by the abstraction. The Update contract updates the occurrence counter of an initialized abstraction.

7.4 Termination Manager Abstraction

Termination Manager is a control abstraction, and has the responsibility to store and retrieve the system termination. To fulfill this responsibility, this abstraction collaborates with Termination Data, and presents an interface of Add and Retrieve contracts, as shown in Figure 5. The Add contract allows System Reporter to add termination information to the report, and the Retrieve contract allows System Reporter to retrieve the added termination information. Termination Manager has the same "no memory creation" requirements as System Reporter, and therefore must aggregate one Termination Data abstraction (since termination is binary – either normal or abnormal), and keep it persistent during its lifetime. This abstraction needs to be persistent to work correctly.

7.5 Termination Data Abstraction

Termination Data is a data abstraction, and has the responsibility to encapsulate information specific to the system termination status. To fulfill this responsibility, this abstraction presents an interface of Get and Set contracts, as shown in Figure 5. The copy constructor, assignment operator, or Set contract can be used by Termination Manager to initialize the abstraction. The Get contract returns data encapsulated by the abstraction.

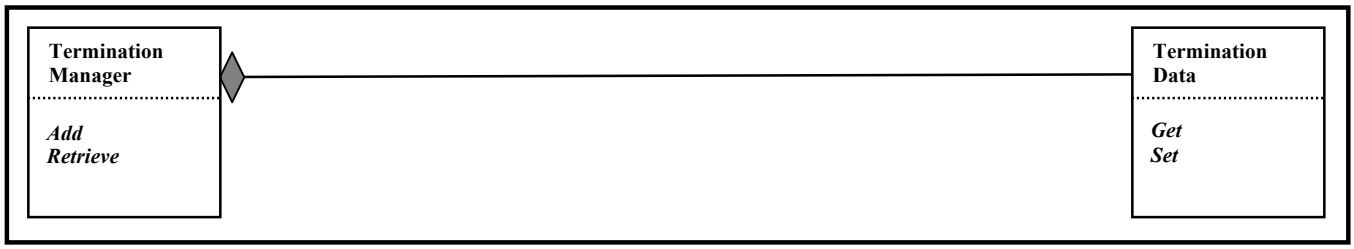

Figure 5 Termination Manager and Termination Data Abstractions

8 Service Package

The Service package has the responsibility to encapsulate all potentially system-wide service abstractions necessary to fulfill the system requirements. If a service abstraction is specific to one package, then it belongs in that package, otherwise it belongs in this package. As discussed in Section 4.1, this package is the second lowest package in the system, and can depend on the Diagnostics package. Figure 6 shows the hierarchy of the abstractions in the package. Each abstraction in the package is detailed further in this section.

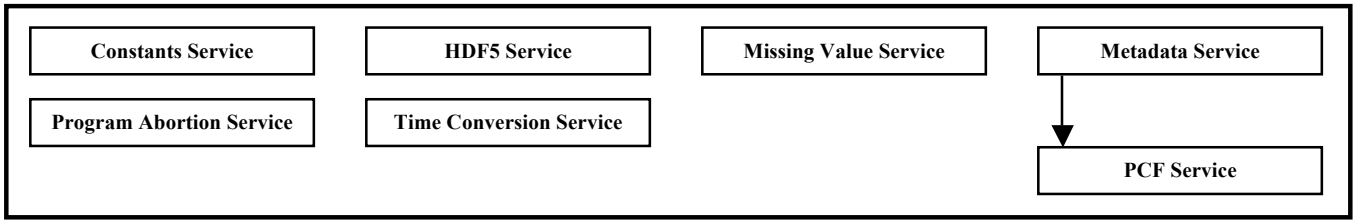

Figure 6 Service Package Hierarchy

8.1 Constants Service Abstraction

Constants Service is a control abstraction, and has the responsibility to provide a single access point to instrument-specific constants. To fulfill this responsibility, this abstraction presents an interface of size constants and contracts to test the

validity of some of the size indices, as show in Figure 7. This abstraction is not meant to be instantiated, but instead provide non-dynamic data into the global space. Persistency is not an issue.

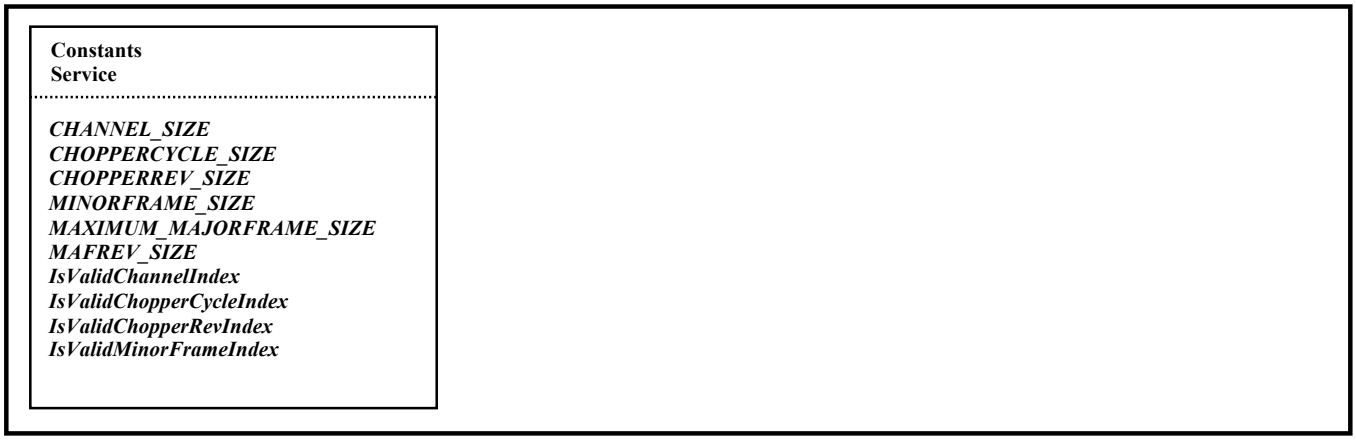

Figure 7 Constants Service Abstraction

8.2 HDF5 Service Abstraction

HDF5 Service is a control abstraction, and has the responsibility to provide simple and coherent access to HDF5 services provided via the SDP Toolkit. To fulfill this responsibility, this abstraction collaborates with the SDP Toolkit, and presents an interface of many file and field access contracts, as shown in Figure 8. The CreateFile contract creates a new HDF5 file, and the OpenFile contract opens an existing HDF5 file. The CloseFile contract closes an HDF5 file. The CreateSwath contract creates a new swath within the newly created HDF5 file, and the OpenSwath contract opens an existing swath. The CloseSwath contract closes access to a swath. The DefineDimension contract defines new dimensions within a newly created HDF5 file, the DefineField contracts provide multiple ways to define data and geolocation fields within a newly created HDF5 file, and DefineCompression defines the compression characteristics of a newly defined data or geolocation field. The GetDimensionSize contract returns a defined dimension size, and the GetFieldFillValue contract returns the fill value of an already defined field. The WriteField contract writes data to a field, and the WriteAttribute contract writes a file-level attribute. The ReadField contract reads a field. This abstraction need not be persistent to work correctly, though this abstraction does return data that needs to stay persistent to work correctly. Therefore, the abstraction that uses this abstraction must either aggregate the returned data, or begin and end service access within a persistent scope.

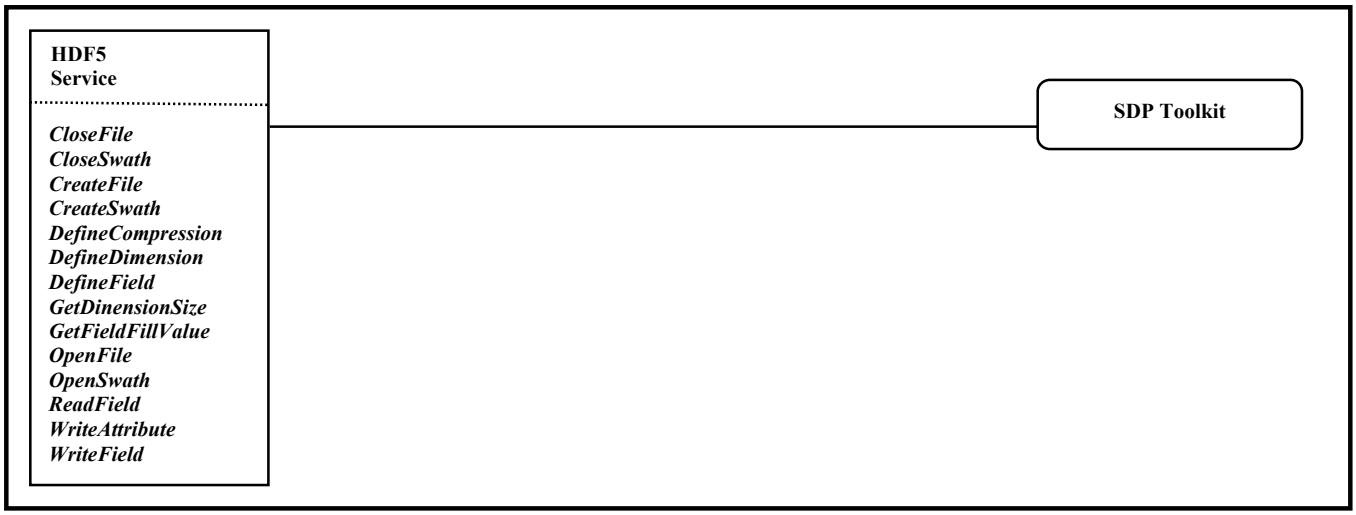

Figure 8 HDF5 Service Abstraction

8.3 Missing Value Service Abstraction

Missing Value Service is a control abstraction, and has the responsibility to provide a single access point for missing value representation and comparison checking. To fulfill this responsibility, this abstraction presents an interface of missing value retrieval contracts and missing value comparison contracts, as shown in Figure 9. The Get*MissingValue contracts all return the primitive-specific representation of system-wide missing value, and the IsMissingValue contracts tests whether a value is the missing value. This abstraction need not be persistent to work correctly.

8.4 Program Abortion Service Abstraction

Program Abortion Service is a control abstraction, and has the responsibility to provide a consistent means to abort the program. To fulfill this responsibility, this abstraction collaborates with System Reporter, and presents an interface of Abort contracts, as shown in Figure 10. These contracts add an abnormal termination notice to the Report Generator abstraction, and then abort the system with a failure indication. This abstraction need not be persistent to work correctly. This abstraction has not been shown in any collaboration figures, but of course is available to any abstraction that needs this access.

Figure 9 Missing Value Service Abstraction

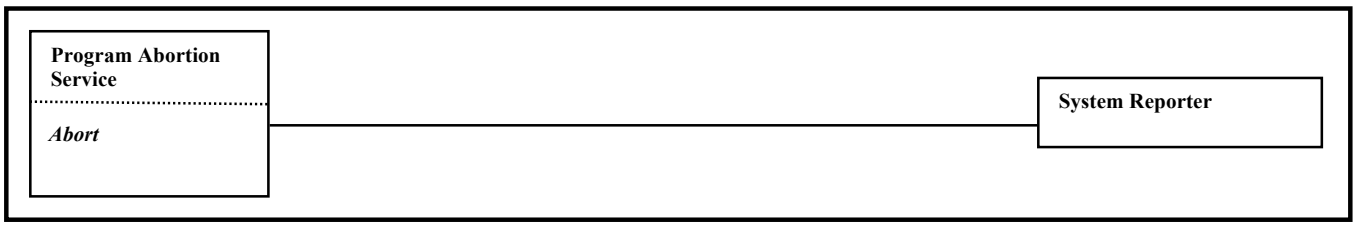

Figure 10 Program Abortion Service Abstraction

8.5 Time Conversion Service Abstraction

Time Conversion Service is a control abstraction, and has the responsibility to provide simple time format conversion service. To fulfill this responsibility, this abstraction collaborates with the SDP Toolkit, and presents an interface of conversion contracts, as shown in Figure 11. Each contract does as its name specifies, converting a value from one time format to another. This abstraction need not be persistent to work correctly.

Figure 11 Time Conversion Service Abstraction

8.6 PCF Service Abstraction

PCF Service is a control abstraction, and has the responsibility to provide simple and coherent access to process control file (PCF) access services provided via the SDP Toolkit. To fulfill this responsibility, this abstraction collaborates with the SDP Toolkit, and presents an interface of GetFilename and GetParameter contracts, as shown in Figure 12. The GetFilename contracts provide multiple ways to retrieve the name of a file listed in the PCF. The GetParameter contract provides a means of retrieving an input parameter listed in the PCF. This abstraction need not be persistent to work correctly.

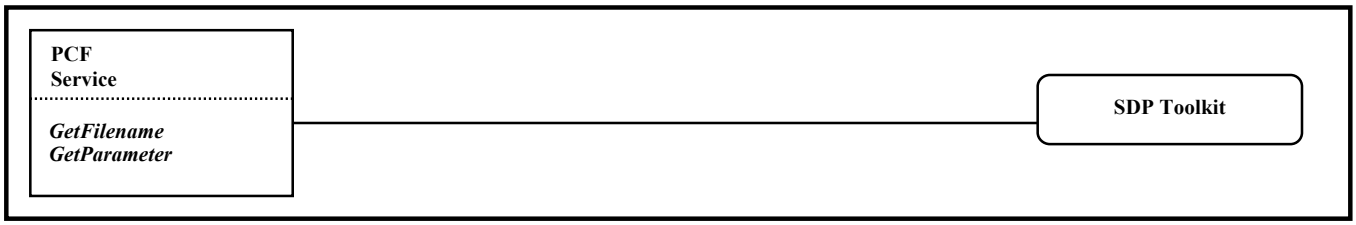

Figure 12 PCF Service Abstraction

8.7 Metadata Service Abstraction

Metadata Service is a control abstraction, and has the responsibility to provide simple and coherent access to ECS metadata services provided via the SDP Toolkit. To fulfill this responsibility, this abstraction collaborates with PCF Service and the SDP Toolkit, and presents an interface of Set and Write contracts, as shown in Figure 13. The Set contracts allow setting of ECS metadata parameters, and the Write contract writes the ECS metadata to the file with which it is connected. ECS service access needs to be initialized and terminated, but that should happen upon abstraction instantiation and destruction, respectively. This abstraction needs to be persistent to work correctly.

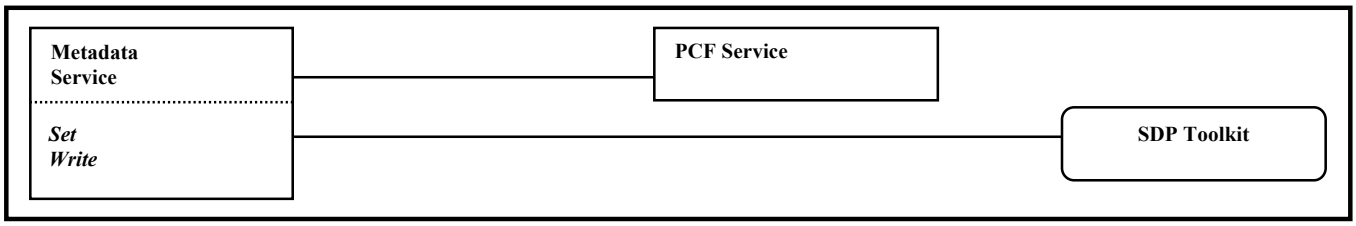

Figure 13 Metadata Service Abstraction

9 File Package

The File package has the responsibility to encapsulate all abstractions necessary to provide the system a consistent means to interact with data files. As discussed in Section 4.1, this package is the third lowest package in the system, and can depend on the Diagnostics and Service packages. Figure 14 shows the hierarchy of the abstractions in the package. Each abstraction in the package is detailed further in this section.

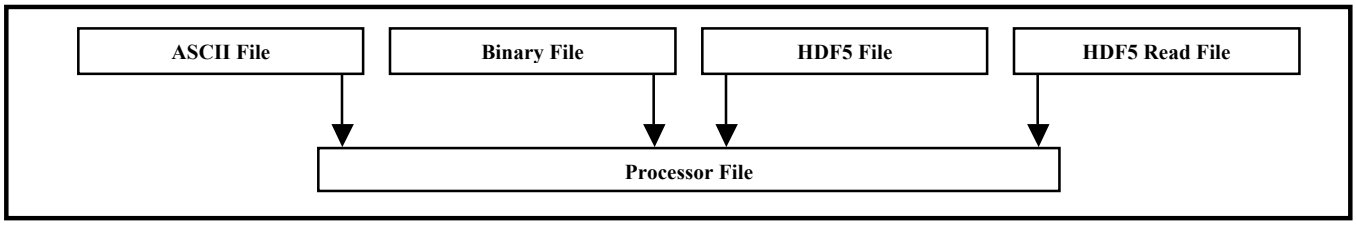

Figure 14 File Package Hierarchy

9.1 Processor File Abstraction

Processor File is a control abstraction, and has the responsibility to manage low-level access to all files within the system, but only as a pure virtual abstraction intended to be used as a base for all file abstractions in the system. To fulfill this responsibility, this abstraction collaborates with PCF Service, and presents an interface of IsValid, GetLogical and GetName contracts, as shown in Figure 15. The IsValid contract determines if the file is valid, he GetLogical contract returns the logical ID of the file, and the GetName contract returns the name of the file. This abstraction does not handle opening and closing, as those are specific to a type of file. For this abstraction to be used in a realistic manner, the abstraction that is derived from it needs to be persistent, unless this abstraction is used solely for the purpose of testing the validity of a file.

Figure 15 Processor File Abstraction

9.2 ASCII File Abstraction

ASCII File is a control abstraction, and has the responsibility to manage access to a read-only, sequential-access ASCII file, but only as a pure virtual abstraction intended to be used as a base for file abstractions that model ASCII files. To fulfill this responsibility, this abstraction collaborates with Processor File, and presents an interface of Open, Close, Read and GetToken contracts, as shown in Figure 16. The Open contract opens the existing file for reading, the Close contract closes the opened file, and the Read contract reads the next line in the opened file. The GetToken contracts are provided to help parse a file line in the usual way: to extract a particular token from the line. For this abstraction to work correctly across multiple Read calls, the abstraction that is derived from this abstraction needs to be persistent.

Figure 16 ASCII File Abstraction

9.3 Binary File Abstraction

Binary File is a control abstraction, and has the responsibility to manage access to a read-only, sequential-access binary file, but only as a pure virtual abstraction intended to be used as a base for file abstractions that model binary files. To fulfill this responsibility, this abstraction collaborates with Processor File, and presents an interface of Open, Close and Read contracts, as shown in Figure 17. The Open contract opens the existing file for reading, the Close contract closes the opened file, and the Read contract reads the next specified chunk size of data in the opened file. For this abstraction to work correctly across multiple Read calls, the abstraction that is derived from this abstraction needs to be persistent.

Figure 17 Binary File Abstraction

9.4 HDF5 File Abstraction

HDF5 File is a control abstraction, and has the responsibility to manage access to a write-only, HDF5-formatted file, but only as a pure virtual abstraction intended to be used as a base for file abstractions that model HDF5 files. To fulfill this responsibility, this abstraction collaborates with Processor File and HDF5 Service, and presents an interface of creating, closing, defining and writing contracts, as shown in Figure 18. The Create contract creates a new HDF5 file, and the Close contract closes a newly created HDF5 file. The DefineDimension contract is used to define dimensions of a newly created HDF5 file, and the DefineField contracts are used to define fields in a newly created HDF5 file. The WriteField contract writes a field's data to the newly created HDF5 file, and the WriteAttribute writes a file-level attribute to a newly created HDF5 file. For this abstraction to work correctly across its multiple calls, the abstraction that is derived from this abstraction needs to be persistent.

9.5 HDF5 Read File Abstraction

HDF5 Read File is a control abstraction, and has the responsibility to manage access to a read-only, HDF5-formatted file, but only as a pure virtual abstraction intended to be used as a base for file abstractions that model HDF5 files. To fulfill this responsibility, this abstraction collaborates with Processor File and HDF5 Service, and presents an interface of opening, closing and reading contracts, as shown in Figure 19. The Open contract opens an existing HDF5 file, and the Close contract closes an opened HDF5 file. The GetDimensionSize contract retrieves the size of a dimension in an opened HDF5 file. The

ReadField contracts read fields of specified data types. For this abstraction to work correctly across its multiple calls, the abstraction that is derived from this abstraction needs to be persistent.

Figure 18 HDF5 File Abstraction

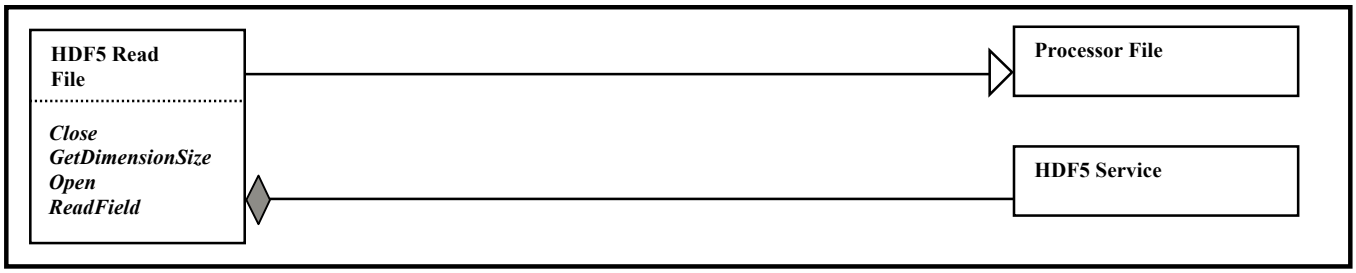

Figure 19 HDF5 Read File Abstraction

10 Ingester Package

The Ingester package has the responsibility to encapsulate all abstractions necessary to provide the system a means to ingest packets from the HIRDLS L0 file(s). As shown in Figure 1, this package is used by the Sequencer and Transformer packages, and has access to the File, Service and Diagnostics packages. Figure 20 shows the hierarchy of the abstractions in the package. Each abstraction in the package is detailed further in this section.

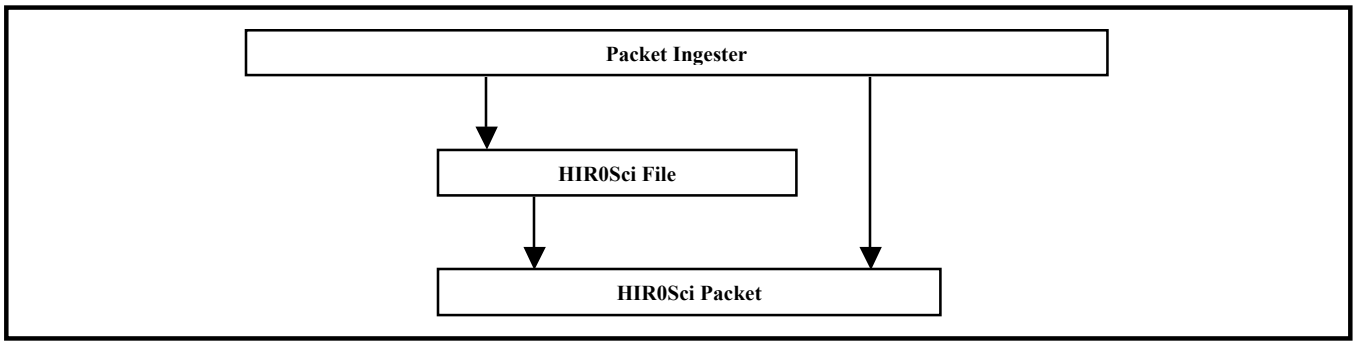

Figure 20 Ingester Package Hierarchy

10.1 Packet Ingester Abstraction

Packet Ingester is a process abstraction, and has the responsibility to ingest packets from the HIR0Sci file(s), and return those raw packets to the system. To fulfill this responsibility, this abstraction collaborates with HIR0Sci File and HIR0Sci Packet, and presents an interface of one Ingest contract, as shown in Figure 21. The Ingest contract is to return the *next* packet in the file(s) stream. The exact mechanisms for how Packet Ingester fulfills its responsibility is left for implementation, whether it reads in all packets from all HIR0Sci File(s) upon instantiation, or if it reads in one packet at a time per call, or any other option. What is specified here is that Packet Ingester must manage the raw packets in such a manner as to seamlessly return the next packet, in sequence, and return a Boolean when it can not return the next packet, whether from error or from a lack of further packets. No matter how Packet Ingester manages its responsibility, this abstraction is specified to necessitate persistence to work correctly.

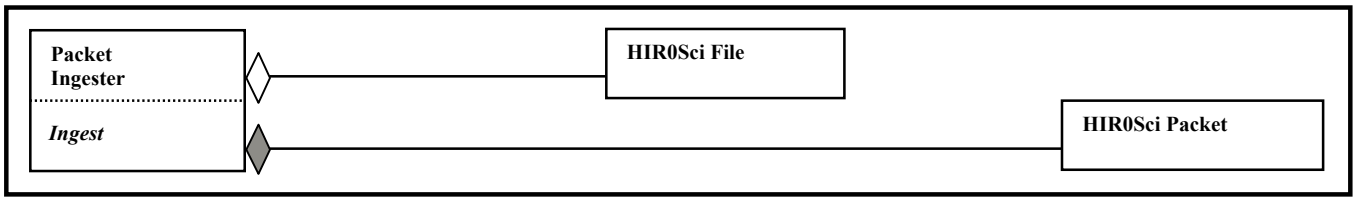

Figure 21 Packet Ingester Abstraction

10.2 HIR0Sci File Abstraction

HIR0Sci File is a control abstraction, and has the responsibility to manage all access to a HIR0Sci file. To fulfill this responsibility, this abstraction collaborates with Binary File and HIR0Sci Packet, and presents an interface of one GetFile contract and one GetNextPacket contract, as shown in Figure 22. The GetFile contract returns an instance of a HIR0Sci File abstraction. At this time, this instance is not considered a Singleton[3], as management of the HIR0Sci file(s) is left for implementation, but the contract for returning an instance of this abstraction is left to look like a Singleton, in the case that that is what is implemented, or changed to during testing and/or maintenance. The GetNextPacket contract returns the next packet in sequence. This contract is to return a Boolean status to the caller, indicating if it was able to retrieve the next packet or not. This abstraction needs to be kept persistent to work correctly, except in the case that the instantiated file is opened, read completely, and closed, in one structural calling sequence.

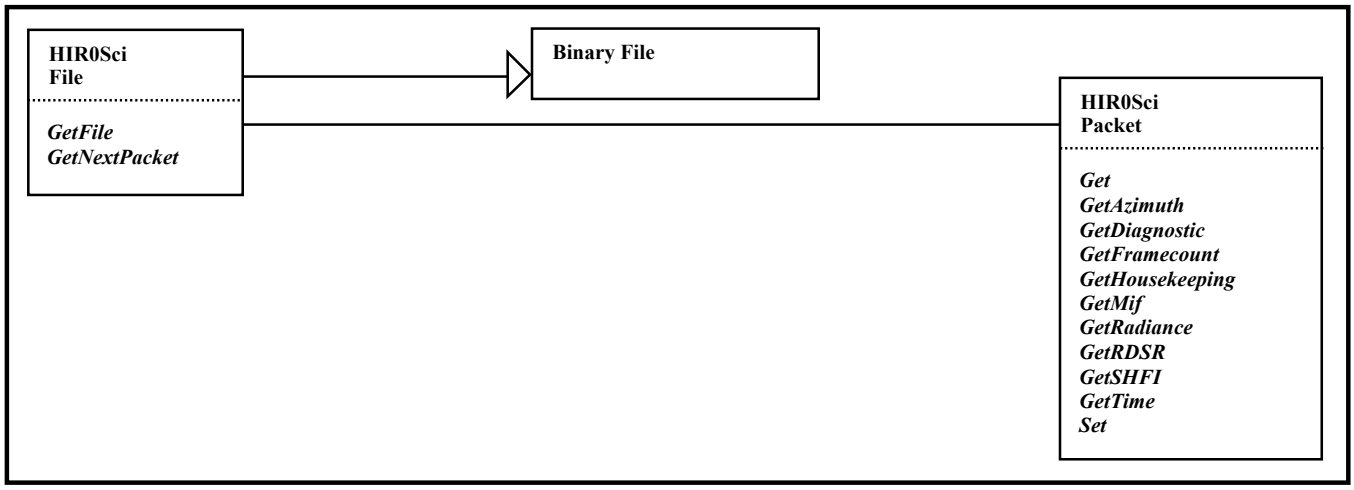

Figure 22 HIR0Sci File and HIR0Sci Packet Abstractions

[3] See Section 7.1

10.3 HIR0Sci Packet Abstraction

HIR0Sci Packet is a data abstraction, and has the responsibility to manage access to the raw packet data extracted from a HIR0Sci File. To fulfill this responsibility, this abstraction presents an interface of one Set contract and various Get contracts, as shown in Figure 22. This abstraction also presents a constant value denoting the size of the data packet it contains. The copy constructor, assignment operator, or Set contract can be used by HIR0Sci File to initialize the abstraction. The various Get contracts enumeration is, at this time, not considered exhaustive. It may certainly shrink or grow. These contracts provide bit manipulation information transformation, i.e., they encapsulate the information necessary to decom bits into data primitive units. For example, the GetRDSR contract retrieves the RDSR value to the caller, encapsulating the knowledge of where that information is in the packet, and how to decom it appropriately.

11 Transformer Package

The Transformer package has the responsibility to encapsulate all abstractions necessary to provide the system a means to transform raw Level 0 data packets into system-usable data packets. As shown in Figure 1, this package is used by the Sequencer, Flagger and Builder packages, and has access to the Ingester, File, Service and Diagnostics packages. Figure 23 shows the hierarchy of the abstractions and sub-packages in the package. Each abstraction and sub-package in the package is detailed further in this section.

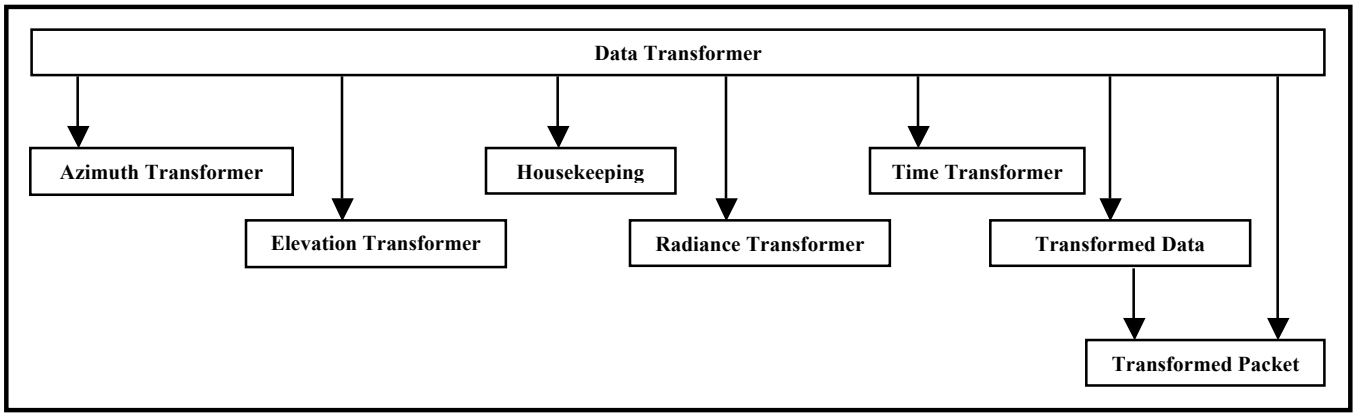

Figure 23 Transformer Package Hierarchy

11.1 Housekeeping Package

The Housekeeping package has the responsibility to encapsulate all abstractions necessary to provide the system a means to transform raw housekeeping data into system-usable units. As shown in Figure 23, this package is used by the Data Transformer package. Figure 24 shows the hierarchy of the abstractions in the package. Each abstraction in the package is detailed further in this section.

11.1.1 Housekeeping Transformer Abstraction

Housekeeping Transformer is a process abstraction, and has the responsibility to transform the housekeeping data in a HIR0Sci Packet. To fulfill this responsibility, this abstraction collaborates with HIR0Sci Packet, Housekeeping File and Housekeeping Data, and presents an interface of one Transform contract, as shown in Figure 25. The Transform contract reads the housekeeping data from HIR0Sci Packet, transforms the data, and then returns the transformed data and a status Boolean to Data Transformer. In order for the Transform contract to work efficiently, this abstraction, upon instantiation, needs to read in and store the data from the system's housekeeping file(s). When the Transform contract is called, the abstraction searches through the stored data and finds the appropriate data set to use to extract and transform the

housekeeping data in HIR0Sci Packet, and return the transformed data to the system. Therefore, in order to work efficiently, this abstraction needs to be persistent.

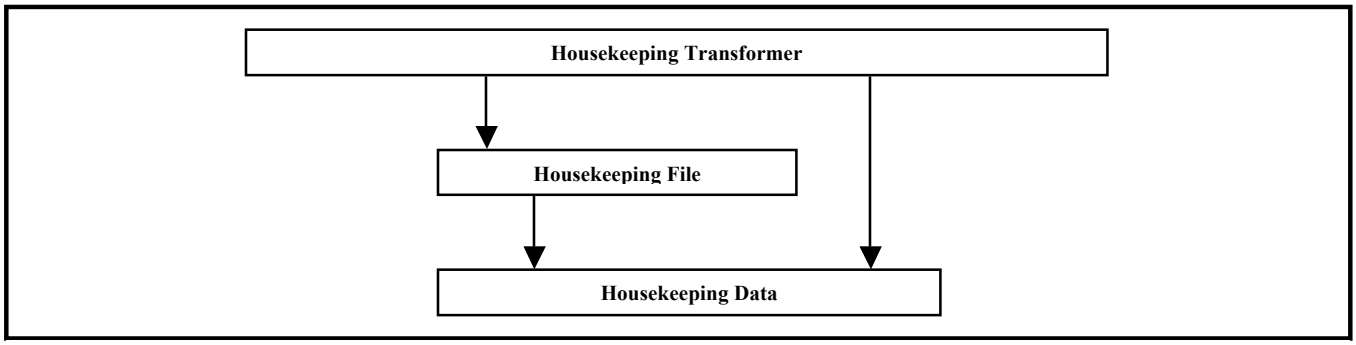

Figure 24 Housekeeping Package Hierarchy

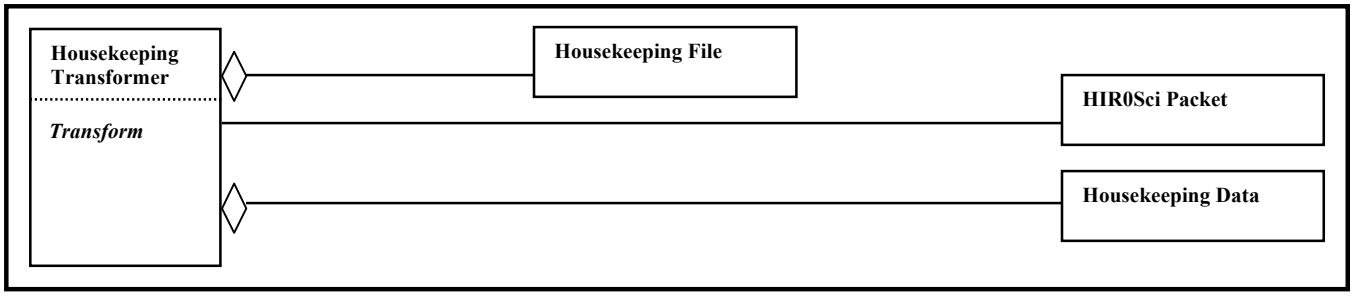

Figure 25 Housekeeping Transformer Abstraction

11.1.2 Housekeeping File Abstraction

Housekeeping File is a control abstraction, and has the responsibility to manage all access to a housekeeping file. To fulfill this responsibility, this abstraction collaborates with ASCII File and Housekeeping Data, and presents an interface of one GetFile contract and one Read contract, as shown in Figure 26. The GetFile contract returns an instance of a Housekeeping File abstraction. At this time, this instance is not considered a Singleton[4], as housekeeping files are read-only files, but the contract for returning an instance of this abstraction is left to look like a Singleton, in the case that that is what is implemented, or changed to during testing and/or maintenance. The Read contract reads the data from the file and stores it into a Housekeeping Data instance. This contract is to return a Boolean status to the caller, indicating if it was able to read the file or not. This abstraction needs to be kept persistent to work correctly, except in the case that the instantiated file is opened, read completely, and closed, in one structural calling sequence.

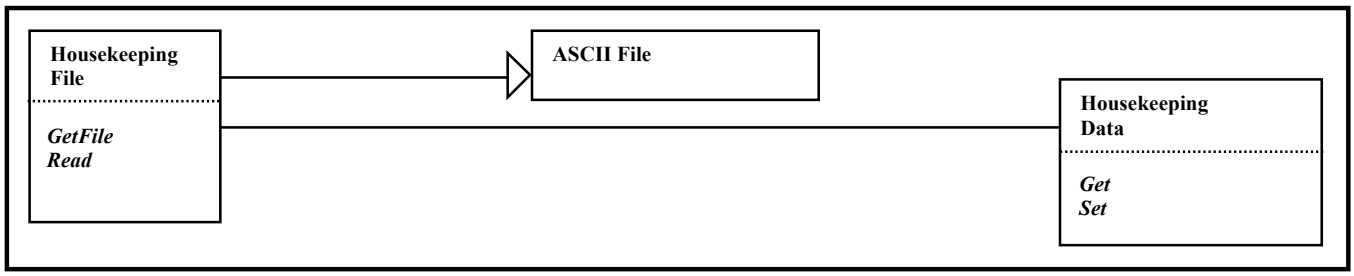

Figure 26 Housekeeping File and Housekeeping Data Abstractions

11.1.3 Housekeeping Data Abstraction

Housekeeping Data is a data abstraction, and has the responsibility to manage access to the data read from a Housekeeping File. To fulfill this responsibility, this abstraction presents an interface of one Set contract and various Get contracts, as shown in Figure 26. This abstraction also presents some constant values denoting various sizes of the data it contains. The copy constructor, assignment operator, or Set contract can be used by Housekeeping File to initialize the abstraction. The various Get contracts allow retrieval of all the data, or different cohesive sub-groups of the data.

11.2 Data Transformer Abstraction

Data Transformer is a process abstraction, and has the responsibility to transform the raw data in a HIR0Sci Packet, and return that system-usable data (e.g., "degrees Kelvin" or "photon counts") to the system. To fulfill this responsibility, this abstraction collaborates with HIR0Sci Packet, Azimuth Transformer, Elevation Transformer, Housekeeping Transformer, Radiance Transformer, Time Transformer, Transformed Data and Transformed Packet, and presents an interface of one Transform contract, as shown in Figure 27. The Transform contract calls the 5 different transformers, passing to them, in turn, the input HIR0Sci Packet, and returning a Boolean to the system to denote transformation success or failure. If the transformers all ran successfully, then Data Transformer fills a Transformed Packet with the usable data, and aggregates that packet into Transformed Data. The HIRDLS L1 Architecture document specifies that Data Transformer is to make available, to the system, an aggregation of all the transformed packets, and that is provided in Transformed Data. This abstraction needs to be persistent to work correctly.

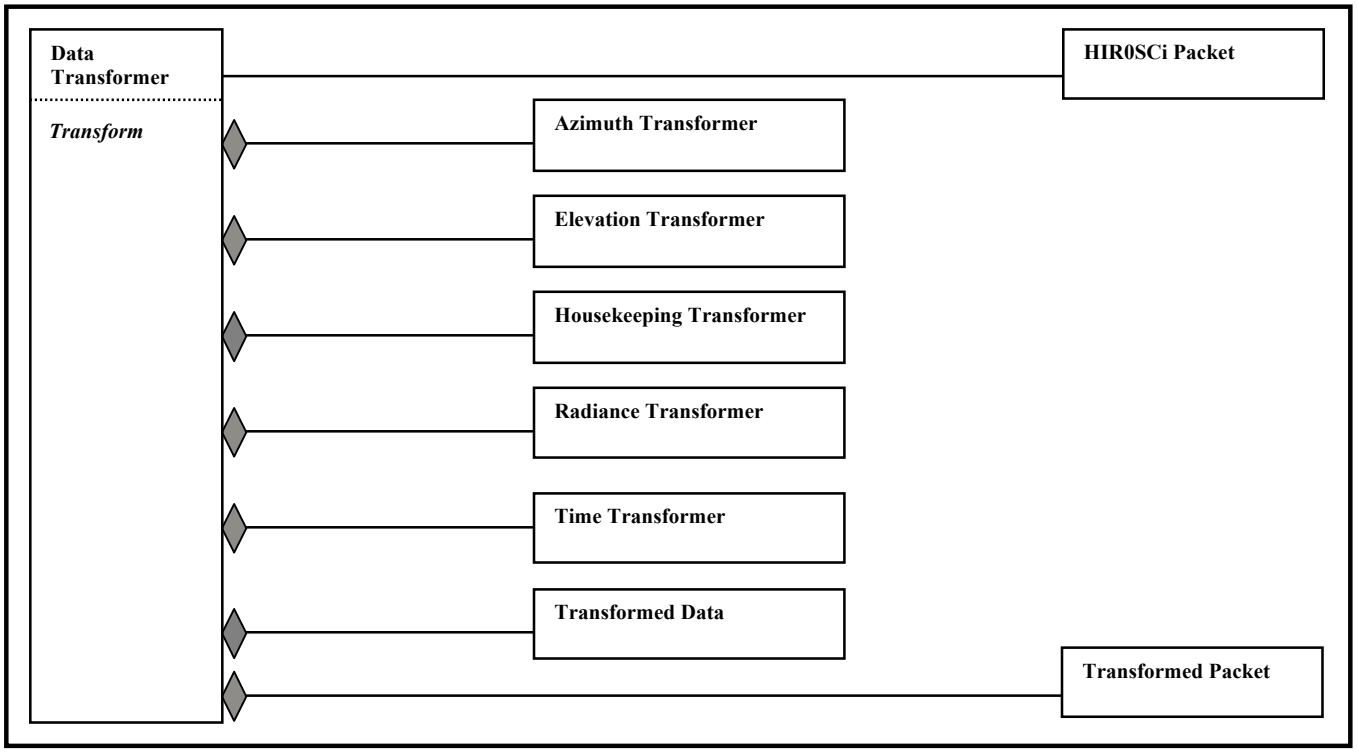

Figure 27 Data Transformer Abstraction

11.3 Azimuth Transformer Abstraction

Azimuth Transformer is a process abstraction, and has the responsibility to transform the azimuth data in HIR0Sci Packet from raw data into system-usable data. To fulfill this responsibility, this abstraction collaborates with HIR0Sci Packet, and presents an interface of one Transform contract, as shown in Figure 28. The Transform contract reads the azimuth data from HIR0Sci Packet, transforms the data, and then returns the transformed data and a status Boolean to Data Transformer.

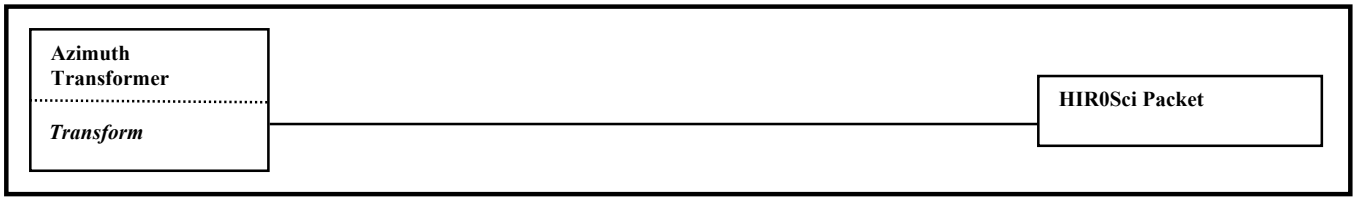

Figure 28 Azimuth Transformer Abstraction

11.4 Elevation Transformer Abstraction

Elevation Transformer is a process abstraction, and has the responsibility to transform the elevation data in HIR0Sci Packet, from raw data into system-usable data. To fulfill this responsibility, this abstraction collaborates with HIR0Sci Packet, and presents an interface of one Transform contract, as shown in Figure 29. The Transform contract reads the elevation data from HIR0Sci Packet, transforms the data, and then returns the transformed data and a status Boolean to Data Transformer.

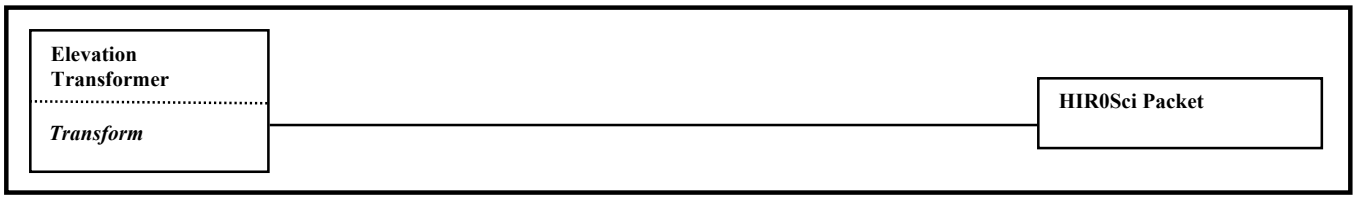

Figure 29 Elevation Transformer Abstraction

11.5 Radiance Transformer Abstraction

Radiance Transformer is a process abstraction, and has the responsibility to transform the radiance data in HIR0Sci Packet, from raw data into system-usable data. To fulfill this responsibility, this abstraction collaborates with HIR0Sci Packet, and presents an interface of one Transform contract, as shown in Figure 30. The Transform contract reads the radiance data from HIR0Sci Packet, transforms the data, and then returns the transformed data and a status Boolean to Data Transformer.

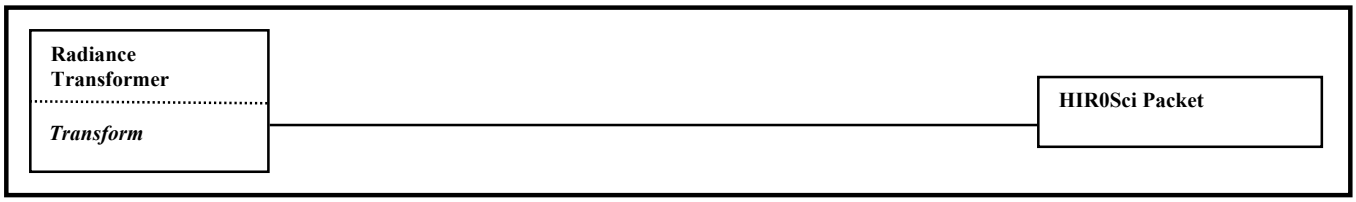

Figure 30 Radiance Transformer Abstraction

11.6 Time Transformer Abstraction

Time Transformer is a process abstraction, and has the responsibility to transform the time data in HIR0Sci Packet, from raw data into system-usable data. To fulfill this responsibility, this abstraction collaborates with HIR0Sci Packet and Time Conversion Service, and presents an interface of one Transform contract, as shown in Figure 31. The Transform contract reads the time data from HIR0Sci Packet, transforms the data, and then returns the transformed data and a status Boolean to Data Transformer.

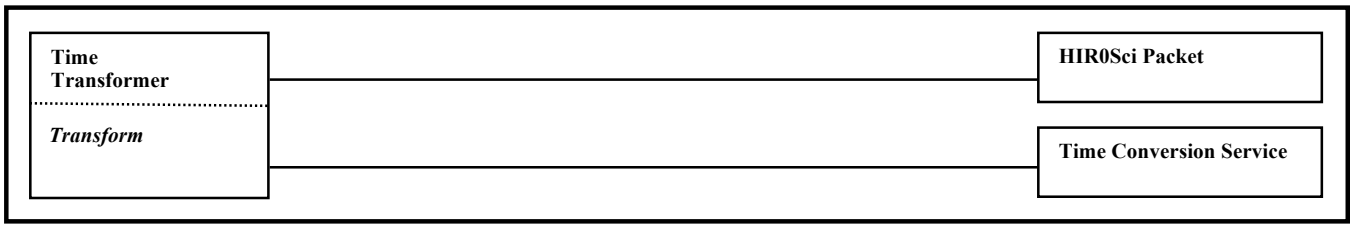

Figure 31 Time Transformer Abstraction

11.7 Transformed Data Abstraction

Transformed Data is a control abstraction, and has the responsibility to manage access to the collection of extracted and transformed packet data. To fulfill this responsibility, this abstraction collaborates with Transformed Packet, and presents an interface of one Add contract and one GetNext contract, as shown in Figure 32. The Add contract allows the system to add a Transformed Packet to the collection. It is specified that the Transformed Packet in the Add contract be chronologically subsequent to the Transformed Packet added in the previous call. The GetNext contract allows the system to have a copy of the Transformed Packet subsequent to the previous GetNext call. It is specified that using GetNext does not alter the Transformed Packets in the collection (e.g., delete them from the collection). It is specified that when this abstraction is deleted, all Transformed Packets in the collection are also deleted.

11.8 Transformed Packet Abstraction

Transformed Packet is a data abstraction, and has the responsibility to manage access to the extracted and transformed packet data. To fulfill this responsibility, this abstraction presents an interface of one Set contract and various Get contracts, as shown in Figure 32. The copy constructor, assignment operator, or Set contract can be used by Data Transformer to initialize the abstraction. The various Get contracts enumeration is, at this time, not considered exhaustive. The list may certainly shrink or grow.

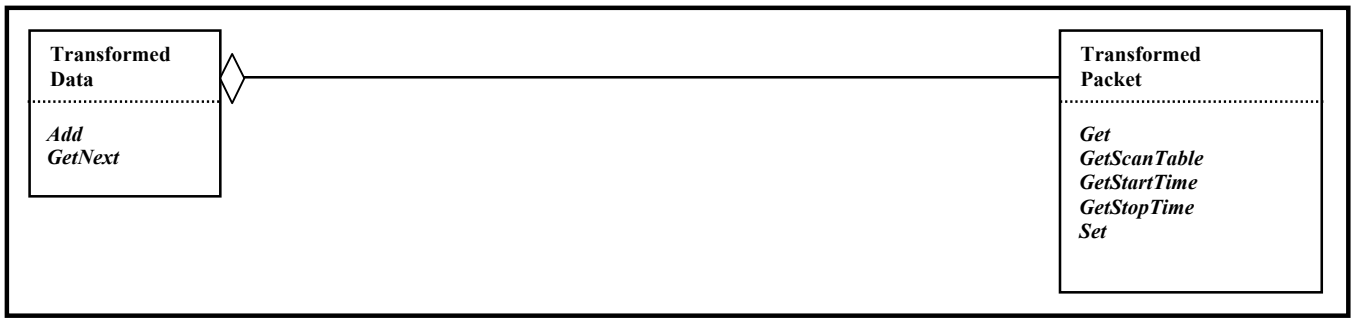

Figure 32 Transformed Data and Transformed Packet Abstractions

12 Flagger Package

The Flagger package has the responsibility to encapsulate all abstractions necessary to provide the system a means to assign various processing flags to all ingested and transformed packets. As shown in Figure 1, this package is used by the Sequencer and Builder packages, and has access to the Transformer, File, Service and Diagnostics packages. Figure 33 shows the hierarchy of the abstractions in the package. Each abstraction in the package is detailed further in this section.

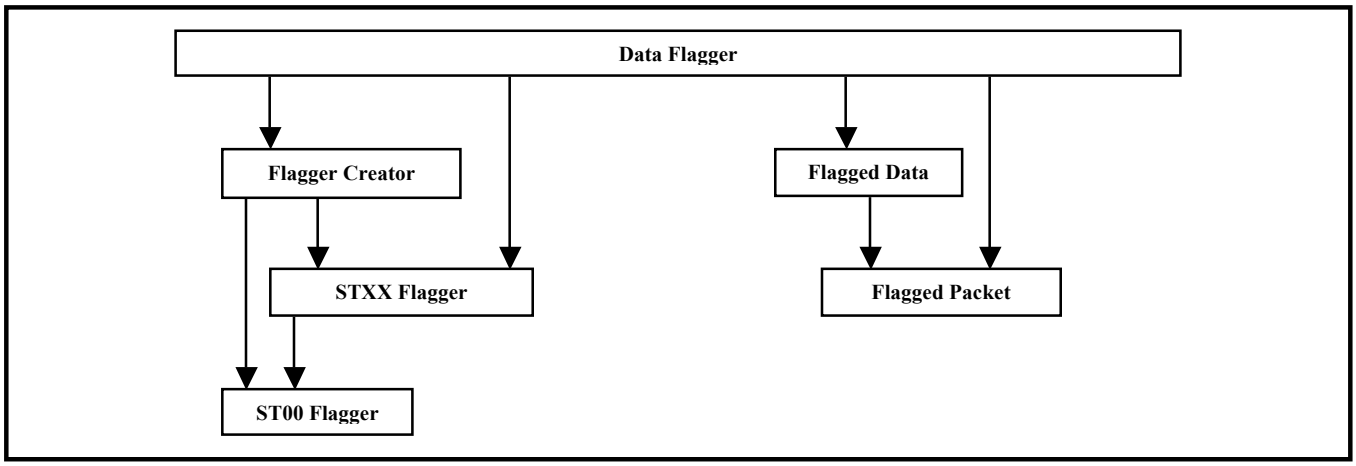

Figure 33 Flagger Package Hierarchy

12.1 Data Flagger Abstraction

Data Flagger is a process abstraction, and has the responsibility to attach various flags to all data points ingested into the system. To fulfill this responsibility, this abstraction collaborates with Transformed Data, Transformed Packet, Flagger Creator, STXX Flagger, Flagged Data and Flagged Packet, and presents an interface of one Flag contract, as shown in Figure 34. The Flag contract loops through the aggregated data in Transformed Data (calling Flagger Creator to return the appropriate ST-specific flagger), derives the flags for the respective data point, and aggregates that Flagged Packet into Flagged Data. The HIRDLS L1 Architecture document specifies that Data Flagger is to make available, to the system, an aggregation of all the packet flags, and that is provided in Flagged Data. Since this abstraction fulfills its tasks in one call, it does not need to be persistent to work correctly.

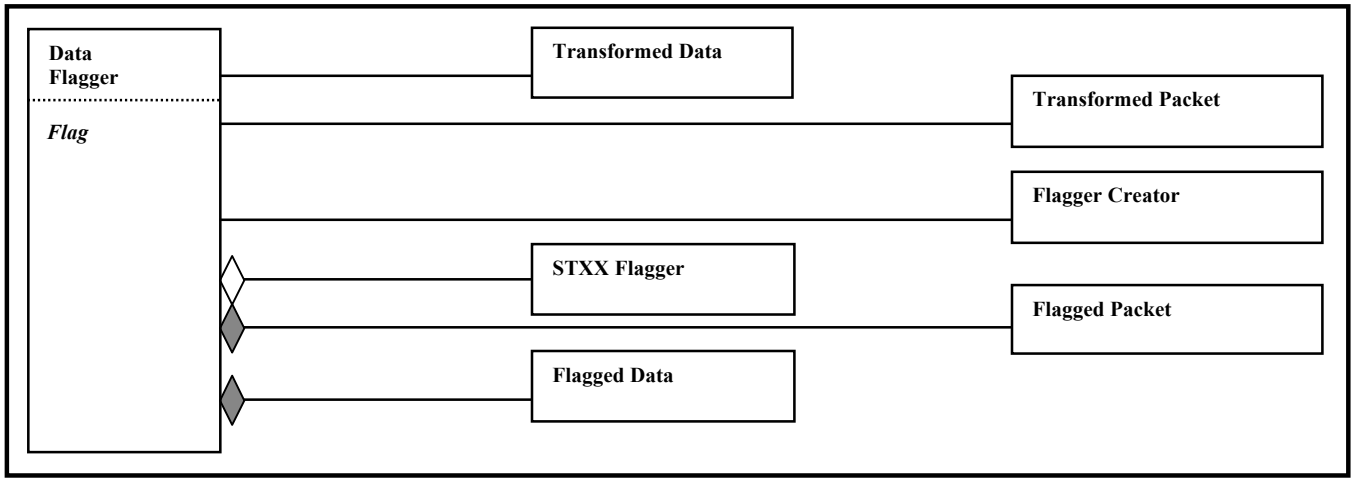

Figure 34 Data Flagger Abstraction

12.2 STXX Flagger Abstraction

STXX Flagger is a placeholder for process abstractions ST02 Flagger to ST33 Flagger, and each has the responsibility to attach various flags to data points ingested into the system that are of their respective scan table. To fulfill this responsibility, these abstractions collaborate with ST00 Flagger, and present an interface of one Flag contract, as shown in Figure 35. The Flag contract decides, given scan mirror movement information, if the data point is part of a nominal scan or is part of a Kapton scan, for instance. At this time, STXX denotes all scan tables, from 02 to 33, inclusive, except for 11, 18, 19, 31 and 32. Those scan table numbers are no longer in use.

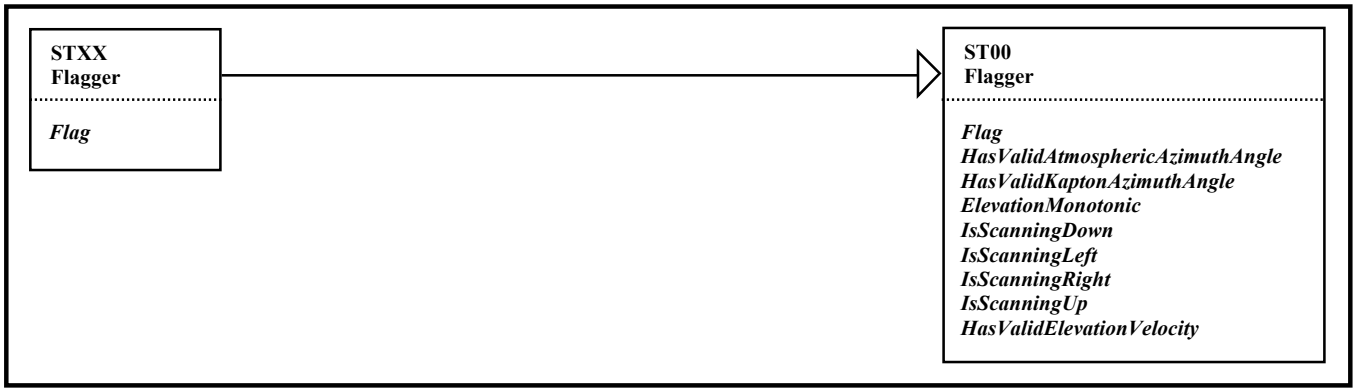

Figure 35 STXX Flagger and ST00 Flagger Abstractions

12.3 ST00 Flagger Abstraction

ST00 Flagger is a process abstraction, and each has the responsibility to provide a base abstraction for other abstractions (placeholded by STXX Flagger) that attach various flags to data points ingested into the system. To fulfill this responsibility, this abstraction presents an interface of one Flag contract and various Boolean-returning contracts that decide mirror movement direction, velocity and aperture location, amongst others, as shown in Figure 35. The Flag contract returns basic fail-safe data point information flags, to be used as a consistent starting point for the STXX Flagger abstractions, which are assumed to override the fail-safe information if their respective scan table processing determines as such. The other contracts are provided as services to the STXX Flagger abstractions.

12.4 Flagger Creator Abstraction

Flagger Creator is a process abstraction, and each has the responsibility to create the appropriate STXX Flagger to be used by the system. To fulfill this responsibility, this abstraction collaborates with STXX Flagger and ST00 Flagger, and presents an interface of one Create contract, as shown in Figure 36. The Create contract is specified to be an *Abstract Factory creational pattern*[5], and return an STXX Flagger (or ST00 Flagger as default). This abstraction is not meant to be instantiated, but instead provide non-dynamic data into the global space. Persistency is not an issue.

12.5 Flagged Data Abstraction

Flagged Data is a control abstraction, and has the responsibility to manage access to the collection of data packet flags. To fulfill this responsibility, this abstraction collaborates with Flagged Packet, and presents an interface of one Add contract, one GetNext contract, and various data point information access contracts, as shown in Figure 37. The Add contract allows the system to add a Flagged Packet to the collection. It is specified that the Flagged Packet in the Add contract be chronologically subsequent to the Flagged Packet added in the previous call. The GetNext contract allows the system to have

[5] As detailed in Design Patterns, Elements of Reusable Object-Oriented Software by Gamma, et.al., 1995

a copy of the Flagged Packet subsequent to the previous GetNext call. It is specified that using GetNext does not alter the Flagged Packets in the collection (e.g., delete them from the collection). It is specified that when this abstraction is deleted, all Flagged Packets in the collection are also deleted. The various data point information access contracts will be necessary for the Builder package to use to determine which packets are to be further processed.

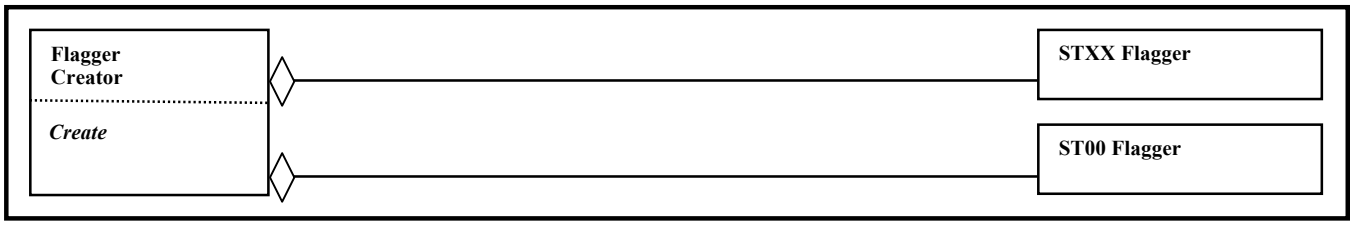

Figure 36 Flagger Creator Abstraction

12.6 Flagged Packet Abstraction

Flagged Packet is a data abstraction, and has the responsibility to manage access to the data packet flags. To fulfill this responsibility, this abstraction presents an interface of one Set contract, various Get contracts, and various information access contracts, as shown in Figure 37. The copy constructor, assignment operator, or Set contract can be used by Data Flagger to initialize the abstraction. The various Get and information access contracts enumeration is, at this time, not considered exhaustive. The list may certainly shrink or grow.

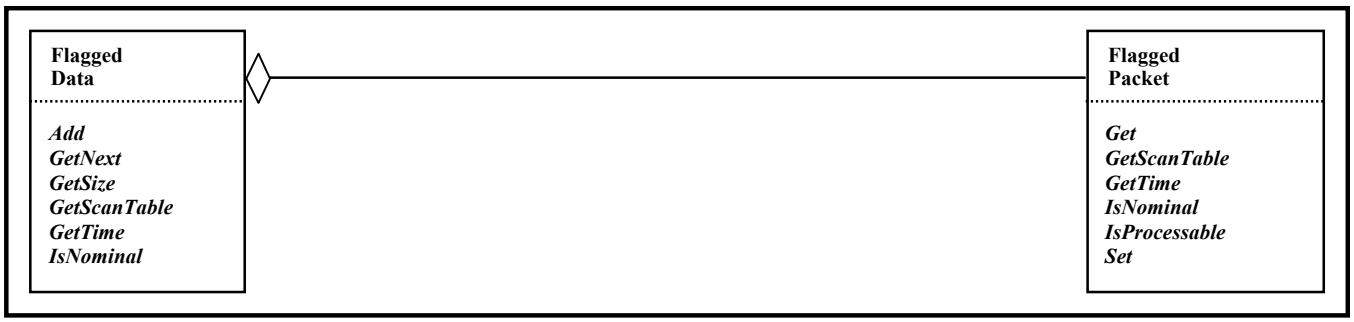

Figure 37 Flagged Data and Flagged Packet Abstractions

13 Builder Package

The Builder package has the responsibility to encapsulate all abstractions necessary to provide the system a means to build the sequence of data packets that are to be further processed by the system. As shown in Figure 1, this package is used by the Sequencer package, and has access to the Transformer, Flagger, File, Service and Diagnostics packages. Figure 38 shows the hierarchy of the abstractions in the package. Each abstraction in the package is detailed further in this section.

13.1 Sequence Builder Abstraction

Sequence Builder is a process abstraction, and has the responsibility to build the processable sequence of ingested, transformed and flagged data. To fulfill this responsibility, this abstraction collaborates with Transformed Data, Transformed Packet, Flagged Data, Flagged Packet, Sequenced Data and Sequenced Packet, and presents an interface of one Build contract, as shown in Figure 39. The Build contract loops through the aggregated data in Transformed Data and Flagged

Data, throwing away the packets that are marked unprocessable, and sequencing the remainder into Sequenced Packets, aggregated into Sequenced Data. The HIRDLS L1 Architecture document specifies that Data Flagger is to make available, to the system, an aggregation of all the processable data, in sequence, and that is provided in Sequenced Data. Since this abstraction fulfills its tasks in one call, it does not need to be persistent to work correctly.

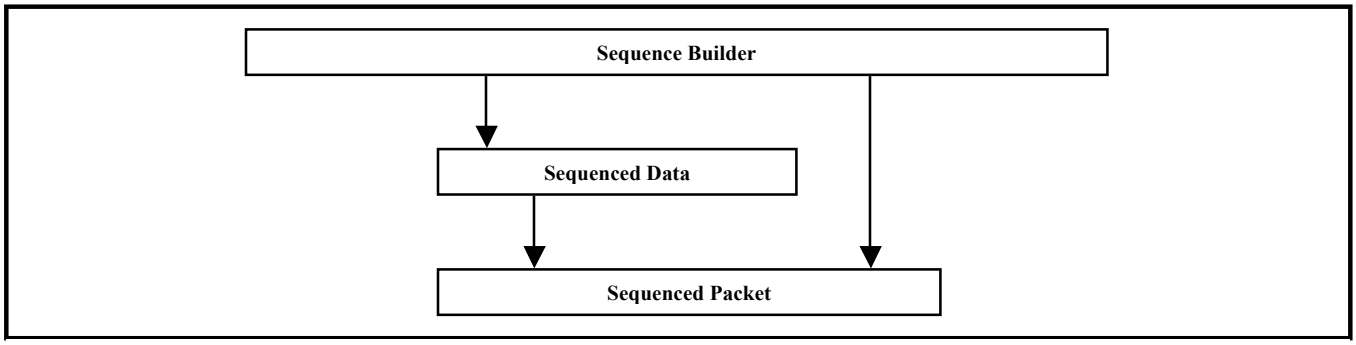

Figure 38 Builder Package Hierarchy

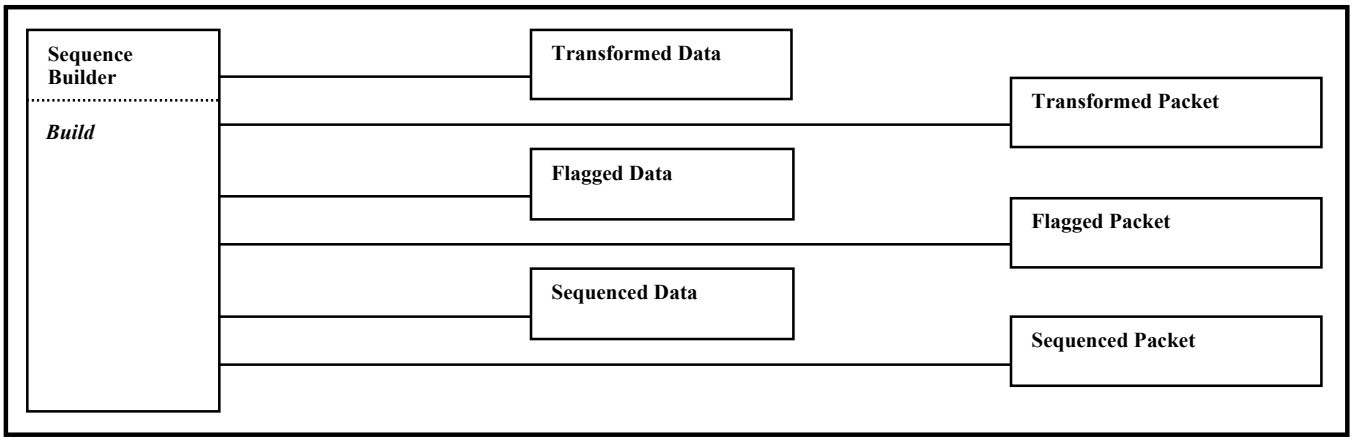

Figure 39 Sequence Builder Abstraction

13.2 Sequenced Data Abstraction

Sequenced Data is a control abstraction, and has the responsibility to manage access to the collection of processable sequenced data packets. To fulfill this responsibility, this abstraction collaborates with Sequenced Packet, and presents an interface of one Add contract, and various Get and Update contracts, as shown in Figure 40. The Add contract allows the system to add a Sequenced Packet to the collection. It is specified that the Sequenced Packet in the Add contract be chronologically subsequent to the Sequenced Packet added in the previous call. The Get contracts allow subsequent process abstractions (e.g., Data Locator, detailed in Section 15) to access the data in the collection, to be used for the respective abstraction's task fulfillment. The results of those task fulfillments can then be added into the collection via the Update contracts. It is specified that using the various Get contracts does not alter the Sequenced Packets in the collection (e.g., delete them from the collection). It is specified that when this abstraction is deleted, all Sequenced Packets in the collection are also deleted.

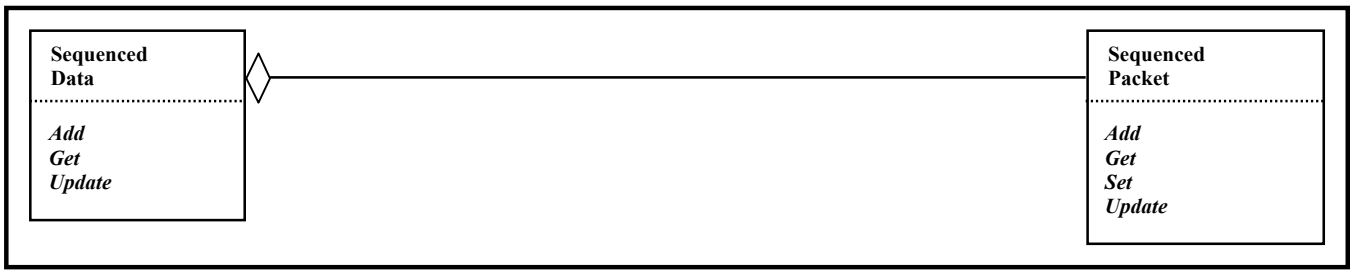

Figure 40 Sequenced Data and Sequenced Packet Abstractions

13.3 Sequenced Packet Abstraction

Sequenced Packet is a data abstraction, and has the responsibility to manage access to the processable sequenced data packet. To fulfill this responsibility, this abstraction presents an interface of one Set contract, various Get contracts, and various Update contracts, as shown in Figure 40. The copy constructor, assignment operator, or Set contract can be used by Sequence Builder to initialize the abstraction. The various Get and Update contracts enumeration is, at this time, not considered exhaustive. The list may certainly shrink or grow.

14 Sequencer Package

The Sequencer package has the responsibility to encapsulate all abstractions necessary to provide the system a means to ingest packets of raw HIR0Sci data, transform those packets into system-usable units, flag the packets to denote processability (amongst other things), and finally, build a sequence of the processable data packets. As shown in Figure 1, this package is used by the Processor, Locator and Calibrator packages, and has access to the Ingester, Transformer, Flagger, Builder, File, Service and Diagnostics packages. The only abstraction in this package is Data Sequencer, and it is detailed further in this section.

14.1 Data Sequencer Abstraction

Data Sequencer is a process abstraction, and has the responsibility to ingest packets of raw HIR0Sci data, transform those packets into system-usable units, flag the packets to denote processability (amongst other things), and build a sequence of the processable data packets. To fulfill this responsibility, this abstraction collaborates with Packet Ingester, HIR0Sci Packet, Data Transformer, Transformed Data, Data Flagger, Flagged Data, Sequence Builder and Sequenced Data, and presents an interface of one Sequence contract, as shown in Figure 41. The Sequence contract calls, in a loop, Packet Ingester and Data Transformer, to ingest and transform packets. When the HIR0Sci Packets have all been ingested and transformed, Data Sequencer calls Data Flagger to flag all the ingested and transformed packets. When Data Flagger has finished flagging all the data, Data Sequencer calls Sequence Builder to build the sequence of processable packets, returning that collection in Sequenced Data. Since this abstraction fulfills its tasks in one call, it does not need to be persistent to work correctly.

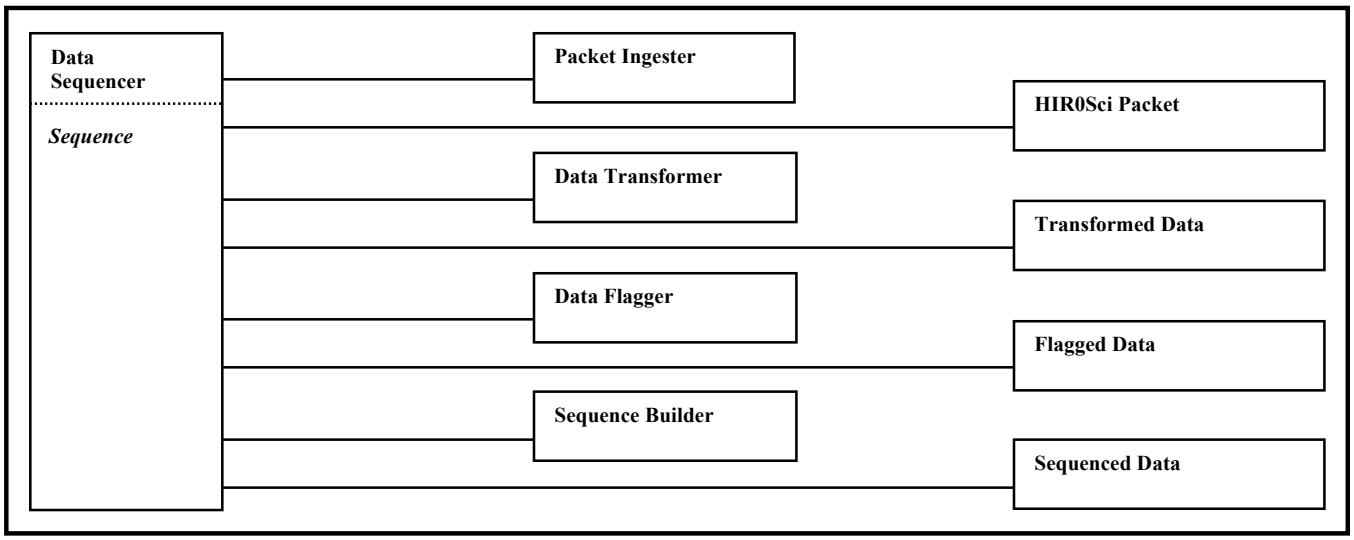

Figure 41 Data Sequencer Abstraction

15 Locator Package

The Locator package has the responsibility to encapsulate all abstractions necessary to provide the system a means to geo-locate the sequenced data packets. As shown in Figure 1, this package is used by the Processor package, and has access to the Sequencer, File, Service and Diagnostics packages. Figure 42 shows the hierarchy of the abstractions in the package. Each abstraction in the package is detailed further in this section.

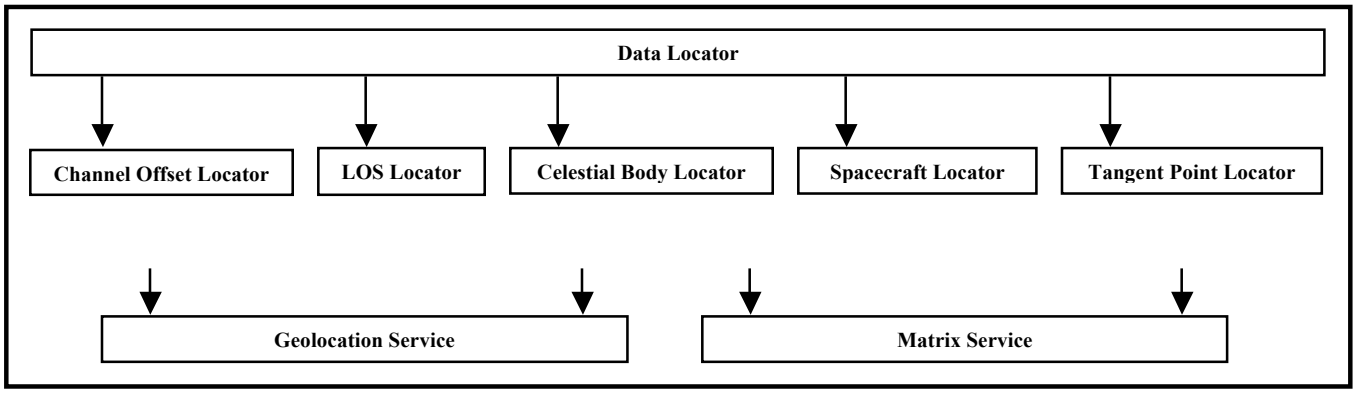

Figure 42 Locator Package Hierarchy

15.1 Geolocation Service Abstraction

Geolocation Service is a control abstraction, and has the responsibility to provide a simple interface to the SDP Toolkit geolocation routines. To fulfill this responsibility, this abstraction collaborates with the SDP Toolkit, and presents an interface of geolocation retrieval contracts, as shown in Figure 43. This contract listing in Figure 43 is intentionally minimal, as the extent of the contracts is expected to be large. Please see Appendix A for a more comprehensive listing. This abstraction need not be persistent to work correctly.

15.2 Matrix Service Abstraction

Matrix Service is a control abstraction, and has the responsibility to provide a cohesive matrix manipulation interface. To fulfill this responsibility, this abstraction presents various Multiply, Invert and Transpose contracts, as shown in Figure 44. This contract listing is expected to shrink or grow during implementation.

Figure 43 Geolocation Service Abstraction

Figure 44 Matrix Service Abstraction

15.3 Data Locator Abstraction

Data Locator is a process abstraction, and has the responsibility to geo-locate the processable data packets in the system. To fulfill this responsibility, this abstraction collaborates with Sequenced Data, Sequenced Packet, Geolocation Service, Matrix Service, Celestial Body Locator, Channel Offset Locator, LOS Locator, Spacecraft Locator and Tangent Point Locator, and presents an interface of one Locate contract, as shown in Figure 45. The Locate contract calls Sequenced Data to return the next Sequenced Packet, then calls the various abstractions in the order necessary to generate all the required geo-location data, and then calls the appropriate Update contracts of Sequenced Packet and Sequenced Data. Since this abstraction is to fulfill its tasks in one call, all of this happens in a loop, until there are no more packets of data to geo-locate. And since all of this happens in one call, this abstraction need not be persistent to work correctly.

15.4 Celestial Body Locator Abstraction

Celestial Body Locator is a process abstraction, and has the responsibility to derive celestial body geolocation information. To fulfill this responsibility, this abstraction collaborates with Geolocation Service and Matrix Service, and presents an interface of one Locate contract, as shown in Figure 46. The Locate contract derives information about celestial bodies in the HIRDLS field of view, and returns that information to the caller, along with a Boolean denoting the status of the call.

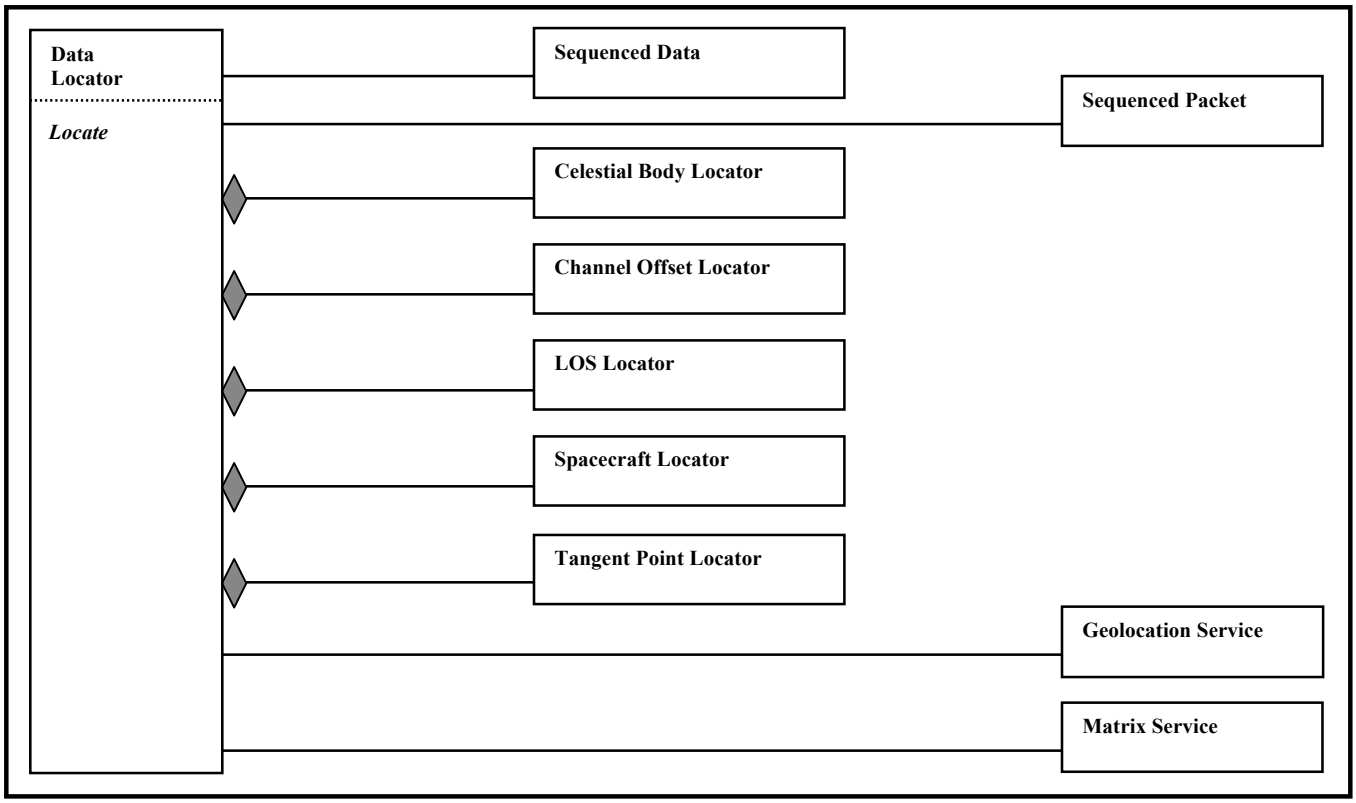

Figure 45 Data Locator Abstraction

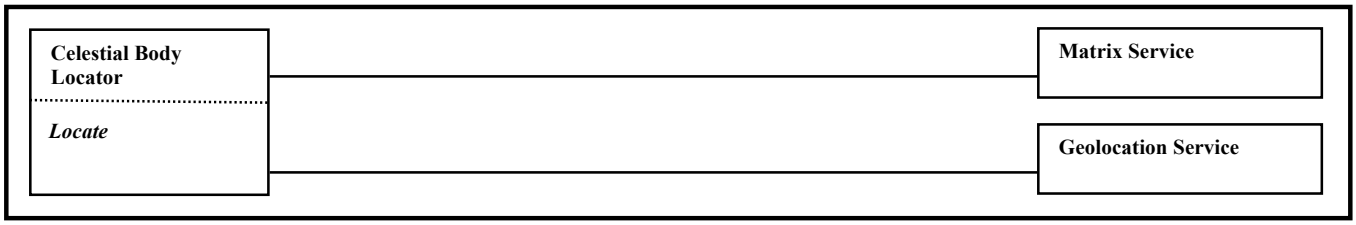

Figure 46 Celestial Body Locator Abstraction

15.5 Channel Offset Locator Abstraction

Channel Offset Locator is a process abstraction, and has the responsibility to derive channel offset-from-boresight altitude information. To fulfill this responsibility, this abstraction collaborates with Geolocation Service and Matrix Service, and presents an interface of one Locate contract, as shown in Figure 47. The Locate contract derives the offset-from-boresight information for each channel, and returns that information to the caller, along with a Boolean denoting the status of the call.

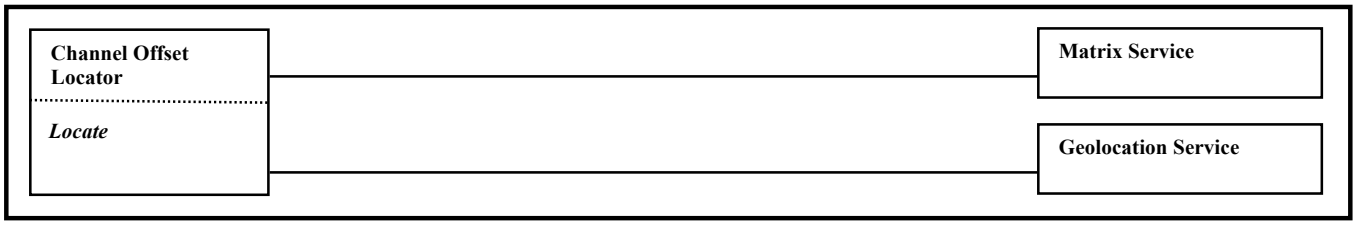

Figure 47 Channel Offset Locator Abstraction

15.6 LOS Locator Abstraction

LOS Locator is a process abstraction, and has the responsibility to derive HIRDLS line-of-sight location information. To fulfill this responsibility, this abstraction collaborates with Geolocation Service and Matrix Service, and presents an interface of one Locate contract, as shown in Figure 48. The Locate contract derives the line-of-sight information, and returns that information to the caller, along with a Boolean denoting the status of the call.

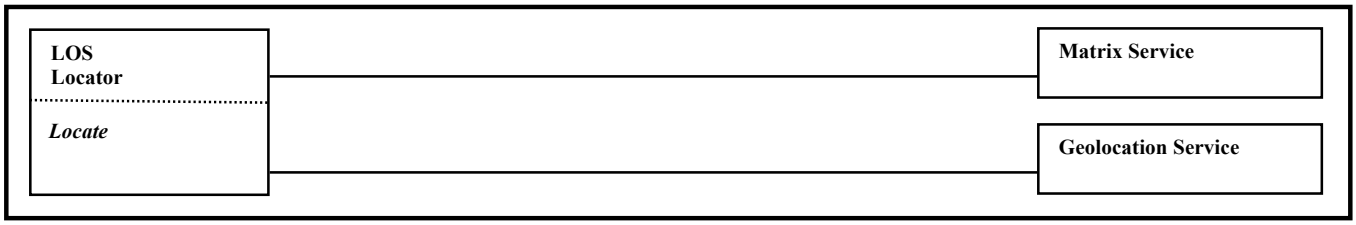

Figure 48 LOS Locator Abstraction

15.7 Spacecraft Locator Abstraction

Spacecraft Locator is a process abstraction, and has the responsibility to derive Aura spacecraft location information. To fulfill this responsibility, this abstraction collaborates with Geolocation Service and Matrix Service, and presents an interface of one Locate contract, as shown in Figure 49. The Locate contract derives the spacecraft information, and returns that information to the caller, along with a Boolean denoting the status of the call.

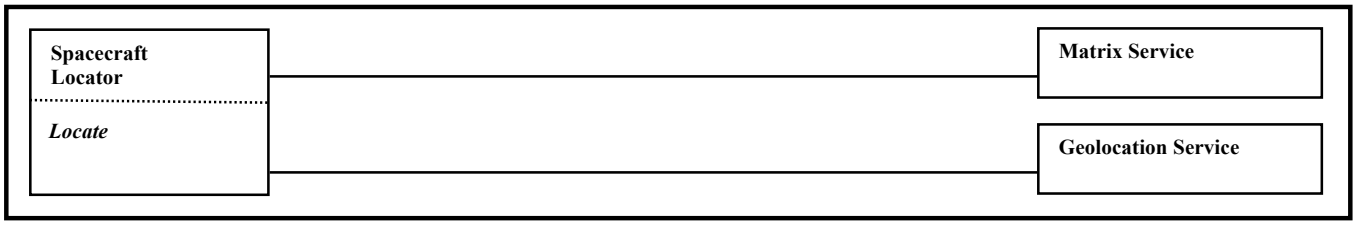

Figure 49 Spacecraft Locator Abstraction

15.8 Tangent Point Locator Abstraction

Tangent Point Locator is a process abstraction, and has the responsibility to derive HIRDLS tangent point location information. To fulfill this responsibility, this abstraction collaborates with Geolocation Service and Matrix Service, and presents an interface of one Locate contract, as shown in Figure 50. The Locate contract derives the tangent point information, and returns that information to the caller, along with a Boolean denoting the status of the call.

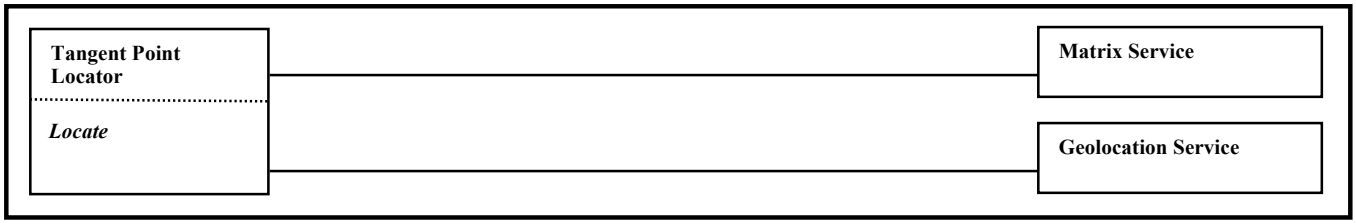

Figure 50 Tangent Point Locator Abstraction

16 Calibrator Package

The Calibrator package has the responsibility to encapsulate all abstractions necessary to provide the system a means to calibrate the radiances in the sequenced data packets. As shown in Figure 1, this package is used by the Processor package, and has access to the Sequencer, File, Service and Diagnostics packages. Figure 51 shows the hierarchy of the abstractions and sub-packages in the package. Each abstraction and sub-package in the package is detailed further in this section.

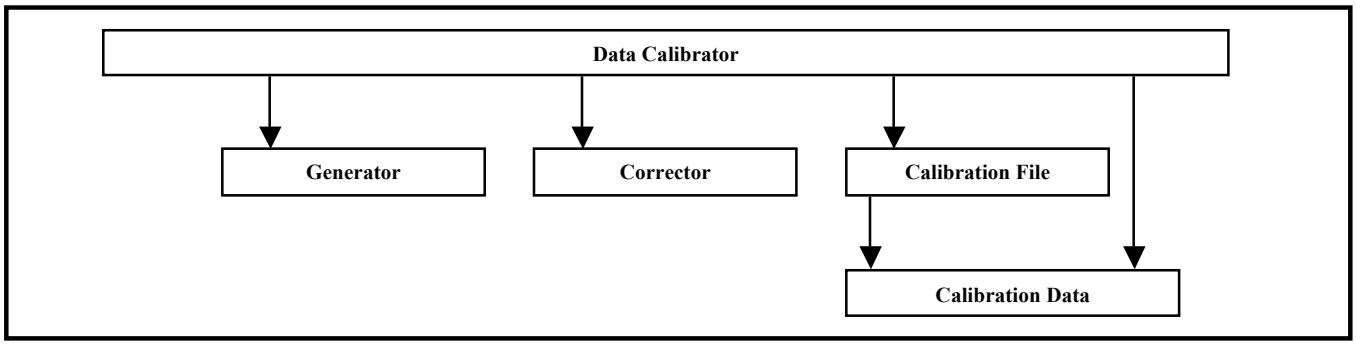

Figure 51 Calibrator Package Hierarchy

16.1 Generator Package

The Generator package has the responsibility to encapsulate all abstractions necessary to provide the Data Calibrator a means to generate radiometric offset data, to be used for calibration. As shown in Figure 51, this package is used by the Data Calibrator abstraction. Figure 52 shows the hierarchy of the abstractions in the package. Each abstraction in the package is detailed further in this section.

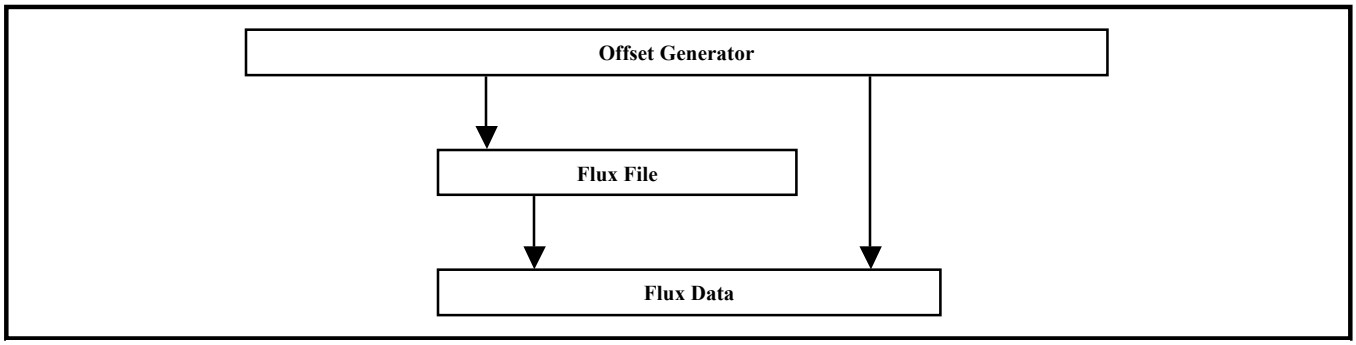

Figure 52 Generator Package Hierarchy

16.1.1 Offset Generator Abstraction

Offset Generator is a process abstraction, and has the responsibility to generate radiometric offset data for a given data point or points. To fulfill this responsibility, this abstraction collaborates with Flux File and Flux Data, and presents an interface of one Generate contract, as shown in Figure 53. The Generate contract uses the stored flux data to derive a data point's radiometric offset data. The stored flux data is specified to be read in at instantiation, so therefore this abstraction needs to be persistent to work efficiently.

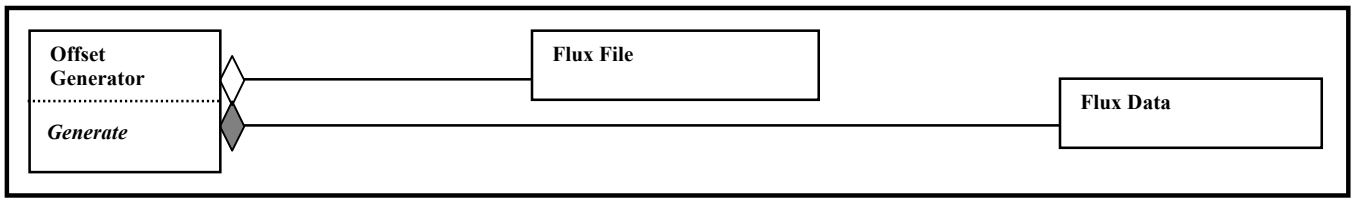

Figure 53 Offset Generator Abstraction

16.1.2 Flux File Abstraction

Flux File is a control abstraction, and has the responsibility to manage all access to a radiometric flux file. To fulfill this responsibility, this abstraction collaborates with ASCII File and Flux Data, and presents an interface of one GetFile contract and one Read contract, as shown in Figure 54. The GetFile contract returns an instance of a Flux File abstraction. At this time, this instance is not considered a Singleton[6], as flux files are read-only files, but the contract for returning an instance of this abstraction is left to look like a Singleton, in the case that that is what is implemented, or changed to during testing and/or maintenance. The Read contract reads the data from the file and stores it into a Flux Data instance. This contract is to return a Boolean status to the caller, indicating if it was able to read the file or not. This abstraction needs to be kept persistent to work correctly, except in the case that the instantiated file is opened, read completely, and closed, in one structural calling sequence.

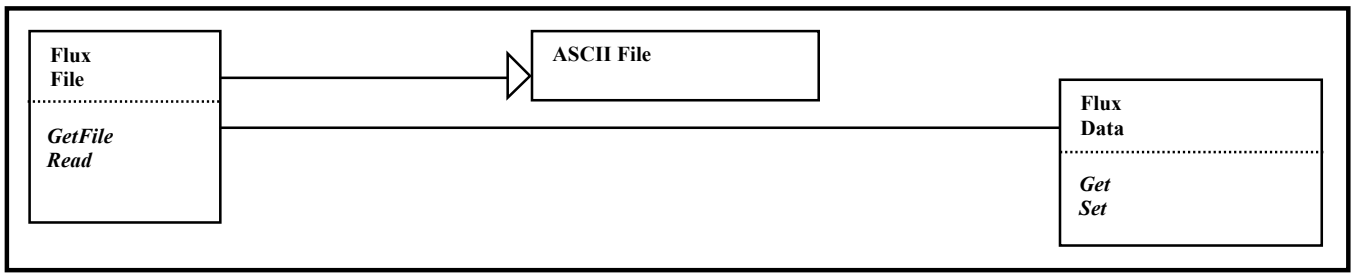

Figure 54 Flux File and Flux Data Abstractions

16.1.3 Flux Data Abstraction

Flux Data is a data abstraction, and has the responsibility to manage access to the data read from a Flux File. To fulfill this responsibility, this abstraction presents an interface of one Set contract and one Get contract, as shown in Figure 54. This abstraction also presents some a constant value denoting the size of the data it contains. The copy constructor, assignment operator, or Set contract can be used by Flux File to initialize the abstraction. The Get contract allows retrieval of all the data.

16.2 Corrector Package

The Corrector package has the responsibility to encapsulate all abstractions necessary to provide the Data Calibrator a means to correct for out-of-field contamination of the radiances. As shown in Figure 51, this package is used by the Data Calibrator abstraction. Figure 55 shows the hierarchy of the abstractions in the package. Each abstraction in the package is detailed further in this section.

[6] See Section 7.1

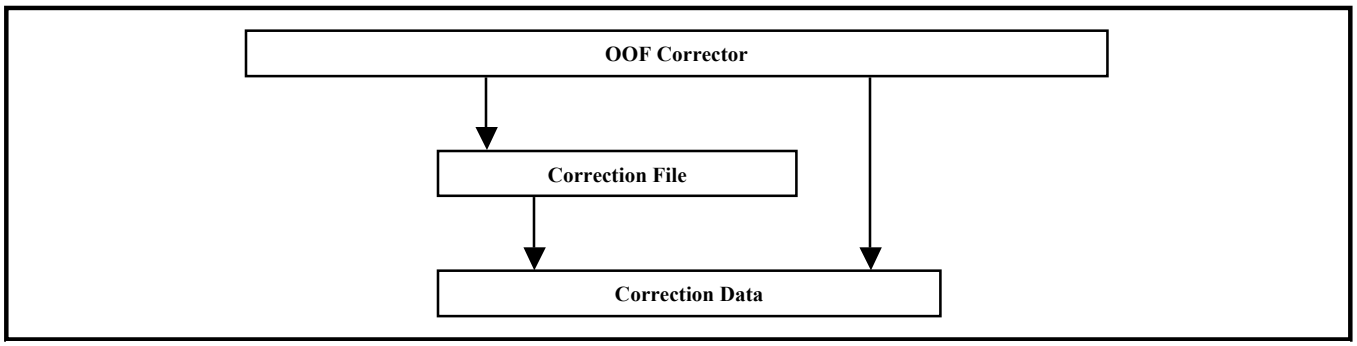

Figure 55 Corrector Package Hierarchy

16.2.1 OOF Corrector Abstraction

OOF Corrector is a process abstraction, and has the responsibility to correct for radiometric out-of-field contamination of a data point or points. To fulfill this responsibility, this abstraction collaborates with Correction File and Correction Data, and presents an interface of one Correct contract, as shown in Figure 56. The Correct contract uses the stored correction data to correct for the out-of-field contamination. The stored correction data is specified to be read in at instantiation, so therefore this abstraction needs to be persistent to work efficiently.

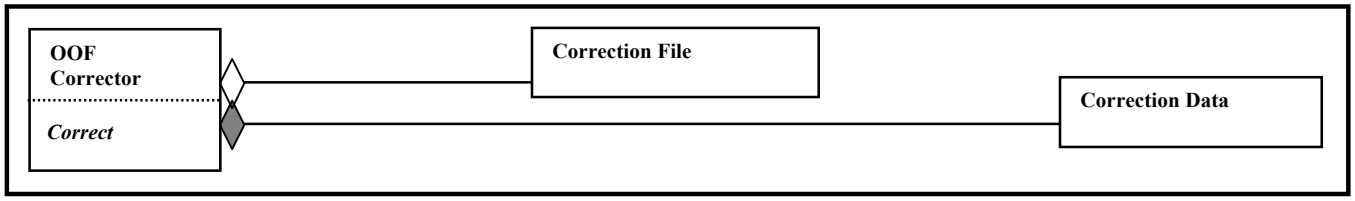

Figure 56 OOF Corrector Abstraction

16.2.2 Correction File Abstraction

Correction File is a control abstraction, and has the responsibility to manage all access to an out-of-field correction file. To fulfill this responsibility, this abstraction collaborates with HDF5 Read File and Correction Data, and presents an interface of one GetFile contract and one Read contract, as shown in Figure 57. The GetFile contract returns an instance of a Correction File abstraction. At this time, this instance is not considered a Singleton[7], as correction files are read-only files, but the contract for returning an instance of this abstraction is left to look like a Singleton, in the case that that is what is implemented, or changed to during testing and/or maintenance. The Read contract reads the data from the file and stores it into a Correction Data instance. This contract is to return a Boolean status to the caller, indicating if it was able to read the file or not. This abstraction needs to be kept persistent to work correctly, except in the case that the instantiated file is opened, read completely, and closed, in one structural calling sequence.

[7] See Section 7.1

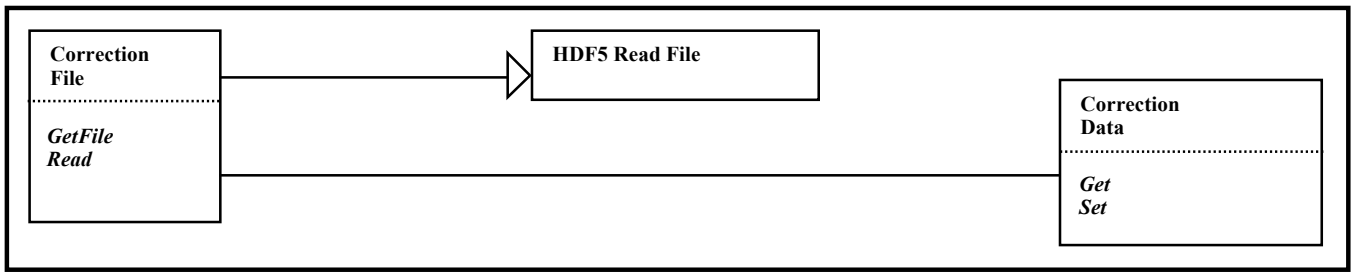

Figure 57 Correction File and Correction Data Abstractions

16.2.3 Correction Data Abstraction

Correction Data is a data abstraction, and has the responsibility to manage access to the data read from a Correction File. To fulfill this responsibility, this abstraction presents an interface of one Set contract and one Get contract, as shown in Figure 57. This abstraction also presents some a constant value denoting the size of the data it contains. The copy constructor, assignment operator, or Set contract can be used by Correction File to initialize the abstraction. The Get contract allows retrieval of all the data.

16.3 Data Calibrator Abstraction

Data Calibrator is a process abstraction, and has the responsibility to calibrate the radiances in the processable data packets in the system. To fulfill this responsibility, this abstraction collaborates with Sequenced Data, Sequenced Packet, Offset Generator, OOF Corrector, Calibration File and Calibration Data, and presents an interface of one Calibrate contract, as shown in Figure 58. The Calibrate contract calls Sequenced Data to return the next Sequenced Packet, then calls the Offset Generator and OOF Corrector to help calibrate the radiances (Calibration File and Calibration Data are used on instantiation to store calibration data), and then calls the appropriate Update contracts of Sequenced Packet and Sequenced Data. Since this abstraction is to fulfill its tasks in one call, all of this happens in a loop, until there are no more packets of data to geo-locate. And since all of this happens in one call, this abstraction need not be persistent to work correctly.

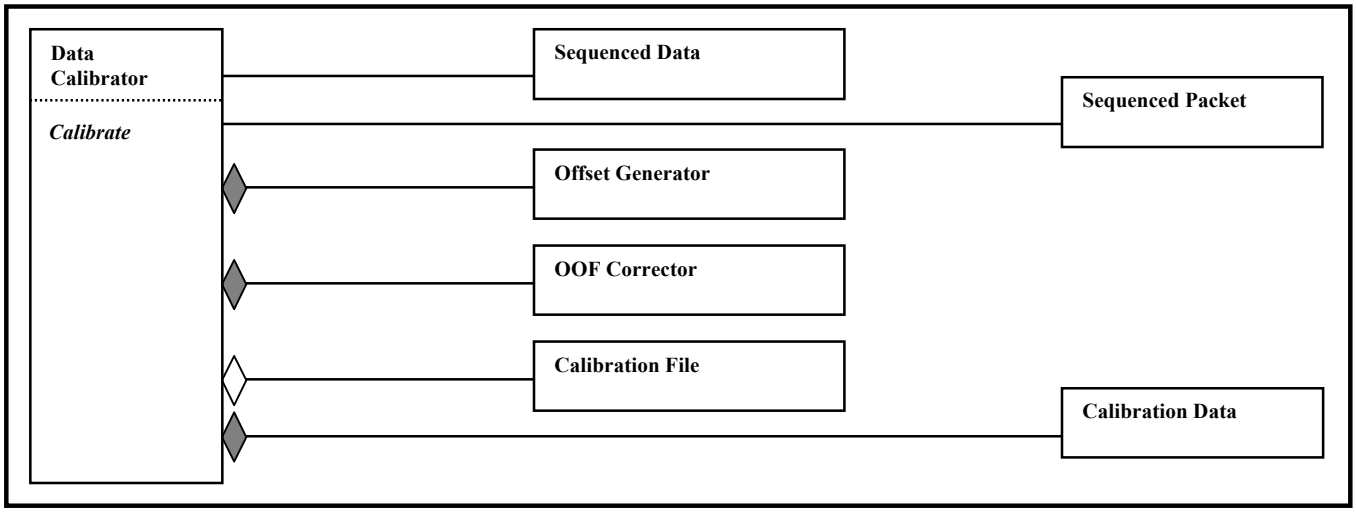

Figure 58 Data Calibrator Abstraction

16.4 Calibration File Abstraction

Calibration File is a control abstraction, and has the responsibility to manage all access to a radiometric calibration data file. To fulfill this responsibility, this abstraction collaborates with ASCII File and Calibration Data, and presents an interface of one GetFile contract and one Read contract, as shown in Figure 59. The GetFile contract returns an instance of a Calibration File abstraction. At this time, this instance is not considered a Singleton[8], as calibration files are read-only files, but the contract for returning an instance of this abstraction is left to look like a Singleton, in the case that that is what is implemented, or changed to during testing and/or maintenance. The Read contract reads the data from the file and stores it into a Calibration Data instance. This contract is to return a Boolean status to the caller, indicating if it was able to read the file or not. This abstraction needs to be kept persistent to work correctly, except in the case that the instantiated file is opened, read completely, and closed, in one structural calling sequence.

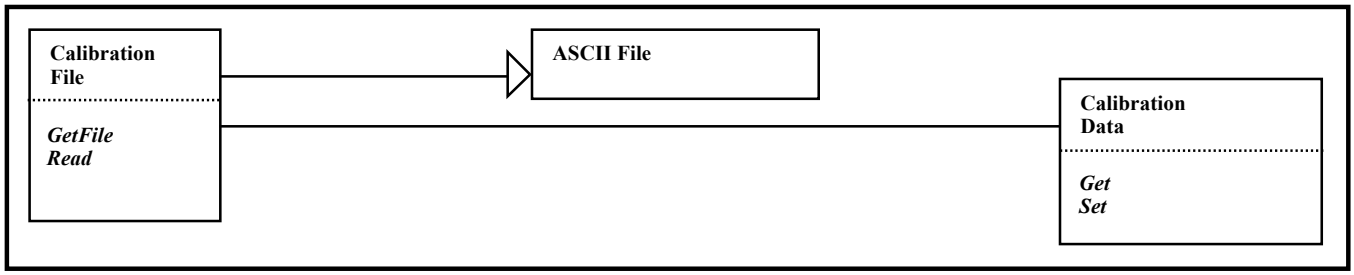

Figure 59 Calibration File and Calibration Data Abstractions

16.5 Calibration Data Abstraction

Calibration Data is a data abstraction, and has the responsibility to manage access to the data read from a Calibration File. To fulfill this responsibility, this abstraction presents an interface of one Set contract and one Get contract, as shown in Figure 59. This abstraction also presents a constant value denoting the size of the data it contains. The copy constructor, assignment operator, or Set contract can be used by Calibration File to initialize the abstraction. The Get contract allows retrieval of all the data.

17 Reformer Package

The Reformer package has the responsibility to encapsulate all abstractions necessary to provide the system a means to reform the geo-located and calibrated sequenced data packets into a form for writing to the output HIRDLS1 file. As shown in Figure 1, this package is used by the Writer and Egester packages, and has access to the File, Service and Diagnostics packages. Not shown in Figure 1 is that this package also has access to the Sequencer package. Figure 60 shows the hierarchy of the abstractions in the package. Each abstraction in the package is detailed further in this section.

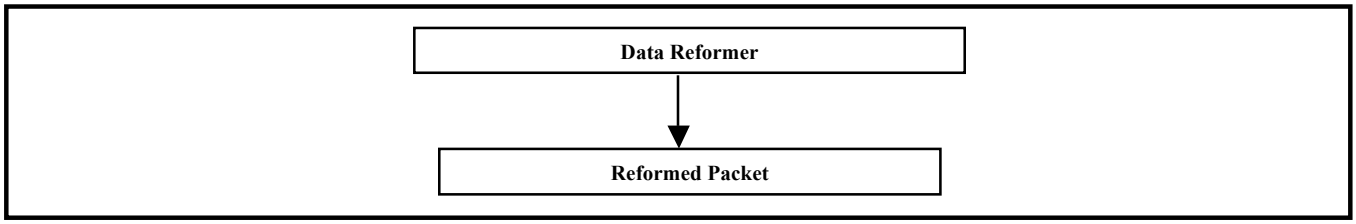

Figure 60 Reformer Package Hierarchy

[8] See Section 7.1

17.1 Data Reformer Abstraction

Data Reformer is a process abstraction, and has the responsibility to reform a packet of geo-located and calibrated data into the form required for writing to the HIRDLS1 file. To fulfill this responsibility, this abstraction collaborates with Sequenced Packet and Reformed Packet, and presents an interface of one Reform contract, as shown in Figure 61. The Reform contract extracts the data from Sequenced Packet, reforms it, then writes the reformed data to a Reformed Packet. This abstraction need not be persistent to work correctly, but should be kept persistent to work efficiently.

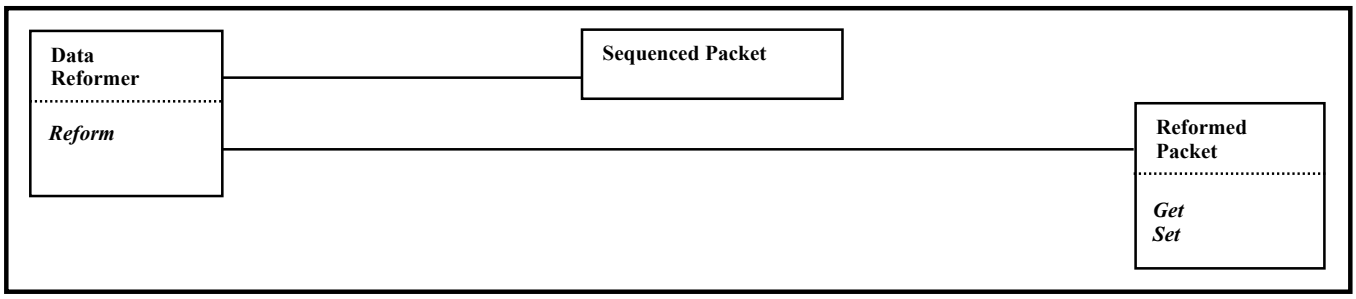

Figure 61 Data Reformer Abstraction

17.2 Reformed Packet Abstraction

Reformed Packet is a data abstraction, and has the responsibility to manage access to a packet of reformed data. To fulfill this responsibility, this abstraction presents an interface of one Set and one Get contract, as shown in Figure 61. The copy constructor, assignment operator, or Set contract can be used by Data Reformer to initialize the abstraction. The various Get contracts (see Appendix A for a full listing) allow the caller to retrieve all or part of the reformed data.

18 Egester Package

The Egester package has the responsibility to encapsulate all abstractions necessary to provide the system a means to egest (write) reformed data chunks to the HIRDLS1 file. As shown in Figure 1, this package is used by the Writer package, and has access to the Reformer, File, Service and Diagnostics packages. Figure 62 shows the hierarchy of the abstractions in the package. Each abstraction in the package is detailed further in this section.

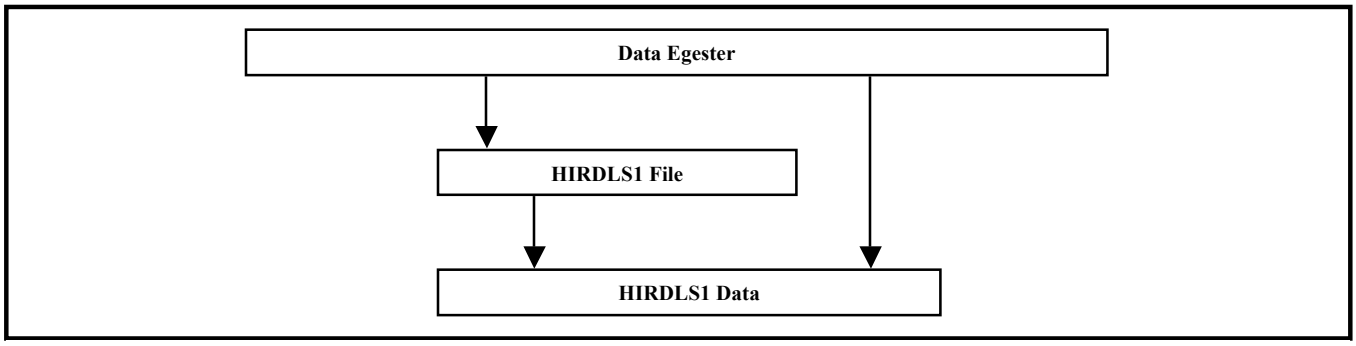

Figure 62 Egester Package Hierarchy

18.1 Data Egester Abstraction

Data Egester is a process abstraction, and has the responsibility to egest (write) a reformed data chunk to a HIRDLS1 file. To fulfill this responsibility, this abstraction collaborates with Reformed Packet, HIRDLS1 Data and HIRDLS1 File, and presents an interface of one Egest contract, as shown in Figure 63. The Egest contract takes a Reformed Packet and adds it to HIRDLS1 Data. When HIRDLS1 Data is full, it is written to HIRDLS1 File, which was created during Data Egester's instantiation. When the Data Egester instance is destroyed, HIRDLS1 Data is checked one last time, and if there is data in it, that data is to be written to HIRDLS1 File. This instantiation must be kept persistent to work correctly.

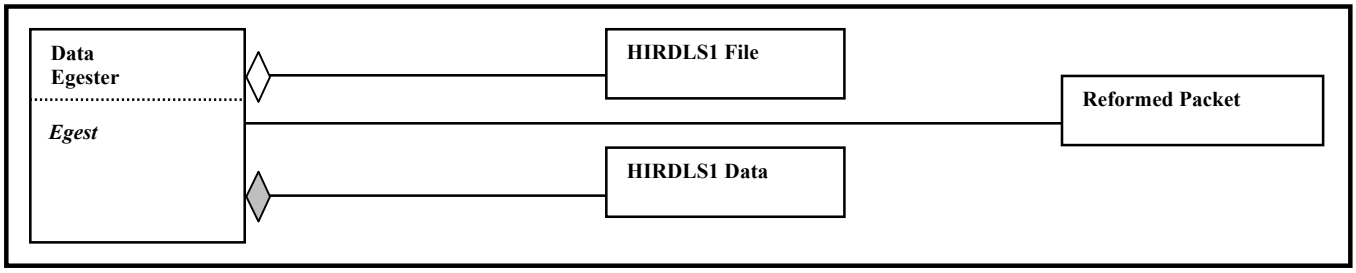

Figure 63 Data Egester Abstraction

18.2 HIRDLS1 File Abstraction

HIRDLS1 File is a control abstraction, and has the responsibility to manage all access to a HIRDLS1 file. To fulfill this responsibility, this abstraction collaborates with HDF5 File, HIRDLS1 Data and Metadata Service, and presents an interface of one GetFile contract and one WriteNext contract, as shown in Figure 64. The GetFile contract returns the one instance of a HIRDLS1 File abstraction. This instance must be a Singleton[9], as there must be exactly one HIRDLS1 File in the system. The WriteNext contract writes the given data chunk into the "next" space in the file. This contract is to return a Boolean status to the caller, indicating if it was able to write the data or not. When the instance of this abstraction is destroyed, Metadata Service calls are made to write the necessary metadata, which has been accumulated during the WriteNext calls, to the file. This abstraction needs to be kept persistent to work correctly, except in the case that the instantiated file is opened, written completely, and closed, in one structural calling sequence.

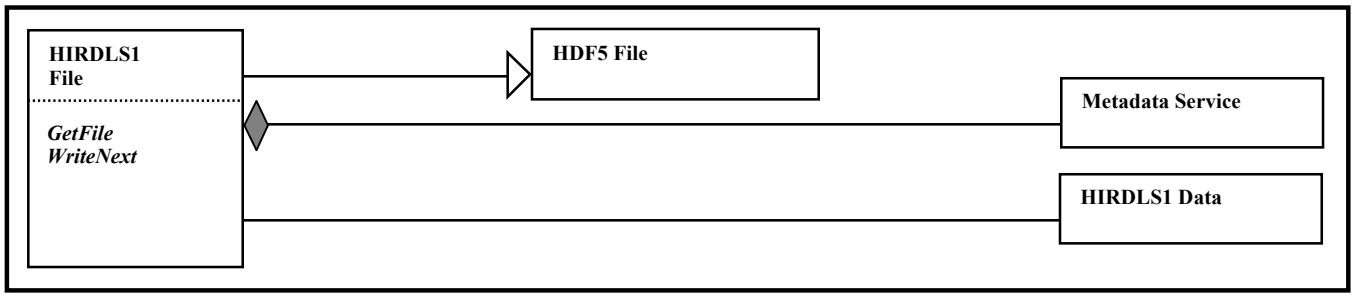

Figure 64 HIRDLS1 File Abstraction

[9] See Section 7.1

18.3 HIRDLS1 Data Abstraction

HIRDLS1 Data is a data abstraction, and has the responsibility to manage access to the data chunk to be written to HIRDLS1 File. To fulfill this responsibility, this abstraction collaborates with Reformed Packet, and presents an interface of one Add contract, one multi-purpose Get contract, one IsFull contract, and one Size contract, as shown in Figure 65. This abstraction also presents a constant value denoting the size of the maximum data it can contain. The copy constructor, assignment operator, or Add contract can be used by HIRDLS1 Data to initialize the abstraction. The Add contract adds data from a Reformed Packet. The Get contract allows retrieval of a particular data field, given a field code. The IsFull contract returns a Boolean denoting if the container is full or not, and the Size contract returns the number of data points in the container.

Figure 65 HIRDLS1 Data Abstraction

19 Writer Package

The Writer package has the responsibility to encapsulate all abstractions necessary to provide the system a means to write the sequenced, geo-located and calibrated data to the HIRDLS1 File. As shown in Figure 1, this package is used by the Processor package, and has access to the Sequencer, Reformer, Egester, File, Service and Diagnostics packages. The only abstraction in this package is Data Writer, and it is detailed further in this section.

19.1 Data Writer Abstraction

Data Writer is a process abstraction, and has the responsibility to manage the reformation and egesting of the sequenced, geo-located and calibrated data. To fulfill this responsibility, this abstraction collaborates with Sequenced Data, Sequenced Packet, Data Reformer, Reformed Packet and Data Egester, and presents an interface of one Write contract, as shown in Figure 66. The Write contract calls, in a loop, Sequenced Data, Data Reformer and Data Egester, to extract a sequenced packet, reform it, and egest it. Since this abstraction fulfills its tasks in one call, it does not need to be persistent to work correctly.

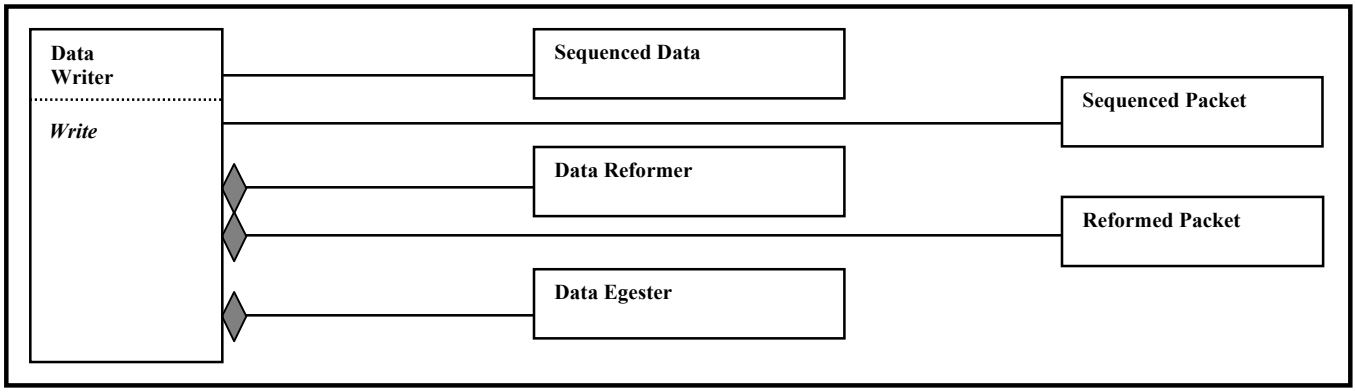

Figure 66 Data Writer Abstraction

20 Processor Package

The Processor package has the responsibility to encapsulate all abstractions necessary to process L0 data into L1 data. As shown in Figure 1, this package is the highest-level package in the system, and has access to the Sequencer, Locator, Calibrator, Writer, File, Service and Diagnostics packages. The only abstraction in this package is L1 Processor, and it is detailed further in this section.

20.1 L1 Processor Abstraction

L1 Processor is a process abstraction, and has the responsibility to manage the processing of HIRDLS L0 data into HIRDLS L1 data. To fulfill this responsibility, this abstraction collaborates with System Reporter, Data Sequencer, Sequenced Data, Data Locator, Data Calibrator and Data Writer, and presents an interface of one Process contract, as shown in Figure 67. The Process contract first instantiates a System Reporter abstraction, and then calls, in sequence: Data Sequencer, Data Locator, Data Calibrator and Data Writer. Since this abstraction fulfills its tasks in one call, it does not need to be persistent to work correctly.

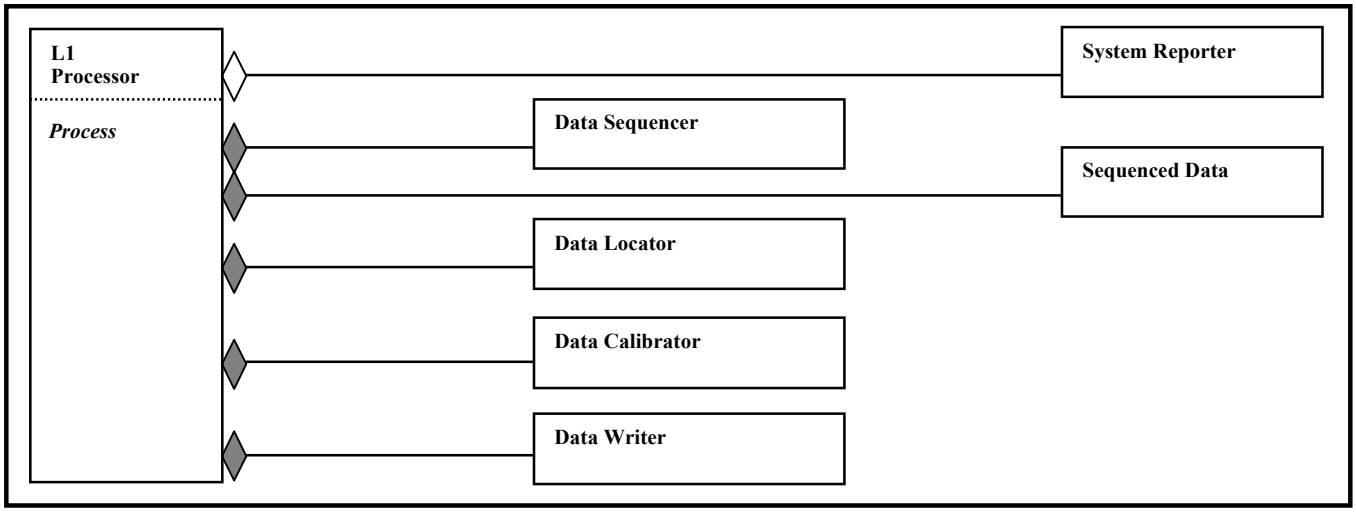

Figure 67 L1 Processor Abstraction

Appendix A – Abstraction Interfaces

A.1 System Reporter

```
static system_reporter* GetReporter ()

~system_reporter ()

void Add (enum diagnostic_code code)
void Add (enum termination_code code, const string& message)
```

A.2 Diagnostic Manager

```
diagnostic_manager ()

~diagnostic_manager ()

static int const MAXIMUM_DIAGNOSTICS

void Add (enum diagnostic_code code)
bool Retrieve (enum diagnostic_code code, diagnostic_data& data) const
```

A.3 Diagnostic Data

```
diagnostic_data ()
diagnostic_data (const diagnostic_data& data)
diagnostic_data& operator= (const diagnostic_data& data)

~diagnostic_data ()

enum diagnostic_code {<the list of acceptable diagnostic codes>}

void Get (enum diagnostic_code& code, long& occurrences, string& message) const
void Set (enum diagnostic_code code, long occurrences, const string& message)
void Update ()
```

A.4 Termination Manager

```
termination_manager ()

~termination_manager ()

void Add (enum termination_code code, const string& message)
void Retrieve (termination_data& data) const
```

A.5 Termination Data

```
termination_data ()
termination_data (const termination_data& data)
termination_data& operator= (const termination_data& data)

~termination_data ()

enum termination_code {TERMINATION_NORMAL, TERMINATION_ABNORMAL}

void Get (enum termination_code& code, string& message) const
void Set (enum termination_code code, const string& message)
```

A.6 Constants Service

```
static int const CHANNEL_SIZE
static int const CHOPPERCYCLE_SIZE
static int const CHOPPERREV_SIZE
static int const MINORFRAME_SIZE
static int const MAXIMUM_MAJORFRAME_SIZE
static int const MAFREV_SIZE

static inline bool IsValidChannelIndex (int index)
static inline bool IsValidChopperCycleIndex (int index)
static inline bool IsValidChopperRevIndex (int index)
static inline bool IsValidMinorFrameIndex (int index)
```

A.7 HDF5 Service

```
hdf5_service ()

~hdf5_service ()

enum hdf5service_fieldtype {HDF5SERVICE_DATAFIELD, HDF5SERVICE_GEOFIELD}

bool CloseFile (long fileid) const
bool CloseSwath (long swathid) const
bool CreateFile (const string& filename, long& fileid) const
bool CreateSwath (long fileid, const string& swathname, long& swathid) const
bool DefineCompression (long swathid, int level, int count, const int dimensions[]) const
bool DefineDimension (long swathid, const string& name, long value) const
bool DefineField (enum hdf5service_fieldtype type, long fileid, const string& fieldname,
                  const string& dimensionnames, char fillvalue, long& swathid) const
bool DefineField (enum hdf5service_fieldtype type, long fileid, const string& fieldname,
                  const string& dimensionnames, short fillvalue, long& swathid) const
bool DefineField (enum hdf5service_fieldtype type, long fileid, const string& fieldname,
                  const string& dimensionnames, int fillvalue, long& swathid) const
bool DefineField (enum hdf5service_fieldtype type, long fileid, const string& fieldname,
                  const string& dimensionnames, long fillvalue, long& swathid) const
bool DefineField (enum hdf5service_fieldtype type, long fileid, const string& fieldname,
                  const string& dimensionnames, float fillvalue, long& swathid) const
bool DefineField (enum hdf5service_fieldtype type, long fileid, const string& fieldname,
                  const string& dimensionnames, double fillvalue, long& swathid) const
bool DefineField (enum hdf5service_fieldtype type, long fileid, const string& fieldname,
                  const string& dimensionnames, unsigned short fillvalue, long& swathid)
                  const
bool GetDimensionSize (long swathid, const string& name, long& size) const
bool GetFieldFillValue (long swathid, const string& name, void* value) const
bool OpenFile (const string& name, long& id) const
bool OpenSwath (long fileid, const string& name, long& id) const
bool ReadField (long swathid, const string& name, int rank, const long starts[],
                const long edges[], void* data) const
bool WriteAttribute (long fileid, const string& name, const string& value) const
bool WriteField (long swathid, const string& fieldname, int dimensioncount,
                 const long starts[], const long edges[], const void* data) const
```

A.8 Time Conversion Service

```
static bool ConvertTAI58ToTAI93 (double tai58, double& tai93) const
static bool ConvertTAI93ToUTC (double tai93, string& utc) const
static bool ConvertUTCToJD (const string& utc, double& jd) const
static bool ConvertUTCToTAI93 (const string& utc, double& tai93) const
```

A.9 Missing Value Service

```
static inline double GetDoubleMissingValue ()
static inline float GetFloatMissingValue ()
static inline int GetIntMissingValue ()
static inline long GetLongMissingValue ()
static inline short GetShortMissingValue ()
static inline unsigned short GetUnsignedShortMissingValue ()
bool IsMissingValue (double value) const
bool IsMissingValue (float value) const
bool IsMissingValue (int value) const
bool IsMissingValue (long value) const
bool IsMissingValue (short value) const
bool IsMissingValue (unsigned short value) const
```

A.10 Program Abortion Service

```
static void Abort () const
static void Abort (const string& message) const
```

A.11 PCF Service

```
static bool GetFilename (int id, string& name) const
static bool GetFilename (int id, int version, string& name) const
static bool GetParameter (int id, string& parameter) const
```

A.12 Metadata Service

```
ecs_service (int datafilelogical, int ecsfilelogical)

~ecs_service ()

bool Set (const string& name, int value)
bool Set (const string& name, const void* value)
bool Set (const string& name, const string& value)
bool Write ()
```

A.13 Processor File

```
processor_file (int id)
processor_file (int id, int version)

~processor_file ()

inline int GetLogical () const
inline string GetName () const
# inline bool IsValid () const
```

A.14 Binary File

```
binary_file (int id, int version)

~binary_file ()

# void Close ()
# bool Open ()
# bool Read (int count, unsigned short words[])
```

A.15 ASCII File

```
ascii_file (int id)

~ascii_file ()

# void Close ()
# bool GetToken (const string& line, int number, double& value) const
# bool GetToken (const string& line, int number, float& value) const
# bool GetToken (const string& line, int number, int& value) const
# bool GetToken (const string& line, int number, string& value) const
# bool Open ()
# bool Read (string& line)
# bool Read (string& line, char commentchar)
```

A.16 HDF5 File

```
hdf5_file (int id, const string& swathname)

~hdf5_file ()

# void Close ()
# bool Create ()
# bool DefineDimension (const string& name, long value)
# bool DefineField (enum hdf5file_fieldtype type, const string& name,
                     const string& dimensionnames)
# bool DefineField (enum hdf5file_fieldtype type, const string& name,
                     const string& dimensionnames, int compressiondimensioncount,
                     int compressiondimensions[])
# bool WriteAttribute (const string& name, const string& value)
# bool WriteField (const string& name, int dimensions, const long starts[],
                   const long edges[], const void* data)
```

A.17 HDF5 Read File

```
hdf5_readfile (int id, const string& swathname)

~hdf5_readfile ()

# void Close ()
# bool GetDimensionSize (const string& dimensionname, long& value) const
# bool Open ()
# bool ReadField (const string& name, int dimensions, const long start[], const long edge[],
                   short& fillvalue, short* data) const
# bool ReadField (const string& name, int dimensions, const long start[], const long edge[],
                   float& fillvalue, float* data) const
# bool ReadField (const string& name, int dimensions, const long start[], const long edge[],
                   int& fillvalue, int* data) const
# bool ReadField (const string& name, int dimensions, const long start[], const long edge[],
                   double& fillvalue, double* data) const
```

A.18 Packet Ingester

```
packet_ingester ()

~packet_ingester ()

bool Ingest (hir0sci_packet& packet)
```

A.19 HIR0Sci File

```
static hir0sci_file* GetFile (int version)

~hir0sci_file ()

bool GetNextPacket (hir0sci_packet& packet)
```

A.20 HIR0Sci Packet

```
hir0sci_packet ()
hir0sci_packet (const hir0sci_packet& packet)
hir0sci_packet& operator= (const hir0sci_packet& packet)

~hir0sci_packet ()

static int const DATAWORDS_SIZE

inline int GetFramecount () const
inline int GetMif () const
inline int GetRDSR () const
inline int GetSHFI () const
void Get (unsigned short datawords[DATAWORDS_SIZE]) const
void GetAzimuth (int& primarycount, int& primaryoffset, int& secondarycount,
                 int& secondaryoffset) const
void GetDiagnostic (int& count, int& offset) const
void GetElevation (int& primaryvarcount, int& primaryvaroffset, int& secondaryvarcount,
                   int& secondaryvaroffset, int& primary2count, int& primary2offset,
                   int& secondary2count, int& secondary2offset) const
void GetGyro (int& g0count, int& g0offset, int& g1count, int& g1offset, int& g2count,
              int& g2offset, int& g3count, int& g3offset) const
void GetHousekeeping (int& count, int& offset) const
void GetRadiance (int& count, int& offset) const
void GetTime (int& count, int& offset) const
void Set (const unsigned short datawords[DATAWORDS_SIZE])
```

A.21 Housekeeping Transformer

```
housekeeping_transformer ()

~housekeeping_transformer ()

bool Transform (const hir0sci_packet& packet, short& orbitfractionid, short& scanmirrorindex,
                short& scantable, int& scanmodeid, float& azimuthhousing1temp,
                float& azimuthhousing2temp, float& calmirror1temp, float& calmirror3temp,
                float& chopperfrequency, float& chopperhousing3temp, float& doormotortemp,
                float& focalplaneatemp, float& focalplanebtemp, float& hotwaxactuatortemp,
                float& ifcfrontplatetemp, float& lens13temp, float& lens23temp,
                float& lensassembly1temp, float& lensassembly2temp, float& mirror13temp,
                float& mirror22temp, float& opticalbenchplatetemp, float& opticalbench2temp,
                float& opticalbench6temp, float& opticalbench7temp, float& scanmirror3temp,
                float& smamountringtemp, float& spacemirror3temp, float& sundoorangle,
                float& sundoortemp, float& sundooraperturetemp,
                float& sundoornegzsurfacetemp, float& sundoorposzsurfacetemp,
                float& sunsensor1temp, float& sunsensor2temp, float& sunsensor3temp,
                short signaloffsets[constants_service::CHANNEL_SIZE])
```

A.22 Housekeeping File

```
static housekeeping_file* GetFile (int shfi, int id)

~housekeeping_file ()

bool Read (housekeeping_data& data)
```

A.23 Housekeeping Data

```
housekeeping_data ()
housekeeping_data (const housekeeping_data& data)
housekeeping_data& operator= (const housekeeping_data& data)

~housekeeping_data ()

static int const ALL_MINORFRAME
static int const COEFFS_MAXSIZE

enum housekeeping_code {<the list of acceptable housekeeping codes>}

void Get (int& shfi, int& minorframe, enum housekeeping_code& code) const
void Get (int& bits, int& order, int& offset, double& converter,
          double coeffs[COEFFS_MAXSIZE]) const
void Get (int& shfi, int& bits, int& order, int& offset, int& minorframe, double& converter,
          double coeffs[COEFFS_MAXSIZE], enum housekeeping_code& code) const
void Set (int shfi, int bits, int order, int offset, int minorframe, double converter,
          const double coeffs[COEFFS_MAXSIZE], enum housekeeping_code code)
```

A.24 Data Transformer

```
data_transformer ()

~data_transformer ()

bool Transform (const hir0sci_packet& hir0scipacket, transformed_packet& transformedpacket)
```

A.25 Azimuth Transformer

```
azimuth_transformer ()

~azimuth_transformer ()

bool Transform (const hir0sci_packet& packet, double
angles[constants_service::CHOPPERREV_SIZE]) const
```

A.26 Radiance Transformer

```
radiance_transformer ()

~radiance_transformer ()

bool Transform (const hir0sci_packet& packet, bool activity[constants_service::CHANNEL_SIZE],
                long signals[constants_service::CHANNEL_SIZE]
                            [constants_service::CHOPPERREV_SIZE]
                            [constants_service::CHOPPERCYCLE_SIZE]) const
```

A.27 Elevation Transformer

```
elevation_transformer ()

~elevation_transformer ()

bool Transform (const hir0sci_packet& packet, double
angles[constants_service::CHOPPERREV_SIZE]) const
```

A.28 Time Transformer

```
time_transformer ()

~time_transformer ()

bool Transform (const hir0sci_packet& packet,
                double hirdlstimes[constants_service::CHOPPERREV_SIZE],
                double tai93times[constants_service::CHOPPERREV_SIZE]) const
```

A.29 Transformed Data

```
transformed_data ()

~transformed_data ()

void Add (const transformed_packet& packet)
bool GetNext (transformed_packet& packet)
```

A.30 Transformed Packet

```
transformed_packet ()
transformed_packet (const transformed_packet& packet)
transformed_packet& operator= (const transformed_packet& packet)

~transformed_packet ()

inline short GetMif () const
inline short GetScanTable () const
inline double GetStartTime () const
inline double GetStopTime () const
void Get (int revindex, double& shaftangleazimuth, double& shaftangleelevation,
          double& hirdlstime, double& taitime) const
void Get (int revindex, short& framecount, short& orbitfractionid, short& scanmirrorindex,
          short& scantable, int& scanmodeid, float& azimuthhousing1temp,
          float& azimuthhousing2temp, float& calmirror1temp, float& calmirror3temp,
          float& chopperfrequency, float& chopperhousing3temp, float& doormotortemp,
          float& focalplaneatemp, float& focalplanebtemp, float& hotwaxactuatortemp,
          float& ifcfrontplatetemp, float& lens13temp, float& lens23temp,
          float& lensassembly1temp, float& lensassembly2temp, float& mirror13temp,
          float& mirror22temp, float& opticalbenchplatetemp, float& opticalbench2temp,
          float& opticalbench6temp, float& opticalbench7temp, float& scanmirror3temp,
          float& smamountringtemp, float& spacemirror3temp, float& sundoorangle,
          float& sundoortemp, float& sundooraperturetemp, float& sundoornegzsurfacetemp,
          float& sundoorposzsurfacetemp, float& sunsensor1temp, float& sunsensor2temp,
          float& sunsensor3temp, double& shaftangleazimuth, double& shaftangleelevation,
          double& hirdlstime, double& tai93time,
          bool signalactivity[constants_service::CHANNEL_SIZE],
          short signaloffsets[constants_service::CHANNEL_SIZE],
          int signals[constants_service::CHANNEL_SIZE]) const
void Get (short& mif, short& framecount, short& orbitfractionid, short& scanmirrorindex,
          short& scantable, int& scanmodeid, float& azimuthhousing1temp,
```

```
          float& azimuthhousing2temp, float& calmirror1temp, float& calmirror3temp,
          float& chopperfrequency, float& chopperhousing3temp, float& doormotortemp,
          float& focalplaneatemp, float& focalplanebtemp, float& hotwaxactuatortemp,
          float& ifcfrontplatetemp, float& lens13temp, float& lens23temp,
          float& lensassembly1temp, float& lensassembly2temp, float& mirror13temp,
          float& mirror22temp, float& opticalbenchplatetemp, float& opticalbench2temp,
          float& opticalbench6temp, float& opticalbench7temp, float& scanmirror3temp,
          float& smamountringtemp, float& spacemirror3temp, float& sundoorangle,
          float& sundoortemp, float& sundooraperturetemp, float& sundoornegzsurfacetemp,
          float& sundoorposzsurfacetemp, float& sunsensor1temp, float& sunsensor2temp,
          float& sunsensor3temp,
          double shaftangleazimuths[constants_service::CHOPPERREV_SIZE],
          double shaftangleelevations[constants_service::CHOPPERREV_SIZE],
          double hirdlstimes[constants_service::CHOPPERREV_SIZE],
          double tai93times[constants_service::CHOPPERREV_SIZE],
          bool signalactivity[constants_service::CHANNEL_SIZE],
          short signaloffsets[constants_service::CHANNEL_SIZE],
          int signals[constants_service::CHANNEL_SIZE][constants_service::CHOPPERREV_SIZE])
         const
void Set (short mif, short framecount, short orbitfractionid, short scanmirrorindex,
          short scantable, int scanmodeid, float azimuthhousing1temp,
          float azimuthhousing2temp, float calmirror1temp, float calmirror3temp,
          float chopperfrequency, float chopperhousing3temp, float doormotortemp,
          float focalplaneatemp, float focalplanebtemp, float hotwaxactuatortemp,
          float ifcfrontplatetemp, float lens13temp, float lens23temp,
          float lensassembly1temp, float lensassembly2temp, float mirror13temp,
          float mirror22temp, float opticalbenchplatetemp, float opticalbench2temp,
          float opticalbench6temp, float opticalbench7temp, float scanmirror3temp,
          float smamountringtemp, float spacemirror3temp, float sundoorangle,
          float sundoortemp, float sundooraperturetemp, float sundoornegzsurfacetemp,
          float sundoorposzsurfacetemp, float sunsensor1temp, float sunsensor2temp,
          float sunsensor3temp,
          const double shaftangleazimuths[constants_service::CHOPPERREV_SIZE],
          const double shaftangleelevations[constants_service::CHOPPERREV_SIZE],
          const double hirdlstimes[constants_service::CHOPPERREV_SIZE],
          const double tai93times[constants_service::CHOPPERREV_SIZE],
          const bool signalactivity[constants_service::CHANNEL_SIZE],
          const short signaloffsets[constants_service::CHANNEL_SIZE],
          const int signals[constants_service::CHANNEL_SIZE]
                         [constants_service::CHOPPERREV_SIZE]) const
```

A.31 Data Flagger

```
data_flagger ()

~data_flagger ()

bool Flag (const transformed_data& transformeddata, flagged_data& flaggeddata)
```

A.32 STXX Flagger

```
stxx_flagger ()

~stxx_flagger ()

bool Flag (short previousscantable, double previousazimuthangle,
           double previousprevious elevationangle, double previouselevationangle,
           double thisazimuthangle, double thiselevationangle, bool& isnominal,
           bool& iskapton, bool& isatmosphere, bool& isazimuthslew, bool& isstare,
           bool& isscanningup, bool& isscanningdown, bool& isscanningright,
           bool& isscanningleft)
```

A.32 ST00 Flagger

```
st00_flagger ()

~st00_flagger ()

inline bool AzimuthMoving (double angle1, double angle2) const
inline bool HasVaidAtmosphericAzimuthAngle (double angle) const
inline bool HasValidKaptonAzimuthAngle (double angle) const
inline bool HasValidNominalScanAzimuthAngle (double angle) const
inline bool ElevationMonotonic (double prepreangle, double preangle, double thisangle) const
inline bool ElevationMoving (double angle1, double angle2) const
inline bool IsScanningDown (double preelevationangle, double thiselevationangle) const
inline bool IsScanningLeft (double preazimuthangle, double thisazimuthangle) const
inline bool IsScanningRight (double preazimuthangle, double thisazimuthnangle) const
inline bool IsScanningUp (double preelevationangle, double thiselevationangle) const
inline bool HasValidElevationAcceleration (double acceleration) const
inline bool HasValidElevationAngle (double angle) const
inline bool HasValidElevationVelocity (double angle1, double angle2) const
inline bool HasValidElevationVelocity (double velocity) const
bool HasValidElevationMovement (double prepreangle, double preangle, double thisangle) const
bool Flag (short previousscantable, double previousazimuthangle,
           double previousprevious elevationangle, double previouselevationangle,
           double thisazimuthangle, double thiselevationangle, bool& isnominal,
           bool& iskapton, bool& isatmosphere, bool& isazimuthslew, bool& isstare,
           bool& isscanningup, bool& isscanningdown, bool& isscanningright,
           bool& isscanningleft)
```

A.33 Flagger Creator

```
static st00_flagger* Create (short scantable)
```

A.34 Flagged Packet

```
flagged_packet ()
flagged_packet (const flagged_packet& packet)
flagged_packet& operator= (const flagged_packet& packet)

~flagged_packet ()

inline double GetHIRDLSTime () const
inline double GetTAITime () const
inline short GetScanTable () const
inline short GetSequenceIndex () const
inline bool IsNominal () const
inline bool IsProcessable () const
inline bool IsUpscan () const
inline void SetNominality (bool nominality)
void Get (bool& isnominal, bool& iskapton, bool& isatmosphere, bool& isazimuthslew,
          bool& isstare, bool& isscanningup, bool& isscanningdown, bool& isscanningright,
          bool& isscanningleft) const
void Get (short& sequenceindex, short& scantable, double& hirdlstime, double& taitime,
          bool& isnominal, bool& iskapton, bool& isatmosphere, bool& isazimuthslew,
          bool& isstare, bool& isscanningup, bool& isscanningdown, bool& isscanningright,
          bool& isscanningleft, bool& isprocessable) const
void Set (short sequenceindex, short scantable, double hirdlstime, double taitime,
          bool isnominal, bool iskapton, bool isatmosphere, bool isazimuthslew,
          bool isstare, bool isscanningup, bool isscanningdown, bool isscanningright,
          bool isscanningleft, bool isprocessable)
```

A.35 Flagged Data

```
flagged_data ()

~flagged_data ()

inline int GetSize () const
inline double GetHIRDLSTime (int index) const
inline double GetTAITime (int index) const
inline short GetScanTable (int index) const
inline bool IsNominal (int index) const
inline bool IsUpscan (int index) const
inline void SetNominality (int index, bool nominality)
void Add (const flagged_packet& packet)
void GetNext (flagged_packet& packet)
```

A.36 Sequence Builder

```
sequence_builder ()

~sequence_builder ()

bool Build (const transformed_data& transformeddata, const flagged_data& flaggeddata,
            sequenced_data& sequenceddata)
```

A.37 Sequenced Data

```
sequenced_data ()

~sequenced_data ()

inline int GetSize () const
void Add (const sequenced_packet& packet)
void Get (int index, sequenced_packet& packet)
void Update (int index, const sequenced_packet& packet)
```

A.38 Sequenced Packet

```
sequenced_packet ()
sequenced_packet (const sequenced_packet& packet)
sequenced_packet& operator= (const sequenced_packet& packet)

~sequenced_packet ()

inline int GetSize () const
inline bool IsFull () const
inline short GetScanNumber () const
void Get (int index, double& azimuthangle, double& elevationangle, double& taitime) const
void Get (int index, float& chopperhousingtemp, float& mirror1temp, float& scanmirrortemp,
          float& spacemirrortemp, bool signalactivity[constants_service::CHANNEL_SIZE],
          short signaloffsets[constants_service::CHANNEL_SIZE],
          int signals[constants_service::CHANNEL_SIZE]) const
void Get (short& framecount, short& orbitfractionid, short& scanmirrorindex,
          short& scantable, int& scanmodeid, float& azimuthhousing1temp,
          float& azimuthhousing2temp, float& calmirror1temp, float& calmirror3temp,
          float& chopperfrequency, float& chopperhousing3temp, float& doormotortemp,
          float& focalplaneatemp, float& focalplanebtemp, float& hotwaxactuatortemp,
          float& ifcfrontplatetemp, float& lens13temp, float& lens23temp,
          float& lensassembly1temp, float& lensassembly2temp, float& mirror13temp,
          float& mirror22temp, float& opticalbenchplatetemp, float& opticalbench2temp,
          float& opticalbench6temp, float& opticalbench7temp, float& scanmirror3temp,
```

```
float& smamountringtemp, float& spacemirror3temp, float& sundoorangle,
float& sundoortemp, float& sundooraperturetemp, float& sundoornegzsurfacetemp,
float& sundoorposzsurfacetemp, float& sunsensor1temp, float& sunsensor2temp,
float& sunsensor3temp, float& solarbetaangle,
bool nominal[constants_service::MINORFRAME_SIZE]
            [constants_service::CHOPPERREV_SIZE],
bool atmosphere[constants_service::MINORFRAME_SIZE]
               [constants_service::CHOPPERREV_SIZE],
bool kapton[constants_service::MINORFRAME_SIZE]
           [constants_service::CHOPPERREV_SIZE],
bool azimuthslew[constants_service::MINORFRAME_SIZE]
                [constants_service::CHOPPERREV_SIZE],
bool stare[constants_service::MINORFRAME_SIZE]
          [constants_service::CHOPPERREV_SIZE],
bool scanningup[constants_service::MINORFRAME_SIZE]
               [constants_service::CHOPPERREV_SIZE],
bool scanningdown[constants_service::MINORFRAME_SIZE]
                 [constants_service::CHOPPERREV_SIZE],
bool scanningleft[constants_service::MINORFRAME_SIZE]
                 [constants_service::CHOPPERREV_SIZE],
bool scanningright[constants_service::MINORFRAME_SIZE]
                  [constants_service::CHOPPERREV_SIZE],
bool spacecraftinshadow[constants_service::MINORFRAME_SIZE]
                       [constants_service::CHOPPERREV_SIZE],
bool mooninfov[constants_service::MINORFRAME_SIZE]
              [constants_service::CHOPPERREV_SIZE],
bool marsinfov[constants_service::MINORFRAME_SIZE]
              [constants_service::CHOPPERREV_SIZE],
bool venusinfov[constants_service::MINORFRAME_SIZE]
               [constants_service::CHOPPERREV_SIZE],
bool jupiterinfov[constants_service::MINORFRAME_SIZE]
                 [constants_service::CHOPPERREV_SIZE],
bool signalactivity[constants_service::CHANNEL_SIZE]
                   [constants_service::MINORFRAME_SIZE],
short scannumber[constants_service::MINORFRAME_SIZE]
                [constants_service::CHOPPERREV_SIZE],
int auraorbitnumber[constants_service::MINORFRAME_SIZE]
                   [constants_service::CHOPPERREV_SIZE],
float solarzenithangle[constants_service::MINORFRAME_SIZE]
                      [constants_service::CHOPPERREV_SIZE],
float solarazimuthangle[constants_service::MINORFRAME_SIZE]
                       [constants_service::CHOPPERREV_SIZE],
float spacecraftlatitude[constants_service::MINORFRAME_SIZE]
                        [constants_service::CHOPPERREV_SIZE],
float spacecraftlongitude[constants_service::MINORFRAME_SIZE]
                         [constants_service::CHOPPERREV_SIZE],
float spacecraftsolarelevation[constants_service::MINORFRAME_SIZE]
                              [constants_service::CHOPPERREV_SIZE],
float tangentpointazimuth[constants_service::MINORFRAME_SIZE]
                         [constants_service::CHOPPERREV_SIZE],
float tangentpointlatitude[constants_service::MINORFRAME_SIZE]
                          [constants_service::CHOPPERREV_SIZE],
float tangentpointlongitude[constants_service::MINORFRAME_SIZE]
                           [constants_service::CHOPPERREV_SIZE],
float radiometricgain[constants_service::CHANNEL_SIZE]
                     [constants_service::MINORFRAME_SIZE]
                     [constants_service::CHOPPERREV_SIZE],
float radiometricoffset[constants_service::CHANNEL_SIZE]
                       [constants_service::MINORFRAME_SIZE]
                       [constants_service::CHOPPERREV_SIZE],
double shaftangleazimuth[constants_service::MINORFRAME_SIZE]
                        [constants_service::CHOPPERREV_SIZE],
double shaftangleelevation[constants_service::MINORFRAME_SIZE]
                          [constants_service::CHOPPERREV_SIZE],
double tai93time[constants_service::MINORFRAME_SIZE]
```

```
                    [constants_service::CHOPPERREV_SIZE],
        double fovrotation[constants_service::MINORFRAME_SIZE]
                      [constants_service::CHOPPERREV_SIZE],
        double localsolartime[constants_service::MINORFRAME_SIZE]
                         [constants_service::CHOPPERREV_SIZE],
        double losazimuth[constants_service::MINORFRAME_SIZE]
                       [constants_service::CHOPPERREV_SIZE],
        double loselevation[constants_service::MINORFRAME_SIZE]
                        [constants_service::CHOPPERREV_SIZE],
        double spacecraftaltitude[constants_service::MINORFRAME_SIZE]
                            [constants_service::CHOPPERREV_SIZE],
        double tangentpointaltitude[constants_service::MINORFRAME_SIZE]
                              [constants_service::CHOPPERREV_SIZE],
        double nearestraypoint[constants_service::MINORFRAME_SIZE]
                         [constants_service::CHOPPERREV_SIZE][3],
        double nearestsurfacepoint[constants_service::MINORFRAME_SIZE]
                            [constants_service::CHOPPERREV_SIZE][3],
        double spacecraftciposition[constants_service::MINORFRAME_SIZE]
                              [constants_service::CHOPPERREV_SIZE][3],
        double spacecraftecrposition[constants_service::MINORFRAME_SIZE]
                              [constants_service::CHOPPERREV_SIZE][3],
        double tangentpointaltitudeoffset[constants_service::CHANNEL_SIZE]
                                   [constants_service::MINORFRAME_SIZE]
                                   [constants_service::CHOPPERREV_SIZE],
        double radiance[constants_service::CHANNEL_SIZE]
                      [constants_service::MINORFRAME_SIZE]
                      [constants_service::CHOPPERREV_SIZE],
        double radianceerror[constants_service::CHANNEL_SIZE]
                         [constants_service::MINORFRAME_SIZE]
                         [constants_service::CHOPPERREV_SIZE]) const
void Set (short framecount, short orbitfractionid, short scanmirrorindex,
        short scantable, int scanmodeid, float azimuthhousing1temp,
        float azimuthhousing2temp, float calmirror1temp, float calmirror3temp,
        float chopperfrequency, float chopperhousing3temp, float doormotortemp,
        float focalplaneatemp, float focalplanebtemp, float hotwaxactuatortemp,
        float ifcfrontplatetemp, float lens13temp, float lens23temp,
        float lensassembly1temp, float lensassembly2temp, float mirror13temp,
        float mirror22temp, float opticalbenchplatetemp, float opticalbench2temp,
        float opticalbench6temp, float opticalbench7temp, float scanmirror3temp,
        float smamountringtemp, float spacemirror3temp, float sundoorangle,
        float sundoortemp, float sundooraperturetemp, float sundoornegzsurfacetemp,
        float sundoorposzsurfacetemp, float sunsensor1temp, float sunsensor2temp,
        float sunsensor3temp, float solarbetaangle,
        const bool nominal[constants_service::MINORFRAME_SIZE]
                        [constants_service::CHOPPERREV_SIZE],
        const bool atmosphere[constants_service::MINORFRAME_SIZE]
                         [constants_service::CHOPPERREV_SIZE],
        const bool kapton[constants_service::MINORFRAME_SIZE]
                       [constants_service::CHOPPERREV_SIZE],
        const bool azimuthslew[constants_service::MINORFRAME_SIZE]
                         [constants_service::CHOPPERREV_SIZE],
        const bool stare[constants_service::MINORFRAME_SIZE]
                      [constants_service::CHOPPERREV_SIZE],
        const bool scanningup[constants_service::MINORFRAME_SIZE]
                         [constants_service::CHOPPERREV_SIZE],
        const bool scanningdown[constants_service::MINORFRAME_SIZE]
                           [constants_service::CHOPPERREV_SIZE],
        const bool scanningleft[constants_service::MINORFRAME_SIZE]
                           [constants_service::CHOPPERREV_SIZE],
        const bool scanningright[constants_service::MINORFRAME_SIZE]
                             [constants_service::CHOPPERREV_SIZE],
        const bool spacecraftinshadow[constants_service::MINORFRAME_SIZE]
                                [constants_service::CHOPPERREV_SIZE],
        const bool mooninfov[constants_service::MINORFRAME_SIZE]
                         [constants_service::CHOPPERREV_SIZE],
```

```
const bool marsinfov[constants_service::MINORFRAME_SIZE]
                    [constants_service::CHOPPERREV_SIZE],
const bool venusinfov[constants_service::MINORFRAME_SIZE]
                     [constants_service::CHOPPERREV_SIZE],
const bool jupiterinfov[constants_service::MINORFRAME_SIZE]
                       [constants_service::CHOPPERREV_SIZE],
const bool signalactivity[constants_service::CHANNEL_SIZE]
                         [constants_service::MINORFRAME_SIZE],
const short scannumber[constants_service::MINORFRAME_SIZE]
                      [constants_service::CHOPPERREV_SIZE],
const int auraorbitnumber[constants_service::MINORFRAME_SIZE]
                         [constants_service::CHOPPERREV_SIZE],
const float solarzenithangle[constants_service::MINORFRAME_SIZE]
                            [constants_service::CHOPPERREV_SIZE],
const float solarazimuthangle[constants_service::MINORFRAME_SIZE]
                             [constants_service::CHOPPERREV_SIZE],
const float spacecraftlatitude[constants_service::MINORFRAME_SIZE]
                              [constants_service::CHOPPERREV_SIZE],
const float spacecraftlongitude[constants_service::MINORFRAME_SIZE]
                               [constants_service::CHOPPERREV_SIZE],
const float spacecraftsolarelevation[constants_service::MINORFRAME_SIZE]
                                    [constants_service::CHOPPERREV_SIZE],
const float tangentpointazimuth[constants_service::MINORFRAME_SIZE]
                               [constants_service::CHOPPERREV_SIZE],
const float tangentpointlatitude[constants_service::MINORFRAME_SIZE]
                                [constants_service::CHOPPERREV_SIZE],
const float tangentpointlongitude[constants_service::MINORFRAME_SIZE]
                                 [constants_service::CHOPPERREV_SIZE],
const float radiometricgain[constants_service::CHANNEL_SIZE]
                           [constants_service::MINORFRAME_SIZE]
                           [constants_service::CHOPPERREV_SIZE],
const float radiometricoffset[constants_service::CHANNEL_SIZE]
                             [constants_service::MINORFRAME_SIZE]
                             [constants_service::CHOPPERREV_SIZE],
const double shaftangleazimuth[constants_service::MINORFRAME_SIZE]
                              [constants_service::CHOPPERREV_SIZE],
const double shaftangleelevation[constants_service::MINORFRAME_SIZE]
                                [constants_service::CHOPPERREV_SIZE],
const double tai93time[constants_service::MINORFRAME_SIZE]
                      [constants_service::CHOPPERREV_SIZE],
const double fovrotation[constants_service::MINORFRAME_SIZE]
                        [constants_service::CHOPPERREV_SIZE],
const double localsolartime[constants_service::MINORFRAME_SIZE]
                           [constants_service::CHOPPERREV_SIZE],
const double losazimuth[constants_service::MINORFRAME_SIZE]
                       [constants_service::CHOPPERREV_SIZE],
const double loselevation[constants_service::MINORFRAME_SIZE]
                         [constants_service::CHOPPERREV_SIZE],
const double spacecraftaltitude[constants_service::MINORFRAME_SIZE]
                               [constants_service::CHOPPERREV_SIZE],
const double tangentpointaltitude[constants_service::MINORFRAME_SIZE]
                                 [constants_service::CHOPPERREV_SIZE],
const double nearestraypoint[constants_service::MINORFRAME_SIZE]
                            [constants_service::CHOPPERREV_SIZE][3],
const double nearestsurfacepoint[constants_service::MINORFRAME_SIZE]
                                [constants_service::CHOPPERREV_SIZE][3],
const double spacecraftecipositon[constants_service::MINORFRAME_SIZE]
                                 [constants_service::CHOPPERREV_SIZE][3],
const double spacecraftecrposition[constants_service::MINORFRAME_SIZE]
                                  [constants_service::CHOPPERREV_SIZE][3],
const double tangentpointaltitudeoffset[constants_service::CHANNEL_SIZE]
                                       [constants_service::MINORFRAME_SIZE]
                                       [constants_service::CHOPPERREV_SIZE],
const double radiance[constants_service::CHANNEL_SIZE]
                     [constants_service::MINORFRAME_SIZE]
```

```
                                [constants_service::CHOPPERREV_SIZE],
          const double radianceerror[constants_service::CHANNEL_SIZE]
                                     [constants_service::MINORFRAME_SIZE]
                                     [constants_service::CHOPPERREV_SIZE]) const
void Update (int index, const float radiometricgain[constants_service::CHANNEL_SIZE],
             const float radiometricoffset[constants_service::CHANNEL_SIZE],
             const double radiance[constants_service::CHANNEL_SIZE],
             const double radianceerror[constants_service::CHANNEL_SIZE])
void Update (int index, bool spacecraftinshadow, bool mooninfov, bool marsinfov,
             bool venusinfov, bool jupiterinfov, int auraorbitnumber, float solarbetaangle,
             float solarazimuthangle, float solarzenithangle, float spacecraftlatitude,
             float spacecraftlongitude, float spacecraftsolarelevation,
             float tangentpointazimuth, float tangentpointlatitude,
             float tangentpointlongitude, double fovrotation, double localsolartime,
             double losazimuth, double loselevation, double spacecraftaltitude,
             double tangentpointaltitude, const double nearestraypoint[3],
             const double nearestsurfacepoint[3], const double spacecrafteciposition[3],
             const double spacecraftecrposition[3],
             const double tangentpointaltitudeoffset[constants_service::CHANNEL_SIZE])
```

A.40 Data Sequencer

```
data_sequencer ()

~data_sequencer ()

bool Sequence (sequenced_data& sequenceddata)
```

A.41 Geolocation Service

```
static double const DEGREES_TO_RADIANS
static double const RADIANS_TO_DEGREES
static double const PI

static void ComputeSolarPositions (double jdtime, double& declination,
                                   double& rightascension)
static bool GetAltitude (const double ecrdetectorunitposition[3],
                         const double ecrspacecraftposition[3], double& altitude)
static bool GetAttitudeAndEphemeris (const string& utctime, double spacecraftposition[3],
                                     double spacecraftvelocity[3])
static bool GetAzimuth (const double ecrboresight[3], double latitude, double longitude,
                        double altitude, double& azimuth)
static bool GetFOVContaminants (const string& utctime, const double fovvector[3],
                                const double perimetervectors[4][3], bool& mooninfov,
                                bool& marsinfov, bool& venusinfov, bool& jupiterinfov)
static bool GetGrazingRay (const double ecrboresight[3], const double ecrposition[3],
                           double& latitude, double& longitude, double& altitude,
                           double nearestraypoint[3], double nearestsurfacepoint[3])
static bool GetOrbitNumber (const string& utctime, int& orbitnumber)
static bool GetQuaternion (const string& utctime, double quaternion[3][3])
static bool GetSlantRange (const double ecrboresight[3], const double ecrspacecraft[3],
                           double& slantrange)
static bool GetSolarBetaAngle (const string& utctime, const double ecispacecraftposition[3],
                               const double ecispacecraftvelocity[3], double& solarbetaangle)
static bool GetSubSatellitePoint (const string& utctime, double& latitude, double& longitude,
                                  double& altitude)
static void GetUnitVector (const double vector[3], double unitvector[3])
```

A.42 Matrix Service

```
static void Multiply (const double matrix1[3][3], const double matrix2[3][3],
                      double result[3][3])
static void Multiply (const double matrix[3][3], const double vector[3], double result[3])
static void Transpose (const double matrix[3][3], double result[3][3])
```

A.43 Data Locator

```
data_locator ()

~data_locator ()

bool Locate (sequenced_data& sequenceddata)
```

A.44 Celestial Body Locator

```
celestialbody_locator ()

~celestialbody_locator ()

bool Locate (const string& utctime, const double rotationmatrix[3][3], bool& mooninfov,
             bool& marsinfov, bool& venusinfov, bool& jupiterinfov) const
```

A.45 Channel Offset Locator

```
channeloffset_locator ()

~channeloffset_locator ()

bool Locate (const string& utctime, double fovrotation, const double rotationmatrix[3][3],
             const double ecrspacecraftposition[3],
             double offsets[constants_service::CHANNEL_SIZE]) const
```

A.46 LOS Locator

```
los_locator ()

~los_locator ()

bool Locate (const string& utctime, const double eciboresightposition[3], double& azimuth,
             double& elevation) const
```

A.47 Spacecraft Locator

```
spacecraft_locator ()

~spacecraft_locator ()

bool Locate (const string& utctime, double& latitude, double& longitude, double& altitude,
             double& solarelevationangle, bool& inearthshadow, double eciposition[3],
             double ecivelocity[3], double ecrposition[3]) const
```

A.48 Tangent Point Locator

```
tangentpoint_locator ()

~tangentpoint_locator ()

bool Locate (const string& utctime, const double ecrspacecraftposition[3],
             const double eciboresightunitvector[3], double& altitude, double& azimuth,
             double& latitude, double& longitude, double& localsolartime,
             double& solarzenithangle, double& solarazimuthangle,
             double nearestecirayypoint[3], double nearestecisurfacepoint[3]) const
```

A.49 Offset Generator

```
radiometricoffset_generator ()

~radiometricoffset_generator ()

bool Generate (float chopperhousingtemp, float mirror1temp, float scanmirrortemp,
               float spacemirrortemp,
               const short signaloffset[constants_service::CHANNEL_SIZE],
               const float radiometricgain[constants_service::CHANNEL_SIZE],
               float radiometricoffset[constants_service::CHANNEL_SIZE]) const
```

A.50 Flux Data

```
flux_data ()
flux_data (const flux_data& data)
flux_data& operator= (const flux_data& data)

~flux_data ()

static int const FLUX_SIZE

void Get (int& fluxsize, int temperatures[FLUX_SIZE],
          double flux[FLUX_SIZE][constants_service::CHANNEL_SIZE]) const
void Set (int fluxsize, const int temperatures[FLUX_SIZE],
          const double flux[FLUX_SIZE][constants_service::CHANNEL_SIZE])
```

A.51 Flux File

```
static flux_file* GetFile ()

~flux_file ()

bool Read (flux_data& data)
```

A.52 OOF Corrector

```
oof_corrector ()

~oof_corrector ()

bool Correct (const bool signalactivity[constants_service::CHANNEL_SIZE],
              const int signals[constants_service::CHANNEL_SIZE],
              int correctedsignals[constants_service::CHANNEL_SIZE]) const
```

A.53 Correction Data

```
correction_data ()
correction_data (const correction_data& data)
correction_data& operator= (const correction_data& data)

~correction_data ()

void Get (float weights[constants_service::CHANNEL_SIZE][constants_service::CHANNEL_SIZE)
         const
void Set (const float weights[constants_service::CHANNEL_SIZE]
                             [constants_service::CHANNEL_SIZE)
```

A.54 Correction File

```
static correction_file* GetFile ()

~correction_file ()

bool Read (correction_data& data)
```

A.55 Data Calibrator

```
data_calibrator ()

~data_calibrator ()

bool Calibrate (sequenced_data& sequenceddata)
```

A.56 Calibration Data

```
calibration_data ()
calibration_data (const calibration_data& data)
calibration_data& operator= (const calibration_data& data)

~calibration_data ()

void Get (float gains[constants_service::CHANNEL_SIZE],
          float kterms[constants_service::CHANNEL_SIZE],
          float calmirroremissivities[constants_service::CHANNEL_SIZE],
          float chopperemissivities[constants_service::CHANNEL_SIZE]) const
void Set (const float gains[constants_service::CHANNEL_SIZE],
          const float kterms[constants_service::CHANNEL_SIZE],
          const float calmirroremissivities[constants_service::CHANNEL_SIZE],
          const float chopperemissivities[constants_service::CHANNEL_SIZE])
```

A.57 Calibration File

```
static parameter_file* GetFile ()

~calibration_file ()

bool Read (calibration_data& data)
```

A.58 Data Reformer

```
data_reformer ()

~data_reformer ()

bool Reform (const sequenced_packet& sequencedpacket, reformed_packet& reformedpacket)
```

A.59 Reformed Packet

```
reformed_packet ()
reformed_packet (const reformed_packet& packet)
reformed_packet& operator= (const reformed_packet& packet)

~reformed_packet ()

void Get (short& orbitposition, short& scantable, short& scanmirrorindex,
          short& solarazangle, short& solarzenangle, short& viewdirection,
          short& chopperperiod, short& framecounter, int& scanmodeid, float& solarbetaangle,
          float& azhousingtemp, float& calmirror1temp, float& calmirror3temp,
          float& chopperhousingtemp, float& doormotortemp, float& focalplaneatemp,
          float& focalplanebtemp, float& hotwaxactuatortemp, float& ifcfrontplatetemp,
          float& lens2temp, float& lenshousingtemp, float& mirror1temp, float& mirror2temp,
          float& opticalbenchplatetemp, float& opticalbench2temp, float& opticalbench6temp,
          float& opticalbench7temp, float& scanmirrortemp, float& smamountringtemp,
          float& spacemirrortemp, float& sundoornegzsurfacetemp,
          float& sundoorposzsurfacetemp, float& sundooraperturetemp, float& sundoorangle,
          float& sundoortemp, float& sunsensor1temp, float& sunsensor2temp,
          float& sunsensor3temp, int scecrposition[3],
          int orbitnumber[constants_service::MINORFRAME_SIZE],
          float radscaleoffset[constants_service::CHANNEL_SIZE],
          float radscalefactor[constants_service::CHANNEL_SIZE],
          float radgain[constants_service::CHANNEL_SIZE],
          float radoffset[constants_service::CHANNEL_SIZE],
          short azlosangle[constants_service::MINORFRAME_SIZE]
                          [constants_service::CHOPPERREV_SIZE],
          short fieldrotation[constants_service::MINORFRAME_SIZE]
                             [constants_service::CHOPPERREV_SIZE],
          short scannumber[constants_service::MINORFRAME_SIZE]
                          [constants_service::CHOPPERREV_SIZE],
          int altitude[constants_service::MINORFRAME_SIZE]
                      [constants_service::CHOPPERREV_SIZE],
          int flags[constants_service::MINORFRAME_SIZE]
                   [constants_service::CHOPPERREV_SIZE],
          int localsolartime[constants_service::MINORFRAME_SIZE]
                            [constants_service::CHOPPERREV_SIZE],
          int ellosangle[constants_service::MINORFRAME_SIZE]
                        [constants_service::CHOPPERREV_SIZE],
          int spacecraftalt[constants_service::MINORFRAME_SIZE]
                           [constants_service::CHOPPERREV_SIZE],
          float azshaftangle[constants_service::MINORFRAME_SIZE]
                            [constants_service::CHOPPERREV_SIZE],
          float elshaftangle[constants_service::MINORFRAME_SIZE]
                            [constants_service::CHOPPERREV_SIZE],
          float latitude[constants_service::MINORFRAME_SIZE]
                        [constants_service::CHOPPERREV_SIZE],
          float longitude[constants_service::MINORFRAME_SIZE]
                         [constants_service::CHOPPERREV_SIZE],
          float scsolarelangle[constants_service::MINORFRAME_SIZE]
                              [constants_service::CHOPPERREV_SIZE],
          float spacecraftlat[constants_service::MINORFRAME_SIZE]
                             [constants_service::CHOPPERREV_SIZE],
          float spacecraftlon[constants_service::MINORFRAME_SIZE]
```

```
                        [constants_service::CHOPPERREV_SIZE],
        double taitime[constants_service::MINORFRAME_SIZE]
                      [constants_service::CHOPPERREV_SIZE],
        short raderror[constants_service::CHANNEL_SIZE]
                      [constants_service::MINORFRAME_SIZE],
        int nearestraypoint[constants_service::MINORFRAME_SIZE]
                        [constants_service::CHOPPERREV_SIZE][3],
        int nearestsurfacepoint[constants_service::MINORFRAME_SIZE]
                            [constants_service::CHOPPERREV_SIZE][3],
        int sceciposition[constants_service::MINORFRAME_SIZE]
                      [constants_service::CHOPPERREV_SIZE][3],
        short altitudeoffset[constants_service::CHANNEL_SIZE]
                         [constants_service::MINORFRAME_SIZE]
                         [constants_service::CHOPPERREV_SIZE],
        unsigned short radiance[constants_service::CHANNEL_SIZE]
                            [constants_service::MINORFRAME_SIZE]
                            [constants_service::CHOPPERREV_SIZE]) const
void Set (short orbitposition, short scantable, short scanmirrorindex,
        short solarazangle, short solarzenangle, short viewdirection,
        short chopperperiod, short framecounter, int scanmodeid, float solarbetaangle,
        float azhousingtemp, float calmirror1temp, float calmirror3temp,
        float chopperhousingtemp, float doormotortemp, float focalplaneatemp,
        float focalplanebtemp, float hotwaxactuatortemp, float ifcfrontplatetemp,
        float lens2temp, float lenshousingtemp, float mirror1temp, float mirror2temp,
        float opticalbenchplatetemp, float opticalbench2temp, float opticalbench6temp,
        float opticalbench7temp, float scanmirrortemp, float smamountringtemp,
        float spacemirrortemp, float sundoornegzsurfacetemp,
        float sundoorposzsurfacetemp, float sundooraperturetemp, float sundoorangle,
        float sundoortemp, float sunsensor1temp, float sunsensor2temp,
        float sunsensor3temp, const int scecrposition[3],
        const int orbitnumber[constants_service::MINORFRAME_SIZE],
        const float radscaleoffset[constants_service::CHANNEL_SIZE],
        const float radscalefactor[constants_service::CHANNEL_SIZE],
        const float radgain[constants_service::CHANNEL_SIZE],
        const float radoffset[constants_service::CHANNEL_SIZE],
        const short azlosangle[constants_service::MINORFRAME_SIZE]
                           [constants_service::CHOPPERREV_SIZE],
        const short fieldrotation[constants_service::MINORFRAME_SIZE]
                              [constants_service::CHOPPERREV_SIZE],
        const short scannumber[constants_service::MINORFRAME_SIZE]
                           [constants_service::CHOPPERREV_SIZE],
        const int altitude[constants_service::MINORFRAME_SIZE]
                        [constants_service::CHOPPERREV_SIZE],
        const int flags[constants_service::MINORFRAME_SIZE]
                      [constants_service::CHOPPERREV_SIZE],
        const int localsolartime[constants_service::MINORFRAME_SIZE]
                            [constants_service::CHOPPERREV_SIZE],
        const int ellosangle[constants_service::MINORFRAME_SIZE]
                         [constants_service::CHOPPERREV_SIZE],
        const int spacecraftalt[constants_service::MINORFRAME_SIZE]
                            [constants_service::CHOPPERREV_SIZE],
        const float azshaftangle[constants_service::MINORFRAME_SIZE]
                            [constants_service::CHOPPERREV_SIZE],
        const float elshaftangle[constants_service::MINORFRAME_SIZE]
                            [constants_service::CHOPPERREV_SIZE],
        const float latitude[constants_service::MINORFRAME_SIZE]
                         [constants_service::CHOPPERREV_SIZE],
        const float longitude[constants_service::MINORFRAME_SIZE]
                          [constants_service::CHOPPERREV_SIZE],
        const float scsolarelangle[constants_service::MINORFRAME_SIZE]
                              [constants_service::CHOPPERREV_SIZE],
        const float spacecraftlat[constants_service::MINORFRAME_SIZE]
                              [constants_service::CHOPPERREV_SIZE],
        const float spacecraftlon[constants_service::MINORFRAME_SIZE]
                              [constants_service::CHOPPERREV_SIZE],
```

```
const double taitime[constants_service::MINORFRAME_SIZE]
                    [constants_service::CHOPPERREV_SIZE],
const short raderror[constants_service::CHANNEL_SIZE]
                    [constants_service::MINORFRAME_SIZE],
const int nearestraypoint[constants_service::MINORFRAME_SIZE]
                         [constants_service::CHOPPERREV_SIZE][3],
const int nearestsurfacepoint[constants_service::MINORFRAME_SIZE]
                             [constants_service::CHOPPERREV_SIZE][3],
const int sceciposition[constants_service::MINORFRAME_SIZE]
                       [constants_service::CHOPPERREV_SIZE][3],
const short altitudeoffset[constants_service::CHANNEL_SIZE]
                          [constants_service::MINORFRAME_SIZE]
                          [constants_service::CHOPPERREV_SIZE],
const unsigned short radiance[constants_service::CHANNEL_SIZE]
                             [constants_service::MINORFRAME_SIZE]
                             [constants_service::CHOPPERREV_SIZE])
```

A.60 Data Egester

```
data_egester ()

~data_egester ()

bool Egest (const reformed_packet& reformedpacket)
```

A.61 HIRDLS1 File

```
static l1b_file* GetFile ()

~l1b_file ()

bool WriteNext (const hirdls1_data& hirdls1data)
```

A.62 HIRDLS1 Data

```
hirdls1_data ()
hirdls1_data (const hirdls1_data& data)
hirdls1_data& operator= (const hirdls1_data& data)

~hirdls1_data ()

static int const MAJORFRAME_WRITESIZE

enum hirdls1data_code {<the list of acceptable hirdls1 data codes>}

inline bool IsFull () const
inline int  Size () const

void Add (const reformed_packet& reformedpacket)
const void* Get (enum hirdls1data_code) const
```

A.63 Data Writer

```
data_writer ()

~data_writer ()

bool Write (const sequenced_data& sequenceddata)
```

A.64 L1 Processor

```
l1_processor ()

~l1_processor ()

void Process ()
```