INFORMATION TRANSFER USING THE SFDU

Fred Billingsley, John Johnson, Ed Greenberg, Merv MacMedan
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California, 91109
USA

OVERVIEW

A primary focus of Information Systems activities is to provide the techniques and tools to enable scientists to locate, acquire, and utilize data in the continuing search for understanding. This requires the interfacing of a myriad of discipline oriented data centers and data archives, providing access to their data sets and supporting tools and services.

The growth in scientific understanding is based on our ability to transfer acquired information. At present, the science user community is diverse but associated, is discipline focused, and utilizes heterogenous processors. This places stringent data documentation requirements upon the users to facilitate a common understanding of the transferred data. This common understanding (multi-discipline focused) will only be achieved after the diverse disciplines begin to document their semantic definitions.

Information can be defined as the communication or reception of knowledge or intelligence. In the context of this document, information transfer must include not only the science data bits, but also the ancillary data needed for meaningful analysis. In addition all data must be uniquely identified, and its syntax and semantics rigorously defined and included.

The Consultative Committee for Space Data Systems (CCSDS) is an internat-onal committee which is defining a system to facilitate this information transfer among the various space agencies and experimenters of the member countries. The system - the Standard Formatted Data Unit system - outlines recommendations for data structure description, format registration, and associated services. [1]

INFORMATION TRANSFER - THE SFDU CONCEPT

The SFDU Concept provides:

1) a means for globally defining and identifying data products,
2) a means for aggregating instances of science, ancillary and meta data into data products, and
3) a means for administering the data product definitions and description to ensure their accessability and understanding.

The SFDU methodology promotes documentation rigor through the administrative services provided by the CCSDS Member Agency Control Authorities. The data registration procedures establish a global data identification mechanism, which, combined with standard data labelling and aggregation conventions, enables the self identification process needed to support meaningful data interchange. The SFDU concept focuses on the standard

labeling of data to enhance the transmission, storage, and manipulation of the data contained therein. The contents may be in any arrangement that can be expressed in a precise way.

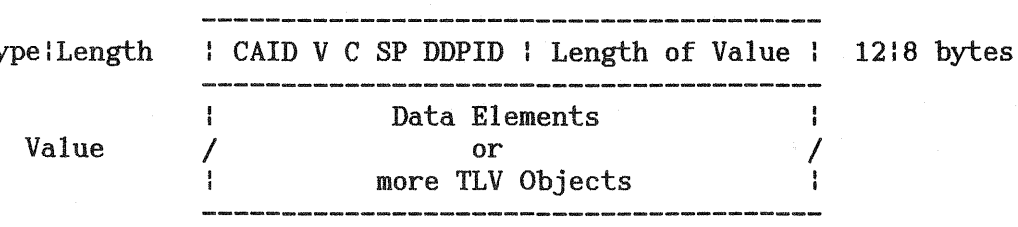
In fulfilling the above provisions, the SFDU system is designed to provide three types of services:

- The SFDU Structure Services
  - Standardized labeling techniques
  - Standardized aggregation discipline
- Data Services
  - Standard format description technique
  - Standardized interface to recipient
  - Methods of numerical data conversion
- Information Services
  - Standard method for registering and disseminating:
    - Format descriptions
    - Vocabulary and dictionaries
    - Related data

Taxonomy and Structuring

The taxonomy of information transfer ranges from single data elements to completely identified and defined products. A data element is an individually named item of data that is used in a processing algorithm. Elements are collected and structured into data objects (aggregations of elements or groups) and units (aggregations of objects) with identifying SFDU labels. Data units range in complexity from simple messages to entire collections of space-acquired data plus ancillary data from a mission. A data product consists of units containing not only the data, but data formats and representations, data element dictionaries, cataloging information, etc.

The fundamental structure used for carrying the various kinds of data is a Type-Length-Value Object (TLVO). This is a self identified and self delimited data object which follows CCSDS labelling rules. It consists of a fixed length label followed by a variable length value field. The basic structure of the object is given in the figure below. The two fields of the label are: a) the TYPE field (which includes the reference name of the description of the value field) and, b) the LENGTH field (which provides the length of the value field). The value field may contain data elements or embedded TLVOs.



Structured TLV Data Object

In the TYPE field, the Control Authority Identifier (CAID) identifies the CA office which maintains the format definition. The VERSION (V) field

gives the structure of the label, the CLASS (C) field gives a high level classification of the content of the value field, and SP denotes two spare characters. The Data Definition Package Identifier (DDPID) uniquely defines (within a CA) the package which contains the complete definition of the data object. The CAID and the DDPID together (called the Control Authority and Data Definition Package Identifier or ADI) provide globally unique identification and definition of the object.

When the value field consists of data elements, the ADI identifies the appropriate Data Description Package (DDP) describing the (user) data. Alternately, it may contain additional TLVOs, in which case the ADI identifies the label processor. TLVOs may be nested arbitrarily deeply, forming data products or Standard Formatted Data Units (SFDU).

If a data product, consisting of several classes of objects where the total length is not known, is to be created, then aggregation by reference is used. There are several such methods available, as well as the aggregation by envelope used when the length is known.

Data Definition

The Data Definition Package (DDP) structure and content is the subject of current study by the CCSDS. The intent is to supply the data product user with the conceptual or logical description of the data as well as the format and representation of the data. This information will be packaged with the data such that a suite of standard software, conforming to SFDU recommendations, at the user's installation can transform the data to conform to his machine architecture and can present standard views for applications.

The content of the DDP will include categories of information such as the following:

1) Data Entity Dictionaries (DED) which include semantic information.

2) Data Definition Records (DDR) which contain the data object formats and representations.

The DDP can be kept in a library and accessed as needed through the ADI of a TLVO being processed. In the case of archives, the DDP information may be kept with the data and sent along with an order.

Data Product Formation

Data products are structures arranged according to a specific taxonomy to improve one's ability to understand and process the information contained therein. To facilitate aggregation, identification, and definition, standard labels are employed to form a succession of Standard Formatted Data Units (SFDUs) and TLVOs into a product. The term data product is synonymous with data unit although data product normally refers to a larger collection of data.

The method for formulating a data product for transfer is to assemble all of the required data in the desired order and construct an "envelope" or container that aggregates the combination, binding the enclosed data into a

named and delimited data product. It is required that the labeling technique utilized in the "envelope" be globally recognizable and interpretable to ensure that the contents of the data product are readable.

SFDU construction rules have been defined to define how aggregations of TLVOs can be assembled. This enables data products to be assembled from "standardized" TLVO data objects (modules) such as headers, DDPs, and ancillary data objects which are specially formulated to supply the necessary data required by a system service (e.g. data catalog systems).

See References [2],[3],[4] for a formal description of the construction rules.

The various types of data which might be transferred may be grouped into (at least) four general categories:

Selection (catalog search) items
The main data file
Ancillary data pertaining to the main file
Data structure items

Each of these categories involve the system services in potentially unique ways.

Selection - these include information about the data instance which will be used in catalogs and which will be searched during the process of finding a file to meet specific criteria. They will typically be included with the main file and passed to the catalog function without modification.

Main Data - the data file which is the subject of the transfer.

Ancillary - This is information of use to the experimenter in interpreting the main data or to any SFDO with which it is associated. This includes data such as conditions of observation of the main file data, calibrations, units. It typically will not invoke any services action, but is available on request. Some of these, such as predefined units, may be found in a data element dictionary and not passed with the main file. Others, such as engineering parameters, are more instance-specific and may be passed with the main file.

Data definition package - This includes two types of information: 1) ancillary definitions pertaining to the data items (the data element dictionary), and 2) format description items. These latter may take two basic forms: 1) items which completely describe the format of the SFDO with which it is associated; 2) items which complete a generic format description, to be passed to the generic format at run time. The former items are typically found in a format library and not passed with the data file; the latter, such as array sizes, are passed with the data file to complete the format definition for the specific instance involved.

FORMAT DESCRIPTION - THE TSDL

A data description language, the Transfer Syntax Description Language (TSDL) is being developed as one approach toward describing the data files. [5],[6] Following is a brief overview of the intended TSDL capabilities.

These capabilities, not found in other data descriptions, serve as the reason for developing the TSDL.

TSDL is conceived as a media-independent, content-independent tool for the transfer of information between dissimilar computer systems. It is NOT a tool for the internal processing of information. It does not require the insertion of data field terminators, or any change in a user data file, and thus may be used to describe archived files. Machine numerical forms may be used and described, without modification. It permits the sender to describe the transferred information and to send this description separately or as an integral part of the transfer file. It permits the description of both character and bit field information in fixed- (without delimiters) or variable-width (delimited) fields or subfields. It further permits the identification of fields and subfields by arbitrarily long names and labels which serve to give meaning to the data. In addition, it provides for the definition and labeling of complex structures and commutated data.

TSDL Structure

Punctuation symbols used are as follows:

< >	indicates a logical entity
[ ]	indicates optionally present
{ }	indicates optional repetition
( )	indicates grouping

The TSDL Module consists of Core, Extension, and Data records. The Core contains information about the module and data as a whole, and the Extension carries the desriptions of the data fields and their interrelationships.

<Module> ::= <Core> [<Extension>] [{<Data1>}] ... [{<Datan>}]

The Core and Extension records each consist of a seies of segments having a single Backus-Nauer (BNF) form:

<Core> ::= {}

<Extension> ::= {}.

Each segment consists of a Length-Type-Value series of fields:

 ::= <Length> <Tag> IS1 <SegValue> ISn where

Tag is the segment Type (Name)
SegValue is the Segment data contents
IS(1-4) is an ASCII information separator
The Length field (2 bytes) is the length of the Segment, from the [Tag] to the IS, inclusive.

The TSDL may be considered as Keyword-driven, where the TSDL keywords are the segment tags. A standard, recognizable, group of segment tags is specified in the TSDL, from which a given instance may be assembled. This allows the building of a TSDL Module from a relatively small group of specification-defined Tags plus user-defined Tags. Similarly, the

user may define keywords (Labels) for the data fields of the user records. These fields are described by the TSDL and may be located by application software using the labels as keywords. Thus, only those Tags and Labels necessary for the instance need be included. This approach provides a more flexible and extensible descriptive form than pre-defined descriptive formats.

Data Field Structure Description

The philosophy behind the structure definitions is that in a transmitted series of bytes, there is no inherent logical structure, either to the grouping of the bytes or to any numerical or logical forms recognized by computers. Therefore, everything must be defined.

The data field structures are described in a series of entries called Type Definitions which are related to the corresponding data fields through labels as follows:

New variables, arrays and complex data field structures may be defined once in a Structure segment or a Type segment and subsequently used:

STRUCT IS1 <label> , [<type>] : {<Type Definition>)}
TYPE IS1 ,{(<label>,<Type Definition>)}

Attributes for the variables or data fields may be carried in a Domain Segment.

Type definitions describe the Integer, Real, Character, or other form of the data field. Type definitions are nested in the sense that definitions of previously-defined structures or fields may be included by reference, using a preceding asterisk, in the definitions:

STRUCT IS1 <Label> , : [{*<Label1>}] [{<Type Definition>}]

where Label1 is a previously-defined Label.

Record Format segments are used to describe data records, and are structured as:

RECFMT IS1 <Xref> , {(<Label>,<Width>,<Offset>,<Type Definition>)}

Xref is the identifier of the data record being described.
Labels are the user-defined field or other aggregate labels.
Width is the width of a data field.
Offset is the offset of the field from the beginning of the data record.

The structures of externally-defined fields may be referenced using the EXAF (External Authority and Format) segment:

EXAF IS1 {<Label> , <Authority> , <Format ID>} where

Label is the user-defined label for this instance,
Authority is the external authority being referenced, and
Format ID is its format reference in the external definition.

Dataflds

The DATAFLDS segment contains an optionally parenthesized list of the user data field labels, to whatever depth the user desires, allowing a nested field(subfield) structure definition. The allowable set of labels are those specified in the user application specification. The same labels are used in the TYPE, RECFMT, STRUCT, EXAF, DOMAIN, and the logical description segments to allow correlation of the various descriptions.

Hierarchical and Network Stsuctures

The parenthesized Dataflds form will describe a field-subfield type of hierarchical structure. An alternate method of description is to provide a list of node labels in a preorder traverse sequence from a single root representing the entire section of data, plus an ordered list of the last node of preorder traverse sequences beginning at each node (including leaf nodes). These two sequences are carried in the TRAV(erse) and LASTNODE segments.

Network structures may be described by cutting the structure into a hierarchical tree, and sending this plus a list of the cut links, using the LINKLIST Segment.

Relational Structures

Relational structures may be decomposed into a set of orthogonal relational tables. The structure of the lines of these tables may be described using the Structure segment. The column headings may be given as labels in a Dataflds segment, and are the data entry names.

```
STRUCT:         <TableLabel>,Table:<Typdef1>,<Typdef2>,...,<Typdefn>
DATAFLDS:<Xref>,<TableLabel>(<ColHdg1>,<ColHdg2>, ... ,<ColHdgn>)
RECFMT:  <Xref>,*<TableLabel>              where
```

the DATAFLDS and RECFMT Xref field refers to data records by that name, and the RECFMT form indicates that each row has the form specified in the TableLabel Struct segment.

Machine and language independence will be accomplished by: 1) defining the transfer as a series of bytes, thus eliminating media byte-interchange problems such as the VAX vs IBM tape formats; 2) providing methods for defining binary data fields such as machine representations in such a way that suitable new target representations may be constructed; 3) defining a canonical interface as a pair of ASCII tables which describe the data records in such a manner that data fields may be located, read, and converted to the desired representations on the target machine, using the desired target programming language or DBMS.

The canonical interface is not part of the TSDL specification, as it is expected that the application will wish to define this interface to suit the language being used. The task of the TSDL is to transmit the information required to build the tables. The canonical ASCII interface tables are expected to have contents such as the following:

Segment Table (for each segment)

Segment Tag Segment contents verbatim

Access Table (for each data field)

Record Label Field Label Structure Width Position Type Definitions

The total set of capabilities, from the consistent segment structures, through the type definition techniques, through the canonical interface, will constitute a unique and new tool for the systematic transfer of data. With it available, local software which will be needed to convert user files to and from the canonical interface will be appreciably simplified. This software on each end may be independent, one end from the other, thus reducing the $O(n^2)$ problem to an $O(2n)$ problem.

ACKNOWLEDGEMENT

The material reported in this paper has been sponsored by the Office of Tracking and Data Acquisition (Code TS) and the Office of Space Science and Applications Information Systems (Code EI) of NASA Headquarters, and the work performed at the Jet Propulsion Laboratory under NASA contract.

REFERENCES

[1] "Report Concerning Space Data Systems Standards: Space Data Systems Operation with Standard Formatted Data Units: Systems and Implementation Aspects", CCSDS 610.0-G-5, Green Book, Issue 5, February 1987 or later issue.

[2] "Standard Formatted Data Units - Structure and Construction Rules", Recommendation CCSDS 620.0-B-1, Blue Book, Consultative Committee for Space Data Systems, December 1987 or later issue.

[3] "Standard Formatted Data Units - Structure and Construction Rules: Volume II", Recommendation CCSDS 610.TBD, White Book, March 1988 or later issue.

[4] "Operations with Standard Formatted Data Units: Product Aggregation Aspects", Green Book, Issue-1, March 1988

[5] "Transfer Syntax Description Language (General Data Interchange Language)", Proposed White Book, May, 1988

[6] "Transfer Syntax Description Language, Supporting Documents", Proposed Green Book, May 1988