Consultative Committee for Space Data Systems

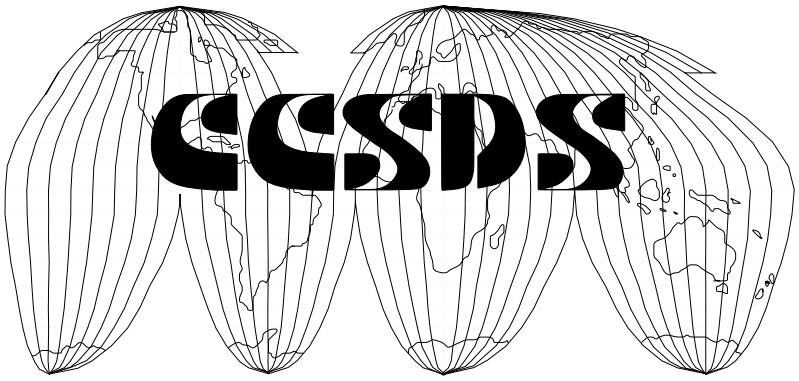
REPORT CONCERNING SPACE DATA SYSTEM STANDARDS

LANGUAGE USAGE IN INFORMATION INTERCHANGE TUTORIAL

CCSDS 642.1-G-1

GREEN BOOK

October 1989



AUTHORITY

Issue:	Green Book, Issue-1
Date:	October 1989
Location:	CCSDS Panel 2 meeting, September 1989 Ottawa, Ontario, Canada

This report reflects the consensus technical understanding of the Panel 2 members representing the following member Agencies of the Consultative Committee for Space Data Systems (CCSDS):

- British National Space Center (BNSC)/United Kingdom
- Canadian Space Agency (CSA)/Canada
- Centre National D'Etudes Spatiales (CNES)/France
- Deutsche Luft and Raumfahrt e.V (DLR)/FRG
- European Space Agency (ESA)/Europe
- Indian Space Research Organization (ISRO)/India
- Instituto de Pesquisas Espaciais (INPE)/Brazil
- National Aeronautics and Space Administration (NASA)/USA
- National Space Development Agency of Japan (NASDA)/Japan

This report is published and maintained by:

CCSDS Secretariat
Communications and Data Systems Division, (Code-TS)
National Aeronautics and Space Administration
Washington, DC 20546, USA

FOREWORD

This document is a tutorial on the use of languages for descriptive purposes in information interchange. It has been prepared by the Consultative Committee on Space Data Systems (CCSDS) Panel 2, Standard Data Interchange Structures (SDIS).

This document discusses some of the challenges involved with the interchange of information in the international space community. It assumes the use of the standard formatted data unit (SFDU), one of several CCSDS recommendations, as a methodology for information interchange.

This document evaluates the choice and usage of languages in each of three interchange categories. These categories reflect two factors involving the relationship between the producing and receiving environments:

- Capability for communication between environments
- Degree of dissimilarity between the environments

Based on these two factors, interchanges are classified into three categories: closed, negotiated, and open. Examples are provided to illustrate these categories and highlight the language considerations involved with each category.

DOCUMENT CONTROL

Document No.	Title	Date	Status/ Remarks
CCSDS 642.1-G-l	Report Concerning Space Data System Standards: Language Usage in Information Interchange Tutorial	October 1989	Current Issue

[Page intentionally left blank.]

CONTENTS

REFERENCES

[l] Consultative Committee for Space Data Systems, CCSDS 620.1-W-3, Standard Formatted Data Units--Structure and Construction Rules (recommendation), June 1989 or later issue

[2] --, CCSDS 610.1-G-2, Standard Formatted Data Units--Product Aggregation Aspects, June 1989 or later issue

[3] --, CCSDS 640.0-W-3, Transfer Syntax Notation (specification), June 1989 or later issue

[4] Head, T., "A Pilot Data Descriptive Language," paper presented at the Consultative Committee for Space Data Systems Panel 2 International Workshop, Garmisch, Germany, 1987

[5] Liskov, B., et al., "Abstraction Mechanisms in CLU," Communications of the ACM, vol. 20, number 8, August 1977, pp 564-576

[6] Tremblay, J. and Sorenson, P., The Theory and Practice of Compiler Writing, McGraw-Hill, 1985

1. INTRODUCTION

One of the primary goals of the international space community is the interchange of information between space information systems. In an effort to promote and support such interchanges, the Consultative Committee for Space Data Systems (CCSDS) has developed and is developing recommendations for space information interchange. Among these recommendations is the specification of the standard formatted data unit (SFDU) as a methodology for information interchange (Reference 1). Intrinsic to the SFDU specification is the use of a data description record (DDR) to specify the representation of the interchanged information. Use of this specification enables the interchange of information in a form usable by the target system.

1.1 PURPOSE

This document provides background information for users as they choose a language for use in the DDR of an SFDU. The interchange process can be complicated by differences in environments and by priorities between the source and target systems. This document explores what effect these factors have on the evaluation of a language for use in the information interchange process. This document serves as a companion to CCSDS Panel 2 language evaluation working papers, which are produced as appropriate. The SFDU structure standard is discussed in References 1 and 2.

1.2 DOCUMENT ORGANIZATION

This document is divided into four sections and discusses what role languages play in information interchange in the SFDU environment.

Following Section 1, which provides an introduction to language usage in information interchange, Section 2 contains an explanation of the challenges faced in the exchange of space science data. The interchange environment and the role of languages in the interchange process is also discussed.

Section 3 deals with use of language to specify data. It identifies the features within a language that enable the interchange process.

Section 4 shows how the intended usage affects the choice of languages in the interchange process.

2. BACKGROUND

2.1 INTRODUCTION

The collection of space science data is a costly and time-consuming process. It has been known for some time that the science data obtained is of use to a wide community of users, both at the time it is obtained and in the future. The ability to interchange this data in a form that is usable to either primary or secondary users is an important goal. When the processing and storage environments of the source and target systems differ, substantial care and effort must be put forth to obtain this interchange.

CCSDS has recommended the use of the SFDU (References 1 and 2) to provide a methodology for space information interchange. This recommendation provides a labeling mechanism that enables the self-description of the information being interchanged. One type of information supported within the SFDU structure is the data description unit (DDU), which incorporates DDRs to describe data representation. The DDU provides or references the information necessary to process the data.

While it is desirable that the information in the DDU/DDR be machine interpretable, it is not required. The use of English for data description is currently the default; formalized notations and languages addressing the concerns of heterogeneous interchange are under development (References 3 and 4). This document, however, focuses on languages that lend themselves to automated procedures.

2.2 INTERCHANGE ENVIRONMENT

Current information interchange practices require communication between the source and target systems. Such communication is necessary to ensure that data is transported in a representation usable by the target system. This is a trivial problem if the systems are congruent (identical in relevant attributes) but may become more challenging if the systems utilize different machine architectures. The following are among the differences encountered in heterogeneous-interchanges:

- Differences in the representation of character data
- Differences in the number and ordering of octets used in the representation of integers and real numbers

- Differences in numbering of most significant to least significant bits (ascending or descending) in binary data

- Differences in the use of complemented notation of numeric values

At present such discrepancies are accommodated by negotiating the physical representation of the data being presented to the target system. A priori knowledge of the target system by the source system makes such a negotiated interchange possible. When the source and target systems are independent in time and space, such negotiations are not possible.

One approach of the negotiated interchange is to use utilities on the source system to make it acceptable to the target system. The transformation is made on the outgoing system in order to avoid untransformed data, which may be spuriously interpreted by the target system as a control character. Utilities may also be used on the source system to convert incoming data, but caution must be used to avoid misinterpreting data as control characters on the target system. Another approach is the use of a highly structured interchange format, using a custom encoding of the data values. This approach has been used in highly specialized user communities, where the information interchanged conforms to a number of restrictions.

As the number of partner environments grows, the number of negotiated procedures grows geometrically. For example, between two systems there are generally two required procedures: one to take information from site A to site B, and the other from site B to site A. With 4 sites, each requiring different procedures to communicate, the number of required procedures rises to 12: site A to site B, site A to site C, site A to site D, etc.

Even within a relatively constant processing environment, upgrades to systems and media may produce incongruities over time. The transformations necessary for upgrades are usually well supported between sequential releases of a product (i.e., there is generally a suite of utilities or other support to aid the migration between release 1 and release 2). Problems are more likely to arise when the data produced from a much earlier release of software, hardware, media, or operating system needs to be transported to a later system.

The changeover in media from punched card to floppy disk or tape and on to optical disk is an example of advances in media technology. The types of data organization used to optimize storage and access vary widely between media. Advances in hardware and operating systems have similar effects in physical representations of data. The changes in representation that have occurred over the last 20 years suggest that it is prudent to have as complete a specification as possible for data that may be archived for retrieval decades later.

The ability to have knowledge of the partner environment is the key factor in determining what assumptions can be made and what specific differences can be accommodated in a given interchange. If a priori knowledge of the partner environment is not possible or practical, the source environment must take care to provide as complete a specification of the data representation as possible.

Such information may include both formal notations and textual material. It should be noted, however, that if the data is expected to have a long latency period before reuse, a simple pointer to manufacturer and model of hardware and version of software/operating system is insufficient.

Interchanges can be classified into three groups based on the ability to have information of the partner environment and, if such knowledge exists, the degree of congruence between environments. These three classifications are closed interchange, negotiated interchange and, open interchange.

A closed interchange is characterized by full knowledge among congruent environments. This can be viewed as a degenerate case of interchange because there is no need for transformation or for accommodation between such systems.

A negotiated interchange is how most heterogeneous interchanges are currently handled. This type is characterized by the fact that both source and target environments are known to each other. The differences between the systems are accommodated by the use of system utilities or custom software. As one would expect, the ability to communicate is essential in order to negotiate the physical representation of the data to be transported.

An open interchange makes no assumptions about the partner environment in an interchange. To operate in the absence of knowledge of the partner environment, the rigorous and exhaustive specification of the data being transported or

stored is required. Such specification is vital when systems are separated by time. There is no way of knowing what new machine architectures will evolve or if there will be any utilities or documentation available to aid in the transformation of data from current representations to those in use at the time of future need.

Table 2-1 summarizes what types of interchange can be used for a given set of environmental circumstances. In this table, case 1 includes environments that are congruent and known to each other, case 2 includes environments noncongruent and known to each other, and case 3 includes environments unknown to each other (where congruence is not a factor).

As illustrated in Table 2-1, the open interchange model may be used under all conditions, and any model may be used in conditions where the environments are well understood and congruent. While the more restrictive environment of the closed interchange may provide more processing efficiency, it depends on a restricted environment and may require substantial reworking if those restrictions are violated, as they may be as systems evolve over time.

Table 2-1. Conditions for Interchange Model Usage

Conditions	Case 1	Case 2	Case 3
Ability for interchange partners to communicate (or have full knowledge of each other)	Yes	Yes	No
Congruent environments	Yes	No	-
Model available for use:			
1. Closed	Yes	No	No
2. Negotiated	Yes	Yes	No
3. Open	Yes	Yes	Yes

If an interchange is part of an ongoing sequence of interchanges, the cost effectiveness of any given choice will be impacted by the number of interchanges to be performed over a given timeframe, by possible requirements for system independence in the interchange implementation, and by the probability of environmental change in that period. Types of environmental change likely to cause concern are changes in

processing machinery or architecture and changes (including upgrades/new releases) in operating system or in transport media and protocol.

These impacts are most sharply felt in more restrictive environment requirements of the closed interchange and the negotiated open interchange than in the more robust generic open interchange. Those choosing languages for interchange must look at the expected life cycle of the interchange and probability of environmental change/upgrade in that timeframe in reaching a decision.

Space science information may be involved in all three types of interchange within its life cycle. The operational transfer of data is likely to take place between known parties on a regular basis. Generic open interchange methods are currently being developed; as experience is gained, performance and efficiency gains will be made. Until generic open interchange methods mature, the volume of data, frequency of transfer, and longevity of the transfer agreements may make closed or negotiated interchange methods more attractive.

2.3 ROLE OF LANGUAGES

Languages allow the specification of data being transported or stored. This specification may be a physical or logical representation of the information that the data represents. Logical specifications are found in the declarative portion of programming languages. The specification notes the basic data type of the information [i.e., character, Boolean (logical), integer, or real], as well as interrelationships as found in records, vectors, arrays, and sets. Some languages also allow the specification of special-use types, which are relevant to the type of information being described. A language or notation that describes how the information is logically organized is referred to as a data description language (DDL).

Data also has a physical representation that differs between machine architectures, operating systems, and media. This physical representation may be viewed as describing the actual bit patterns encountered in the interchange process. These differences include, but are not restricted to

- Differences in the representation of character data [e.g., Extended Binary Coded Decimal Interchange Code (EBCDIC), 7-bit American Standard Code for Information Interchange (ASCII), 8-bit ASCII]

- Differences in the number of bytes used to represent numeric values (16- versus 32-bit integers)

- Differences in the numbering of bits in an octet (right to left or left to right); a critical difference if a series of bit flags is being used and in the representation of real numbers

- Differences in the complement notation used to represent numerics (i.e., one's or two's complement notations create quite different bit patterns in representing the same value)

A language or notation that depicts the physical representation of the data being interchanged is referred to as a data interchange language (DIL). DILs currently under development contain substantial DDL capabilities. While the presence of such information is necessary for interchange between dissimilar systems, the development of DILs is still at an early stage.

There are two basic approaches that have been taken in the development of DILs: canonical and descriptive. The canonical approach utilizes a predefined, interim, data type representation that is independent of the internal machine representation of either partner. The descriptive approach provides a complete bit-level specification of each data type used in the interchange.

Many file transfer utilities use the canonical approach and convert numerics into character representation for the transfer process over communications lines. This character representation is probably the most general way of transferring numeric data but has large overhead costs in both storage and performance. The choice of a single numeric representation other than character representation or binary encoding (with specified most-significant-byte ordering) is the subject of study by standards committees. Other canonical representation efforts have specified a choice of currently supported machine representations.

Descriptive representation efforts provide flexibility in specification of representation and enable user extensions to representation types. This flexibility may result in some cost efficiency, depending on the implementation.

Most programming languages concern themselves primarily with the logical description of data. This is a proper approach

to ensure that the algorithms used in the data manipulation portion of the language are portable and do not tie themselves to a particular machine implementation.

The use of information provided by DDLs and DILs is essential for the interchange of information where the parties are separated by time or when the processing or storage environments are divergent.

3. LANGUAGE FEATURES AND INTERCHANGE

3.1 INTRODUCTION

The use of language for specification in data interchange makes use of various features that enable the user to describe, in detail, both the logical and physical representations of the data. As discussed in the previous section, a language or notation used for describing the logical representation of data is referred to as a DDL; one used for describing the physical representation is referred to as a DIL. Some languages/notations provide both DDL and DIL capabilities.

The suitability of a language for use in the interchange process involves factors such as portability, commercial availability, support, and the existence of standards. This section discusses various language features and their role in information interchange.

3.2 GENERAL FEATURES

For any language or notation, there are several features regarding expressiveness that are necessary to allow for the identification and access of data values in a generalized fashion.

3.2.1 DESCRIPTION BY REFERENCE

Description by reference is the ability to separate the description of the data from the data itself. By using this capability, it is possible to access portions of the data separately and distinctly by name. Cataloging and retrieval processes access data fields by name. This need for named reference also extends to the aggregated data records, enabling logically related data items to be accessed as a group. Without this capability, data is not easily decoupled from a specific application, and the possibility for its reuse is highly restricted. This feature is required if DDRs are to be written for the interchange process.

3.2.2 SUPPORT OF BASIC DATA TYPES

The "atomic" types of character and numeric real and integer must be supported within the language at a minimum. Additionally, Boolean (logical), bit, and complex types should be able to be accommodated within the framework of the language set.

3.3 DATA TYPE FEATURES

The data type features discussed in this section focus primarily on the logical representation of data and the ability to define user data types.

3.3.1 DATA TYPE DEFINITION CAPABILITIES

Data type definition is the ability of the language to define and name user data types for use within an application, library, or community. An example of this capability would be the definition of time or date as a specific data type to ensure that its representation and usage are consistent. This utilizes the notion of data abstraction, as described in the following quotation from Reference 5:

> A data abstraction is used to introduce a new type of data object that is deemed useful in the domain of the problem being solved. At the level of use, the programmer is concerned with the behavior of these data objects, what kinds of information can be stored in them and obtained from them. The programmer is NOT concerned with how data objects are presented in storage nor the algorithms used to store and access information in them. In fact, a data abstraction is often used to delay such decisions until a later stage in the design.

In addition to data abstraction, data definition capabilities may also include the naming of data types representing composite data units, such as the structures discussed in the next section.

Another example of a composite data type used within space information systems is the image type. Images are composed of a number of individual values but are used as a cohesive unit. The ability to define and access an image as a data unit greatly enhances the usability of a language.

Note that specification of legal operations on a data type is not part of the transfer description, as these operations are dependent on the data manipulation language.

3.3.2 DATA TYPE STRUCTURING CAPABILITIES

Data type structuring is the ability of the language to describe the logical relationship of "atomic" data items. This type of relationship may be conveyed through the use of data structures that support the ability to classify and

aggregate data, which in turn support the processes of generalization and association. As mentioned previously, images are an example of this type of composite data structure.

Classification is the grouping of entities that share characteristics over which uniform conditions hold. A "class" corresponds to an explicit or implicit "type definition." In this respect one could reasonably expect a type such as an image to have a certain set of common characteristics such as parameters to describe the number of rows and pixels in an image.

Aggregation is grouping of entities into a unit. A record can be seen as an aggregate of its component fields. Images are a collection of values that define a unified whole.

Generalization captures the commonalities of one or more given classes, utilizing the uniform condition shared by class members. There are a number of operations that can be performed on images, such as displaying them, which would be inappropriate to perform on other data types. One is able to generalize over that common set of conditions found in images to perform these operations.

Association describes a collection of entities of the same class. Array types and tables are associations of members of a given type. Some images are actually a series of images from different spectra, although each image may be used independently. It is the association of these images that provides the comprehensive unit.

3.4 PHYSICAL REPRESENTATION CAPABILITIES

The ability to describe physical representation is not within the scope of current programming languages. Indeed, in an effort to promote portability, standard programming practices may encourage the "hiding" of actual physical representation and implementation details. This capability is used to gain the benefits of data abstraction, as discussed in Section 3.3.1. This orientation discourages the deliberate use of machine-dependent methods and algorithms, which can shorten the life cycle of applications by making transport or upgrade to new machines or systems unwieldy.

Ironically, the very methods needed to support portable applications lead to difficulty in transporting the data produced or used by those applications. This difficulty

arises because, although the details of representation remain constant within any given environment, these details are hidden to discourage the use of environment-specific (i.e., nonportable) optimizations. However, this is the very information necessary to account for when moving data between systems.

A DIL needs to be able to specify the bit pattern representation of data to be transported. This representation must specify not only basic data types, but also how the implementation producing the information has chosen to represent these types. Among the representation issues are the encoding rules used for representing character data and the varied representations of numerical data.

Current programming systems do not address these issues. There are two efforts under way by CCSDS to provide this level of specification capability. They are the transfer syntax data notation (TSDN), discussed in Reference 3, and the pilot data descriptive language (PDDL), discussed in Reference 4. Both efforts provide a method for specifying the physical representation, as well as adding some of the logical representation capabilities discussed previously.

3.5 LANGUAGE IMPLEMENTATION CONSIDERATIONS

Among the considerations for choosing a DDL and/or DIL is the availability and support of a given language. For a language to be useful in the interchange process, the capability to parse the language or notation must be available and consistent in each environment. The availability of national or preferably international standards for a language is a definite advantage in the viability of its use in the interchange process.

Beyond the availability of standards, the commercial availability and support of a language adds to the expectation of continued availability and the maintenance of the language and provides recourse in the event of unanticipated results. The availability of software tools such as compilers, interpreters, and service libraries, as well as interfaces to data base management packages, may make a language attractive for use in a particular environment.

The ease of use of a language is also a consideration. A practical explanation of ease of use is provided in Reference 6:

It is desirable that the constructs of a language be easy to learn and remember. Once programmers are familiar with the language, it should not be necessary for them to consult manuals constantly. The language should be so simple and straightforward that the proper way to do something is reasonably apparent. On the other hand, there should not be many ways to do the same thing, as the programmer will then flounder helplessly trying to decide which is best.

Ease of use is degraded if needed data constructs cannot be described without resorting to creative use of a language's declarative capabilities. The need for procedural methods or workarounds to compensate for lack of descriptive robustness in a language may offset the advantages of familiarity by increasing both processing and maintenance overhead.

4. INTERCHANGE LIFE-CYCLE CONSIDERATIONS

4.1 INTRODUCTION

The circumstances, requirements, and constraints surrounding any information interchange will have an influence on the choice of language used in the interchange. While an interchange may be viewed as a single transaction, it is likely that it is one of a series of interchanges between various sites. The partner environment and/or content type of the interchange may vary in this series. The expected number of cycles of interchange, the life-cycle calendar timeframe, and the number and diversity of the recipients are all factors that have practical implications.

The life cycle of the interchange being addressed will impact the relative priority of performance and flexibility. These priorities will influence language choice. To explore these differences, interchanges can be divided into three categories: production, ad hoc, and archival. These categories focus on the predictability of repeated or multiple requests and the likelihood of a priori communication or agreements. A scenario is provided in the following subsections for each category, using one of the interchange models described in Section 2.

4.2 PRODUCTION INTERCHANGE DESCRIPTION AND SCENARIO

This category of interchange is characterized by the fact that the same type of information is interchanged repeatedly between known systems or from node to node within a system. In some sense, this may be viewed almost as a subscription service. Because of its predictability and the amount of prior communication required, this type of interchange is most likely to be a closed interchange or negotiated interchange. Until a highly efficient set of SFDU services is available for a recommended DIL, these interchanges are likely to use implied DILs, such as system utilities and custom applications to perform transformations.

This category of interchange may well have a known set of recipients and a relatively constant DDU. This is the category-of interchange in which standard information products are distributed on a regular basis. The recipients are likely to be prime investigators, research organizations, other agencies, distribution points, and archive centers.

While this approach gives performance gains, it assumes that the environment will remain unchanged at both ends. A data

base management system (DBMS) upgrade at the source system may produce files incompatible with the target system. Upgrades to machine or operating systems on either end of the interchange could well entail substantial refitting efforts to accommodate change. The cost and efficiency tradeoffs between a closed system and a more flexible, open system must be viewed against the expected life cycles of the mission and the partner processing environments.

The volume and predictability of the data and its distribution may make the use of a closed model appear cost effective. It assumes that the processing environments are unlikely to change substantially over the life cycle of the interchange agreement. However, the probability of upgrade and processing environment evolution over years may make an open system approach more cost effective and efficient in the long term, especially from a maintenance standpoint.

This scenario is one in which an investigator wishes to receive a monthly extract of data from an ongoing 2-year mission. The data is distributed from a central source that utilizes a proprietary DBMS. The investigator is using the same DBMS under the same hardware and operating system configuration as the source.

The investigator decides to capitalize on performance advantages available under severe restrictions. Under this closed interchange, it is possible to use the proprietary format to transport data between the two systems. Depending on the DBMS, the DDR may consist of schema or may only be a reference to the DBMS software that can access the proprietary format, which includes data description.

The source site upgrades to a new version of the DBMS, which produces files incompatible with the previous version. The new version can read and convert data from the previous version, but backward conversion is not possible. The investigator must find alternative methods to obtain his or her data until the DBMS at his or her site is upgraded to the new version.

4.3 AD HOC INTERCHANGE DESCRIPTION AND SCENARIO

This interchange category is characterized by the fact that the actual description of the data to be interchanged may not be known a priori, and there is no reason to believe that the particular request will be repeated, although the same recipient is likely to make future requests.

The recipients are likely to be users who are performing additional analysis and requesting information that fits a certain set of criteria. This interchange category is likely to generate nonreusable DDUs. This need for the generation of custom DDUs for each interchange would indicate the need for a robust DDL to describe the logical representation of the data.

If this request is between well-acquainted systems, it may be possible to use the implicit DIL of a negotiated transfer. Note that the use of a negotiated interchange is sensitive to changes in the partner environments. Continued communication is required to allow the coordination of necessary adjustments to accommodate changes in either environment.

This type of interchange is certainly a candidate for use of a formal DIL. The use of open interchange is particularly attractive in a situation where new machine architectures or transport media are evolving and where multiple environments are involved. The fact that any particular data request representation is not necessarily to be reused and the fact that requests may come from a variety of systems may make the use of a generic open interchange model attractive from a maintenance and administrative viewpoint.

This scenario is one in which another investigator wishes to receive sets of data from the same ongoing 2-year mission, based on the values in the data. This investigator is at a site that uses different hardware and software than the source site. Communication between the systems/operations staff at the two sites reveals that standard utilities exist to transform the data into a form usable at the target site.

A DDU is generated with a DDR written in Ada to describe the logical representation of the data. While no DDR is written to describe the physical representation, an implicit DIL exists in the usage of the standard utilities, which transform the physical representation.

An upgrade occurs in the operating system at the receiving site, which impacts the internal representation of data. The receiving site needs to use a second set of utilities to transform the data to its current status, as the sending site utilities have not been upgraded to reflect the new format.

4.4 ARCHIVAL INTERCHANGE DESCRIPTION AND SCENARIO

This type of interchange often involves separation of source and target environments by time, which may preclude communication between the source and target systems. This inability to depend on communication makes use of an explicit DIL essential in the specification of the archival representation of information. It is conceivable that archived data will need to be accessed decades after its representation has been specified. It is likely that the evolution of processing and storage technology will make use of implicit DIL unfeasible. By necessity, these will be open interchanges. The explicit DIL may be textual until an automatable notation/language is available.

This scenario is 25 years after the end of the 2-year mission discussed in the previous scenarios; an investigator requests a copy of the archived data. This investigator wishes to compare recently gathered data with the historic readings of the earlier mission. Luckily, the data was stored in SFDU format with a DDU containing both DDL and DIL descriptions.

Although the environment in which the data was generated and stored is no longer supported, automated SFDU services are used to put the information into a form usable by the requesting investigator's system. The information contained within the SFDU also contains other information, such as a data dictionary and supplemental information, which enables a better understanding of the archived data.

Another set of historic data that this investigator requests was also archived in SFDU format but used a textual DIL. The information is still retrievable but requires substantial manual intervention. One of the principal investigator's (PI's) graduate assistants matches the description with a binary dump of the data and then writes custom software to recover the required information.

ANNEX A

GLOSSARY OF TERMINOLOGY

ANNEX A - GLOSSARY OF TERMINOLOGY

ASCII	American Standard Code for Information Interchange: a 7-bit code also known as USA Standard Code for Information Interchange (USASCII)
Closed System Interchange	An information interchange between systems using private agreements on data representation; includes reliance on common environments
Congruent Environments	Processing environments that are consistent with each other in the processing and representation of data
Data	Representation forms of information dealt with by information systems and their users
DDL	data description language: used for specifying the logical representation of data
DIL	data interchange language: used for specifying the physical (bit pattern) representation of data
Implicit DIL	Use of system utilities or custom software in place of an explicit DIL in the interchange data between heterogeneous environments
Information	Any kind of knowledge that is exchangeable between users
Language	A definition comprised of a grammar and associated semantics
Negotiated System Interchange	Interchange between systems using different physical representations, where transformation utilities or custom software is used in place of an explicit DIL
Open System Interchange	Interchange between systems where communication and/or congruence cannot be assumed
SFDU	standard formatted data unit: data units that conform to CCSDS recommendations for structure, construction rules, and field specification definition

ANNEX B

LIST OF ACRONYMS

ANNEX B - LIST OF ACRONYMS

ASCII	American Standard Code for Information Interchange
CCSDS	Consultative Committee for Space Data Systems
DBMS	data base management system
DDL	data description language
DDR	data description record
DDU	data description unit
DIL	data interchange language
EBCDIC	Extended Binary Coded Decimal Interchange Code
PDDL	pilot data description language
SFDU	standard formatted data unit
TSDN	transfer syntax data notation