NBSIR 88-3700
(Supersedes NBSIR 85-3164)

A Technical Overview of the Information Resource Dictionary System (Second Edition)

Alan Goldfine
Patricia Konig

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
Institute for Computer Sciences and Technology
Gaithersburg, MD 20899

January 1988

U.S. DEPARTMENT OF COMMERCE
NATIONAL BUREAU OF STANDARDS

NBSIR 88-3700
(Supersedes NBSIR 85-3164)

A TECHNICAL OVERVIEW OF THE INFORMATION RESOURCE DICTIONARY SYSTEM (Second Edition)

Alan Goldfine
Patricia Konig

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
Institute for Computer Sciences and Technology
Gaithersburg, MD 20899

January 1988

U.S. DEPARTMENT OF COMMERCE, C. William Verity, *Secretary*
NATIONAL BUREAU OF STANDARDS, Ernest Ambler, *Director*

A TECHNICAL OVERVIEW OF THE INFORMATION RESOURCE DICTIONARY SYSTEM (Second Edition)

Alan Goldfine
Patricia Konig

This publication provides a technical overview of the computer software specifications for an Information Resource Dictionary System (IRDS). It summarizes the data architecture and the software functions and processes of the IRDS. The IRDS Specifications are the foundation for an American National Standard, a U.S. Federal Information Processing Standard (expected to be issued in 1988), and a Draft Proposal within the International Organization for Standardization (ISO). This Overview also provides background information on the development of the IRDS software specifications.

Key words: American National Standard; computer software; data dictionary; data dictionary system; data management; Federal Information Processing Standard; FIPS; Information Resource Dictionary System; IRDS; information resource management; IRM; International Standard.

ACKNOWLEDGMENTS

We gratefully acknowledge the technical contributions and thorough review of this publication by members of Technical Committee H4 of Accredited Standards Committee X3. We also appreciate the technical contributions of Dr. Henry C. Lefkovits, President of AOG Systems Corporation, and his staff. Dr. Margaret Henderson Law's thorough review and recommendations on an earlier version of this publication greatly enhanced the clarity of presentation.

TABLE OF CONTENTS

Page

1. INTRODUCTION

1.1 BACKGROUND

Significant changes have occurred in the evolution of computer and information processing technology in the past decade. Technology advances have reduced the costs of computing and sparked an enormous growth in the use of computers. For many organizations, the proliferation of computing capabilities has resulted in a corresponding proliferation of redundant and inconsistent data. Increasingly, organizations are now viewing data and information as resources that must be managed.

The data dictionary system is a key computer software tool for the management of data and information resources. It provides facilities for recording, storing and processing descriptions of an organization's significant data and data processing resources. In 1980, both the American National Standards Institute (ANSI) and the National Bureau of Standards of the United States Department of Commerce initiated efforts to develop standards for dictionary software. The ANSI effort began with the approval by the Accredited Standards Committee for Information Processing Systems (X3) of a project to develop a standard for an "Information Resource Dictionary System" (IRDS). This resulted in the July 1980 convening of what is now Technical Committee H4 of Accredited Standards Committee X3 (X3H4), the group responsible for developing the Standard for an IRDS.

As the world's largest user of information processing technology, the U.S. Federal Government depends on this technology to carry out Government-wide programs and deliver essential public services. The National Bureau of Standards' effort focused on the development of a Federal Information Processing Standard (FIPS) for Data Dictionary Systems.

Although X3H4 and the National Bureau of Standards used different titles (i.e., "Information Resource Dictionary System" and "Federal Information Processing Standard for Data Dictionary Systems"), the two groups had identical goals and a similar development approach.

The two efforts came together in September 1983 when X3H4 voted to adopt the August 1983 version of the draft Federal Information Processing Standard for Data Dictionary Systems as its Base Document. The Base Document then evolved into the current specification for the American National Standard IRDS [1]. The draft proposed American National Standard was twice circulated for public review and comment, in 1985 and, as revised, in 1986. Comments on the same draft document, considered as a potential FIPS, were solicited from the U. S. Federal community in 1985.

In January 1984, the International Organization for Standardization (ISO) Technical Committee 97 assigned the task of reviewing and commenting upon the IRDS Specifications as a potential International Standard to Subcommittee 21, Working Group 3 (ISO TC97/SC21/WG3). In 1986, after extensive study, Subcommittee 21 voted to register the IRDS as a Draft Proposal, the first official step in the IRDS becoming an International Standard.

1.2 DEVELOPMENT APPROACH

Since the Institute for Computer Sciences and Technology (ICST) at the National Bureau of Standards initially developed much of the IRDS Specifications, it is important to review the methods that ICST used. ICST interacted closely with U. S. Federal Government users to develop software specifications that would support U. S. Federal Government requirements and that would be implemented by a wide spectrum of software suppliers and thus be available "off the shelf." ICST pursued this goal by:

- Preparing and disseminating the Prospectus for Data Dictionary System Standard [2] in 1980. The Prospectus discusses the use of data dictionary systems and describes the plan to develop a Federal Information Processing Standard.

- Conducting two Data Base Directions workshops in October, 1980 and October, 1985. The first workshop investigated how managers can evaluate, select, and effectively use information resource management tools, especially data dictionary systems. The second workshop assessed the nature of current information resource management practice and problems, and reported on solutions that have proven workable. Again, a considerable emphasis was on data dictionary systems.

The two workshop proceedings were published in 1982 [3] and 1986 [4], respectively.

- Conducting interviews with U.S. Federal agency personnel who were knowledgeable about dictionary systems to identify current and projected requirements for a Standard. Based on these interviews and comments on the Prospectus, the Federal Requirements for a Federal Information Processing Standard Data Dictionary System [5] was published and disseminated in the Fall of 1981.

- Conducting a series of seven workshops in 1982-86 for representatives of more than fifty U.S. Federal agencies. The purpose of the workshops was to obtain feedback from the representatives on the evolving Specifications, and to then incorporate the feedback into the next revision of the Specifications.

- Preparing and disseminating an interim publication, Functional Specifications for a Federal Information Processing Standard Data Dictionary System [6], for review and comment early in 1983. ICST received and analyzed comments on this publication from more than 100 U. S. Federal Government agencies.

- Preparing and disseminating, in August 1983, the draft Specifications for the Federal Information Processing Standard for Data Dictionary Systems, the document that became the X3H4 Base Document.

ICST prepared the Functional Specifications and the August 1983 draft Federal Information Processing Standard Specifications with extensive contract assistance from AOG Systems Corporation.

To facilitate industry evaluation and acceptance of the Standard, ICST personnel held discussions with current and potential vendors of dictionary systems. ICST also sponsored three workshops for vendors to review the Specifications as they evolved. Vendors participating in the discussions and/or workshops included:

Advanced Systems Technology	Applied Data Research
Burroughs Corp.	CINCOM Systems
Computer Corp. of America	Cullinet Software
D&B Computing Services	Digital Equipment Corp.
Frontier Solutions	GEAC Canada
General Electric Corp.	Hewlett-Packard Corp.

Honeywell Information Systems	IBM Corp.
Infodata Systems	INTECH Corp.
Intel Corp.	International Computers, Ltd.
Manager Software Products	NCR Corp.
Pansophic Systems	SAS Institute
Software AG of North America	Sperry Corp.
TSI International	UCCEL Corp.
Wang Laboratories	

ICST also disseminated the interim publications and draft Specifications to more than 200 private industry organizations, universities, and state and local governments in the United States and organizations in Australia, Austria, Brazil, Canada, England, Federal Republic of Germany, France, Israel, Japan, Luxembourg, Mexico, the Netherlands, Saudi Arabia, Scotland, and Sweden. This was in addition to the distribution of the documents to U. S. Federal Government agencies, software suppliers, and standards committees.

ICST personnel have been active in X3H4 since its inception, and have provided X3H4 members with copies of all the documents discussed above and reviewed with them the results of the workshops. Many of the X3H4 members also attended the three workshops that ICST conducted for vendors of dictionary software.

1.3 DEVELOPMENT OF AN IRDS PROTOTYPE

ICST has been developing a prototype IRDS that will eventually implement the entire IRDS Command Language. The prototype uses SQL calls to a relational DBMS to model the IRDS data structures and to provide the underlying data management. A set of C language programs interpret the commands and interface with the DBMS. The current source code for the prototype is available from ICST.

1.4 BENEFITS OF AN IRDS

A preliminary cost-benefit overview prepared for ICST [7] estimates that the U. S. Federal Government could realize over $120 million (in constant 1983 dollars) in benefits by the early 1990s from use of a standard IRDS. Opportunities identified for cost reduction and avoidance included the following:

- Improved identification of existing, valuable information resources that can be used by others in the same organization or shared with other organizations.

- Reductions of unnecessary development of computer programs when suitable programs already exist.

- Simplified software and data conversion through the provision of consistent documentation.

- Increased portability of acquired skills resulting in reduced personnel training costs.

Similar savings can be expected in non U. S. Government organizations.

Although the Standard does not require an organization to use a dictionary system or use one in a prescribed manner, the IRDS can be used to:

- Aid development, modification, and maintenance of manual and automated systems throughout their lifecycle.

- Support an organization-defined data element standardization program.

- Support records, reports and forms management, spanning the range from non-automated to fully automated environments.

Even before the availability of systems that conformed to the IRDS Standard, the Specifications helped users become more informed consumers by providing a common framework and terminology to specify required dictionary system capabilities and to help evaluate vendor offerings.

1.5 IRDS DESIGN OBJECTIVES

In developing the Specifications for a standard IRDS, X3H4 and ICST recognized that dictionary system technology is evolving and that the use of dictionary systems is expanding. In view of this, X3H4 and ICST identified the following three major objectives:

- The IRDS should contain the major features and capa-

bilities that exist in currently available dictionary systems.

- The IRDS should be modularized to support a wide range of user environments and to support cost-effective procurement.

- The IRDS should support portability of skills and data.

1.5.1 The IRDS: An Outgrowth of Existing Systems

During the initial phase of development, both X3H4 and ICST analyzed relevant literature and existing commercial and Federally-developed dictionary systems. Features and capabilities in the current generation of dictionary systems and projected technology trends were identified. Major U. S. Federal Government dictionary system users reviewed and rated 96 features of existing systems. The rating results and conclusions appear in Federal Requirements for a Federal Information Processing Standard Data Dictionary System [5].

As discussed in Section 1.2, U. S. Federal Government representatives and dictionary software vendors reviewed draft versions of the Specifications. These reviews focused on: (1) the functions required or desired by users of their dictionary systems; and (2) the technical and economic feasibility of implementing the specified IRDS functions. As a result of these analyses and reviews, the IRDS Specifications contain the most commonly used facilities of existing systems, and thus represent a "state-of-practice" level of technology in dictionary systems.

1.5.2 Flexibility of Use and Procurement Cost-Benefits

To provide IRDS flexibility and procurement cost-effectiveness, X3H4 and ICST adopted a modularized approach. The IRDS Standard includes specifications for a "Core" dictionary system Module plus specifications for five additional Modules. The six Modules constitute the "base level" standard.

Although the five additional IRDS Modules interface with the Core Module, they are independent of one another. Organizations, therefore, can acquire one or more of the five additional Modules if it supports their requirements.

They would not have to procure Modules that they do not need.

The Core IRDS Module contains the basic capabilities that organizations generally need. These Core Specifications are intended for implementation on large microprocessors and small minicomputers as well as large computers. The five additional Modules in the IRDS contain specifications for: (1) a "starter set" of dictionary data structures; (2) an IRDS security facility; (3) a facility supporting an organization's life-cycle management methodology; (4) a facility for defining and executing procedures of IRDS Commands; and (5) an application program interface.

ICST, X3H4 and ISO TC97/SC21/WG3 are actively developing additional Modules. Chapter 11 discusses these Modules and others that are under consideration.

To provide additional flexibility, capabilities are specified in the Core IRDS that enable organizations to customize or extend the type of data that can be stored in the IRD. These capabilities will provide the ability to describe unique resources and define organization specific system development methodologies.

1.5.3 Portability of Skills and Data

The Core IRDS Module contains two user interfaces: a menu-driven "Panel" Interface and a Command Language Interface. The Panel Interface is designed to support interactive processing, especially by inexperienced users. This Interface leads users down a structured path of screens (i.e., panels) that result in the execution of IRDS functions. Thus, non-technical staff as well as technical staff will be able to execute IRDS functions without having to understand or use the more complex syntax of the Command Language Interface. The Command Language Interface may be used in either a batch or interactive mode.

The IRD-IRD interface facility, discussed in Chapter 7, provides a controlled method of moving data from one standard Information Resource Dictionary to another. Organizations using a standard IRDS could, for example, extract data from decentralized IRDs and add it to a central IRD that focused on organization-wide data management. The specified IRD-IRD interface supports this transportability of data even in the case where the standard IRD systems are devel-

oped by different vendors and are resident on different hardware systems at different locations.

1.6 CONFORMANCE TO THE STANDARD

1.6.1 User Interfaces

An implementation of the IRDS conforms to the Standard if it has either one or both of the user interfaces. The Command Language is the same, except for some implementor options, in all IRDSs that have this user interface. Likewise, the Panel Interface is similar. Thus, individuals will, without significant retraining, be able to use different IRDSs that have the same user interface.

1.6.2 Standard/Extended Mode

There are two modes of operation for an IRDS Module:

- Standard Mode: The IRDS shall perform as specified by the Standard. No extensions or modifications to IRDS functionality are allowed.

- Extended Mode: The IRDS shall, as a minimum, perform all functionality defined by correct syntax and semantics, as specified by the Standard. It may also provide extensions to the Standard.

An implementation of the IRDS is then conformant if either:

- It operates in Standard Mode, or

- It operates in Extended Mode, but can, at the option of the user organization, be restricted to operate in Standard Mode.

Thus, the IRDS implementor may provide extensions to the Standard, so that functions not allowed in Standard Mode are allowed in Extended Mode. However, use of all functionality that would have been correct in Standard Mode must still produce the same results in Extended Mode.

1.7 SCOPE OF REPORT

The remainder of this report contains a summary of the features of the IRDS. General topics, such as the effective use of a dictionary system, are not addressed. Therefore, readers of the subsequent chapters are presumed to be familiar with general data processing concepts and the purposes of a data dictionary system.

Chapters 2 and 3 provide a technical summary of the IRDS structure and processes. The remaining chapters, providing more technical detail, are designed for those individuals reviewing the actual Specifications. It is recommended that reviewers of the IRDS Specifications also read Using the Information Resource Dictionary System Command Language (Second Edition) [8], and those interested in applications of the IRDS read Guide to Information Resource Dictionary System Applications: General Concepts and Strategic Systems Planning [9].

2. OVERVIEW OF THE IRDS DATA ARCHITECTURE

This chapter presents an overview of the IRDS data architecture--the framework in which Information Resource Dictionary (IRD) data is organized and presented to the user. Also discussed here are the properties of the various names that can be used to refer to data stored in the IRD.

2.1 AN IRDS USER'S VIEW OF DATA

The IRDS Standard, including the Command Language and Panel Interfaces, is specified in terms of entities, relationships, and attributes. An IRDS entity represents or describes a "real world" concept, person, event, or quantity, but it is not the actual data that exists in an application file or database. Thus, an IRDS entity might be Social-Security-Number or Payroll-Record. It would not be the actual social security number "123-45-6789" or the actual contents of a payroll record. A relationship is an association between two IRD entities (e.g., the Payroll-Record "CONTAINS" Social-Security-Number). Attributes represent properties of an entity or relationship. For example, one attribute of the entity Social-Security-Number is its LENGTH. In this example, the value of LENGTH is 9 characters.

The reason for specifying the IRDS through the use of entities, relationships, and attributes is that the majority of current dictionary system implementations either use this approach or can be easily modeled with it. Nevertheless, the Standard <u>does not dictate an implementation approach</u>. Although users of the IRDS "see" entities, relationships, and attributes, the software system implementing the Standard can be designed, for example, as a relational, network, or other database management system application. (See section 1.3 and also [10].)

Relationships in the IRDS are binary, denoting that an association exists between two entities in the IRDS. The reasons for choosing the binary relationship approach, rather than a 3-part or more relationship approach, are: (1) the vast majority of current implementations use binary relationships; and (2) the IRDS should be "simple" enough

not to preclude implementation on large microprocessors or small minicomputers.

A small subset of an Information Resource Dictionary might conceptually have the form presented in Figure 1. In this example, Finance-Department, Payroll-System, Personnel-Department, Personnel-System, etc., represent "entities." As depicted, the Finance-Department is responsible for the Payroll-System and the Personnel-Department is responsible for the Personnel-System. The "relationships" between these entities reflect these responsibilities.

Both the Payroll-Record and the Personnel-Record entities contain the lower-level entities Social-Security-Number and Employee-ID. The LENGTH of the Social-Security-Number is 9 characters and that of the Employee-ID is 7 characters. This information is conveyed as "attributes" of the entities Social-Security-Number and Employee-ID, respectively. Although they are not depicted in Figure 1, other attributes might describe the average number of Payroll-Records in the Payroll-File and the average number of Personnel-Records in the Personnel-File.

An important aspect of the IRDS is the concept of type. Different attributes will in general have different meanings. For example, the length of Social-Security-Number and the number of RECORDs in a FILE are different. This situation is represented in the IRDS by declaring that each attribute has a "type" called an "attribute-type." Thus, there would be attribute-types called LENGTH and NUMBER-OF-RECORDS.

Attributes of a specific type will often apply to only some of the entities. In this example, LENGTH is only meaningful to Social-Security-Number and Employee-ID. NUMBER-OF-RECORDS only has meaning for Payroll-File and Personnel-File.

In a similar manner, Social-Security-Number and Finance-Department are instances of different types of entities. As depicted in Figure 1, Social-Security-Number is defined in the IRDS as an ELEMENT entity, Finance-Department is defined as a USER entity, Payroll-System is defined as a SYSTEM entity, etc.

Subset of an Information Resource Dictionary

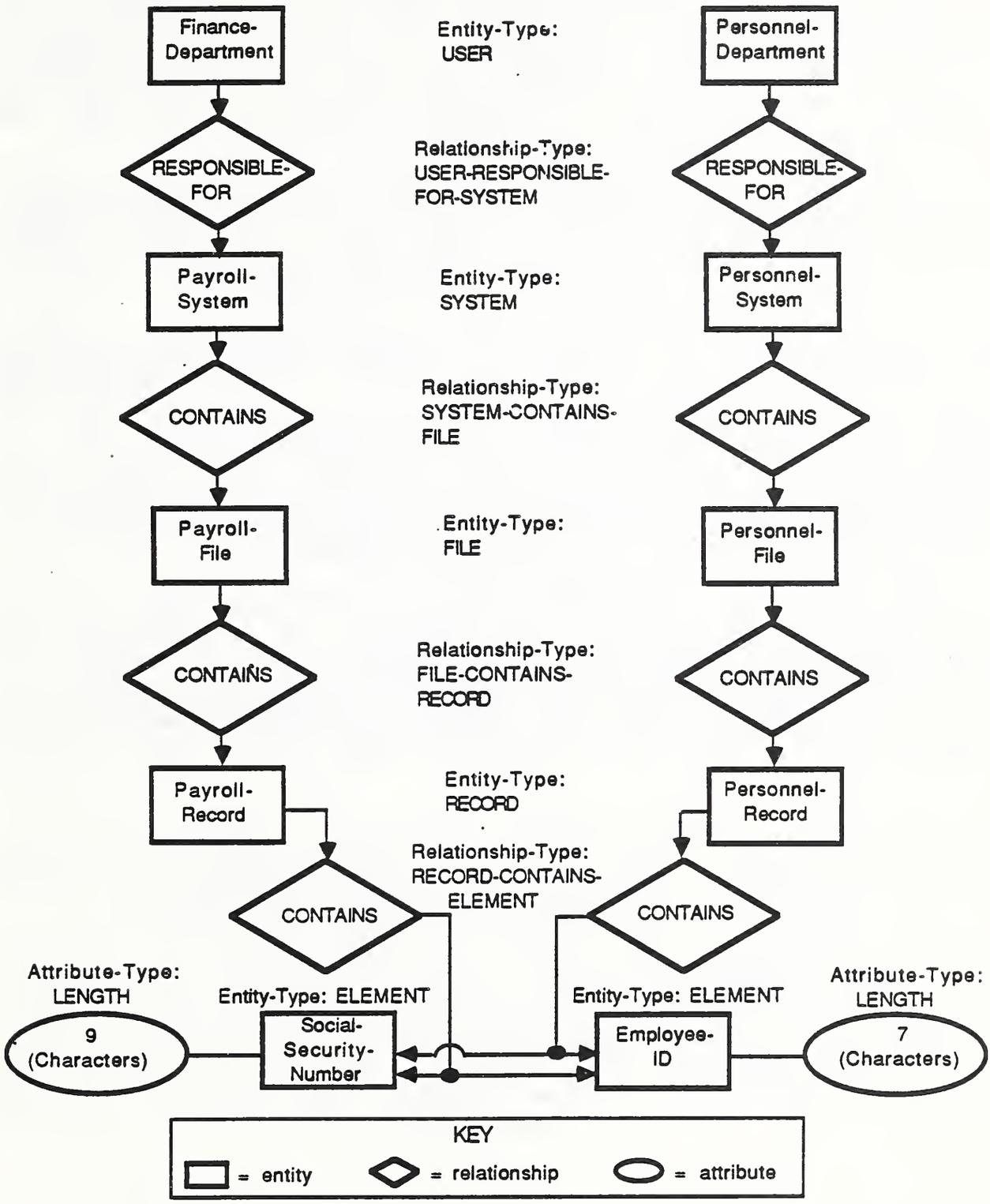


Figure 1

Chapter 2 - OVERVIEW OF THE IRDS DATA ARCHITECTURE

The concept of type also applies to the relationships shown in Figure 1. "CONTAINS" has the same general meaning in Payroll-File CONTAINS Payroll-Record and Personnel-File CONTAINS Personnel-Record. The relationship-type of both is FILE-CONTAINS-RECORD. Although it is not shown in Figure 1, Personnel-File could CONTAIN multiple record descriptions (e.g., Personnel-File CONTAINS Consulting-Record and Personnel-File CONTAINS Temporary-Personnel).

Payroll-Record CONTAINS Social-Security-Number, however, has a different meaning, because it is a RECORD-CONTAINS-ELEMENT relationship. Thus, relationships between entities of different types always have different meanings.

Relationships also can have attributes. For example, the relationship in Figure 1 between Payroll-Record and Social-Security-Number could have a RELATIVE-POSITION attribute-type with an attribute (a value) of 2 to document that Social-Security-Number is the second ELEMENT in the Payroll-Record. The value of the same attribute-type on the relationship between Personnel-Record and Social-Security-Number might be 4 to indicate that Social-Security-Number is the fourth ELEMENT in the Personnel-Record.

In addition, ordered sets of attributes called "attribute-groups" exist. For example, an ALLOWABLE-RANGE "attribute-group-type" might consist of the pair of attribute-types LOW-OF-RANGE and HIGH-OF-RANGE. The value for LOW-OF-RANGE does not convey sufficient meaning by itself. Therefore, it must be "grouped" with the HIGH-OF-RANGE value.

Entities, relationships, attributes, and attribute-groups will sometimes be referred to as "instances" of their respective types. Thus, Finance-Department is an instance of USER, and 5600 is an instance of NUMBER-OF-RECORDS.

2.2 THE IRD SCHEMA

Section 2.1 addressed the organization of data in the Information Resource Dictionary. This section and the following one focus on the purpose and contents of the Information Resource Dictionary Schema.

Figure 2 shows the relationship between IRD processes and IRD data. This view of the IRDS also illustrates both the self-describing nature of the IRDS, and the utility of using the same descriptive technique for both the IRD and its Schema. Figure 3 gives examples of typical contents at the four IRDS data levels.

The IRD Schema describes the structure of the IRD. Thus, for every entity, relationship, attribute, and attribute-group that can exist in the IRD, the IRD schema will contain the corresponding entity-type, relationship-type, attribute-type, and attribute-group-type. The Standard specifies specific entity-types, relationship-types, attribute-types, and attribute-group-types. These types, organized into the Minimal Schema of Module 1 and the Basic Functional Schema of Module 2, are discussed in sections 2.3 and 2.4.

The concept of the IRD Schema is important for two reasons. First, the IRDS Specifications include facilities that enable an organization to "extend" or "customize" the Minimal Schema and the Basic Functional Schema. This means that an organization can add additional entity-types, relationship-types, attribute-types, and attribute-group-types to satisfy its unique requirements.

Second, the IRD Schema supports the Core plus Module approach described in Chapter 1. The IRD Schema provides a mechanism not only to extend Schema data but also to define and develop additional IRDS functions and control facilities. This is similar to adding a new application into a database environment.

2.3 THE MINIMAL SCHEMA

Every software package conforming to the IRDS Specifications includes the Minimal Schema, a collection of entity-types, relationship-types, attribute-types, and other descriptors necessary to establish controls over, and ensure the integrity of, the IRD Schema and the IRD.

IRD PROCESSES

IRD "DATA" LAYERS

Implementor
Unique

IRD Schema Maintenance,
Selection, Reporting
IRD Schema Control Facilities
(Life Cycle)

IRD Maintenance, Selection,
Reporting
IRD Control Facilities
(Views)

No IRDS Functionality

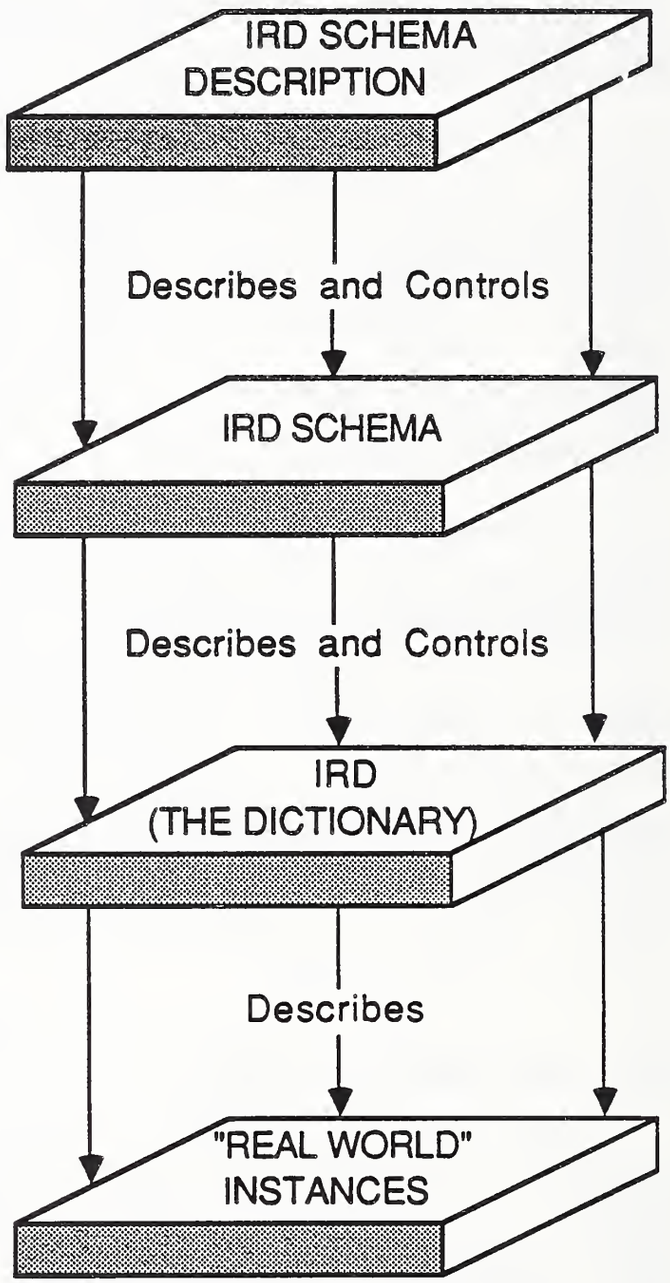


Figure 2

Information Resource Dictionary System (IRDS) Contents

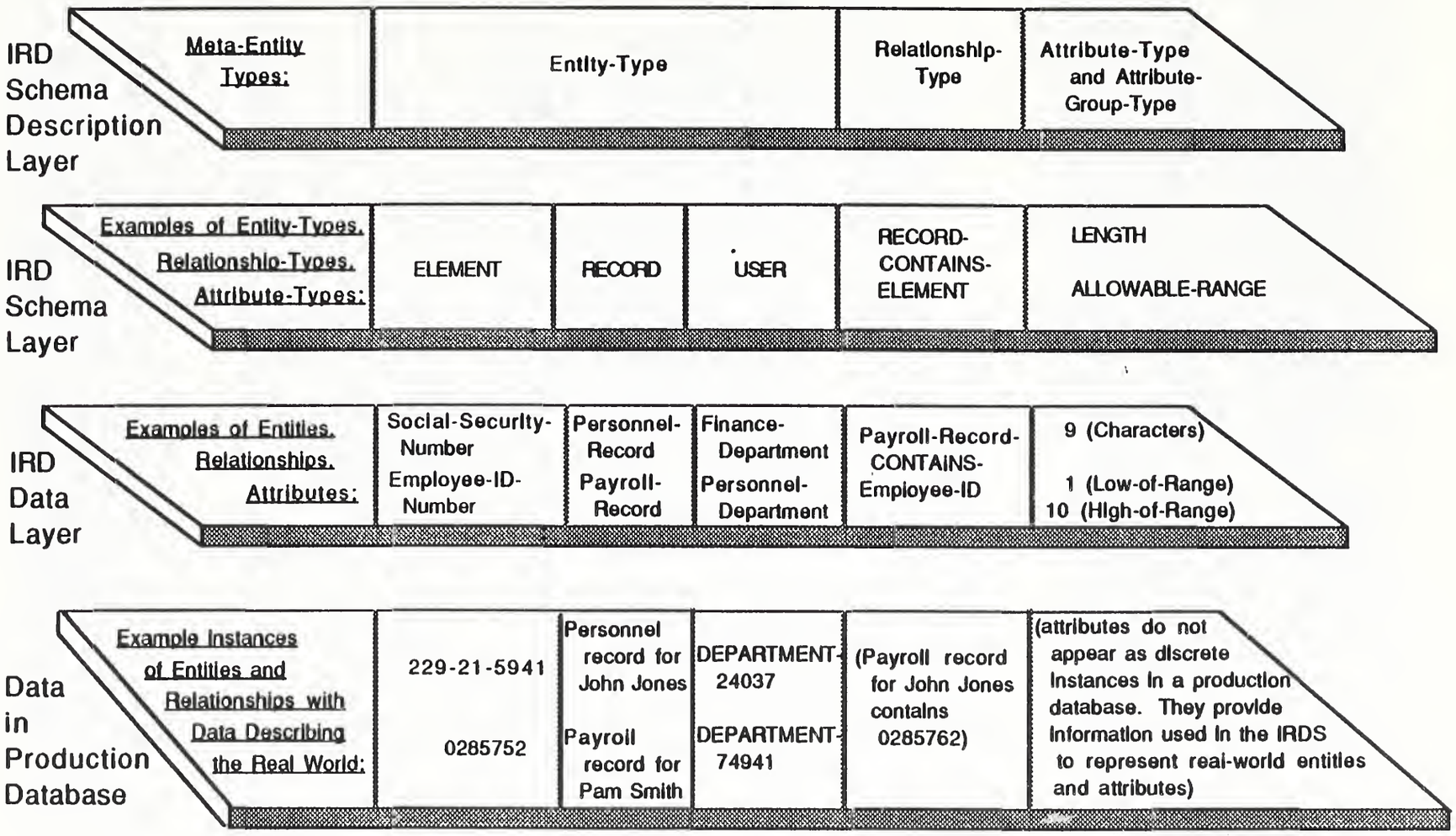


Figure 3

For example, the Minimal Schema includes the entity-types IRDS-USER, IRD-VIEW, and IRD-SCHEMA-VIEW, the relationship-types IRDS-USER-HAS-IRD-VIEW and IRDS-USER-HAS-IRD-SCHEMA-VIEW, and the attribute-type DEFAULT-VIEW, which are used by the organization to control and regulate access to the contents of the IRD and the IRD Schema. The attribute-types ADDED-BY, LAST-MODIFIED-BY, and NUMBER-OF-TIMES-MODIFIED, and the attribute-group-types DATE-TIME-ADDED and DATE-TIME-LAST-MODIFIED automatically document "audit" information concerning changes to the IRD and its schema. The complete Minimal Schema is listed in Appendix A.

2.4 THE BASIC FUNCTIONAL SCHEMA

To support intra- and inter-organization communication about information resources, the IRDS includes a Basic Functional Schema Module as Module 2 of the Specifications.

The Basic Functional Schema defines a "starter set" of IRDS entity-types, relationship-types, and attribute-types. This starter set reflects agreements reached by members of X3H4 and attendees at user workshops sponsored by the National Bureau of Standards. These groups believed that this Schema can be used by organizations to describe most existing and planned manual and automated systems.

Since it is not feasible to identify all entity-types, relationship-types, and attribute-types that might be useful, an organization can augment the Basic Functional Schema using IRDS extensibility. Organizations, for example, that have large amounts of scientific data or who have distributed processing applications may want to add additional entity-types, relationship-types, and attribute-types to the Basic Functional Schema.

Section 2.4.1 provides an overview of the Basic Functional Schema. The complete Basic Functional Schema is listed in Appendix B.

In the remainder of this document, the names of Minimal Schema and Basic Functional Schema entity-types, relationship-types, relationship-class-types, attribute-types, and attribute-group-types will be represented as they appear in the IRDS Specifications, using all upper-case letters (e.g., ELEMENT, FILE-CONTAINS-RECORD). Examples of

entities, relationships, and attributes will be represented using lower-case letters, with the initial letter, all letters following hyphens, and embedded relationship-class-type names capitalized (e.g., Payroll-File, Budget-System PROCESSES Cost-Center-File).

2.4.1 Basic Functional Entity Types

The Basic Functional Schema contains eight entity-types that conceptually can be grouped into three categories: **DATA**, **PROCESS**, and **EXTERNAL**.

DATA Entity-Types

1. **DOCUMENT**, describing instances of human readable data collections. Typical DOCUMENTs are Form-1040 and 1984-Annual-Report.

2. **FILE**, describing instances of an organization's data collections. Typical FILEs are Payroll-File and Personnel-File.

3. **RECORD**, describing instances of logically associated data that belong to an organization. Typical RECORDs are Personnel-Record and Payroll-Record.

4. **ELEMENT**, describing instances of data belonging to an organization. Typical ELEMENTs are Social-Security-Number and Employee-Id.

PROCESS Entity-Types

5. **SYSTEM**, describing instances of collections of processes and data. Typical SYSTEMs are Personnel-System and Airline-Reservation-System.

6. **PROGRAM**, describing instances of automated processes. Typical PROGRAMs are Print-Paychecks and COBOL-Compiler.

7. **MODULE**, describing instances of automated processes that are either logical subdivisions of PROGRAM entities or independent processes that are called by PROGRAM entities. Typical MODULEs are Sort-Records and Check-Spelling.

EXTERNAL Entity-Types

8. **USER**, describing individuals or organizational components. Typical USERs are Finance-Department and John-Doe.

2.4.2 Basic Functional Relationship-Types

The collection of relationship-types provided by the Basic Functional Schema is listed in Appendix B. This collection includes virtually all the connections between Basic Functional entity-types that might prove useful to most organizations most of the time.

Most of these relationship-types are grouped into seven "relationship-class-types":

1. **CONTAINS**, describing instances of an entity being composed of other entities. A typical CONTAINS relationship-type is RECORD-CONTAINS-ELEMENT, which has as a possible instance the relationship Payroll-Record-CONTAINS-Employee-Name.

2. **PROCESSES**, describing associations between PROCESS and DATA entities. A typical PROCESSES relationship-type is SYSTEM-PROCESSES-FILE, which has as a possible instance the relationship Budget-System-PROCESSES-Cost-Center-File.

3. **RESPONSIBLE-FOR**, describing associations between entities representing organizational components and other entities, to denote organizational responsibility. A typical RESPONSIBLE-FOR relationship-type is USER-RESPONSIBLE-FOR-SYSTEM, which has as a possible instance the relationship Finance-Department-RESPONSIBLE-FOR-Payroll-System.

4. **RUNS**, describing associations between USER and PROCESS entities, illustrating that a person or organizational component is responsible for running a certain process. A typical RUNS relationship-type is USER-RUNS-PROGRAM, which has as a possible instance the relationship John-Doe-RUNS-System-Backup.

5. **GOES-TO**, describing "flow" associations between PROCESS entities. A typical GOES-TO relationship-type is PROGRAM-GOES-TO-PROGRAM which has as a possible instance the relationship Input-Program-GOES-TO-Processing-Program.

6. **DERIVED-FROM**, describing associations between entities where the target entity is the result of a calculation involving the source entity. A typical DERIVED-FROM relationship-type is DOCUMENT-DERIVED-FROM-FILE, which has as a possible instance Annual-Report-DERIVED-FROM-Program-File.

7. **CALLS**, describing "calling" associations between PROCESS entities. A typical CALLS relationship-type is PROGRAM-CALLS-MODULE, which has as a possible instance Main-Program-CALLS-Sort-Routine.

2.4.3 Basic Functional Attribute-Types

The attribute-types developed for inclusion in the Basic Functional Schema are the ones that organizations generally want applied to Basic Functional entity-types. Some attribute-types in this collection are common to all entity-types. These common attribute-types provide a synonym facility for IRDS names (see section 2.5.3) and also general documentation for entities (DESCRIPTION and COMMENTS).

Other Basic Functional attribute-types are associated with just one or a few entity-types. For example, NUMBER-OF-RECORDS, with possible attribute instance 240, is unique to the FILE entity-type.

As an additional feature of the Basic Functional Schema, certain relationship-types have attribute-types associated with them. For example, the attribute-type ACCESS-METHOD is associated with the relationship-types SYSTEM-PROCESSES-FILE, PROGRAM-PROCESSES-FILE, and MODULE-PROCESSES-FILE. Thus, for the relationship Input-Module-PROCESSES-Master-File, the attribute-type ACCESS-METHOD has a possible value of Indexed-Sequential.

The attribute-types in the Basic Functional Schema are listed in Appendix B.

In the remainder of this document, we will assume, unless otherwise specified, that the user organization's

IRDS includes the Basic Functional Schema, and we will use the contents of the Basic Functional Schema to provide examples of IRDS functionality.

2.5 ENTITY NAMES

The IRDS contains a flexible and generalized facility that enables users to assign different kinds of names to an entity. The different names serve distinct purposes, and several important conventions exist regarding them. The use of and distinction between access-name, descriptive-name, and alternate-name is basic to an understanding of the IRDS, and thus is discussed in this Chapter.

2.5.1 Purpose of Access-Names and Descriptive-Names

The most important name of an entity is its access-name. This name is the entity's primary identifier, and the structure of most commands and panels are based on it. In most organizations, the access-name will probably be terse, to minimize the number of keystrokes required to manipulate the IRD, thereby saving time and reducing the potential for error.

The access-name has two parts: an assigned access-name and a version-identifier. The structure of the version-identifier is discussed briefly in Chapter 3 and in more detail in Chapter 8. Normally a user will be responsible for specifying the assigned access-name of an entity. An option exists, however, to have the IRDS generate, using a standard algorithm, the assigned access-name for entities of a given type. This facility allows a user to enter new entities into the IRD before the final names of the entities have been determined. The initial entry of entities using the system-generated name option frequently will take place in an auxiliary IRD. Once the correct names of the entities are known, the user can modify the system-generated access-names and then move the entities, using functions available in the IRD-IRD interface facility, to the IRD that contains more standardized or precise names.

Terse access-names do have a disadvantage for the user, however, because they may not convey the meaning of the object represented by the entity. This terseness can cause problems, particularly in the preparation of reports for non-technical users and managers unfamiliar with the

contents of the IRDS. To address this problem, the IRDS allows users to assign a **descriptive-name** to an entity. The descriptive-name will normally be longer and more meaningful than the access-name. The structure of the descriptive-name is the same as that of the access-name, (i.e., there is an assigned descriptive-name and a version-identifier). SSN and Social-Security-Number are examples, respectively, of an assigned access-name and an assigned descriptive-name.

When output is generated from the IRDS, the user may specify whether the access-name, the descriptive-name, or both, are to appear.

2.5.2 Uniqueness of Access-Names and Descriptive-Names

Access-names and descriptive-names must be unique throughout a particular IRD. During the development of these Specifications, members of Technical Committee X3H4 and attendees at a user workshop sponsored by the National Bureau of Standards voted for uniqueness of name throughout an entire IRD, rather than name uniqueness only within an entity-type. This means that a user cannot, for example, have a FILE entity with an access-name Payroll and a RECORD entity also called Payroll.

Uniqueness of assigned access-names and assigned descriptive-names in the IRD simplifies the Command Language and Panel Interfaces. Except during the actual creation of new entities, the IRDS immediately recognizes the type of every entity whose name is included in a command or panel. Thus, the user is not repeatedly forced to specify an entity's type. Organizations that want to use assigned access-names or descriptive-names that are unique only within an entity-type could adopt an organization-defined naming convention. For example, all assigned names could be prefixed with a mnemonic of the appropriate entity-type name. Thus, prefixing all file names with "F-" and all record names with "R-" would allow two different entities with the "same" name Payroll to be represented as F-Payroll and R-Payroll. This convention would assure uniqueness in the IRD.

2.5.3 Alternate-Names

In addition to the assigned access-name and the assigned descriptive-name of an entity, a user, applying the Basic

Functional Schema, may specify alternate-names for an entity. The term alternate-name is used here in the same sense as the terms "synonym" and "alias" are often used. Alternate-names document the different names, if any, used to identify the same "real-world" object. Alternate-names are ordinary attributes of entities--they do not have version-identifiers, they do not have to be unique, different entities can have the same alternate-name, and the IRDS does not include any rules for the use of these names. For example, the ELEMENT whose access-name is Social-Security-Number might have alternate-names SSN, Soc-Sec-No, Soc_Sec_No, and Social_Security_Number.

ALTERNATE-NAME attributes are frequently used as part of an IDENTIFICATION-NAMES attribute-group, in conjunction with ALTERNATE-NAME-CONTEXT attributes. Thus, an organization might define IDENTIFICATION-NAMES attribute-groups to categorize its alternate-names according to programming language environment: (SSN, FORTRAN), (Soc-Sec-No, COBOL), (Soc_Sec_No, PL/I).

Data element naming conventions, within the framework of the IRDS, are discussed in [11].

3. OVERVIEW OF IRDS FUNCTIONS AND PROCESSES

As discussed in the Introduction, the IRDS Standard specifies two user interfaces: a Command Language Interface and a Panel Interface. An implementation of the IRDS will conform with the Standard if it has either one or both of these interfaces. This chapter presents an overview of the IRDS functions and processes specified for both user interfaces. Subsequent chapters provide more detail on the individual functions and processes. Unless stated otherwise, each specified facility is part of the Core IRDS.

3.1 POPULATING, MAINTAINING, AND REPORTING ON THE IRD

This section briefly describes the specified IRDS facilities that enable a user to populate and maintain an IRD, and retrieve individual or groups of entities with their associated relationships and attributes. The following section, 3.2, discusses maintenance and reporting facilities for the IRD Schema.

3.1.1 IRD Population and Maintenance

Facilities exist to create and delete entities and relationships in the IRD. Existing entities and relationships can also be modified by changing their attributes. Users can also modify existing assigned access-names and assigned descriptive-names. In addition, a user can copy an entity to create a new entity. This new entity will have the same attributes as the original entity. Optionally, the new entity can have relationships with the same entities to which the original entity is related. Versioning, discussed in Section 3.4, can be used to distinguish between the "copies."

3.1.2 IRDS Output

A General Output function produces reports and prepares query responses on specified IRD entities, their relationships, and their associated attributes. The precise format of IRDS reports will be defined by implementors of the Standard. The Standard specifies facilities that enable users to define: (1) the contents of a report or query

(i.e., the entities, attributes and relationships that should appear); (2) the kinds of names to be displayed; (3) the sequence of information; and (4) the report destination.

This report customization facility enables IRDS users to vary the contents of a report depending on the intended use. For example, a report for managers might display only some attributes, but would probably include such things as the descriptive-name and decoded rather than encoded attributes. A report prepared for the technical staff might contain the access-name rather than the descriptive-name, and show all attributes, in code form, associated with the selected entities and relationships. A response to an on-line query might be designed to display the minimum required information.

There are two special-purpose output facilities. One reports on all entities that would be affected by a change to a specified entity (e.g., all RECORD, MODULE, PROGRAM, and FILE entities that would be affected by a change to a given ELEMENT entity). A second facility produces output in Command Language format that can then be used as a training aid. To use this facility, an organization must have the Command Language Interface.

3.1.3 Entity-Lists

As an aid in preparing reports and queries and modifying the contents of the IRD, the IRDS Standard specifies facilities that enable users to develop lists of entities. If the entity-list is developed for report preparation purposes, the general output facilities can be used to customize the report(s).

A user first performs an initial retrieval, resulting in the selection of either: (1) all entities or (2) a group of entities based on the entities' access-names or descriptive-names, or on designated character strings within the name(s). An option also exists to include entities related to the ones retrieved.

This initial list is then "pared down" through the specification of other entity characteristics, such as: entity-type; relationships of the entities; attributes and attribute-groups; text attribute strings; and alternate-names. A user also can create a new entity-list by performing: the union of two or more entity-lists; the intersec-

tion of two or more entity-lists; the symmetric difference between two entity-lists; or the subtraction of one entity-list from another.

An entity-list can be named. This named list will remain available to the user who created it for the duration of an IRDS session. If an entity-list is not named, it will become, by default, the "current list." The current list is retained until: (1) another unnamed list is created (which will overwrite the old current list); (2) the current list is named; or (3) the IRDS session ends.

3.2 IRD SCHEMA MAINTENANCE, CUSTOMIZATION, AND REPORTING

As discussed in Chapter 2, the IRD Schema describes the structure of the IRD. For every entity, relationship, attribute, and attribute-group that can exist in the IRD, the IRD schema will contain a description of the corresponding entity-type, relationship-type, attribute-type, and attribute-group-type. The IRD schema itself is also described in terms of entities, relationships, and attributes. However, because of the potential for misunderstanding that could occur in discussions of the IRD schema versus the IRD, similar yet distinct terminology is used in the Specifications to describe the IRD Schema. Thus, the IRD schema contains:

- "Entities" called **meta-entities**.
- "Relationships" between meta-entities that are specified as **meta-relationships**.
- "Attributes," called **meta-attributes**, that document the characteristics of meta-entities and meta-relationships.
- "Attribute-Groups," called **meta-attribute-groups**.

IRDS extensibility provides an organization with the capability of customizing the schema, and thus the IRD. A user can, for example, add, modify, and delete meta-entities, meta-relationships, and their associated meta-attributes and meta-attribute-groups.

A user can report on the contents of the IRD schema. The format of this output can be tailored by specifying:

- The meta-entities that should be displayed, by name or type. The display of all meta-entities may also be requested.

- The meta-attributes and meta-relationships that should be displayed along with the selected meta-entities.

3.3 IRD-IRD INTERFACE

The IRD-IRD Interface provides a controlled mechanism for moving data from one standard IRDS implementation to another. The interface includes a set of four functions permitting selected parts of a source IRD to be transferred to a target IRD without affecting the integrity of either IRD.

One function specifies the set of entities and relationships that the user wants to extract from an existing IRD. These entities and relationships are copied to an "IRD export file" in a format specified by encoding rules based on ISO Standards 8824 [12] and 8825 [13]. This function also generates a file, in the appropriate ISO Standard format, that contains the schema of the source IRD.

Another function creates an "empty" IRD. (The creation of an empty IRD is required whenever a standard IRD is initialized.) When the empty IRD is created, the Minimal Schema (or some other IRD schema in export format) must be loaded. The user also can load IRD data that is in export format.

A third function checks the compatibility between the schema of the IRD in which the user is operating and another IRD schema that resides in either an IRD schema export file or another IRD. Since IRD schema compatibility depends on which schema is the source and which is the target, the user must specify this information.

Finally, a fourth function imports a previously exported IRD schema and an IRD subset into the target environment. This requires that the IRD subset reside in an IRD Standard export file, and the source IRD schema reside in the same or another export file. An IRD schema compatibility check is again performed automatically before execution of the IRD import.

3.4 IRD CONTROL FACILITIES

The Core IRDS contains four facilities that are important in populating and maintaining the IRD and in reporting on the contents of the IRD. These are: (1) Versioning; (2) Life-Cycle-Phases; (3) Quality-Indicators; and (4) IRD-Views. There are direct analogues of (1), (2), and (4) for control of the IRD Schema.

3.4.1 Versioning

An IRD entity describes a "real world" object. As the object changes, the corresponding entity will have to be changed. The Core IRDS allows a user to track such changes by using **revision-numbers**. These revision-numbers represent the chronology of the entity (and thus the chronology of the object that the entity describes) in the sense that the highest revision-number represents the most recent version of the entity. Each revision is stored as a distinct entity in the IRD.

A related concept is the desirability of identifying multiple "variations" of an entity. An example is the 5 versus 9 digit U.S. Postal Service Zip Code. Some files might still exist where the old 5-digit Zip Code was used, and others might contain the new 9-digit code. These two Zip Codes could be represented by distinct entities whose access-names show that one is a variation of the other. Variations are denoted by a **variation-name**, a specified string that must begin with an alphabetic character. A facility exists to control valid variation-names for each entity-type.

The revision-number and the variation-name are both appended to the assigned access-name and the assigned descriptive-name. The precise structure and the associated integrity rules are presented in Chapter 8.

3.4.2 Life-Cycle-Phases

IRDS Life-Cycle-Phases directly support the life cycle methodology used by an organization. A user can therefore document, in the IRD, the life-cycle-phase in which an entity exists. For example, different entities can be associated with the phases Requirements Analysis, Logical Database Design, etc.

The IRDS recognizes the following three classes of Life-Cycle-Phases:

- **UNCONTROLLED** -- multiple UNCONTROLLED life-cycle-phases can be defined and used by an organization.

- **CONTROLLED** -- there is only one CONTROLLED life-cycle-phase, named CONTROLLED-LIFE-CYCLE-PHASE. CONTROLLED entities are used in operational systems.

- **ARCHIVED** -- there is only one ARCHIVED life-cycle-phase, named ARCHIVED-LIFE-CYCLE-PHASE. As the name implies, archived entities are no longer used in operational systems, but are retained for historical or audit purposes.

In the Core IRDS, life-cycle-phases pertaining to the contents of the IRD (as opposed to the contents of the IRD Schema) are primarily for documentation purposes. Specific integrity rules and customization facilities to control the movement of entities through the life cycle are features of Module 4 (The Extensible Life-Cycle-Phase Facility) of the IRDS Specifications. See section 11.3.

3.4.3 Quality-Indicators

IRDS quality-indicators are similar in application to IRDS life-cycle-phases. A quality-indicator denotes such things as: (1) the level of standardization of element entities (e.g., program standard, agency or organization standard, national standard, or international standard); or (2) the degree to which the entity satisfies the organization's quality assurance or quality testing methodology. Each organization can define the quality-indicator names to be used with their IRDS.

3.4.4 IRD-Views

An IRD-view is a "window" or "gateway" into a life-cycle-phase that helps support project-oriented activities. For example, in the initial phases of Requirements Analysis for a large organizational system, different project teams or different analysts could use different IRD-views of the overall IRD to simplify and control their work.

3.4.5 Correspondence Between Views and Life-Cycle-Phases

There may be many overlapping IRD-views acting as windows into a given life-cycle-phase. Entities in a given life-cycle-phase thus may appear in many IRD-views. For example, the phase supporting Requirements Analysis may include multiple IRD-views for one or more project teams. However, all entities in a designated view must be in the same life-cycle-phase.

3.5 IRDS MODULES

As stated in the Introduction, the IRDS Standard includes specifications for five Modules that extend the capabilities of the Core Module. Although these Modules interface with the Core, they are independent of one another. Organizations may not need any of the Modules, or they may prefer to acquire and use any number and combination of the five. The five Modules are:

- Basic Functional Schema.
- IRDS Security.
- Extensible Life-Cycle-Phase Facility.
- Procedure Facility.
- Application Program Interface.

An additional Module, called the IRDS Services Interface, is currently under development.

3.5.1 Basic Functional Schema

The Basic Functional Schema was introduced in section 2.4, and is described in more detail in Appendix B.

3.5.2 IRDS Security

This Module defines an access control facility that allows organizations to restrict access to the IRD and IRD Schema content and functionality. The facility provides for two levels of access control:

- Global Security, which is based on functionality, type, and view. Read-only permission to access the IRD or IRD Schema, or more comprehensive levels of permission, may be granted. Access can also be controlled at the entity-type level (e.g., an organization might allow a certain user to read entities of all types, but to add, modify, and delete only ELEMENTs.

- **Entity-Level Security**, which controls access to individual entities. This feature operates as an additional layer of security beyond Global Security. An organization using Entity-Level Security can assign IRDS users READ and/or WRITE privileges for specific entities. For users not having the appropriate READ permission, any entity with the additional security is treated as though it does not exist in the view in which the user is working. For users having READ but not WRITE permission, secured entities can be examined but not modified or deleted.

3.5.3 Extensible Life-Cycle-Phase Facility

This Module extends the life-cycle-phase facilities of the Core IRDS by implementing integrity rules and customization features needed to control the movement of entities through the life-cycle. The structure of the life-cycle itself can be customized to match the organization's preferred methodology. Based on this structure, entities can be moved from UNCONTROLLED to CONTROLLED phases, from CONTROLLED to ARCHIVED phases, and from CONTROLLED to UNCONTROLLED phases, with the organization retaining full control over the integrity constraints that regulate this movement.

3.5.4 Procedure Facility

This Module provides the ability to define, store, maintain, and execute procedures involving the IRD and the IRD Schema. These procedures are composed of IRDS Commands, along with flow-control and assignment statements. A procedure may also call another procedure. A set of built-in functions is provided with this Module for character and numerical manipulations, and to extract system information.

The operation of this Module requires the presence of the Command Language interface.

3.5.5 Application Program Interface

This Module provides an interface through which IRDS Commands and resulting output can be passed between the IRDS and programming languages that have a CALL feature. An organization can develop software to use IRD data for special purposes. In this situation, the IRDS is treated by the user-developed application as a subroutine.

3.5.6 IRDS Services Interface

This Module defines a specific protocol for an interface through which software external to an IRDS can directly access the IRD and IRD Schema. It provides the means to construct an environment in which the IRDS can be truly "active."

4. POPULATING AND MAINTAINING THE IRD

This chapter describes the functions specified as part of the Core IRDS to add, modify, and delete entities and relationships in the IRD. The copy function is also described.

4.1 ENTITIES

This section presents a technical summary of the IRDS functions that: add or create new entities in the IRD; modify an existing entity by adding, changing, or deleting the entity's attributes and attribute-groups; and delete an entity and its associated attributes and attribute-groups.

4.1.1 Adding Entities

Using this function, a user can add new entities to the IRD. The most important aspects of creating a new entity are:

- Declaring the type of the entity. This designated entity-type must be one that exists in the IRD schema. The type may be in the IRDS Minimal Schema, the Basic Functional Schema (if present), or may have been added to the IRD schema by the user organization.
- Designating the assigned access-name of the entity.
- Optionally, assigning a descriptive-name to the entity.
- Declaring attributes and attribute-groups for the new entity.

As discussed in Chapter 2, the access-name and the descriptive-name each have two parts: the assigned access-name or descriptive-name and the version-identifier. Two methods exist for assigning an access-name to the entity. Either the assigned access-name is specified by the user or it is automatically generated by the IRDS. To be valid, the user assigned access-name must satisfy the following rules:

1. The character string representing the assigned access-name must conform to the length and picture rules in

the IRD schema. For entities of each type, these rules are specified by MINIMUM-ASSIGNED-ACCESS-NAME-LENGTH, MAXIMUM-ASSIGNED-ACCESS-NAME-LENGTH, and PICTURE meta-attributes in the schema.

2. The name must not exist in the IRD either as an assigned access-name or an assigned descriptive-name.

3. If the assigned access-name is to be system-generated, the user must specify the entity-type. The name assigned by the system will be displayed to the user.

4. A user-assigned access-name or descriptive-name must not lead to a potential conflict with a system-generated assigned access-name. For example, if ELEMENTs have system generated names RE001, RE002, ..., a conflict could exist if a user assigned an access-name or descriptive-name of RE004 to a new RECORD entity. In this situation, the IRDS would not allow RE004 to be added by a user.

While adding entities, the user may specify an assigned descriptive-name, to which the above rules 1, 2, and 4 also apply.

A user may also specify attributes and attribute-groups for the new entity. The user must provide both the names of the attribute-types or attribute-group-types and the values assigned.

4.1.2 Modifying Entities

This function is used to change the attributes and attribute-groups of an existing entity. Use of this function is restricted to a single entity at a time (i.e., the function does not operate against a list of entities). Execution of the modify entity function may result in:

- Creation of new attributes and attribute-groups.
- Replacing existing attributes or attribute-groups.
- Deletion of existing attributes and attribute-groups.

There is an option on the modify entity function that causes the existing entity to remain unchanged, and instead creates a new entity with the specified modifications. This

new entity has the same assigned access-name as the existing entity, but has a different version-identifier. A new descriptive-name for the new version is constructed if the original entity had a descriptive-name. This new descriptive-name will have the same assigned descriptive-name as the one belonging to the original entity, but its version-identifier will be set equal to the version-identifier of the access-name of the new entity.

When a new entity is created, new relationships can be created to correspond to the relationships in which the original entity participates. These new relationships will have the same attributes and attribute-groups as the existing relationships.

4.1.3 Deleting Entities

A user may specify one or more entities to be deleted by specifying any of the following:

- The access-names of entities to be deleted.
- Entity selection criteria that will result in the creation of a new entity-list.
- The name of an entity-list created earlier in the session. The current list may also be specified.

Each entity specified, through whichever of the above mechanisms, must exist in the IRDS. An option exists to delete entities that participate in relationships. In this case, every relationship in which each specified entity participates will also be deleted from the IRD.

4.2 RELATIONSHIPS

This section summarizes the IRDS functions that are specified to: add relationships between entities; modify existing relationships; and delete relationships.

4.2.1 Adding Relationships

This function creates new relationships in the IRD. The most important aspects of creating a new relationship include designating:

- The entities that are to be members of the relationship.
- The relationship-type or relationship-class-type.
- Optionally, attributes and attribute-groups for the new relationship.

In creating a new relationship, if both entities that are to be members of the relationship exist, the user simply specifies the access-names of these entities and states the new relationship. If one entity exists and the other does not, the user again specifies two access-names, but the one referring to the non-existing entity includes a specification of that entity's type. This will result in the automatic creation of a new entity identified by the second access-name.

The designated relationship-type or relationship-class-type must be one that already exists in the IRD schema.

4.2.2 Modifying Relationships

Using this function, a user can:

- Change a relationship's attributes and attribute-groups.
- Create new attributes and attribute-groups.
- Delete existing attributes and attribute-groups.

To modify relationships in any of these ways, the user must specify:

- The type or class-type of the relationship.
- The access-names of the member entities of the relationship to be modified.
- The attributes and attribute-groups to be added, changed, or deleted.

4.2.3 Deleting Relationships

Using this function, a user may delete relationships by specifying any of the following:

- One or more existing relationships.
- Relationship selection criteria. These criteria allow the user to select relationships for deletion based on:
  - The entities that participate in the relationships. The entities are specified to designate the relationship to be deleted, but the entities themselves are not deleted.
  - Particular relationship-types or relationship-class-types.
  - Certain attributes and attribute-groups associated with a relationship.
  - The existence of a particular character string within a text attribute associated with a relationship.

4.3 COPYING ENTITIES AND RELATIONSHIPS

A user can create a new entity with the same attributes, the same attribute-groups, and the same relationships as an existing entity. The user must specify:

- The access-name (i.e., the assigned access-name and the version-identifier) of the entity to be copied.
- The access-name of the entity to be created.

Optionally, the user may also designate:

- That the existing entity's relationships are also to be copied.
- A descriptive-name for the new entity.

The new entity created by the copy function is designated in one of the following ways:

- By specifying a valid assigned access-name that does not currently exist in the IRD.

- By specifying the new-version option. The existing entity is copied to a new entity whose assigned access-name is the same as that of the existing entity but whose version-identifier is different.

- By specifying a null mark. This is valid only if the type of the existing entity is defined in the IRD schema to have system-generated assigned access-names.

The user may specify an assigned descriptive-name for the new entity. If one is specified, the name must not exist in the IRD either as an assigned access-name or as an assigned descriptive-name.

If the user designates that the entity's relationships are to be copied, a new relationship is established corresponding to each existing relationship. For some exceptions to this rule, see the IRDS specification of the Copy function [1].

The new entity's access-name will be the first (or second, as appropriate) member of any new relationship. The other member of the new relationship will be the same as that in the original relationship.

5. IRD OUTPUT

IRDS users may employ a **General Output** function to produce output on IRD entities, their associated relationships, and the attributes of these entities and relationships. The contents of the output, as discussed in Section 5.1, can be specified by the user. The output can be in response to an on-line query, or in the form of a report.

Another output function, the **Impact-of-Change** function, reports on those entities that might be affected in some manner by a change to a specified entity. An **Output Syntax** function produces output on selected entities in the same format as that used to create the entities using the Command Language Interface.

5.1 GENERAL OUTPUT

The following six steps are involved in specifying the execution of an output function. Steps 2 and 4 are always required, the other steps are optional. System defaults exist for all optional steps.

1. Specifying the views to which retrieval applies.

2. Selecting the entities. The selection criteria may be specified by the user at the time the operation is entered. These criteria include:

   - The type(s) of entities to be retrieved.

   - Character strings within the assigned access-names or descriptive-names.

   - Character strings within the associated version-identifiers.

   - Designated attributes or attribute-groups.

   - Life-cycle-phases or quality-indicators.

   - Relationships to other entities.

   The selection criteria may also be based on an existing entity-list. Entity-list functions are explained later in this chapter.

3. Sorting the selected entities. A series of sort parameters are available to designate the sort order of the selected entities. Each parameter may be requested on an ascending or descending basis within that parameter. The sort parameters include the following:

   - Entity-Type.
   - Non-repeating attribute-types associated with entity-types.
   - Life-Cycle-Phase.
   - Assigned Access-Name.
   - Complete Access-Name.
   - Assigned Descriptive-Name.
   - Complete Descriptive-Name.
   - Version-identifier associated with the assigned access-name or assigned descriptive-name.

4. Designating what information is to be displayed for each selected entity. For this show function, the information includes:

   - The kinds of entity names (i.e., access-names, descriptive-names, alternate-names).
   - The life-cycle-phase for each entity.
   - One or more of an entity's attributes or attribute-groups. There is an option to show all attributes. For text attributes, the numbers of the text lines to be displayed may be specified.
   - One or more of the relationships in which an entity participates. There are options to show all relationships, to show either forward or inverse relationships, and to show all relationships of a particular relationship-class. The output of all, some, or none of the relationship's attributes may be specified.

5. Routing the output contents to a particular destination. A system defined destination will be used if no destination is specified.

6. Assigning a character string to be used as a title for the output. This title can appear either on the first page or on every page of the output.

The following example demonstrates a potential use of the General Output function. The syntax used is that of the IRDS Command Language:

Suppose a user wished to report on all version-identifiers associated with an assigned access-name of PROGRAM-Z. (As discussed in Chapter 3, the version-identifier consists of two parts: a variation-name and a revision-number.) The user would first specify, in the entity selection criteria, the appropriate assigned access-name, and would use the "wild-card" designation provided to select all entities with that particular assigned access-name, as in:

```
select entities with
    access-name = PROGRAM-Z (*:*)
```

In this example, (*:*) designates all revision-numbers and all variation-names.

Suppose the user wishes to sort the selected entities based first on entity-type, then on variation-name, then assigned access-name, and finally on revision-number. Logically, this is specified in the following way:

```
entity-type (ascending),
variation-name (ascending),
assigned access-name (ascending),
revision-number (descending).
```

Now, using the show capability, the user specifies the information that is to be output for each of the selected entities. To see the assigned access-name, the assigned descriptive-name, and all attributes of each entity, the user would specify the following:

show assigned access-name
show assigned descriptive-name
show all attributes.

After the user specifies any remaining options, the IRDS will produce the desired output.

5.2 OUTPUT IMPACT-OF-CHANGE

In addition to the facilities discussed in the previous section, two additional options exist for reporting Impact-of-Change. The first, called the **Cumulative Impact-of-Change** option, will produce a single list of all distinct entities that will be affected by a change to any of the selected entities. The second, called the **Individual Impact-of-Change option**, will produce separate lists of entities for each of the originally specified entities. Each of these lists represents the set of entities that will be affected by a change to that specified entity. An example of the distinction between these options is:

> Suppose the specified entity selection criteria resulted in an initial list consisting of two entities, Pers-File and Acct-File. The selection of the Cumulative Impact-of-Change option would result in a single list of all distinct entities that would be affected by a change to either Pers-File or Acct-File.
>
> The selection of the Individual Impact-of-Change option, however, would result in the output of two lists of entities, one containing the entities that would be affected by a change to Pers-File, and the second containing the entities that would be affected by a change to Acct-File.

5.3 OUTPUT SYNTAX

The Output Syntax function produces output that includes, for each entity selected, all the information about the entity that might have been entered into the IRD with the use of either the **Add Entity** or **Add Relationship** commands. The output structure for each entity and relationship will reflect the same basic order and format as that in which the information might have been originally input.

The output for this function may be shown in one of two formats, as requested by the user.

The first format, which displays each entity's relationships immediately after displaying the entity's attributes, provides the information in the following order:

```
ENTITY-1
   [ All ENTITY-1 information, in the same general
     format as that used in the ADD ENTITY command ]

   RELATIONSHIP-1 in which ENTITY-1 participates
       [  All RELATIONSHIP-1 information, in the same
                general format as that used in the ADD
          RELATIONSHIP command ]
             .
             .
             .
   RELATIONSHIP-j in which ENTITY-1 participates
       [ All RELATIONSHIP-j information ]
     .
     .
     .
ENTITY-n
   [ All ENTITY-n information ]

   RELATIONSHIP-1 in which ENTITY-n participates
       [ All RELATIONSHIP-1 information ]
     .
     .
     .
   RELATIONSHIP-k in which ENTITY-n participates
       [ All RELATIONSHIP-k information ]
```

The second format displays all entities and their attributes first, followed by all distinct relationships in which the entities participate:

```
ENTITY-1
   [ All ENTITY-1 information, in the same general
        format as that used in the ADD ENTITY command ]
     .
     .
     .
ENTITY-n
   [ All ENTITY-n information ]
```

RELATIONSHIP-1 in which any entity above participates
[All RELATIONSHIP-1 information, in the same format as that used in the ADD RELATIONSHIP command]

.
.
.

RELATIONSHIP-j in which any entity above participates
[All RELATIONSHIP-j information]

The major difference between the two formats is that the second will not duplicate the display of any relationships that are shared by the set of selected entities.

The user, in specifying the report or query contents, also can designate the relationships that are to be output along with the specified entities. The relationships may be specified in one of three ways:

- **All relationships.** In this case all relationships in which the specified entities participate will be output.

- **Relationships of certain types.** In this case, the user may specify one or more valid relationship-types.

- **No relationships.** If this option is specified, then no relationships will be output, and thus the designation of the output format is no longer relevant.

In a given IRDS, the Output Syntax function is available only if the Command Language Interface has been implemented.

5.4 ENTITY-LISTS

The IRDS allows a user to create and manipulate lists of access-names based on user-specified selection criteria. These "entity-lists" may then be input to other IRDS output functions and certain maintenance functions. The entities contained in an entity-list are always a subset of those contained in IRD-views specified by and authorized for the user.

5.4.1 Creating New Entity-Lists

To create a new entity-list without using existing entity-lists, a user will perform the following sequence of steps:

1. Select a set of entities. This set is specified in one of the following ways:

   - By selecting all entities in the view(s) indicated.
   - By selecting entities by their access-names or a substring within their assigned access-names.
   - By selecting entities by their descriptive-names or a substring within their assigned descriptive-names.
   - By selecting entities according to their relationship to other specified entities, and the nature of the relationship with these other entities (i.e., either direct or indirect).

2. Enter restriction criteria to reduce the initial set. These criteria allow restricting the set:

   - To certain entity-types.
   - To those entities that participate in particular relationship-types.
   - To those entities that contain certain attributes.
   - To those entities that contain certain attribute-groups.
   - To those entities that have a particular substring within a certain text attribute.
   - To certain life-cycle-phases.
   - To those entities that contain certain audit attributes.
   - To those entities that have a particular alternate-name.

3. Designate a name for the newly created entity-list. The specified name must be one that has not previously been assigned to an entity-list during the same IRDS session. If the user does not explicitly assign a name to the new entity-list, the IRDS designates the entity-list as the current list.

5.4.2 Entity-List Set Operations

The IRDS allows the set operations of **union**, **intersection**, **difference**, and **subtraction** to be performed on two (or sometimes more) existing entity-lists to produce a new entity-list.

To execute any of these operations, the user specifies the following:

- For union and intersection, the names of two or more existing entity-lists; for difference and subtraction, the names of precisely two existing entity-lists. In each case, one of the input entity-lists may be the current list.

- The name of an entity-list into which the resulting set of entities will be placed. If no such name is specified, the resulting entity-list will be designated as the current list.

The following are examples of how these functions operate:

Suppose that three entity-lists contain the following entities:

Entity-List-A	Entity-List-B	Entity-List-C
Entity-1	Entity-1	Entity-1
Entity-3	Entity-2	Entity-3
Entity-4	Entity-5	Entity-4
Entity-6	Entity-6	Entity-5
Entity-7		

Entity-list union of Entity-List-A, Entity-List-B, and Entity-List-C will result in the creation of a new Entity-List-D, which will contain all entities that appear in any of the input lists, except for

duplicates. Specifically, the results of the entity-list union will be:

Entity-List-D

Entity-1
Entity-2
Entity-3
Entity-4
Entity-5
Entity-6
Entity-7

Entity-list intersection of Entity-List-A, Entity-List-B, and Entity-List-C will result in the creation of Entity-List-E, which will contain those entities that appear in each of the input lists. Specifically, the results of the entity-list intersection will be:

Entity-List-E

Entity-1

Entity-list difference of Entity-List-A and Entity-List-B will create the new Entity-List-F, which will contain precisely those entities that are not common to both input lists. Specifically, the results of the entity-list difference will be:

Entity-List-F

Entity-2
Entity-3
Entity-4
Entity-5
Entity-7

Entity-list subtraction, in which Entity-List-C is subtracted from Entity-List-A, will result in the creation of the new Entity-List-G, which will contain precisely those entities that are in Entity-List-A but not in Entity-List-C. Specifically, the results of the entity-list subtraction will be:

Entity-List-G

Entity-6
Entity-7

5.4.3 Other Entity-List Functions

The **name current list** function allows an IRDS user to assign an entity-list name, (one that does not currently exist), to the current list. The current list can be empty.

The **output entity-list** function is used to display the contents of a specified entity-list created by a user during a particular IRDS session. The function will list the access-names of all entities contained in the entity-list. The contents of the current list may also be displayed.

The **output entity-list names** function will display the names of all entity-lists that a particular user has defined during the current session. For each entity-list name, the number of access-names within the entity-list will also be shown.

6. IRD SCHEMA MAINTENANCE AND OUTPUT

In this chapter, we will expand upon our discussion of the IRD Schema and its description. These concepts were introduced in Chapter 2, and illustrated as the two top layers in Figure 2. Readers of this overview who are not interested in the specific mechanisms for changing or supplementing the Minimal Schema and the Basic Functional Schema can skip this chapter and proceed to Chapter 7.

We have seen that the IRD schema includes ENTITY-TYPEs, RELATIONSHIP-TYPEs, RELATIONSHIP-CLASS-TYPEs, ATTRIBUTE-TYPEs, and ATTRIBUTE-GROUP-TYPEs. These types are specified as meta-entities in the schema. The meta-entities are linked by meta-relationships, and both meta-entities and meta-relationships can have meta-attributes and meta-attribute-groups associated with them.

In the same way that the IRD schema describes the entities, relationships, relationship-classes, attributes, and attribute-groups in the IRD, the IRD schema itself is described using the terms meta-entity, meta-relationship, meta-relationship-class, meta-attribute, and meta-attribute-group.

IRD schema maintenance functions are also addressed in this chapter, including methods for adding, deleting, and modifying meta-entities and meta-relationships. The chapter ends with a description of the various modes in which a user can specify IRD schema output.

6.1 THE CONTENT OF THE IRD SCHEMA

As discussed previously, the IRD schema contains "entities" called meta-entities. These meta-entities can be linked by meta-relationships, and both meta-entities and meta-relationships can have meta-attributes and meta-attribute-groups associated with them.

6.1.1 Meta-Entities

A meta-entity can be any of the following:

- An ENTITY-TYPE.
- A RELATIONSHIP-TYPE.
- An ATTRIBUTE-TYPE.
- A RELATIONSHIP-CLASS-TYPE.
- An ATTRIBUTE-GROUP-TYPE.
- An ATTRIBUTE-TYPE-VALIDATION-PROCEDURE.
- An ATTRIBUTE-TYPE-VALIDATION-DATA.
- A VARIATION-NAMES-DATA.
- An IRD-PARTITION.
- A QUALITY-INDICATOR.
- An IRDS-DEFAULTS.
- An IRDS-LIMITS.
- An IRDS-RESERVED-NAMES.
- A NAMES.

Examples in the Basic Functional Schema of ENTITY-TYPE meta-entities are ELEMENT and RECORD. Examples of RELATIONSHIP-TYPE meta-entities are PROGRAM-CALLS-MODULE and RECORD-CONTAINS-ELEMENT. Examples of ATTRIBUTE-TYPE meta-entities are DESCRIPTION, LENGTH, and NUMBER-OF-LINES-OF-CODE. An example of an ATTRIBUTE-GROUP-TYPE meta-entity is ALLOWABLE-RANGE.

6.1.2 Meta-Relationships

Meta-relationships are associations between meta-entities. The name of a meta-relationship-type consists of the name of the first meta-entity-type, then the name of the meta-relationship-class-type, followed by the name of the second meta-entity-type, all connected by "-". The meta-

relationship itself is denoted by the name of the first component meta-entity, the name of the meta-relationship-type, and the name of the second component meta-entity.

For example, to document the fact that LENGTH is an allowable attribute-type for ELEMENT (i.e., that ELEMENTs can have LENGTH attributes), we need to associate the meta-entity LENGTH with the meta-entity ELEMENT. In the IRD Schema, this is done by saying that there exists the meta-relationship ELEMENT ENTITY-TYPE-CONTAINS-ATTRIBUTE-TYPE LENGTH. Figure 4A illustrates this meta-relationship.

Similarly, to associate an attribute-type with a relationship-type, a meta-relationship is constructed. For example, the attribute-type RELATIVE-POSITION is associated, in the Basic Functional Schema, with the relationship-type RECORD-CONTAINS-ELEMENT. (This association documents the relative position of an ELEMENT in a RECORD.) This is implemented in the IRD Schema by establishing the meta-relationship RECORD-CONTAINS-ELEMENT RELATIONSHIP-TYPE-CONTAINS-ATTRIBUTE-TYPE RELATIVE-POSITION. Figure 4B shows this meta-relationship.

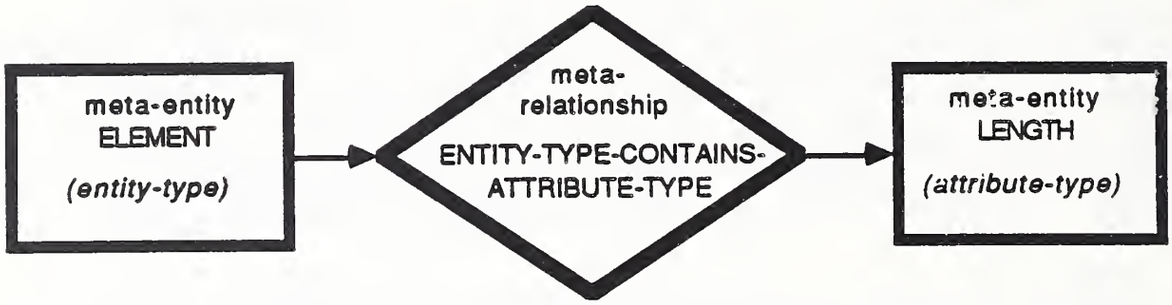
The fact that two particular entity-types are the components of a particular relationship-type is represented in the IRD Schema by <u>two</u> meta-relationships, one linking each of the two ENTITY-TYPE meta-entities to the RELATIONSHIP-TYPE meta-entity. For example, PROGRAM-CALLS-MODULE is a meta-entity in the schema. Without further information, however, the IRDS does not infer that PROGRAM, CALLS, or MODULE are in any way associated with the given relationship-type. The association must be made explicit, with one meta-relationship between PROGRAM and PROGRAM-CALLS-MODULE, and another between MODULE and PROGRAM-CALLS-MODULE. This use of two meta-relationships to implement the association of a relationship-type with its component entity-types is illustrated in Figure 5.

<u>6.1.3 Meta-Attributes and Meta-Attribute-Groups</u>

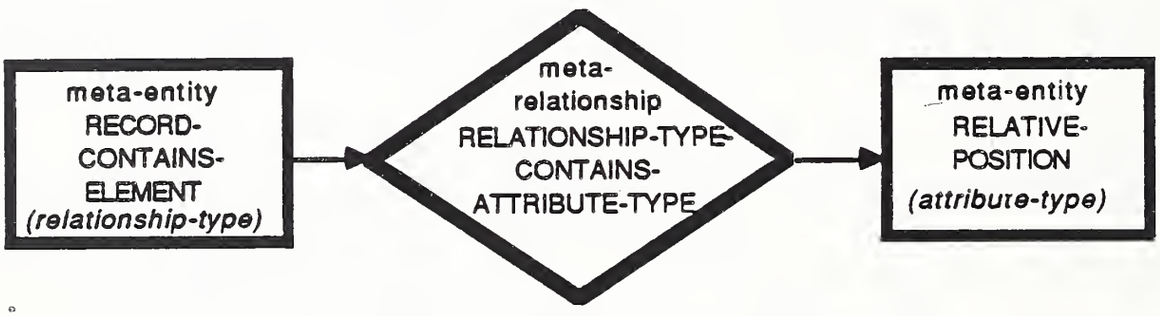
Meta-attributes and meta-attribute-groups perform a descriptive role with respect to meta-entities and meta-relationships. Generally speaking, there are four kinds of meta-attributes and meta-attribute-groups:

1. **Documentation** meta-attributes. Using them, a user can document the purpose of the meta-entity. For example,

Examples of Meta-Relationships
Associating Two Meta-Entities



4A. An Entity-Type Associated with an Attribute-Type



4B. A Relationship-Type Associated with an Attribute-Type

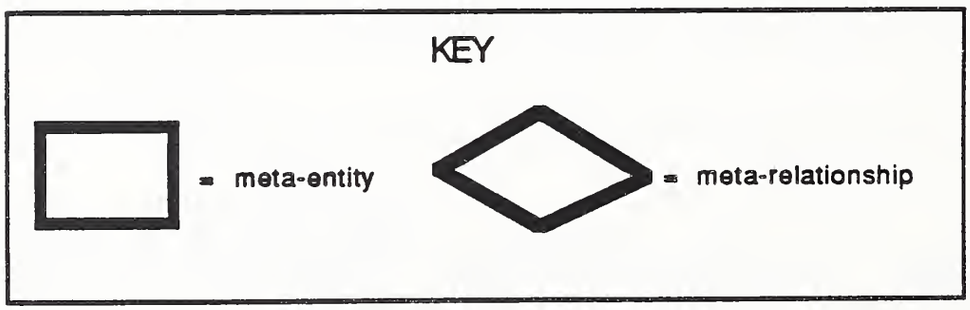


Figure 4

Example of a Relationship-Type Meta-Entity
Implemented by Two Meta-Relationships

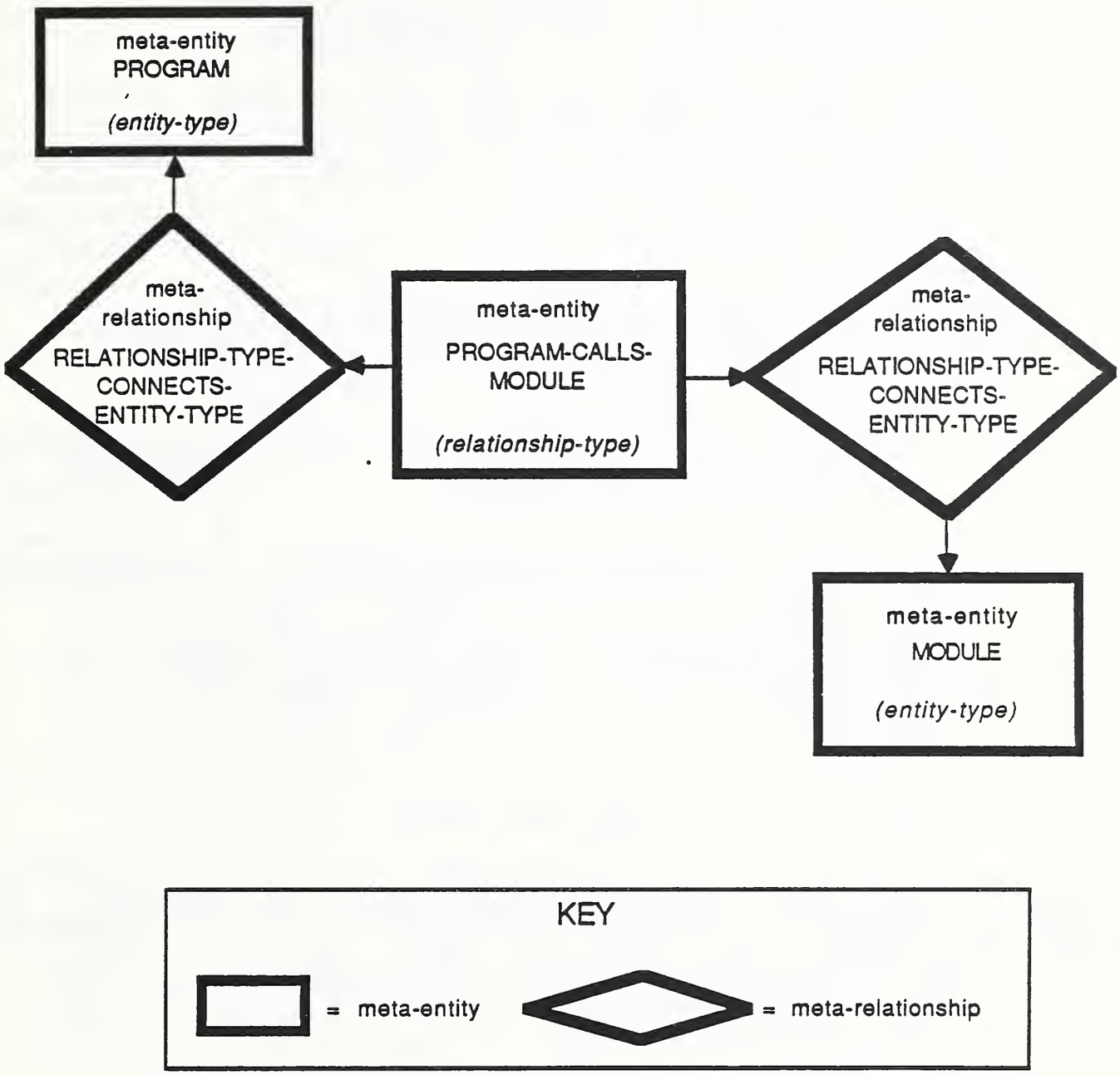


Figure 5

the Minimal Schema contains the PURPOSE meta-attribute-type.

2. **Audit** meta-attributes and meta-attribute-groups, which are analogous to the audit attributes and attribute-groups in the IRD. Examples are the meta-attribute-types ADDED-BY and NUMBER-OF-TIMES-MODIFIED and the meta-attribute-group-type DATE-TIME-ADDED.

3. **IRD Schema control** meta-attributes, which provide certain controls over what can and cannot be done in the schema. For example, some meta-attributes can be used to prevent deletion of a meta-entity, or can be structured to require the use of a privileged function to delete a meta-entity. Examples are SYSTEM-LOCK and IMPLEMENTATION-LOCK.

4. **IRD control** meta-attributes, which are used to impose rules on the IRD. These include meta-attributes that specify:

   - Allowable lengths (minimum and maximum) of names of entities of a given type (e.g., MAXIMUM-ENTITY-ASSIGNED-ACCESS-NAME-LENGTH).

   - Allowable lengths of the attributes of a given type (e.g., MINIMUM-ATTRIBUTE-LENGTH).

   - Whether a particular entity-type can have more than one attribute of a given type, and if so, what the maximum allowable number is (e.g., SINGULAR).

6.1.4 An Example of Part of an IRD Schema

To illustrate how meta-entities, meta-relationships, meta-attributes, and meta-attribute-groups work together, Figure 6 shows a part of the Basic Functional Schema involving FILE. The entity-type FILE, the relationship-types FILE-CONTAINS-RECORD and USER-PROCESSES-FILE, the attribute-type NUMBER-OF-RECORDS, and the attribute-group-type DATE-TIME-LAST-MODIFIED are all meta-entities.

As the figure shows, the relationship-types FILE-CONTAINS-RECORD and USER-PROCESSES-FILE are connected to FILE by means of RELATIONSHIP-TYPE-CONNECTS-ENTITY-TYPE meta-relationships, indicating that FILE does in fact

Subset of the IRDS Basic Functional Schema

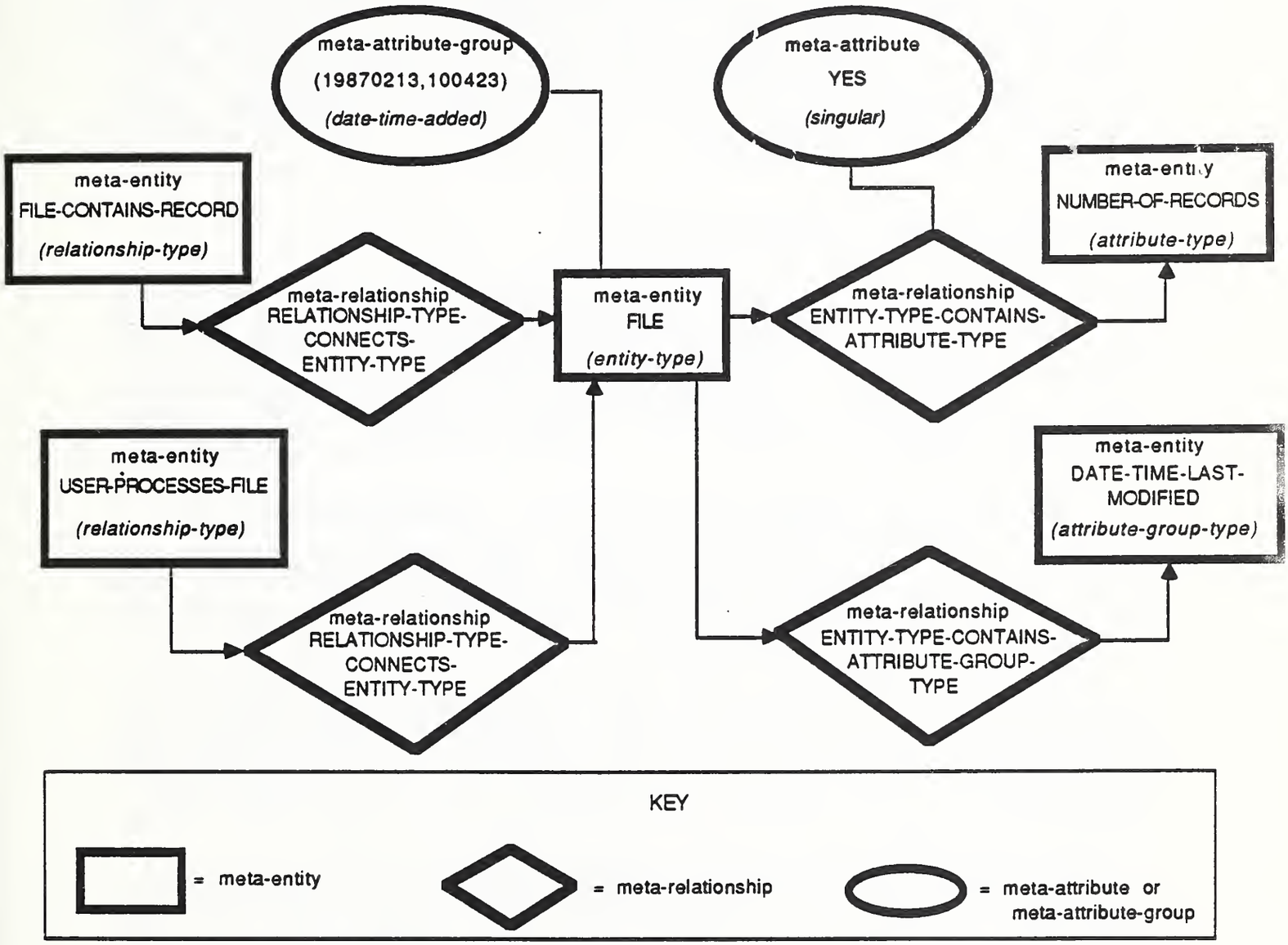


Figure 6

participate in these two relationship-types. The ENTITY-TYPE-CONTAINS-ATTRIBUTE-TYPE meta-relationship between FILE and NUMBER-OF-RECORDS indicates that this attribute-type is associated with FILE. The ENTITY-TYPE-CONTAINS-ATTRIBUTE-GROUP-TYPE meta-relationship between FILE and DATE-TIME-LAST-MODIFIED documents that this audit attribute-group-type is associated with FILE. Also illustrated are two meta-attributes: a DATE-TIME-ADDED meta-attribute-group associated with the meta-entity FILE; and a SINGULAR meta-attribute, associated with the meta-relationship linking FILE and NUMBER-OF-RECORDS.

6.1.5 Other IRD Schema Constructs

The IRDS provides facilities that allow an organization to control the values, or ranges of values, of non-textual attribute-types. ATTRIBUTE-TYPE-VALIDATION-PROCEDURE meta-entities represent procedures that can be used to validate these attributes. ATTRIBUTE-TYPE-VALIDATION-DATA meta-entities contain sets of valid values for specific attribute-types.

The Minimal Schema contains one IRDS-DEFAULTS and one IRDS-LIMITS meta-entity. These are used to store the defaults and numerical limits for meta-attributes and IRDS names, and are used by the IRDS if appropriate values are not specified by the IRDS user.

The Minimal Schema also contains an IRDS-RESERVED-NAMES meta-entity, to specify which assigned-access-names of entities and meta-entities are reserved by the IRDS and cannot be changed. A single NAMES meta-entity contains a description of the rules for naming both entities and meta-entities.

The meta-entities that support the Version-Identifier, Life-Cycle-Phase, and Quality-Indicator facilities will be discussed in Chapter 8, where the IRDS naming and control facilities are described.

6.2 IRD SCHEMA MANIPULATION

The IRDS user can manipulate and redefine the IRD schema by adding, modifying, and deleting meta-entities and meta-relationships.

6.2.1 Adding Meta-Entities

An IRDS user can add a new meta-entity (and a set of associated meta-attributes) to the schema. Meta-entities of the following types may be added:

- ENTITY-TYPE.
- RELATIONSHIP-TYPE.
- RELATIONSHIP-CLASS-TYPE.
- ATTRIBUTE-TYPE.
- ATTRIBUTE-GROUP-TYPE.
- ATTRIBUTE-TYPE-VALIDATION-PROCEDURE.
- ATTRIBUTE-TYPE-VALIDATION-DATA.
- VARIATION-NAMES-DATA.
- IRD-PARTITION.
- QUALITY-INDICATOR.
- IRDS-DEFAULTS.
- IRDS-LIMITS.
- IRDS-RESERVED-NAMES.
- NAMES.

A new meta-entity may not have the same meta-entity-assigned-access-name as an existing meta-entity.

6.2.2 Modifying Meta-Entities

A user can associate new meta-attributes with meta-entities, and modify and delete existing meta-attributes of meta-entities. For example, the allowable length of an assigned-access-name for a particular meta-entity can be modified by changing the MINIMUM-ENTITY-ASSIGNED-ACCESS-

NAME-LENGTH and MAXIMUM-ENTITY-ASSIGNED-ACCESS-NAME-LENGTH meta-attributes for that meta-entity.

6.2.3 Deleting Meta-Entities

A user can delete an existing meta-entity from the schema.

Several rules apply to this operation. The meta-entity to be deleted:

- Must not have instances in the IRD.
- Must not have a value of ON for the SYSTEM-LOCK meta-attribute. (For each meta-entity, the value of this meta-attribute is set and maintained by the implementation. An ON value implies that the presence and precise definition of the meta-entity is necessary to the operation of the IRDS. An ON value cannot be changed by the user organization.)
- Must not have a value of ON for the IMPLEMENTATION-LOCK meta-attribute. (This meta-attribute allows the organization to have additional control over the modification of the IRD schema.)

6.2.4 Adding Meta-Relationships

As stated earlier, meta-relationships are associations between meta-entities. When adding a new meta-relationship to the schema, the user specifies the name of the meta-relationship, and the meta-attributes to be associated with the meta-relationship.

6.2.5 Modifying Meta-Relationships

A user can change the meta-attributes of an existing meta-relationship in the schema.

The user supplies the name of the meta-relationship, and then associates new meta-attributes, or modifies or deletes existing meta-attributes, of the meta-relationship.

6.2.6 Deleting Meta-Relationships

To delete an existing meta-relationship from the schema, the user specifies the name of the meta-relationship.

6.2.7 Modifying Meta-Entity Names

A modify meta-entity-access-name function is used to change the assigned-access-name of an existing meta-entity in the schema. The new meta-entity-assigned-access-name may not already appear in the IRD schema as a meta-entity-assigned-access-name or a meta-entity-substitute-name.

The function has the effect of deleting the existing meta-entity from the IRD schema and creating a new meta-entity. The new meta-entity will:

- Have the same meta-attributes and meta-attribute-groups as those of the predecessor meta-entity.
- Participate in the same meta-relationships as the predecessor meta-entity.
- Have the same IRD instances as did the predecessor meta-entity.

An analogous modify meta-entity-descriptive-name function is used to change the assigned-descriptive-name of a meta-entity.

6.2.8 Copying Meta-Entities

This function creates a new meta-entity with the same user specified meta-attributes as an existing meta-entity.

6.2.9 IRD Schema Testing

To allow an organization to test proposed changes to a schema, the IRDS provides three related functions:

- **Deactivate IRDS**, which stops all IRDS activity and restricts access to the IRD Schema to a single user.
- **Activate IRDS**, which enables IRDS activity by "cancelling" the Deactivate IRDS. The IRDS-user who

issues this function must be the same IRDS-user who issued the Deactivate IRDS function.

- **Restore Schema**, which restores the IRD schema to its state as of the last time a Deactivate IRDS function was issued. The IRDS must be deactivated, and the IRD-user who issues this function must be the same IRD-user who issued the Deactivate IRDS function.

6.3 IRD SCHEMA OUTPUT

This function produces generalized output on the contents of the schema.

The user must select the meta-entities to be displayed. These meta-entities may be selected by specifying one of the following:

- That all meta-entities are to be displayed.
- That all meta-entities of one or more specific meta-entity types are to be displayed.
- That only the meta-entities whose names are specified are to be displayed.

The resulting set of meta-entities may then be sorted based on the standard set of sort parameters. Each parameter may be designated as ascending or descending. The parameters available are:

- Meta-entity-type.
- Meta-entity access-name.
- Non-repeating meta-attribute-types associated with a meta-entity.

The user also must specify the information that should be shown for each meta-entity in the output, along with the sequence in which this information should appear. This information can include:

- Meta-entity access-names.
- The meta-entity-type.

- One or more of a meta-entity's meta-attributes. (There is an option to show all meta-attributes.)

- All, or optionally none, of the meta-relationships in which a meta-entity participates. The user may request that only direct, only indirect, or both direct and indirect meta-relationships be included. (Meta-entities A and Z are directly meta-related if there is a meta-relationship between A and Z; they are indirectly meta-related if A is directly meta-related to B, B is directly meta-related to C, etc., eventually leading to a meta-entity directly meta-related to Z.) The display of all or none of the meta-attributes of the meta-relationship may also be specified.

The user can specify the destination to which the output is to be routed. A system default destination will be used if no destination is specified.

Finally, a character string may be specified to be used as a title for the IRD schema output. This title can be specified to appear on either the first page or every page of the output.

7. THE IRD-IRD INTERFACE

The IRD-IRD Interface is an important feature of the IRDS because it is the only controlled means for moving data from one IRD to another. If an organization has two or more IRDs, each under the control of a Standard IRDS, this facility allows the organization to select and transport some or all of the entities and relationships (along with their attributes) from one IRD to another.

This facility supports the transportability of IRD data, even in the case where the two Standard IRDSs were developed by different vendors and are resident on different hardware systems at different locations. In this latter case, it is assumed that either a communications link exists between the two computer systems or that some other means of physically moving the data (e.g., transport of tapes) is employed. The IRDS Standard does not address how this physical movement takes place. It is assumed that the details of the IRD-IRD Interface data representation are known to both systems.

This chapter describes problems that may arise when an IRD-IRD transfer is attempted, and presents the transfer methodology that overcomes these problems. The IRD from which the data is exported is called the "source IRD," and the IRD into which the data is imported is the "target IRD." The schema of the source IRD is the "source IRD schema," and the schema of the target IRD is the "target IRD schema."

7.1 INTEGRITY CONSIDERATIONS

This section discusses the types of incompatibilities that may exist between the source and target IRD schemas and the associated source and target IRDs.

7.1.1 IRD Schema Incompatibility

Since the IRDS provides facilities that allow an organization to customize an IRD schema, both the source and target schemas may have been customized in a manner that will make them "incompatible." If such differences exist and are not resolved before the data transport, they can

affect the integrity of the target IRD. The following are examples where the source and target schemas may not be compatible:

- The source IRD might contain an entity-type that does not exist in the target schema. In this case, an entity of this type could not be stored in the target IRD.

- Even if the source and target schemas contain the same entity-types, an entity-type in the source might have associated with it an attribute-type that does not exist in the target schema. It would be possible to import a corresponding entity into the target IRD without the particular attribute. However, a loss of information would occur.

- Each of the schemas may contain different rules for the minimum and maximum lengths of assigned access-names for entities of a given type. Consider the following example:

  > The minimum length for an assigned access-name of an ELEMENT entity is 6 characters in the source IRD schema and 8 characters in the target IRD schema.
  >
  > The maximum length for such names is 36 characters in the source IRD schema, and 32 characters in the target IRD schema.
  >
  > When an ELEMENT is extracted from the source IRD, the length of the assigned access-name of this entity may be 7 characters, although this is not legal in the target IRD. The same situation occurs if the length of this name is 34 characters, since the maximum allowable length in the target IRD schema is 32.

- An IRD schema may contain a list of the legal attributes or ranges of attributes for an attribute-type. Problems in the import of data exist if a value is legal in the source but not in the target.

Some differences between two IRD schemas, however, may not be significant. Consider, for example, the audit meta-attributes and meta-attribute-groups, such as DATE-TIME-

ADDED, associated with the same meta-entity in two different IRD schemas. They will in general be distinct, but that does not make any difference to the importing of data, since the target values will be used.

Thus, there is "IRD schema compatibility" between the source and target IRD schemas when:

1. Every CONTROLLED meta-entity (see section 8.2) in the source IRD schema, except for NAMES, IRDS-RESERVED-NAMES, IRDS-LIMITS, and IRD-PARTITIONS, has the same assigned-access-name as a meta-entity in the target IRD schema.

2. The integrity rules (in the source IRD schema) for the data in the source IRD are compatible with the integrity rules (in the target IRD schema) for the data in the target IRD.

7.1.2 IRD Incompatibility

The above discussion concerned incompatibilities between source and target IRD schemas. Incompatibilities regarding content differences between the source and target IRDs can also exist. The following examples illustrate such situations:

- The descriptors that implement IRDS-USER identifications, the view entities, and the assignment of the views to IRDS users reside in the IRD, rather than the IRD schema. When the data from the source is brought into the target IRD, the IRD-IRD Interface facility must be able to establish such access control in the target IRD.

- The revision-numbers of entities with the same assigned access-name in the source and target IRDs may not be the same, since more modifications may have taken place in one IRD than the other.

- Importing the source IRD audit attributes and attribute-groups such as DATE-TIME-ADDED and ADDED-BY is not meaningful, because the date/time of creation and the responsible users will be different for the target IRD. As discussed in section 7.2, the IRD-IRD Interface facility will reset these attributes to

reflect the import date and the IRDS user who performed the importing.

7.2 THE INTERFACE PROCEDURE

The IRD-IRD Interface facility can be used, together with some user actions, to correct incompatibilities between the source and target IRD schemas and dictionaries. The following steps are required to export data from a source IRD, and to then import it into a target IRD:

1. The IRDS user specifies the subset to be exported by designating an entity-list.

2. The subset is exported from the source IRD using the **export IRD** function. The source IRD schema is also extracted, because it will be necessary to check its compatibility with the target IRD schema. At this point the IRD subset exists in IRD Export format, and the IRD schema in IRD Schema Export format. (The IRDS Specifications define the sequence of entities, relationships, etc., in an exported IRD subset, and the sequence of meta-entities, meta-relationships, etc., in an exported IRD schema.) The exported IRD subset and IRD schema are each in a physical format specified by encoding rules based on Abstract Syntax One (ASN.1), ISO 8824 [12], and ASN.1 Basic Encoding Rules, ISO 8825 [13]. These rules are currently being developed.

This format guarantees that the data can be carried or transmitted from the computer system on which the source IRD resides to the computer system on which the target IRD exists. The data being exported is not intended to be available for user processing while it is in this export format, because: (a) the required security and integrity constraints that control IRD access could not be enforced; and (b) the desired audit trail for such processing would not be available.

The IRDS will not allow an IRD to be imported unless the source and target IRD schemas are compatible. If the source is incompatible with the target (as determined by the **check IRD schema compatibility** function), the following intermediate processing (Steps 3, 4, and 5) will be required to achieve compatibility. If the source and target IRD schemas are compatible, Steps 3, 4, and 5 are not necessary.

3. The IRD Export subset is loaded into an "empty" IRD. At this time an IRD schema corresponding to the subset is also created. There are two options for designating the IRD schema to be used:

   - A file name containing an IRD schema in IRD Schema Export format.

   - The Minimal Schema.

An "empty" IRD is not empty in the literal sense of the word. Rather, it contains entities dealing with the IRDS Control Facilities (see Chapter 8) that will have to be in effect for this new IRD.

The audit attributes are initialized to the date and time of creation of the "empty" IRD and corresponding IRD schema.

Step 3 can be performed at the site where the source IRD is located, or at the target site, or even at a completely different location. The availability of a Standard IRDS is required at the selected location.

4. The source (or target) IRD subset, and its schema, must be modified so that the two IRD schemas are compatible. The IRDS contains an **IRD schema comparison** function. If the comparison indicates a lack of compatibility, the IRDS provides the user with an analysis showing the reasons for the incompatibility. A user must then make changes to one or both of the IRD schemas using IRD schema maintenance facilities. This may involve changing entity names, attribute lengths, etc.

5. Once IRD schema compatibility is achieved, the contents of the IRD subset and its schema are again exported in Export format.

6. This new source IRD subset and schema may now be imported into the target IRD by the importing IRDS. The IRDS requires that a life-cycle-phase be designated in the target IRD for the subset to be imported. No entities can exist in this phase in the target IRD at the time of import. If the target IRD is not "empty," the IRDS will examine the revision-numbers of the entities in the source IRD and will increase them so that they are greater than the revision-numbers of any entities in the target IRD that have the same assigned access-names and variation-names.

During the import, the target IRDS checks the assigned access-name and variation-name for potential conflict with system-generated access-names in the target IRD. The method for checking is the same as that discussed for the add entity function. If a potential conflict exists, the entity is written to an error file, for subsequent resolution by the user.

After all entities, and their associated attributes, of the import set have been imported, the relationships and any associated attributes in the import set are loaded into the target IRD.

8. IRDS CONTROL FACILITIES

The Core IRDS contains four facilities that are important in populating and maintaining the IRD and in reporting on the contents of the IRD. These are: (1) Versioning; (2) Life-Cycle-Phases; (3) Quality-Indicators; and (4) Views. An overview of these facilities appears in Chapter 3. This chapter presents more detail on their structure and use.

8.1 VERSIONING

A version-identifier is part of the access-name and descriptive-name of both entities and meta-entities. Every entity and meta-entity has a version-identifier (by default, if not explicitly specified) but the use of this facility is optional.

8.1.1 Versions of Entity Names

A complete version-identifier is composed of two parts--a variation-name and a revision-number. The existence of a variation-name is optional, i.e., only those entities that have been explicitly assigned variation-names have them. All entities have revision-numbers, since a default mechanism allows their specification to be optional. To specify a complete version-identifier using the Command Language syntax, the user encloses the version-identifier in parentheses and appends it to the assigned access-name and the assigned descriptive-name. Within these parentheses the variation-name (if used) is followed by the revision-number, separated by a colon.

A revision-number of "1" represents the "0th" revision (i.e., the initial entity before the first revision). If the user does not specify a revision-number when creating a new entity, the revision-number defaults to 1. This default mechanism operates for all subsequent revisions (i.e., if a user does not specify a valid new revision-number, the revision-number default is one greater than the highest revision-number associated with the assigned access-name and the variation-name). The following example illustrates this facility:

Suppose, for example, a certain statistical module exists that produces results accurate to 5 decimal places, and a similar statistical module provides results accurate to 8 places. We can describe both with the assigned access-name Stat-Module, and differentiate the two with different variation-names. Thus, we would have Stat-Module(Precision-5) and Stat-Module(Precision-8). The sixth revision of the statistical module with 5 digit precision would be represented as Stat-Module (Precision-5:7). The statistical module with 8 digit precision and no revisions would be represented as Stat-Module(Precision-8:1).

All entity access-names, including those with the same assigned access-name and different variation-names or revision-numbers, represent distinct entities. In addition, the version-identifier associated with an access-name of an entity must be identical to the version-identifier in the descriptive-name of that entity. Thus, if there is an entity with the access-name SSN(4), and this entity has the assigned descriptive-name Social-Security-Number, then the full descriptive-name of the entity automatically becomes Social-Security-Number(4).

8.1.2 Versions of Meta-Entity Names

Meta-entity-access-names and meta-entity-descriptive-names have version-identifiers that are constructed the same as those for access-names and descriptive-names in the IRD. However, in these IRD schema names the variation-name portion of the version-identifier is always null.

The rules governing IRD schema name versioning are analogous to those for IRD names.

8.2 LIFE-CYCLE-PHASES

The Life-Cycle-Phase facility in the Core IRDS:

- Allows an organization to define, for entities in the IRD, life-cycle-phases that correspond to the methodology used by the organization.
- Provides facilities to assign each IRDS entity to one of the defined phases.

- Provides facilities to assign IRD schema meta-entities to one of three built-in IRD schema life-cycle-phases.

- Enforces integrity rules controlling the movement of meta-entities from one IRD schema life-cycle-phase to another.

8.2.1 IRD Life-Cycle-Phases in the Core

Each IRD life-cycle-phase is represented as a meta-entity in the IRD schema. Thus, the specific phases required by an organization can be created using IRD schema manipulation, as discussed in Chapter 6.

As will be described more fully in Section 8.4, an IRDS user always operates in a "IRD-view," and each IRD-view is associated with a IRD-partition. Hence, when an entity is added to the IRD, we can say that the entity is "in" the life-cycle-phase corresponding to the partition associated with the IRD-view in which the user is working.

Every IRD-partition belongs to a "life-cycle-phase class," and the Core Standard IRDS recognizes three such classes:

1. **UNCONTROLLED** -- UNCONTROLLED phases generally represent "non-operational" stages of a system life cycle, such as "specification," "design," or "development." The Minimal Schema contains the UNCONTROLLED IRD-partition UNCONTROLLED-LIFE-CYCLE-PHASE for this purpose. (The Minimal Schema also contains the UNCONTROLLED IRD-partition SECURITY, used for IRDS-USER entities. The SECURITY partition is not considered a life-cycle-phase.)

2. **CONTROLLED** -- CONTROLLED phases are designed to be used for entities in the IRD that describe data existing in "operational" systems. The Minimal Schema contains the CONTROLLED IRD-partition CONTROLLED-LIFE-CYCLE-PHASE.

3. **ARCHIVED** -- ARCHIVED phases are used to document and classify entities no longer in use. The Minimal Schema contains the ARCHIVED IRD-partition ARCHIVED-LIFE-CYCLE-PHASE.

IRD-partition meta-entities of any phase class, and hence the life-cycle-phases corresponding to them, can be arbitrarily created and deleted. IRD entities may be moved from one phase to another. A much more powerful IRD change control facility, that allows the imposition of integrity rules on the movement of entities within the life-cycle-phase framework, is contained in Module 4 of the IRDS Specifications, An Extensible Life Cycle Phase Facility [1].

8.2.2 IRD Schema Life-Cycle-Phases in the Core

The Core life-cycle-phase facility for the IRD Schema (i.e., for meta-entities) is similar to that for the IRD, but is more complete in terms of integrity rules and control.

A user modifying the IRD schema always operates in an "IRD-schema-view," and each IRD-schema-view is associated with an IRD schema life-cycle-phase. Thus, meta-entities added to the IRD schema are "in" the IRD schema life-cycle-phase associated with the IRD-schema-view in which the user is working.

The IRD Schema is divided into the three life-cycle-phases UNCONTROLLED, CONTROLLED, and ARCHIVED. Each meta-entity is in one and only one of these phases.

Using the UNCONTROLLED IRD schema life-cycle-phase, users responsible for maintaining the IRD schema can review and study the potential effect of IRD schema modifications before making the changes effective. When a set of modifications to the IRD structure is ready to be made effective, the meta-entities may be moved to the CONTROLLED IRD schema life-cycle-phase. Finally, by moving superseded meta-entities to the ARCHIVED IRD schema life-cycle-phase, the contents of the ARCHIVED phase would document the prior IRD structure.

The Core Standard IRDS enforces specific integrity rules for meta-entities in either the CONTROLLED or ARCHIVED IRD schema life-cycle-phases. There are no integrity rules for meta-entities in the UNCONTROLLED life-cycle-phase.

For movement of a meta-entity from UNCONTROLLED to CONTROLLED (i.e., the meta-entity is made operational) these integrity rules enforce a set of constraints exemplified by the following:

- If an entity-type is to be moved, all attribute-types associated with the entity-type must have already been moved to CONTROLLED. If a relationship-type is to be moved, both member entity-types must have already been moved. Conversely, at the time the attribute-types or member entity-types were moved, the associated entity-type (or relationship-type, respectively), must still have been UNCONTROLLED.

Similar rules and examples apply to CONTROLLED to UNCONTROLLED, CONTROLLED to ARCHIVED, and ARCHIVED to CONTROLLED moves.

8.2.3 Deactivation and Reactivation of the IRDS

If IRD accesses were to continue while these IRD schema life-cycle-phase modifications were taking place, the integrity of the IRD could be compromised and IRDS functions might not operate properly. Thus, all accesses to the IRD must be suspended while meta-entities are moved between IRD schema life-cycle-phases. Functions are provided which support the deactivation and reactivation of the IRDS, along with a function that restores an original IRD schema.

8.3 QUALITY-INDICATORS

The quality-indicator facility in the Core IRDS allows an organization to define quality-indicators and assign them to entities. These quality-indicators denote such things as: (1) the level of standardization of element entities (e.g., program standard, agency or organization standard, national standard, or international standard); or (2) the degree to which the entity satisfies the organization's quality assurance or quality testing methodology.

Each quality-indicator is a meta-entity in the IRD schema. The Minimal Schema and Basic Functional Schema do not include any indicators, so an organization will have to explicitly define a set of quality-indicator meta-entities to make use of this facility. Although indicators are not attributes, they are handled similarly when adding, modifying, or reporting on entities.

These quality-indicators are available for documentation and search purposes, but no integrity rules are applied. As

discussed in Chapter 11, a future Module could specify additional functions for the use of these indicators.

8.4 VIEWS

The Core IRDS provides for both IRD-views and IRD-schema-views. A user perceives these as windows or gateways into life-cycle-phases, and hence as logical subsets of the IRD and IRD Schema, respectively.

8.4.1 IRD-Views

An IRD-view is specified as:

- A set of entities of specified types, with the entities' attributes and attribute-groups. All the entities in the IRD-view are in the same IRD partition (i.e., either in SECURITY or in an IRD life-cycle-phase.)

- A set of relationships of specified types, with the relationships' attributes and attribute-groups, that exist between the entities in the IRD-view.

Thus, a view defines an environment in which a user works with an IRD. A view can be shared by many users. A user may also have access to many views.

8.4.2 IRD-Schema-Views

An IRD-schema-view is specified as:

- A set of meta-entities, with the meta-attributes and meta-attribute-groups associated with the meta-entities. All the meta-entities are in the same IRD schema life-cycle-phase.

- A set of meta-relationships (with the meta-attributes and meta-attribute-groups associated with the meta-relationships), that exist between the meta-entities in the IRD-schema-view.

- Analogous to IRD-views, an IRD-schema-view can be shared by many users, and a user may have access to many IRD-schema-views.

8.4.3 Defining Views

Structurally, IRD-VIEW and IRD-SCHEMA-VIEW are each entity-types in the IRDS Minimal Schema. We emphasize the distinction between the entity-type and the IRD or IRD schema subset: "IRD-VIEW" (or "IRD-SCHEMA-VIEW") is used for the former and "IRD-view" (or "IRD-schema-view") for the latter. An IRD-VIEW or IRD-SCHEMA-VIEW entity is connected to IRDS-USER entities to allow these users to access the view. When a user is assigned more than one IRD-view or IRD-schema-view, one IRD-view and one IRD-schema-view will be designated as the "default IRD-view" and the "default IRD-schema-view," respectively.

8.4.4 Accessing the IRDS Through a View

When a user accesses the IRD and the IRD Schema, the default IRD-view and the default IRD-schema-view will be presented to the user unless the user specifically indicates that the default view is not to be used. For IRD and IRD schema output, one or more existing views (to which the user has access) can be requested by the user. The output instructions will operate on the union of the entities contained in multiple views.

9. MISCELLANEOUS TOPICS IN THE CORE

The IRDS Specifications contain several utility functions that allow users to display the session status, set defaults, obtain help from the system, exit the IRDS, and switch between IRDS interfaces.

9.1 IRDS SESSION DEFAULTS AND INFORMATION

Default IRD-views and IRD-schema-views exist for each IRDS user, represented as attributes of the relevant IRDS-USER-HAS-IRD-VIEW and IRDS-USER-HAS-IRD-SCHEMA-VIEW relationships.

The code/decode option specifies whether codes or decoded text will appear in IRD output. The normal default, decoded, specifies that decoded text will appear as the values of attribute-types. The code option specifies that codes will appear instead. Codes are always used for input. The following example illustrates this option:

> Suppose that an organization uses an IRDS to help design and document an international travel or transportation system. Names of airports might be important in this application and the organization might want to document the allowable code values in the IRD. Thus, some possible values of the LOCATION attribute-type might be LHR and CDG. The respective decoded values would be London Heathrow for LHR and Charles de Gaulle for CDG. The organization then could use the decode option to prepare reports for managers and users who are not familiar with the codes used in the IRD.

9.1.1 Displaying the Session Status

A user can display the current status of the IRDS, including:

- The defaults in effect.
- The name of the IRD currently in use.

- The IRD-views and IRD-schema-views to which the user has access.

Other implementor-defined session information may also be displayed.

9.1.2 Setting the Session Defaults

A user can set and change the following defaults in effect for the current IRDS session:

- IRD-view and IRD-schema-view
- Code/decode.

The IRDS implementor may define additional defaults that can be set and changed using this function.

The user may "save" the session defaults. If saved, these defaults will be in effect for that user for subsequent sessions until the defaults are reset and saved again.

9.2 HELP

The IRDS contains a Help facility that enables a user to obtain assistance during an interactive session. This facility allows a user to obtain help on any IRDS function or on the most recent IRDS error or warning message. While it is likely that several levels of help will be available to the IRDS user, the precise nature of the facility will be determined by the implementor. The user may specify a function name, error condition, or warning condition for which help is desired, and the system will provide appropriate explanatory information.

9.3 EXITING THE IRDS

The exit function allows the user to leave the IRDS. The following occurs upon execution:

- Where appropriate, session statistics are accumulated, logged, and displayed.
- A message indicating completion of the IRDS session is provided.

- The user is logged off the IRDS.

9.4 ENTERING OTHER INTERFACES

IRDS implementations containing more than one user interface (e.g., a Command Language and a Panel Interface) have facilities for switching from one interface to another. Thus, a person using the Command Language can switch to the Panel Interface, and a person using the Panel Interface can invoke the "command option" to gain access to the Command Language.

10. USER INTERFACES

This chapter discusses the two IRDS user interfaces: the Command Language Interface and the Panel Interface. An IRDS implementation may include either interface, or both. Both interfaces provide the full capabilities of the IRDS.

10.1 THE COMMAND LANGUAGE

The Command Language supports user interaction with the IRDS in both batch and interactive modes. The collection of commands corresponds closely with the collection of "functions" discussed elsewhere in this publication. The Command Language is explained and illustrated in Using the Information Resource Dictionary System Command Language (Second Edition) [8].

10.2 THE PANEL INTERFACE

The Panel Interface provides the IRDS user with a set of logical screens (or panels). The Panel Interface may be considered "user-friendly," in that it leads the user through the appropriate panels to accomplish the desired function. The specified traversal paths are equivalent to the execution of IRDS commands and all their associated clauses.

Although the IRDS Specifications for the Panel Interface assume that a screen-oriented display exists, they cannot specify the physical characteristics of either the device or the screen. Thus, a panel is defined as a "logical screen," and it is the implementor's responsibility to map each panel tree (as defined below) into one or more panels, and to map each of these panels into one or more physical screens on the device or devices that the implementor supports.

10.2.1 Structure of the Panel Interface

Each distinct panel has a name, unique among the set of all panel names, that may be used to reference the panel. A function also exists that allows an organization to rename the panels to customize them to the particular environment in which the IRDS is installed.

The Panel Interface has an inherent "inter-panel structure," that defines a default progression of panels displayed to the user when performing certain IRDS functions. This default progression may always be over-ridden by a user by transferring control to another named panel in the Panel Interface, as long as this does not affect IRD integrity.

Conceptually, the **Home Panel** is the topmost panel of the interface, and it is the panel from which the user is able to traverse the entire inter-panel structure.

10.2.2 Panel Trees and Panel Areas

The structure of the Panel Interface is defined in terms of panel trees and panel areas.

A **panel tree** is defined as a collection of one or more panels used to represent the semantics of a single function in the IRDS. There is a one-to-one correspondence between the set of panel trees and the set of IRDS commands. Each panel tree has a "root" node, which is the logical beginning point from which the remainder of the tree may be traversed.

A **panel area** is a portion of a panel that is always identified with a particular category of information, and that deals with a particular aspect of user interaction with the IRDS. A panel area may, for example, be implemented as a permanent window within a panel in a fixed physical area, or it may be displayed using a special key or action code. At least six specific panel areas exist in the Panel Interface:

- **State Area** -- The State Area informs the user about the name of the IRD being accessed, what is being done with the current panel (e.g., Adding a RECORD; Deleting an ELEMENT; Creating an entity-list), or what the IRDS may be doing (e.g., Updating the IRD; Retrieving information). The State Area also displays the system defaults in effect (e.g., default view;).

- **Data Area** -- The Data Area supports the user in one of two ways: It displays labels that guide the user during data entry, showing the placement of the information to be input; and if the user is retrieving information, it displays the results of the requested output function.

- **IRD Schema Area** -- The IRD Schema Area is primarily used during IRD updating operations. The IRD schema contains information that controls the actions a user can take. The IRD Schema Area displays the available options or the limitations in effect. One way in which this panel area can be used is to flag labels displayed in the Data Area for which relevant IRD schema information exists. Thus, the user can request this IRD schema information by selecting a flagged item. The appropriate IRD schema information would then be displayed in the IRD Schema Area. For example, when a user is adding an entity, the IRD Schema Area might successively display:

  - The list of all valid entity-types.

  - The naming rules for entities of the selected type.

  - The names of attribute-types that may be associated with entities of the selected type.

  - The allowable values or ranges for attributes being entered.

- **Action Area** -- The IRDS displays in the Action Area the options that exist to move from the current panel to another panel. This panel area also contains the COMMIT function, by which the user instructs the IRDS that the specified IRD updates or retrievals are to be performed.

- **Message Area** -- The IRDS displays in this panel area any error and warning messages.

- **Help Area** -- Information that the system can provide in response to a request for "Help" is displayed in the Help Area. Although the responsibility for the precise design and wording resides with the implementor, Help information will include:

  - a general overview of the purpose and operation of each panel.

  - information about the actions that will take place upon selecting any of the operations in the Action Area of a particular panel.

- information relating to the options available for transferring to other panels.

- specific actions which may be taken to overcome error conditions displayed in the Message Area.

An example of how the panel trees might be mapped to a panel structure is given in Figures 7 through 13. Circles represent collections of panel trees, ovals drawn with solid lines represent panel trees that directly correspond to Command Language commands, and dotted ovals represent auxiliary panel trees used to help specify IRD and IRD schema output.

10.2.3 Special Features

There are two features found in the Panel Interface that provide special capabilities to an IRDS user:

- **Saving a panel.** The panel on which the user is working can be saved. If this is done at system log-off, the last panel worked on (and, in some cases, certain associated panels) will be saved in their current form for retrieval at the beginning of the next IRDS session. An example of this feature might be the case where a user is creating an entity-list. If the user requests this option, the IRDS will save the entity-list panel on which the user was operating and all associated panels related to the creation of the entity-list. During the next session, the user may request the saved panel. At this point, the original panel, and all associated panels, will be restored to their former state.

- **Marking a panel.** At any point during a Panel Interface session, a user may specify that the current panel is to be "marked." This feature allows the user to move to any other panel that displays IRDS information. The marked panel remains intact and available to be referenced directly at any time later in the session. After marking a panel, movement to a panel that modifies the IRD or IRD Schema is not allowed, because such modification could affect the integrity of the contents of the marked panel.

The Panel Interface -- Overall Structure

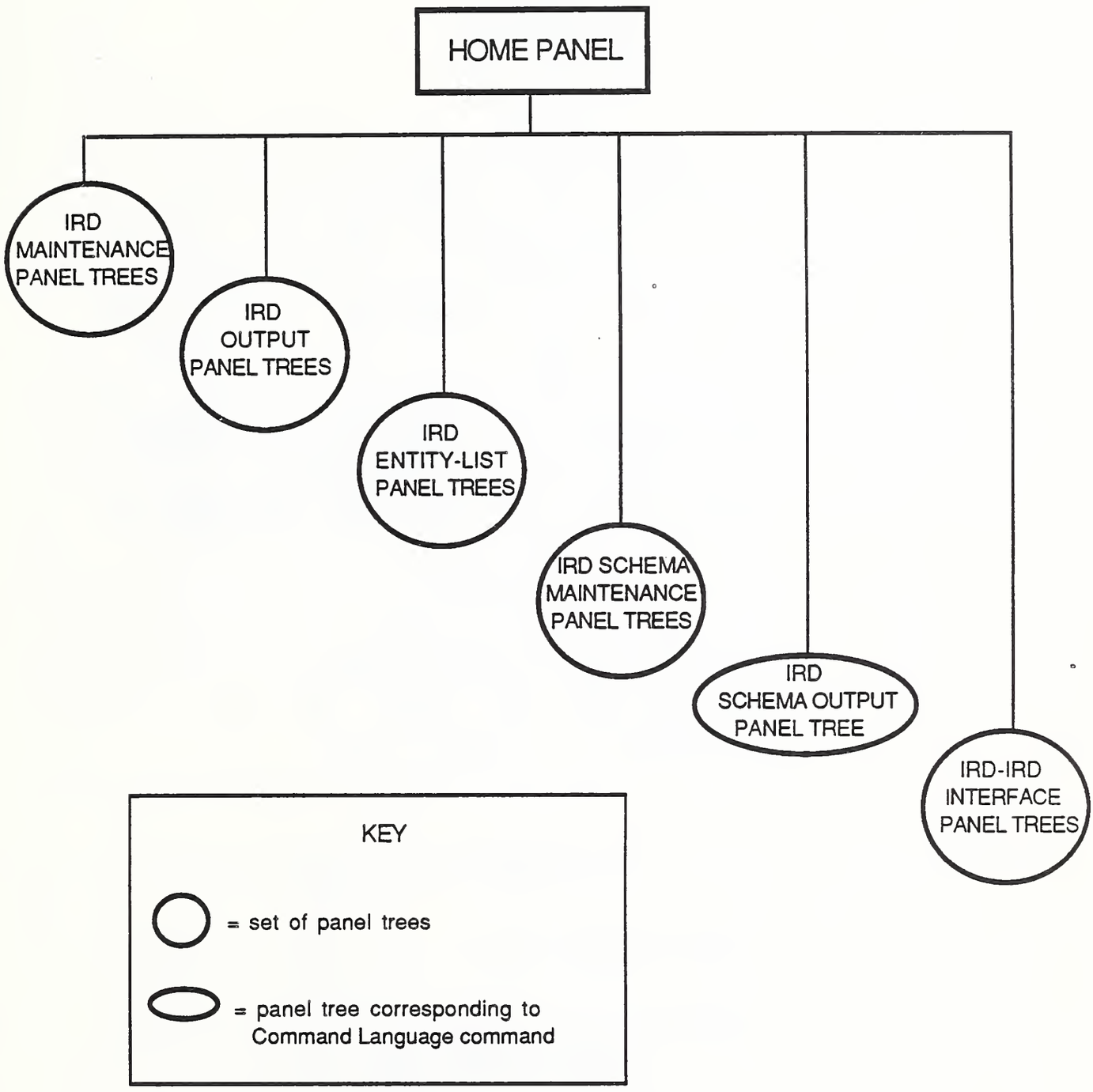


Figure 7

IRD Maintenance Panel Trees

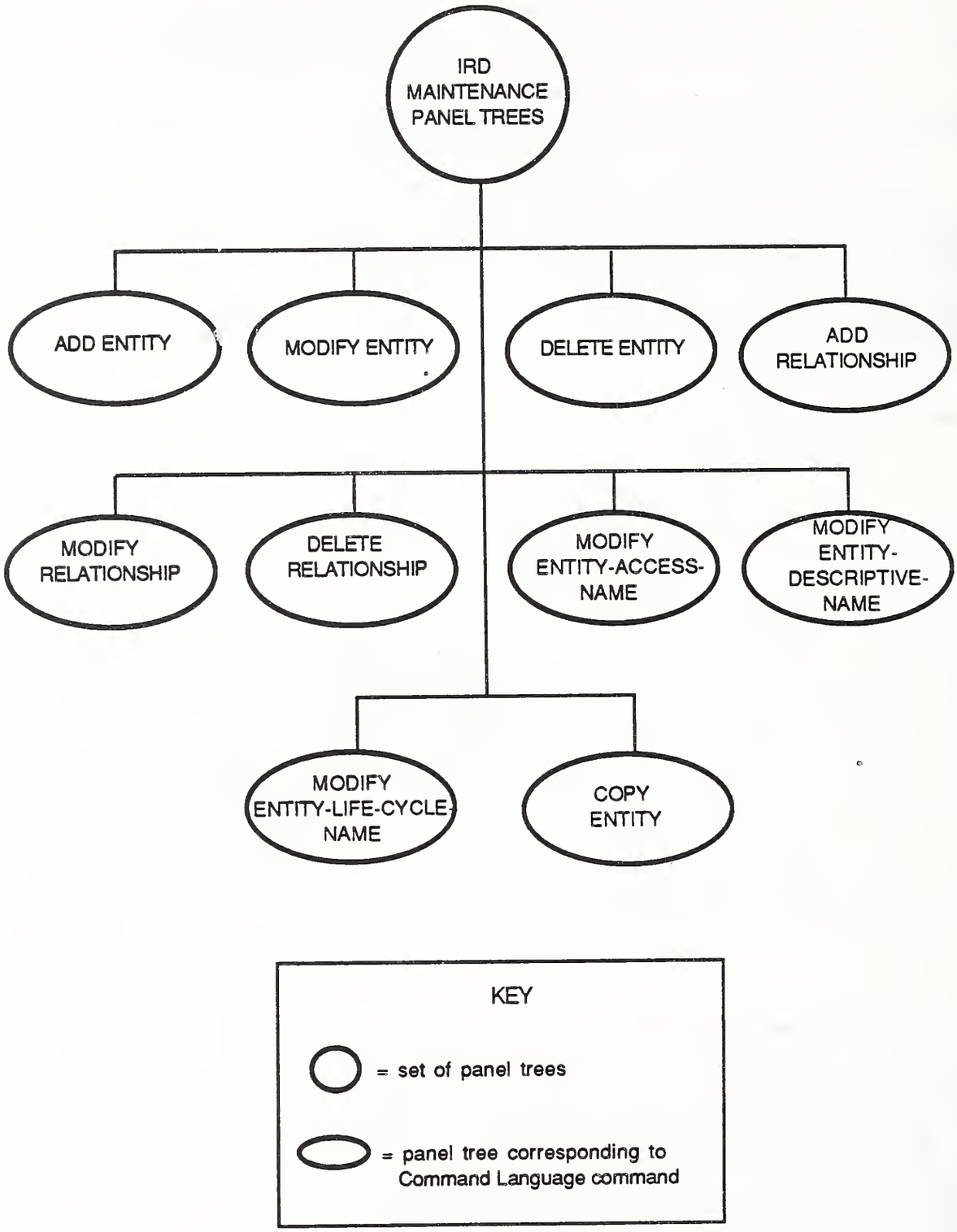


Figure 8

IRD Output Panel Trees

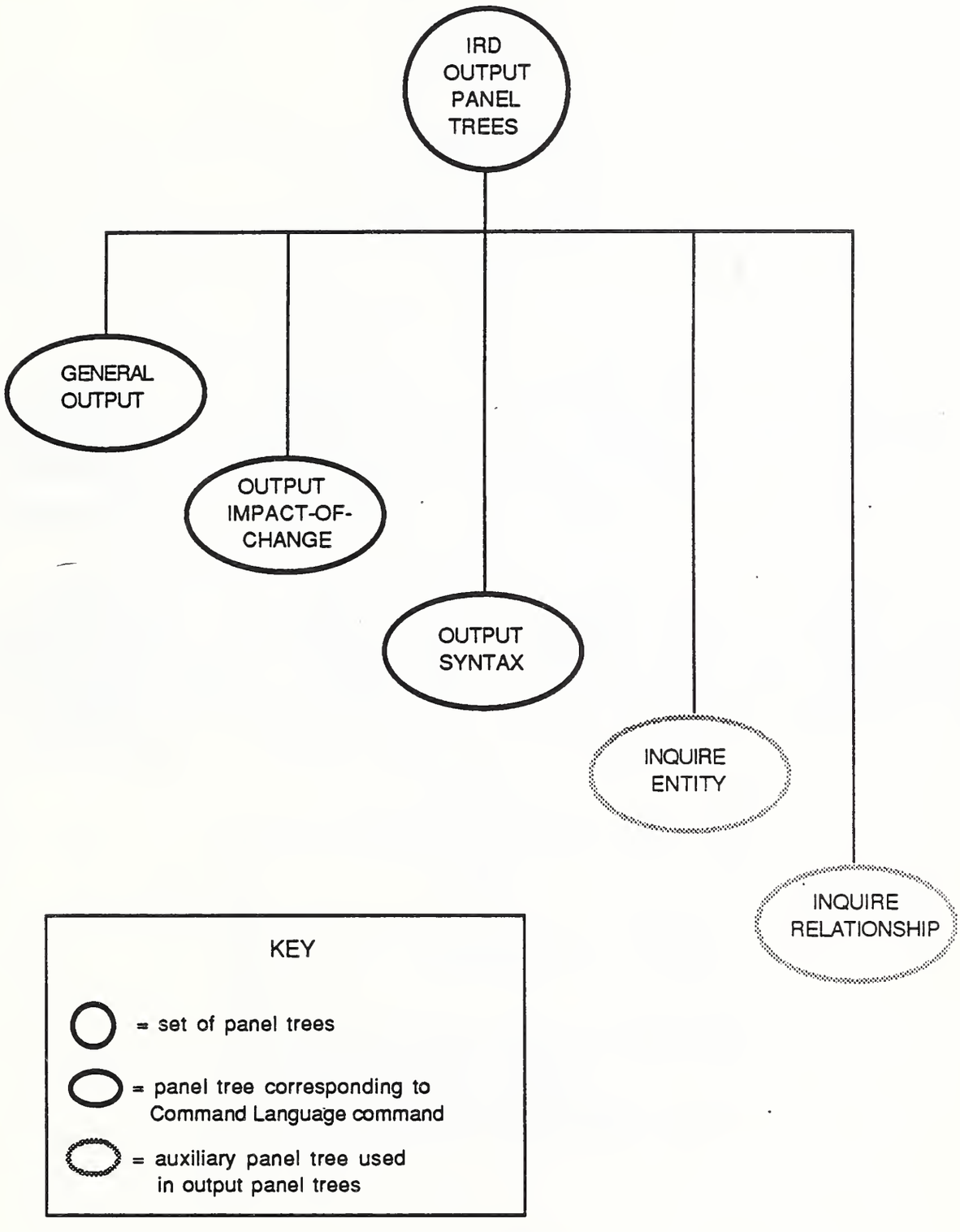


Figure 9

IRD Entity-List Panel Trees

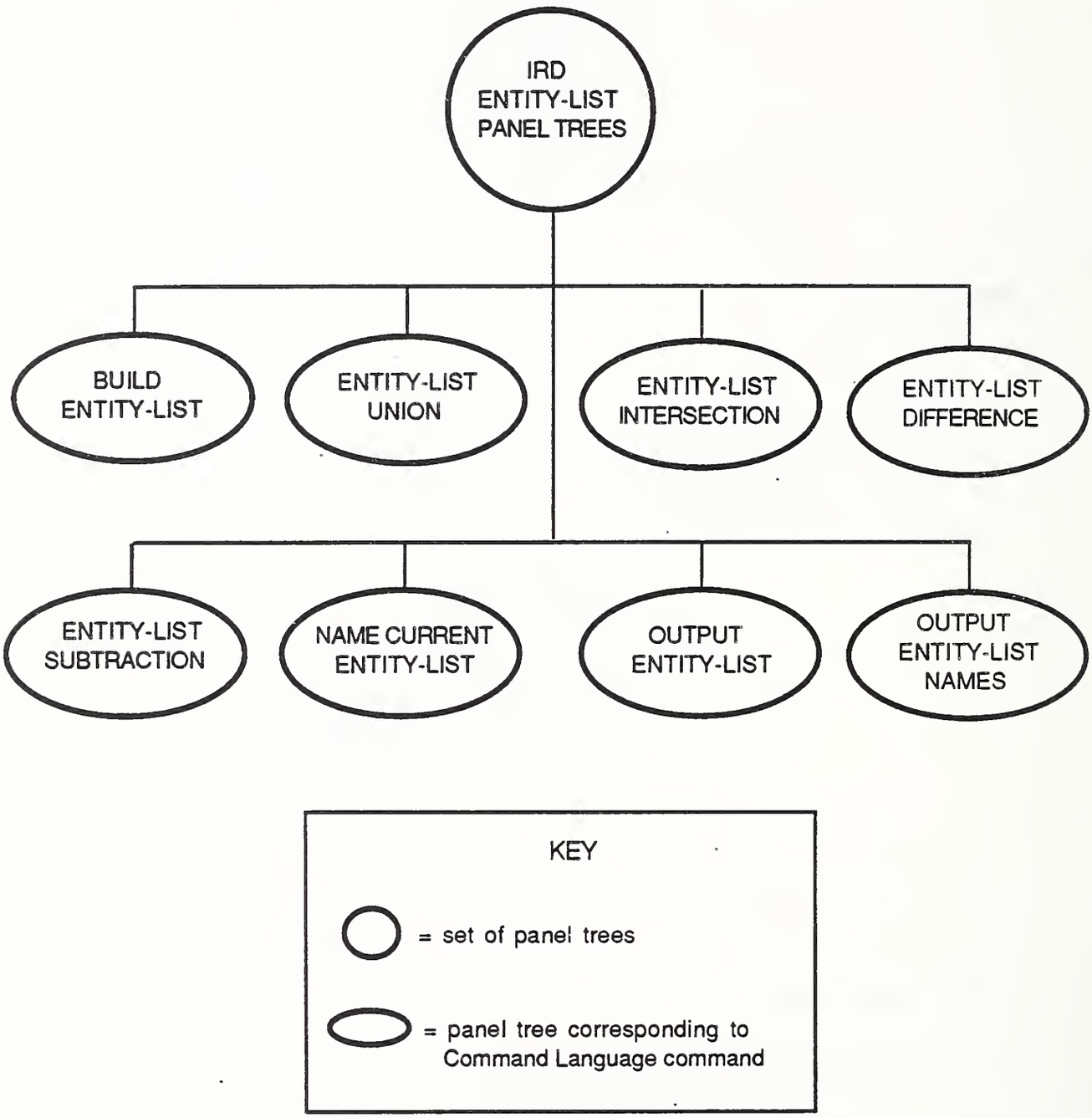


Figure 10

IRD Schema Maintenance Panel Trees

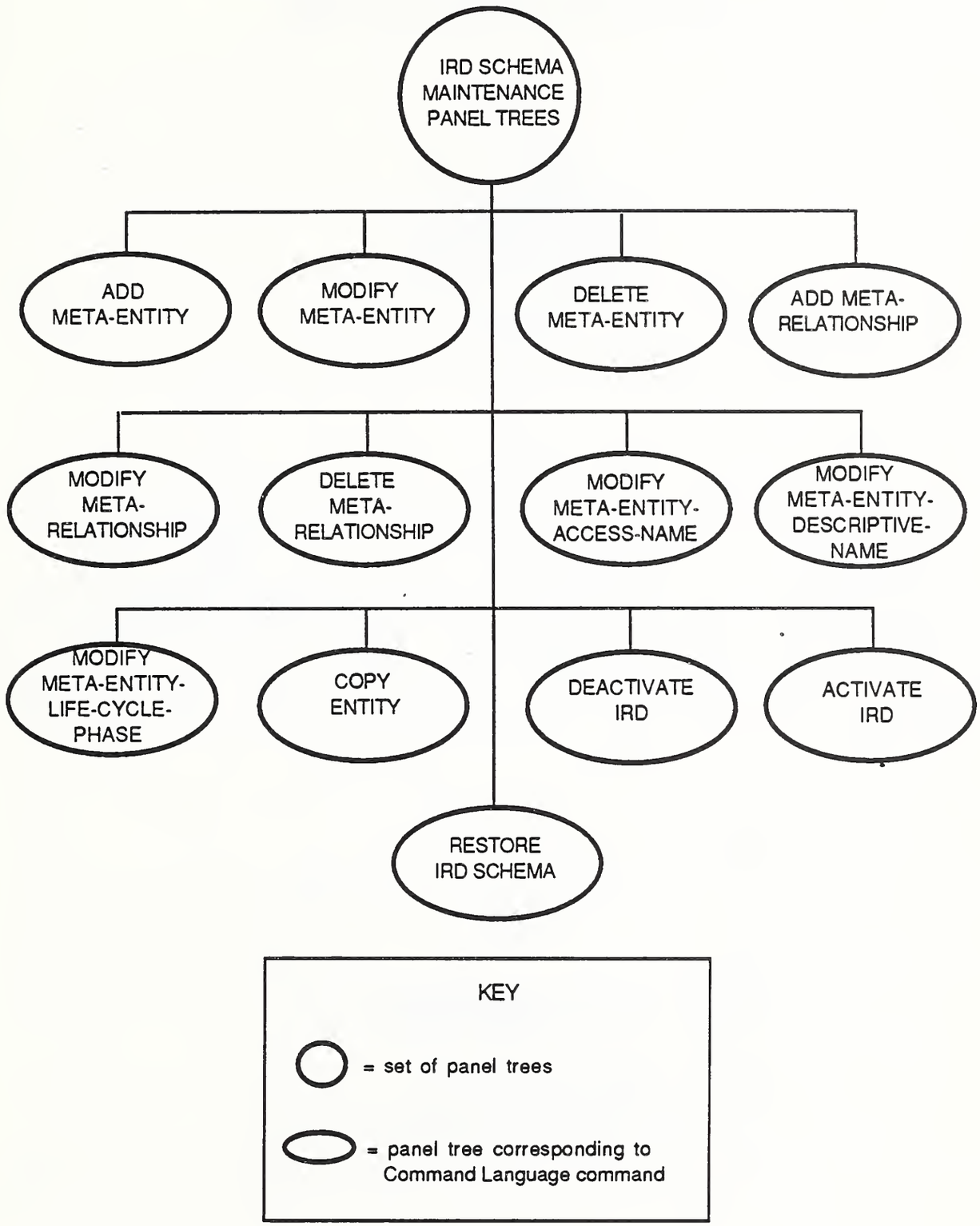


Figure 11

IRD Schema Output Panel Tree

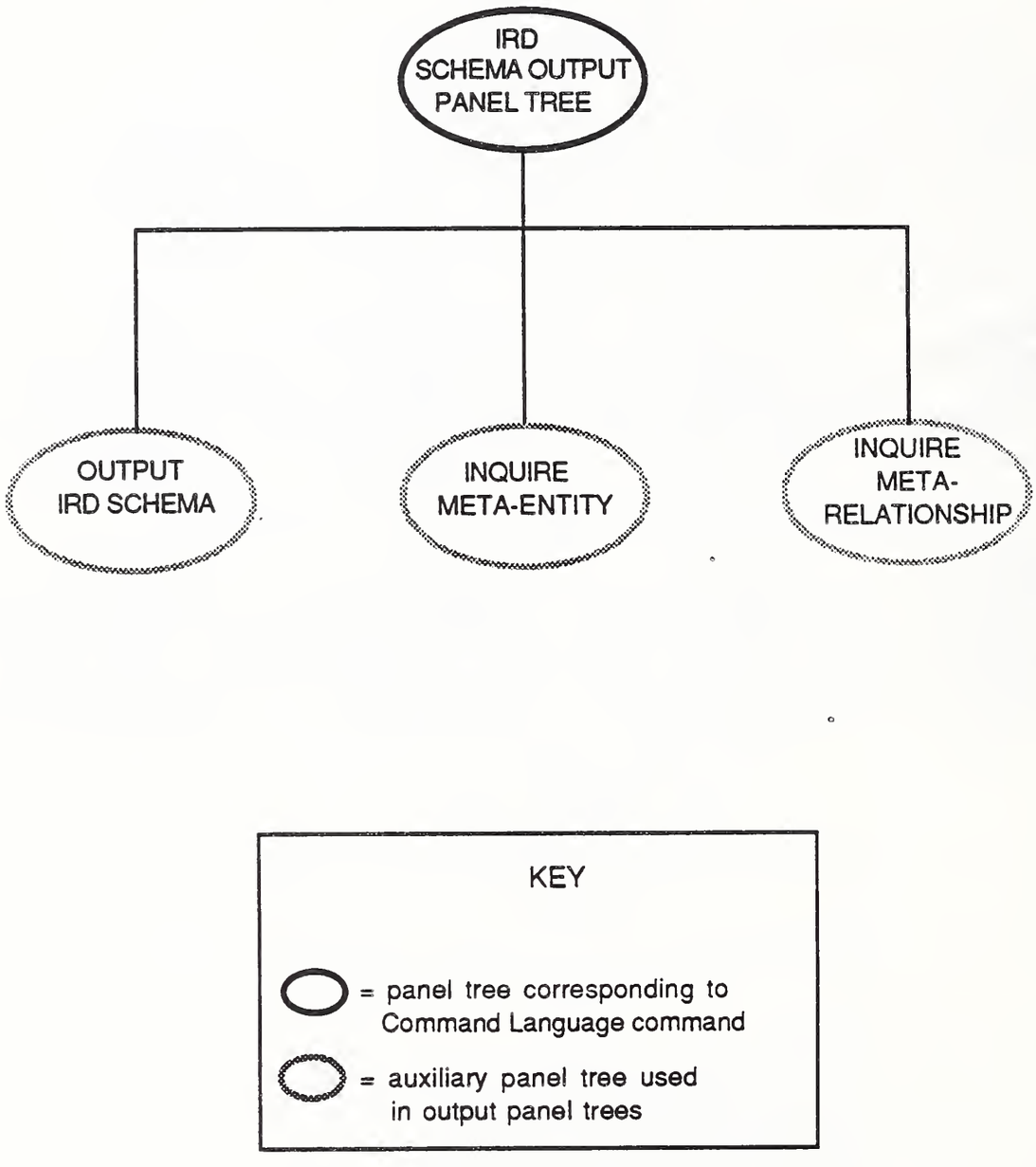


Figure 12

IRD-IRD Interface Panel Trees

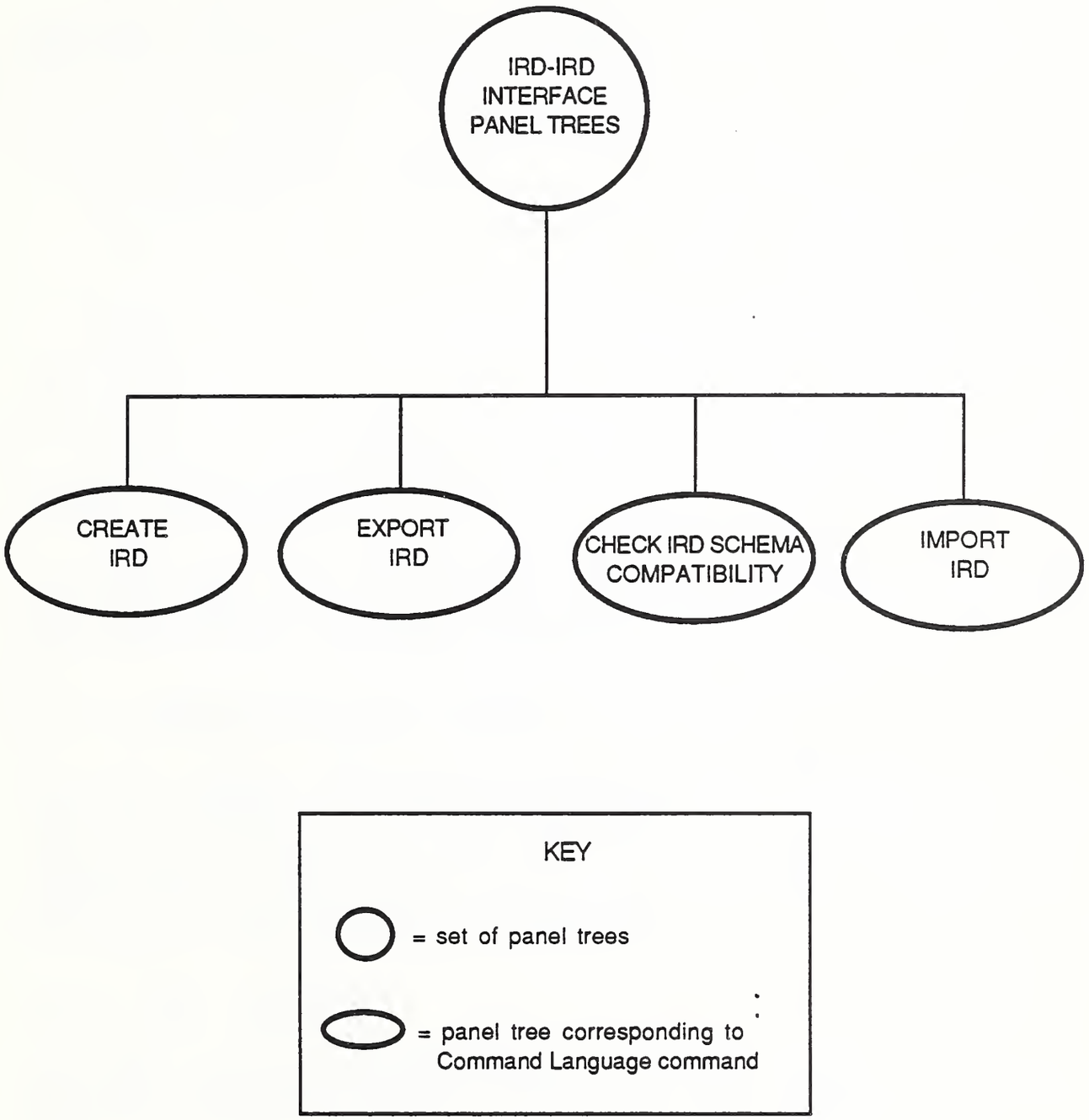


Figure 13

11. IRDS MODULES

The IRDS Standard contains specifications for five Modules, in addition to the Core:

1. Basic Functional Schema (Module 2)
2. IRDS Security (Module 3)
3. Extensible Life-Cycle-Phase Facility (Module 4)
4. Procedure Facility (Module 5)
5. Application Program (Call) Interface (Module 6)

These Modules are discussed in sections 11.1 through 11.5.

Section 11.6 discuses the IRDS Services Interface, a "low-level" external software interface that is currently under development.

Section 11.7 discusses several additional dictionary software features that have been identified as candidates for development as future Modules.

11.1 BASIC FUNCTIONAL SCHEMA

The Basic Functional Schema, the "starter set" of entity-types, relationship-types, attribute-types, attribute-group-types, and other IRD schema descriptors, was discussed in some detail in Section 2.4, and has been continually used as a source of examples. Appendix B contains a more complete discussion.

11.2 IRDS SECURITY

This Module provides facilities that allow an organization to restrict access to IRD and IRD schema content and functionality. There are two levels of access control:

- **Global Security,** with restrictions and permissions based on type and partition. Global Security applies to both the IRD and the IRD schema.

- Entity-Level Security, based on the "locking" of individual entities. Entity-level Security applies only to the IRD.

An operation on the IRD issued by an IRDS user is first checked by the Global Security facility. If the operation passes this check, it is then passed to the Entity-level Security facility for checking. Both levels use the IRD view concept, which is extended and strengthened over the view facility defined in the Core IRDS.

11.2.1 Global Security

The general mechanism that implements Global Security consists of the following:

1. For each authorized user of the IRDS, one IRDS-USER entity exists. Associated with this entity are attributes that define the user's level of access (e.g., permission to use the Command Language Interface, if one exists, and permissible access to the IRD schema).

2. Associated with each IRD-VIEW and IRD-SCHEMA-VIEW entity are attributes and attribute-groups that define the permissions and restrictions that apply to all IRDS users allowed to use the views. These include the abilities (independently specified for each entity-type and meta-entity-type) to read, add to, modify, and delete the entities and meta-entities that comprise the IRD-view and IRD-schema-view, respectively.

3. Finally, each IRDS-USER entity is linked (via IRDS-USER-HAS-IRD-VIEW and IRDS-USER-HAS-IRD-SCHEMA-VIEW relationships) to those IRD-VIEW and IRD-SCHEMA-VIEW entities representing views that the user can access.

11.2.1.1 Access Permissions to the IRD

Most IRD access permissions are associated with IRD-VIEW entities, and, for each corresponding IRD-view, the permissions apply to all entities within the view. Each permission consists of several parts, including:

- The name of the entity-type for which the permissions are specified.

- An indicator showing if permission exists to read entities of the specified type.

- An indicator showing if permission exists to add entities of the specified type. This permission also allows relationships to be added, provided that the permission exists for the entity-types of both entities in the relationship. The ability to copy entities also can be allowed. In addition, this permission enables all entities of the specified type to be accessed and used when building an entity-list.

- An indicator showing if permission exists to delete entities of the specified type. As above, relationships can be deleted if this permission exists for both entities of the relationship. If a relationship spans views, the relationship can be deleted only if the user has permission to delete the entities in both views. Read permissions are also granted as above.

- An indicator showing if permission exists to modify the life-cycle-phase of entities of the specified type. Since a life-cycle-phase modification creates an entity in another life-cycle-phase, the user must have both the phase modification permission and permission to create entities in the view corresponding to the phase into which the entity is being moved.

These permissions are stored in the IRD as an IRD-PERMISSIONS attribute-group. Multiple such permissions can be assigned to any one IRD-view. A separate attribute identifies those relationship-types for which no permissions are granted.

11.2.1.2 Access Permissions to the IRD Schema

In a manner analogous to IRD access permissions, most IRD schema permissions are associated with IRD-SCHEMA-VIEW entities. Each permission consists of several parts, including:

- The name of the meta-entity-type for which permissions are specified.

- An indicator showing if permission exists to read meta-entities of the specified type.

- An indicator showing if permission exists to add meta-entities of the specified type.

- An indicator showing if permission exists to modify meta-entities of the specified type.

- An indicator showing if permission exists to delete meta-entities of the specified type.

These permissions are stored in the IRD as an IRD-SCHEMA-PERMISSION attribute-group. Multiple such permissions can be assigned to any one IRD-schema-view. A separate attribute identifies those meta-relationship-types for which no permissions are granted.

11.2.2 Entity-Level Security

This facility allows an organization to assign read or write privileges for individual entities. Read or write "locks" are associated with each entity to be secured. Users attempting to access a secured entity must have an appropriate read or write "key."

This facility operates as an extension to the Global Security facility. This means, for example, that even though a user working in a IRD-view may have access to all entities of a given type (e.g., all ELEMENTs) in that view, the organization can use Entity-level Security to specify that the user must have additional permission to access certain specific entities (e.g., specific ELEMENTs).

To accomplish Entity-level Security, the facility introduces the new entity-type ACCESS-CONTROLLER, and a set of SECURED-BY relationship-types that allow an ACCESS-CONTROLLER entity to be connected with entities of all other types (except IRDS-USER, IRD-VIEW, IRD-SCHEMA-VIEW, or ACCESS-CONTROLLER). Four new attribute-types are introduced:

- Associated with the ACCESS-CONTROLLER entity-type are the attribute-types:

  READ-LOCK
  WRITE-LOCK.

These locks are 10 digit numbers assigned and controlled by the IRDS.

- Associated with the IRD-VIEW entity-type are the attribute-types:

  READ-KEY
  WRITE-KEY.

  These keys are also 10 digit numbers.

To secure an entity, a relationship is established between that entity and an ACCESS-CONTROLLER entity. This can be done at the time the secured entity is added to the IRD, or at any later time. The read-lock and write-lock on the access-controller serve to protect the entity. The organization can then instruct the IRDS to assign, to selected views, a read or write key matching the respective lock on the access-controller.

Suppose that a user, accessing the IRD through a given IRD-view, attempts to access a protected entity. A write access will cause the IRDS to attempt to find a write-key (a WRITE-KEY attribute of the IRD-VIEW entity) that matches the WRITE-LOCK attribute of the ACCESS-CONTROLLER protecting the entity being accessed. If such a WRITE-KEY attribute is found, the operation will be executed. Otherwise, the entity will be treated as though it did not exist in the IRD-view. A similar sequence of events takes place if a read is attempted. However, since write access also provides read privileges, matches of both read-keys and write-keys are attempted against read-locks and write-locks, respectively.

The precise 10 digit number representing a key or a lock is never visible to an IRDS user.

This mechanism can be illustrated by the following example:

Suppose that it is decided to restrict access to the following entities in the IRD-view Payroll-View:

- A FILE called Payroll-File.
- A RECORD called Payroll-Record.

- An ELEMENT called Employee-Salary.

The organization connects the three entities to the ACCESS-CONTROLLER entity Payroll-Controller, which has a read-lock (IRDS assigned) value of 1234567890, and a write-lock (IRDS assigned) value of 9876543210. Suppose an IRDS user named Jones attempts to access one or more of these entities through the IRD-view named Payroll-View. Jones will be granted access to these entities if the required keys have been assigned to Payroll-View. If only the read-key 1234567890 is available, Jones is able to read these entities; if the write-key 9876543210 is also available, Jones will be allowed to both read and modify any of these entities. Figure 14 illustrates this example.

11.3 EXTENSIBLE LIFE-CYCLE-PHASE FACILITY

This Module provides facilities for flexible, controlled life-cycle management of the contents of the IRD. The Core IRDS, while providing integrity rules and customization features for life-cycle management of the IRD schema, allows only limited, uncontrolled documentation level facilities for the life-cycle management of IRD entities and relationships.

As in the Core, this Module:

- Provides a single controlled life-cycle-phase designed to be used for entities that describe "operational data."
- Provides a single archived phase, used to document and classify entities no longer in use.
- Allows the definition of an arbitrary number of uncontrolled phases that generally represent "non-operational" stages of a system life-cycle.

In addition, this Module allows an organization to:

- Organize all the life-cycle-phases into a hierarchy, within which a given phase could be said to be "greater than" another. The archived phase is greater than the controlled phase, which is greater than all

The Protection of Individual Entities

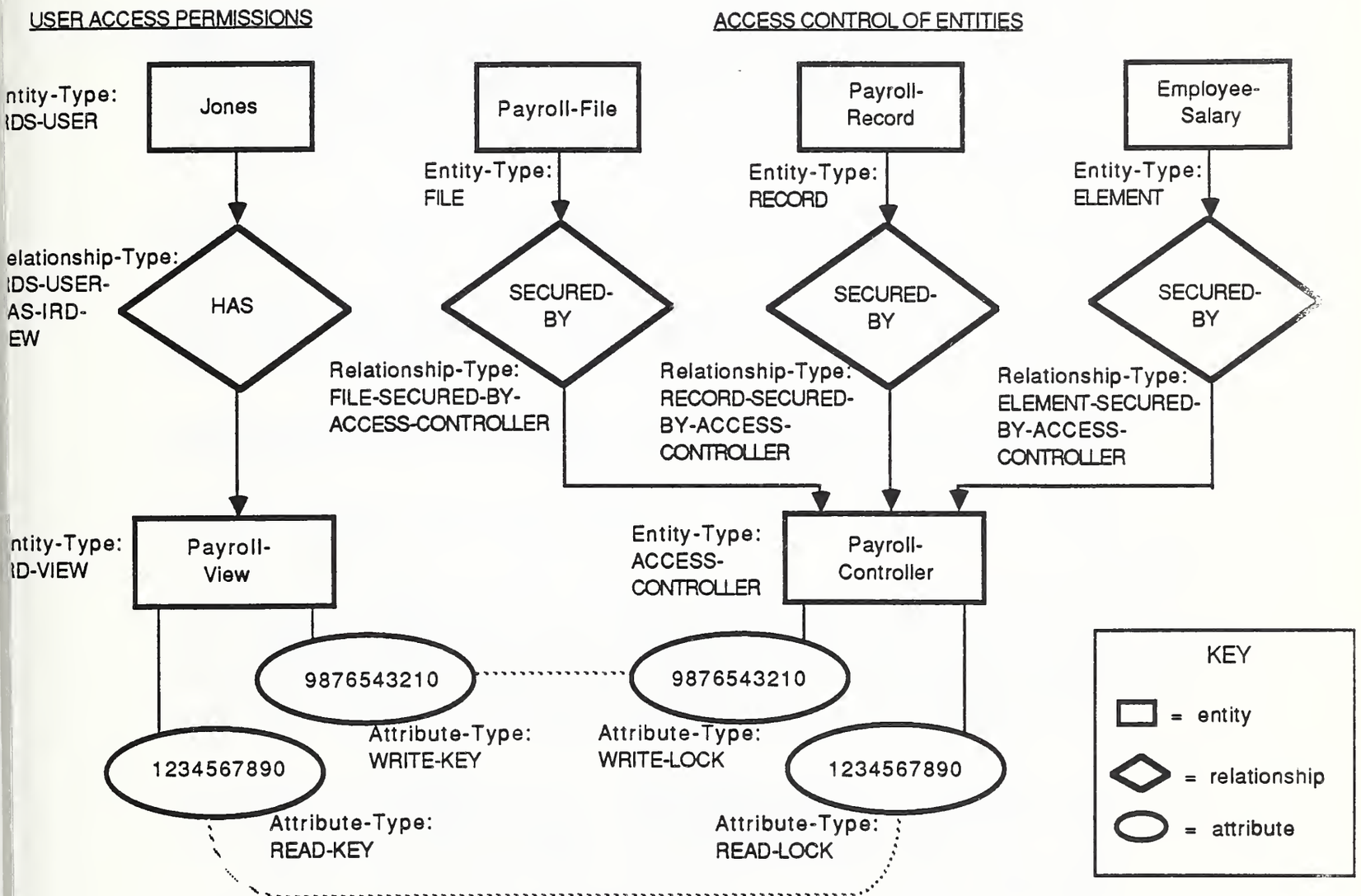


Figure 14

uncontrolled phases. Thus, the "development" phase might be greater than "design," which in turn is greater than "specification."

- Use the formally defined concept of "phase sensitivity structure" to categorize certain relationship-types and their instances as "phase-related."

- Be able to say that an arbitrary entity "depends on" another entity, if the two entities are connected by an appropriate phase-related relationship. Thus, if the relationship FINANCE-DEPARTMENT-RESPONSIBLE-FOR-PAYROLL-SYSTEM exists, and RESPONSIBLE-FOR is phase-related, then FINANCE-DEPARTMENT would "depend on" PAYROLL-SYSTEM.

The basic life-cycle-phase integrity rules can then be stated as:

1. For an entity being moved to the controlled IRD life-cycle-phase, each entity that it depends on must already be in the controlled life-cycle-phase.

2. For an entity being moved to the archived life-cycle-phase, each entity that it depends on must already be in the archived life-cycle-phase.

These integrity rules can best be illustrated by an example:

> Suppose that an entity named Payroll-File exists in the IRD, and that this FILE contains the RECORD Payroll-Record. This association would be represented by a FILE-CONTAINS-RECORD relationship with members Payroll-File and Payroll-Record.
>
> It would be reasonable to expect that if the Payroll-File is operational, Payroll-Record must also describe records in an operational system. Likewise if Payroll-Record contained the field Employee-Salary, one would expect that, for the record to be operational, the entity Employee-Salary (of type ELEMENT) would also have to be operational.
>
> Therefore, if the IRD is constructed so as to assure that FILE-CONTAINS-RECORD and RECORD-CONTAINS-ELEMENT are phase-related, then Payroll-

File will depend on Payroll-Record and Payroll-Record will depend on Employee-Salary. The IRDS will then enforce the appropriate integrity rules.

11.4 IRDS PROCEDURE FACILITY

This Module provides a mechanism for defining and executing procedures on the IRDS. The procedures contain IRDS commands, as well as flow control and assignment statements. They may also contain substitutable parts whose values are set at execution time.

An IRDS procedure may be used for a variety of purposes. It may, for example, be used to store a lengthy (and frequently used) IRDS command for subsequent use. It may be used to store a command, with the ability to alter, at execution time, one or more of the terms of that command. It may be used to simplify entry of repetitive commands, or to hold the set of commands necessary to accomplish a particular task.

It should be noted that the Procedure Module requires the presence of the IRDS Command Language Interface.

11.4.1 Using IRDS Procedures

The Procedure facility defines the ENTITY-TYPE meta-entity IRDS-PROCEDURE. Instances of this entity-type (which are created and maintained using the standard IRD maintenance functions) represent IRDS procedures. Attributes of type DEFINITION, associated with IRDS-PROCEDUREs, are used to store the statements of the procedure.

IRDS procedures are executed by issuing a Run IRDS Procedure statement. Arguments may be passed to the procedure to set run-time options and values.

11.4.2 IRDS Procedure Statements

The following are the allowable types of statements that may be contained within a procedure:

- **IRDS Commands.** These include all the commands specified in the Core IRDS.

- **Assignment Statements**, to assign values to variables.
- **Arg Statements**, to retrieve the values of the arguments passed to the procedures and to specify the names by which the arguments will be identified within the procedure.
- **Do Statements**, to group instructions together and, optionally, to execute them repetitively.
- **If Statements**, to conditionally execute a procedure statement or group of statements.
- **Case Statements**, to conditionally execute one of several alternate statements.
- **Nop Statements**, which have no effect.
- **Return Statements**, to unconditionally terminate execution of a procedure.

These statements are supported by a collection of procedure functions and rules for constructing statements.

11.4.3 Example of an IRDS Procedure

The following example illustrates the passing of parameters to set run-time options. It, and other examples, can be found in Annex 5B of Module 5 of the IRDS Specifications.

Defining the procedure:

```
ADD ENTITY Out-T-Elem
   ENTITY-TYPE = IRDS-PROCEDURE
   DESCRIPTIVE-NAME = Output Test Elements
   WITH ATTRIBUTES
   DESCRIPTION =
   "Procedure to output highest or specified revision
        of elements in Test Phase."
   DEFINITION =
   " VIEW(IRD,Div-101)
     /* Set revision-number to passed value  */
     REVNO = ARG(1);
     /* If no passed value, use highest revision  */
     IF ARG(1,0) THEN REVNO = HIGHEST;
```

```
        OUTPUT IRD SELECT ALL ENTITIES WHERE
              ENTITY-TYPE = ELEMENT
              AND REVISION = REVNO
              AND PHASE = TEST
           SHOW ALL;
      RETURN; " ;
```

Executing the procedure:

```
RUN IRDS-PROCEDURE Out-T-Elem PASSING 5;
```

would be equivalent to executing the IRDS command

```
OUTPUT IRD
  SELECT ALL ENTITIES WHERE
    ENTITY-TYPE = ELEMENT
    AND REVISION = 5
    AND PHASE = TEST
  SHOW ALL;
```

11.5 APPLICATION PROGRAM (CALL) INTERFACE

An implementation of the specifications of this Module will provide an interface from a standard programming language to the IRDS. An organization could develop software in, for example, COBOL, FORTRAN, or PL/I to access the data in the IRD.

The interface is accomplished by using the Call feature of the programming language. The IRDS is thus treated by the application as a subroutine. Parameters are passed through the Call, including:

- A quoted string of characters denoting a syntactically correct IRDS command or sequence of commands.
- A parameter for receiving any IRD or IRD schema output generated by the operation.
- A parameter for receiving any error condition returned by the IRDS. The application must decode the condition to provide the user with a meaningful error message.

This interface enforces all IRDS integrity and security rules. To use this Module, the Command Language Interface must be present.

11.6 IRDS SERVICES INTERFACE

This Module, currently under development, defines a specific protocol for an interface through which any software external to an IRDS can access the IRD and IRD schema. It provides the means to construct an environment in which the IRDS can be truly "active."

The Services Interface uses data structures that are more basic than those used by the Command Language or Panel Interfaces, and is more flexible and potentially much more efficient (if more complicated) than the Application Program Interface. Examples of external software that could make use of the provided services are:

- Programming language compilers directly extracting data definitions.
- SQL [14] and NDL [15] database management systems maintaining database definitions.
- Information locator/retrieval systems.
- Text editors.
- Report writers.
- Open System Interconnection (OSI) systems maintaining directory information.

The design of the Services Interface was influenced by certain objectives and constraints, some of which differed from those of the Application Program Interface. These objectives and constraints include:

- **Complete IRDS Functionality**, as defined by the Core IRDS and the other IRDS Modules. No limits are then placed on the types of applications that the Services Interface can support, and IRDS implementors could conceivably write their Command and Panel interfaces on top of the Services Interface, thus reducing the overall implementation effort.
- **Transactional Processing**, so that when the IRDS is used actively through the Services Interface, the system should provide a recovery mechanism to protect the

integrity of the IRD and IRD schema from system or hardware failure.

- The use of **Normalized Data Structures**, so that entities, relationships, and attributes (and meta-entities, meta-relationships, and meta-attributes) are transferred across the interface one occurrence at a time.

- **Subschema Level Processing**, whereby subschemas of the IRD schema can be defined to correspond to particular host applications, with the Services Interface defining new supporting security entity-types to help provide the host application with a level of isolation from changes to the IRD schema.

- **Native Language Interface**, whereby facilities are specified in terms of external procedures that can be invoked by any language capable of invoking an external subroutine, rather than in terms of a special purpose imbedded language.

- **Set Oriented Processing**, similar to that in SQL, whereby "cursors" are opened and closed, and are used to establish position on a member of a set (of meta-entities, meta-relationships, plural meta-attributes, entities, relationships, and plural attributes).

- **Host-Controlled Error Message Format**, whereby all diagnostic messages generated by activity against the IRDS are placed in a buffer by the interface and are returned to the host application on request.

The specification of an external software interface of this type must include three things:

1. An explicit definition of the data content of the IRDS as seen via the interface.

2. The data definitions for the various parameters used by the interface.

3. The definition of the services provided by the Module and the concrete protocol used in invoking them.

11.6.1 Definition of the IRDS Data Content

The Services Interface, which is a "low-level" interface to the IRDS, needs a more detailed and structured description of the IRDS data content than does the Command Language Interface. For the purpose of the Services Interface, this Module presents the data content definition as a set of SQL tables. The use of SQL DDL for this purpose does not imply that a compliant IRDS must be implemented as an SQL application. Nor does it imply that compliant implementations that are based in SQL must use the specified table definitions. SQL DDL was chosen as the definition language for the IRDS data content because it is the most widely accepted and understood database definition language.

11.6.2 The Interface Data Structures

The Services Interface specifies the data constants and types that can be used to create the various parameters used by the Module. These include:

- **Basic Data Constants**, such as name length limits, entity classes, attribute formats, life-cycle-phases, etc.
- **Basic Data Types**, such as date and time, name types, control identifiers, etc.
- **IRD Schema Object Data Types**, including the key and record types for meta-entities, meta-relationships, and meta-attributes.
- **IRD Object Data Types**, including the key and record types for entities, relationships, and attributes.

11.6.3 The Interface Protocols

The Services Interface specifies a collection of services (i.e., low-level "commands") used to access the IRDS, and includes the concrete protocol for invoking them. These services include:

- **Operational Services**, which are used to control the initiation and termination of processes. The services are Open, Close, Commit, and Rollback.

- **IRD Schema Services**, which are used to add, delete, modify, and retrieve objects in the IRD schema.

- **IRD Services**, which are used to add, delete, modify, and retrieve objects in the IRD.

These services resemble limited, more navigational versions of analogous IRDS Commands. For example, instead of a powerful General Output command, there are separate Retrieve Entity, Retrieve Relationship, and Retrieve Attribute services, each utilizing a collection of required parameters, and each returning simply the "next" object satisfying the given criteria.

The Module's data structures and protocols are presented using ISO Standard Pascal [16]. Pascal was chosen because of its wide recognition and its ability to clearly define data types and procedure parameter types. The Services Interface Module does have appendices containing equivalent data structure definitions and calling sequences in COBOL, PL/I, FORTRAN, C, and Ada.

11.7 POTENTIAL MODULES

Technical Committee X3H4 and participants at workshops sponsored by the Institute for Computer Sciences and Technology of the National Bureau of Standards have identified several modules that would enhance the IRDS capabilities. These potential modules are discussed in the following sections.

11.7.1 Data Management Support Module

This module will provide additional support for: (1) standardization of data elements; and (2) the location of data in an IRD when a user does not know the appropriate access-name or descriptive-name. Support for data element standardization must occur throughout the standardization life-cycle. Once an ELEMENT is identified during the analysis and design phases, facilities will be required to assure that:

- The name associated with the ELEMENT is used consistently throughout the life-cycle. For example, when an ELEMENT is referred to in a programming or database language, only the standard name applicable to that

environment should be used. This requires enforcement of usage based on the ALTERNATE-NAME and ALTERNATE-NAME-CONTEXT attributes of the ELEMENT.

- The usage of the ELEMENT is proper for the given context. That is, just as the proper name must be used, the proper characteristics must also be used. For example, the attribute-types BIT-STRING, CHARACTER-STRING, FIXED-POINT, and FLOAT that are used to describe the characteristics of ELEMENTs must be checked.

- During the operational phase, validation criteria associated with the standard data element and the variety of usage environments for the standard must be controlled and available to the facilities that perform validation. For example, if a user linked a data entry screen to a validation facility, the validation criteria associated with the elements entered on the screen should come from the IRD.

A requirement also exists to verify that any subset of a group of standard data elements uses the appropriate validation criteria. For example, "Countries of the World" would be associated with a table of codes and names representing all of the countries of the world. A subset of these might be "Countries of the Western Hemisphere." In this case, the set of legal codes for the Western Hemisphere should be a subset of the standard codes for countries of the world.

The second major function of this module will increase the support for indexing or classifying entities in an IRD. An attribute-type, CLASSIFICATION, currently appears in the Basic Functional Schema. A series of keywords can appear as values of this attribute-type. Many organizations want to index entities with specific keywords as well as with broader or more generic keywords to help its users locate data when they do not know specific entity names. We expect that this module will specify "thesaurus" software functions or specify an interface to such software to help organizations resolve synonym and homonym problems and develop a "controlled" or consistent keyword vocabulary.

For example, keywords for Finance-Department might be Accounting and Payroll. A user could locate the Finance-Department entity by specifying Accounting or Payroll.

In addition, support for the development and manipulation of data structures to facilitate organizational modeling and logical and physical database design may be provided in this module or in another module.

11.7.2 Support of Distributed Databases and Applications Module

A module for these environments will include extensions to the IRD schema to support network directory functions. These facilities will document what exists in the network, what the dependencies are between processes and data in the network, and where in the network the processes and data reside. This module also must support or help support all traffic management within the network.

Other special features of this module might include:

- Mappings between database structures and mappings among database languages when the network allows processing across heterogeneous database systems.
- Scheduling information with regard to query and application processing.

11.7.3 Life-Cycle and Configuration Management Support Module

Specifications for this module might include:

- Integration of the Life-Cycle-Phase and Quality-Indicator facilities. Such integration would allow metrics to be associated with IRD entities based on their life-cycle-phase and quality-indicator values. This combination could then be used to determine the "suitability" of moving entities to another phase.
- A facility to:
  - establish and manage configurations (i.e., treating assemblages of processes and data as a structure).
  - establish baselines associated with life-cycle-phases, and rules to control movement across these baselines, in both directions.

11.7.4 N-ary Relationship Module

The Core IRDS is specified in terms of binary relationships. That is, the description of the IRD schema, the IRD, and all associated functions are specified in terms of relationships and meta-relationships with precisely two members. This provides a model appropriate for use in a wide range of user environments. This model, however, does not readily support certain more complex environments such as those involving control flow, or some aspects of programming and database languages structure semantics.

X3H4 recognized the need for an n-ary module for some users of the IRDS and unanimously adopted the following statements regarding development of the specifications for such a module:

- An n-ary module is not included in this Standard. It is the intent of X3H4 to continue the development of a module with n-ary capability (n>2) for a future release of the Standard.

- X3H4 recognizes that situations exist where the lack of a n-ary facility may inhibit transportability and may impact some users.

- It is felt that it would not be in the best interest of the majority of potential users to hold back the release of the Standard until a module with n-ary (n>2) capability is specified.

It is intended that specifications for this module will support n-ary schema extensibility and will include the schema descriptors necessary to describe an n-ary schema.

APPENDIX A: THE MINIMAL SCHEMA

Appendix A describes the IRDS Minimal Schema and its structural characteristics. The Minimal Schema consists of those meta-entities, meta-relationships, meta-attributes, meta-attribute-groups, and other IRD schema descriptors necessary to establish controls over the IRD Schema and the IRD. The Minimal Schema is included in every implementation of the IRDS.

A.1 ATTRIBUTE-TYPES AND ENTITY-TYPES

This section presents the attribute-types and attribute-group-types associated with each entity-type. The following are the entity-types in the Minimal Schema:

- IRDS-USER (DUSER)
- IRD-VIEW (DVIEW)
- IRD-SCHEMA-VIEW (SVIEW)

The following table displays the attribute-types and attribute-group-types associated with the entity-types identified above. Attribute-group-types can be identified by the existence of their component attribute-types, which are indented and immediately follow the attribute-group-type name. At the intersection of a row and column, S denotes that an entity of the given type can have at most a single attribute of the given type.

(ATTRIBUTE-GROUP-TYPE) and ATTRIBUTE-TYPE	ENTITY-TYPE		
	DUSER	DVIEW	SVIEW
ADDED-BY	S	S	S
(DATE-TIME-ADDED)	S	S	S
SYSTEM-DATE			
SYSTEM-TIME			
(DATE-TIME-LAST-MODIFIED)	S	S	S
SYSTEM-DATE			
SYSTEM-TIME			
IRD-PARTITION-NAME		S	
LAST-MODIFIED-BY	S	S	S
NUMBER-OF-TIMES-MODIFIED	S	S	S
IRD-SCHEMA-PHASE-NAME			S

A.2 RELATIONSHIP-CLASS-TYPES AND RELATIONSHIP-TYPES

The Minimal Schema contains the single relationship-class-type HAS, with inverse-name OF.

The Minimal Schema contains the two relationship-types:

- IRDS-USER-HAS-IRD-VIEW, with inverse-name IRD-VIEW-OF-IRDS-USER, first member entity-type IRDS-USER, and second member entity-type IRD-VIEW.

- IRDS-USER-HAS-IRD-SCHEMA-VIEW, with inverse-name IRD-SCHEMA-VIEW-OF-IRDS-USER, first member entity-type IRDS-USER, and second member entity-type IRD-SCHEMA-VIEW.

A.3 ATTRIBUTE-TYPES AND RELATIONSHIP-TYPES

The relationship-types IRDS-USER-HAS-IRD-VIEW and IRDS-USER-HAS-IRD-SCHEMA-VIEW are associated with the attribute-type DEFAULT-VIEW, which can have, at most, a single attribute per relationship. The Minimal Schema does not contain any attribute-group-types associated with any relationship-type.

A.4 ATTRIBUTE-TYPE-VALIDATION-DATA META-ENTITIES

The Minimal Schema contains one attribute-type-validation-data meta-entity: YES-OR-NO-VALUE, used to validate that an attribute is either YES or NO.

A.5 ATTRIBUTE-TYPE-VALIDATION-PROCEDURE META-ENTITIES

The Minimal Schema contains the following two attribute-type-validation-procedures:

- RANGE-VALIDATION, used to restrict the attributes of a given attribute-type to a specified range of values.

- VALUE-VALIDATION, used to restrict the attributes of a given attribute-type to a predefined set of values.

A.6 IRD-PARTITION META-ENTITIES

The Minimal Schema contains four IRD-partition meta-entities. These are:

- UNCONTROLLED-LIFE-CYCLE-PHASE, the IRD-partition in which entities may be added and modified.
- CONTROLLED-LIFE-CYCLE-PHASE, the IRD-partition into which are placed entities used in an operational environment.
- ARCHIVED-LIFE-CYCLE-PHASE, the IRD-partition into which are placed entities no longer in use.
- SECURITY, the IRD-partition used for those entities of type IRDS-USER.

A.7 IRDS-DEFAULTS META-ENTITIES

There is one IRDS-DEFAULTS meta-entity in the Minimal Schema. This meta-entity, called EXISTING-IRDS-DEFAULTS, is used to establish organizational defaults for assigned name-lengths, for the number of occurrences for entities of a particular type, and for attribute-lengths.

A.8 IRDS-LIMITS META-ENTITIES

There is one IRDS-LIMITS meta-entity in the Minimal Schema. This meta-entity, called EXISTING-IRDS-LIMITS, is used to establish organization defined limits and initial values for name-lengths, dates and times, and values of other meta-attribute-types.

A.9 IRDS-RESERVED-NAMES META-ENTITIES

The Minimal Schema contains one IRDS-RESERVED-NAMES meta-entity, called STANDARD-RESERVED-NAMES. This meta-entity specifies the assigned-access-names of those entities and meta-entities that are required by the IRDS Specifications.

A.10 NAMES META-ENTITIES

The Minimal Schema contains one NAMES meta-entity, called NAMING-RULES. This meta-entity contains a description of the rules for naming entities and the rules for naming meta-entities. It is intended for user documentation.

APPENDIX B: THE BASIC FUNCTIONAL SCHEMA

Appendix B describes the Basic Functional Schema and its structural characteristics. The Basic Functional Schema is the "starter set" of entity-types, relationship-types, attribute-types, attribute-group-types, and other IRD schema descriptors used to support intra- and inter-organizational communication about information resources. While the Basic Functional Schema satisfies the requirements of many IRDS environments, an organization can customize its IRD Schema using IRD schema extensibility, as discussed in previous chapters.

B.1 ATTRIBUTE-TYPES AND ENTITY-TYPES

In this section, the attribute-types and attribute-group-types associated with each entity-type are given. The following are the entity-types in the Basic Functional Schema:

- USER (USR)
- SYSTEM (SYS)
- PROGRAM (PGM)
- MODULE (MDL)
- FILE (FIL)
- DOCUMENT (DOC)
- RECORD (REC)
- ELEMENT (ELE)

The following table presents the attribute-types and attribute-group-types associated with the entity-types listed above. Attribute-group-types can be identified by the existence of their component attribute-types, which are indented and immediately follow the attribute-group-type name. At the intersection of a row and column, the following denote that an entity of the given type:

S Can have at most a single attribute of the given type.

P Can have multiple (plural) attributes of the given type.

Those attribute-types designated with a suffix of "(M.S.)" are defined in the Minimal Schema.

(ATTRIBUTE-GROUP-TYPE) and ATTRIBUTE-TYPE	ENTITY-TYPE							
	USR	SYS	PGM	MDL	FIL	DOC	REC	ELE
ADDED-BY (M.S.)	S	S	S	S	S	S	S	S
(ALLOWABLE-RANGE) LOW-OF-RANGE HIGH-OF-RANGE	.	.	.	.	.	.	.	P
ALLOWABLE-VALUE	.	.	.	.	.	.	.	P
CLASSIFICATION	P	P	P	P	P	P	P	P
CODE-LIST-LOCATION	.	.	.	.	.	.	.	P
COMMENTS	S	S	S	S	S	S	S	S
DATA-CLASS	.	.	.	.	.	.	.	S
(DATE-TIME-ADDED) (M.S.) SYSTEM-DATE SYSTEM-TIME	S	S	S	S	S	S	S	S
(DATE-TIME-LAST-MODIFIED) (M.S.) SYSTEM-DATE SYSTEM-TIME	S	S	S	S	S	S	S	S
DESCRIPTION	S	S	S	S	S	S	S	S
DOCUMENT-CATEGORY	.	.	.	.	.	S	.	.
(DURATION) DURATION-VALUE DURATION-TYPE	.	S	S	S	.	.	.	.

(ATTRIBUTE-GROUP-TYPE) and ATTRIBUTE-TYPE	ENTITY-TYPE							
	USR	SYS	PGM	MDL	FIL	DOC	REC	ELE
EXTERNAL-SECURITY	S	S	S	S	S	S	S	S
(IDENTIFICATION-NAMES) ALTERNATE-NAME ALTERNATE-NAME-CONTEXT	P	P	P	P	P	P	P	P
LAST-MODIFIED-BY (M.S.)	S	S	S	S	S	S	S	S
LENGTH	.	.	.	.	.	.	.	S
LOCATION	P	P	P	P	P	P	.	.
NUMBER-OF-LINES-OF-CODE	.	.	S	S	.	.	.	.
NUMBER-OF-TIMES-MODIFIED (M.S.)	S	S	S	S	S	S	S	S
NUMBER-OF-RECORDS	.	.	.	.	S	.	.	.
PRECISION	.	.	.	.	.	.	.	S
RECORD-CATEGORY	.	.	.	.	.	.	S	.
SCALE	.	.	.	.	.	.	.	S
SYSTEM-CATEGORY	.	S	.	.	.	.	.	.
USAGE	.	.	.	.	.	.	.	P

B.2 RELATIONSHIP-CLASS-TYPES AND RELATIONSHIP-TYPES

This section presents the relationship-class-types and relationship-types in the Basic Functional Schema. The relationship-class-types, where they exist, are provided in parentheses as headers for the relationship-types to which they apply. The inverse-name (which allows the specification of the member entity-types in reverse order) and substitute inverse-name are given for each relationship-class-type. Where no relationship-class-type applies to a particular relationship-type, its inverse-name and substitute inverse-name are given directly.

(RELATIONSHIP-CLASS-TYPE) and RELATIONSHIP-TYPE	SUBSTITUTE-NAME	INVERSE-NAME	SUBSTITUTE-INVERSE-NAME
(CONTAINS)	**CON**	**CONTAINED-IN**	**CON-IN**
SYSTEM-CONTAINS-SYSTEM	SYS-CON-SYS		
SYSTEM-CONTAINS-PROGRAM	SYS-CON-PGM		
SYSTEM-CONTAINS-MODULE	SYS-CON-MDL		
PROGRAM-CONTAINS-PROGRAM	PGM-CON-PGM		
PROGRAM-CONTAINS-MODULE	PGM-CON-MDL		
MODULE-CONTAINS-MODULE	MDL-CON-MDL		
FILE-CONTAINS-FILE	FIL-CON-FIL		
FILE-CONTAINS-DOCUMENT	FIL-CON-DOC		
FILE-CONTAINS-RECORD	FIL-CON-REC		
FILE-CONTAINS-ELEMENT	FIL-CON-ELE		
DOCUMENT-CONTAINS-DOCUMENT	DOC-CON-DOC		
DOCUMENT-CONTAINS-RECORD	DOC-CON-REC		
DOCUMENT-CONTAINS-ELEMENT	DOC-CON-ELE		
RECORD-CONTAINS-RECORD	REC-CON-REC		
RECORD-CONTAINS-ELEMENT	REC-CON-ELE		
ELEMENT-CONTAINS-ELEMENT	ELE-CON-ELE		
(PROCESSES)	**PR**	**PROCESSED-BY**	**PR-BY**
USER-PROCESSES-FILE	USR-PR-FIL		
USER-PROCESSES-DOCUMENT	USR-PR-DOC		
USER-PROCESSES-RECORD	USR-PR-REC		
USER-PROCESSES-ELEMENT	USR-PR-ELE		
SYSTEM-PROCESSES-FILE	SYS-PR-FIL		
SYSTEM-PROCESSES-DOCUMENT	SYS-PR-DOC		
SYSTEM-PROCESSES-RECORD	SYS-PR-REC		
SYSTEM-PROCESSES-ELEMENT	SYS-PR-ELE		
PROGRAM-PROCESSES-FILE	PGM-PR-FIL		
PROGRAM-PROCESSES-DOCUMENT	PGM-PR-DOC		

(RELATIONSHIP-CLASS-TYPE) and RELATIONSHIP-TYPE	SUBSTITUTE-NAME	INVERSE-NAME	SUBSTITUTE-INVERSE-NAME
PROGRAM-PROCESSES-RECORD	PGM-PR-REC		
PROGRAM-PROCESSES-ELEMENT	PGM-PR-ELE		
MODULE-PROCESSES-FILE	MDL-PR-FIL		
MODULE-PROCESSES-DOCUMENT	MDL-PR-DOC		
MODULE-PROCESSES-RECORD	MDL-PR-REC		
MODULE-PROCESSES-ELEMENT	MDL-PR-ELE		
(RESPONSIBLE-FOR)	**R-FOR**	**RESPONSIBILITY-OF**	**R-OF**
USER-RESPONSIBLE-FOR-SYSTEM	USR-R-FOR-SYS		
USER-RESPONSIBLE-FOR-PROGRAM	USR-R-FOR-PGM		
USER-RESPONSIBLE-FOR-MODULE	USR-R-FOR-MDL		
USER-RESPONSIBLE-FOR-FILE	USR-R-FOR-FIL		
USER-RESPONSIBLE-FOR-DOCUMENT	USR-R-FOR-DOC		
USER-RESPONSIBLE-FOR-RECORD	USR-R-FOR-REC		
USER-RESPONSIBLE-FOR-ELEMENT	USR-R-FOR-ELE		
(RUNS)	**RUNS**	**RUN-BY**	**RUN-BY**
USER-RUNS-SYSTEM	USR-RUNS-SYS		
USER-RUNS-PROGRAM	USR-RUNS-PGM		
USER-RUNS-MODULE	USR-RUNS-MDL		
(GOES-TO)	**TO**	**COMES-FROM**	**FR**
SYSTEM-GOES-TO-SYSTEM	SYS-TO-SYS		
PROGRAM-GOES-TO-PROGRAM	PGM-TO-PGM		

(RELATIONSHIP-CLASS-TYPE) and RELATIONSHIP-TYPE	SUBSTITUTE-NAME	INVERSE-NAME	SUBSTITUTE-INVERSE-NAME
MODULE-GOES-TO-MODULE	MDL-TO-MDL		
(DERIVED-FROM)	**D-FR**	**PRODUCES**	**PRD**
DOCUMENT-DERIVED-FROM-FILE	DOC-D-FR-FIL		
DOCUMENT-DERIVED-FROM-DOCUMENT	DOC-D-FR-DOC		
DOCUMENT-DERIVED-FROM-RECORD	DOC-D-FR-REC		
ELEMENT-DERIVED-FROM-FILE	ELE-D-FR-FIL		
ELEMENT-DERIVED-FROM-DOCUMENT	ELE-D-FR-DOC		
ELEMENT-DERIVED-FROM-RECORD	ELE-D-FR-REC		
ELEMENT-DERIVED-FROM-ELEMENT	ELE-D-FR-ELE		
FILE-DERIVED-FROM-DOCUMENT	FIL-D-FR-DOC		
FILE-DERIVED-FROM-FILE	FIL-D-FR-FIL		
RECORD-DERIVED-FROM-DOCUMENT	REC-D-FR-DOC		
RECORD-DERIVED-FROM-FILE	REC-D-FR-FIL		
RECORD-DERIVED-FROM RECORD	REC-D-FR-REC		
(CALLS)	**CLS**	**CALLED-BY**	**CLD-BY**
PROGRAM-CALLS-PROGRAM	PGM-CLS-PGM		
PROGRAM-CALLS-MODULE	PGM-CLS-MDL		
MODULE-CALLS-MODULE	MDL-CLS-MDL		

(RELATIONSHIP-CLASS-TYPE) and RELATIONSHIP-TYPE	SUBSTITUTE-NAME	INVERSE-NAME	SUBSTITUTE-INVERSE-NAME
ELEMENT-STANDARD-FOR-ELEMENT	ELE-ST-FOR-ELE	ELEMENT-STANDARD-OF-ELEMENT	ELE-ST-OF-ELE
FILE-HAS-SORT-KEY-ELEMENT	FIL-H-S-K-ELE	ELEMENT-SORT-KEY-OF-FILE	ELE-S-K-OF-FIL
FILE-HAS-ACCESS-KEY-ELEMENT	FIL-H-A-K-ELE	ELEMENT-ACCESS-KEY-OF-FILE	ELE-A-K-OF-FIL

The last three relationship-types are not members of a relationship-class, and so are listed separately.

B.3 ENTITY-TYPES AND RELATIONSHIP-TYPES

The following table depicts the entity-types participating as members of the relationship-types in the Basic Functional Schema. The following notation is used to denote that the entity-type is:

1 The first member of the relationship-type.

2 The second member of the relationship-type.

R Both the first and second member of the relationship-type (i.e., the relationship is recursive).

(RELATIONSHIP-CLASS-TYPE) and RELATIONSHIP-TYPE	USR	SYS	PGM	MDL	FIL	DOC	REC	ELE
(CONTAINS)								
SYSTEM-CONTAINS-SYSTEM	.	R	.	.	.	.	.	.
SYSTEM-CONTAINS-PROGRAM	.	1	2	.	.	.	.	.
SYSTEM-CONTAINS-MODULE	.	1	.	2	.	.	.	.
PROGRAM-CONTAINS-PROGRAM	.	.	R	.	.	.	.	.
PROGRAM-CONTAINS-MODULE	.	.	1	2	.	.	.	.
MODULE-CONTAINS-MODULE	.	.	.	R	.	.	.	.

(RELATIONSHIP-CLASS-TYPE) and RELATIONSHIP-TYPE	USR	SYS	PGM	MDL	FIL	DOC	REC	ELE
FILE-CONTAINS-FILE	.	.	.	.	R	.	.	.
FILE-CONTAINS-DOCUMENT	.	.	.	.	1	2	.	.
FILE-CONTAINS-RECORD	.	.	.	.	1	.	2	.
FILE-CONTAINS-ELEMENT	.	.	.	.	1	.	.	2
DOCUMENT-CONTAINS-DOCUMENT	.	.	.	.	.	R	.	.
DOCUMENT-CONTAINS-RECORD	.	.	.	.	.	1	2	.
DOCUMENT-CONTAINS-ELEMENT	.	.	.	.	.	1	.	2
RECORD-CONTAINS-RECORD	.	.	.	.	.	.	R	.
RECORD-CONTAINS-ELEMENT	.	.	.	.	.	.	1	2
ELEMENT-CONTAINS-ELEMENT	.	.	.	.	.	.	.	R
(PROCESSES)								
USER-PROCESSES-FILE	1	.	.	.	2	.	.	.
USER-PROCESSES-DOCUMENT	1	.	.	.	.	2	.	.
USER-PROCESSES-RECORD	1	.	.	.	.	.	2	.
USER-PROCESSES-ELEMENT	1	.	.	.	.	.	.	2
SYSTEM-PROCESSES-FILE	.	1	.	.	2	.	.	.
SYSTEM-PROCESSES-DOCUMENT	.	1	.	.	.	2	.	.
SYSTEM-PROCESSES-RECORD	.	1	.	.	.	.	2	.
SYSTEM-PROCESSES-ELEMENT	.	1	.	.	.	.	.	2
PROGRAM-PROCESSES-FILE	.	.	1	.	2	.	.	.
PROGRAM-PROCESSES-DOCUMENT	.	.	1	.	.	2	.	.
PROGRAM-PROCESSES-RECORD	.	.	1	.	.	.	2	.
PROGRAM-PROCESSES-ELEMENT	.	.	1	.	.	.	.	2
MODULE-PROCESSES-FILE	.	.	.	1	2	.	.	.
MODULE-PROCESSES-DOCUMENT	.	.	.	1	.	2	.	.
MODULE-PROCESSES-RECORD	.	.	.	1	.	.	2	.
MODULE-PROCESSES-ELEMENT	.	.	.	1	.	.	.	2

(RELATIONSHIP-CLASS-TYPE) and RELATIONSHIP-TYPE	USR	SYS	PGM	MDL	FIL	DOC	REC	ELE
(RESPONSIBLE-FOR)								
USER-RESPONSIBLE-FOR-SYSTEM	1	2	.	.	.	.	.	.
USER-RESPONSIBLE-FOR-PROGRAM	1	.	2	.	.	.	.	.
USER-RESPONSIBLE-FOR-MODULE	1	.	.	2	.	.	.	.
USER-RESPONSIBLE-FOR-FILE	1	.	.	.	2	.	.	.
USER-RESPONSIBLE-FOR-DOCUMENT	1	.	.	.	.	2	.	.
USER-RESPONSIBLE-FOR-RECORD	1	.	.	.	.	.	2	.
USER-RESPONSIBLE-FOR-ELEMENT	1	.	.	.	.	.	.	2
(RUNS)								
USER-RUNS-SYSTEM	1	2	.	.	.	.	.	.
USER-RUNS-PROGRAM	1	.	2	.	.	.	.	.
USER-RUNS-MODULE	1	.	.	2	.	.	.	.
(GOES-TO)								
SYSTEM-GOES-TO-SYSTEM	.	R	.	.	.	.	.	.
PROGRAM-GOES-TO-PROGRAM	.	.	R	.	.	.	.	.
MODULE-GOES-TO-MODULE	.	.	.	R	.	.	.	.
(DERIVED-FROM)								
FILE-DERIVED-FROM-FILE	.	.	.	.	R	.	.	.
FILE-DERIVED-FROM-DOCUMENT	.	.	.	.	1	2	.	.
DOCUMENT-DERIVED-FROM-FILE	.	.	.	.	2	1	.	.
DOCUMENT-DERIVED-FROM-DOCUMENT	.	.	.	.	.	R	.	.
DOCUMENT-DERIVED-FROM-RECORD	.	.	.	.	.	1	2	.
RECORD-DERIVED-FROM-DOCUMENT	.	.	.	.	.	2	1	.
RECORD-DERIVED-FROM-FILE	.	.	.	.	2	.	1	.
RECORD-DERIVED-FROM-RECORD	.	.	.	.	.	.	R	.

(RELATIONSHIP-CLASS-TYPE) and RELATIONSHIP-TYPE	USR	SYS	PGM	MDL	FIL	DOC	REC	ELE
ELEMENT-DERIVED-FROM-FILE	.	.	.	.	2	.	.	1
ELEMENT-DERIVED-FROM-DOCUMENT	.	.	.	.	.	2	.	1
ELEMENT-DERIVED-FROM-RECORD	.	.	.	.	.	.	2	1
ELEMENT-DERIVED-FROM-ELEMENT	.	.	.	.	.	.	.	R
(CALLS)								
PROGRAM-CALLS-PROGRAM	.	.	R	.	.	.	.	.
PROGRAM-CALLS-MODULE	.	.	1	2	.	.	.	.
MODULE-CALLS-MODULE	.	.	.	R	.	.	.	.
ELEMENT-STANDARD-FOR-ELEMENT	.	.	.	.	.	.	.	R
FILE-HAS-SORT-KEY-ELEMENT	.	.	.	.	1	.	.	2
FILE-HAS-ACCESS-KEY-ELEMENT	.	.	.	.	1	.	.	2

The last three relationship-types are not members of a relationship-class, and so are listed separately.

B.4 ATTRIBUTE-TYPES AND RELATIONSHIP-TYPES

The following are the attribute-types associated with the relationship-class-types and relationship-types in the Basic Functional Schema:

- The relationship-types
  - SYSTEM-PROCESSES-FILE
  - PROGRAM-PROCESSES-FILE
  - MODULE-PROCESSES-FILE

  have the single-valued attribute-type ACCESS-METHOD associated with them.

- All PROCESSES and RUNS relationship-types have the single-valued attribute-type FREQUENCY associated with them.

- The relationship-type RECORD-CONTAINS-ELEMENT has the single-valued attribute-type RELATIVE-POSITION associated with it.

INDEX

REFERENCES

1. ANSI, American National Standard X3-138-1988, Information Resource Dictionary System, American National Standards Institute, New York, 1988.

2. Application Systems Division, Prospectus for Data Dictionary Standard, NBSIR 80-2115, National Bureau of Standards, Gaithersburg, MD, September, 1980.

3. Goldfine, A. H., Editor, Data Base Directions: Information Resource Management--Strategies and Tools, NBS Special Publication 500-92, National Bureau of Standards, Gaithersburg, MD, September, 1982.

4. Fong, E. N. and Goldfine, A. H., Editors, Data Base Directions: Information Resource Management--Making It Work, NBS Special Publication 500-139, National Bureau of Standards, Gaithersburg, MD, June, 1986.

5. Konig, P. A. and Newton, J. J., Federal Requirements for a Federal Information Processing Standard Data Dictionary System, NBSIR 81-2354, National Bureau of Standards, Gaithersburg, MD, September, 1981.

6. Konig, P. A., Goldfine, A. H., and Newton, J. J., Editors, Functional Specifications for a Federal Information Processing Standard Data Dictionary System, NBSIR 82-2619, National Bureau of Standards, Gaithersburg, MD, January, 1983.

7. Chipman, M. L. and Fiorello, M., Cost-Benefit Analysis of a Prospective Data Dictionary System Standard, prepared for the Institute for Computer Sciences and Technology, National Bureau of Standards, Gaithersburg, MD, October, 1983.

8. Goldfine, A. H., Using the Information Resource Dictionary System Command Language (Second Edition), NBSIR 88-3701, National Bureau of Standards, Gaithersburg, MD, 1988.

9. Law, M. H., Guide to Information Resource Dictionary System Applications: General Concepts and Strategic Systems Planning, NBS Special Publication 500-152,

National Bureau of Standards, Gaithersburg, MD, 1988.

10. Dolk, D. R. and Kirsch, R. A., A Relational Information Resource Dictionary System, Communications of the ACM 30, 1, January, 1987, 48-61.

11. Newton, J. J., Guide on Data Entity Naming Conventions, NBS Special Publication 500-149, National Bureau of Standards, Gaithersburg, MD, October, 1987.

12. ISO, Abstract Syntax One (ASN.1), ISO 8824, American National Standards Institute, New York.

13. ISO, ASN.1 Basic Encoding Rules, ISO 8825, American National Standards Institute, New York.

14. ISO, International Standard Database Language SQL, ISO 9075, American National Standards Institute, New York, 1987.

15. ISO, International Standard Database Language NDL, ISO 8907, American National Standards Institute, New York, 1987.

16. ISO, Pascal Computer Programming Language, ISO 7185, American National Standards Institute, New York

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NBSIR 88-3700	2. Performing Organ. Report No.	3. Publication Date JANUARY 1988

4. TITLE AND SUBTITLE

A Technical Overview of the Information Resource Dictionary System (Second Edition)

5. AUTHOR(S)

Alan Goldfine, Patricia Konig

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

7. Contract/Grant No.

8. Type of Report & Period Covered

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

10. SUPPLEMENTARY NOTES

This document supersedes NBSIR 85-3164, A Technical Overview of the Information Resource Dictionary System.

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

This publication provides a technical overview of the computer software specifications for an Information Resource Dictionary System (IRDS). It summarizes the data architecture and the software functions and processes of the IRDS. The IRDS Specifications are an American National Standard, a U.S. Federal Information Processing Standard, and a Draft Proposal within the International Organization for Standardization (ISO). This Overview also provides background information on the development of the IRDS software specifications.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

American National Standard; computer software; data dictionary; data dictionary system; data management; Federal Information Processing Standard; FIPS; Information Resource Dictionary System; IRDS; information resource management; IRM; International Standard.

13. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☐ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

☒ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

142

15. Price

$18.95