Worldwide Response Center

HP 3000 APPLICATION NOTE #62

Setting Up A
System Dictionary

HEWLETT PACKARD

November 15, 1989
Document P/N #5959-7803

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

RESPONSE CENTER APPLICATION NOTES

HP 3000 APPLICATION NOTES *are published by the Worldwide Response Center and are distributed with the Software Status Bulletin. These notes address topics where the volume of calls received at the Center indicates a need for addition to or consolidation of information available through HP support services.*

Following this publication you will find a list of previously published notes and a Reader Comment Sheet. You may use the Reader Comment Sheet to comment on the note, suggest improvements or future topics, or to order single copies of back issues. We encourage you to return this form; we'd like to hear from you.

Setting up a System Dictionary

Introduction

System Dictionary is one of Hewlett-Packard's data dictionary products. It is used to maintain a central and consistent definition of databases, files, VPLUS forms, and other files. This Application Note is intended for a dictionary administrator or programmer maintaining the dictionary. It is a guide in setting up and loading information into the dictionary.

The topics covered in this Application Note are:

1. System Dictionary Concepts
2. Setting up a System Dictionary
3. Unique Features of System Dictionary
4. System Dictionary Uses

For your reference, a list of System Dictionary manuals is provided at the end of this Application Note.

In general, the purpose of data dictionaries is to provide a central location where definition of databases, files, and other objects reside. Simply put, the dictionary contains data about your data. This is referred to as "metadata". It allows you to manage the data resources on your system.

The benefits of a System Dictionary are:

- Well-managed data resources
- More data sharing and less data redundancy
- Increased data integrity
- Standard naming conventions and consistency
- Integration with tools that use the dictionary
- Central documentation

Having a dictionary that documents the information systems environment does not enforce the standards. With the integrated tools that leverage off the dictionary, the standards are naturally followed. However, without the enforcing standards, the definitions may vary from program to program.

System Dictionary uses the Entity-Relationship model. The next section presents terminology and concepts in the Entity-Relationship model.

SYSTEM DICTIONARY CONCEPTS

Entities are the names of real-world objects in an information systems environment. A TurboIMAGE dataset, MPE file, VPLUS form, record layout, an item within a file are all examples of entities. Entities are grouped within a category; the category is known as the Entity-Type. For example, your financial database and manufacturing database belong to a category known as Image-Database; INVOICE-NO and ORDER-NO items within a database belong to a category known as Element.

Relationships refer to the logical connections between real-world objects. For example, ORDER-DETAIL record *contains* the element ORDER-NO. This is a relationship between a RECORD entity-type and an ELEMENT entity-type. Like entities, relationships are grouped within a category known as its Relationship-Type. The example of "ORDER-DETAIL contains ORDER-NO" relationship is a relationship-type of "RECORD contains ELEMENT".

Relationships may logically connect anywhere from 2 to 6 entities. Typically, a relationship is between 2 entities such as with "RECORD contains ELEMENT". On the other hand, there are 6-way entity-relationships. An example of a 6-way entity-relationship is the Image Chain relationship. To describe the relationship between an Image master and detail dataset chain, six entity-types are used:

1. IMAGE-DATASET (detail)
2. ELEMENT (search item in chain)
3. ELEMENT (sort item in the chain)
4. IMAGE-DATASET (master)
5. IMAGE-DATABASE.

The verb connecting the real-world objects in a relationship is known as a Relationship Class. The relationship class "CONTAINS" is commonly referenced in many relationships. However, many other relationship classes exist, such as "CHAINS", "KEY", "USES", "ACCESSES", and so on.

Attributes are characteristics to describe a relationship or entity. For example, the attributes of ORDER-NO are that it is a character field and 6 bytes long. In System Dictionary terms, ELEMENT-TYPE has a value of "X" and BYTE-LENGTH has a value of 6. Each entity-type and relationship type have an appropriate, yet varying set of attributes.

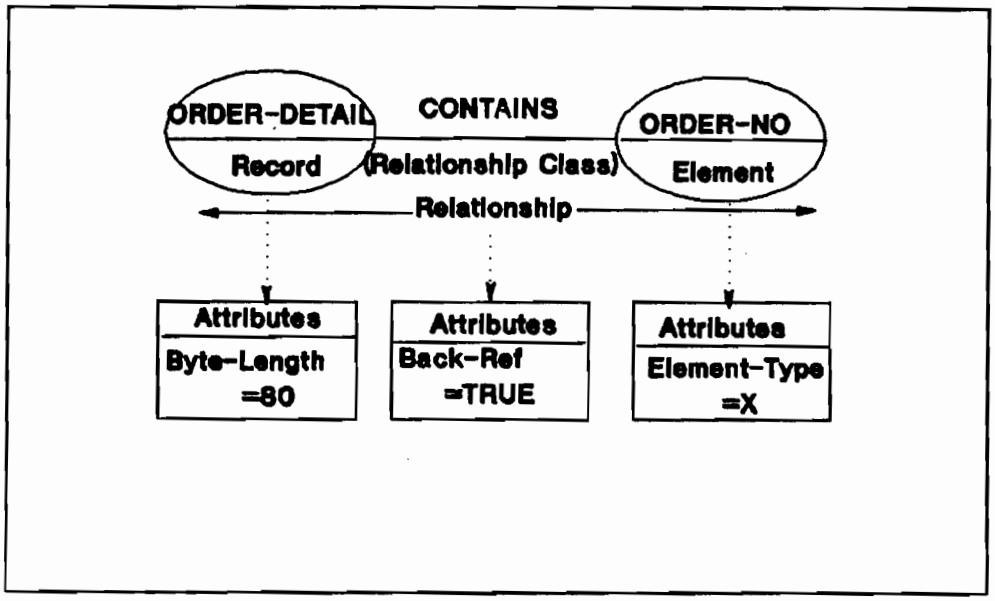

Diagram 1: Entity-Relationship Model

Diagram 1 shows the Entity-Relationship model. For a list of the entity types, relationship types and their associated attributes, refer to the section on "Core Set" in one of the following manuals:

HP System Dictionary/V General Reference Manual, Volume 1
(Part-No. 32254-90004)

HP System Dictionary/XL General Reference Manual, Volume 1
(Part-No. 32256-90004).

The "Core Set" section lists the entity-types, relationship-types and attributes which are "standard" with the product. The ones we have mentioned so far are considered part of the core-set. However, System Dictionary is customizable to your environment. Therefore, user-defined entity-types, relationship-types, classes, attributes may be tailored to fit a particular need. Be aware however, that any tool integrated with the dictionary must understand these user-defined structures in order to use them correctly.

Domains and versions, a unique feature of System Dictionary, allows you to logically partition the dictionary. Domains may be thought of as logically separate dictionaries. Therefore, domains may be used to keep different applications' definitions within its own space; yet have one central dictionary.

By default when System Dictionary is created, one domain called "COMMON" exists. COMMON is a public domain sharable by all. If you do not wish to logically partition the dictionary, you may simply use COMMON domain to hold data definitions. To access the COMMON domain, specify a blank domain name when opening the dictionary.

A version is a further partition of a domain. Versions are used to control changes occurring within a dictionary. Versions contain complete copies of entities and relationships. Each version within a domain may have entities and relationships that the prior or subsequent versions do not contain. In addition, the attributes of the entities and relationships between versions may differ. A version is at one of 3 stages of development: TEST, ARCHIVAL or PRODUCTION. The stage of development is known as the Version Status. For each domain there may be only one production version and multiple test and archival versions. The only version which you may make modifications is TEST. System Dictionary allows you to change versions from one status to another easily.

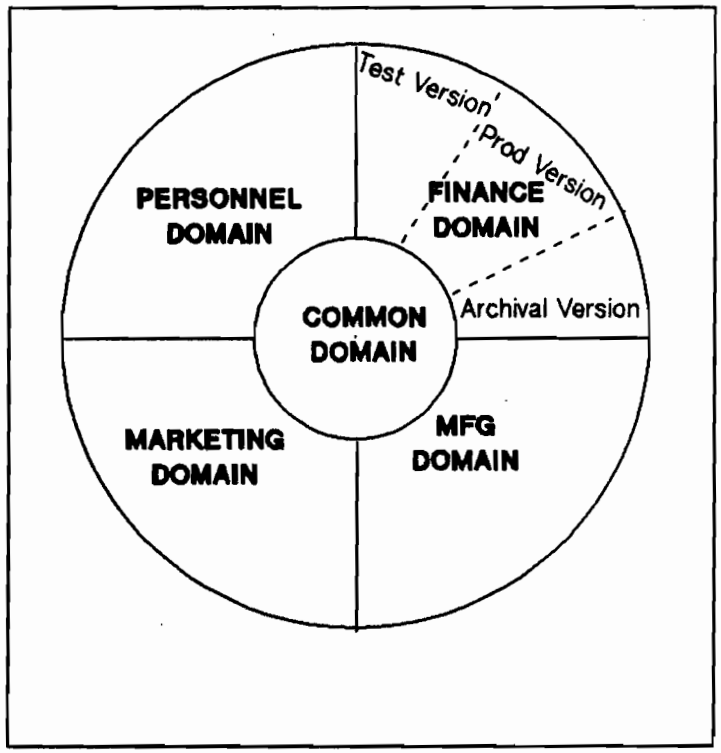

Diagram 2: Domain and Versions

Diagram 2 illustrates the partitioning of a Dictionary with domains and versions.

Another term in System Dictionary is "scope". A scope is a user name in System Dictionary. The scope is assigned a password and given a set of rights or capabilities. When the dictionary is built, one scope is pre-defined with the maximum rights and referred to as the Dictionary Administrator (DA).

Name mode is another term used in reference to opening the dictionary. When objects (domains, versions, scopes, entity-types, relationship-classes, attributes) are created, the object has 2 primary names: an internal name and an external name. At creation time, the object's internal and external names are the same by default. An internal name is not modifiable after creation. However, a rename command may be used to change the external name. When you open the dictionary, you may specify the name mode (internal or external) in which you wish to operate.

SETTING UP A SYSTEM DICTIONARY

Now that the concepts and terminology of the entity-relationship model have been defined, we will define the steps required to create and load a System Dictionary. Various System Dictionary utilities are used to load information into the dictionary, as well as one utility to create the actual dictionary.

1. **Create the system dictionary**

```
:RUN SDINIT.PUB.SYS
```

SDINIT program creates the Image-based System Dictionary. This program creates the dictionary structure and loads core definitions into the dictionary such as Entity-Types, Relationship-Types, and so on.

Prompts for SDINIT include:

- Capacities - these may be the default capacities or user provided capacities.
- Dictionary Administrator's scope name
- Dictionary Administrator's passwords
- Information to stream jobstream creating the System Dictionary.

After the job completes, a SYSDIC database is created in the logon group and account. The System Dictionary also consists of other files which are tables to enhance the retrieval of entities and relationships. The associated files are:

```
SYSDICEA - Table of attribute information by external attribute name

SYSDICET - Table of entity type information

SYSDICED - Table of attribute edits

SYSDICIA - Table of attribute information by internal attribute name

SYSDICPW - Encryption code for passwords

SYSDICRT - Table of relationship type information

SYSDICTI - Table of indexes into SYSDICET and SYSDICRT
```

NOTE

Prior to System Dictionary Version A.01.03, there is a known problem with the version name assigned when System Dictionary is originally created. The version name for the first version created with the common domain is "V1" despite the name assigned during SDINIT. This problem has been corrected in version A.01.03 and later.

2. Load database definitions.

For databases, SDDBD is available to automatically load the database definition into the System Dictionary.

To load a database:

```
:RUN SDDBD.PUB.SYS
```

DEFINE, IMAGE and LOAD commands are used in SDDBD to define the entities and relationships:

```
>DEFINE
  SCOPE = scope-name;
  PASSWORD = scope-password;
  DOMAIN = domain-name;
  VERSION = version-name;
  OPEN-MODE = SU (Shared-Update) or EU (Exclusive-Update).

>IMAGE
  DB = database-name;
  PASSWORD = image-password;
  OPEN-MODE = image-mode;
  BACK-REFERENCE = TRUE or FALSE;
  SENSITIVITY = READ, MODIFY, or PRIVATE;
  COMPATIBILITY-LEVEL = IMAGE-COMPATIBILITY or COMPLETE;
  OPTION = PROMPT, SKIP, REPLACE, NEW, TERMINATE;
  QUIET;
  VERBOSE.
```

The database, password and open mode parameters are required in the IMAGE command. The rest of the parameters are optional.

```
>LOAD.
```

This starts the process of loading database definitions into System Dictionary.

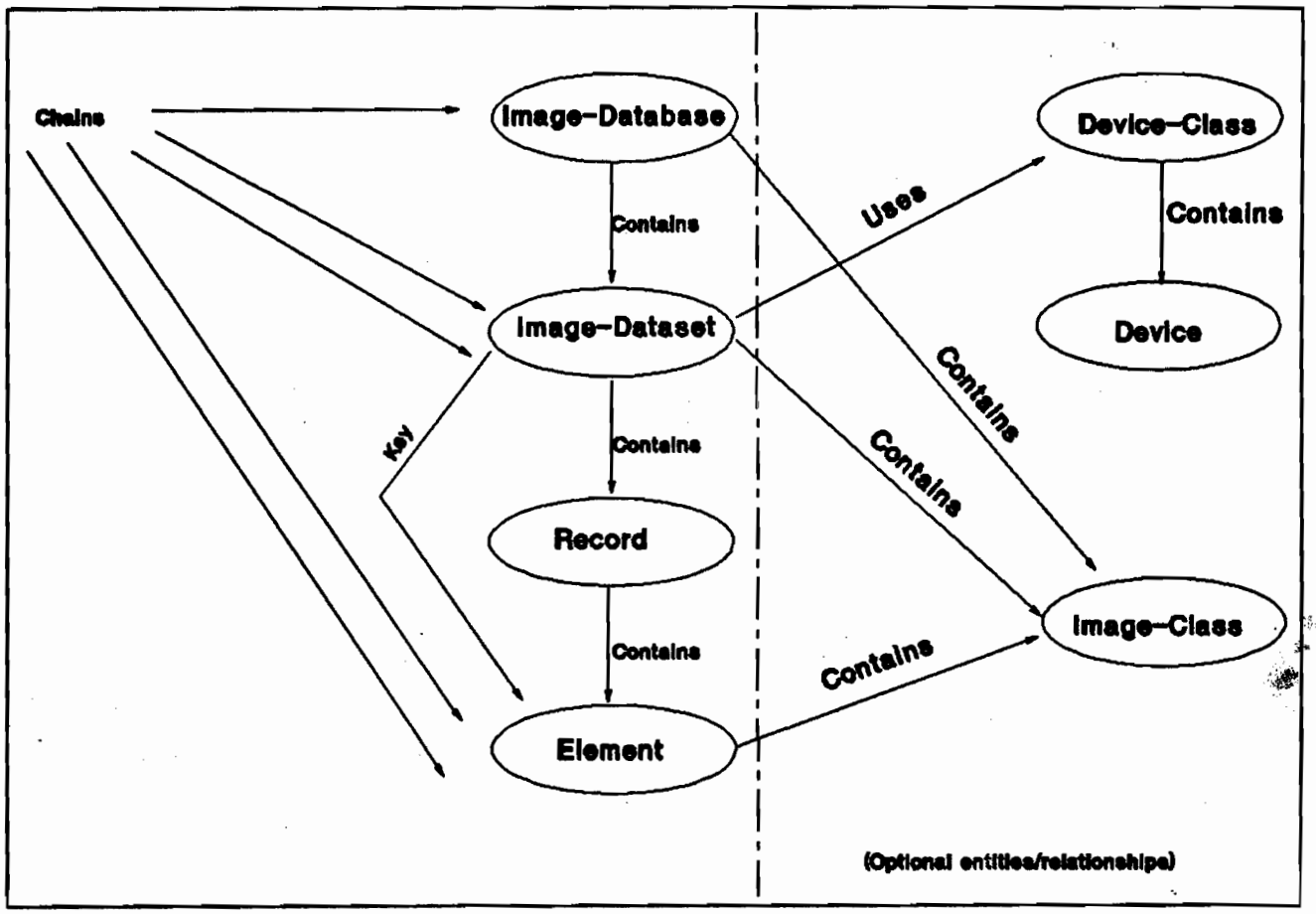

Diagram 3: Image Entity-Relationship Model

The definitions loaded by SDDBD create the entities and relationships as illustrated in Diagram 3. Entities are represented as ovals and relationships as arrows.

3. Load VPLUS formsfile definitions.

Like Image databases, there is a utility to automatically load the formsfile definitions. SDVPD loads VPLUS formsfile definitions into the dictionary.

To load a formsfile definition:

:RUN SDVPD.PUB.SYS

DEFINE, VPLUS and LOAD commands are used in SDVPD to define a VPLUS formsfile entities and relationships.

```
>DEFINE
  SCOPE = scope-name;
  PASSWORD = scope-password;
  DOMAIN = domain-name;
  VERSION = version-name;
  OPEN-MODE = SU (Shared-Update) or EU (Exclusive-Update).
```

```
>VPLUS
  FF = formsfile-name;
  CHAR = OFF, X or 9;
  HYPHEN = ON or OFF;
  CHECK-ELEM-TYPE = ON or OFF;
  COMPATIBILITY-LEVEL = VPLUS-COMPATIBILITY or COMPLETE;
  SENSITIVITY = READ, MODIFY, or PRIVATE;
  OPTION = PROMPT, SKIP, REPLACE, NEW, TERMINATE;
  QUIET;
  VERBOSE.
```

The only required parameter for the VPLUS command is FF. However, it is recommended that HYPEN parameter is set to "ON". Underscores are converted to dashes in the primary name, and the underscore is kept in the alias name.

```
>LOAD.
```

This starts the process of loading VPLUS definitions into System Dictionary.

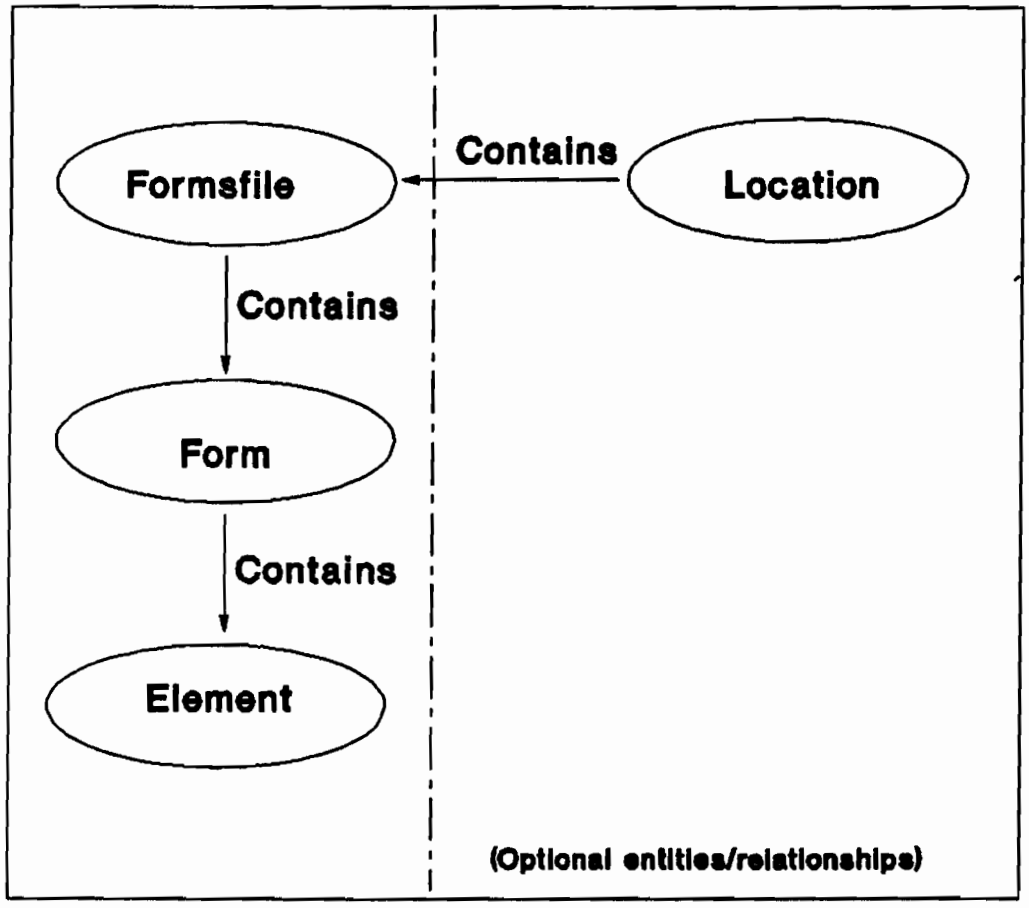

Diagram 4: VPLUS Formsfile Entity-Relationship Model

The Entity-Relationship definitions loaded by SDVPD are illustrated in Diagram 4.

4. Load MPE file definitions.

There are no utilities to automatically load an MPE file because there is no way to differentiate the start or length of a field. SDMAIN, a command-driven interface to System Dictionary, is used to manually create the Entity-Relationship definitions for an MPE file. Here is an example of creating a file called MPEDATA with 2 elements, ELEMENT1 and ELEMENT2.

NOTE

SDMAIN commands may be abbreviated. As you can see in the examples on the following pages, the abbreviated or non-abbreviated commands may be used.

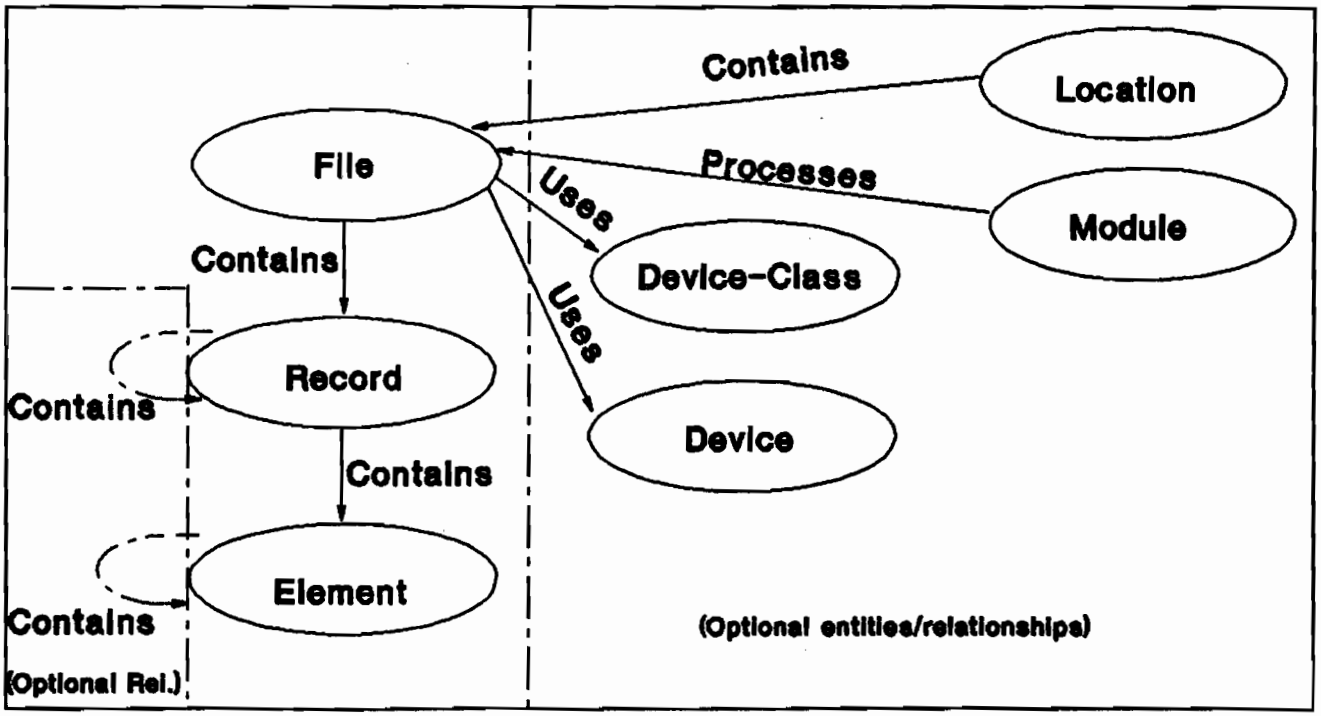

Diagram 5: MPE File Entity-Relationship Model

Diagram 5 illustrates the entities and relationships that are manually created within SDMAIN to define an MPE file.

```
:RUN SDMAIN.PUB.SYS
 >DEFINE SCOPE=DA;PASSWORD=DA;DOMAIN=;STATUS=TEST;
 >>OPEN-MODE=SU.

 >CREATE ENTITY ELEMENT1;ENTITY-TYPE=ELEMENT.
  (Assign pertinent attribute values for this element such
   as Element-Type, Byte-Length, Decimal, and any others).
```

```
>C E ELEMENT2;ET=ELEMENT.
 (This is an abbreviated form of "CREATE ENTITY" command.
  Any pertinent attribute values for this element are assigned when
  creating the element).

>C E MPEDATA;ET=RECORD.
 (Assign attribute BYTE-LENGTH=proper number of bytes)

>C E MPEDATA;ET=FILE.

>CREATE RELATIONSHIP MPEDATA, MPEDATA;
>>    RELATIONSHIP-TYPE=FILE,RECORD;
>>    RELATIONSHIP-CLASS=CONTAINS.

>C R MPEDATA,ELEMENT1;RT=RECORD,ELEMENT;RC=CONTAINS.
 (This is an abbreviated form of "CREATE RELATIONSHIP" command.
  Set the BACK-REFERENCE-FLAG attribute to "TRUE" so that
  the definition of ELEMENT1 is retrieved from the entity level).

>C R MPEDATA,ELEMENT2;RT=RECORD,ELEMENT;RC=CONTAINS.
 (Assign the BACK-REFERENCE-FLAG attribute to "TRUE").
```

5. **Load KSAM file definitions.**

Likewise for a KSAM file, there are no utilities to automatically load a definition of a KSAM file into System Dictionary. SDMAIN is used to create the individual Entity-Relationship definitions. Here is an example of defining a KSAM data and key file with a primary key in column 1.

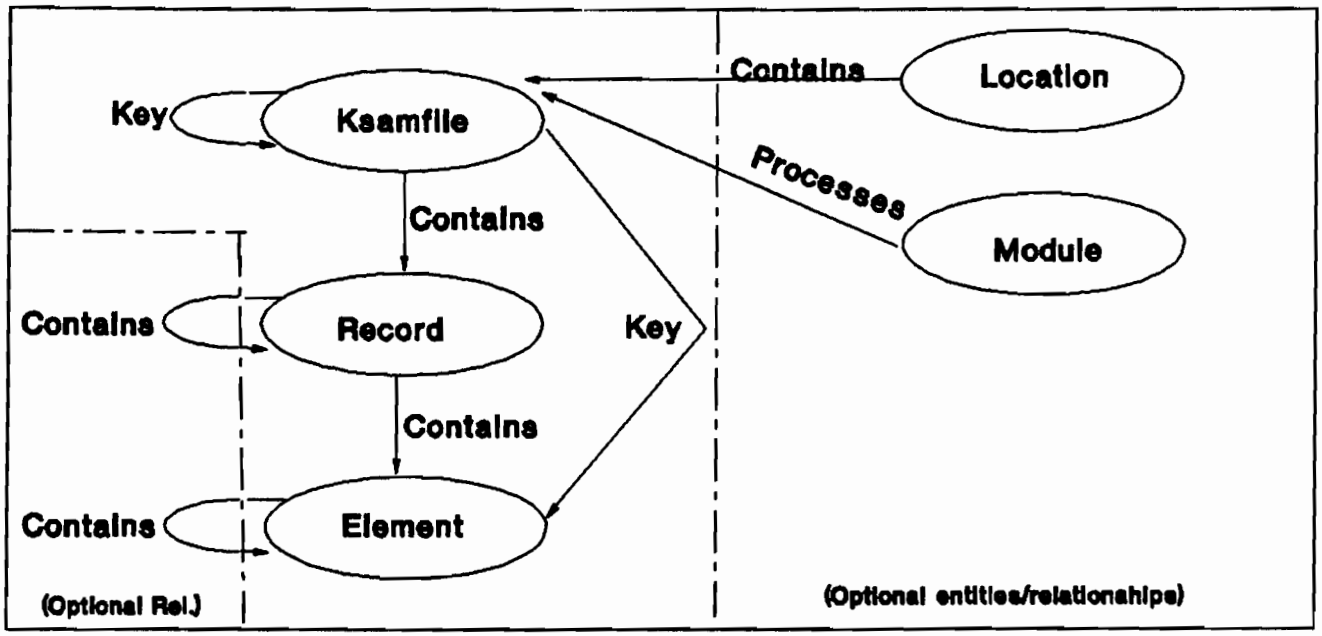

Diagram 6: KSAM file Entity-Relationship Model

Diagram 6 illustrates the entities and relationships that are created manually within SDMAIN to define a KSAM file.

```
:RUN SDMAIN.PUB.SYS
>DEF S=DA;P=DA;D=;STAT=TEST;OM=SU.

>C E KSAMD;ET=KSAMFILE.
 (Assign the KSAMFILE-TYPE attribute a value of "DATA".)

>C E KSAMK;ET=KSAMFILE.
 (Assign the KSAMFILE-TYPE attribute a value of "KEY".)

>C E KSAMREC;ET=RECORD.
 (Assign BYTE-LENGTH attribute the proper number of bytes)

>C E KSAMITM1;ET=ELEMENT.
 (Assign appropriate attribute values for ELEMENT-TYPE,
 BYTE-LENGTH, DECIMAL, and any others).

>C E KSAMITM2;ET=ELEMENT.
 (Assign pertinent attribute values)

>C R KSAMD,KSAMK;RT=KSAMFILE,KSAMFILE;RC=KEY.
 This establishes the relationship between a KSAM data
 file and a KSAM key file.

>C R KSAMD,KSAMITM1;RT=KSAMFILE,ELEMENT;RC=KEY.
 This establishes the relationship between a KSAM file and
 its key.  The attribute of PRIMARY-FLAG allow you to specify
 whether the element is a primary or secondary key.  PRIMARY-FLAG
 takes on a value of "TRUE" or "FALSE".

 The attribute of UNIQUE specifies whether the key allows
 duplicates or not.  It takes on a value of "TRUE" or "FALSE".

>C R KSAMD,KSAMREC;RT=KSAMFILE,RECORD;RC=CONTAINS.

>C R KSAMREC,KSAMITM1;RT=RECORD,ELEMENT;RC=CONTAINS.
 The BACK-REFERENCE attribute should be set to "YES" so
 the definition of KSAMITM1 is retrieved from the entity level.

>C R KSAMREC,KSAMITM2;RT=RECORD,ELEMENT;RC=CONTAINS.
 The BACK-REFERENCE attribute should be set to "YES".
```

6. (Optionally) Load DICTIONARY/3000 definitions into System Dictionary

You may have an existing DICTIONARY/3000 with definitions already defined. SDCONV utility is used to copy and convert definitions from DICTIONARY/3000 to System Dictionary. SDCONV does not modify DICTIONARY/3000 definitions.

3 Commands used in SDCONV to convert DICTIONARY/3000 to System Dictionary are SYSDIC, DICT3000 and LOAD:

```
>SYSDIC
    NAME = System dictionary name;
    SCOPE = scope-name;
    PASSWORD = scope-password;
    DOMAIN = domain-name;
    VERSION = version-name;
    OPEN-MODE = open-mode;
    NAME-MODE = name-mode.

>DICT3000
    NAME = dictionary-3000-name;
    OPEN-MODE = image-mode;
    SCOPE-OWNER = RESPONSIBLE, IDENTITY-CREATE, or LOGON;
    BACK-REFERENCE = TRUE or FALSE;
    SENSITIVITY = READ, MODIFY, or PRIVATE;
    ALIAS = COB, HPSQL, IMAGE, PASCAL, STD, VPLUS;
    PACK-DESCRIPTION = TRUE or FALSE;
    QUIET;
    VERBOSE.
```

One important note to keep in mind is that you cannot specify the DICTIONARY/3000 password. SDCONV uses the Dictionary/3000's creator password of " ; ". Therefore, when running SDCONV you must logon as the creator of the DICTIONARY/3000 AND have MPE security access to System Dictionary.

These parameters except dictionary name, and open mode are optional.

```
>LOAD
```

This starts the process of converting Dictionary/3000 definitions to System Dictionary.

7. Fine Adjustment Definitions.

Fine adjustment of the definitions are performed after the IMAGE, VPLUS VPLUS, KSAM and MPE files have been loaded. The fine adjustments may include attaching descriptions to entities and relationships, ensuring the decimal attribute is set for IMAGE elements, and so on.

To make adjustments, you must use SDMAIN to modify the definitions.

A summary on modifying and reporting entities and relationships follow:

- The syntax for modifying an entity within SDMAIN is:

```
>MODIFY ENTITY entityname;ENTITY-TYPE=entity-type.

(Abbreviated: >M E entityname;ET=entity-type.)
```

SDMAIN then prompts for attributes that you wish to change.

- The syntax for modifying a relationship within SDMAIN is:

```
>MODIFY RELATIONSHIP relationshipname;
>>  RELATIONSHIP-TYPE=relationship-type;
>>  RELATIONSHIP-CLASS=relationship-class.

(Abbreviated:  >M R relationshipname;RT=relationship-type;
                  >>   RC=relationship-class.)
```

Again, SDMAIN prompts for the attribute changes.

- The syntax for displaying an entity's attributes is:

```
>REPORT ENTITY entityname;ENTITY-TYPE=entity-type.

(Abbreviated: >R E entityname;ET=entity-type.)
```

- The syntax for displaying a relationship's attributes is:

```
>REPORT RELATIONSHIP entity1,entity2;
>> RELATIONSHIP-TYPE=relationship-type;
>> RELATIONSHIP-CLASS=relationship-class.

(Abbreviated: >R R entity1,entity2;RT=relationship-type;
                 >>   RC=relationship-class.)
```

The caret symbol may be substituted as a wildcard character for an entity name. Therefore, to get all elements in a record, the caret symbol is used in place of an element name.

8. Handle Name Conflicts.

During the load of your IMAGE, VPLUS, MPE and KSAM files, you may encounter conflicting definitions for an item with the same name. There are 2 ways to handle these conflicts.

- Specify another "primary" name for the entity and use the alias attribute to hold the real-world entity name. The primary name is the internal or external name depending on the name mode which you opened the dictionary.

  There are several aliases which may hold the real-world entity name if a name conflict occurs.

  For example, the IMAGE-ALIAS attribute is used for SDDBD. If SDDBD detects a name conflict while loading the dictionary, it gives you an option to assign a new primary name to the item and then assigns the real-world name to the IMAGE-ALIAS attribute.

  Likewise, SDVPD gives you the same options as SDDBD when SDVPD encounters a name conflict. However with SDVPD, the real-world name is assigned as the VPLUS-ALIAS attribute.

There are several other attributes which you may use to hold an entity's alias: STANDARD, PASCAL, COBOL, and HPSQL.

- The second method of handling conflicting definitions is to use a separate partition of the dictionary, that is use a separate domain. Since domains are logically separate dictionaries, this prevents name conflicts.

UNIQUE FEATURES OF SYSTEM DICTIONARY

System Dictionary features make it a unique and flexible dictionary product. Some of the features are: the ability to customize entity-types and relationship-types; availability of intrinsics to programmatically access System Dictionary; partitions within System Dictionary; and finally, an extensive security scheme.

▪ SYSTEM DICTIONARY INTRINSICS

System Dictionary intrinsics are available as part of the product for programmatic access. The intrinsics allow you to report or modify the definitions within System Dictionary. For example, the intrinsic SDFindRelList retrieves a list of relationships which meet the criteria specified within the search.

▪ COMPREHENSIVE SECURITY

System Dictionary has a comprehensive security scheme. There are 5 levels of security in addition to the normal MPE security. Diagram 7 shows these 5 levels of security.

System Dictionary Security Levels

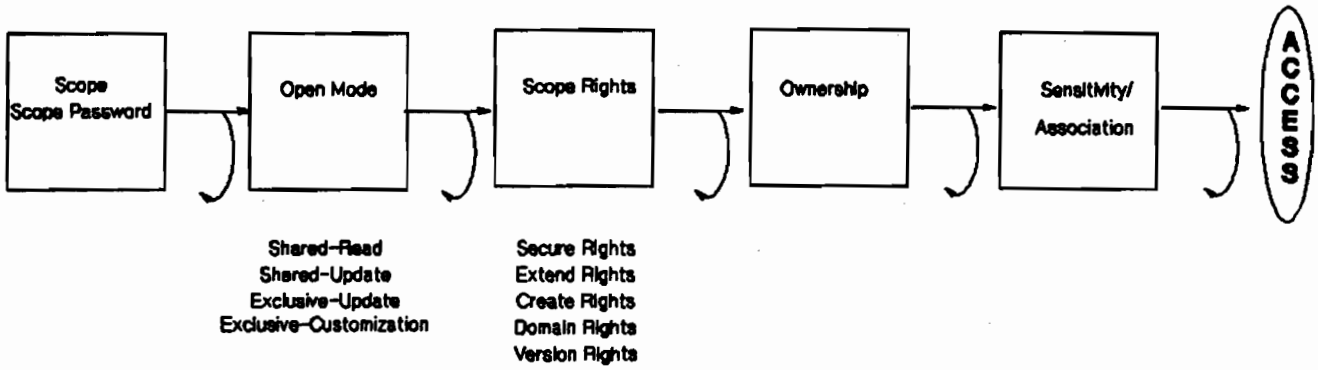

Diagram 7

First, a correct scope (dictionary user name) and the scope password must be specified to open the dictionary.

Second, the open mode specified allows you certain activities and prevents you from other activities. The open modes which may be used to open the dictionary are: Shared-Read, Shared-Update, Exclusive-Update, or Exclusive-Customization.

Third, your scope rights again allow you certain activities and prevents you from others. When a scope is created, certain rights are granted to it. When the dictionary is created, the Dictionary Administrator scope exists and has all rights. The rights that may be granted to scopes are:

```
SECURE - Capability to create and own other scopes

EXTEND - Capability to customize the dictionary

CREATE - Capability to create, modify, delete and read entities &
         relationships

DOMAIN - Capability to create and own domains.  Includes the right
         of VERSION CONTROL.

VERSION CONTROL - Capability to create and own versions.
```

Fourth, being an owner gives you sole rights over an object created.

Fifth, SENSITIVITY attribute and association allow non-owner scopes certain access. The SENSITIVITY attribute and association are set by the owner for a particular object (entity, relationship, domain). SENSITIVITY level on an entity or relationship may be set to: PRIVATE, PUBLIC READ, PUBLIC MODIFY. It is the owner's discretion to assign the sensitivity level when setting up an entity, relationship, or domain.

If the SENSITIVITY is PRIVATE on an entity, relationship or domain; the OWNER may allow certain scopes to override this security in exception cases. Associating a scope with an entity, relationship, or domain overrides the SENSITIVITY level. Thus, those associated scopes have their level of scope rights to the object. For example, if the associated scope has a scope right of READ when it was created; the associated scope may only read.

▪ CUSTOMIZABLE DICTIONARY

So far, the entity-types and relationship-types that we have mentioned are included with the "core-set". The core-set includes pre-defined entity-types and their associated attributes, relationship-types and their associated attributes, and relationship classes.

You may customize the dictionary by defining new entity types, relationship types, attribute associations or relationship classes. For example, you may define a company organization chart within System Dictionary by defining reporting entity-types and relationship-types.

In addition, you may customize the core-set to a limited extent.

▪ CUSTOMIZING SDMAIN USER INTERFACE

System Dictionary also has the capability to use MACRO files so that lengthy commands may be abbreviated to certain keywords and parameters. MACRO files are similar to setting up a User-Defined-Command (UDC) within MPE. This makes using SDMAIN friendlier and reduces keystrokes.

INCLUDE files also make using SDMAIN easier and quicker. INCLUDE files contain a sequence of commands that you "batch" together. This saves you the time and effort of typing the sequence of commands over and over.

▪ PARTITIONED DICTIONARY

And finally, domains and versions are a unique feature which allow you to partition the System Dictionary. This feature was discussed in a the System Dictionary Concept section.

SYSTEM DICTIONARY USES

Below lists the current use of System Dictionary. The first use listed is mentioned within the introduction of this paper.

1. Provide a central location for documentation and standards on the information systems environment.

2. Generate copylib members through SDCDE and SDPDE SDCDE is the COBOL Definition Extractor, and SDPDE is the Pascal Definition Extractor. Note: These are separate products from System Dictionary.

3. Store data definitions used by BRWSD and/or RSYSDIC to create a compiled version of the dictionary for use by BRW/V and BRW/XL.

4. Store data definitions used by fourth-generation languages such as TRANSACT.

5. Store data definition used by HP's Virtuoso product, a COBOL code generator.

For detailed information on the operation of any particular feature, refer to the appropriate System Dictionary manual,

for MPE V/E:

HP System Dictionary/V General Reference Manual, Volume 1
(Part-No. 32254-90004)

HP System Dictionary/V General Reference Manual, Volume 2
(Part-No. 32254-90005)

HP System Dictionary/V Utilities Reference Manual (Part-No. 32254-90003)

HP System Dictionary/V SDMAIN Reference Manual (Part-No. 32254-90001)

HP System Dictionary/V Intrinsics Reference Manual (Part-No. 32254-90002)

for MPE/XL:

HP System Dictionary/XL General Reference Manual, Volume 1
(Part-No. 32256-90004)

HP System Dictionary/XL General Reference Manual, Volume 2
(Part-No. 32256-90005)

HP System Dictionary/XL Utilities Reference Manual (Part-No. 32256-90003)

HP System Dictionary/XL SDMAIN Reference Manual (Part-No. 32256-90001)

HP System Dictionary/XL Intrinsics Reference Manual (Part-No. 32256-90002)

BACK ISSUE INFORMATION

Following is a list of the Application Notes published to date. If you would like to order single copies of back issues please use the *Reader Comment Sheet* attached and indicate the number(s) of the note(s) you need.

Note #	Published	Topic
1	2/21/85	*Printer Configuration Guide (superseded by note #4)*
2	10/15/85	*Terminal types for HP 3000 HPIB Computers (superseded by note #13)*
3	4/01/86	*Plotter Configuration Guide*
4	4/15/86	*Printer Configuration Guide - Revised*
5	5/01/86	*MPE System Logfile Record Formats*
6	5/15/86	*Stack Operation*
7	6/01/86	*COBOL II/3000 Programs: Tracing Illegal Data*
8	6/15/86	*KSAM Topics: COBOL's Index I/O; File Data Integrity*
9	7/01/86	*Port Failures, Terminal Hangs, TERMDSM*
10	7/15/86	*Serial Printers - Configuration, Cabling, Muxes*
11	8/01/86	*System Configuration or System Table Related Errors*
12	8/15/86	*Pascal/3000 - Using Dynamic Variables*
13	9/01/86	*Terminal Types for HP 3000 HPIB Computers - Revised*
14	9/15/86	*Laser Printers - A Software and Hardware Overview*
15	10/01/86	*FORTRAN Language Considerations - A Guide to Common Problems*
16	10/15/86	*IMAGE: Updating to TurboIMAGE & Improving Data Base Loads*
17	11/01/86	*Optimizing VPLUS Utilization*
18	11/15/86	*The Case of the Suspect Track for 792X Disc Drives*
19	12/01/86	*Stack Overflows: Causes & Cures for COBOL II Programs*
20	1/01/87	*Output Spooling*
21	1/15/87	*COBOLII and MPE Intrinsics*
22	2/15/87	*Asynchronous Modems*
23	3/01/87	*VFC Files*
24	3/15/87	*Private Volumes*
25	4/01/87	*TurboIMAGE: Transaction Logging*
26	4/15/87	*HP 2680A, 2688A Error Trailers*
27	5/01/87	*HPTrend: An Installation and Problem Solving Guide*
28	5/15/87	*The Startup State Configurator*
29	6/01/87	*A Programmer's Guide to VPLUS/3000*
30	6/15/87	*Disc Cache*
31	7/01/87	*Calling the CREATEPROCESS Intrinsic*
32	7/15/87	*Configuring Terminal Buffers*
33	8/15/87	*Printer Configuration Guide*
34	9/01/87	*RIN Management (Using COBOLII Examples) (A)*
34	10/01/87	*Process Handling (Using COBOLII Examples) (B)*
35	10/15/87	*HPDESK IV (Script files, FSC, and Installation Considerations)*
34	11/01/87	*Extra Data Segments (Using COBOLII Examples) (C)*
36	12/01/87	*Tips for the DESK IV Administrators*
37	12/15/87	*AUTOINST: Trouble-free Updates*
38	1/01/88	*Store/Restore Errors*
39	1/15/88	*MRJE Emulates a HASP Workstation*
40	2/01/88	*HP 250 / 260 to HP 3000 Communications Guidelines*
41	4/01/88	*MPE File Label Revealed - Revised 6/15/88*
42	7/15/88	*System Interrupts*
43	7/15/88	*Run Time Aborts*
44	8/01/88	*HPPA Pathing Conventions For HP3000 900 Series Processors*
45	8/15/88	*Vplus & Multiplexers*
46	8/15/88	*Setting Up An HPDesk/HPTelex For The First Time*

READER COMMENT SHEET

Worldwide Response Center Support
HP 3000 Application Note 62: Setting Up A System Dictionary
(November 15, 1989)

We welcome your evaluation of this Application Note. Your comments and suggestions help us to improve our publications. Please explain your answers under Comments, below, and use additional pages if necessary.

Is this Application Note technically accurate? ☐ **Yes** ☐ **No**

Are the concepts and wording easy to understand? ☐ **Yes** ☐ **No**

Is the format of this Application Note convenient in size, arrangement and readability? ☐ **Yes** ☐ **No**

Comments and/or suggestions for future Application Notes:

This form requires no postage stamp if mailed in the U.S. For locations outside the U.S., your local HP representative will ensure that your comments are forwarded.

FROM: **Date** _____

Name _____

Company _____

Address _____

_____

_____

FOLD FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 95, MT. VIEW, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Application Note Comments
Hewlett-Packard Manufacturing Specifications
690 E. Middlefield Road
Mail Stop 30-0
Attention: AN ORDERS
Mt. View, CA 94043

FOLD FOLD