
System 1032[®]

User's Guide

System 1032 Version 9.00

CompuServe
Data Technologies

1 Managing System 1032 Catalogs

Overview

This chapter describes the steps for creating System 1032 catalogs. A catalog is a dataset, database, or library. Specifically, this chapter tells you how you can:

- Create a dataset, both interactively and from a user definition file.
- Define datasets within a database, either interactively or from external description files.
- Create databases and libraries.

This chapter assumes that you are familiar with the System 1032 commands needed to create a simple dataset and to examine and update datasets. You should also be familiar with the System 1032 concepts described in Module 1.

To design datasets, first determine how their information should be organized: how many attributes will be in each dataset and what information will be in each one. The CREATE command sets up the dataset according to your dataset definition.

After you set up the dataset, you can then *populate* it with data. If your data is stored in a Record Management Services (RMS) file, use the LOAD command to transfer, move, or copy the data from the file into the System 1032 dataset. You can also populate a dataset by using the ADD command to type the data directly into the dataset.

Datasets are the basic building blocks of databases. This chapter describes the structure of a dataset, then outlines two ways you can create them. It also refers to the commands that you use to define the items in a dataset or database. In most cases, these commands are fully described in the *System 1032 Programmer's Reference Manual*.

Basic Dataset Structure

A System 1032 dataset is the basic unit for accessing and manipulating data. You can define a dataset independently or as part of a database. However, you can always access a dataset independently, even when it is defined as part of a database.

Figure 1-1 illustrates the contents of a dataset, showing, from top to bottom, the order of items as they are often defined.

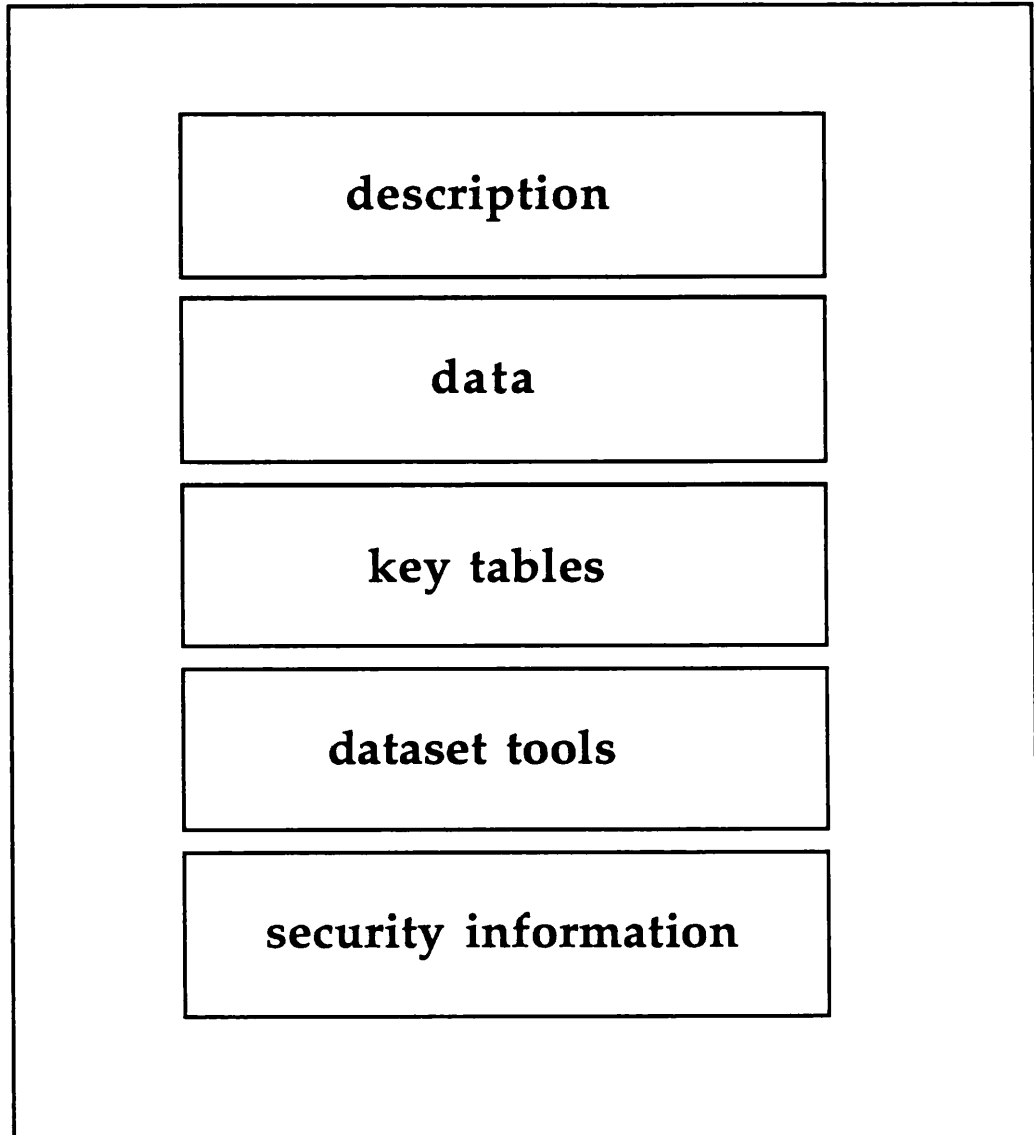


Figure 1-1 System 1032 Dataset

The dataset record template is a group of attributes that define the structure of the records in a dataset. Each attribute describes one item of data that occurs in every record in the dataset. The record template specifies what kinds of data can be stored in the dataset and how the user can manipulate the data. The template specifies the name, data type, and number of attributes in the dataset.

The records contain the data corresponding to the attributes defined in the record template. Each record is identified by a unique \$ID, a system attribute used internally as the record index.

System 1032 uses key tables to determine which records contain a particular value or a set of values. A key table contains all values for one keyed attribute and identifies the records (using the \$IDs) that contain each value. For example, System 1032 uses the key tables to find data to fill a selection set.

Creating Datasets

A System 1032 dataset may contain records, key tables, tools, and security information. When you define the dataset, you can specify the files used to store these items. If you do not specify individual files for these items, System 1032 stores all parts of the dataset in one file.

Dataset creation is a two-step process:

- Define the structure of the dataset.
- Load the data into the dataset.

After you set up the dataset with the CREATE DATASET command, you are ready to enter data into the dataset.

Use the INSERT and REMOVE commands to add or remove definitions of dataset tools, including forms, procedures, RDs, and variables. For more information, refer to the *System 1032 Programmer's Reference Manual*.

The Dataset Record Attributes

When you create a dataset, you are defining the record attributes, which are comprised of ATTRIBUTE commands. Each ATTRIBUTE command specifies the qualities of an attribute, such as its name, data type, external length, key status, print defaults, prompts, and comments.

You can also provide additional information about the attribute: load defaults, key defaults, RDs, procedures, forms, and variables.

In addition, you can also use the ATTRIBUTE command to describe the length or column position of external data in a data file.

Before you create record attributes, decide the following:

- How many attributes (that is, what kind of information) the dataset will have.
- What kind of data you already have.
- What data you want to add in the future.
- How you want to use the information.

While this chapter provides basic information about dataset creation, subsequent chapters describe more specialized operations. Table 1-1 describes the types of information you can define in a dataset, lists the related commands, and tells where in this module you can find detailed information. For more information, refer to the ATTRIBUTE command in the *System 1032 Programmer's Reference Manual*.

Table 1-1 Information Types Allowed in Datasets

Information Type	Related Commands	Command Description	Additional Information
Attribute definition	ATTRIBUTE	Specifies the qualities of an attribute, such as name, data type, and length.	Chapter 3-1
Dataset name and comment	DATASET	Specifies the dataset name and comment in a dataset definition.	Chapter 3-1
External data commands	RD, LOAD, APPEND, DUMP	These commands include record descriptors (RD), variables, procedures, and forms. Use the RD command to describe the format of an external record. System 1032 uses a record descriptor in the LOAD, APPEND, and DUMP commands and for communicating with the host language interface.	Chapter 3-2 .
Forms			
Journaling	JOURNAL	Establishes a journal file for the dataset to be used during recovery in case of dataset damage. You can specify a name for the journal file.	Chapter 3-1, 3-4
Keying attributes	KEY_DEFAULTS	Data definition command. KEY_DEFAULTS sets defaults used for key table creation and controls some efficiency parameters that you cannot control with KEY. In the KEY_DEFAULTS command, you can specify the files used to store the key tables.	Chapter 3-1
Loading default parameters	LOAD_DEFAULTS	Data definition command. LOAD_DEFAULTS sets the defaults for the LOAD, APPEND, DUMP, ADD, DM_ADD_REC, and DM_ADD_SET commands and controls some efficiency parameters that you cannot specify with those commands. In the LOAD_DEFAULTS command, you can specify the file used to store the records.	Chapter 3-2
Procedures, variables	PROCEDURE, VARIABLE	PROCEDURE defines a compiled Module 2 or external procedure that can consist of PL1032 or other high-level language. VARIABLE defines a command repository in which you can temporarily place values.	
Record security	RECORD_SECURITY	Establishes defaults for reevaluating user access to records when the PREVENT RECORDS command is in effect.	Chapter 3-3.

Naming Attributes

Attribute names must begin with a letter and can include only letters, numbers, and underscores. Embedded spaces are not allowed. For example, if you want to use a name

with two or more words in it, separate the words with an underscore, such as LAST_NAME. The maximum length allowed is 127 characters.

Choose a name that reflects the content of the attribute, to help others using the dataset. For example, when a user gives the SHOW ATTRIBUTE command to see the attributes in the dataset, LAST_NAME is self-explanatory while ATTR3 is not. Specify a name that is unique for the dataset and avoid using System 1032 keywords as names. Also, use a short name to ensure more efficient data handling by key tables and forms.

An ATTRIBUTE command assigns one primary name to one attribute. This primary name is used as the default column title by the PRINT command and as the default prompt text by the ADD command.

You can also assign one or more secondary names to the attribute. While a primary name describes the attribute, the secondary name can be an abbreviation for the primary name. For example, LN is a useful abbreviation for LAST_NAME. You can use the primary and secondary names interchangeably. For example, use the OR option to assign one or more secondary names to the attribute.

```
DDL_ ATTRIBUTE FILM_TITLE OR FILM OR NAME TEXT 35
```

The attribute FILM_TITLE has two secondary names: FILM and NAME. TEXT is the data type and 35 characters is the assigned length of the attribute.

Defining Two or More Attributes at Once

You can define two or more attributes that share the same data type and qualities by using one ATTRIBUTE command.

To define more than one attribute, separate the names with a comma and repeat for any number of attributes.

```
DDL_ ATTRIBUTE SURNAME OR LAST_NAME, FIRST_NAME -  
DDL_ TEXT 15 KEYED
```

This command defines two separate attributes; the first attribute also has a secondary name.

Like all dataset qualities, all attributes defined with the ATTRIBUTE command share the same comment.

Note that you cannot define multiple attributes with one ATTRIBUTE command if the attribute has a group data type or is an element of a group (see the Language chapter of the *System 1032 Programmer's Reference Manual*).

Specifying Data Types

After you've named the attribute, you must specify its data type. The data type determines what kind of value the attribute can have, such as text characters (TEXT) or whole numbers (INTEGER). The data type determines what kind of operations you can perform with the values. For example, you can print a total for integers but not for text.

Attribute Qualities and Types

After you specify the basic data type, you can use any combination of attribute qualities in any order. The type qualities that you can use, such as MISSING and PRECISION,

are valid only for certain data types. You can use attribute qualities such as COMMENT and PROMPT for any attribute. Other qualities you can define, such as LENGTH and COLUMN, are discussed in later chapters.

If you are using one command to define more than one attribute, do not specify different type or attribute qualities for individual attributes.

The following sections define several common attribute qualities and types. For more information, see the Data Types chapter of the *System 1032 Programmer's Reference Manual*.

Associating a Comment with an Attribute

Use the COMMENT option to specify a comment that is permanently associated with the attribute. The SHOW ATTRIBUTE COMMENT command displays the attribute comment. This provides very useful information for the dataset users. The attribute comment can be any series of characters, up to the maximum of 32,767.

```
DDL_ ATTRIBUTE DIRECTOR_ID INTEGER KEYED COMMENT -  
DDL_ "Used for mapping to DIRECTORS"
```

Preventing the Insertion of Blank Data

If you want to prevent a user from entering a missing value during data entry, specify MISSING PROHIBIT in the ATTRIBUTE command along with the INITIALLY clause. In the following example, if the user presses <Return>, the initial value of TRUE will be entered into the dataset. If MISSING PROHIBIT were not part of the attribute definition, System 1032 would enter a MISSING value when the user pressed <Return>.

```
DDL_ ATTRIBUTE COLOR LOGICAL TITLE CENTERED "Film in/Color" -  
DDL_ FORMAT L PROMPT "Is film in color " INITIALLY TRUE -  
DDL_ MISSING PROHIBITED
```

Controlling Prompts for Data Entry

You can control whether or not the user is prompted for the attribute while entering data and if so, what prompt System 1032 uses. In addition, you can specify that the user must provide a value for the attribute.

If you design the attribute with a prompt, the prompt answers when the user types the ADD command, followed by a <Return>.

By default, if the ADD command does not specify an attribute and a value for the attribute, System 1032 prompts with the primary name of the attribute. The PROMPT option allows you to specify the prompt text that is used in place of the attribute primary name. PROMPT is also used in generating default forms, using the FIELD COPY command. A prompt can be any text constant up to 80 characters.

```
DDL_ ATTRIBUTE FILM_TITLE TEXT 35 PROMPT "Title of Film"
```

The above example defines the attribute FILM_TITLE and specifies a prompt.

The prompt "Title of Film" appears and the user must specify the film title. The user then types the title and presses <Return> to move to the next entry. If FILM_TITLE is defined as a DEMAND attribute, and the user enters no value, he or she receives a warning message and is re-prompted. If the FILM_TITLE is *not* a DEMAND attribute

and the user enters no value, System 1032 enters a missing attribute. However, if the attribute is defined with an initial value, that value appears at the prompt and is added if the user presses <Return>.

For attributes of all data types except logical, System 1032 prints a colon and a space after the prompt. For logical attributes, System 1032 prints a question mark instead of a colon. Keep this in mind when defining the prompt text for logical attributes.

If you want to require that the user enter a value while adding data, specify the DEMAND option. If the user types <Return> at the prompt, System 1032 displays the message "value is required" and repeats the prompt. If you specified SET MESSAGE NO PRINT, no message is displayed, but System 1032 displays the prompt each time a user presses <Return>.

```
DDL_ ATTRIBUTE FILM_TITLE TEXT 35 -  
DDL_ KEYED TITLE "Film Title" PROMPT "Title of Film" -  
DDL_ DEMAND
```

In this case, the user is prompted for a FILM_TITLE. The DEMAND option specifies that the user cannot add a FILMS record without typing the name of the film, which must comprise at least one character.

Keying Attributes for Rapid Access

This section describes the two operations that key attributes during dataset creation: the KEYED option and the KEY_DEFAULTS Data Description Language (DDL) command.

A key table is an index to records in the dataset with the same attribute values and corresponding indexes. It indicates which values are associated with which particular records. System 1032 uses key tables to locate records rapidly in a database. When you issue a FIND, DFIND, or MAP command, System 1032 uses the key table to point to records that contain a particular value or set of values.

Other commands rely on key tables. The SORT command can sort more efficiently if an attribute is keyed. The VALUES command also requires key tables to produce an efficient summary of the values for an attribute in a selection set. The JOIN command requires keyed attributes. The number of keyed attributes in a dataset also affects the efficiency of ADD, CHANGE, LOAD, and APPEND. Figure 1-2 illustrates a key table of the RELEASE_YEAR attribute in the FILMS datasets.

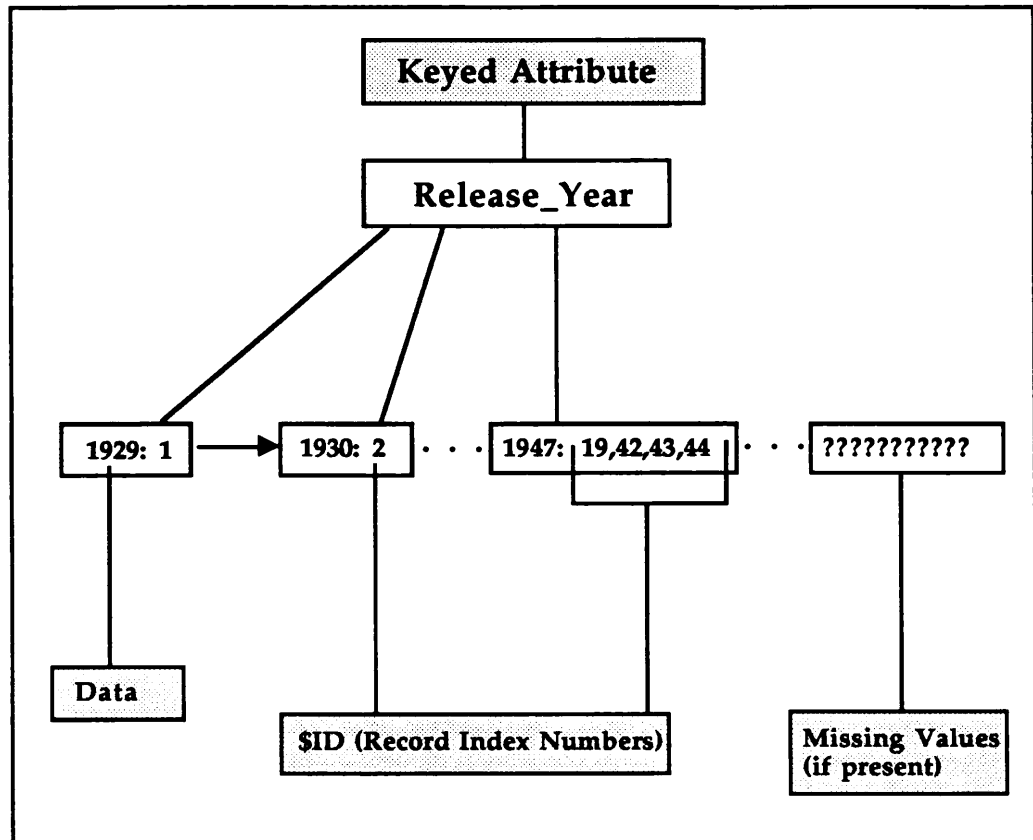


Figure 1-2 Typical Key Table

In the above figure, the `RELEASE_YEAR` attribute is keyed. The key table contains an index of all the records in which the names 1929, 1930, and 1947 appear. For example, if you search for all the records that contain 1947, System 1032 examines the key tables and selects record IDs 19, 42, 43, and 44. If there were missing values within this attribute, they would appear at the end.

A key table is part of a dataset. Depending on how you set up the dataset, the key table is stored in the database file (`<filename>.DMB`), the dataset file (`<filename>.DMS`) or in a special key file (`<filename>.DMK`). The amount of space taken up by a key table depends on:

- The number of records in the dataset.
- The number of different values in the particular attribute.
- The amount of null space built into the keys. For more information, refer to the `KEY_DEFAULTS` command in the *System 1032 Programmer's Reference Manual*.
- The length of the primary attribute name.
- The size of the attribute values.

You can control the keys for a dataset during both dataset creation and dataset maintenance. However, this chapter discusses only dataset creation. The chapter on maintenance tells how to use the two key commands that apply to dataset maintenance: `KEY` and `UNKEY`.

Defining Keys During Dataset Creation

During dataset creation, you can use the `ATTRIBUTE` command's `KEYED` option to specify the following:

- Whether the attribute, or part of a text attribute, is to be keyed.
- Whether the key table for an array attribute contains information about array element positions. For more information about keying array attributes, refer to the chapter about external data.
- Whether text values are keyed case-sensitive or not.

Efficiency Tip

You should not key all the attributes in a dataset. Key tables not only take up space in the dataset, but updating them consumes time and resources.

Key only those attributes that you intend to use with the following commands:

- `DFIND`
- `FIND`
- `JOIN`
- `MAP`
You must key the target attribute. You don't have to key the source attribute, but in many cases `MAP` works more efficiently if this attribute is keyed.
- `SORT ASCENDING`
Use only if greater than 5% of the datasets records are in the selection set.
- `VALUES`

Keying attributes does not have to be a permanent operation. After you load the dataset, you can use `KEY` and `UNKEY` to add and delete key tables for any attributes in the dataset.

The `KEYED` element has three options you can specify:

- `BY_ELEMENT` or `FOR_ALL`
Control how an array attribute is keyed.
- `USE_CASE` or `IGNORE_CASE`
Determine whether upper- and lower-case characters are treated as equivalent or different during data retrieval. The default is `IGNORE_CASE`.
- `Substring`
You can key a text substring, which is a range of characters on a certain attribute. For example, if you specify `1:5`, only the first five characters of the `FILM` attribute are keyed.

For more information about these options, refer to the *System 1032 Programmer's Reference Manual*.

Specifying Key Defaults

During dataset definition, System 1032 sets certain defaults that it uses whenever it creates a key table. You can set individual defaults for particular keyed attributes as well as general defaults for all keyed attributes. These defaults are the following:

- No key messages are displayed. To override this default, use the MESSAGE option.
- System 1032 leaves 15% null space per page in the key table. To override this default, use the NULL option.
- The key tables are stored in the dataset file. To override this default, use the INTO option to store the key tables in a VMS file.

You can override these defaults by using the KEY_DEFAULTS command. For example, you can specify that System 1032 display key information messages produced by the execution of the LOAD and KEY commands.

Use KEY_DEFAULTS after you have defined the attributes for the dataset. The defaults set by the KEY_DEFAULTS command take effect whenever you issue the KEY, LOAD, and ADD commands.

You can give only one KEY_DEFAULTS command for the dataset. Therefore, you must specify all the attributes for which you want to set key defaults in one command.

The following sections explain how to override the key defaults of a dataset.

Storing Key Tables

By default, System 1032 stores the key tables in the dataset or database file. Use the INTO option to store the key tables in separate key files. There are three advantages to storing key tables in special files:

- You can store the most frequently used key tables on different disks. The FIND and MAP commands operate more efficiently when the key tables for different attributes in a dataset are stored on different disks.
- Storing the key tables on a separate disk from the records improves updating efficiency.
- When you use the UNKEY ENTIRELY command to delete the key table, System 1032 deletes the key file and frees the space for use by VAX/VMS. If the keys are in the dataset file, System 1032 reuses the space but cannot delete the file.

Usage Tip

Storing the keys in a separate file may make it more difficult to copy, move, or back up your dataset. Refer to the chapter on database maintenance for information on how to move and back up datasets stored in more than one file.

If you specify a new file name in the KEY_DEFAULTS command, System 1032 creates a key file. If you specify an existing System 1032 file, System 1032 adds the key tables to the file.

The following example shows how to store the key tables of the FILM_TITLE attribute in a file called FILMS.DMK. Remember that KEY_DEFAULTS is a *data definition language* (DDL) command. After you issue a CREATE command, System 1032 goes into DDL mode. Once in DDL mode you can then define attributes and defaults for a dataset.

```
DDL_ KEY_DEFAULTS FILM_TITLE INTO FILMS.DMK
```

You do not have to specify a file type. System 1032 assigns the type DMK (Data Management Keys). You can also store all the key tables in one file or assign individual key tables or groups of key tables to specific files. For more information, refer to the *System 1032 Programmer's Reference Manual*.

Increasing the Empty Space in the Key Table

You can specify the percentage of empty or null space allotted per page for the particular key table. Each time a key table is created, System 1032 leaves a percent of each page in the key table empty. The default is 15%, but it is recommended that you use 20% null space. This amount provides enough extra space for updating the key table, without heavily affecting efficiency of query commands.

You can also assign this null space to higher- or lower-level blocks within the key table. The following example assigns a 20% null space to the higher-level blocks in the key table.

```
DDL_ KEY_DEFAULTS NULL HIGHER 20
```

If you do not specify HIGHER or LOWER, NULL assigns the percentage of empty space to both higher- and lower-level blocks.

The *System 1032 Programmer's Reference Manual* discusses the benefits of using higher- and lower-level blocks and their respective effects on different configurations.

Selectively Applying Defaults to Keyed Attributes

Use the ALL_KEYED keyword to apply key defaults to all keyed attributes in the dataset. You can then specify that one set of options apply to all the keyed attributes and that a different option applies to a single attribute. For example:

```
DDL_ KEY_DEFAULTS FILM_TITLE ALL_KEYED NO_MESSAGE
```

In this example, the default key options apply to all the keyed attributes in the dataset except the FILM_TITLE attribute. Messages are printed for this attribute. Options specified for individual attributes override the general settings.

To store all the key tables in one file, use the ALL_KEYED keyword, as shown below.

```
DDL_ KEY_DEFAULTS ALL_KEYED INTO FILMS.DMK
```

Specifying that Keying Messages Are Produced

By default, the LOAD and KEY commands do not display keying information messages. Use the MESSAGE option to produce these messages for each keyed attribute. These messages provide information on the size of the key table being created and the number of values a new table contains.

```
DDL_ KEY_DEFAULTS MESSAGE
```

For more information about keying messages, refer to the *System 1032 Programmer's Reference Manual*.

Activating a Journal File During Dataset Creation

A *journal file* is a sequential record of the dataset's update history that you can use to recover a damaged dataset. You can activate and deactivate a journal file for a dataset if:

- You have UPDATE access to an *independently defined dataset*. An independently defined dataset is one that resides in its own VAX/VMS file. This dataset cannot contain items from another dataset, database, or library. Also, it cannot reside within a database.
- No other user has currently opened the dataset.

To specify a name for a dataset's journal file, use the DDL command JOURNAL .

```
DDL_ JOURNAL CINEMA
```

This command creates the journal file CINEMA.DMJ (Data Management Journal) for the independently defined dataset FILMS that resides in FILMS.DMS. Note that if you do not specify a name for the journal file, System 1032 creates a default file name that uses the name of the dataset with the extension DMJ.

Specifying Record Security

When you define a dataset, you can define how users can access the records in the dataset. The security defaults are used only when record security is in effect, that is, when there is at least one PREVENT DATASET RECORDS command specified for the dataset.

The DDL command RECORD_SECURITY defines how users' security is affected when they add and change records in the dataset. You can specify the RECORD_SECURITY command during dataset definition. The following example gives a user access only to his or her own added records, not records added by other users.

```
DDL_ RECORD_SECURITY AUTO_EXPAND
```

For more information about record security, refer to the chapter on security.

Sample Dataset Definition

The following sections describe the two ways you can define datasets:

- Interactive
Allows you to define a dataset by specifying the information line-by-line.
- Description file
Loads a data file that contains attribute definitions.

Use the definition commands described in the following sections to define a dataset or database. Remember that you can enter only the commands to define one dataset, one database, or one library (although the database definition can contain many datasets). You can also use <CHOICE> , <HELP>, <Recognize>, <PF1, <PF2>, and <PF3> to help you with the definition commands.

For a list of allowable data definition commands, refer to the "Command Categories" section of the *System 1032 Programmer's Reference Manual*

Deciding Between Data Entry Methods

Before defining the dataset, you must decide which method you will later use to enter the data, as it affects dataset definition. You can enter data by using either the ADD or the LOAD command:

- **ADD**
Lets you enter the data interactively. The ADD command is easier to use than the LOAD command. If you have only a small number of records to enter (under 50), and if data does not exist in a separate file, use the ADD command.
- **LOAD**
Lets you load data from an external data file. It is more efficient to use the LOAD command to load many records from a data file than to use the ADD command.

When you use the LOAD command to enter the data, you can use LENGTH or COLUMN options in the ATTRIBUTE command. However, the \$DEFAULT_RD assigns default lengths and column positions that specify the position of the data in the external file. For text attributes, you must always specify LENGTH. (You can create a System 1032 record descriptor instead of using LENGTH and COLUMN.)

The following section shows how to define the structure of a dataset with the CREATE DATASET command.

Entering an Interactive Definition

The following example shows how to define a dataset using an interactive definition. It defines a dataset named DIRECTORS, providing explanations when necessary.

1. Type **S1032** at the VAX system prompt. The System 1032 prompt appears.
2. Type the following command:

```
1032> CREATE DATASET DIRECTORS
1032 Data Definition Language
DDL>
```

System 1032 then enters data definition language (DDL) mode, displaying the DDL> prompt. DDL mode allows you to define attributes.

3. Specify the first attribute by typing the following information. (Since this attribute definition cannot fit onto one line, type the hyphen character, then press <Return>. System 1032 prompts with the "DDL_" prompt. The underscore indicates that the definition is incomplete and is repeated from line to line until you complete the dataset definition.)

```
DDL> ATTRIBUTE DIRECTOR_ID OR DID INTEGER Comment -
DDL_ "This attribute is used for mapping to DIRECTORS" -
DDL_ Range 1 : 999 Format I3 Prompt "Director's ID number" -
DDL_ Demand Missing -1 Keyed
```

This defines an integer attribute named `DIRECTOR_ID`, which has an alternate name of `DID`. The attribute is "keyed," which means it has a key table associated with it. Finally, the attribute has a comment attached.

System 1032 checks the syntax of definition commands as you enter them. An error message is displayed when a definition command contains an error. Simply enter the corrected command, and continue with the definition. You can also use `<Delete>` or `<X>` to back up and fix the problem.

4. Enter the following attribute definition:

```
DDL_ ATTRIBUTE FIRST_NAME OR FN OR DFN -  
DDL_ TEXT 15 KEYED PROMPT "Director's First Name" DEMAND
```

Since the `DEMAND` attribute quality is included in this definition, System 1032 will prompt users adding data to the dataset with "Director's First Name."

5. Enter the next attribute definition.

```
DDL_ ATTRIBUTE SURNAME OR SN OR LAST_NAME OR -  
DDL_ LN OR DLN TEXT 15 KEYED
```

Note that you can specify several alternate names for each of these attributes.

6. Enter the `YEAR_OF_BIRTH` attribute:

```
DDL_ ATTRIBUTE YEAR_OF_BIRTH OR YOB DATE -  
DDL_ RANGE '1798':'2000' PRECISION KEYED
```

This attribute definition specifies a `DATE` data type with a range occurring between January 1, 1798 and January 1, 2000. Any birth date entered outside of this range produces a warning message that says the value is out of range. It then reprompts you for a valid answer. For more information about `DATE`, refer to the *System 1032 Programmer's Reference Manual*.

7. Enter the `COUNTRY_OF_BIRTH` and `CITY_OF_BIRTH` definitions:

```
DDL_ ATTRIBUTE COUNTRY_OF_BIRTH OR COUNTRY -  
DDL_ TEXT 15 KEYED USE_CASE  
DDL_ ATTRIBUTE CITY_OF_BIRTH OR CITY TEXT 25 KEYED USE_CASE
```

Note that both of these attributes are keyed and specify that they are case-sensitive. This means that if the data is entered in lowercase, users must search for the data by specifying it in lowercase.

8. Enter the following `KEY_DEFAULTS` statement:

```
DDL_ KEY_DEFAULTS COUNTRY INTO COUNTRY
```

This `KEY_DEFAULTS` command specifies that the key table for the `COUNTRY` attribute resides in a file named `COUNTRY.DMK`.

9. Enter the following `JOURNAL` definition.

```
DDL_ JOURNAL DIRECTORS
```

This JOURNAL command specifies that a JOURNAL file is being built for this dataset and resides in a file named DIRECTORS.DMJ.

10. End the dataset definition by typing:

```
DDL_ END_DATASET
1032>
```

Canceling the CREATE Command

You can cancel the CREATE DATASET command at any point by using the ABORT command before typing END_DATASET in your catalog definition.

First press <CTRL/G> to cancel the partially entered definition. Then enter the ABORT command at the DDL> prompt. System 1032 cancels the CREATE command, displays an error message and returns to the "1032>" prompt. The following example shows how to cancel a CREATE command.

```
1032> CREATE DATASET DIRECTORS
System 1032 Data Definition Language
DDL> ATTRIBUTE DIRECTOR_ID INTEGER LENGTH 3
DDL_ ATTRIBUTE FIRST_NAME TEXT 15 LENGTH 15
DDL_ ^G
DDL> ABORT
%SSystem 1032-E-CREABORT, create command aborted due to user request
1032>
```

Creating a Dataset Using a Description File

Instead of typing an interactive definition, you can supply a dataset definition in a description file, and then create a dataset based on that definition.

Use the DESCRIPTION option of the CREATE command to specify a description file name. System 1032 uses the data definition commands in this file to create the catalog.

The steps below show how to create a dataset using a description file. Remember that you can only include allowable data definition commands within the file.

1. Use a text editor such as VAX/VMS EDT to create a description file containing the following data. Note if you type a <Return> before the end of the attribute definition you need to use the hyphen as a continuation character.

```
DATASET DIRECTORS COMMENT -
"CONTAINS BIOGRAPHICAL INFORMATION ABOUT FILM DIRECTORS" -
ATTRIBUTE DIRECTOR_ID OR DID INTEGER COMMENT -
"THIS ATTRIBUTE IS USED FOR MAPPING TO FILMS" -
RANGE -1 : 999 TITLE "ID" -
FORMAT (I3) PROMPT "DIRECTOR'S ID NUMBER" DEMAND MISSING -1 -
KEYED
ATTRIBUTE FIRST_NAME OR FN OR DFN TEXT 15 TITLE
"DIRECTOR'S/FIRST NAME" -
PROMPT "DIRECTOR'S FIRST NAME" KEYED
```

```
ATTRIBUTE CITY_OF_BIRTH OR CITY TEXT 15 TITLE "CITY OF/BIRTH"  
PROMPT -  
"DIRECTOR'S CITY OF BIRTH"  
End_Dataset
```

This is not the complete DIRECTORS dataset that is in the DEMO database area, nor it is a complete listing of allowable DDL commands.

2. Name the file DIRECTORS.DMD (.DMD for Data Management Definition file). Note that this description file defines only one dataset.
3. Finish the description file with the END_DATASET command (or the abbreviation END) and save the text file.
4. Enter System 1032 and issue the CREATE command with the DESCRIPTION option, followed by the description file name.

```
1032> CREATE DATASET DESCRIPTION DIRECTORS
```

Note that you do not have to specify the file extension if the file has a DMD extension.

Errors in the Description File

If there are errors in the description file, the CREATE command does not accept the definition. System 1032 reports the errors and displays a message indicating that the CREATE command was aborted. The error message shows the line numbers of the lines in the file that contain errors. You can then edit the file, correct the errors, and reenter the CREATE command again.

Several error messages may appear, but there may actually be only one mistake in the description file. If you cannot detect more than one error in the file, the first error in the description file may be causing the rest of the errors. Check by correcting the first error in the description file and then using the CREATE command again.

Dataset Security

When you create a dataset, you become its owner. You can use the System 1032 security commands to restrict access to the dataset. You can use the PREVENT and ADMIT commands to define the dataset security and restrict access to selected records. For more information about these security commands, refer to the chapter on security.

Note

The only other username that can access this dataset is the \$SITE_DBA account holder (usually the database administrator).

Allocating Storage Space for the Dataset

After you create a dataset, you should assign disk space for storing new records. The ALLOCATE BLOCKS command assigns contiguous disk space for the new records. This process is called "contiguous best try." System 1032 tries to assign memory blocks that are closest to each other; if it cannot assign all the blocks you allocate, it attempts to assign as many contiguous ones as possible. When the new records are stored contiguously, retrieval is more efficient. In addition, when you use ALLOCATE, System 1032 does not have to extend the dataset each time you add a record.

If you expect the dataset to grow substantially, use the `ALLOCATE BLOCKS` command to reserve enough blocks to accommodate the expected number of records. This command allocates the specified number of blocks, whether or not an equivalent number of free blocks is already available for records in the dataset. System 1032 produces a message indicating the number of blocks allocated by the command, as well as the number of unused records reserved for the dataset.

```
1032> OPEN DATASET STUDIOS
1032> ALLOCATE 500 BLOCKS
500 blocks added, 1390 records reserved (unused)
in STUDIOS records in "file name"
```

Note that the dataset must be open before you can allocate blocks. The actual number of blocks allocated may be more than the number you request. This is due to disk clustering and internal system rules.

The `LOAD` command automatically estimates the number of records being loaded and pre-allocates enough contiguous disk space.

You can also allocate records. For more information about allocating storage, refer to the *System 1032 Programmer's Reference Manual*.

Defining Databases

A database is a System 1032 unit that includes a database name, one or more datasets, database tool definitions, and join paths. When you open a database, you have access to all datasets and tools in the database. You can also join datasets that are in the same database.

Databases are flexible. If you want to add and remove datasets from your database, you can recreate it. Like other VAX files, you can also transfer the files that contain System 1032 databases to different directories or devices.

You can modify and update definitions in a database. Use the `MODIFY` command to change the definitions of database tools, join paths, and the database comment.

A System 1032 database is stored in one or more System 1032 files. When you define the database, you can specify the files used to store the following items:

- The database itself.
- Individual datasets (or the dataset shells, if records are in DMR files).
- Individual key tables in each dataset.
- Records in each dataset.

In addition, any of these items can be stored in a System 1032 file that contains other System 1032 datasets or databases. If you do not specify individual files for these items, System 1032 stores the entire database in one file. If you include an existing dataset in a database definition, the dataset stays in its original file.

Working with Databases

There are two ways you can create and populate a database:

- Define the database by creating datasets within that database.

- Define the database by including existing datasets in the database. This method is recommended for the following reasons:
 - You can maintain a database comprised of independent datasets better than one that contains the actual datasets. If you plan on adding a new dataset to your database or altering the attribute definitions, you should define your datasets separately and include them in your database.
 - You can attach a journal file to your dataset.

Defining Datasets Within a Database

To define a dataset within a database, use the following steps as a guide.

1. Create a new database named VIDEO_RENTALS.

```
1032> CREATE DATABASE VIDEO_RENTALS
```

System 1032 then enters Data Definition Language, displaying the DDL> prompt.

2. Create a dataset named STORES, by issuing the DATASET command:

```
DDL> DATASET STORES
```

3. The DDL_ prompt appears. Now you can interactively define this dataset within the database, using the same procedure documented in the section entitled "Entering an Interactive Definition." Remember that you can specify up to 32 datasets in a database.
4. End the database definition by typing:

```
DDL_ END_DATABASE
1032>
```

Including Existing Datasets in a Database

To include an existing dataset in a database, use the INCLUDE command. You can specify one or more open datasets in an INCLUDE command. If you have opened the dataset independently of its database (or if it has none yet), System 1032 assigns it to the temporary database named \$TEMP.

The following steps walk you through the process of including a dataset in a database.

1. Open the FILMS, DIRECTORS, and STUDIOS_HEADER, STUDIOS_DETAIL datasets from the MOVIES database. You can optionally separate the names of the datasets with commas.

```
1032> OPEN DATASET FILMS, DIRECTORS
```

A dataset cannot be open under the same name in two databases at the same time.

2. Create a database by issuing the CREATE DATABASE command.

```
1032> CREATE DATABASE VIDEO_RENTALS
```

3. The DDL_ prompt then appears. Type the INCLUDE command for two of the datasets in the MOVIES database.

```
DDL_ INCLUDE FILMS
DDL_ INCLUDE DIRECTORS
```

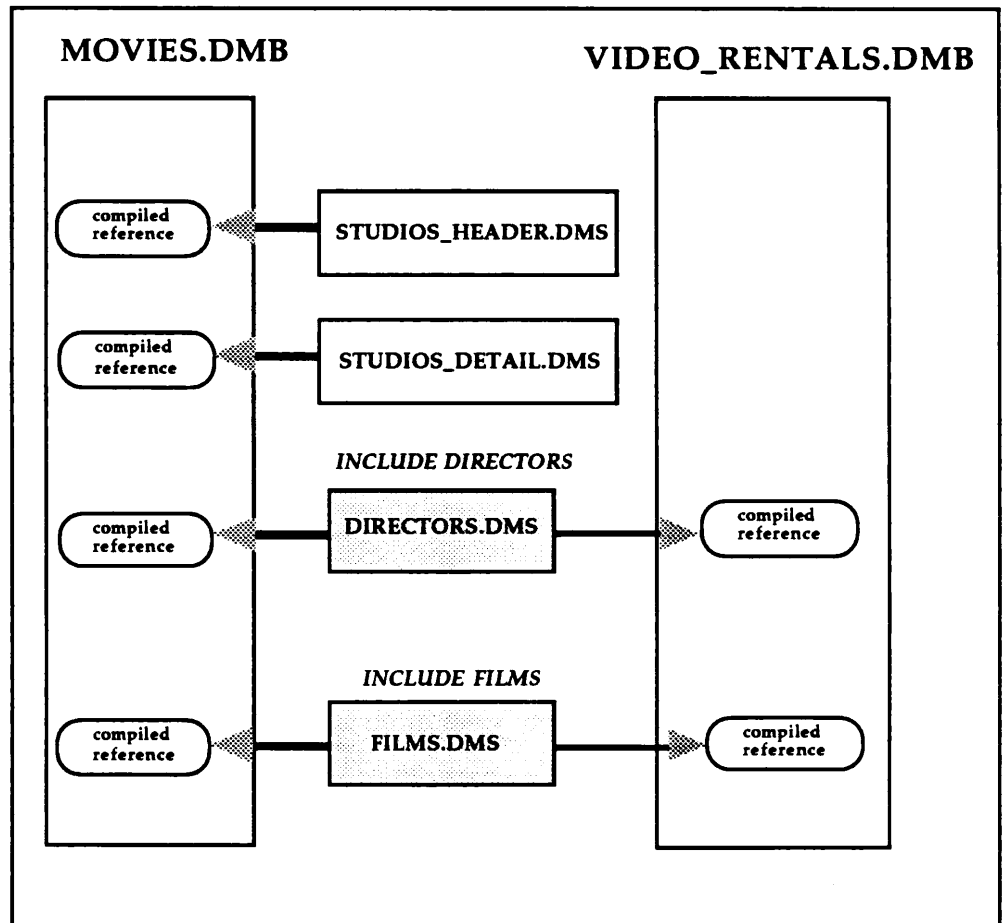
4. End the database definition by typing:

```
DDL_ END_DATABASE
1032>
```

The newly created VIDEO_RENTALS is your current database and the first dataset in the database becomes your current dataset.

After the VIDEO_RENTALS database is defined, it does not actually contain copies of the FILMS and DIRECTORS datasets. Since they were originally defined in the MOVIES database, they remain in that database.

You can define a dataset to be in more than one database. Only one copy of the dataset exists, but you can access it by opening any of the databases that include it. If you already have a dataset open as part of one database, you cannot open it in another database until you close the first (unless you open it AS ALIAS). Figure 1-3 shows independent datasets that are defined in more than one database. Note that while they do not exist within either of the two database files, their compiled references do.



Defining Libraries

A library is a tool repository. You can define any tool in a library that you can in a dataset, including variables, procedures, and forms. Figure 1-4 illustrates the sample tool contents of a library.

System 1032 Library

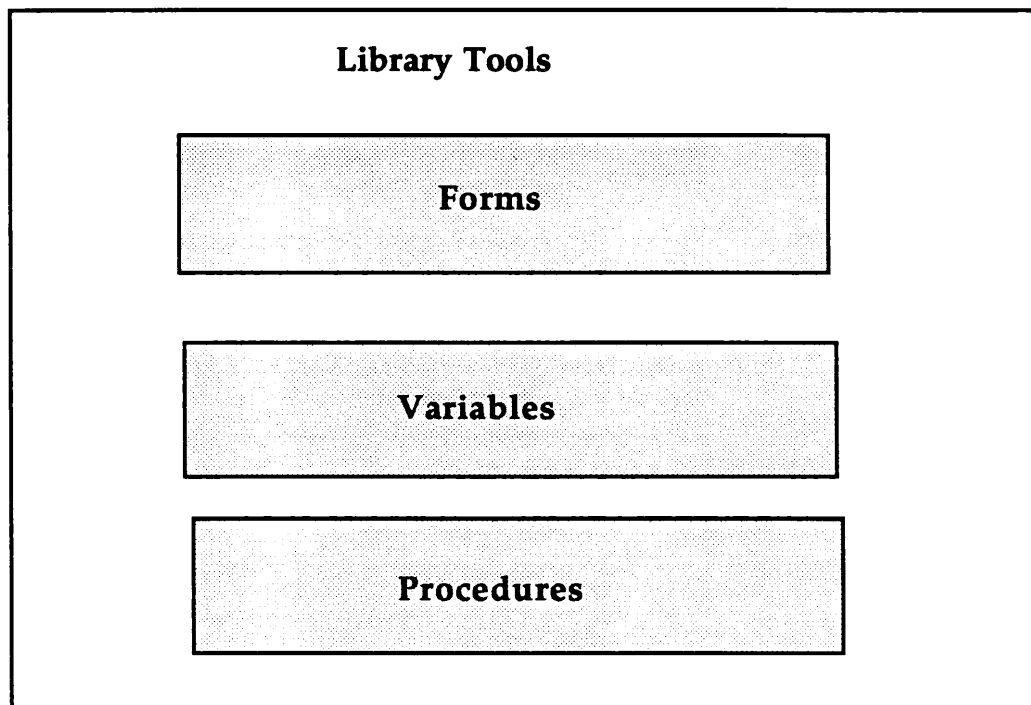


Figure 1-4 Library Tools Within a System 1032 Library

A library cannot be part of a dataset or database. It can reside in its own VMS file or it can be in the same VMS file as datasets, databases, and other libraries.

If you store tools in a library (rather than a dataset or database), it is easier to maintain the dataset or database. It does not include extra tools.

Creating a Library

To create a library, you define the tools to be stored in that library. The following example creates a library that contains a simple procedure.

You can create and compile the procedure before putting it into a library. The following example creates a simple procedure that references the DIRECTORS dataset.

1. When you define a procedure that references a dataset, the dataset must be open when you define the procedure. Open the DIRECTORS dataset.

```
1032> OPEN DATASET DIRECTORS
```

2. Create a file containing the following procedure. Name the file PROCEDURE1.DMC. Within that file, name the procedure PRINT_SURNAME.

```
PROCEDURE PRINT_SURNAME
FIND ALL
PRINT SURNAME
END_PROCEDURE
```

3. Create the library, assigning it the name PRINTNAME. Also, use the OUTPUT option to create an external catalog file named PRINTNAME_LIB. You can also use the OUTPUT option to specify a logical that points to an external file.

```
1032> CREATE LIBRARY PRINTNAME OUTPUT PRINTNAME_LIB
1032 Data Definition Language
DDL>
```

5. Put the procedure in the library, using the @ command and specifying the file name that contains the procedure (in this case PROCEDURE1). Since you gave the file a DMC (command file) extension, you do not need to respecify this extension. Then end the library definition.

```
DDL> @PROCEDURE1
DDL_ END_LIBRARY
```

6. Call the procedure.

```
1032> CALL PRINT_SURNAME
```

```
93 DIRECTORS
records found
Surname
(Last Name)
-----
Allen
Altman
Anderson
.
.
.
Costner
```

Ordinarily, before calling a procedure within a library, you must open the library first. However, since the library was automatically opened when you created it, you do not need to issue an OPEN LIBRARY command.

7. Close the library.

```
1032> CLOSE PRINTNAME
```

Note that, like other System 1032 catalogs, you can also create libraries from description files. For more information about the OUTPUT option, refer to the *System 1032 Programmer's Reference Manual*

Modifying a Library

You can make the following changes to a library:

- **Modify the tools.** Use the MODIFY command to change tool definitions. For example, you can alter the name of a procedure or variable.
- **Modify the library comment.**
- **Update the library.** The INSERT, REMOVE and REPLACE commands allow you to enter, remove, or replace complete definitions without recreating the library.

In the following example, PROCEDURE2 refers to the new procedure contained in the PROCEDURE2.DMD file.

```
1032> OPEN LIBRARY PRINTNAME
1032> REPLACE IN LIBRARY PRINTNAME DESC PROCEDURE2
```

PROCEDURE2 replaces the other procedure defined in the steps listed above. Before you perform a replace, make sure the datasets and libraries are open.

For more information about INSERT, REMOVE, or REPLACE, refer to the *System 1032 Programmer's Reference Manual*.