VAX DSM

digital

DASL Reference Manual

DECLIT
AA
VAX
KM67D

Order Number: AA–KM67D–TE

94-003/049/01

VAX DSM
DASL Reference Manual

AA–KM67D–TE

September 1990

This reference manual describes how to use the DSM Application Software Library (DASL) software.

Revision/Update Information:	This is a revised document.
Operating System and Version:	VMS Version 5.2 and higher
Software Version:	VAX DSM Version 6.0 DASL Version 6.0

Digital Equipment Corporation
Maynard, Massachusetts

76502

First Printing, September 1987
Revised, November 1988
Revised, November 1989
Revised, September 1990

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation.

Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

Any software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license. No responsibility is assumed for the use or reliability of software or equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

Restricted Rights: Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227–7013.

© Digital Equipment Corporation 1987, 1988, 1989, 1990

All Rights Reserved.
Printed in U.S.A.

The Reader's Comments form at the end of the hardcopy version of this document requests your critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation: DASL, DEC, DECwindows, DSM, DSM-11, EDT, PDP-11, VAX, VAX DOCUMENT, VAX DSM, VMS, VT100, VT300, and the DIGITAL Logo.

MUMPS is a trademark of Massachusetts General Hospital. CCSM is a trademark of MGlobal. DTM-PC is a trademark of DataTree Inc.

HOW TO ORDER ADDITIONAL DOCUMENTATION
DIRECT MAIL ORDERS

USA*	CANADA	INTERNATIONAL
Digital Equipment Corporation P.O. Box CS2008 Nashua, New Hampshire 03061	Digital Equipment of Canada Ltd. 100 Herzberg Road Kanata, Ontario K2K 2A6 Attn: Direct Order Desk	Digital Equipment Corporation PSG Business Manager c/o Digital's local subsidiary or approved distributor

In Continental USA, Alaska, and Hawaii call 800–DIGITAL.

In Canada call 800-267-6215.

*Any order from Puerto Rico must be placed with the local Digital subsidiary (809-754-7575).

Internal orders should be placed through the Software Supply Business (SSB), Digital Equipment Corporation, Westminster, Massachusetts 01473.

This document is available in both printed and CDROM versions.

This document was prepared with VAX DOCUMENT, Version 1.2.

Contents

Chapter 2 Screen Driver

Chapter 3 Report Driver

Chapter 4 Query Driver

Chapter 5 Development Environment

Chapter 6 Application Environment

Examples

Figures

Tables

Preface

Intended Audience

The *DASL Reference Manual* is intended for programmers with a working knowledge of the VAX DSM™ language and some knowledge of the VMS™ operating system.

Manual Objectives

The *DASL Reference Manual* describes the elements of the DSM™ Application Software Library (DASL™) software. It presents reference information about the DASL data elements, screens, and commands associated with each DASL module.

The *DASL Reference Manual* contains six chapters and four appendixes.

The six chapters of the *DASL Reference Manual* correspond to the major modules on the DASL Main Menu. These chapters contain the following information:

Chapter 1 **Data Dictionary**

Describes the DASL Data Dictionary Menu. Includes reference sections about Data Dictionary screens and concepts such as data names and cross references.

Chapter 2 **Screen Driver**

Describes the DASL Screen Driver Menu. Includes reference sections about Screen Driver screens and concepts such as data-entry, option, and text screens.

Chapter 3 **Report Driver**

Describes the DASL Report Driver Menu. Includes reference sections about Report Driver screens and concepts such as report collections and format specifications.

Chapter 4 Query Driver

Describes the DASL Query Driver Menu. Explains how to create tables for a relational view of the DASL database and how to define queries to extract information from the tables.

Chapter 5 Development Environment

Describes the DASL Development Environment Menu and its submenus. Explains Development Environment tasks that programmers perform to develop an application, such as:

- Defining groups
- Setting application and site parameters
- Creating a non-English application
- Creating a portable application

Chapter 6 Application Environment

Describes the DASL Application Environment Menu and its submenus, including:

- Security System
- Report Directory
- Message Center
- Event Logging
- Query Database

Appendixes A through D include the following information:

Appendix A ASCII and Multinational Character Sets

Contains tables describing the ASCII Character Set, DEC Multinational Character Set, and the ISO Latin-1 Character Set.

Appendix B Entry Points

Lists entry points to the DASL internal software.

Appendix C Function Keys

Shows DASL function keys and describes the meaning of the keys.

Appendix D SQL Keywords

Shows SQL keywords that are not allowed as column names or table names.

Related Documents

Other documentation about the DASL software consists of:

- *DASL Handbook*

  This document, for first-time users of DASL applications, describes how to use applications that were created with the DASL software. It also describes how to use application environment options such as the Report Directory, the Message Center, and the Query Database Menu.

- *DASL Management Guide*

  This document provides information about DASL system management tasks. It describes how to set up your terminal, initialize the DASL software, and perform database conversions.

- *DASL Pocket Reference*

  This document provides a summary of DASL commands and input conventions.

- *DASL Programmer's Guide*

  This document introduces the DSM Application Software Library (DASL) software and explains how to create a simple application using the DASL software.

- *DASL Version 6.0 Master Index*

  This document contains index entries for manuals in the DASL documentation set.

The DASL manuals are part of the VAX DSM documentation set. This set also includes:

- *Introduction to DSM*

  This document introduces the common syntax and language elements of DSM-11™ (Digital Standard MUMPS™ on the PDP-11™ computer) and VAX DSM (Digital Standard MUMPS layered on the VMS operating system).

- *VAX DSM Database Operations Guide*

  This document describes how to maintain the integrity and reliability of the VAX DSM database. It also describes transaction processing.

- *VAX DSM Callable Routines Reference Manual*

  This document describes the callback interface, which allows routines written in software languages that run layered on the VMS operating system to access a subset of the functionality offered by the VAX DSM language.

- *VAX DSM Installation and Management Guide*

  This document describes how to install and manage VAX DSM systems and explains the internal structure of the VAX DSM database.

- *VAX DSM Language Pocket Reference*

  This document summarizes VAX DSM language elements, the DSM command syntax, and I/O options.

- *VAX DSM Language Reference Manual*

  This document describes the syntax and elements of the VAX DSM language.

- *VAX DSM Programmer's Guide*

  This document describes how to use the programming capabilities of the VAX DSM language.

- *VAX DSM Version 6.0 Master Index*

  This document includes index entries for manuals in the VAX DSM documentation set (excluding the DASL manuals).

- *VAX DSM Version 6.0 Release Notes*

  This document contains enhancement and upgrade information for Version 6.0 of VAX DSM.

Further information about the MUMPS programming language can be obtained from the *ANSI MUMPS Standard*.

The VAX DSM and DASL documentation sets are now available on the VMS Online Documentation Library Compact Disk. These documentation files can be used with the VMS DECwindows™ Bookreader. The Bookreader is an online documentation viewer that is provided with the VMS DECwindows software.

Conventions Used in This Document

The *DASL Reference Manual* uses the following conventions:

Convention	Meaning
italic text	Introduces new terms. Indicates the title of a manual.
bold text	Emphasizes important information. Indicates user input for online documentation.
red text	Indicates user input for hardcopy documentation.
Return	Indicates that you press the key labeled Return.
SP	Indicates that a space must separate components of a command or command line.
{*item*}	Indicates that the enclosed item is optional.
. . .	Indicates that additional parameters can be added to the command line.

Acknowledgment

Digital Standard MUMPS (DSM) is an extension of the ANSI Standard Specification for the Massachusetts General Hospital Utility Multi-Programming System (MUMPS). MUMPS was originally developed at the Laboratory of Computer Science at Massachusetts General Hospital and was supported by grant HS00240 from the National Center for Health Services Research and Development.

Chapter 1
Data Dictionary

This chapter describes the following Data Dictionary Menu options, utilities, and elements:

- Define Data Names Option

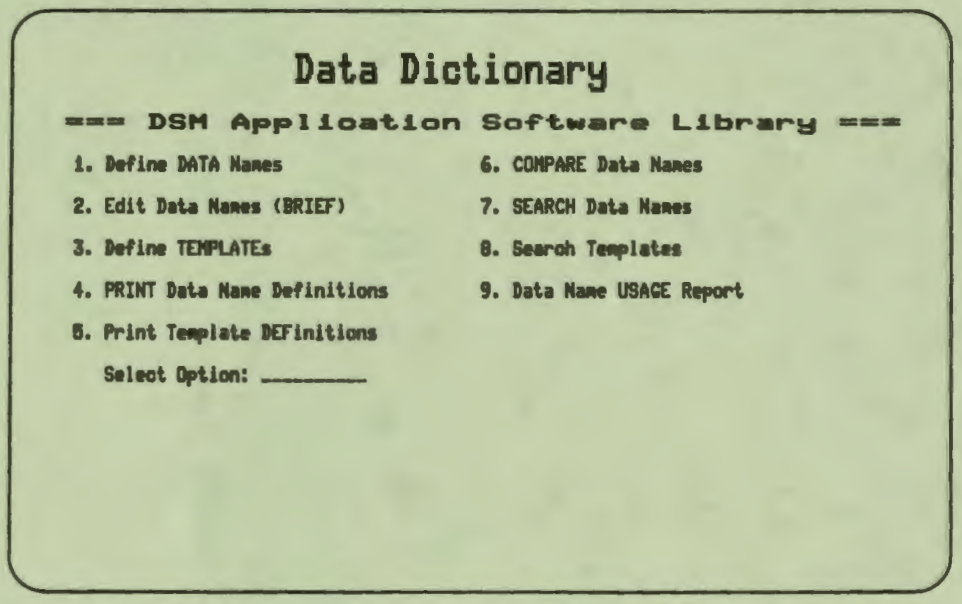
  References
  Piece
  Extract
  Groups
  Pointers
  Data Types
  Data-Name Attributes
  Transforms
  Triggers
  Cross References
  Comments Screen

- Edit Data Names (BRIEF) Option
- Define Templates Option
- Data Dictionary Utilities

  Print Data Name Definitions Option
  Print Template Definitions Option
  Compare Data Names Option
  Search Data Names Option
  Search Templates Option
  Data Name Usage Report Option

Figure 1–1 shows the Data Dictionary Menu and its options.

Figure 1–1 Data Dictionary Menu



This chapter describes each option in detail. See the *DASL Programmer's Guide* for a discussion of principles of database design and an example of how to define a Data Dictionary.

Define Data Names Option

Data names, called Data Dictionary Names or DDNs, contain the information that the DASL software needs to prompt for, validate, store, and retrieve a data item.

Data names are records in a Data Dictionary global; they become VAX DSM local variables at run time.

You use the Define Data Names Option of the Data Dictionary Menu to create data names for all data items you want your applications to reference.

Explanation

You can create data names for the following types of data items:

- Items that you want to store and retrieve later for computation or display. For example, if you are creating a hospital information system, you create DDNs for patient names, patient identification codes, patient addresses, and so forth.
- Prompts that only solicit user input to determine the flow of screen control. You define these prompts in the Data Dictionary to assure standardization across an application. For example, you can create data names for the decision prompts you see at the bottom of DASL screens such as "Save, Edit, or Quit:" or "Continue, Edit, or Quit:".

Information for Defining Data Names

Table 1–1 shows the information you provide to define a data name.

Table 1–1 Information for Data Name Definition

Item	Type of Information	Examples
Data name	A standard DASL local variable name containing a leading alphabetic character followed by up to 6 alphanumeric characters.	EMPNAM ADDRESS1 PHHOME
Description	A free-text description for documentation and lookup purposes. Helps users make accurate selections from lookup lists.	Employee name First line of address Home phone
Reference	A VAX DSM global reference, with or without subscripts, that describes how the system evaluates and files data values entered for this data name. You can enter lowercase alphabetic characters and extended global syntax. The DASL software automatically changes lowercase extended references and data names within the global subscripts to uppercase.	^PAT(*) ^PAT(CODE,"DEMOG") ^PAT(CODE,"EXAM",#) ^[XXX,YYY]PAT(CODE)
Piece	A piece reference and optional delimiter indicate the position of the data name's value within the global reference.	3 3^
Extract	The range of character positions in the global or in the piece that contains the data name value.	3 4–19
Pointer to	An optional field where you enter the primary key data name that the current data name points to. The current data name is a dependent data name in one global that has the same value as a primary key data name in another global and therefore *points to* the other global.	DOCID, where the current data name is PHYSID, and the structure is: ^PT(ID,"D")=PHYSID ⇓ ^DOC(DOCID) with PHYSID as a pointer
Group(s)	The group or groups to which the data name belongs.	HELP, BILLING,SCHEDULE
Active flag	A Y or N that indicates whether the data name is currently active or inactive. Inactive data names cannot be used in an application, and do not appear on lookup lists except at the first field in the Define Data Names screen. You can declare a data name inactive to discontinue its use in an application.	Y, N
Template	The name of a predefined template of default values to assign to the data name.	TDDN, TAME
Data Type	One of the five DASL data types assigned to the data name.	DATE, TIME, NAME, STRING, NUMERIC

Table 1–1 (Cont.) Information for Data Name Definition

Item	Type of Information	Examples
Data Length	The maximum number of characters that can be entered in a field associated with the DDN.	1, 7
Attributes	One or more commands that specify how the DASL software handles input.	FULL, LCASE, LCASEQ, NOECHO, NOFILTER, REQUIRED, SPACEL, SPACES, SPACET
Prompt	The default prompt that users see at fields using this data name.	Save, Edit, or Quit: Address:
Validation	One or more DASL validations that specify how values entered for the data name are to be checked for accuracy.	DATE, PATTERN, LOOKUP
Help Text	A line of not more than 70 characters to display to a user who enters a question mark (?) or presses Help at a field that prompts for values for this data name.	1 alpha and 0 to 6 alphanumeric characters
Help Screen	The name of the help screen or screens to be displayed when a user requests help at any field that uses this data name. You create the help screen using the Screen Driver options before entering its name at this field.	HNAME
Trigger	A routine or screen to call whenever execution of a FILE command modifies the global value of the data name, typically used to establish an audit trail or print out notices of changes that must be made to paper data records.	^AUDIT NCHANGE
Input Transform	VAX DSM code or a call to a VAX DSM routine or %command that changes the value the user enters into an internal format that is stored in the database.	S %INT=%EXT
Output Transform	VAX DSM code or a call to a VAX DSM routine or %command that converts data from the internal format stored in the database into the external format that the user sees.	S %EXT=+$H-%INT\365.25 S:%EXT>0 %EXT=%EXT_" year"_$S(%EXT>1:"s",1:"")

Table 1–1 (Cont.) Information for Data Name Definition

Item	Type of Information	Examples
Cross-Reference	The VAX DSM global node reference below which the DASL software builds cross-reference indexes, used by the Screen Driver LOOKUP command and the Report Driver during collections or data. You can enter lowercase characters for the global name and use extended global syntax in this field.	^X ^X("X") ^X(DDNM)
Type	The type of cross-reference global the DASL software builds. Six types are available.	KWIC, NAME, NUMERIC, STRING, TRAN1, TRAN2
KWIC Stop Ref	A global reference for the KWIC stop reference global that contains stop words for the data name. The DASL software checks the KWIC stop reference global in addition to the Stop Word List defined in the Site Parameters screen of the Development Environment Menu when filing data and performing lookups. You can enter lowercase characters and use extended global syntax in this field.	^STOP ^GLO("STOP")
Primary Key	A data name that points to the primary record for this data name. The primary key usually is a unique subscript in the global reference for the data name you are currently defining.	ID
Cross-Reference Transform	VAX DSM code or a call to a VAX DSM routine or %command that files data in the cross-reference global in internal format for lookups. Cross-reference transforms can create two types of cross-reference globals: TRAN1 globals have a single subscript and TRAN2 globals have two subscripts.	S %INT1=$P(%EXT,"-",3), %INT2=$P(%EXT,"-",1)_$P(%EXT,"-",2)
Legend	An optional line of text to be displayed across the top of lookup lists. The legend labels this data name and any additional data names that you include on lookup lists. Dots mark off 10-column sections, beginning with column 0 of the screen, so that you can align label positions with the column positions you specify for data names. You specify legends and additional DDNs only for key data names or data names with cross references.	Name Birthdate Address

Table 1–1 (Cont.) Information for Data Name Definition

Item	Type of Information	Examples
Additional DDNs	Up to four additional data names to be displayed across each selection line of a lookup screen. An additional DDN must be able to be evaluated at run time. It can be the primary key if a cross reference is used, or it can be dependent upon the item being looked up or upon the primary key.	BIRTH, ADDR1
Columns	The vertical screen position in which each additional data name appears. Columns are numbered 0 through 79. By default, data is left-justified at the specified column position. If you enter R after the number to specify right justification, the data name appears at the column position minus the length of the data name+1.	30, 45R

Using the Define Data Names Option

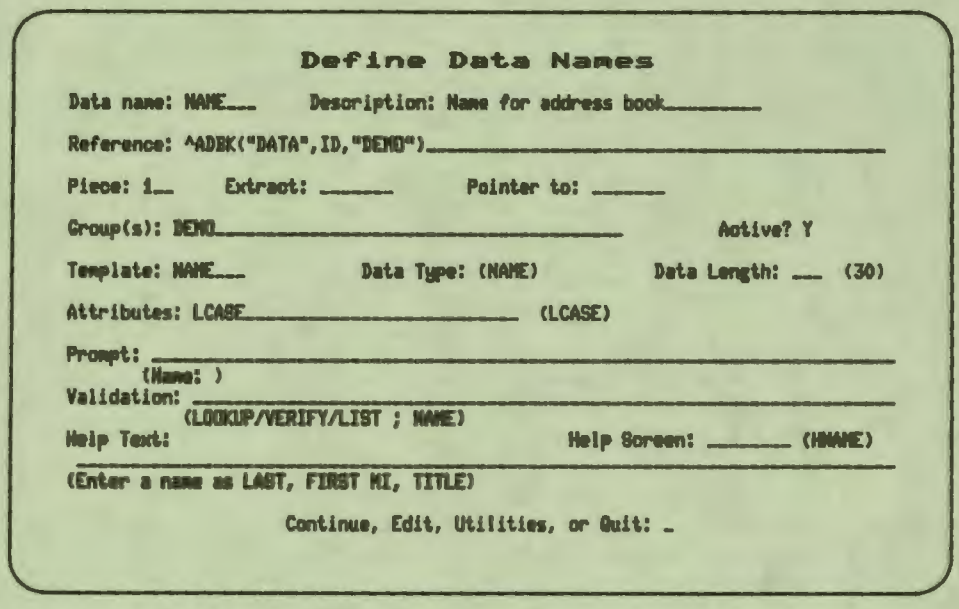
You use the Define Data Names Option to create new data names and to edit, copy, delete, print, or add comments to existing data names.

The Define Data Names Option has two screens. The first contains all required data-name information. The second contains optional information.

Filling In the First Define Data Names Screen

Figure 1–2 shows the first Define Data Names screen.

Figure 1–2 Define Data Names Screen #1



To define a new data name, enter a new name (one letter followed by 0 to 6 alphanumeric characters) after the "Data name:" prompt. Keep the following points in mind when you assign a data name:

- Assign names that describe the data being gathered. For instance, use PATNAM for a data name that contains the full name of a patient. In addition, develop a standard naming convention throughout your application. For example, you can begin all data names for employee information with E: ENAME, ENO, EAD, and so on.
- Do not assign the name DDN or DDNB to a data name because these names are used in the Screen Driver to access the Define Data Names screen and the Define Data Names (BRIEF) screen.

After you assign the data name, you can then fill in the remaining fields.

You do not need to give a global reference in response to the "Reference:" prompt. If you do not specify a global reference, you are not prompted for a piece or extract reference. You are also not prompted for a piece or extract reference if you specify that the data name is a key data name.

If you specify a template for the data name, the DASL software fills in the template values as defaults. These defaults are displayed in parentheses below the field.

Press Return to accept any of the default values, or edit the field to change them. **You cannot change the default data type if you use a template.**

Filling In the Second Define Data Names Screen

After you save the data definitions on the first screen, you see a second screen as shown in Figure 1–3.

Figure 1–3 Define Data Names Screen #2

```
                    Define Data Names

Data name: NAME___     Description: Name for address book_________

Reference: ^ADBK("DATA",ID,"DEMO")_____________________________________

Input Transform:                                  Trigger: _____________
_______________________________________________________________________
()
Output Transform:
_______________________________________________________________________
()
Cross-Reference: ^ADBK("XNAME")_____________          Type: NAME____
KWIC Stop Ref: _____________________          Primary Key: ID_____
Cross-Reference Transform:
_______________________________________________________________________

Legend:  .       .       .       .       .       .       .
          Name                      Birthday     Address_____________

Additional DDNs: BTHDAY_      STREET_      _______      _______
        Columns: 40__         55__         ____         ____
                        Edit, Utilities, or Quit: _
```

All of the fields on the second Define Data Names screen are optional. To fill in any of these fields, enter E after the "Edit, Utilities, or Quit:" prompt. Enter Q to return to the first Define Data Names screen. Enter U to use the utilities associated with data names.

Editing an Existing Data Name

To edit an existing data name, enter the data name at the "Data name:" prompt on the first Define Data Names screen and then enter E at the "Continue, Edit, Utilities, or Quit:" decision prompt. You can change any value in the data name, or you can add new information.

Using the Utilities

You can delete, copy, print, or add documenting comments to any existing data name by entering the data name and choosing the utilities at the "Continue, Edit, Utilities, or Quit:" prompt. The DASL software then displays a utilities decision prompt.

To delete the data name, enter D in response to the utilities decision prompt and verify your choice where prompted.

To copy the data-name values to a new data name, enter C and specify the new data name. You can later edit the new data name to change any of the copied values.

To print the data-name definition, enter P and, where prompted, take the following steps:

1. Specify Y or N to indicate whether to use a batch (detached) process to print the data name.
2. Enter the VMS device or file name to use.

To add comments to document the data name, enter E. You then see a Comments Screen. Use the Comments Screen to specify the creator, editor, and version number of the data name. You can also enter text in a scroll region to document the data name.

Using a Template to Define a Data Name

Many data names share common characteristics. You can create templates that define default values for such data names. Using the Define Templates Option, you can create and maintain a database more efficiently by making entries and changes once to a common template used by multiple data names. Fields that use those data names are automatically updated when you change the template.

Related Sections

Cross References
Data Types
Define Templates Option
Extract
Piece
Pointers
References
Transforms
Triggers
Validations, Chapter 2

References

You can specify a reference when you define a data name through the Define Data Names Option. A reference, also known as a global reference, identifies the DSM global in which the DASL software stores values for the data name.

Explanation

You use global references for:

- Key data names
- Dependent data names
- Scroll region counters

Key Data Names

Key data names are data names whose values are subscripts in the global reference. Key data names are unique and identify the primary data record for a data name.

When you enter a reference for a key data name, use an asterisk (*) in the subscript. Specify only the node in which the key data name appears as a subscript, even if that node has descendants.

You always specify the subscript level at which you are storing the data name, even if there are lower levels of subscripts in the global. That is, the last rightmost subscript in the reference is the subscript level at which the DASL software stores values for the data name.

Consider the data name ID, a primary key data name in the ^ADBK global of the DASL demonstration system. ID is a second-level subscript in the ^ADBK global. The first-level subscript of ^ADBK is "DATA". The reference in the Data Dictionary record for ID is as follows:

```
Reference:    ^ADBK("DATA",*)
```

Dependent Data Names

Dependent data names are data names whose values are stored as data. When you enter a reference for a dependent data name, specify the global and subtree level at which the data name values appear.

You can specify a global with or without subscripts. Any subscripts you specify must be one of the following:

- String literals (for example, "EXAMS")
- Numeric literals (for example, 1)
- Other data names (for example, PATCODE)

For example, the data name NAME is a dependent data name in the ^ADBK global of the DASL demonstration system. NAME contains the value of a name for the address book. The reference in the Data Dictionary for NAME is:

```
Reference:    "ADBK("DATA",ID,"DEMO")
```

For dependent data names, you can also specify a piece or extract reference to identify the data item within the global data field. For example, NAME is defined as piece 1 in the ^ADBK("DATA",ID,"DEMO") global.

Scroll Region Counters

Scroll region counters are a series of sibling global nodes with sequential numeric subscripts from 1 to n that hold data you collect in a scroll region. Use a number sign (#) as the subscript notation in the references for data names you use as scroll region counters.

For example, in the DASL demonstration system, the data name MCNT is the scroll region counter. The reference in the Data Dictionary for MCNT is:

```
Reference:    "MEET(#)
```

Using DSM Extended Global Syntax

You can use DSM extended global syntax to enter a global reference in some DASL screens. See the *VAX DSM Language Reference Manual* for a description of extended global syntax.

Table 1–2 shows the fields that allow extended global syntax.

Table 1–2 DASL Fields that Allow Extended Global Syntax

Field	Screen
Reference	Define Data Names screen #1
KWIC Stop Ref	Define Data Names screen #2
Cross-Reference	Define Data Names screen #2
Cross-Reference Global	Build Cross-Reference Indexes screen

Caution: Use extended syntax sparingly. In most cases, it is preferable for the system manager to translate globals in the translation table.

Using Lowercase Alphabetic Characters

You can use lowercase alphabetic characters to define a global name. The DASL software automatically changes any extended references and data names within a global reference to uppercase characters. Some valid global references include:

```
^ADBK("DATA",ID, "DEMO")

^Adbk("DATA",ID, "DEMO")

^[KLB,DAS]patient(ID)

^[KLB,DAS]PATIENT(ID)
```

Related Sections

Extract
Piece

Piece

A piece reference describes the position of a data item in a global node containing many items. You can also specify the piece delimiter in the piece field of the first Define Data Names screen. The piece you specify must be in the global you entered in response to the "Reference:" prompt.

Format

piece number{delimiter}

where:

piece number	is the number of the piece within the node value
delimiter	is the piece delimiter character

Comments

Keep the following points in mind when you specify piece references for data names:

- The piece reference is identical to the third argument in the VAX DSM $PIECE function. Enter a piece reference if the data name's value is only part of the value of the global reference.
- The piece delimiter is the character you use in a global reference to separate and distinguish each piece. The DASL software supports only single-character delimiters. Any piece delimiter you specify must be limited to one character.
- If you specify no piece delimiter, the default is the system default delimiter.

  You specify the default delimiter in the Application Parameters screen of the Development Environment Menu.
- The DASL software automatically checks data for delimiter characters. If application users enter the piece delimiter or the system delimiter, the DASL software displays an error message and prompts for data again.
- You cannot specify a piece reference for key data names. Key data names are those data names whose global references contain an asterisk (*) as subscript.
- You cannot specify a piece reference for scroll region counters. Scroll region counters are those data names whose global references contain a number sign (#) as subscript.

- The piece specification is optional for all other data names. If a node has only one piece but can be expanded in the future, enter a 1 to force the DASL software to strip out the system delimiter. This ensures that you can later expand the node without concern that the node contains the delimiter.
- Some fields can contain any printable character, including the character defined as the system delimiter. In this case, specify the NOFILTER data-name attribute, and do not piece the data-name record. The NOFILTER attribute suppresses DASL's standard action of checking data for delimiters.
- If you store a value in one data name and then define a second data name that uses input and output transforms to calculate a new value based on the first value, assign the identical piece reference to both data names. For example, the address book data names BTHDAY and AGE share the same piece reference. BTHDAY accepts a date as input; AGE uses input and output transforms to calculate age using the date stored in BTHDAY.

Related Sections

Application Parameters Option, Chapter 5
Data-Name Attributes
Define Data Names Option
Extract
NOFILTER, Chapter 2
References

Examples

In the following example, you specify that the data name you are defining is the third piece in the reference global, and that the node uses the default delimiter.

```
Piece:   3
```

In the following example, you specify that the data name is the fifth piece in the reference global, and that the node uses the comma (,) as a delimiter. Use the explicit piece delimiter when you override the default piece delimiter.

```
Piece:   5,
```

Extract

The extract reference specifies the range of character positions the data item occupies in a global node. The data item in the global node must be of fixed length. You can specify an extract reference at the "Extract:" prompt in the first Define Data Names screen.

Format

start{-end}

where:

start	is a number representing the extract position of a single-character data item or the start of the extract range of a multicharacter, fixed-length data item
-	is a hyphen character
end	is a number representing the end of the extract range of a multicharacter, fixed-length data item

Comments

Keep the following points in mind when you specify extract references for data names:

- The extract reference is identical to the second (and optional third) argument in the VAX DSM $EXTRACT function.
- You cannot specify an extract reference for key data names. Key data names are those data names that contain an asterisk (*) in their Data Dictionary reference entry.
- You cannot specify an extract reference for scroll region counters. Scroll region counters have a number sign (#) in their Data Dictionary reference entry.
- The extract specification is optional for all other data names.
- If you do not specify a piece reference after the "Piece:" prompt, the character positions you specify are character positions in a node of the global you specified at the "Reference:" prompt.

  If you specify a piece reference, the character positions you specify after the "Extract:" prompt are character positions within that piece.

  In the following example, you indicate that the data name occupies the first through the thirtieth characters in the node if no piece reference is specified, or the first through the thirtieth characters in the specified piece of the node.

```
Extract:   1-30
```

Related Sections

Define Data Names Option
Piece
References

Example

Suppose you have a global ^EMPFIL that contains demographic information about employees. Each node in ^EMPFIL contains six data names and has the following structure:

```
^EMPFIL(ID) = NAME;STREET;CITY;STATE_ZIP_PHONE
```

The first three data names have variable length; that is, they can be any length up to the maximum length. The last three data names, however, are always of fixed length. STATE is 2 characters; ZIP is 9 characters; and PHONE is 10 characters. To treat the fixed-length items as a single unit, you can construct piece and extract references as shown in Table 1–3.

Table 1–3 Piece and Extract References

DDN	Piece Reference	Extract Reference
NAME	1	–
STREET	2	–
CITY	3	–
STATE	4	1–2
ZIP	4	3–11
PHONE	4	12–21

As Table 1–3 shows, you do not specify extract references for variable-length data.

Groups

The group is a basic unit in a DASL application. Groups are keywords you assign to data names, tables, screens, and reports. Each DASL group functions as a unit (for example, an application module).

Comments

Keep the following points in mind about groups:

- **You must assign all DASL data names, tables, screens, and reports to one or more groups.**
- Before you assign anything to a group, define that group in the Group Dictionary Option of the Development Environment Menu. For each group, you specify:

  — A name (up to 10 characters, letters, or numbers)

  — A description (up to 32 characters)

  The Group Dictionary Option maintains an activity flag for each defined group under the scroll region Active field. The activity flag is Y or N. When you define a group, the DASL software sets the activity flag to Y. To declare a group obsolete, change the activity flag for the group's entry to N.
- You can assign any data name, screen, table, or report to more than one group. Enter the multiple group names as a list, separated by commas, after the "Group(s):" prompt, as shown in the following example:

  ```
  Group(s):    EMPLYEE,SECRTY,CLINIC
  ```
- You can use group names to document and organize your application. For example, you can give all help text screens the group name HELP, or you can give all application reports the group name REPORT.
- Use group names as wildcards when you compile and print reports about data names, tables, screens, and reports, or when you use certain DASL utilities and routines. When you use a group name, you ensure that you are including all data names, tables, screens, and reports that you have assigned to the group.

  To use a group name as a wildcard, enter the name preceded by the at sign (@). For example, to specify all items with the group name REGIST, enter @REGIST.

Related Sections

Group Dictionary Option, Chapter 5

Pointers

You can access dependent data in one global from a second global by storing the first global's primary key data name as a *pointer* in the second global. You define pointers at the "Pointer to:" prompt of the Define Data Names Option.

Explanation

DASL data names can be:

- Key data names that you use as global subscripts
- Dependent data names that you store as data in a global node

In many cases, you need to access data in one global using data stored in another global. You can make such an access route through pointers. Each pointer is a dependent data name in one global that has the same value as a primary key data name in another global.

You define the pointer relationship when you create the dependent data name through the Define Data Names Option. At the "Pointer to:" prompt, you enter the name of the primary key data name that the current dependent data name points to.

Suppose you are creating a hospital registration application with a global ^PAT, containing basic patient information, and ^DOC, containing a list of physicians with the following node structures:

```
^PAT(ID)="NAME;ADDR;LOC"

^DOC(DOCKEY)="DOCNAME;DOCPHONE;DOCLOC"
```

To correlate patients with their physicians, you can add to ^PAT(ID) a dependent data name DOCPTR, as follows:

```
^PAT(ID)="NAME;ADDR;LOC;DOCPTR"
```

When defining DOCPTR through the Define Data Names Option, you enter DOCKEY after the "Pointer to:" prompt. DOCPTR now points to the primary key data name DOCKEY. After you define these data names, the DASL software can evaluate a dependent data name in ^DOC (such as DOCNAME) from the data name DOCPTR.

Pointer Chains

Using pointers, you can construct complex *pointer chains* that you can use with the EVAL and EVALS actions of the Screen Driver or as sort items and print items in the Report Driver.

Pointer chains can be any length, as long as the logic flow is as follows:

```
dependent data name -> primary key -> dependent data name -> primary key
```

Pointer chains require different syntax and are evaluated differently in the Screen Driver and the Report Driver. The following sections describe how to use pointer chains in the Screen Driver and the Report Driver.

Constructing Pointer Chains in the Screen Driver

In the Screen Driver, you can use a pointer chain with the EVAL and EVALS actions to traverse a chain of data names and retrieve dependent data from records. Pointers can point to fields, data names, and other pointers.

Pointer chains in the Screen Driver take the following syntax:

EVAL SP **namelist{<pointer chain>}{:postcond}**

EVALS{/qualifiers} SP **namelist{<pointer chain>}**

where:

namelist	is a list of one or more field or data names
pointer chain	is a chain of one or more pointers to additional fields, data names, or other pointers
postcond	is a postconditional expression

When specifying pointers with EVAL and EVALS, precede the pointers by a left angle bracket (<). The following statements are examples of valid EVAL and EVALS statements using pointer chains:

```
EVAL Field1,Field2<PADDN1

EVAL (Field1,Field2)<PADDN1

EVAL Field1<PADDN1<PADDN2

EVAL Field1,%DDN,Field2<PADDN1,Field3<PADDN2
```

In the Screen Driver, the DASL software traverses the pointers from right to left. For example, suppose that you have the following globals defined in your database:

```
^PAT(PATID)=NAME;STREET;CITY;ZIP;PHYS1
                                    /
          ------------------------
         /
^PHYS(PHYSID)=PHYSNAME;PHYSADD;HOSP1
                                  /
         ------------------------
        /
^HOSP(HOSPID)=HOSPNAME
```

In this example, the dependent data name PHYS1 in the ^PAT(PATID) global is a pointer to the key data name PHYSID in the ^PHYS(PHYSID) global. The dependent data name HOSP1 in the ^PHYS(PHYSID) global is a pointer to the key data name HOSPID in the ^HOSP(HOSPID) global.

To evaluate the field HOSPNAME (which uses the data name HOSPNAME), you can construct the following pointer chain:

```
EVAL HOSPNAME<HOSP1<PHYS1
```

To access HOSPNAME, the following process occurs in the pointer chain:

1. The pointer data name PHYS1 in the ^PAT(PATID) global points to the key data name PHYSID in the global ^PHYS(PHYSID).
2. Then, the pointer HOSP1 in the ^PHYS(PHYSID) global points to the key data name HOSPID in the ^HOSP(HOSPID) global.
3. Because HOSPNAME is a dependent data name in ^HOSP(HOSPID), you can now access the values for HOSPNAME.

The value of %FND is determined by success or failure in evaluating the first field on the pointer chain list. Failure to retrieve a pointer, a null pointer, or failure to retrieve the dependent node results in %FND=0.

Constructing Pointer Chains in the Report Driver

In the Report Driver, you can construct a pointer chain as follows:

result(pointer1{,pointer2,...,pointerN})

where:

result	is a dependent data name to print
pointer1	is a dependent data name that points to a key data name
pointer2	is a second dependent data name that is pointed to by the pointer data name associated with pointer1 (Pointer 2, in turn, points to a key data name.)
pointerN	is the last dependent data name in the chain (PointerN points to a key data name that contains the desired data as a field in its global node.)

For example, the report PATLIST contains the following collection specifications and sort list based on the globals described in the previous example:

```
For   PATID

Sort by:   NAME, HOSPNAME(PHYS1,HOSP1)
```

In the sort list, the item HOSPNAME(PHYS1,HOSP1) represents a pointer chain. HOSPNAME is a dependent data name that is not stored in the same global as the collection key data name PATID.

The DASL software evaluates the pointer chain using the same logic described in the previous section, "Constructing Pointer Chains in the Screen Driver".

Note that the Report Driver traverses the pointer chain from left to right. In the Report Driver, the DASL software evaluates pointer chains at print time rather than at collection time. In addition, the DASL software reevaluates pointer chains each time they are encountered.

Summary of Pointer Chain Differences

Table 1–4 summarizes the differences between pointer chain syntax in the Report Driver and in the Screen Driver.

Table 1–4 Pointer Chain Syntax

Module	Uses	Example	Evaluation
Screen Driver	EVAL EVALS	EVAL HOSPNAME<HOSP1<PHYS1	Right to left
Report Driver	Sort List Print Item	HOSPNAME(PHYS1,HOSP1)	Left to right

Related Sections

Define Reports Option, Chapter 3
EVAL, Chapter 2
EVALS, Chapter 2
Print Items, Chapter 3
References
Sort Lists, Chapter 3

Examples

Suppose you have the following globals and data names:

```
^PAT(PID)=PATNAME;PATDID
                  /
        ----------
       /
^DOC(DID)=DOCNAME;DOCBID
                  /
        ----------
       /
^BILL(BID,"BLUEX")=BLNAME;BLPER
```

Assume that PID is defined, and that PATDID is a pointer to DID and DOCBID is a pointer to BID. PATNAME, DOCNAME, BLNAME, and BLPER are field names and data names.

In the Screen Driver, you can construct the following pointer chain to evaluate the value of BLNAME:

```
EVAL BLNAME<DOCBID<PATDID
```

In the Report Driver, you can construct the following pointer chain as a print item:

```
BLNAME(PATDID,DOCBID)
```

Data Types

Data types describe the type of input data required. You assign a data type to each template and data name. Each data type gives the template or data name to which it is assigned a default validation or (in some cases) translation logic.

Comments

You assign data types through the Define Templates Option and the Define Data Names Option. Table 1–5 lists the DASL data types and the actions they perform.

Table 1–5 DASL Data Types

Data Type	Action
DATE	Defaults template or data-name validation to DATE. Requires entry of characters in date format. Stores input internally in $HOROLOG format, and displays output in the standard external date format you specified through the Site Parameters Option of the Development Environment Menu.
NAME	Requires entry of characters in name format. Stores input internally, and displays output as uppercase name — LAST, FIRST {MIDDLE, TITLE}. NAME normalizes data by stripping punctuation and changing all data to uppercase characters. When you are entering cross-reference information in a data-name record, NAME causes the cross-reference type to default to NAME.
NUMERIC	Defaults template or data name validation to NUMERIC 1 — numeric data with the lowest value of 1. Requires entry of numeric characters, 0 through 9, +, and −. In the case of the numeric cross-reference type, strips leading zeros to convert to a canonic number: 007 becomes 7.
STRING	Accepts any input except the application delimiter character. Stores input internally, and displays output exactly as entered. The application delimiter is by default a semicolon (;). You define the delimiter character your application uses through the Application Parameters Option.
TIME	Defaults template or data-name validation to TIME. Requires entry of characters in time format. Stores input internally in $HOROLOG format, and displays output in the standard external time format you specified through the Site Parameters Option.

Related Sections

Application Parameters Option, Chapter 5
Canonic Numbers (*VAX DSM Language Reference Manual*)
Cross References
DATE, Chapter 2
Define Data Names Option
NAME, Chapter 2
NUMERIC, Chapter 2
Site Parameters Option, Chapter 5
TIME, Chapter 2
Validations, Chapter 2

Data-Name Attributes

Attributes are commands that you use to define input specifications for values entered for a data name, control how the DASL software displays a screen or field, or add function or control logic to a screen or field.

Explanation

DASL has three types of attributes:

- Data-name
- Screen
- Field

Data-name attributes describe any specifications for values entered for a data name. You specify data-name attributes at the data-name level or the template level through the Define Data Names Option or the Define Templates Option.

Table 1–6 lists DASL data-name attributes and their uses.

Table 1–6 DASL Data-Name Attributes

Attribute	Abbreviation	Use
FULL	FL	Requires that values have the full number of characters specified in the data-name length.
LCASE	LC	Accepts lowercase input.
LCASEQ	LQ	Accepts lowercase input that is enclosed within quotation marks. Converts all other lowercase input to uppercase.
NOECHO	NE	Turns off character echo.
NOFILTER	NOF	Permits entry of the default system delimiter or the data-name delimiter.
REQUIRED	REQ	Requires users to enter nonnull data.
SPACEL	SL	Preserves leading blank spaces and strips trailing blank spaces.
SPACES	SP	Permits leading and trailing blank spaces.
STACET	ST	Preserves trailing blank spaces and strips leading blank spaces.

See the "Screen Attributes" and the "Field Attributes" sections of Chapter 2 for a discussion of screen and field attributes.

Comments

Keep the following points in mind when you use data-name attributes:

- The DASL software displays data-name attributes in the field screen, and does not override these attributes at the field level.
- When you use a data name in a data screen field, the DASL software combines the attributes you give to the data name and the attributes you give to the field to determine the field attributes.

Related Sections

Define Data Names Option
FULL, Chapter 2
LCASE, Chapter 2
LCASEQ, Chapter 2
NOECHO, Chapter 2
NOFILTER, Chapter 2
REQUIRED, Chapter 2
SPACEL, Chapter 2
SPACES, Chapter 2
SPACET, Chapter 2

Transforms

You can use input and output transforms to store data in an internal format, and display data to the user in an external format. A transform can be either a line of VAX DSM code or an application-defined %command. You define transforms through the Define Data Names Option or the Define Templates Option.

Explanation

Any data name that uses an input and output transform causes the DASL software to generate an internal field variable. This field variable contains the external format of the data value for that data name, while the data-name variable contains the internal format.

All transforms must be paired: each input transform must have an output transform.

You can perform transforms only on nonnull data.

Using the Variables %INT and %EXT

The DASL software provides two variables that you can use when you define transforms:

- The %EXT variable represents the external value of data.
- The %INT variable represents the internal value of data.

Input transforms modify data that you store in the database. In an input transform, the DASL software passes the external value of the data in %EXT, and returns the internal value in %INT. When you define an input transform, you must set the %INT variable.

Output transforms modify data that you retrieve from the database for the Screen Driver or the Report Driver. In an output transform, the DASL software passes the internal value of the data in %INT, and returns the external value in %EXT. When you define an output transform, you must set the value of %EXT.

The following example shows the input and output transform for the data name AGE in the DASL demonstration system. Users enter a date of birth in the BTHDAY field (data name BTHDAY); the system then displays current age in the AGE field (data name AGE). The AGE and the BTHDAY data names are stored in the same piece in the global reference. The AGE data name uses an output transform to calculate current age.

```
Input Transform:   S %INT=%EXT

Output Transform:

S %EXT=+$H-%INT\365.25 S:%EXT>0 %EXT=%EXT_"year"_$S(%EXT>1:"S",1:"")
```

DATE and TIME Data Types and Transforms

Transforms take precedence over standard date and time conversions associated with DATE and TIME data types. In normal operation, the DASL software automatically converts DATE and TIME data names. If transforms for date and time are used, you must provide valid internal and external formats. If you pass invalid data to the DASL date and time utility ^%DAUDTTM, the DASL software returns null values, resulting in null values being displayed in the field and stored in the database.

Transforms to TIME data types modify the variables %TMI (internal time) and %TMX (external time). Transforms to DATE data types modify the variables %DTI (internal date) and %DTX (external date).

Screen Driver Actions with Transforms

Several Screen Driver actions can affect transforms. This section contains examples of programming logic when you use transforms with the following actions:

- ASSIGN field=expr
- ASSIGN/INTERNAL field=expr
- DEFAULT expr
- ERASE field
- EVAL field
- FILE field
- LOOKUP
- RESET field

ASSIGN field=expr

In this example of DASL programming logic, EXPR must be data in external format. The DASL software stores EXPR in the field variable.

```
IF (data name uses transforms) THEN
        BEGIN
          Store EXPR in the field variable
          Perform input transform on EXPR
          Store result of input transform in the data-name variable
          Display external value (from field variable)
        END
ELSE
IF (data name is DATE or TIME) THEN
        BEGIN
          Store EXPR in the field variable
          Perform date and time conversion on EXPR
          Store result (internal value) in the data-name variable
          Display external value (from field variable)
        END
ELSE
        BEGIN
          Store EXPR in the data-name variable
          Display the value in the data-name variable
        END
```

ASSIGN/INTERNAL field=expr and DEFAULT expr

In the case of ASSIGN/INTERNAL field=expr, the data in EXPR must be in internal format, and EXPR is stored in the data-name variable. DEFAULT expr is implemented using the same logic, as follows:

```
IF (data name uses transforms) THEN
         BEGIN
           Perform output transform on EXPR
           Store result (external value) in the field variable
           Display external value (from field variable)
         END
IF (data name is a date or time) THEN
         BEGIN
           Perform date and time conversion on EXPR
           Store result (external value) in the field variable
           Display external value (from the field variable)
         END
ELSE
         Display the data-name variable
```

ERASE field

ERASE erases a field from a screen, and returns the field's data value to its default state. If a field uses the DEFAULT attribute, then ERASE uses the same logic as DEFAULT expr.

EVAL field

The DASL software executes output transforms when it performs the EVAL, that is, when it retrieves data from the database and stores it in the data-name variable. EVAL field uses the following logic:

```
IF (data name uses transforms) THEN
        BEGIN
          Perform output transform on internal value
          Store external value in field variable
          Display external value (from field variable)
        END
ELSE
IF (data name is a DATE or TIME) THEN
        BEGIN
          Perform date and time conversion on internal value
          Store result (external value) in the field variable
          Display external value (from the field variable)
        END
ELSE
          Display data-name variable
```

FILE field

The FILE action always stores values from the data-name variable in the database. If the data name uses transforms, the data-name variable contains the internal value.

LOOKUP

The DASL software always displays data names in external format in a lookup list. If the data name uses transforms, the DASL software performs the output transform during execution of the LOOKUP.

RESET field

RESET returns a field to its default value. If a field uses the DEFAULT attribute, then RESET uses the same logic as DEFAULT expr.

Screen Driver Postvalidation Processing and Transforms

The DASL software stores user input in the variable %RES and performs validations on the values in %RES. After a validation succeeds, the DASL software stores the postvalidation value in an internal field variable. The DASL software performs an input transform on this value, and stores the result in the data-name variable.

A validation can modify user input, serving as a transform that modifies the displayed value. For example, T, which stands for today's date, is valid input to a date field, yet the DATE validation converts T to today's actual date in the special variable %RES. The DASL software then displays the external value (%RES) for the date in the field.

Report Driver Print Items and Transforms

The DASL software always displays data names used as print items in external format. If you use transforms, the DASL software performs the output transform during the print phase of the report. The data-name variable contains the internal value.

Related Sections

Actions, Chapter 2
ASSIGN, Chapter 2
Data Types
DATE, Chapter 2
DEFAULT, Chapter 2
Define Reports Option, Chapter 3
ERASE, Chapter 2
EVAL, Chapter 2
FILE, Chapter 2
LOOKUP, Chapter 2
RESET, Chapter 2
TIME, Chapter 2
Validations, Chapter 2

Triggers

You specify triggers when you define data names through the Define Data Names Option. A trigger is the name of a routine or screen to call whenever a FILE action takes place that changes an existing value of the data name.

Explanation

You can use triggers to keep track of database changes. For example, you can use a trigger to call a report that notifies you when changes are made on any of the data names that make up an employee global (name, address, employee identification, and so forth). The DASL software then prints that change report whenever any user modifies the values in any of those employee data names.

You can also use a trigger to implement an audit trail. Specify a trigger to record user information related to data that is being modified.

Related Sections

Define Data Names Option
Define Data Screens Option, Chapter 2
FILE, Chapter 2

Cross References

You can use cross-reference globals to look up information that is stored as global data in dependent data names.

Explanation

Normally, you can access values for a data name only through the primary key for that data name. Suppose you are building a patient record global, ^PAT("DATA",ID), with a patient identification number, ID, as the primary key. Then, you use the patient identification number to perform LOOKUP validations, actions, or report collections on any of the data names in each ^PAT node; that is, you specify the right value of ID to access ^PAT("DATA",ID).

Often, however, you need to access data by specifying a dependent data value found in the node. For example, you need to access (through report collections) the information in a patient record in ^PAT("DATA",ID) by any of the following dependent data names:

- Patient name, dependent data name PNAME
- Patient Social Security number, dependent data name SSN

You can access these dependent data names through cross-reference globals.

Creating Cross-Reference Globals

You can create a cross-reference global for a dependent data name when you create the Data Dictionary entry for the data name. You can specify the following items on the second screen of the Define Data Names Option:

- Cross-reference name (global reference)
- Cross-reference type
- KWIC stop reference
- Primary key
- Cross-reference transform

The DASL software creates the cross-reference global when the data name or field is filed using the FILE screen action.

Cross-Reference Name and Primary Key

For each dependent data name you want to access directly, specify a root node for a cross-reference global.

The cross-reference name is a full VAX DSM global node reference for the cross-reference global. The cross-reference name can be a global or a subtree of a global. The DASL software creates the cross-reference entries as descendants of the node you specify.

Each cross-reference global takes the general form:

^name(sub1,{sub2,}key)=""

where:

name is the name of the global that contains the cross reference. Can be a global name or a subtree of the main global.

sub1 is the first subscript of the cross-reference global. Sub1 can be a string literal representing the cross-reference subtree of a main global, or a dependent data name.

sub2 is the second subscript of the cross-reference global. Sub2 is a dependent data name.

key is a primary key data name.

For example, you can build one of two different cross references for the dependent data names PNAME and SSN.

The reference for the main global that contains PNAME and SSN is:

```
^PAT("DATA",ID)=PNAME;SSN
```

You can create two cross-reference globals to access PNAME or SSN as follows:

```
^PX(LAST,FIRST,ID)

^SX(SSN,ID)
```

Alternatively, you can define a subtree of the ^PAT global as a cross reference as follows:

```
^PAT("XPN",LAST,FIRST,ID)

^PAT("XSSN",SSN,ID)
```

Note that you can only define cross-reference globals for dependent data names that are associated with primary keys. You must specify the primary key data name for the dependent data name at the "Primary Key:" prompt.

The primary key subscript in the cross-reference global allows you to access any dependent data names that are stored in the main global that contains that primary key. For example, when you look up PNAME, you can also access the values for SSN in ^PAT("DATA",ID).

Cross-Reference Type

The cross-reference type determines the structure of the cross-reference global. The cross-reference type indicates what type of data you enter in a field associated with the data name.

Table 1–7 shows cross-reference global types and the types of data associated with each global type.

Table 1–7 Cross-Reference Types

Type	Description
KWIC	String data that you access by any word in the string.
NAME	Standard DASL names.
NUMERIC	VAX DSM number in canonic numeric form.
STRING	String literals.
TRAN1	Cross-reference transform, creates a cross-reference global with a single subscript.
TRAN2	Cross-reference transform, creates a cross-reference global with two subscripts.

The following sections describe each cross-reference type.

KWIC Type

In a KWIC (Key Word in Context) type cross-reference global, values for the dependent data name are string data that you access by any word in the string. For example, with a KWIC cross-reference global, you can access the book title *Decline and Fall of the Roman Empire* by entering any one or combination of the following:

- Decline
- Fall
- Roman
- Empire

When storing a data value in a KWIC cross-reference global, the DASL software strips punctuation and converts the remainder to uppercase characters. The DASL software allows numbers as input, and checks that the length of data is at least two characters. Then, the DASL software checks the value against a Stop Word List that you define in the Site Parameters Option of the Development Environment Menu. Using the Stop Word List, the DASL software removes all items on the list, such as common articles and prepositions (words such as the, and, and of).

The DASL software then files the remaining words individually in the cross-reference global.

KWIC Stop Reference Global

To specify additional stop words for a data name, other than the words that appear in the Stop Word List in the Site Parameters screen, you can define a KWIC stop reference global for that data name. For example, you can build a KWIC stop reference global, such as ^GLO("WORD"), containing common medical terminology for a data name that is a medical diagnosis. First, you define a KWIC stop reference global that is a subscript of the KWIC cross-reference global. Then, you build the KWIC stop reference global to list the desired medical stop words. Alternately, you can create a screen in which users can specify stop words to build the KWIC stop reference global.

When you use a Stop Word List and a KWIC stop reference global, the DASL software first checks the Stop Word List when filing data and performing lookups. Then, the DASL software checks the KWIC stop reference global for the data name.

If you make any additions or deletions to the KWIC stop reference global, you can rebuild the KWIC cross-reference global for that data name. To rebuild the cross-reference global, you can:

- Use the Build Cross-Reference Indexes Option of the Development Environment Menu to generate code to rebuild the cross-reference global.
- Call a routine to rebuild the cross-reference global from the screen in which you build the KWIC stop reference global.

If you do not use any utilities or routines to maintain the cross-reference global, the cross-reference global becomes self-correcting as data is filed.

NAME Type

The NAME cross-reference type indicates that the values for the dependent data name are DASL names. When storing a data value in a NAME cross-reference global, the DASL software strips the input value of nonalphabetic characters, and converts all characters to uppercase. The DASL software then files the resulting first and last name in a way that permits access by any of the following:

- Last name or partial last name only
- Last name and first name
- Partial last name and first name

Last name is always required for this cross-reference type. The DASL software performs an implied NAME validation before each lookup.

NUMERIC Type

The NUMERIC cross-reference type specifies that the values for the dependent data name are VAX DSM numbers. When storing a data value in a NUMERIC-type cross-reference global, the DASL software converts the input value to canonic numeric form by applying a Unary PLUS operator. This ensures that all cross-reference global nodes collate in standard VAX DSM collating sequence (in this case, numbers before alphanumeric strings).

A numeric cross-reference global can contain string data. For example, you strip leading zeros in string data: you can look up the value 007 by entering 7. The only difference between a numeric cross-reference type and a string cross-reference type is the Unary PLUS operation that the DASL software performs with the numeric cross-reference type.

For more information about Unary PLUS and collating sequence, see the *VAX DSM Language Reference Manual*.

STRING Type

The STRING cross-reference type indicates that the values for the dependent data name are string literals. When storing a data value in a STRING cross-reference global, the DASL software converts all alphabetic characters to uppercase and strips leading and trailing spaces.

Cross-Reference Transforms: TRAN1 and TRAN2 Types

The DASL software uses cross-reference transforms to create cross-reference globals that store user input in internal format for lookups. When you specify a TRAN1 or a TRAN2 cross-reference type, you must also specify a cross-reference transform for that data name. The DASL software uses the cross-reference transform to file user input in internal format, as follows:

- A TRAN1 cross-reference type creates a cross-reference global with a single subscript. The DASL software passes the external value of the data in the variable %EXT and returns the internal formatted data in the variable %INT.
- A TRAN2 cross-reference type creates a cross-reference global with two subscripts. The DASL software passes the external value of the data in the variable %EXT and returns the internal formatted data in the variables %INT1 and %INT2. TRAN2 cross-reference types are useful for data such as names or coding systems that use the decimal character.

You can use cross-reference transforms to determine data output in lookup lists and reports. Using cross-reference transforms, you can:

- Specify a non-standard collating order for data names used in report output and lookup lists. This function is especially useful when you are creating a non-English application. You can use a cross-reference transform to create a collating order that considers language-specific punctuation and collating practices.
- Create a soundex type lookup list. You can use cross-reference transforms to file letters in numeric order according to their phonetic similarity. In a soundex type cross-reference global, for example, the letter f and the phrase ph can have the same numeric value.

When storing a data value in a TRAN1 or TRAN2 type cross-reference global, the DASL software converts all characters to uppercase.

Related Sections

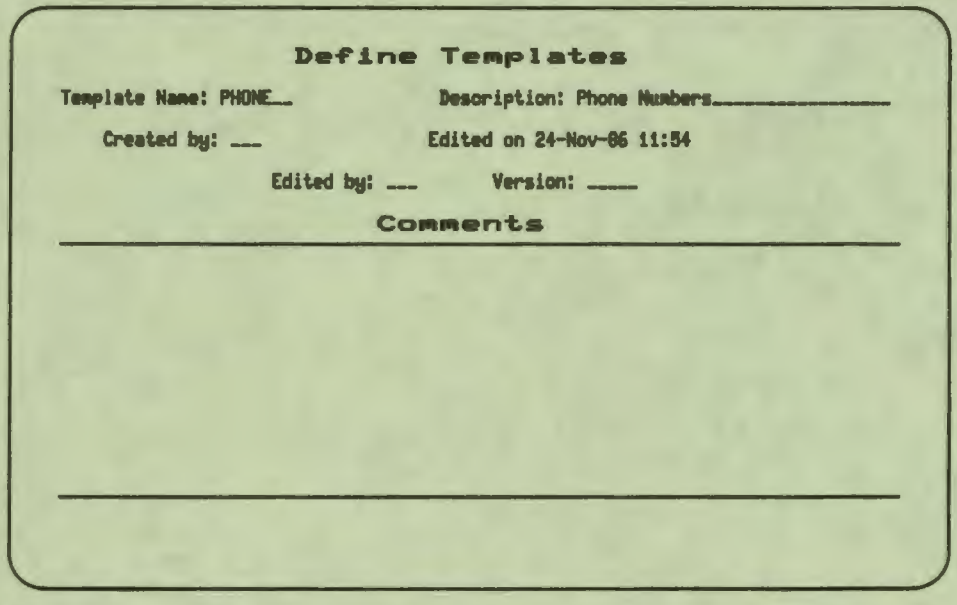
Define Data Names Option
LOOKUP, Chapter 2
References

Comments Screen

You use the Comments Screen to document your application. You can access the Comments Screen whenever you edit an existing template, data name, screen, table, query, or report from any of the DASL options where you define these elements.

Figure 1–4 shows a sample Comments Screen for the PHONE template.

Figure 1–4 Comments Screen



Comments

Keep the following points in mind when you use the Comments Screen:

- You can access the Comments Screen by choosing Utilities at the decision prompt while editing a template, data name, screen, table, query, or report. When the option displays the utilities prompt, specify E to edit comments.
- The first time you enter comments for a DASL component, the DASL software displays the current date and time as the creation and editing date and time. You can specify the creator of the component and the following:
  - — Editor (defaulted to the creator)
  - — Current version number (a decimal number in the form n.n)
  - — Any number of lines of comments

- Whenever you edit comments, the DASL software displays the current date and time as the date and time of the latest edit. You can now change the editor, the current version number, or any of the comment lines. You can also add new comment lines and delete existing lines.

Related Sections

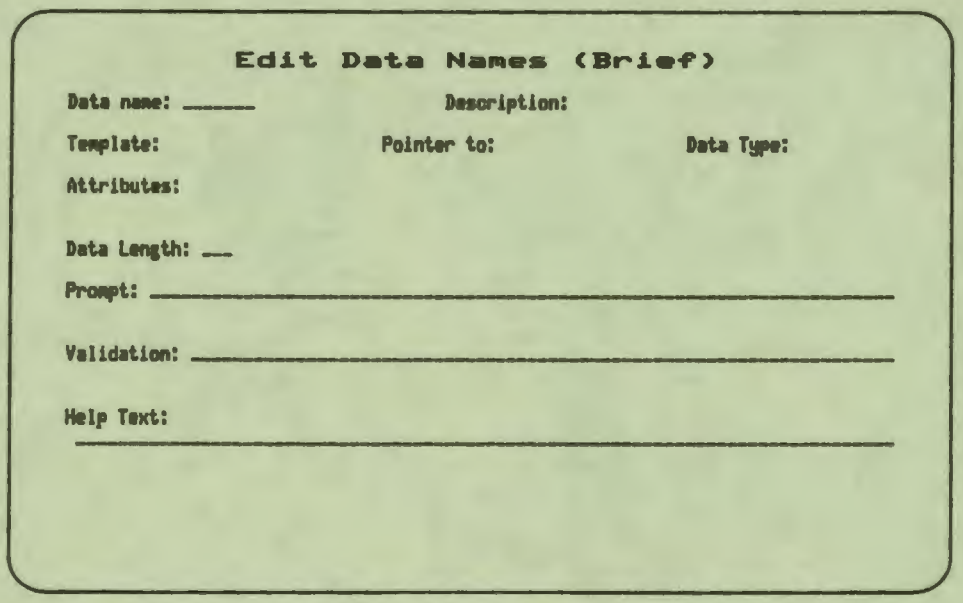
Define Data Names Option
Define Data Screens Option, Chapter 2
Define Option Screens Option, Chapter 2
Define Query Option, Chapter 4
Define Reports Option, Chapter 3
Define System Queries Option, Chapter 4
Define Tables Option, Chapter 4
Define Templates Option
Define Text Screens Option, Chapter 2

Edit Data Names (BRIEF) Option

You use the Edit Data Names (BRIEF) Option to change some of the values in an existing data name.

Figure 1–5 shows a sample Edit Data Names (Brief) screen.

Figure 1–5 Edit Data Names (Brief) Screen



Explanation

When you choose the Edit Data Names (BRIEF) Option, you specify an existing data name. You can enter an asterisk (*) at the "Data name:" prompt to view a lookup list of data names. After you select an item from the lookup list or enter an existing data name, the DASL software displays the values for the data name you select in the appropriate field. Any values defaulted from the template are displayed below the field in parentheses.

You can use the Edit Data Names (BRIEF) Option to change the data length, prompt, validation, and help text.

Related Sections

Define Data Names Option
Define Templates Option

Define Templates Option

Templates are simplified data name records. After you create a template, you can use that template as a starting point for creating data names in your applications.

When you create a data name record, through the Define Data Names Option, you can specify a template to use as the model for the data name. The DASL software then displays the values in that template as the default values of the data name you are creating. You can accept the default template values, or change any of them. Any template values you accept become values of the data name.

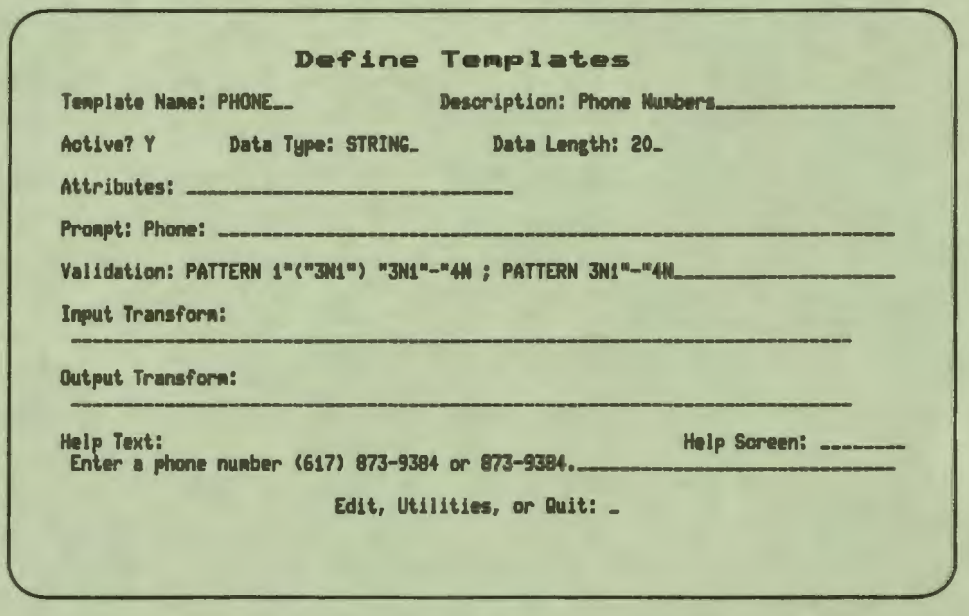
Many data names can share the same template default specifications. For example, you can define templates for data names whose values are phone numbers, names, postal codes, or currency.

Define Templates Option Screen

You use the Define Templates Option to create new templates and edit, copy, delete, print, or add comments to existing templates. When you select the Define Templates Option, the DASL software displays the Define Templates Option screen.

Figure 1–6 shows the Define Templates Option screen.

Figure 1–6 Define Templates Option Screen



Defining a New Template

As shown in Figure 1–6, each template record can consist of several data name specifications. See Table 1–1, Information for Data Name Definition, for a full description of these specifications.

To create a new template, enter a new name (one letter followed by 0 to 6 alphanumeric characters) after the "Template Name:" prompt. You can then fill in template information. Only the following data is required:

- Description
- Data type

Editing an Existing Template

To edit an an existing template, enter the template name after the "Template Name:" prompt and enter E after the "Edit, Utilities, or Quit:" decision prompt. **You can change any value in the template definition except the template name.**

Using the Utilities

You can delete, copy, print, or add documenting comments to any existing template by entering the template name and choosing utilities at the "Edit, Utilities, or Quit:" decision prompt.

To delete the template, enter D when the DASL software displays the utilities prompt, and verify your choice where prompted.

To copy the template values to a new template, enter C and specify the name of the new template. You can later edit that new template definition to change any of the copied values.

To print the template, enter P and, where prompted, take the following steps:

1. Enter Y or N to indicate whether to use a batch (detached) process to print the template.
2. Enter the VMS device or file name to use.

To add comments documenting a template, enter E. You then see a Comments Screen that you can use to specify the creator, editor, and version number of the template, and to enter text documenting the template in a scroll region.

Related Sections

Comments Screen
Define Data Names Option
Print Template Definitions Option

Example

The following example shows a sample template definition for NAME type data names. The NAME template provides basic default information on name length, help text, and validation.

```
Template Name: NAME                Description: Standard Name format
Created by:        Edited by:  on 6-Mar-89 10:07 AM        Version:
Active? Y          Data Type: NAME          Data Length: 30
Prompt: "Name: "
Attributes: LCASE
Validation: LOOKUP/VERIFY/LIST ; NAME
Help Text: Enter a name as LAST, FIRST MI, TITLE   Help Screen: HNAME
```

You can use such a template as the basis for all data names that solicit personal names (of employees, patients, doctors, and so forth). By doing so, and using the template-specified defaults, you can ensure consistency in default items such as data length, prompt wording, and help text.

Data Dictionary Utilities

The Data Dictionary Menu includes several options that are user utilities. You can use these options to search data names and templates, compare data names, or run various reports about data names and templates you have defined. These options include:

- Print Data Name Definitions Option
- Print Template Definitions Option
- Compare Data Names Option
- Search Data Names Option
- Search Templates Option
- Data Name Usage Report Option

The following sections of this chapter describe the reports produced and any special directions for each option.

Specifying Items in Utility Scroll Regions

When you use some Data Dictionary utilities, the DASL software displays a scroll region so that you can specify the items to search or include in a report.

List the items you want by entering one or more of the following:

- Data names or templates
- An asterisk (*) for all data names or templates
- A partial name preceded or followed by an asterisk (*)
- An at sign (@) followed by the name of a defined DASL group for all items in that group
- A range of items in the form name1-name2

If you choose a range in the form name1-name2, the DASL software includes all items that fall within the specified range.

If you specify one or more ranges, you can verify the items in the range. When the DASL software displays the "Continue, eXpand wildcards, Edit, or Quit:" prompt, enter X. The DASL software then displays all items that fall within the range.

To exclude any of the listed items, enter D in response to the "Continue, Delete, or Quit:" prompt that the DASL software displays after expanding the list. You then move back to the top of the scroll region. You can use the scroll region editing keys to change any of the entries.

The other choices at this prompt are:

- Enter C to continue to the device selection screen.
- Enter E to edit the list of screens.
- Enter Q to start over again.

Choosing the Destination

When you finish choosing items, the DASL software displays the device selection screen. You can use the device selection screen to produce reports through a batch process or in your current process. The device selection screen also prompts you to select a device where you want to print the report.

If you use your current process, you can display the report on your terminal, print it on any VMS device, or write the report to a file.

Related Sections

Define Data Names Option
Define Templates Option
Device Specifications, Chapter 6

Print Data Name Definitions Option

You use the Print Data Name Definitions Option to list the contents of any data names you specify.

Example 1–1 shows a sample Data Name Definitions Report. The Data Name Definitions Report lists all values entered for the data names you choose to list. This sample shows the definition for the data name CITY which is used in the DASL demonstration system.

Example 1–1 Data Name Definitions Report

```
15-Apr-88 11:45 AM                    DASL                              Page 1
                        DSM Application Software Library
                              Data Name Definitions

-------------------------------------------------------------------------------

Data Name: CITY        Description: City for address book      Active? Y
Created by:        Edited by:          on 16-Feb-88  2:45 PM   Version:
Reference: ^ADBK("DATA",ID,"DEMO")                  Piece: 3
Group(s): DEMO
                            Data Type: STRING              Data Length: 20
Attributes: LCASE
Prompt: "City: "
Validation: PATTERN 1U.E
Help text: "Enter the City."

-------------------------------------------------------------------------------
```

Explanation

When you choose the Print Data Name Definitions Option, the DASL software displays a decision prompt. You can specify whether to list data-name definitions by:

- Global reference
- Data name

Listing by Global Reference

To list all data names associated with specific globals, enter G (for global reference) after the decision prompt. Then, list the globals in the displayed scroll region. The DASL software then compiles definitions of all data names associated with all the globals you specify.

You can enter the full global reference as defined in the Define Data Names Option, or you can enter a partial global reference with a wildcard character (*).

Listing by Data Name

To list individual data names regardless of globals, enter D (for data names) after the decision prompt. Then, list the data names in the displayed scroll region.

Related Sections

Compare Data Names Option
Data Dictionary Utilities
Define Data Names Option

Print Template Definitions Option

You use the Print Template Definitions Option to list the contents of any templates you specify.

Example 1–2 shows a sample Template Definitions Report for the PHONE template used in the DASL demonstration system. The Template Definitions Report lists all the values for the templates you specify.

Example 1–2 Template Definitions Report

```
15-Apr-88 11:46 AM                DASL                          Page 1
                      DSM Application Software Library
                           Template Definitions

-------------------------------------------------------------------------

Template Name: PHONE                  Description: Phone Numbers
Created by:          Edited by:  on 24-Nov-86 11:54 AM       Version:
Active? Y            Data Type: STRING        Data Length: 20
Prompt: "Phone: "
Validation: PATTERN 1"("3N1") "3N1"-"4N ; PATTERN 3N1"-"4N
Help Text: Enter a phone number (617) 873-9384 or 873-9384.

-------------------------------------------------------------------------
```

Explanation

When you choose the option, the DASL software displays a scroll region. You then specify templates by entering them in the scroll region.

Related Sections

Data Dictionary Utilities
Define Templates Option

Compare Data Names Option

You use the Compare Data Names Option to list the values in two data names you specify.

Example 1–3 shows a sample Data Names Comparison Report. The Data Names Comparison Report lists all values in the two data names vertically in parallel columns. The values listed can also include any comments (entered through the Comments Screen) associated with the data names.

Example 1–3 Data Names Comparison Report

```
  9-May-88   4:16 PM              DASL                          Page 1
                       DSM Application Software Library
                             Data Name Comparison

Compare DDN: AGE                          With DDN: BTHDAY

Description:                              Description:
  Age                                       Birthday for address book
Data Type: STRING                         Data Type: DATE
Created By: DBS                           Created By:
Edited By: KB                             Edited By:
Edit Date: 4-Aug-88                       Edit Date: 25-Aug-88
Edit Time:  3:25 PM                       Edit Time:  9:51 AM
Prompt:                                   Prompt:
  Age:                                      Birthday:
Help Text:                                Help Text:
                                            Enter a date in the format:
                                            2-Jun-86, 6/2/86, T for today,
                                            or T +/- number.
Validation:                               Validation:
                                            DATE
Input transform:                          Input transform:
  S %INT=%EXT
Output transform:                         Output transform:
  S %EXT=+$H-%INT\365.25 S:%EXT>0 %EX
  T=%EXT_" year"_$S(%EXT>1:"s",1:"")
```

Explanation

When you choose the Compare Data Names Option, the DASL software displays a screen where you enter the two data names to compare.

Related Sections

Data Dictionary Utilities
Define Data Names Option
Print Data Name Definitions Option

Search Data Names Option

You use the Search Data Names Option to run a report listing all occurrences of a specified string in specified data names.

Example 1–4 shows a sample Search Data Names Report created through the Search Data Names Option.

Example 1–4 Search Data Names Report

```
15-Apr-88 11:48 AM              DASL                          Page 1
                    DSM Application Software Library
                            Search Data Names

Searching for: "city"

------------------------------------------------------------------

                  Data Name: CITY  City for address book
Help Text: Enter the City.
Prompt: City:

------------------------------------------------------------------
```

Explanation

When you choose the Search Data Names Option, the DASL software displays a screen where you specify the strings to search for. Then, the DASL software displays a second screen where you specify which data names to search. You can specify multiple search strings and multiple data names in the scroll regions provided in these screens.

Related Sections

Data Dictionary Utilities
Define Data Names Option

Search Templates Option

You use the Search Templates Option to run a report listing all occurrences of a specified string in specified data names.

Example 1–5 shows a sample Search Templates Report created through the Search Templates Option.

Example 1–5 Search Templates Report

```
15-Apr-88 11:49 AM              DASL                          Page 1
                      DSM Application Software Library
                              Search Templates

Searching for: "name"

-------------------------------------------------------------------

                  Template: NAME  Standard Name format
Help Text: Enter a name as LAST, FIRST MI, TITLE or * for a list.
Prompt: Name:

-------------------------------------------------------------------
```

Explanation

When you choose the Search Templates Option, the DASL software displays a screen where you specify the strings to search for. Then, the DASL software displays a second screen where you specify which templates to search. You can specify multiple search strings and multiple templates in the scroll regions provided in these screens.

Related Sections

Data Dictionary Utilities
Define Data Names Option

Data Name Usage Report Option

You use the Data Name Usage Report Option to print a report listing the use of data names you specify in other data names, screens, reports, or database tables.

Example 1–6 shows a sample Data Name Usage Report. The Data Name Usage Report lists all occurrences of the data name NAME in the DASL demonstration system in the reports ADBKALL and ADBKONE, in the screen ADBOOK, and in the NAMES table.

Example 1–6 Data Name Usage Report

```
    29-Jan-90  3:11 PM                DASL                              Page
                        DSM Application Software Library
                             Data Name Usage Report

    Searching for:   NAME

Components     Name           Data Name      Field/Level     Sub-field/Line
                                                            Column Name
-------------------------------------------------------------------------

Report         ADBKALL        NAME           Sort

               ADBKONE        NAME           Data            1

Screen         ADBOOK         NAME           EDNAME          Data Name
                                             NAME            Data Name

Table          NAMES          NAME                           NAME
```

Explanation

When you choose the Data Name Usage Report Option, the DASL software displays a decision prompt. You can specify whether to list data-name usage by:

- Data name
- Field/level

To list by data name, enter D after the decision prompt. To list by screen fields or report levels, enter F after the decision prompt.

The DASL software then displays a scroll region where you list the data names for the Data Name Usage Report.

After you enter data names in this scroll region, the DASL software displays a series of screens where you specify:

- Other data names to search for usage of the specified data names
- Screens to search
- Reports to search
- Tables to search

You can specify multiple data names, screens, reports, and tables in the scroll regions provided in these screens.

Related Sections

Compare Data Names Option
Define Data Names Option
Print Data Name Definitions Option

Chapter 2
Screen Driver

This chapter describes the following Screen Driver Menu options, utilities, and elements:

- Define Data Screens Option

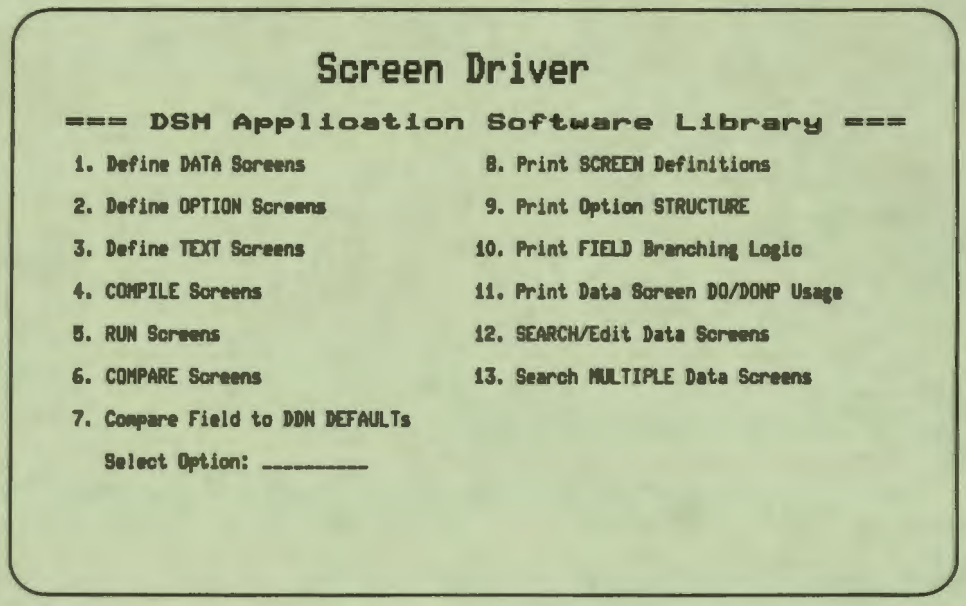
  Screen Attributes
  Fields
  Field Attributes
  Validations
  Actions
  Display Designer
  Scroll Regions
  Screen Driver Variables
  Batch Screens
  Transaction Processing

- Define Option Screens Option
- Define Text Screens Option
- Screen Driver Utilities

  Compile Screens Option
  Run Screens Option
  Compare Screens Option
  Compare Fields to DDN Defaults Option
  Print Screen Definitions Option
  Print Option Structure Option
  Print Field Branching Logic Option
  Print Data Screen DO/DONP Usage Option
  Search/Edit Data Screens Option
  Search Multiple Data Screens Option

Figure 2–1 shows the Screen Driver Menu and its options.

Figure 2–1 Screen Driver Menu



This chapter describes each option in detail. See the *DASL Programmer's Guide* for a discussion of screen design and examples of data-entry, option, and text screens.

Define Data Screens Option

Data screens are a basic component of a DASL application. Data screens allow users to enter, display, validate, and file data.

Overview

Each data screen can consist of two elements:

- A screen definition
- One or more field definitions

You can define a data screen that has a screen definition, but no field definitions, such as the driving screen for a batch screen.

The screen definition describes the general nature of the data screen. The screen definition also describes display characteristics you want the screen to show.

The field definitions describe elements within the screen. A field can specify:

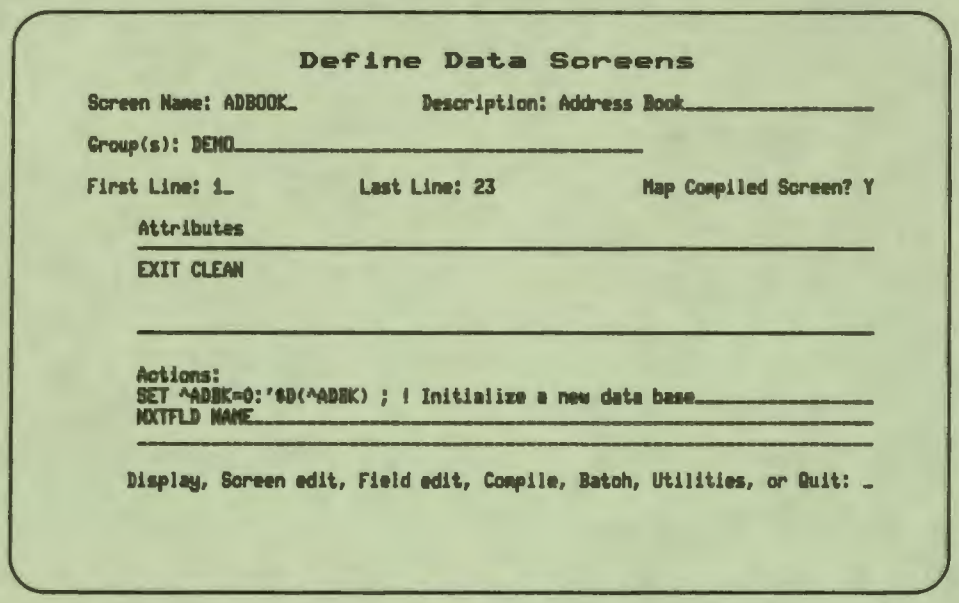
- A message or heading to display
- An item of data to acquire and store
- An action to perform

Creating Data Screens

You create new data screens for your applications through the Define Data Screens Option. You also use the Define Data Screens Option to edit, copy, delete, print, or add comments to data screens.

Figure 2–2 shows the first screen you see when you choose the Define Data Screens Option.

Figure 2–2 Define Data Screens Screen #1



Providing Screen Definitions

You can create a new screen or edit an existing screen by entering a screen name (one letter followed by 0 to 6 alphanumeric characters) after the "Screen Name:" prompt.

If the screen is new, the DASL software prompts you to fill in screen specifications.

Table 2–1 shows the information you can provide when you define a screen.

Table 2–1 Screen Definition Information

Item	Description
Screen Name	The name of the screen; a standard DASL name made up of one alphabetic character followed by up to six alphanumeric characters.
Description	A text description of the screen that appears in lookup lists. Can be up to 35 alphanumeric characters in length.
Group(s)	The names of the DASL groups to which the screen belongs. The group name is a required item that the DASL software uses to classify screens by application or by package. Before you can assign a data screen to a group, you must define the group through the Group Dictionary Option of the Development Environment Menu.

Table 2–1 (Cont.) Screen Definition Information

Item	Description
First Line	A numeric value specifying the first line of the data screen display. The first-line value must be between 1 and 23.
Last Line	A numeric value specifying the last line of the data screen display. The last-line value must be between the first-line value and 23.
Map Compiled Screen	A flag used by the Build Mapped Section Utility of the Development Environment Menu. If the flag value is Y, the compiled data screen is mapped. If the flag value is N, the compiled data screen is not mapped.
Attributes	DASL commands that control general data screen display features.
Actions	DASL commands that perform functions such as assigning values to fields within the screen and specifying branching logic.

If the screen is an existing one, the DASL software fills in the screen specifications and displays the following menu prompt:

```
Display, Screen edit, Field edit, Compile, Batch, Utilities, or Quit: _
```

You can use the menu prompt to perform any of the following:

- Use the Display Designer to edit the screen and its fields.
- Edit the screen definition information.
- Edit the field definitions.
- Compile the screen.
- Compile the screen in batch mode.
- Use the utilities to delete, copy, chart, print, or add documenting comments to the screen.
- Quit and start over.

Using the Display Designer

Enter D to invoke the DASL Display Designer. You can then use the Display Designer editing keys to edit the screen and its fields. See the "Display Designer" section for more information.

Editing the Screen Definition

Enter S to change any of the screen definitions. You can then change any of the screen definitions (except the screen name) and save your changes.

Editing the Field Definitions

Enter F to define new fields or edit existing field definitions. In the screen that appears, you can then define new fields for the screen or edit any existing fields.

Figure 2–3 shows the second Define Data Screens screen that you see when you choose Field edit at the prompt on the first Define Data Screens screen.

Figure 2–3 Define Data Screens Screen #2

```
Screen: ADBOOK        Field Name: NAME___        Data Name: NAME___

Description: Enter a new or old name._____________________________________
      Line: 5_    Column: 5__    Justify: L    Data Length: ___ (30)

Prompt: ___________________________________________________________________
      (Name: )

Help Text:                                    Help Screen: _______ (HNAME)
___________________________________________________________________________
(Enter a name as LAST, FIRST MI, TITLE)
Attributes: _______________________________________________________________
                        Data Name Attributes: LCASE
Validation: LOOKUP/LIST/VERIFY ; NAME_____________________________________
          (LOOKUP/VERIFY/LIST ; NAME)
     ------------------------ Actions ------------------------
     ! If lookup finds the entry, the cross-reference pointer will be
     ! returned.  If it's not found, or no entry is selected from the
     ! lookup list,    ID will be equal to null.
     ! For a new entry:  EXIT EQ to be certain the user "visits" all field
     ! the required fields, and branch to field STREET.
     ---------------------------------------------------------

         Edit, edit Actions, Copy, View defaults, Delete, or Quit: _
```

See Table 2–3 in the "Fields" section of this chapter for a description of the information you enter to define fields.

Processing the Data Screen

Before running a data screen, you must compile it. At compile time, the DASL software translates all Screen Driver commands into executable VAX DSM code.

Enter C in response to the menu prompt to compile the screen in your current process. The DASL software then compiles the screen and returns control to your terminal only when it completes the compile operation.

You can compile the screen using a command procedure by entering B in response to the menu prompt. The DASL software then submits a batch job and returns control to your terminal immediately.

When displaying a compiled data screen, the DASL software does the following:

1. Clears the display from the first line to the last line.
2. Processes all screen attributes.
3. Initializes field variables and data names to their default values.
4. Paints the screen with its fields onto the terminal display.
5. Passes control to a field specified in the screen action NXTFLD.
6. Continues to transfer control as specified.
7. Clears the terminal display and exits from the current data screen when no further actions direct control to another field or screen.

   If the data screen was called from another screen, the DASL software restores the context and continues execution of the calling screen.

You can test your data screens by using the Run Screen Option of the Screen Driver Menu, or by using the ^%DAS entry point routine.

Using the Utilities

You can delete, copy, create a flowchart of, or add documenting comments to the screen. Enter U in response to the menu prompt. The DASL software then displays the following utilities decision prompt:

```
Edit comments, Delete, Copy, Flowchart, Print or Search:
```

To add comments to document the screen, enter E. You then see a comments screen that allows you to specify the creator, editor, and version number of the screen, and also enter text (in a scroll region) to document the screen.

To delete the screen, enter D in response to the utilities decision prompt and verify your choice where prompted.

To make a copy of the screen under a different name, enter C and specify the new screen name. You can later edit the new screen name to change any of the copied values.

To make a flowchart of the screen, enter F. The DASL software then calls the Print Field Branching Logic Option so that you can create a Field Branching Logic Report.

To print the screen, enter P. The DASL software displays a device selection screen that allows you to produce the report through a batch process or through your current process. If you choose to use your current process, you can display the report on your terminal, print it on any VMS device, or write the report to a file.

To search for and edit text strings on the screen, enter S. The DASL software then calls the Search/Edit Data Screens Option.

Starting Over

Enter Q in response to the menu prompt to start over. The DASL software moves the cursor back to the "Screen Name:" prompt. You can also use the exit keys at any point to return to the previous screen or the DASL Main Menu.

Recommended Practices for Data Screen Design

Keep these recommended practices in mind as you design your data screens:

- Use DASL naming conventions for fields that perform routine tasks. Fields can use the same data name definitions if their functions are identical.
- Create a CLEAN field to unlock global variables and clean up the local symbol table. Use an EXIT CLEAN as the field attribute for any decision field, or at any field where users can press Main Menu or Exit to exit the field.
- When it is important to restrict information to only one user at a time, use the LOCK command to lock global information so that users cannot perform simultaneous edits.

To ensure database integrity in a transaction-based application, take the following steps in the event of a system failure:

- Bracket your transactions by using LOGON and LOGOFF commands.
- Dejournal all subsequent complete transactions (those that begin with a LOGON and end with a LOGOFF for each VAX DSM process).

Related Sections

Actions
Batch Screens
Display Designer
Appendix B, Entry Points
Fields
Field Attributes
Print Field Branching Logic Option
Screen Attributes
Screen Driver Variables
Search/Edit Data Screens Option

Screen Attributes

Screen attributes are commands that you use to control how the DASL software displays and handles processing for an entire screen.

Explanation

The DASL software has two additional types of attributes:

- Data-name attributes describe any specifications for values entered for a data name. See the "Data-Name Attributes" section of Chapter 1 for a discussion of data-name attributes.
- Field attributes control how the DASL software displays and handles input operations for the field with which you associate them. See the "Field Attributes" section of this chapter for a discussion of field attributes.

You specify screen attributes in the first screen of the Define Data Screens Option.

Table 2–2 lists DASL screen attributes and their uses.

Table 2–2 DASL Screen Attributes

Attribute	Abbreviation	Use
132		Display — Sets display width to 132 characters. The default display width is 80 characters.
BATCH{/qualifier}		Control — Specifies screen is a batch screen.
BOX[SP]x1,y1,x2,y2{,string}		Display — Draws a line or box as specified by arguments.
CHECKPT	CP	Control — Declares a checkpoint at the beginning of the screen.
EXIT[SP]field	EX	Control — Sets exit field.
NOPAD	NP	Display — Sets pad character to space — no underscores displayed.
PERM	PM	Display — Prohibits erasing of screen image on exit.

Table 2–2 (Cont.) DASL Screen Attributes

Attribute	Abbreviation	Use
SCROLL(/qualifier)SPfirst,last	SC	Display — Displays scroll region in lines specified by arguments.
TIMEOUTSPtime	TO	Input — Sets timeout value to number of seconds specified by argument.

Comments

The DASL software processes screen attributes before processing any field data.

The following sections describe most screen attributes. See the "EXIT" section of this chapter for more information about the EXIT screen attribute.

Related Sections

Actions
Data-Name Attributes, Chapter 1
Define Data Names Option, Chapter 1
Define Data Screens Option
Fields
Field Attributes

132

The 132 screen attribute specifies that a screen display is 132 characters wide.

Format

132

Comments

Keep the following points in mind when you use the 132 attribute:

- The default DASL display width is 80 columns. If you do not specify 132 as an attribute when defining a screen, the DASL software displays that screen with an 80-column width.
- When the flow of control passes to screens of different widths (wide to narrow screens or narrow to wide screens) on which data is present, the DASL Screen Driver always clears and repaints the display. The DASL software does not clear and repaint the display if control passes between screens of the same width (that is, from one 80-column screen to another or from one 132-column screen to another).
- You can use the 132 attribute to fit more than 80 columns of information in a scroll region. When you use the 132 attribute for this purpose, use the attribute WIDE or attributes WIDE and TALL on fields outside the scroll region for readability.

Related Section

Define Data Screens Option

BATCH

The BATCH screen attribute specifies that the associated data screen is a batch screen, that is, a screen that collects data noninteractively. A batch screen is designed to process data from a VMS sequential file or local array.

Format

BATCH{/DEBUG}{=n}

where:

/DEBUG	is an optional qualifier that causes write statements to be compiled into a generated routine and allows you to view the batch process on your screen
=n	is an optional modifier, an integer between 0 and 9 that causes the batch process to hang for 0 to 9 seconds after each transaction so that you can view the screen logic and errors on your screen

Comments

Keep the following points in mind when you use BATCH:

- When you use BATCH, the DASL software:
  - — Receives input from a VMS sequential file or local array and stores this data in an internal area.
  - — Stores error messages in an internal array. You can write the contents of this array to a batch log file.

  When you use BATCH without the DEBUG qualifier, the DASL software does not display the batch process on your screen.
- When you use BATCH/DEBUG, the DASL software:
  - — Displays the batch process on your screen as it reads data.
  - — Displays messages as errors occur.

  You can use BATCH/DEBUG to check your VMS sequential file for data-entry format errors and your screen for logical errors.

  When you use BATCH/DEBUG=n, the DASL software halts for **n** seconds after each read.

  BATCH/DEBUG causes write statements to be compiled into a generated routine. To remove these statements, edit the attribute to BATCH and recompile the screen.
- A screen with a BATCH attribute calls itself repeatedly with a NXTSCN action. Therefore, open the data file prior to calling the batch screen.

Related Sections

Batch Screens
LOG
LOGDMP
Screen Driver Variables

BOX

The BOX screen attribute displays a line or box on the screen at the coordinates you specify.

Format

BOX ⎵**x1,y1,x2,y2{,"string"}**

where:

x1	**is the x coordinate of the upper left corner**
y1	**is the y coordinate of the upper left corner**
x2	**is the x coordinate of the lower right corner**
y2	**is the y coordinate of the lower right corner**
"string"	**is a quoted character string that specifies the characters you use to draw the line or box**

Comments

Keep the following points in mind when you use BOX:

- The DASL software draws the line or box from coordinates x1 and y1, the upper left corner, to coordinates x2 and y2, the lower right corner.
- The x and y coordinates are relative to the screen's first line. The x coordinates are columns. The y coordinates are rows. Therefore, an x coordinate of 3 and a y coordinate of 4 refers to the third column, fourth row.
- There are 80 columns and 24 lines on a screen. The DASL software reserves line 24 for help text. You cannot specify this line as part of your box.
- If you do not specify the fifth (string) argument, the DASL software uses the default graphic line character to draw a box with straight lines.
- If you specify a string of more than one character as the fifth (string) argument, the DASL software tries as closely as possible to display a box or line of the dimensions you specify. However, it may not be able to match your specifications exactly.
- The DASL software draws a horizontal line if y1=y2 and a vertical line if x1=x2.
- When you draw a box, do not intersect wide and tall lines with normal lines.

Related Sections

Define Data Screens Option
Display Designer

Examples

The following example creates a box around the entire data screen. Note that the box surrounds the usable portion of the data screen. The DASL software reserves line 24 for help text.

```
Attributes:    BOX 1,1,79,23
```

The following example creates a horizontal line on line 5 extending across the display.

```
Attributes:    BOX 1,5,79,5
```

The following example creates a box using the asterisk (*) around the upper half of the screen.

```
Attributes:    BOX 1,1,79,12,"*"
```

CHECKPT

The CHECKPT screen attribute specifies that the DASL software execute a security checkpoint whenever it executes the screen. You can use CHECKPT to determine whether a user is allowed to continue current application activity.

Format

CHECKPT

Comments

Keep the following points in mind when you use CHECKPT:

- You can abbreviate CHECKPT as CP.
- You can use CHECKPT as a way for users to exit cleanly from closed applications. If you specify CHECKPT as a screen attribute, the DASL software executes a checkpoint whenever the screen is invoked. If the application is closed (through the Application Environment Menu) or if the current system status does not permit access to the option, then the DASL software exits from the screen when it finds a checkpoint.
- The DASL software always executes an implicit checkpoint whenever it executes an option screen or a menu screen.
- At each checkpoint, the DASL software takes the following steps:
  1. Reevaluates system status from the database to check for changes.
  2. Concatenates a + privilege with the system status, since the system is always open to a user with system manager (+) privileges.
  3. Performs a logical AND operation with system privileges and the user's process privileges, which are determined at login.
  4. Checks privileges joined by an AND for a match with option privileges.
  5. Exits from the option if the checkpoint fails.

Related Sections

Security System Option, Chapter 6
System Control Option, Chapter 6
System Status Dictionary Option, Chapter 6

NOPAD

The NOPAD screen attribute displays a blank space as the default pad character for all fields on the screen. The NOPAD field attribute displays a blank space as the default pad character at a specified field.

Format

NOPAD

Comments

Keep the following points in mind when you use NOPAD:

- You can abbreviate NOPAD as NP.
- The default DASL pad character is the underscore (_). Unless you specify otherwise, the DASL software always displays a line of underscores after the field prompt to indicate the length of the field. The number of underscores displayed equals the data length you specify when you define the field. Therefore, a field with a data length of 7 displays as:

  ```
  Prompt: _ _ _ _ _ _ _
  ```

  The NOPAD attribute displays a blank space as the default character. The maximum number of characters that users can enter at a field is still limited to the data length you specify for the field. Therefore, a NOPAD field with a data length of 7 displays as:

  ```
  Prompt:
  ```

- If you specify NOPAD as a screen attribute, the DASL software uses the blank space as the pad character for the entire screen.

  If you specify NOPAD as a field attribute, the DASL software uses the blank space as the pad character only for fields so marked.

- The NOPAD attribute is especially useful in scroll fields and noecho fields.

Related Sections

NOECHO
Scroll Regions

PERM

The PERM screen attribute specifies that the current screen not be erased when the user exits from the screen.

Format

PERM

Comments

Keep the following points in mind when you use PERM:

- You can abbreviate PERM as PM.
- Normally when a screen exits, the DASL software clears any part of the display that the screen occupies. PERM suppresses screen clearing.
- The following conditions overwrite or erase a screen image with a PERM attribute:
  - — Pressing Ctrl/W (screen refresh)
  - — Returning control to a previous screen
  - — Performing a NXTSCN action
- You can use PERM as an attribute for an application logout screen.
- You can also use PERM as an attribute of a subscreen, that is, a screen that occupies only part of a terminal display. The subscreen retains certain information that a user needs even after control passes to a new screen that occupies another part of the terminal display.

Related Sections

Application Parameters Option, Chapter 5
Define Data Screens Option
NXTSCN

SCROLL

You can use the SCROLL screen attribute to create a scroll region, specify the size of the scroll region, and declare the array to use for data collected.

Format

SCROLL{/qualifiers} SF first,last

where:

qualifiers are one or more qualifiers that specify how the DASL software handles the scroll region data. The qualifiers available are:

/ARRAY=name — specifies the name of the scroll array

/GLOBAL — specifies that the scroll array is to be a global

/NOKILL — directs the DASL software not to kill the scroll array on exit

/NORESET — directs the DASL software not to reset scroll region pointers when you use the NXTFLD action to move from a field within a scroll region to an outside field

first is the first line in the scroll region

last is the last line in the scroll region

Comments

Keep the following points in mind when you use SCROLL:

- You can abbreviate SCROLL as SC. You can abbreviate /GLOBAL as /GL, /NOKILL as /NO, /ARRAY as /AR, and /NORESET as /NRS.
- By default, the DASL software performs the following tasks in scroll regions:
  - — Stores scroll region data in a local array and uses the name of the screen as the name of the array.
  - — Reinitializes the scroll array whenever it first initializes a screen (including when it encounters a NXTSCN action without arguments to reprocess the current screen) and when it exits from the screen.
- Use the /GLOBAL qualifier to specify a global scroll array for data storage. A global scroll array is especially useful for a large amount of data such as the data found in the Report Directory of the Application Environment Menu.

If you specify a global array, the DASL software uses the following global array:

^DATG($J,"S",screen)

where:

^DATG	is a scratch global used for a number of different DASL operations
$J	is the VAX DSM $JOB special variable
"S"	identifies the subtree of the global for storage of scroll region data
screen	is the name of the current screen

Lower level subscripts identify the scroll region line and the data name.

- Use the /ARRAY=name qualifier to specify the name of the local or global array to use for the scroll region data. The name you specify must be a valid VAX DSM name of one uppercase alphabetic letter followed by 2–6 uppercase alphanumeric characters. The name must not contain a percent sign (%).

  If you also specify the /GLOBAL qualifier, you do not have to include the circumflex (^) in the array name.

- The DASL software uses the same subscripting structure for any local or global array you specify with the /ARRAY=name qualifier as it does for the default arrays. See the "Scroll Regions" section for more information on scroll array structure.

- Use the /NOKILL qualifier to direct the DASL software not to delete the scroll array upon screen exit.

- The arguments **first** and **last** are numeric values that are relative to the first line of the screen. Therefore, if the screen starts on line 3 and you specify a **first** value of 2, the scroll region starts on line 5 of the screen.

- **You must define all fields within a scroll region on the first line of the scroll region.** That is, if the first line of a screen you define as a scroll region is line 8, you locate all fields in the scroll region on line 8, although at different column positions. The DASL software repeats the first-line field definition for all lines in the scroll region.

  Therefore, the DASL software recognizes only one occurrence of the field at a time. In a scroll region, that occurrence is at the current scroll line. The current scroll line is called the *active line*. Users can change the active line with the ⇑ and ⇓ keys, if the KEY attribute enables movement within the scroll region.

- More lines can exist within the array than the DASL software can display in the specified scroll window (between **first** and **last**). In this case, the DASL software maintains a window and displays a section of the data, always displaying the current active line.
- Do not define any nonscroll fields to display within a scroll region (between **first** and **last**). Because you must define all scroll fields on the first line of the scroll region, any fields that you define on other scroll region lines are not considered to be a part of the scroll region. These fields scroll off the display.
- The DASL software resets the scroll region pointers whenever users leave the scroll region to work in a field. When users then return to the scroll region, the cursor moves to the top of the scroll region. Use the /NORESET qualifier to direct the DASL software not to reset the scroll region pointers. Users can then go to a field and return to the scroll region at the same place where they exited.

  When you use the SCROLL/NORESET screen attribute, you can also use the SCROLL/RESET field action to direct the DASL software to reset the scroll region pointers at a specified field.
- Use the BOX screen attribute to place lines above and below the scroll region.
- Use the NOPAD field attribute in all fields within the scroll region.

Related Sections

Define Data Screens Option
KEY
SCROLL
Scroll Regions

Examples

The following example is from the Send Messages Option of the Message Center, which is in the Application Environment Menu. The Send Messages Option creates a scroll region from lines 10 to 21 and specifies that the DASL software use the local array SEND as the scroll array. The Send Messages Option also uses BOX to highlight the scroll region by drawing lines just above (line 9) and below (line 22) the scroll region.

```
Attribute:   BOX 1,9,80,9; BOX 1,22,80,22 Scroll/ARRAY=SEND 10,21
```

The following example is from the Report Directory Option of the Application Environment Menu. The Report Directory Option creates a scroll region from lines 6 to 20 and specifies that the DASL software use the default global array. Again, the Report Directory uses BOX to highlight the scroll region.

```
Attribute:   SCROLL/GLOBAL 6,20; BOX 1,5,80,5 ; BOX 1,21,80,21
```

TIMEOUT

The TIMEOUT screen attribute sets a default timeout for the screen. The TIMEOUT field attribute sets a timeout for a field.

Format

TIMEOUT SP **time**

where:

time is an integer whose value is equal to the timeout in seconds

Comments

Keep the following points in mind when you use TIMEOUT:

- You can abbreviate TIMEOUT as TO.
- The value you specify in **time** is an integer that specifies the number of seconds that the DASL software must wait for user input. If no value is entered before **time** expires, the DASL software proceeds as if the user pressed Exit. Control passes to any designated exit field. If no exit field was specified, control passes to the menu or data screen from which the data screen was called.

  As more timeout periods expire, the DASL software returns to the main application menu and eventually logs the user out.
- If you do not specify a TIMEOUT value for your data screens, the DASL software uses a default timeout value that you set with the Site Parameters Option. If you do not set a default timeout, the DASL software uses a system default timeout of 300 seconds.
- TIMEOUT as a screen attribute sets a timeout value for all fields on a screen.
- TIMEOUT as a field attribute sets a timeout value for a field that overrides any value set for the screen with a TIMEOUT screen attribute.

Related Sections

DEFAULT
Define Data Screens Option
Site Parameters Option, Chapter 5

Fields

Fields are the basic units within data screens. Each field is like a subroutine you can use to display messages, solicit input, or perform actions.

Explanation

Data screens can have three basic types of fields:

- Display only
- Data entry
- Action only

Display-Only Fields

Display-only fields appear on the screen. They require no action on the part of the user. With display-only fields, you specify a line, column, and justification. Length, prompt, and attributes are optional. You use display-only fields to display any fixed recurring information, such as a column heading, the title of a screen, or messages for the user.

You can also use display-only fields as demand fields. Demand fields are fields with the DEMAND attribute that do not normally appear on the terminal. The DASL software displays demand fields only when processing or user action demands their appearance.

You can use demand fields to display user-friendly messages. For example, if a user performs an action that takes much processing time, you can have your application display a message such as the following to reassure the user that everything is still operating correctly:

```
Please stand by. Working ....
```

When your processing is complete, you can use the ERASE action to erase this message.

Because they are not data-entry fields, display-only fields do not generally need a data name. However, display-only fields can use a data name to provide consistent headers and messages throughout your application.

Data-Entry Fields

Data-entry fields solicit the user to enter new information or change existing information and validate that information when entered.

Generally, data-entry fields are based on a data name. The information defined in the Data Dictionary for the data name provides all the information that the DASL software needs to handle data evaluation and storage at the field.

A data-entry field must have a field length that is nonnull and is greater than 0.

After accepting an input value at a data-entry field that is associated with a data name, at run time the DASL software assigns the value to a local variable of the same name as the data name. Whenever the DASL software processes an ASSIGN, DEFAULT, EVAL, or RESET action, it updates the value contained in the local variable. If the field is in a scroll region, the DASL software also assigns the value to an entry in the scroll array.

Action-Only Fields

Action-only fields consist of a field name, a description, and actions. Action-only fields are subroutines consisting of DASL field actions that are performed as a result of data entry.

During the course of screen execution, the DASL software branches to action-only fields. Some action-only fields file or evaluate information in the database while others delete local variables and perform general cleanup before screen exit.

Action-only fields are not displayed on the screen. They have null or zero data length. They have no line or column positions.

Defining Fields

You define fields for your application in the second screen of the Define Data Screens Option. Table 2–3 shows the information you can provide when you define a field.

Table 2–3 Field Definition Information

Item	Description
Name	The name of the field, a standard DASL name made up of one alphabetic character followed by up to six alphanumeric characters.
Description	Up to 35 characters describing the field that appears in lookup lists.
Data name	A data name from the Data Dictionary. **You must enter a data name for fields that are evaluated or filed. You can only use the EVAL and EVALS or FILE and FILES actions on fields that use data names.**

Table 2–3 (Cont.) Field Definition Information

Item	Description
	If you specify a data name, the DASL software displays the default values for that data name in the field. You can accept any of these values or substitute new ones with the exception of the data-name attributes. Since data-name attributes cannot be overridden at the field level, total field attributes are the union of data-name attributes and field attributes. At this field, you can enter DDN to access the Define Data Names Screen, or DDNB to access the Define Data Names (Brief) screen. When you exit from the screen in which you define a data name, you return to this field.
Line	A number specifying the screen line on which the field is to be placed. **The number is relative to the first line on the screen you are defining rather than the first line of the terminal display.** That is, if the first line on the screen is 5 and the line value is 6, then the DASL software displays the field on terminal line 11. You must enter a line value for any display-only or data-entry field. You do not enter a line value for an action-only field.
Column	A number that is the column on which the field is to be placed. The actual position in which the DASL software displays the field depends on the justification you specify.
Justify	A single letter that controls the placement of the field on the display. The values you can use are as follows: R = right justification L = left justification C = center Actual field placement depends on the column value and the total field length. Field length, in turn, is the sum of the length of the prompt and the data length. Right justification means that the far right end of the field is at the position specified by the column value. Left justification means that the left end of the field is at the position specified by the column value. Center justification means that the center of the field is at the position specified by the column value.
Data length	A number that is the maximum number of characters that can be entered at the field. For display-only fields, such as screen titles or messages, you use a length of 0 or null.
Prompt	The characters to be displayed at the field as a message or for user input. The prompt is always left justified within the field.

Table 2–3 (Cont.) Field Definition Information

Item	Description
	Include terminating punctuation such as a colon (:), and a trailing space to separate the prompt from the data the user enters. If you do not specify a prompt, but do associate a data name with the field, the DASL software uses the default data-name prompt in the field. You can enter a tilde (~) to force DASL to use a null string as a prompt for the field. Instead of associating a prompt with a data-entry field, you can use the prompt as a display element such as a screen title by entering a null or zero value at the "Data Length:" prompt.
Help text	A single line of text to be displayed on the last line of the screen when a user presses Help or enters a question mark (?) at the field.
Help screen	The name of a screen that is called to provide more detailed help when a user presses Help twice or enters ? twice at the field. The screen you specify can be a data screen, an option screen, or a text screen. You must define the screen before you name it in the field.
Attributes	One or more attribute commands that specify how the DASL software displays the field and handles input operations.
Validations	One or more validation commands that specify how to check user input for correct format and content. The DASL software stores validations only for fields with nonzero data length. The DASL software processes validation commands only on nonnull responses.
Actions	One or more action commands that specify any special processing actions to take at the field. If the field is a data-entry field, the DASL software processes the actions after successful validation of user input. If the field is an action-only field, the DASL software processes the actions immediately after control passes to the field.

Field Processing

The following steps summarize DASL's run-time field logic as it processes a field:

1. When the screen is first displayed, the DASL software displays the field using your position, justification, prompt, and attribute specifications if the field is a data-entry field or a demand field. That is, the field is displayed with its default values.

2. If the field is branched to by a NXTFLD action, the DASL software reads the data length to determine if the field is a data-entry field. If the data length is zero or null, the field is an action-only field, and the DASL software proceeds to step 3.

   If the data length is positive, the field is a data-entry field, and the DASL software does the following additional steps:

   a. Saves the current data value in the DASL variable %OLD.

   b. Positions the cursor to the right of the prompt.

   c. Calls a standard read subroutine that handles data entry and function key processing.

   d. Handles any help requests from the user (through the use of the Help key or ?) after it executes the read subroutine.

      If the user has pressed Help or entered ?, the DASL software handles the help request and prompts for an input value again.

   e. Removes any leading or trailing blank spaces from the input value unless the SPACES attribute was used.

   f. Tests for a nonnull value if the REQUIRED attribute was used.

   g. Checks for full data length if the FULL attribute was used.

   h. Filters out any control characters.

   i. Filters out the default delimiter or the data-name delimiter unless the NOFILTER attribute was used.

   j. Processes all data attributes after the read subroutine, but before processing validations. If any of the checks fail, the DASL software displays an error message and prompts the user for input again.

   k. Stores the input data in the DASL variables %RES and %INP. Then, the DASL software checks the input value by using the validations supplied in the field definition or the data-name definition. Validations may change the value of %RES, but not %INP.

      If the input data is incorrect, the DASL software reports the error to the user by displaying a message on line 24 and solicits for input again by moving the cursor immediately to the right of the prompt.

3. The DASL software executes any field actions. If the field is a data-entry field, the input value is still available through %RES during action processing. In addition, any data value associated with the field when the DASL software last branched to the field is available in the DASL variable %OLD.

Related Sections

Actions
Define Data Names Option, Chapter 1
Define Data Screens Option
Display Designer
Field Attributes
Screen ADBOOK, Field CLEAN in DASL Demo
Screen ADBOOK, Field SEQ in DASL Demo
Screen ADBOOK, Field TITLE in DASL Demo
Screen Driver Variables
Validations

Examples

The following example shows the field definition for a display-only field used as a screen title. This example comes from the demonstration system screen ADBOOK.

```
Field Name: TITLE     Description: Title
Line: 2   Column: 20   Justify: C     Data Name:      Data Length:
Prompt: "Address Book"
Attributes: BOLD; WIDE
```

The following example shows the field definition for a data-entry field that solicits and validates data. This example also directs processing to the next field through NXTFLD branching actions. Notice that most values in this field are default values from the data name STREET.

```
Field Name: STREET   Description: Street Address
Line: 7   Column: 5   Justify: L   Data Name: STREET  Data Length: (30)
Prompt: (Street: )
Help Text: (Enter the street address.)
Attributes: (LCASE)
Validations: (PATTERN 1UN.E)
------------------------------ Actions -------------------------------
NXTFLD CITY
----------------------------------------------------------------------
```

The following example shows an action-only field. This field is not based on a data name and contains no information on field positioning.

```
Field Name: FILE      Description: File fields (data names).
------------------------------ Actions -------------------------------
! LOGON and LOGOFF are used to create a logical database transaction.
LOGON
! Generate new primary key if its a new entry.
! COUNT locks and increments the "counter node" in the global.
COUNT IDNEXT:'ID ; SET ID=IDNEXT:'ID
FILE NAME,STREET,CITY,STATE,ZIP,BTHDAY,PHHOME,PHWORK,PHOTH1
LOGOFF ; NXTFLD SAME
----------------------------------------------------------------------
```

Field Attributes

Field attributes are commands that you use to control how the DASL software displays and handles input operations for the field with which you associate them.

Explanation

The DASL software has two additional types of attributes:

- Data-name attributes describe any specifications for values entered for a data name. See the "Data-Name Attributes" section in Chapter 1 for a discussion of data-name attributes.
- Screen attributes control how the DASL software displays and handles processing on an entire screen. See the "Screen Attributes" section in this chapter for a discussion of screen attributes.

You specify field attributes on the second screen of the Define Data Screens Option.

Table 2–4 lists DASL field attributes and their uses.

Table 2–4 DASL Field Attributes

Attribute	Abbreviation	Use
BLINK	BL	Display — Displays blinking prompt at field.
BOLD	BO	Display — Displays bold prompt at field.
DEFAULT[SP]value	DF	Input — Sets field to default value if null. The value is written before screen actions occur.
DEMAND	DM	Display — Field is displayed only when control reaches it or when you issue a DISPLAY command to display it.
EXIT[SP]field	EX	Control — Sets exit field.
FULL	FL	Input — Requires input of exact number of characters specified in the data-name length.
KEY/qualifier		Input — Enables special keys in scroll regions.
LCASE	LC	Input — Accepts lowercase input.

Table 2–4 (Cont.) DASL Field Attributes

Attribute	Abbreviation	Use
LCASEQ	LQ	Input — Accepts lowercase input that is enclosed within quotation marks. Converts all other lowercase input to uppercase.
NOBACK	NB	Input — Prohibits use of the backup key to back up before the current field.
NOECHO	NE	Input — Turns off character echo.
NOFILTER	NOF	Input — Values can include the default system delimiter or the data-name delimiter.
NOPAD	NP	Display — Sets pad character to space — no underscores displayed.
REQUIRED	REQ	Input — Forces user to enter nonnull data for the value of a data name or field.
REVERSE	REV	Display — Sets display to reverse video.
SPACEL	SL	Input — Preserves leading blank spaces and strips trailing blank spaces.
SPACES	SP	Input — Permits leading and trailing blank spaces.
STACET	ST	Input —Preserves trailing blank spaces and strips leading blank spaces.
TALL	TL	Display — Displays prompt in double-height, double-width characters.
TERM	TE	Input — Allows users to enter values without pressing Return at a field.
TIMEOUT[SP]time	TO	Input — Sets timeout value to number of seconds specified by argument.
UNDER	UN	Display — Underlines prompt.
WIDE	WD	Display — Displays prompt in double-width characters.

Comments

Keep the following points in mind when you use field attributes:

- The DASL software processes screen attributes before processing any field attributes.
- The DASL software displays data-name attributes in the Field Definition Screen and does not override these attributes at the field level. When you use a data name in a field, the DASL software combines the attributes you give to the data name and the attributes you give to the field to determine the field attributes.
- The DASL software processes field attributes that specify field display characteristics after processing screen attributes and before processing screen actions.
- The DASL software processes field attributes that specify field input operations after the flow of control reaches the field and before soliciting user input.

The following sections describe most field attributes. See the "EXIT" section of this chapter for more information about the EXIT attribute. See the " TIMEOUT" and "NOPAD" sections of this chapter for more information on these field attributes.

Related Sections

Actions
Data-Name Attributes, Chapter 1
Define Data Names Option, Chapter 1
Define Data Screens Option
Fields
Screen Attributes

BLINK

The BLINK field attribute displays a blinking prompt.

Format

BLINK

Comments

Keep the following points in mind when you use BLINK:

- You can abbreviate BLINK as BL.
- To have a field prompt blink, give the field the BLINK attribute. By default, a field prompt does not blink.
- A field with a BLINK attribute has the following characteristics:
  - — If the field is only a prompt, then the prompt takes on the BLINK attribute.
  - — If the field is a prompt with data displayed by a DEFAULT attribute, then only the prompt takes on the BLINK attribute.
- Both the BLINK attribute and the DISPLAY/BLINK action cause the current field to blink. However, the BLINK attribute is a permanent part of the field while the DISPLAY/BLINK action applies only to the current branching to the field. DISPLAY/BLINK stays in effect only until you leave the current field, erase the field, or redisplay the field.

  If you use the BLINK attribute, the field blinks each time the DASL software displays it (that is, each time the field becomes active). If you use the DISPLAY/BLINK action, the field blinks only on the first display of the field and does not blink on subsequent displays unless you reissue a DISPLAY/BLINK action.
- A blinking prompt can be annoying and distracting. Use BLINK sparingly. BLINK is most useful for calling user attention to fields that display warnings and messages, or values that are out of range.

Related Sections

DISPLAY
Fields

Example

The following example shows the WAIT field used with compile and print options in the DASL software. WAIT is the "one moment please . . . " message that appears during the delay in compilation and printing. For that reason, WAIT field attributes include:

- DEMAND (to prevent display until the flow of processing reaches the field)
- BLINK (to make the message stand out)

```
Field Name: WAIT    Description: Wait message
Prompt:  one moment please...
Attributes:  DEMAND ; BLINK
--------------------------- Actions-------------------------------
DISPLAY WAIT ; DO TAG^ROUTINE ; ERASE WAIT ;
NXTFLD FIELD
-----------------------------------------------------------------
```

BOLD

The BOLD field attribute displays the field prompt in boldface characters.

Format

BOLD

Comments

Keep the following points in mind when you use BOLD:

- You can abbreviate BOLD as BO.
- A field with the BOLD attribute has the following characteristics:
  - — If the field is only a prompt, then the prompt takes the BOLD attribute.
  - — If the field is a prompt with data displayed by a DEFAULT attribute, then only the prompt takes on the BOLD attribute.
- Both the BOLD attribute and the DISPLAY/BOLD action cause the DASL software to display the current field in boldface. However, the BOLD attribute is a permanent part of the field while the DISPLAY/BOLD action applies only to the current branching to the field.

  If you use the BOLD attribute, the DASL software displays the field in boldface each time the field becomes active. If you use the DISPLAY/BOLD action, the DASL software displays the field in boldface at the first display, but does not show the field in boldface on subsequent displays unless you reissue a DISPLAY/BOLD action.

Related Sections

BLINK
DISPLAY

Examples

The following example shows the TITLE field from the Define Data Names Option Screen. As with most DASL screen titles, the title field has been given the attributes BOLD and WIDE.

```
Attributes:   BOLD ; WIDE
```

The following example shows the field attributes for the COPYMSG field from the Define Data Names Option Screen. COPYMSG is a message that alerts users that they are attempting to copy data-name information into a data name that is already defined. The attributes for this field include:

- NOPAD (to suppress the display of underscores)
- DEMAND (to display the field only if the user attempts to copy to an existing data name)
- BOLD (to highlight the message)

```
Attributes:   NOPAD ; DEMAND ; BOLD
```

DEFAULT

The DEFAULT field attribute assigns a default value to a field and to any data name associated with that field.

Format

DEFAULT SP**value**

where:

value is the default value you assign

Comments

Keep the following points in mind when you use DEFAULT:

- You can abbreviate DEFAULT as DF.
- The default value you assign can be any expression that is valid input to the field. The DASL software evaluates the expression before displaying the screen and assigns the value of the expression to the field (%RES) and to a data name if one is present.
- You can use $TIME and $DATE as arguments to the DEFAULT attribute. $TIME displays the current time in output format. $DATE displays the current date in output format. (You specify the output format for times and dates in the Site Parameters Screen of the Development Environment Menu.)
- When you use DEFAULT to assign a value to a field and later reset the field with RESET, the DASL software reassigns the default value as the field value.

Related Section

RESET

Example

The following example shows the field CEQ displayed by a number of DASL options as the "Continue, Edit, or Quit:" prompt. Before displaying the screen, the DASL software assigns the DEFAULT value C to the data name GNCEQ. If a user presses Return, the DASL software retains C as the value of %RES and GNCEQ.

```
Field Name: CEQ    Description: Continue, Edit, or Quit field
Prompt: Continue, Edit, or Quit
Help Text: Enter "C" to continue, "E" to edit, or "Q" to quit.
Attributes: EXIT CLEAN ; REQUIRED ; DEMAND ; DEFAULT "C"
Validation: TABLE C,E,Q
```

DEMAND

The DEMAND field attribute displays a field upon demand (when input is required at that field or when you use the DISPLAY action to display the field).

Format

DEMAND

Comments

Keep the following points in mind when you use DEMAND:

- You can abbreviate DEMAND as DM.
- The DASL software displays a field with the DEMAND attribute in the following two circumstances:
  - — When user input is required (after the field is branched to by a NXTFLD action)
  - — When you include a DISPLAY action to show the field (at some other point on the screen)
- You can place two or more demand fields at the same or overlapping coordinates.

Related Sections

ASSIGN
DISPLAY
ERASE
NXTFLD

Example

The following example shows the attributes in SEQ, the field that displays the "Save, Edit, or Quit:" prompt used on the Define Data Dictionary Option.

- The EXIT CLEAN attribute specifies that processing passes to an exit field called CLEAN if you press Exit.
- The REQUIRED attribute specifies that you must enter a value.
- The DEMAND attribute specifies that the decision prompt appears only when users branch to this field with NXTFLD.

When you branch to another field with NXTFLD, the decision prompt is erased.

```
Attributes:    EXIT CLEAN ; REQUIRED ; DEMAND
```

FULL

The FULL attribute requires that the user enter the full data length as specified through the Define Data Names Option, the Define Templates Option, or the Define Data Screens Option.

Format

FULL

Comments

Keep the following points in mind when you use FULL:

- You can abbreviate FULL as FL.
- When you use FULL, the DASL software handles the error processing for input that is too short or too long. The DASL software displays a standard error message (contained in %MSG) to the user and prompts for the data again.
- The DASL software processes the FULL attribute only after removing leading or trailing spaces from the value entered.
- When you specify the FULL attribute at a field, users cannot enter an asterisk to perform lookups at that field. The DASL software displays a standard error message and prompts for data again.

Related Sections

Define Data Names Option, Chapter 1
Define Data Screens Option
Screen Driver Variables

KEY

The KEY field attribute enables the arrow keys for scroll region editing.

Format

KEY/qualifier

where:

qualifier specifies the keys to enable. The qualifiers are:

/DELETE — enables the Remove key to delete a line

/FORWARDBACK — enables the Prev Screen and Next Screen keys to move up or down one window in the scroll region

/INSERT — enables the Insert Here key to insert a line

/UPDOWN — enables ⇑ and ⇓ to move up and down within the scroll region

Comments

Keep the following points in mind when you use the KEY attribute:

- You can abbreviate KEY/DELETE as KEY/D, KEY/INSERT as KEY/I, KEY/UPDOWN as KEY/UD, and KEY/FORWARDBACK as KEY/FB.
- You can use the KEY attribute only for scroll region fields. You must explicitly enable the various scroll region function keys for them to have an effect in the scroll region. Any key you do not enable cannot be used.
- KEY/DELETE enables the Remove key. When users press Remove, the DASL software deletes the current scroll region line. Data on any succeeding scroll lines moves up one line. The DASL software maintains the current scroll line as the active line.
- KEY/FORWARDBACK enables the Prev Screen and Next Screen keys. When users press Prev Screen or Next Screen, the DASL software moves the cursor in the scroll region up or down one window.
- KEY/INSERT enables the Insert Here key. When users press Insert Here, the DASL software moves the data associated with the current scroll line (and any succeeding scroll lines) down one line. The DASL software creates a new blank scroll line as the active line, and initializes all the scroll region data names to their default values.

- KEY/UPDOWN enables ⇑ and ⇓. When users press ⇑, the DASL software moves the cursor and the active line up one line. If the active line is the first line in the scroll region, the DASL software takes no action.

  When users press ⇓, the DASL software moves the active line down one line. If the current active line is the last line of data in the scroll region, the DASL software takes no action.

  KEY/UPDOWN automatically enables KEY/FORWARDBACK.

Related Sections

Define Data Screens Option
SCROLL
Scroll Regions

Examples

The following example enables ⇑ and ⇓ in a scroll region. Users can move the cursor up and down in the region, but they cannot add and delete lines. Note that KEY/UPDOWN also enables the Prev Screen and Next Screen keys.

```
Attributes:    KEY/UPDOWN
```

The following example enables all the arrow keys in a scroll region.

```
Attributes:    KEY/UPDOWN/INSERT/DELETE/FORWARDBACK
```

LCASE

The LCASE attribute allows users to enter lowercase input at this field.

Format

LCASE

Comments

Keep the following points in mind when you use LCASE:

- You can abbreviate LCASE to LC.
- Normally, the DASL software converts all lowercase input to uppercase. The LCASE field attribute overrides this default condition. The DASL software stores input values in the database exactly as users enter it, and displays input in lowercase.
- The DASL software always converts cross-reference subscripts to uppercase. Therefore, cross-references are case insensitive to data entered and stored in lowercase.

Related Sections

Data-Name Attributes, Chapter 1
LCASEQ

LCASEQ

The LCASEQ attribute allows users to enter lowercase data within quotation marks at a field.

Format

LCASEQ

Comments

Keep the following points in mind when you use the LCASEQ attribute:

- You can abbreviate LCASEQ to LQ.
- Normally, the DASL software converts all lowercase input to uppercase. The LCASEQ field attribute overrides this default condition. The DASL software converts lowercase input values to uppercase, with the exception of lowercase values within quotation marks. The DASL software stores input values within quotation marks in the database exactly as users enter it.

Related Sections

Data-Name Attributes, Chapter 1
LCASE

Examples

You can use LCASEQ as a field attribute for a field that requires users to enter MUMPS code. Users can then enter all MUMPS code in lowercase values and include any literal lowercase code within quotation marks. The DASL software converts any lowercase values outside the quotation marks into uppercase values. For example, the user can enter the following values:

```
CODE:   s var="test" Return
```

The DASL software stores the following values in the database:

```
S VAR="test"
```

NOBACK

The NOBACK field attribute prevents the user from using the Backup key to back up from the current field to the previous field.

Format

NOBACK

Comments

Keep the following points in mind when you use NOBACK:

- You can abbreviate NOBACK as NB.
- Normally the DASL software maintains a *backup list*. A backup list is a list of fields on the current screen where the user has already entered data. If the user presses Backup, then the DASL software moves the cursor to the previous field, the latest entry on the backup list. Each additional backup moves the cursor to the previous field.

  If you use NOBACK as a field attribute, the DASL software deletes the backup list. Users are not able to back up from the field. Pressing Backup at the field has no effect.

Related Section

Define Data Screens Option

NOECHO

The NOECHO attribute specifies that the DASL software does not display data entered by a user on the screen.

Format

NOECHO

Comments

Keep the following points in mind when you use NOECHO:

- You can abbreviate NOECHO as NE.
- NOECHO is especially useful for entering sensitive data, such as application system passwords. Use NOECHO with the NOPAD attribute for password fields.
- Use NOECHO with the NOPAD attribute for prompts such as the "Press return to continue" prompt.

Related Section

Define Data Screens Option

Example

The following example shows the PASSWORD field from the login screen for an application. Because application users enter a confidential password at this field, the field attributes are set to suppress both data display and the default underscore field character.

```
Prompt:    Password:
Help Text:    Enter your User Password.
Attributes:    NOECHO ; NOPAD
```

NOFILTER

The NOFILTER attribute permits users to enter the delimiter character.

Format

NOFILTER

Comments

Keep the following points in mind when you use NOFILTER:

- You define the default system delimiter in the Application Parameters Option of the Development Environment Menu.
- If you do not use NOFILTER, the DASL software automatically checks delimiter characters. On finding a delimiter in entered text, the DASL software displays an error message and prompts for data again.
- NOFILTER permits users to enter the default system delimiter or the data-name delimiter if the field data name overrides the default delimiter.
- Use NOFILTER only on data that is not pieced. If application users enter the piece delimiter as data at a field for data that is pieced, the database is corrupted.

Related Sections

Application Parameters Option, Chapter 5
Define Data Names Option, Chapter 1
Fields

REQUIRED

The REQUIRED attribute forces the user to enter nonnull data for the value of a data name or field.

Format

REQUIRED

Comments

Keep the following points in mind when you use REQUIRED:

- You can abbreviate REQUIRED as REQ.
- The user must enter values at a field with a REQUIRED attribute. If you do not want to allow null values, use the REQUIRED attribute.
- If the user presses Return to enter a null value at a required field, the DASL software displays an error message on line 24 and moves the cursor to the right of the prompt to solicit input again.

Related Sections

Data-Name Attributes, Chapter 1
Define Templates Option, Chapter 1

REVERSE

The REVERSE field attribute displays a specified field in reverse video.

Format

REVERSE

Comments

Keep the following points in mind when you use REVERSE:

- You can abbreviate REVERSE as REV.
- The default display mode for all screens and fields is standard video (white on black).
- A field with the REVERSE attribute has the following characteristics:
  - — If the field is only a prompt, then the prompt takes the REVERSE attribute.
  - — If the field is a prompt with data displayed by a DEFAULT attribute, then only the prompt takes the REVERSE attribute.
  - — If the field accepts new data, then the new data receives the REVERSE attribute.
- Both the REVERSE attribute and the DISPLAY/REVERSE action cause the DASL software to display the current field in reverse video. However, the REVERSE attribute is a permanent part of the field while the DISPLAY/REVERSE action applies only to the current branching to the field.

  If you use REVERSE, the DASL software shows the field in reverse video whenever it displays the field. If you use DISPLAY/REVERSE, the terminal displays the field in reverse video at the first display, but does not show the field in reverse video on subsequent displays unless you reissue the DISPLAY/REVERSE action.

Related Sections

DISPLAY
Fields

SPACEL

The SPACEL attribute directs the DASL software to preserve leading spaces and strip trailing spaces when it stores values in the database.

Format

SPACEL

Comments

Keep the following points in mind when you use the SPACEL attribute:

- You can abbreviate SPACEL to SL.
- Normally the DASL software strips both leading and trailing spaces in input data when it stores values in the database. The SPACEL attribute overrides this default condition so that the DASL software preserves leading spaces and strips only trailing spaces.

Related Sections

Data-Name Attributes, Chapter 1
SPACES
SPACET

Example

You can use the SPACEL attribute at a field in which the user enters text that becomes a heading in a screen or report. In the following example, in which the user enters text for a heading, the SPACEL attribute preserves the leading spaces and strips the trailing spaces:

```
Heading: _ _ _ _ _ _Progress Report
```

SPACES

The SPACES attribute permits input to retain leading and trailing blank spaces.

Format

SPACES

Comments

Keep the following points in mind when you use SPACES:

- You can abbreviate SPACES as SP.
- By default, the DASL software removes all leading or trailing spaces from values entered at a field. To retain leading or trailing spaces, specify the SPACES attribute.

Related Sections

Data-Name Attributes, Chapter 1
SPACEL
SPACET

SPACET

The SPACET attribute directs the DASL software to strip leading spaces and preserve trailing spaces when it stores values in the database.

Format

SPACET

Comments

Keep the following points in mind when you use the SPACET attribute:

- You can abbreviate SPACET to ST.
- Normally, the DASL software strips both leading and trailing spaces in input data when it stores the data in the database. The SPACET attribute overrides this default condition so that the DASL software strips leading spaces and preserves trailing spaces.

Related Sections

Data-Name Attributes, Chapter 1
SPACEL
SPACES

Example

You can use the SPACET attribute at a field in which the user enters a value that becomes a prompt on a screen or report. In the following example, in which the user must specify a prompt, the SPACET attribute strips leading spaces but keeps trailing spaces that the user enters:

```
Prompt: Name:_ _ _ _ _ _ _
```

TALL

The TALL field attribute displays the field prompt in double-height, double-width characters.

Format

TALL

Comments

Keep the following points in mind when you use TALL:

- You can abbreviate TALL as TL.
- When you specify TALL as a field attribute, the terminal displays the prompt in double-height, double-width letters. The terminal displays the top half of the prompt characters at the line and column you specify and the bottom half of the prompt characters at the specified column on the next line.
- TALL is line oriented. The terminal also displays any other fields, boxes, and lines on the same line as a field with the TALL attribute in double-height, double-width characters.
- The number of characters and character positions on a double-height, double-width line is half that normally available (that is, 40 instead of 80 characters).
- Use TALL as an attribute only for screen titles and other headings where you are not soliciting input values. The terminal displays only the top half of data values entered in a tall field.

Related Sections

Fields
WIDE

TERM

The TERM attribute allows users to enter values without pressing Return at a field. Users must enter values that correspond to the full length of the field.

Format

TERM

Comments

Keep the following points in mind when you use the TERM attribute:

- You can abbreviate TERM as TE.
- Normally users must press Return to enter data at a field. The TERM attribute overrides this default condition. The user does not need to press Return to enter a value at a field with the TERM attribute. When you use TERM, the DASL application accepts a value at a field and moves to the next field if the user has entered a value that corresponds to the full length of the field.
- Fields that use the TERM attribute also automatically exhibit the FULL attribute. The user must enter the full data length as specified through the Define Data Names Option, the Define Templates Option, or the Define Data Screens Option.
- The TERM attribute eliminates repetitive keystrokes, and is useful for long forms that contain standard fields of fixed length, such as fields that request phone numbers, sex, or a one-letter response (Y or N).

Related Section

Define Data Screens Option

Example

In this example, the user enters an identification number, such as a Social Security number, in three fields of fixed length. You can give the first two fields the TERM attribute, so that the user does not need to press Return before moving to the next field. The user presses Return only at the third (and last) field.

```
ID Number:   908 - 04 - 5987 [Return]
```

Note that if the user enters only two characters in the first field of the identification number (a field that requires three characters), an error message is displayed.

UNDER

The UNDER field attribute specifies that the DASL software underline the prompt associated with that field.

Format

UNDER

Comments

Keep the following points in mind when you use UNDER:

- You can abbreviate UNDER to UN.
- A field with the UNDER attribute has the following characteristics:
  - — If the field is only a prompt, then the prompt takes the UNDER attribute.
  - — If the field is a prompt with data displayed by a DEFAULT attribute, then only the prompt takes the UNDER attribute.
  - — If the field accepts new data, then the new data receives the UNDER attribute.
- Both the UNDER attribute and the DISPLAY/UNDER action cause the DASL software to underline the field. However, the UNDER attribute is a permanent part of the field while the DISPLAY/UNDER action applies only to the current branching to the field.

  If you use UNDER, the DASL software underlines the field whenever it displays the field. If you use DISPLAY/UNDER, the DASL software underlines the field at the first display, but does not underline the field on subsequent displays unless you reissue the DISPLAY/UNDER action.

Related Section

DISPLAY

WIDE

The WIDE field attribute displays the field prompt in double-width characters.

Format

WIDE

Comments

Keep the following points in mind when you use WIDE:

- You can abbreviate WIDE as WD.
- WIDE is line oriented. The terminal also displays any other fields on the same line as a field with the WIDE attribute in double-width characters.
- The number of characters and character positions on a double-width line is half that normally available (that is, 40 instead of 80 characters).
- You can use the TALL and WIDE attributes to display screen titles.

Related Section

Fields

Validations

Validations are commands you use to check the data that a user enters at a field.

Format

VALIDATION{/qualifier}{ SP argument list}

where:

qualifier	**is an optional qualifier or qualifiers — used only with LOOKUP**
argument list	**is one or more arguments separated by commas**

Explanation

You can specify one or more validations when you define templates, data names, and data-entry fields. The DASL software executes validations only for fields at which you solicit data. Do not use validations for action-only or display-only fields.

The DASL software always uses the most specific validation. That is, field validations override the default validations defined for the data name on which the field is based. Data-name validations in turn override the validations defined for the template on which the data name is based.

On the other hand, if a field does not have an explicit validation defined at the field level, the DASL software uses the next defined validation (for the data name or template) as the default validation.

DASL Validations

Table 2–5 lists the validations you can use.

Table 2–5 DASL Validations

Validation	Argument	Abbrev	Input Required
COND	**condition**	CO	Any input that makes **condition** true
DATE	**{early{,late}}**	DT	A date or range of dates
DATEN	**{early{,late}}**	DTN	Any, including inexact, date format
DO	**reference**		Value that passes the validation test performed by the VAX DSM routine specified by the argument
INTEGER	**{low value{,high value}}**	IN	Specified integer or specified range of integers
LOGICAL		LGC	Y for yes or N for no
LOOKUP	**{/qualifier}argument**	LU	Values to be looked up as specified by the qualifier and the argument list
NAME		NM	Name format
NUMERIC	**{low value{,high value{,decimal}}}**	NU	Numeric data, or range of numeric data, specified by the argument
PATTERN	**pattern**	PA	Value matching the pattern specified by **pattern**
TABLE	**table**	TB	Value exactly matching one of the table values
TABLEM	**list**	TBM	Multiple input values matching the table values
TABLEX	**table**	TBX	Value partially matching one of the table values
TIME	**{early{,late}}**	TM	Without argument, any input in time format. With argument, the time or range of time specified by the argument
XECUTE	**DSMcode**	XE	Value that passes the validation test performed by the VAX DSM statements specified by the argument
%command			Value that passes the validation test performed by the VAX DSM routine specified by the %command

Validation Processing

During execution, the DASL software invokes a field's validation only if the user enters data. If the user requests help, for example, the DASL software displays the appropriate help message or text screen and then returns to prompt for the field again. If the user enters a null value (presses Return), the DASL software executes the defined field actions and does not invoke the validation. If you do not want to allow null values, you can use the REQUIRED field attribute or data-name attribute.

After the user enters data, the DASL software processes the validations. The DASL software performs validations from left to right until one succeeds.

The DASL software tests the success of each validation by examining the DASL variable %MSG. If %MSG contains a null value, the validation is successful, and DSM stops evaluation. **The DASL software does not evaluate any validations to the right of the satisfied validation.**

If %MSG has a value, the validation was not successful. The DASL software proceeds to the validation to the right of the unsuccessful validation. If there are no more validations, the DASL software displays the message in %MSG and prompts for data entry at the field again.

Because the DASL software tests each validation in turn, you cannot perform a logical AND operation on validations to use multiple validations. **You can, however, use any validation as an action.** You can perform logical AND operations on validations used as actions.

When you use validations as actions, the DASL software tests the variable %RES and returns %MSG when appropriate. You must test %MSG and take appropriate action.

With most validations, the DASL software supplies the appropriate %MSG message for incorrect data. **With DO, XECUTE, or a %command, however, you must provide your own %MSG message.**

Some successful validations can change the value of %RES. If the value of %RES changes, the DASL software displays the modified value in the field. For example, a user can enter a T for TODAY in a date field. The DASL software then erases the T and displays today's date, the modified value of %RES.

Related Sections

Actions
Data Types
Fields
Screen Driver Variables

COND

The COND validation requires that its truth-valued argument be true for entered data to be valid.

Format

COND SP condition

where:

condition is a VAX DSM truth-valued expression

Comments

Keep the following points in mind when you use COND:

- You can abbreviate COND as CO.
- You can use the DASL variable %RES as part of the truth-valued expression. %RES always contains the data entered at the current field.
- When you use COND, the DASL software handles the error processing. If the validation fails, the DASL software returns the standard error message "Improper Format" and prompts for the data again.

Related Sections

Screen Driver Variables
Validations

Example

In the following example, the validation succeeds if the value entered has not changed from a previous entry.

```
COND %RES=%OLD
```

DATE

Without an argument, the DATE validation requires that application users enter a complete date (day, month, and year) in one of the standard DASL date formats. With an argument, the DATE validation requires that users enter a complete date in the range specified by the argument.

Format

DATE{ SP early{,late}}

where:

early	is an expression that evaluates to the earliest (or only) acceptable date
late	is an expression that evaluates to the latest acceptable date

Comments

Keep the following points in mind when you use DATE:

- You can abbreviate DATE as DT.
- The Define Data Names Option and Define Templates Option on the Data Dictionary Menu require you to specify a data type for a data name or template. If you declare a DATE data type, the validation for that data name or template defaults to DATE.
- The DASL software recognizes input as a valid date if it contains any of the following:
  - — Numbers
  - — Punctuation characters such as slash (/), hyphen (–), comma (,), space, or period (.)
  - — Three-letter strings representing the months of the year
  - — The letter T (for today) and an optional offset (+ or – n) representing days in the past or future

  Table 2–6 shows some standard complete date input formats.

Table 2–6 Standard Date Input Formats

Form	Example	Explanation
dd-mmm-yy	13-Jun-89	Basic format
dd-mmm-yyyy	13-Jun-1898	For dates previous to 1900
mm-dd-YY	06-13-89	Uses numbers only

Table 2–6 (Cont.) Standard Date Input Formats

Form	Example	Explanation
T	JUN-13-89	Current date (today)
T+n	T+4	Current date plus n days If current date is JUN-13-89, input represents JUN-17-89
T–n	T–4	Current date minus n days If current date is JUN-13-89, input represents JUN-9-89

- The DASL software maintains all dates in internal and external format. The internal format is the internal format maintained in the VAX DSM special variable $HOROLOG. (See the *VAX DSM Language Reference Manual* for more information on the $HOROLOG format.)

  The default external format is the standard VMS format, that is:

  DD-MMM-YY

  The DASL software uses the internal format for data storage. Whenever a DATE validation is successful, DSM stores the date in internal format in the DASL variable %DTI. When you file the date, the DASL software stores the date in internal format.

  The DASL software uses the external format for display purposes. Whenever a validation is successful, the DASL software stores the external-format date in the DASL variables %RES and %DTX. The DASL software also displays the complete external-format date on the screen. Whenever the DASL software must display the date at some later point (in response to an inquiry or on a report), it transforms the date to external format.

- You can change the standard external date format for your applications in the Site Parameters Option on the Development Environment Menu.

  Table 2–7 shows the characters you can use with punctuation to produce standard date output.

Table 2–7 Standard Date Output Characters

Character	Explanation
D	Displays the day as a number
W	Displays the day as a word
N	Displays the month as a number
M	Displays the month as a word
Y	Displays the year
CY	Displays the century and year
S	Entered before a D or an N, displays a space before numbers less than 10
Z	Entered before a D or an N, displays a zero before numbers less than 10

Related Sections

Data Types, Chapter 1
DATEN
Screen Driver Variables
Site Parameters Option, Chapter 5

Examples

The following example checks for a complete date match with DATE.

```
Validation:   DATE
```

The following example checks for a date between January 1, 1875 and the current date.

```
Validation:   DATE "1-Jan-1875","T"
```

The following example checks for a date in the range represented by the values of VAR1 and VAR2.

```
Validation:   DATE VAR1,VAR2
```

The following example shows some standard date output.

Format	Output
M D, CY	May 7, 1989
W M D, CY	Thursday May 7, 1989
ZM/ZD/Y	05/07/89

DATEN

The DATEN validation without an argument requires that users enter a complete or partial date. The DATEN validation with an argument requires that users enter a complete or partial date in the range specified by the argument.

Format

DATEN{ SP early{,late}}

where:

early	is an expression that evaluates to the earliest (or only) acceptable date
late	is an expression that evaluates to the latest acceptable date

Comments

Keep the following points in mind when you use DATEN:

- You can abbreviate DATEN as DTN.
- DATEN accepts any of the complete date formats described under DATE. DATEN also accepts partial dates. When a user enters a partial date, the DASL software interprets the date and displays it as a complete date in the standard external format.

  Table 2–8 shows examples of partial dates and how the DASL software interprets them.

Table 2–8 DATEN Partial Date Interpretations

User Enters	Interpreted As
mmm	First day of month in current year
yyyy	January 1 of specified year
mmm/Y	First day of month in specified year

- The DASL software stores and displays all dates validated with DATEN in the internal and external formats discussed under the DATE validation. See the "DATE" section for more information.

Related Sections

Data Types, Chapter 1
DATE
Screen Driver Variables

Examples

The following example requires a partial or complete date.

```
Validation:   DATEN
```

The following example requires a complete or partial date falling between January 1, 1875 and the current date.

```
Validation:   DATEN "1-Jan-1875","T"
```

DO

DO is a validation, a screen action, and a field action. As a validation, DO calls the VAX DSM routine specified in the argument. As an action, DO calls the screen or VAX DSM routine specified in the argument.

Format

DO [SP]**entryref**

where:

entryref is an entry reference to a VAX DSM routine

Comments

Keep the following points in mind when using DO as a validation:

- The DO validation calls a VAX DSM routine for specialized validation and error processing. The called routine can test the DASL variable %RES (which contains the current data value to be validated). If the validation fails, the routine then sets %MSG to an appropriate user error message.
- The user validation routine can modify the value of %RES. If %RES is not equal to the user input, the DASL software automatically displays the updated value of %RES.

  To alter both the stored value and the displayed value of %RES, change the value of %RES in the validation routine.

  To alter the stored value of %RES, but not the displayed value, use an input and output transform.
- You can use parameter passing with the DO validation. See the *VAX DSM Language Reference Manual* for more information about parameter passing.
- When you use DO as a validation, you cannot use a postconditional expression with the argument.

Related Sections

DO
DONP
Appendix B, Entry Points
Screen Driver Variables
Transforms, Chapter 1

Example

Use the DO validation when you need a special validation that the DASL validations do not provide. Call a routine to perform your validation as follows:

```
DO TAG^ROUTINE
```

INTEGER

The INTEGER validation requires that the value entered be an integer number.

Format

INTEGER{ SP low value{,high value}}

where:

low value	**is the low integer value**
high value	**is the high integer value**

Comments

Keep the following points in mind when you use INTEGER:

- You can abbreviate INTEGER as IN.
- Without an argument, the INTEGER validation requires that the value entered be an integer number.

  With one argument, the INTEGER validation requires that the value be an integer equal to or greater than its argument.

  With two arguments, the INTEGER validation requires that the value entered be an integer within the range specified by its arguments.

Related Sections

Fields
Screen Driver Variables

Examples

The following example specifies integer values greater than or equal to 1.

```
Validation:    INTEGER 1
```

The following example specifies integer values in the range of 1 to 99.

```
Validation:    INTEGER 1,99
```

The following example specifies integer values between twice the value of the local variable VAR1 and twice the value of the local variable VAR2.

```
Validation:    INTEGER (VAR1*2),(VAR2*2)
```

LOGICAL

The LOGICAL validation requires that the user enter Y (for Yes) or N (for No).

Format

LOGICAL

Comments

Keep the following points in mind when you use LOGICAL:

- You can abbreviate LOGICAL as LGC.
- When you use LOGICAL, the DASL software handles the error processing. If the validation fails, the DASL software returns a standard error message to the user (contained in %MSG) and prompts for the data again.

Related Section

Screen Driver Variables

LOOKUP

LOOKUP is a validation and an action. The LOOKUP validation compares input data with existing database entries. For the validation to succeed, the value entered must already exist in the database.

Format

LOOKUP{/qualifiers}{ SP ,select criteria}

LOOKUP{/qualifiers}{ SP data name{,select criteria}}

LOOKUP{/qualifiers}{ SP global ref}

where:

/qualifiers	are a list of one or more of the following that specify how to perform the LOOKUP validation: **/CUTBACK=N** — use N digits as the number of characters in the input string to compare in a search for a cutback match **/EXACT** — the value entered must exactly match an entry in the database **/LIST** — display a lookup list showing all possible values for the data name if a user enters an asterisk (You cannot use this qualifier with KWIC cross-reference globals.) **/NOPAINT** — do not repaint the calling screen when a user selects an entry from the lookup list **/VERIFY** — always display a lookup list for user verification even in the case of a single match **/VERIFY=PARTIAL** — display a lookup list in the case of a single partial match, but not for an exact match
data name	is a data name to look up (If not specified, the data name for the current field is implied.)
select criteria	are a list of one or more data names and values that modify or limit the search
global ref	is a DASL global reference

Abbreviations

You can use the following abbreviations for the LOOKUP validation and its qualifiers.

Command/Qualifier	Abbreviation
LOOKUP	LU
/CUTBACK=N	/CU=N
/EXACT	/EX
/LIST	/LI
/VERIFY	/VE
/VERIFY=PARTIAL	/VE=PA

Explanation

The LOOKUP validation checks the value an application user enters against existing database values.

The LOOKUP validation can check for the following types of matches:

- Exact match

  When the LOOKUP validation attempts an *exact match*, the DASL software displays any matches in a lookup list. The user can select one item from the lookup list. If the user makes a selection, the LOOKUP validation completes successfully. If the /EXACT qualifier is specified and no exact matches are found, or the user does not select an item from the lookup list, the LOOKUP validation fails.

  In an exact match, the characters in the input string must exactly match a database value, for example:

```
Input in %RES = BROWN  matches BROWN in the database
```

- Partial match

  When the LOOKUP validation attempts a *partial match*, the input string must match the leading characters of database entries of greater length, for example:

```
Input in %RES = BRO    matches BROWER
                               BROWN
```

- Partial match with a cutback

  When the LOOKUP validation attempts a partial match with a *cutback*, you specify the number of leading characters in the input string to use in performing the match. The DASL software truncates the input value to the number of characters specified by /CUTBACK before performing the partial match, for example:

```
Input in %RES = BROWN
LOOKUP/CUTBACK=2  means %RES is cutback to BR

Therefore, this LOOKUP matches  BROWER
                                BRANIF
                                BRINK
```

Lookup Lists

Lookup lists are numbered lists of existing values that satisfy the criteria of the LOOKUP validation. The list can contain a title, which is the legend from the data name, and additional data names specified in the Define Data Names Option. For example, in the demonstration system, you can perform a LOOKUP validation on NAME. The lookup list for NAME also displays the values for BIRTHDAY and ADDRESS. The title for the lookup screen (the legend) is:

```
Name                    Birthday          Address
```

The DASL software automatically displays a lookup list in any of the following circumstances:

- The entered value exactly or partially matches more than one existing value in the database.
- The LOOKUP validation includes a /LIST qualifier, and the user enters an asterisk (*) to list all existing values.
- The LOOKUP validation includes a /VERIFY qualifier.

To select a value from the list, enter the number that precedes the value. The DASL software repaints the calling screen and then displays the value in the field. To indicate that none of the values is the one desired, press Return or Exit.

LOOKUP Qualifiers

The LOOKUP qualifiers determine how the DASL software conducts a LOOKUP validation, and under what circumstances the DASL software displays a lookup list to a user.

/CUTBACK=N

N is a number that specifies the number of leading characters in an input string to use in a partial match. If you specify 4, the DASL software displays as partial matches all entries beginning with the first four characters in the entered value.

When searching through a NAME cross-reference global, the DASL software performs a NAME validation before proceeding with the partial match. The last name is cutback to the number of characters specified in the qualifier, and the first name is cutback to one character.

/EXACT

The characters in the input string must exactly match an item in the database. If no exact matches exist, the validation fails. The DASL software sets %FND to 0 and %MSG to an error message.

/LIST

The DASL software allows the user to enter an asterisk (*) to display all existing values on a lookup list. If you do not include /LIST, an entered asterisk causes the DASL software to display the message "List not allowed". You cannot use /LIST with KWIC cross-reference globals.

/NOPAINT

Normally the DASL software repaints the calling screen after a lookup list is displayed. You can use the /NOPAINT qualifier if you do not want to redisplay the calling screen. For example, you may want to branch to another screen when the user selects an entry from the lookup list. The LOOKUP validation sets the variable %FND and does not repaint the calling screen when %FND=0.

/VERIFY

Data is often not unique. For example, two persons can share the same name.

You can use the /VERIFY qualifier in conjunction with a second validation when you look up data that is possibly not unique from a cross-reference file or when you enter new data. The /VERIFY qualifier always displays a lookup list, even for a single exact match.

For example, if the individual name John Doe is an entry in your database, and you want to enter a different individual also named John Doe, you can use the following validation:

```
Validation:   LOOKUP/VERIFY ; NAME
```

The LOOKUP/VERIFY validation displays a lookup list of one entry: Doe, John. Press the carriage return to cause the LOOKUP/VERIFY validation to fail. The DASL software moves to the next validation, NAME. You can then enter the second individual's name, John Doe.

When you next look up Doe, John, two entries appear on your lookup list. You can add additional data names and a legend to the cross-reference definition in the data dictionary to help identify your data, for example:

```
Name              Date of Birth          Street Address
----              -------------          --------------
1. Doe, John            4-Sep-45             16 Hudson St.
2. Doe, John           16-Apr-66             2324 Main St.
```

/VERIFY=PARTIAL

The /VERIFY=PARTIAL qualifier is useful in fields where you want to enter a new code that is a subset of an existing code.

For example, you can use /VERIFY=PARTIAL to enter the code FISH in a database that already contains the code FISHING:

```
Validation:   LOOKUP/VERIFY=PARTIAL ; PATTERN 1U.9UN
```

The DASL software displays a lookup list: FISHING. Press the carriage return to cause the LOOKUP/VERIFY=PARTIAL validation to fail, and to move to the next validation, PATTERN 1U.9UN. The pattern validation succeeds, and FISH is accepted.

The DASL software always displays a lookup list if there are multiple partial matches. You must use the LOOKUP validation with the /VERIFY=PARTIAL qualifier to display a lookup list in the case of a single partial match.

LOOKUP Arguments

The LOOKUP argument form you use determines which global the DASL software accesses.

LOOKUP{/qualifiers}{SP,select criteria}

This syntax uses no data name, but assumes the data name from the current field. If the current data name is a key data name (that is, if the data name's reference in the Data Dictionary contains an asterisk as the last subscript), the DASL software looks up the value in the global you defined as the data name's reference in the Data Dictionary. If the current data name is a dependent data name, the DASL software looks up the value in the global you define as the data name's cross-reference global in the Data Dictionary.

LOOKUP{/qualifiers}{SPdata name{,select criteria}}

The DASL software looks up the value for the specified data name. Data names must either be key names or have a cross-reference defined for them. The DASL software looks up the value for key data names in the reference global and the value for dependent data names in the cross-reference global.

LOOKUP{/qualifiers}{SPglobal ref}

The DASL software looks up the value in the global reference you specify as an argument. The DASL software compares the input value with all nodes below the level of the node you specify.

Therefore, if you specify an unsubscripted global name, the DASL software searches all subscripted nodes of the global. If you specify a subscripted node, the DASL software searches all descendants of that node.

With this syntax, you cannot use additional data names in the lookup list.

LOOKUP Select Criteria

You can use the following select criteria with LOOKUP validation to modify or limit the search.

Data Names or Expressions

You can specify one or more data names after an initial comma as select criteria. The data names must be dependent on the LOOKUP data name, or the primary key of the LOOKUP data name, if a cross-reference is used. These select criteria take the following forms:

```
,data name

,data name=value

,data name'=value
```

You can specify multiple select criteria, separated by commas.

If you specify a data name as a select criteria, the DASL software selects only the values for the LOOKUP data name that also contain the current value of the select criteria data name. In the following example, the DASL software looks for PATNAM values which also have the current value of SEX. You must define the data name SEX at run time, when the LOOKUP validation is performed.

```
Validation:    LOOKUP PATNAM,SEX
```

If you specify a value for the select criteria data name, the DASL software only selects values for the LOOKUP data name that contain the specified value of the select criteria data name. In the following example, the DASL software only looks for PATNAM values which also have the value of F for SEX. In this case, LOOKUP checks the value of SEX in the database directly, as opposed to the previous example where SEX was defined.

```
Validation:    LOOKUP PATNAM,SEX="F"
```

If you specify a data name'=value as a select criteria, the DASL software only selects values of the LOOKUP data name that do not contain the specified value of the select criteria data name. In the following example, the DASL software only looks for PATNAM values which do not have the value of M for SEX.

```
Validation:    LOOKUP PATNAM,SEX'="M"
```

> **Note:** The DASL software only tests nonnull values in the application database. Null values always pass the test. Therefore, the select criteria SEX=F is true if SEX is F or null.

Execute Strings

You can also use an execute string passed in the DASL variable %XS as a select criteria. To execute a complex lookup operation like filtering, set %XS equal to an execute argument, for example:

```
SET %XS="D LINE^ROUTINE"
```

The DASL software executes the execute string upon each entry to be displayed in the lookup list. The data name that is specified in the LOOKUP validation is set to the value of each occurrence in the database. You can use the data name as a local variable in the execute string. If you are using a cross-reference file, the cross-reference key is also set.

Set the variable %MSG to a nonnull value to test for lookup success, for example:

```
SET %XS="I $P(^PAT(ID,""DEMO""),"";"",5)'=""F"" S  %MSG=1"
```

When using the %XS variable, keep these potential problems in mind:

- The LOOKUP utility always performs a KILL action on %XS upon exit. Therefore, you must define %XS for each LOOKUP validation you perform.
- If you define %XS and do not perform a LOOKUP validation, you must manually perform a KILL action on %XS. If %XS is not deleted, it causes side effects in fields other than the intended field.

To avoid these problems, set %XS in this way:

```
Validation:   DO SETXS^UTIL : LOOKUP/LIST PATNAM
```

Routine UTIL does the following:

```
UTIL ;
SETXS S %XS="I $P(^PAT(ID,""DEMO""),"";"",5)'=""F"" S  %MSG=1"
```

Returning %MSG=1 forces the first validation to fail and the DASL software proceeds to the next validation which is the LOOKUP validation.

Variables Used by the LOOKUP Validation

The DASL software uses the special variables %MSG and %FND to signal the success or failure of the LOOKUP validation.

- %MSG is an error indicator. When the LOOKUP validation fails, the DASL software returns an error message in %MSG.
- %FND indicates whether or not the user has selected an item from a lookup list. If the user selects an item from the lookup list, the DASL software sets %FND=1. If the user does not make a selection from the lookup list, the DASL software sets %FND=0.

  Note: On an exact match with only one match in the database, the DASL software sets %FND=1, but does not display a lookup list.

- For %FND to equal 1, %MSG must be equal to null.

Processing the LOOKUP Validation

When processing a LOOKUP validation, the DASL software uses the following logic:

1. The DASL software checks the input value entered against database entries.

   - If only one exact match exists (and the /VERIFY qualifier is not present), the DASL software sets %MSG to null and %FND=1. The LOOKUP validation is successful.
   - If one exact match exists and /VERIFY is present, or if several exact matches exist, the DASL software displays them on a lookup list. If the user makes a selection from the list, the DASL software sets %FND=1 and %MSG to null. The LOOKUP validation is successful.

2. If the user does not choose an item from the lookup list or if there are no exact matches, the DASL software performs a cutback if the /CUTBACK qualifier is specified, and attempts a partial match. Then the DASL software displays any matches on a lookup list.

   If the user makes a selection, the DASL software sets %FND=1 and %MSG to null. The LOOKUP validation is successful. If the user does not select an item from the list, the DASL software sets %FND=0 and %MSG to an error message. The LOOKUP validation fails.

3. If there are no matches, the DASL software sets %FND=0 and %MSG to an error message. The LOOKUP validation fails.

If the LOOKUP data name has a cross-reference global defined, LOOKUP validation also returns a value for the primary key and any pointers defined for that key. If the LOOKUP validation fails, the DASL software sets the value of the primary key to null.

Using the LOOKUP Validation in Batch Screens

When you use the LOOKUP validation in batch screens, the LOOKUP validation is equivalent to LOOKUP/EXACT. The DASL software never displays a lookup list. If there is more than one exact match, the DASL software sets %FND=0 and %MSG to an error message, and the LOOKUP validation fails.

Related Sections

Actions
Cross References, Chapter 1
Screen Driver Variables
Screen ADBOOK, Field NAME in DASL Demo

NAME

The NAME validation requires that users enter a name in the standard DASL name format.

Format

NAME

Comments

Keep the following points in mind when you use the NAME validation:

- You can abbreviate NAME as NM.
- Names are one of the DASL data types. To be valid, names must be in the following format:

  LASTNAME, FIRSTNAME{ MIDDLE{, TITLE}}

  where:

LASTNAME, FIRSTNAME, and MIDDLE	are strings starting with an alphabetic character optionally followed by alphabetic or punctuation characters
TITLE	is an optional string containing either alphabetic or punctuation characters (except a comma)

- The NAME validation and NAME data type require that all names stored in the database:

  — Have no trailing spaces before the comma

  — Have a single space following the comma

  Users, however, do not have to meet these criteria when entering data. Whenever a user enters a name, the DASL software performs a validation check. If the value in %RES is a valid name, the DASL software deletes trailing spaces before the commas and adds spaces after the commas.

  The DASL software then displays the converted name on the screen and, when directed, stores the converted form of the name in the database.

- You can use the Define Templates Option and the Define Data Names Option on the Data Dictionary Menu to declare a data type for a template or data name. If you declare a NAME data type for a template or data name, the validation for the template or data name defaults to NAME.

Related Sections

Define Data Names Option, Chapter 1
Define Templates Option, Chapter 1
Screen Driver Variables

Example

The following example shows the field definition for the NAME field of the ADBOOK screen. This field prompts the application user to enter a name in the standard name format. The NAME validation checks for correct name format.

```
Field Name: NAME      Description: Enter a new or old name.
Line: 5   Column: 5    Justify: L    Data Name: NAME  Data Length: (30)
Prompt: (Name: )
Help Text: (Enter a name as LAST, FIRST MI, TITLE)Help Screen: (HNAME)
Attributes: (LCASE)  (LCASE)
Validations: LOOKUP/LIST/VERIFY ; NAME  (LOOKUP/VERIFY/LIST ; NAME)
------------------------------ Actions -------------------------------
!  If lookup finds the entry, the cross-reference pointer will be
!  returned.  If it's not found, or no entry is selected from the
!  lookup list, ID will be equal to null.
!  For a new entry:  EXIT EQ to be certain the user "visits" all fields
!  the required fields, and branch to field STREET.
!  %FND is a truth value set both by the lookup and the lock command
NXTFLD CLEAN:%RES="" ; EXIT EQ ; NXTFLD STREET:ID=""
LOCK/TIMEOUT=9 ^ADBK("DATA",ID,"DEMO") ; NXTFLD EVAL:%FND
ERROR "This record is currently locked by another user."
----------------------------------------------------------------------
```

NUMERIC

The NUMERIC validation requires that users enter numeric values.

Format

NUMERIC{ SP low value{,high value{,decimal}}}

where:

low value	is an expression that specifies the only acceptable numeric value or the lower limit of a range of acceptable numeric values
high value	is an expression that specifies the upper limit of a range of acceptable numeric values
decimal	is an expression that specifies the maximum number of decimal digits acceptable

Comments

Keep the following points in mind when you use the NUMERIC validation:

- You can abbreviate NUMERIC as NU.
- Without an argument, the NUMERIC validation requires that users enter numeric values, the digits 0 through 9, or the decimal point (.).

  With one argument, the NUMERIC validation requires that users enter numeric data greater than or equal to the numeric values specified by the argument.

  With two arguments, the NUMERIC validation requires that users enter numeric data in the range specified by the arguments.
- If you do not specify **decimal**, the DASL software accepts any number of decimal digits. If you specify zero (0) as **decimal**, the DASL software accepts no decimal digits.
- You can use the Define Templates Option and Define Data Names Option on the Data Dictionary Menu to declare a data type for a template or data name. If you declare a NUMERIC data type, the template or data name validation defaults to numeric with a **low value** argument of 1.

Related Sections

Data Types, Chapter 1
INTEGER

Example

The following example specifies decimal data in the range 1 to 1000 with two decimal digits.

```
Validation:   NUMERIC 1,1000,2
```

PATTERN

The PATTERN validation requires that user input exactly match the VAX DSM pattern match code you specify in the argument.

Format

PATTERN [SP]**pattern**

where:

pattern is an expression specifying a valid VAX DSM pattern code

Comments

Keep the following points in mind when you use PATTERN:

- You can abbreviate PATTERN as PA.
- DASL pattern codes are identical to those you can construct in DSM. Table 2–9 shows the pattern code characters you can use.

Table 2–9 Pattern Code Characters

Character	Specifies
N(umeric)	Any one of the digits from 0 through 9 inclusive
U(ppercase)	Any one of the 26 uppercase letters from A to Z
L(owercase)	Any one of the 26 lowercase letters from a to z
A(lphabetic)	Any one of the 26 uppercase or 26 lowercase letters
P(unctuation)	Any one of the 33 control characters, including a blank space
E(verything)	Any one of the ASCII characters

- The rules for pattern codes follow those for the VAX DSM Binary PATTERN MATCH Operator. To make a pattern, enter a number before each pattern code character to specify the number of characters necessary for a match or enter a period (.) to specify that any number of matches is allowed.

  For example, the pattern 4A2N.E specifies that input must be four alphabetic characters and two numbers followed by any number of characters of any type.

  You can use the period (.) between two numbers to specify a range of matches. For example, the pattern 1.4N specifies that input must be between 1 and 4 numbers.

You can also use a literal in quotation marks to indicate an exact character match is necessary.

- Unlike VAX DSM, you cannot use an AND or an OR operator with DASL pattern match codes. For example, you can write the following pattern match statement using VAX DSM:

  ```
  PA?1U.6UN!(1A.6UN)
  ```

  However, in the DASL software you must define the same pattern match twice, as follows:

  ```
  PA 1U.6UN ; PA 1A.6UN
  ```

  The DASL PATTERN validation works from left to right, and does not contain an AND or an OR operation. If you define a pattern match with an AND or an OR operator, the DASL software ignores it.

- The PATTERN validation can test for specific patterns of alphanumeric codes, such as social security numbers, telephone numbers, or site-specific patient or employee identifications.

Related Sections

Binary PATTERN MATCH Operator (*VAX DSM Language Reference Manual*)
Screen Driver Variables

Example

The following example requires that the data value follow the format for a social security number.

```
Validation:    PATTERN 3N1"-"2N1"-"4N
```

TABLE

The TABLE validation requires that user input exactly match one of a list of values.

Format

TABLE SP **table**

where:

table is a list of acceptable values or a reference to a global node containing a list of acceptable values

Comments

Keep the following points in mind when you use TABLE:

- You can abbreviate the TABLE validation as TB.
- The TABLE validation requires that input match one of the table values exactly (character by character) for the validation to be successful. During validation, the DASL software evaluates each table value in left-to-right order. **The DASL software considers the validation successful only when it finds a table value that matches character for character with the user's response.** The DASL software then stores the full table value in %RES.

  Therefore, a user must enter the full word NORTH to match the first entry in the table validation, for example:

  ```
  Validation:   TABLE NORTH,EAST,SOUTH,WEST
  ```

  Use the TABLEX validation to allow for partial matches, for example: N, NO, or NOR for NORTH.
- Use the TABLEM validation to check a user entry that consists of multiple items. The TABLE validation checks only single-item entries.
- You can use a list of values as the TABLE argument. The values can be single characters or full words. Use commas to separate each acceptable value from those that precede or follow it, for example:

  ```
  Validation:   TABLE S,E,Q
  ```

  Do not enclose the values listed with quotation marks. The values listed in a TABLE argument must be values and not expressions that evaluate to a value.

- You can use a global node that contains a list of values as the TABLE argument, for example:

```
Validation:   TABLE ^ANSWER
```

  The node listed must contain a list of values in the appropriate TABLE format, for example:

```
SET ^ANSWER="S,E,Q"
```

Related Sections

TABLEM
TABLEX

Example

This example shows the TABLE validation from the SEQ field in the screen ADBOOK. The SEQ field is the "Save, Edit, or Quit:" prompt.

```
Field Name: SEQ       Description: Save, Edit or Quit
Line: 20  Column: 40   Justify: C    Data Name:      Data Length: 1
Prompt: "Save, Edit, or Quit: "
Help Text: "Enter "S" to save, "E" to edit or "Q" to quit."
Attributes: EXIT CLEAN ; REQUIRED ; DEMAND
Validations: TABLE S,E,Q
------------------------------ Actions ----------------------------
NXTFLD SAME:%RES="Q" ; NXTFLD FILE:%RES="S"
RESET SEQ ; EXIT SEQ ; NXTFLD EDNAME
-------------------------------------------------------------------
```

TABLEM

The TABLEM validation checks user input that consists of multiple items. The TABLEM validation requires that each input item in a multiple entry exactly match one of a list of values.

Format

TABLEM SP **list**

where:

list is a list of acceptable values separated by commas or a reference to a global node containing a list of acceptable values

Comments

Keep the following points in mind when you use TABLEM:

- You can abbreviate the TABLEM validation as TBM.
- The TABLEM validation requires that each input item in a multiple entry match one of the table values exactly (character by character) for the validation to be successful. During validation, the DASL software evaluates each table value in left-to-right order.

  The DASL software considers the validation successful only when it finds a table value that matches character for character with the user's response. The DASL software then stores the user input in %RES.
- Use the TABLE validation to check only a single user entry.
- Use the TABLEX validation to allow for partial matches. However, you can use TABLEX only with single-item input.
- You can use a list of values as the TABLEM argument. The values can be single characters or full words. Use commas to separate each acceptable value from those that precede or follow it, for example:

  ```
  Validation:   TABLEM CA,MA,CT,ME,RI
  ```

  Do not enclose the values listed with quotation marks. The values listed in a TABLEM argument must be values and not expressions that evaluate to a value.
- You can use a global node that contains a list of values as the TABLEM argument, for example:

  ```
  Validation:   TABLEM ^ANSWER
  ```

  The node listed must contain a list of values in the appropriate TABLEM format, for example:

  ```
  SET ^ANSWER="AL,AK,AZ,AR,CA,...VA,WA,WV,WI,NY"
  ```

Related Sections

TABLE
TABLEX

Example

For example, the following TABLEM validation sets up a table of values that correspond to abbreviations of state names in the "STATE" field.

```
Validation:   TABLEM CA,RI,MA,CT,VT,ME
```

Some valid entries for the table created by the preceding validation are:

```
CA,RI,ME

VT,CT

RI,MA,VT,CT,ME
```

TABLEX

The TABLEX validation requires that user input partially match one of a list of values.

Format

TABLEX SP **table**

where:

table is a list of acceptable values or a reference to a global node containing a list of acceptable values

Comments

Keep the following points in mind when you use TABLEX:

- You can abbreviate the TABLEX validation as TBX.
- The TABLEX validation requires only a partial match for the validation to be successful. During validation, the DASL software evaluates each table value in left-to-right order and reports a successful validation when it finds a table value that begins with the same characters as the user's response. The DASL software then stores the full table value (rather than the partial value entered) in %RES.

  Therefore, a TABLEX validation is successful if a user enters C, CON, or CONT where one of the TABLEX table values is CONTINUE.
- You can use a list of values as the TABLEX argument. The values can be single characters or full words. Use commas to separate each acceptable value from those that precede or follow it, for example:

```
Validation:   TABLEX NORTH,SOUTH,EAST,WEST
```

  Do not enclose the values in quotation marks. The values in a TABLEX argument must be values and not expressions that evaluate to a value.
- You can use a global node that contains a list of values as the TABLEX argument, for example:

```
Validation:   TABLEX ^ANSWER
```

  The node listed must contain a list of values in the appropriate TABLEX format:

```
SET ^ANSWER="NORTH,SOUTH,EAST,WEST"
```

Related Sections

TABLE
TABLEM

TIME

Without an argument, the TIME validation requires that users enter a time in one of the DASL time formats. With an argument, the TIME validation requires that users enter a time in the range specified by the argument.

Format

TIME{ SP early{,late}}

where:

early	is an expression that evaluates to the earliest (or only) acceptable time
late	is an expression that evaluates to the latest acceptable time

Comments

Keep the following points in mind when you use the TIME validation:

- The abbreviation for the TIME validation is TM.
- TIME is also one of the DASL data types.
- The DASL software recognizes input as a valid time if it matches any of the formats shown in Table 2–10.

Table 2–10 Standard Time Input Formats

Format	Example	Explanation
hh:mm (AM or PM)	12:45 AM	External standard format
hhmm	1301	Military format (equals 1:01 PM)
N	N	Present time (now)
N+n	N+10	Present time plus n minutes If current time is 1:10, represents 1:20
N–n	N–10	Present time minus n minutes If current time is 1:10, represents 1:00

TIME can alter the user's input and display the transformed data. This occurs, for instance, if the user types N.

- The DASL software has two standard time formats:
  - — External
  - — Internal

The first item in Table 2–10 shows the external standard format that the DASL software displays on the screen. When a user enters a time in any of the acceptable input formats, the DASL software converts the time to the external standard. The DASL software stores the external time in DASL variables %RES and %TMX and displays the external time.

The internal format is the one maintained in the VAX DSM special variable $HOROLOG. The DASL software uses this internal format for data storage. After a user enters a time, the DASL software stores the internal version of the time in the DASL variable %TMI and in the field data name. When you file the time, the DASL software stores it in the database in internal format.

Whenever the DASL software must redisplay the time, it transforms the time to external format.

- You can change the standard external time format for your applications through the Site Parameters Option on the Development Environment Menu. Table 2–11 shows the characters you can use with punctuation to produce a standard external time format.

Table 2–11 Standard External Time Characters

Character	Explanation
H	Displays the time in hours (1–12)
M	Displays the time in minutes
N	Displays the time in hours (1–24)
A	Displays the time with AM or PM appended

The following examples show some valid external time formats:

```
H:M A       3:56 PM

H:M         3:56

N:M         15:56
```

- You can use the Define Templates and Define Data Names Options on the Data Dictionary Menu to specify a data type for a template or a data name. If you specify a TIME data type for a template or a data name, the validation automatically defaults to TIME whenever you use that template or data name.

Related Sections

Data Types, Chapter 1
Define Data Names Option, Chapter 1
Define Templates Option, Chapter 1
Screen Driver Variables
Site Parameters Option, Chapter 5

Example

The following example allows only time values between 9:00 AM and 5:00 PM.

```
Validation:   TIME 9:00 AM,5:00 PM
```

XECUTE

XECUTE is a validation and an action. The XECUTE validation executes the line of VAX DSM statements you specify as an argument.

Format

XECUTE [SP]**argument{:postcond}**

where:

argument is a string of DSM code enclosed in quotation marks

postcond is an optional postconditional expression

Comments

Keep the following points in mind when you use the XECUTE validation:

- The abbreviation for the XECUTE validation is XE.
- You can use XECUTE as a screen action, a field action, or a validation. The XECUTE validation executes the VAX DSM statements that are the value of its argument. Control returns to the point immediately following the XECUTE argument.
- A DASL XECUTE action becomes an XECUTE statement at run time. The argument to the XECUTE statement is a quoted string.
- When you use XECUTE as a validation, you cannot append a postconditional expression to the XECUTE argument.
- You can use XECUTE as a validation to handle specialized validation and error processing. The VAX DSM statements in the argument can test the DASL variable %RES (which contains the current data value to be validated). If the validation fails, the statements then set %MSG to an appropriate user error message.
- You must set %MSG within the XECUTE argument to return a message for errors or events.

For more information on XECUTE as an action, see the "XECUTE" section in this chapter.

Related Sections

DO
Screen Driver Variables
XECUTE

Actions

Actions are commands that you use to specify special processing operations for a data screen or a field.

Explanation

The DASL software has two types of actions:

- Screen actions
- Field actions

Screen actions are special processing operations for a data screen. You specify screen actions in the first screen of the Define Data Screens Option.

Field actions are special processing operations for a field. You specify field actions in the second screen of the Define Data Screens Option.

Some actions are both screen and field actions.

Table 2–12 lists DASL actions and their meanings.

Table 2–12 DASL Actions

Action	Argument	Abbrev	Use
ASSIGN{/qualifier}	**field=expression{:postcond}**	AS	Gives value of expression to a specified field.
COUNT	**data name{:postcond}**	CN	Increments a counter in the database.
DISPLAY{/qualifier}	**{field{:postcond}}**	DS	Modifies the display characteristics for the specified field.
DO{NP}	**reference{:postcond}**		Calls the screen or VAX DSM routine specified in the argument and returns to the point immediately after the argument. DO repaints the display on return. DONP does not repaint the display on return.
ERASE	**{field{:postcond}}**	ERS	Erases the specified fields.
ERROR	**expression{:postcond}**	ERR	Rings the terminal bell, displays the expression as an error message on help line, and causes reprocessing of the current field.

Table 2–12 (Cont.) DASL Actions

Action	Argument	Abbrev	Use
EVAL	**namelist[:postcond]**	EV	Displays stored values for specified data names or fields.
EVALS/KEY=name	**namelist[:postcond]**	EVS	Retrieves and displays stored values in scroll region.
EXIT	**field[:postcond]**	EX	Specifies the exit field that allows clean exit from a screen and can include decision prompts.
FILE	**namelist[:postcond]**	FI	Stores values in the database.
FILES/KEY=name	**namelist[:postcond]**	FIS	Stores values collected in scroll region.
HANG	**seconds[:postcond]**	HA	Pauses for the specified number of seconds before proceeding.
KILL[/qualifiers]	**namelist[:postcond]**	KL	Deletes the specified local or global variables.
LOCK[/timeout=n]	**argument[:postcond]**	LK	Locks all global or local variables specified.
LOG	**expression[:postcond]**		Adds a message to the log array in batch screen processing.
LOGDMP			Transcribes the information in a log array to a VMS sequential file.
LOGOFF		LF	Writes an end-of-transaction record to the journal file.
LOGON		LN	Writes a start-of-transaction record to the journal file.
LOOKUP[/qualifiers]	**[,select criteria]**	LU	Compares input at the current field with values for the field's data name in the database.
LOOKUP[/qualifiers]	**data name[,select criteria]**	LU	Compares input at the current field with values for the specified data name in the database.
LOOKUP[/qualifiers]	**global ref**	LU	Compares input at the current field with values in the specified global.
MUMPS	**DSMcode**	MU	Executes the line of VAX DSM statements you specify as an argument.
NXTFLD	**field[:postcond]**	NF	Transfers control to the specified field.

Table 2–12 (Cont.) DASL Actions

Action	Argument	Abbrev	Use
NXTSCN	**screen{:postcond}**	NS	Transfers control to the specified screen.
QUIT	**{:postcond}**		Causes termination of field action and screen processing.
REPAINT	**{:postcond}**	RP	Causes the entire video display to repaint.
RESET	**fieldlist{:postcond}**	RS	Sets all data values in the specified fields to their default values.
SCROLL{/qualifiers}	**{:postcond}**	SC	Handles scroll region processing.
SET	**argument{:postcond}**		Assigns values to variables.
TCOMMIT{qualifiers}		TC	Completes (commits) a transaction that began with the TSTART action. Writes an end-of-transaction record to the journal file.
TSTART{TIMEOUT=n}	"name"	TS	Starts a logical transaction for transaction processing and writes a start-of-transaction record to the journal file.
UNLOCK	**argument{:postcond}**	ULK	Unlocks elements locked with a DASL LOCK action.
XECUTE	**DSMcode{:postcond}**	XE	Executes the line of VAX DSM statements you specify as an argument.

Comments

Keep the following points in mind when you use DASL actions:

- The DASL software executes screen actions after it displays the data screen.
- The DASL software executes field actions for action-only or display-only fields immediately following field invocation.
- The DASL software executes field actions for fields that solicit user input immediately after successfully validating user data.
- You can use all actions except ERASE and ERROR as screen or field actions. You can use ERASE and ERROR only as field actions.
- If an action uses a postconditional expression, the DASL software performs the action only if the postconditional expression is true.

The following sections describe most actions. Because LOOKUP is used primarily as a validation, see the "LOOKUP" section for more information about the LOOKUP action.

Related Sections

Define Data Screens Option
Fields
Screen Attributes
Validations

ASSIGN

You use the ASSIGN action to assign a value to a specified field.

Format

ASSIGN{/qualifier} SP field=expression{:postcond}

where:

/qualifier	is an optional qualifier, which can only be: /INTERNAL — assigns the value of expression to the internal value of the data name associated with the field
field	is a field name
expression	is an expression that evaluates to a value
postcond	is an optional postconditional expression

Comments

Keep the following points in mind when you use the ASSIGN action:

- You can abbreviate the ASSIGN action as AS.
- When you assign a value to a field, the DASL software displays that value as the field data. If the field is a demand field, you must issue a DISPLAY action to make the value visible.
- If the field uses a data name, the DASL software assigns the value to the data name.
- When you use the ASSIGN action without a qualifier, the DASL software assigns the value of the expression to the external value of a data name associated with the specified field. You must use the /INTERNAL qualifier with the ASSIGN action to assign the value of the expression to the internal value of the data name associated with the specified field.
- Fields can also receive values through user input, the DEFAULT field attribute, the EVAL action, or the RESET action.

Related Sections

DEFAULT
DEMAND
DISPLAY
EVAL
RESET
Screen ADBOOK, Field BTHDAY on DASL Demo
Transforms, Chapter 1

Example

In this example, the DASL software assigns the value %DTI to the internal value of the AGE data name. %DTI is the internal value of the birth date that the user enters in the BTHDAY field. The data name AGE includes an output transform: the user enters a birth date as the value of BTHDAY, then the DASL software executes the output transform to produce an age in years as the external value of AGE.

```
            Screen: ADBOOK          Field Name: BTHDAY

------------------------------Actions-----------------------------------
!  DATE validation returns %DTI as date in internal ($H) format.
ASSIGN/INTERNAL AGE=%DTI :%RES'=""&(%RES'=%OLD)
NXTFLD PHHOME
------------------------------------------------------------------------
      Edit, edit Actions, Copy, View defaults, Delete, or Quit:____
```

COUNT

The COUNT action locks a specified data name, increments the data name, and then unlocks it. The COUNT action sets the local value of the data name equal to the incremented database value.

Format

COUNT SPdata name{:postcond}

where:

data name is the data name whose value you want to increment

postcond is an optional postconditional expression

Comments

Keep the following points in mind when you use the COUNT action:

- You can use the COUNT action to increment a counter whose value is then used as a primary key in a file. Use the SET command to assign the value stored in the counter data name to the primary key.
- The LOCK action locks all descendants of a specified data name. When you define a counter, you should place it in a separate global node, apart from the global node that contains the record of data. You can then lock the counter for update without locking the entire database.

Related Sections

FILE
LOCK
SET
Screen ADBOOK, Field FILE in DASL Demo

Example

In this example, the COUNT field action increments the value of IDNEXT. The SET command assigns the incremented value to the primary key ID. IDNEXT is stored in a separate global from ID.

```
----------------------Actions---------------------------------
LOGON
! Generate new primary key if its a new entry.
! COUNT locks and increments the "counter node" in the global.
COUNT IDNEXT:'ID ; SET ID=IDNEXT:'ID
FILE NAME,STREET,CITY,STATE,ZIP,BTHDAY,PHHOME,PHWORK,PHOTH1
--------------------------------------------------------------
```

DISPLAY

The DISPLAY action modifies the default video display characteristics for a field.

Format

DISPLAY{/qualifier}{ SP field{:postcond}}

where:

/qualifier is a display qualifier, which can be any of the following:

/BELL — display with bell

/BLINK — display in blinking characters

/BOLD — display in boldface

/REVERSE — display in reverse video

/UNDER — display underlined

/NORMAL — display normally

field is the optional name of the field to modify

postcond is an optional postconditional expression

Comments

Keep the following points in mind when you use the DISPLAY action:

- You can abbreviate the DISPLAY action as DS. You can abbreviate its qualifiers as follows:

Qualifier	Abbreviation
/BELL	/BE
/BLINK	/BL
/BOLD	/BO
/REVERSE	/REV
/UNDER	/UN
/NORMAL	/NO

- When you use the DISPLAY action as a screen action, you must specify a **field** argument. When you use the DISPLAY action as a field action, you do not have to specify a **field** argument.
- If you use the DISPLAY action with a **field** argument, the DASL software sets the display characteristics of the field you specify. If you use DISPLAY without an argument, the DASL software sets the display characteristics of the current field.

- The value affected by the DISPLAY action depends on the data length of the field. If the field has a data length of zero (0), the DISPLAY qualifiers apply only to the prompt. If the field has a data length of one or greater, the DISPLAY qualifiers apply only to the value in the field.

  For example, if the field MESSAGE has a data length of zero and a prompt of "Record not found!", then the action DISPLAY/BLINK/BOLD causes the prompt to blink in boldface. If the field RESULT has a data length of 15 and a prompt of "Exam result", the DISPLAY/BLINK/BOLD action causes any value in the field to blink in boldface.

- You can display action-only fields with prompts in their own action lines by using a DISPLAY action without a field argument.

- DISPLAY and its qualifiers perform the same actions as the BLINK, BOLD, UNDER, and REVERSE video attributes. However, the attributes are permanently associated with the field while the DISPLAY actions apply only to the current branching to the field.

  If you use one of the video attributes, the DASL software displays the field with that attribute each time the field is branched to. If you use the DISPLAY action, the DASL software displays the field with the specified video attributes at the first display, but does not show the field with those video characteristics on subsequent displays unless you reissue the DISPLAY action.

- Demand fields display on the screen when they are branched to with a NXTFLD command or displayed with a DISPLAY action. When a field is displayed with a DISPLAY action, it is displayed until erased with an ERASE action.

- You can append a postconditional expression to a DISPLAY action with an argument. The DASL software executes the DISPLAY action only if the postconditional is true.

  You cannot append a postconditional expression to a DISPLAY action without arguments.

Related Sections

BLINK
BOLD
ERASE
REVERSE
UNDER

Example

The following example shows actions in the field PULSE from a clinical application. PULSE records pulse rates. If the value entered is greater than 120 or less than 50, then the DASL software displays the value blinking in boldface.

```
------------------------ Actions -------------------------

ERROR "Required field.":%RES=""&(TMTYPE="")
DISPLAY/BLINK/BOLD PULSE:(%RES<50)!(%RES>120)
NXTFLD BPRES
------------------------------------------------------------
```

DO

The DO action calls the screen or VAX DSM routine specified in the argument.

Format

DO [SP]reference{:postcond}

where:

reference is an entry reference to a VAX DSM routine or the name of a DASL screen

postcond is an optional postconditional expression

Comments

Keep the following points in mind when you use DO:

- You can use DO as a screen action or a field action. As an action, the DO **reference** can be either:

  — A VAX DSM routine entry reference

  — A DASL screen name

  DSM evaluates and executes the routine or screen specified in the argument. DSM returns control to the point immediately following the DO argument and repaints the screen.

  You can use parameter passing with the DO action. See the *VAX DSM Language Reference Manual* for more information about parameter passing and entry references to routines.

- The DASL software does not repaint the screen if you use the DO action to call a VAX DSM routine.
- If the DO action calls a VAX DSM routine and the validation fails, the routine should then set %MSG to an appropriate user error message.

Related Sections

DO
DONP
Appendix B, Entry Points
REPAINT

Examples

You can use the DO action to call a routine. If the routine destroys the screen display, use the REPAINT action to repaint the display as follows:

```
DO TAG^ROUTINE
REPAINT
```

You can use the DO action to call a DASL screen as follows:

```
DO SCREEN_NAME
```

You can use the DONP action to prevent the screen from repainting. Then, quit the screen with the QUIT action as follows:

```
DONP SCREEN_NAME
QUIT
```

DONP

The DONP action calls the DASL screen or VAX DSM routine specified in the argument but does not repaint the display on return.

Format

DONP SP**reference{:postcond}**

where:

reference	is an entry reference to a VAX DSM routine or the name of a DASL screen
postcond	is an optional postconditional expression

Comments

Keep the following points in mind when you use the DONP action:

- The DONP **reference** can be either:

  — A VAX DSM routine entry reference

  — A DASL screen name

  The DASL software executes the routine or screen specified in the argument. On completion, the DASL software returns control to the point immediately following the DONP argument without repainting the screen.

  See the *VAX DSM Language Reference Manual* for more information on entry references to routines.

- Use the DONP action to avoid visual distraction and unnecessary repaint operations. For example, use the DONP action when you are returning to a screen on which there is no further user interaction before the screen exits.

Related Section

DO

ERASE

The ERASE action erases a field from the screen.

Format

ERASE{ SPfield{:postcond}}

where:

field is the name of the field to erase

postcond is an optional postconditional expression

Comments

Keep the following points in mind when you use ERASE:

- You can abbreviate the ERASE action as ERS.
- When you use ERASE as a screen action, the **field** argument is required. The DASL software erases the field you specify.
- When you use ERASE as a field action, the **field** argument is optional. If you include an argument, the DASL software erases the field you specify. If you do not include an argument, the DASL software erases the current field.
- When you erase a field without a data name, the DASL software removes the field from the display and sets the value for the field to null. If the field uses a data name, the DASL software sets the data name to null, or to the default data name value (as assigned previously through the DEFAULT action).
- When you erase a field in which you set the video display characteristics with the DISPLAY action, you lose those video characteristics. The DASL software does not display those video characteristics on subsequent branching to the field unless you reissue the DISPLAY action with the same video display characteristics.
- When you erase a field in which you set the video display characteristics with a field attribute (such as BLINK, BOLD, or REVERSE), you do not lose those video display characteristics. Because the attributes are permanently attached to the field, the DASL software does display them on subsequent branching to the field.
- You can append a postconditional expression to ERASE with an argument. The DASL software erases the field only if the postconditional expression is true.
- You cannot append a postconditional expression to an ERASE action without an argument.

Related Sections

DEFAULT
DEMAND
DISPLAY
RESET

Example

In the following example, the actions of field RECHECK erase the field EXAM.

```
------------------------ Actions -------------------------
ASSIGN RECHECK="N" ; NXTFLD DELETE:%RES="Y"
ERASE EXAM:%RES=""; NXTFLD RESTART
----------------------------------------------------------
```

ERROR

The ERROR action rings the terminal bell and displays an error message you specify on the standard display help line.

Format

ERROR SP**expression{:postcond}**

where:

expression	is an error message enclosed in quotation marks
postcond	is an optional postconditional expression

Comments

Keep the following points in mind when you use the ERROR action:

- You can abbreviate the ERROR action as ERR.
- You can use the ERROR action to trap application-specific error conditions such as incorrectly formatted input.

  On encountering an ERROR action with a true postconditional expression or no postconditional expression, the DASL software stops processing field actions and displays the expression you use as an ERROR argument in the help line. Then, the DASL software reprocesses the current field.
- When you specify an ERROR action in an action-only field, the DASL software displays the warning message you have specified as the ERROR argument and continues processing. The DASL software does not reprocess the current field.

Related Section

Fields

Examples

The following example uses a postconditional with the ERROR action. If a user enters any string containing R or E, the DASL software displays the ERROR argument and erases the current value of the field. The DASL software then returns control to the same field to prompt for the information again.

See the *VAX DSM Language Reference Manual* for information about the Binary CONTAINS operator ([) used as a test in the postconditional expression.

```
------------------------ Actions -------------------------
ERROR "Illegal Operation!":%RES["R"!(%RES["E") ; !ERASE
----------------------------------------------------------
```

The following example shows an ERROR action without a postconditional expression. In this case, the ERROR action is in an action-only field to which the DASL software transfers control only when a user does not select an item from a scroll region. This action-only field displays the error message and reinitializes the scroll region by deleting the scroll array and using the SCROLL/FILL action to create an empty scroll region.

```
------------------------ Actions -------------------------
ERROR "No item selected!"
KILL ^DATG($J,"S",%L(0)) ; SCROLL/FILL ; NXTFLD FIRST
----------------------------------------------------------
```

EVAL

The EVAL action retrieves and displays data values from your application database.

Format

EVAL [SP]**namelist{<pointer chain>}{:postcond}**

where:

namelist	is a list of one or more field or data names
pointer chain	is a chain of one or more pointers to additional fields, data names, or other pointers
postcond	is a postconditional expression

Comments

Keep the following points in mind when you use EVAL:

- You can abbreviate the EVAL action as EV.
- You can use field names and data names in the same **namelist**.

  Each field name you specify must have a data name associated with it. The DASL software retrieves the appropriate value for the data name and displays it in the field. The DASL software stores the value in a local variable corresponding to the data name.

  If you specify a data name, you must precede the data name with a percent sign (%). The DASL software retrieves the appropriate value for the data name and stores the value in a local variable corresponding to the data name.

  Therefore, to retrieve a value for the data name PHONE, enter the following:

  ```
  EVAL %PHONE
  ```

  The DASL software stores the retrieved phone number in the local variable PHONE.
- When you use the EVAL action, the DASL software uses the reference, piece, extract, and output transform information in the Data Dictionary for the data name you specify or the data name associated with the field you specify to retrieve the appropriate value.
- You can use pointers with the EVAL action to traverse a chain of data names and retrieve dependent data from records for which the pointer is used as a key. Pointers can point to fields, data names, and other pointers.

Precede the pointers by a left angle bracket (<). The DASL software traverses the pointers from right to left. The following statements are examples of valid EVAL statements using pointer syntax:

```
EVAL Field1,Field2<PADDN1

EVAL (Field1,Field2)<PADDN1

EVAL Field1<PADDN1<PADDN2

EVAL Field1,%DDN,Field2<PADDN1,Field3<PADDN2
```

- When executing an EVAL action, the DASL software tests the existence of the global node for the first name in **namelist**. The DASL software applies the $DATA function to that global node and uses %FND as a success flag.

  If $DATA returns 1 or 11, indicating that the global node has a value, the DASL software assigns %FND a value of 1 and processes any other names in **namelist** without performing an existence test. If any nodes for other names in **namelist** do not exist, the DASL software assigns those names a null value.

  If $DATA returns a 0 or 10, indicating that the global node does not have a value, the DASL software assigns %FND a value of 0 and gives the first name a null value. The DASL software does not evaluate any other names in **namelist**.

- You use the EVAL action most often when you want users to view or edit existing data. The process you follow to allow for data editing is as follows:

  1. The user supplies a key data value (such as a name) in the first field on the screen.
  2. Your application performs a validation (such as a LOOKUP) to determine if the name already exists in the database.
  3. If the name exists, your application performs an EVAL action on several fields to retrieve dependent data associated with the key data name and display them on the screen.
  4. The user can change any item.
  5. When the user reaches the end of the screen, your application performs a FILE action to save the changes and update the database.

  You can also use the EVAL action when you only want to display data. For example, if a user must fill in several screens of information about a person for a personnel file, you can use the EVAL action to display certain pieces of information, such as name and employee number, on all the screens as a memory aide.

- The EVAL action displays field data as the fields appear on the screen in a left-to-right, top-to-bottom order no matter what order you specify the field or data names in the **namelist**.
- You can improve performance by designing your database with items you normally evaluate together as pieces of the same node. In that way, you can lessen the number of disk accesses that DSM makes to retrieve the data.

Related Sections

ASSIGN
$DATA Function (*VAX DSM Language Reference Manual*)
EVALS
FILE
FILES
LOOKUP
Pointers, Chapter 1
RESET
Screen Driver Variables
Screen ADBOOK, Field EVAL in DASL Demo

Examples

The following example comes from the EVAL field of the Define Data Screens Option on the Screen Driver Menu. If the user enters the name of an already-defined screen, the DASL software retrieves the data-name values for that screen and enters them into the fields associated with those data names.

```
------------------------- Actions --------------------------
EVAL DES,GRP,MAP,FL,LL,ATT1,ACT1,ACT2,ACT3 ; NXTFLD MENU
------------------------------------------------------------
```

The following example retrieves data for the fields NAME and PHONE, and the data name DOB. This example then uses the value of DOB to assign the age in years to the field AGE.

```
------------------------- Actions --------------------------
EVAL NAME,PHONE,%DOB
ASSIGN AGE=$H-DOB\265.25:DOB'=""
------------------------------------------------------------
```

EVALS

EVALS is a screen and a field action used in scroll screens. You use EVALS to build scroll arrays from a global subtree in the application database and to display the data on the terminal.

Format

EVALS{/INPUT}/KEY=name SP namelist{<pointer chain}

where:

/INPUT	is an optional qualifier that instructs the DASL software to place a blank line as the last entry in a data entry scroll array
/KEY	is a required qualifier that indicates that the argument to EVALS is a key data name or the name of a field that contains a key data name
name	is a scroll region counter (a key data name with a number sign (#) in the reference subscript) or the name of a field that contains an (*) key data name (**Precede each scroll region counter by a percent sign (%).**)
namelist	is a list of one or more dependent data names or fields that contain dependent data names (**Precede each data name with a percent sign (%).**)
pointer chain	is a chain of one or more pointers to additional fields, data names, or other pointers

Comments

Keep the following points in mind when you use the EVALS action:

- You can abbreviate the EVALS/KEY action as EVS/K.
- The EVALS action evaluates a set of data names or fields and builds the scroll array. As part of the EVALS action, the DASL software performs the following actions:
  - — Defines pointers
  - — Performs date and time conversions
  - — Stores DEMAND/DISPLAY flags as it builds the array

  The EVALS action sets %FND=0 to indicate failure, or %FND=N (where N is the number of entries in the scroll region) to indicate success.

- The argument to the EVALS action is a key data name followed by a space and then by a list of fields separated by commas. The EVALS action performs a $ORDER through the entire subtree specified by the key data name. All fields specified must be within the scroll region and must be dependent on that key.

  The key type determines the format of the generated EVALS subroutine. A key must have either a number sign (#) or an asterisk (*) in the last subscript position of the reference field in the Data Dictionary.

  If the key has a number sign (#) in the last subscript position, it is a counting type key. To indicate a scroll region counter, precede the data name by a percent sign (%) in the key qualifier. The percent sign (%) indicates that the argument to KEY is a data name, as follows:

  ```
  EVALS/KEY=%CNT FIELD2,FIELD3,FIELD4
  ```

  If the key has an asterisk (*) in the last subscript position, the key is a data name. To display the key on the screen, you must use the name of the field within the current scroll region that contains the key data name in the key qualifier, as follows:

  ```
  EVALS/KEY=FIELD1 FIELD2,FIELD3,FIELD4
  ```

  Note that all the fields you specify as an argument to EVALS must be in the current scroll region.

- You must use the percent sign (%) with the EVALS/KEY action in the following cases:

  — Precede scroll region counter keys in the /KEY argument with a percent sign, for example:

  ```
  EVALS/KEY=%MCNT FIELD1
  ```

  — You can include both fields and data names in the **namelist**. Precede each data name in the list with a percent sign, for example:

  ```
  EVALS/KEY=%MCNT FIELD1,%DDN1
  ```

- Table 2–13 summarizes the syntax that you use with the EVALS/KEY action.

Table 2–13 Summary of EVALS/KEY Syntax

Use This Syntax...	When...
EVALS/KEY=FIELD FIELD2,FIELD3	Key data name is a primary key data name (has an asterisk (*) subscript). You can use the name of the field that contains the key data name as the argument to the /KEY qualifier.
EVALS/KEY=%DDN FIELD1,FIELD2	Key data name is a scroll region counter (has a number sign (#) subscript). You must precede the scroll region counter with a percent sign.
EVALS/KEY=%DDN FIELD1,%DDN2	Key data name is a scroll region counter and namelist contains a data name, DDN2. You must precede the data name with a percent sign.

- Do not specify a key data name in **namelist**.
- You do not have to evaluate all fields in a scroll region. For example, you can create a scroll region with five fields, three of which are used for data entry and two of which are used to display information. In this case, issue EVALS actions only for the two fields used for display. The DASL software creates null nodes in the scroll array for any data names for which the EVALS actions are not issued.
- All data names referenced in a single EVALS **namelist** action must reside in the same global node or subtree. If you extract data from different globals or subtrees, issue multiple EVALS actions. Each EVALS action must have a separate key qualifier.
- Because each key specified in an EVALS argument results in a separate subroutine being generated, it is important to fit all the arguments for a single key on one action line. For example, do not specify EVALS/KEY=%DDN F1,F2 on one line and EVALS/KEY=%DDN F3,F4 on the next line. This generates two separate subroutines. The more efficient syntax is the following command line:

```
----------Actions---------------------------------------

EVALS/KEY=%DDN F1,F2,F3,F4
--------------------------------------------------------
```

- You can use pointer chains with the EVALS action to traverse a chain of data names and retrieve dependent data from records. Pointers can point to fields, data names, and other pointers.

  When specifying pointers with EVALS, precede the pointers by a left angle bracket (<). The DASL software traverses the pointers from right to left. See the "Pointers" section of Chapter 1 for more information about pointer chains.

- You use the EVALS/INPUT action to place a blank line as the last entry in the scroll array. Use the EVALS/INPUT action in a data entry scroll screen to allow users to enter a new line without pressing Return to move through all the fields of the last line of the scroll array.

  Do not use EVALS/INPUT in a display-only scroll screen. The EVALS/INPUT action can trigger a "Press RETURN for more" decision prompt even if the next page contains only one blank line. If users then press Return, the DASL software displays a blank scroll region. If the last line of a scroll region is null, the DASL software does not perform a FILE action on that line.

- You cannot use a postconditional expression with the EVALS action.

Related Sections

ASSIGN
EVALS
FILE
FILES
Pointers, Chapter 1
RESET
SCROLL
Scroll Regions

Example

In this example, the key data name is the data name MCNT, a scroll region counter. This example is taken from the MEET screen in the demonstration system, a meeting schedule:

```
----------Actions----------
EVALS/KEY=%MCNT MDATE,MTIME,MLOC,MSUB
---------------------------
```

The DASL software stores the values of MDATE, MTIME, MLOC, and MSUB in the scroll array.

EXIT

You use the EXIT action to specify the field that the DASL software transfers control to when a user presses Exit.

Format

EXIT SPfield{:postcond}

where:

field is the name of the exit field

postcond is an optional postconditional expression **(for the EXIT action only)**

Explanation

The EXIT action is a versatile command. You can use the EXIT action to allow application users to:

- Return to the DASL Main Menu with a single keystroke.
- Erase all data and enter it again.
- Enter all required data in a screen, bypass optional fields, and move to a confirmational prompt.
- Abort the screen they are currently in.

You can also use the EXIT action to process a cleanup field when necessary.

Exit Fields

Define only one current exit field for data screens. This exit field, typically called CLEAN, is usually a field that allows a clean exit from the screen by killing local variables.

The EXIT attribute or action defines the current exit field. When you enter a screen, the value of the current exit field is null. The EXIT attribute or action sets the exit field to **field**. This means that whenever a user presses Exit, the DASL software transfers control to the field specified by the argument.

Table 2–14 illustrates the differences in exit processing when the exit field is set by an EXIT attribute or an EXIT action.

Table 2–14 EXIT Processing

EXIT field is set by...	When...	Result
EXIT attribute	At field invocation and before a READ occurs	When a user presses Exit at a field before entering data, the DASL software moves to any exit field specified with that field's EXIT attribute (or to an exit field set previously if the field has no EXIT attribute).
EXIT action	After data entry	If the user presses Exit after entering data at a field, the DASL software goes to the exit field set as a field action.

Using the Super-Exit Function Key

The DASL software provides a super-exit function key, the Main Menu key. When the user presses Main Menu from any point in the application, the DASL software displays the Main Menu. When the user presses Main Menu at a read field, the DASL software does not process actions for that field. When the user presses Main Menu and exits to an action-only field, the DASL software processes the field actions in the action-only field.

The super-exit function key requires careful organization of exit field processing. When the user presses Main Menu at any field, the DASL software first goes to the current exit field set at that field. Then, the DASL software goes to the next field's exit field, and so forth, until the DASL Main Menu is reached.

When the user presses Main Menu, the DASL software changes the value of the variable %EX from 0 to 1.

Comments

Keep the following points in mind when you use EXIT:

- You can abbreviate the EXIT action as EX.
- When you use EXIT as a screen action or attribute, you must include a **field** argument. That is, you must specify an exit field. The DASL software sets the field you specify as the exit field for the entire screen.
- When you use EXIT as a field action or attribute, the **field** argument is optional. If you specify an argument, the DASL software sets the field you specify as the exit field. If you do not specify an argument, the DASL software sets the current field as its own exit field.
- Any exit field specification you make stays in effect until the DASL software encounters another EXIT action.

- When you use the EXIT action to exit to an action-only field, the DASL software executes the actions in the specified field.

  When you use the EXIT action to exit to a read field, the DASL software performs the read action.

- When you use the super-exit key function to exit to an action-only field, the DASL software performs the actions in the specified field.

  When you use the super-exit key function to exit to a read field, the DASL software resets the exit field if an EXIT attribute is specified in the read field, and moves to the next specified exit field. No actions are processed in that field.

- You can append a postconditional expression to the EXIT action or attribute with an argument. The DASL software passes control on exit to the field named only if the postconditional is true.

  You cannot append a postconditional expression to an EXIT action that has no arguments.

Related Sections

Screen ADBOOK in DASL Demo
Screen ADBOOK, Field NAME in DASL Demo
Screen ADBOOK, Field SEQ in DASL Demo

Examples

The following example shows a screen attribute specifying that control pass to the field CLEAN if the user presses Exit at the current field.

```
Attributes:   EXIT CLEAN
```

In the following example, the EXIT action specifies that control pass to the field CLEAN if the user presses Exit in any successive field.

```
------------------------ Actions -------------------------
EXIT CLEAN
----------------------------------------------------------
```

FILE

You use the FILE field action to store data in your database.

Format

FILE SP namelist{:postcond}

where:

namelist is a list of one or more field names or data names (**You must precede any data names with a percent sign (%).**)

postcond is a postconditional expression

Comments

Keep the following points in mind when you use the FILE action:

- You can abbreviate the FILE action as FI.
- The argument to the FILE action is a list of one or more names separated by commas. The names can be field names or data names.

  Any field name you specify must have a data name associated with it. It is the associated data name that the DASL software actually stores.

  If you use a data name, you must precede the data name with a percent sign (%), for example:

  ```
  FILE FIELD1,%DDN2,FIELD2,%DDN3
  ```

- You do not have to specify where to store the data values. The DASL software uses the reference, piece, extract, and input transform information in the Data Dictionary for the data name you specify or for the data name associated with the field you specify to store the value in the appropriate global node.
- When filing a value, the DASL software always applies any piece or extract references from the data name's Data Dictionary record. The DASL software always inserts the correct number of delimiter characters for the piece reference. If a value is shorter than its extract reference, the DASL software pads the value to its full length by concatenating blanks to the right of the data value.
- You cannot use key data names as arguments to the FILE action. Key data names have an asterisk (*) or number sign (#) as the last subscript in their node reference.
- The LOGON and LOGOFF actions mark the beginning and end of a logical transaction for journaling. Every set of FILE actions should be preceded by a LOGON and followed by a LOGOFF. See the "LOGON" and "LOGOFF" sections for more information on transaction control and journaling.

Related Sections

EVAL
FILES
LOGOFF
LOGON
Screen ADBOOK, Field FILE in DASL Demo

Example

The following example stores values for the data names associated with the fields named in the FILE argument. Note that the FILE action is bracketed by the LOGON and LOGOFF actions to ensure the integrity of the database.

```
------------------------ Actions -------------------------

LOGON ; FILE NAME,PHONE,ADDR1,ADDR2,ADDR3 ; LOGOFF
NXTFLD QUEST
------------------------------------------------------------
```

FILES

You use the FILES field action to store values from a scroll region in the application database.

Format

FILES/KEY=name SPnamelist{:postcond}

where:

name	is a scroll region counter (a data name with a number sign (#) as the last character in the reference subscript) or the name of a field that contains an * key data name (**Precede each scroll region counter by a percent sign (%).**)
namelist	is a list of one or more dependent data names or fields that contain dependent data names (**Precede each data name with a percent sign(%).**)
postcond	is an optional postconditional expression

Comments

Keep the following points in mind when you use FILES:

- The only qualifier you can use with the FILES action is a /KEY qualifier.
- You can abbreviate the FILES action as FIS. You can abbreviate the qualifier /KEY as /K.
- The argument to the FILES action is a list of fields separated by commas. All named fields must be located in the first line of the current scroll region and each must have a data name that is dependent on the key you specify in the qualifier.
- If the key data name is a scroll region counter (that is, has a number sign (#) in the last subscript in its global reference), use that data name as the key qualifier. In this case, the key values are integers from 1 to n, where n is either the number of entries in the scroll region or the number of entries for which the optional postconditional expression is true.

  Precede the data name with a percent sign (%). For example, to file values for the dependent data names in the fields TYPE and DATE, which have the scroll region counter EXNO as a key data name, you use a FILES statement as follows:

```
------------------------- Actions -------------------------
FILES/KEY=%EXNO TYPE,DATE
-----------------------------------------------------------
```

The FILES action then stores data values into the database for each numeric value of the scroll region counter. The DASL software stores the new values over any old values in existing nodes with matching numeric subscripts in the range 1 to n. The DASL software deletes any nodes in the global with a value greater than n.

- If the key data name is used as a primary key (that is, has an asterisk as the last subscript in its global reference), use the name of the field for that key data name as the key qualifier. For example, to file values for the dependent data names in the fields ADDR1, ADDR2, NOK, and PHONE, which have the key data name in the field EMPID, use a FILES statement such as the following:

```
--------------------------- Actions ------------------------
FILES/KEY=EMPID ADDR1,ADDR2,NOK,PHONE
------------------------------------------------------------
```

  When you use a primary key data name, the FILES action stores the new dependent data values for each value of the primary key and deletes the old values of the scroll array subtree associated with the key.

- You must use the percent sign (%) with the FILES/KEY action in the following cases:

  — Precede scroll region counter keys in the /KEY argument with a percent sign, for example:

  ```
  FILES/KEY=%MCNT FIELD1
  ```

  — You can include both fields and data names in the **namelist**. Precede each data name in the list with a percent sign, for example:

  ```
  FILES/KEY=%MCNT FIELD1,%DDN1
  ```

- Table 2–15 summarizes the syntax that you use with the FILES/KEY action.

Table 2–15 Summary of FILES/KEY Syntax

Use This Syntax...	When...
FILES/KEY=FIELD FIELD2,FIELD3	Key data name is a primary key data name (has an asterisk (*) subscript). You can use the name of the field that contains the key data name as the argument to the /KEY qualifier.
FILES/KEY=%DDN FIELD1,FIELD2	Key data name is a scroll region counter (has a number sign (#) subscript). You must precede the scroll region counter with a percent sign.
FILES/KEY=%DDN FIELD1,%DDN2	Key data name is a scroll region counter and namelist contains a data name, DDN2. You must precede the data name with a percent sign.

- All data names referenced in a single FILES **namelist** action must reside in the same global node or subtree. If you must file data in different globals or subtrees, use multiple FILES actions.
- Do not use multiple FILES actions for data that resides in the same node. Subsequent FILES actions will interfere with previous ones.
- If you need to perform any special processing, use a MUMPS routine in place of the FILES action.
- If the data is null, the last line of the scroll region is not filed.
- Each FILES action is a transaction that modifies your database. Make sure FILES actions are journaled by bracketing them with LOGON and LOGOFF actions.
- Unlike the EVALS action, the FILES action can accept a postconditional expression. The DASL software tests the postconditional expression for each value of the key and executes the FILES action for that key value only if the postconditional expression is true.

Related Sections

EVALS
LOGON
LOGOFF
SCROLL
Scroll Regions

Example

The following example illustrates the FILES action in the scroll screen MEET.

```
Field Name: FILE      Description: File DDNs
------------------------------ Actions -------------------------------
FILES/KEY=%MCNT MDATE,MTIME,MLOC,MSUB:MDATE]""
NXTFLD QUIT
----------------------------------------------------------------------
```

HANG

The HANG action directs the DASL software to pause a specified number of seconds before proceeding with processing.

Format

HANG ␣ **seconds{:postcond}**

where:

seconds is the number of seconds you want the DASL software to pause

postcond is an optional postconditional expression

Comments

Keep the following points in mind when you use HANG:

- You can abbreviate the HANG action as HA.
- When you use HANG as a screen or field action, the DASL software pauses for the number of seconds you specify before executing the next screen or field command.

Related Sections

DISPLAY
ERASE

Examples

You can use the HANG action to display a message to the user when you are creating prototypes of screens. Later, you can replace the HANG action with a DO action (or other actions as desired). In this example, you use the HANG action to display the WAIT field prompt "Please wait . . . " for two seconds.

```
Attribute: DEMAND
Prompt:    Please wait...
Actions:   DISPLAY ; HANG 2 ; ! Replace with routine call later
           ERASE WAIT
           NXTFLD RESUME
```

To display a message to the user when the database is locked, do not use the HANG action. You can display a message by using the DISPLAY and ERASE actions as shown in the following example. LOCKED is a demand field with a prompt "Locked by another user . . . ". FIELD1 is an action field.

```
Field:      LOCKED
Attributes: DEMAND
Prompt:     "Locked by another user..."
Actions:    DISPLAY LOCKED ; NXTFLD FIELD1

Field:      FIELD1
Actions:    ERASE LOCKED ; LOCK/TIMEOUT=9 ^XYZ ; NXTFLD LOCKED:%FND
```

You can also display a message that is not erased by using the ERROR action. FIELD1 from the previous example then contains these actions:

```
Field:   FIELD1
Actions: LOCK/TIMEOUT=9 ^XYZ ; ERROR "Locked...":%FND
```

KILL

You use the KILL action to delete the value of specified local or global variables.

Format

KILL SP namelist{:postcond}

KILL{/qualifiers} SP{arguments}{:postcond}

where:

namelist	is a list of one or more local or global variables, DASL variables, or data names separated by a comma
/qualifiers	is one of the following that specifies how to perform the KILL action: **/DDN** — delete all local variables associated with data names within the screen (The data names to be killed are not specified.) **/XREF** — delete entries in the cross-reference global for a specified data name **/KEY** — kill the global node that is referenced by a specified key data name
arguments	are a list of data names, a single data name, or a key data name, depending on which qualifier you use
postcond	is a postconditional expression

Comments

Keep the following points in mind when you use the KILL action:

- You can abbreviate the KILL action as KL.
- The KILL argument consists of a list of local variables, global variables, DASL variables, or data names separated by commas. You must precede any global variables you list with a circumflex character (^).
- The effect of the DASL KILL action is identical to that of the VAX DSM selective kill. A KILL action deletes only the local variables, global variables, DASL variables, or data names you specify.

  The KILL action can delete the following:

  — Local variables set by programmers

  — Local variables set by the screen in fields associated with data names

  — Local variables set as a result of an EVAL %DDN action

  — Variables set with the LOOKUP action

The KILL and the KILL/DDN actions delete local variables. The KILL/XREF and the KILL/KEY actions delete global variables in the database.

- If the name you specify does not exist, the KILL action has no effect. If you specify an unsubscripted variable, the DASL software also deletes all nodes of any subscripted variable with the same name.

  If the variable you specify is subscripted, the DASL software also deletes its descendent nodes. Suppose an array contains the following nodes:

```
N(1,2,3)
N(1,2,3,4)
N(1,2,3,7)
N(1,3,1)
```

  If you kill N(1,2,3), the DASL software also kills N(1,2,3,4) and N(1,2,3,7).

  After you kill a node that has no siblings, the DASL software changes the descendent attribute of the parent node. If the parent is a logical pointer node that contains no data, the DASL software deletes the parent.

- You can use the KILL/DDN field action to kill all local variables associated with data names in a screen. The KILL/DDN action takes the following syntax:

  KILL/DDN SP{(exclusion list)}{:postcond}

  where:

exclusion list	is a list of one or more data names to exclude from the KILL/DDN action
postcond	is an optional postconditional expression

  When you specify an exclusion list, the DASL software kills all the local variables associated with data names with the exception of those listed in the exclusion list.

  The KILL/DDN action does not kill the variables associated with a data name that are set by a LOOKUP action.

  Use the KILL/DDN action as an efficient way to kill most of the variables associated with a screen.

- You use the KILL/XREF action to delete global variables in the cross-reference global of a specified data name. The KILL/XREF action takes the following syntax:

KILL/XREF SP **namelist**

where:

namelist is a data name or list of data names for which a cross-reference global is defined in the Data Dictionary

- You use the KILL/KEY action to kill global variables in the node that is referenced by a key data name. The KILL/KEY action takes the following syntax:

  KILL/KEY SP **namelist**

  where:

  namelist is a key data name or a list of key data names (A key data name is a data name with an asterisk (*) in the last subscript position.)

  Use the KILL/KEY action to delete all patient information from the database for a particular patient, for example. However, the KILL/KEY action does not take into account pointers to the deleted data.

  If data names that are dependent on the key are also cross-referenced, use the KILL/XREF action to delete those data names.

Related Sections

EXIT
KILL Command (*VAX DSM Language Reference Manual*)
EXIT

Examples

As the following example shows, the KILL action is especially useful in exit field processing. In this example, the CLEAN exit field, the DASL software deletes all data-name values before returning to the menu from which the screen was called.

```
Field Name: CLEAN    Description: Clean up local symbol table.
------------------------------ Actions -------------------------------
KILL/DDN ; SET (GNSEQ,GNEQ)="" ; QUIT:'$D(ID)
UNLOCK ^ADBK("DATA",ID,"DEMO"):ID'="" ; KILL ID
----------------------------------------------------------------------
```

The following example shows a field that deletes a person from a file. In this field, the KILL/XREF action deletes the cross-reference entries for NAME, then deletes the node ID.

```
Field: DELETE
---------------------------- Actions ----------------------------
KILL/XREF NAME ; KILL/KEY ID
-----------------------------------------------------------------
```

LOCK

You use the LOCK action to prevent two users from trying to update the same database elements at the same time.

Format

LOCK{/timeout=n} SP argument{:postcond}

where:

/timeout=n	is an optional timeout that specifies that the DASL software attempt to lock the data structure specified in the argument for **n** seconds
argument	an argument identical in form to a VAX DSM LOCK + argument
postcond	is an optional postconditional expression

Comments

Keep the following points in mind when you use the LOCK action:

- You can abbreviate LOCK as LK and /timeout as /TO.
- The LOCK action is equivalent to the VAX DSM LOCK + command. The LOCK action simultaneously locks all global or local variables specified in the argument list.

  Note: In previous versions of the DASL software, the LOCK action was equivalent to the VAX DSM ZALLOCATE command. Because of the change to the LOCK + command, the DASL Version 6.0 LOCK action is not completely compatible with previous versions of the LOCK action.

  See the "UNLOCK" section of this chapter for more explanation of the DASL Version 6.0 differences to the LOCK and UNLOCK actions.

  See the *VAX DSM Language Reference Manual* for more information about the LOCK + and LOCK – commands.
- If you lock an unsubscripted name, the DASL software prevents any other user from locking any node of a subscripted variable of the same name.
- You can lock any name, including data names, screen names, and field names. The LOCK action with an argument locks all elements named in the argument. That is, the LOCK action adds those elements (global variables, local variables, and names) to the VAX DSM Lock Table.

  The LOCK action without an argument locks all elements of the current process. That is, a LOCK without an argument adds all elements from the current process to the VAX DSM Lock Table.

- Use the same conventions in all parts of your application. If one option locks the data name ID (employee identification), another option locks the global ^EDATA(ID) (in which the data name is a key), while a third option locks the field name in which the data name ID is used, the DASL software enters three different names in the lock table.
- If your screens attempt to lock a name already locked by another user, the DASL software suspends execution until the name is unlocked. Use a timeout to overcome a potentially long suspension of execution.

  If LOCK is able to lock the specified names before the timeout expires, it locks the names and sets the DASL variable %FND to 1. If LOCK is not able to lock the specified names before the timeout expires, it sets the DASL variable %FND to 0.

  The DASL software does not display a message informing the user that the specified global is already locked. You can test %FND, and display a message informing the user that the record desired is not currently available.
- The LOCK action is a convention, not an absolute lock out. LOCK serves only to prevent access to the same data if all designers of an application follow the LOCK convention.

Related Sections

UNLOCK
VAX DSM Language Reference Manual

Examples

The following example locks the global ^CF(ID). If another user has locked the global, the LOCK action times out after 10 seconds and moves to the field FLOCK.

```
-------------------------- Actions -------------------------

LOCK/TO=10 ^CF(ID) ; NXTFLD FLOCK:'%FND ; NXTFLD EVLCF
------------------------------------------------------------
```

The following example locks the indicated node of the global ^CUSTID.

```
-------------------------- Actions -------------------------

LOCK ^CUSTID(ID,"DEMOG") ; NXTFLD FILEIT
------------------------------------------------------------
```

Note that the DASL software substitutes the current value of the data name ID before locking. Therefore, if the value of ID is currently 49316, then the node actually locked is:

```
^CUSTID(49316,"DEMOG")
```

LOG

You use the LOG action on batch data screens to add a message to the internal log array.

Format

LOG [SP]expression{:postcond}

where:

expression is a VAX DSM expression

postcond is an optional postconditional expression

Comments

Whenever you use a batch screen (that is, a screen with the BATCH screen attribute) to retrieve data from a file or array, the DASL software stores all data records retrieved and error messages generated by the batch operation in an internal log array. You can store a message in this array by using the LOG action. You can write the array to a VMS file by using the LOGDMP action.

Related Sections

BATCH
Batch Screens
LOGDMP

Example

The following example stores an error message describing missing data into the log file. This message can be written to a file, edited, and used as input to the batch screen at a later date.

```
------------------------- Actions -----------------------

LOG "**ERROR** Date of Birth Required DATA":%RES=""
-------------------------------------------------------------
```

LOGDMP

You use the LOGDMP action to transcribe the information in a log array into a VMS sequential file.

Format

LOGDMP

Comments

Keep the following points in mind when you use the LOGDMP action:

- Whenever you use a batch screen (that is, a screen with the BATCH screen attribute) to retrieve data from a file or array, the DASL software stores all data records retrieved and error messages generated by the batch operation in an internal log array.

  If the batch job completes without error, the DASL software deletes the log array. If, however, errors occur, you can transcribe the log array into a VMS file for examination and correction.

- The LOGDMP action transcribes the internal log array containing the data and any error messages (including those you insert using the LOG action) into a VMS file. You specify the name of the file to use by setting the DASL variable %LOGFILE. (If %LOGFILE is undefined or null, the LOGDMP action does not write the internal log array.)

- You can design a batch screen so that the format of the input file is identical to the format of the log file. After you edit the log file to remove the errors, you can rename the file and use it as input to the batch job.

Related Sections

BATCH
Batch Screens
LOG
Screen Driver Variables

LOGOFF

The LOGOFF action marks the end of a transaction (a change to your application database). LOGOFF writes an end-of-transaction record to the journal file.

Format

LOGOFF

Comments

Keep the following points in mind when you use the LOGOFF action:

- You can abbreviate the LOGOFF action as LF.
- The LOGOFF action marks the end of a transaction. Issue the LOGOFF action immediately after a series of SET or FILE actions that change your application database. If you have enabled VAX DSM journaling, issuing a LOGOFF action clears the VAX DSM internal variable ^%JOURNAL.
- You cannot use a postconditional expression with LOGOFF.

Related Sections

FILE
Journaling (*VAX DSM Installation and Management Guide*)
LOCK
LOGON
Screen ADBOOK, Field FILE in DASL Demo
SET

Example

The following example uses the FILE action to update data from several fields. By bracketing the transaction with LOGON and LOGOFF actions, the example saves the data changes as a unit in a journaling file.

Through this transaction bracketing, the journal file includes a start-of-transaction record and an end-of-transaction record for the current process. VAX DSM records all global SETs and KILLs with the process information. The dejournaling utility can only restore the process by complete transactions. The following example is a complete transaction.

```
------------------------- Actions -----------------------

LOGON ; FILE EMPNAM,EMPADD1,EMPADD2,EMPADD3,EPHONE ; LOGOFF
-----------------------------------------------------------
```

LOGON

The LOGON action marks the beginning of a transaction (a change to your application database). The LOGON action writes a start-of-transaction record to the journal file (if journaling is enabled through VAX DSM). The LOGON action also writes the screen name to the journal record.

Format

LOGON

Comments

Keep the following points in mind when you use the LOGON action:

- You can abbreviate the LOGON action as LN.
- The LOGON action marks the beginning of a transaction. Issue the LOGON action immediately before a series of SET or FILE actions that change your application database. If you have enabled VAX DSM journaling, issuing a LOGON action sets the VAX DSM internal variable ^%JOURNAL.
- You cannot use a postconditional expression with the LOGON action.

Related Sections

FILE
Journaling (*VAX DSM Installation and Management Guide*)
LOGOFF
Screen ADBOOK, Field FILE in DASL Demo
SET

Example

The following example uses the FILE action to update data from several fields. By bracketing the transaction with LOGON and LOGOFF actions, the example saves the data changes as a unit in a journaling file.

```
------------------------ Actions -------------------------------------

LOGON ; COUNT EMPKEY ;FILE EMPNAM,EMPADD1,EMPADD2,EMPADD3,EPHONE ; LOGOFF
----------------------------------------------------------------------
```

MUMPS

The MUMPS action executes the line of VAX DSM statements you specify as an argument.

Format

MUMPS SP **DSMcode**

where:

DSMcode is a valid line of VAX DSM statements

Comments

Keep the following points in mind when you use MUMPS:

- The abbreviation for the MUMPS action is MU.
- You can use MUMPS as a screen action or a field action. MUMPS executes the VAX DSM statements that are its argument.

  You can only specify one line of VAX DSM statements after the MUMPS command.
- The XECUTE action also executes VAX DSM statements. Both the XECUTE and the MUMPS action insert MUMPS code in the compiled routine. However, there are some significant differences at run time.
  - A DASL XECUTE action becomes an XECUTE statement at run time. The argument to the XECUTE command is a quoted string or variable name. When you execute the routine, the XECUTE argument is executed in a separate stack frame from the routine.
  - The argument to the MUMPS action is VAX DSM code that is inserted directly into the compiled routine, and is executed in the same stack frame. The argument to the MUMPS command is MUMPS code.
  - If you want to use the VAX DSM NEW command to NEW a variable, specify the DASL MUMPS action rather than the XECUTE action. Using the NEW command with the MUMPS action NEWs the variable for the duration of the screen. Using the NEW command with the XECUTE action has no effect after the scope of the XECUTE.

Related Sections

DO
Screen Driver Variables
XECUTE

NXTFLD

You use the NXTFLD screen action to specify the first field in the screen. You use the NXTFLD field action to specify the next field to process after the current field.

Format

NXTFLD{ SP field{:postcond}}

where:

field is the name of the next field to process

postcond is a postconditional expression

Comments

Keep the following points in mind when you use NXTFLD:

- You can abbreviate the NXTFLD action as NF.
- The NXTFLD action provides branching logic, similar to a VAX DSM GOTO.
- When you use NXTFLD as a screen action, you must specify a **field** argument. The DASL software then uses that field as the first field to which control is transferred.
- When you use NXTFLD as a field action, you do not have to specify a **field** argument. The NXTFLD action with an argument directs the DASL software to next process the specified field. The NXTFLD action without an argument specifies that the DASL software reprocess the current field.

  See the "RESET" section for a description of how to reset the display when you redirect processing back to the current field.
- Although any field can contain multiple NXTFLD action statements (each with a postconditional), a successfully evaluated NXTFLD action is always the last action processed. On encountering a NXTFLD action, the DASL software terminates action processing and directs control to the field specified by the NXTFLD action.
- If the DASL software processes a field that does not have a NXTFLD action, it exits from the current screen and returns to the location from which the screen was called. Therefore, to quit the current screen, issue a NXTFLD action to an exit field as the last field on the screen. In the exit field, perform all clean-up actions necessary (such as deleting local variables and unlocking globals), but do not issue a NXTFLD action.

Related Sections

GOTO Command (*VAX DSM Language Reference Manual*)
NXTSCN
QUIT
RESET
Screen ADBOOK, Field SEQ on DASL Demo

Example

This example is from SCNAME, the first field in the Define Data Screens screen. At SCNAME, the DASL software prompts for a screen name. Then, the DASL software directs processing to various fields depending on user input.

If you press Return (%RES=""), the DASL software directs processing to an exit field named CLEAN. If the screen record is locked, the DASL software directs processing to a field named LOCK that displays a lock message.

If the screen name already exists, the DASL software directs processing to the field EVAL to display the data screen record and allow you to edit. If the screen name does not exist, the DASL software directs processing to the field DES to allow you to enter new data.

```
------------------------ Actions -------------------------
NXTFLD CLEAN:%RES="" ; ERASE LOCKED ; EVAL %SDTYPE
ERROR "Improper Screen Type.":SDTYPE'="DATA"&%FND
LOCK/TO=9  ^DASD(SDNM) ; NXTFLD LOCKED:'%FND
NXTFLD EVAL:SDTYPE=""
EXIT EQ ; NXTFLD DES
----------------------------------------------------------
```

NXTSCN

The NXTSCN screen action specifies the next screen to process after the current screen.

Format

NXTSCN{ sp screen{:postcond}}

where:

screen is the name of the next screen to process

postcond is a postconditional expression

Comments

Keep the following points in mind when you use the NXTSCN action:

- You can abbreviate the NXTSCN action as NS.
- The NXTSCN action is similar to a VAX DSM GOTO command.
- The NXTSCN action with an argument directs the DASL software to next process the specified screen.
- The NXTSCN action without an argument specifies that the DASL software reprocess the current screen.

  When you use the NXTSCN action without an argument, the DASL software calls the same screen and reinitializes all data-name and field variables. You do not need to use a KILL action on data-name variables when the NXTSCN action is used in this way.
- On encountering a NXTSCN action, the DASL software terminates field action processing and directs control to the screen specified by the NXTSCN action.

Related Sections

DO
NXTFLD
Screen ADBOOK, Field SAME in DASL Demo

Examples

The following example shows the actions associated with an EQ field (the "Edit or Quit" prompt). The NXTSCN action directs the DASL software to pass control to the screen ADBOOK if QUIT is specified.

```
----------Actions----------
NXTSCN ADBOOK:%RES="Q"
---------------------------
```

The following example shows the actions associated with the field SAME which branches back to itself.

```
----------Actions----------
UNLOCK ^ADBK(ID):ID'="" ; NXTSCN
---------------------------
```

QUIT

The QUIT action is similar in function to the VAX DSM QUIT command. The QUIT action causes field action processing to stop. A QUIT action is implied after the last action in a field that does not contain a NXTFLD or NXTSCN action.

Format

QUIT{:postcond}

where:

postcond is an optional postconditional expression

Comments

Keep the following points in mind when you use the QUIT action:

- A QUIT action from a field also quits the screen. (Because NXTFLD commands are analogous to DSM GOTO commands, all fields in a screen are on the same DO stack level.)
- You can use a QUIT command with a postconditional expression to quit from a field and screen before all the field actions are executed. If the postconditional expression evaluates to true, no further field actions are executed, and the entire screen is quit.
- You can use a QUIT command without an argument for documentation purposes to call attention to the implied QUIT after the last action in an exit field.

Related Section

NXTFLD

Example

The following example demonstrates the QUIT action with a postconditional expression, and the QUIT action as a documentary aid.

```
---------------------------- Actions ----------------------------
                ! Kill variables and quit this screen
KILL VAR1,VAR2,VAR3
QUIT:VERSION="SHORT" ; ! If true other variables were never set
KILL VAR4,VAR5,VAR6
QUIT
-----------------------------------------------------------------
```

REPAINT

The REPAINT action repaints the entire video display.

Format

REPAINT{:postcond}

where:

postcond is an optional postconditional expression

Comments

Keep the following points in mind when you use the REPAINT action:

- You can abbreviate the REPAINT action as RP.
- You can use the REPAINT action to restore the current screen after you have completed a DONP action with a postconditional expression.
- When you use the REPAINT action with nested screens, the DASL software repaints all nested screens. For example, if you make a series of calls to screens in successive order using a DONP action to suppress repainting, you can use the REPAINT action at the close of your routine to repaint the entire current screen.
- After you call a routine that disturbs the display, you can use the REPAINT action to restore the display.
- An application user can press Ctrl/W to refresh the screen at any DASL read field.

Related Sections

DONP
Function Keys, *DASL Programmer's Guide*

Example

In the following example, you use the DO action to display a bar graph on the terminal screen, and then use the REPAINT action to repaint the screen.

```
------------------ Actions ----------------------

DO DISPLAY^BARGRAPH
REPAINT
-------------------------------------------------
```

RESET

The RESET action sets all data values associated with the fields you specify to their default values.

Format

RESET SP**fieldlist{:postcond}**

where:

fieldlist is a list of one or more field names

postcond is an optional postconditional expression

Comments

Keep the following points in mind when you use the RESET action:

- You can abbreviate the RESET action as RS.
- If you used the DEFAULT action to assign default values to the specified fields, the RESET action restores the fields to those values. If you did not use the DEFAULT action to assign default values to the fields, then the RESET action sets the value of the field to null.
- When you use RESET as a screen action, you must include a **fieldlist** argument. The DASL software then restores the fields you specify to their default or null values.
- When you use RESET as a field action, you do not have to include a **fieldlist** argument. If you specify an argument, the DASL software restores the fields you list to their default or null values. If you do not specify an argument, the DASL software restores the current field to its default or null value.
- You can append a postconditional expression to RESET with an argument. The DASL software resets the field only if the postconditional expression has a value of true (1).

  You cannot append a postconditional expression to a RESET action that has no argument.

Related Sections

DEFAULT
Fields

Examples

The following example resets the fields NAME, DATE, and TIME if %RES has the value of Y.

```
------------------------ Actions -------------------------
RESET NAME,DATE,TIME:%RES="Y"
----------------------------------------------------------
```

The following example checks a flag to reset the values of FIELD1, FIELD2, and FIELD3.

```
------------------------ Actions -------------------------
RESET:FLAG=1 FIELD1,FIELD2,FIELD3
----------------------------------------------------------
```

SCROLL

SCROLL is a screen action (with the /FILL qualifier) and a field action. You use SCROLL to control action within a scroll region.

Format

SCROLL/qualifiers{:postcond}

where:

qualifiers are one or more qualifiers that specify the action the DASL software is to perform in the scroll region. The qualifiers available are:

/BACK — scroll data backward one scroll window

/DELETE — delete the active (current) scroll line

/FILL — display data from scroll array

/FORWARD — scroll data forward one scroll window

/INDEX — move down to next scroll line

/INSERT — insert a scroll line above the active scroll line

/RESET — reset the scroll region pointers when returning from a field outside the scroll region (Use with the SCROLL/NORESET screen attribute.)

postcond is a postconditional expression

Comments

Keep the following points in mind when you use the SCROLL action:

- You can abbreviate SCROLL as SC. You can abbreviate the SCROLL qualifiers as follows:

Command/Qualifier	Abbreviation
/BACK	/BA
/DELETE	/DE
/FILL	/FI
/FORWARD	/FO
/INDEX	/IND
/INSERT	/INS
/RESET	/RS

- The SCROLL/BACK action scrolls the data in the scroll region backward by one scroll window. A scroll window is the number of lines in the scroll region. If the scroll region is defined for lines 3 to 12, then a scroll window is 10 lines.

  The SCROLL/BACK action does not scroll past the first scroll line. If there is less than a scroll window of data from the starting point to the top of the scroll region, the DASL software stops on the first scroll line.

- The SCROLL/DELETE action deletes the active (current) line.

- The SCROLL/FILL action displays data from the scroll array in the scroll region. When the fill occurs, the DASL software paints the data in the scroll region line by line, beginning with the first line and moving to the last line in the scroll region.

  The SCROLL/FILL action is the only form of the SCROLL action that you can also use as a screen action. Use the SCROLL/FILL action when the scroll array is built by a MUMPS routine rather than by an EVALS action.

- The SCROLL/FORWARD action scrolls the data in the scroll region forward by scroll window. The SCROLL/FORWARD action does not scroll past the last scroll line. If there is less than a scroll window of data from the starting point to the end of the scroll region, the DASL software stops on the last scroll line.

- The SCROLL/INDEX action makes the next sequential scroll line the active line. If /FORWARD is in effect, the next lower line becomes the active line. If /BACK is in effect, the next higher line becomes the active line.

- The SCROLL/INSERT action opens a scroll line above the active scroll line. The new scroll line becomes the active scroll line.

- Use the SCROLL/RESET action in conjunction with the SCROLL/NORESET screen attribute. The SCROLL/NORESET screen attribute directs the DASL software not to reset scroll region pointers when users return from a field outside the scroll region. When you use the SCROLL/NORESET screen attribute, you can also use the SCROLL/RESET action at individual fields to direct the DASL software to reset the scroll region pointers when returning to the scroll region from those fields.

Related Sections

EVALS
FILES
KEY
SCROLL
Scroll Regions

Examples

The following example is a field from the Report Directory Option of the Application Environment Menu. In this field, the DASL software uses the SCROLL/FILL action to display a list of all reports in the Report Directory. If there are reports to display, the DASL software then transfers control to the field MOVE. If there are no reports to display, the DASL software displays the message "No entries." set in the local variable MESS. Then, the DASL software transfers control to the field RETURN. (At RETURN, the DASL software transfers control to an exit field for exit field processing.)

```
--------------------------- Actions ----------------------

SCROLL/FILL ; NXTFLD MOVE:%FND
ASSIGN MESS="No Entries." ; DISPLAY MESS ; NXTFLD RETURN
----------------------------------------------------------
```

The following example is from the MEET screen of the demonstration application. At the MDATE field, if a user enters D at the active scroll line, the DASL software uses the SCROLL/DELETE action to delete the scroll line.

```
------------------------- Actions ------------------------

NXTFLD MTIME:%RES]""
SCROLL/DELETE ;! User deleted MDATE, delete the scroll line.
NXTFLD SEQ:%SCX ;! Last line in the scroll region.
NXTFLD MDATE
----------------------------------------------------------
```

SET

You use the SET action to assign values to variables. The syntax of the SET action is similar to that of the VAX DSM SET command.

Format

SET [SP]**argument,...{:postcond}**

In which **argument** can be one of the following:

storage ref=expression{:postcond}

(storage ref{,...})=expression{:postcond}

where:

storage ref	is a defined or undefined local or global variable name with or without subscripts or is a defined data name
expression	is an expression
postcond	is a postconditional expression

Comments

As with the VAX DSM SET command, the specific action the DASL SET action performs depends on the argument form you use.

storage ref=expression

SET assigns the value of **expression** to **storage reference**.

(storage ref{,...})=expression

SET assigns the value of **expression** to all the named storage references.

Related Section

SET Command (*VAX DSM Language Reference Manual*)

Examples

The following example shows the SET action in the FILE field of the ADBOOK screen. The FILE field stores adress book entries in the database. The COUNT action locks and increments the counter node in a global, if an ID value does not already exist for the address book entry. Then, the SET action sets the value of ID to IDNEXT, if the ID value does not already exist. In consequence, each entry in the address book is assigned a unique ID.

```
Field Name: FILE     Description: File fields (data names).
--------------------------- Actions -------------------------------
! LOGON and LOGOFF are used to create a logical database transaction.
LOGON
! Generate new primary key if it's a new entry.
! COUNT locks and increments the "counter node" in the global.
COUNT IDNEXT:'ID ; SET ID=IDNEXT:'ID
FILE NAME,STREET,CITY,STATE,ZIP,BTHDAY,PHHOME,PHWORK,PHOTH1
LOGOFF ; NXTFLD SAME
-----------------------------------------------------------------
```

The following example shows the multiple set form of the SET argument. The example is a listing of the CLEAN field in the first screen in the Define Data Names Option.

The CLEAN field is an example of good DASL programming practice. CLEAN is an exit field in which the DASL software kills all local values, and unlocks all locked globals before leaving the screen. In the last line of the actions, the DASL software uses a multiple SET action to clear GNEQ and GNSEQ (the decision prompts).

```
Field: CLEAN      Description: Kill off local variables

------------------------------  Actions ----------------------------

KILL/DDN
SET (GNEQ,GNSEQ)="" ; UNLOCK
-----------------------------------------------------------------
```

TCOMMIT

The TCOMMIT action marks the end of a transaction for transaction processing that began with a TSTART action. The TCOMMIT action completes (commits) a transaction. The TSTART action also writes an end-of-transaction record to the journal file if journaling is enabled.

Format

TCOMMIT{/qualifier}

where:

qualifier Determines the confirmational message that the system displays when a transaction is committed. The qualifiers are:

/NOMSG — to suppress the display of the system default confirmational message

/MSG=argument — to display a message other than the system default message. **argument** can contain a MUMPS expression.

Comments

Keep the following points in mind when you use the TCOMMIT action:

- You can abbreviate TCOMMIT as TC.
- The TCOMMIT action marks the end of a transaction for transaction processing. When a transaction is committed, changes to the database occur that are permanent and available to other users of the DASL application system after any locks are released.
- By default, the system displays a confirmational message whenever a transaction is committed. The default message is:

  ```
  "Data committed"
  ```

  You can use the /NOMSG qualifier to suppress display of the default confirmational message.

  You can use the /MSG=argument qualifier to define your own confirmational message. The argument of the /MSG qualifier can contain either a quoted character string or a MUMPS expression, or both, for example:

  ```
  TCOMMIT/MSG="Transaction Complete"

  TCOMMIT/MSG="Transaction ID = "_$ZUID
  ```

  In the previous example, $ZUID provides a unique transaction number for each DSM transaction.
- You cannot append a postconditional expression to TCOMMIT.

Related Sections

Transaction Processing
TSTART
VAX DSM Database Operations Guide

TSTART

The TSTART action starts a logical transaction for transaction processing. The TSTART action also writes a start-of-transaction record to the journal file if journaling is enabled. The TCOMMIT action completes (commits) the transaction.

Format

TSTART{/TIMEOUT=n} SF "name"

where:

/TIMEOUT=n	is an optional timeout that specifies that the system abort the transaction if the transaction is not committed within **n** seconds
"name"	is the name of the transaction that is written to the journal record

Comments

Keep the following points in mind when you use the TSTART action:

- You can abbreviate TSTART as TS.
- The TSTART action begins a transaction for transaction processing. A transaction in transaction processing is a set of updates to the database that exhibits the properties of atomicity, consistency, isolation, and durability (ACID). Table 2–16 describes the ACID properties.

Table 2–16 ACID Properties of Transactions

Property	Description
Atomicity	Within a transaction, all operations succeed, or all operations fail. There are no partially completed transactions.
Consistency	A committed transaction permanently changes the database from one consistent state to another.
Isolation	Many transactions can occur concurrently without interfering with each other's processes. Changes to the database only occur when the transaction is committed.
Durability	Once a transaction is committed, the changes to the database cannot be lost due to hardware or media failure.

- A transaction started with the TSTART action must end with the TCOMMIT action. TSTART and TCOMMIT bracket a set of operations that occur within the transaction. All operations that occur within the TSTART and TCOMMIT commands process as a whole.

- You can use the /TIMEOUT=**n** qualifier to direct the system to abort a transaction if the transaction is not committed before **n** seconds.

  In very rare cases, programmer error can cause a *deadlock* to occur. For example, a deadlock can occur when the program contains a misuse of a LOCK command, or attempts to perform a READ from a terminal within a transaction.

  If you do not use a timeout and one user's transaction becomes deadlocked, the deadlock can affect the transactions of other users and stall the system. You can then examine the deadlocked user's process and correct the transaction.

  If you do use a timeout and a deadlock occurs, the deadlocked user's transaction aborts and the system does not stall, but you can no longer examine the deadlocked user's process to determine the error.

- If a transaction fails, the system displays an error message, rolls back the database to the values present when the transaction began, and does not try to restart the transaction. However, the TPABORT error message represents a special case. A TPABORT error message can occur through no fault of the program within the transaction; the system can cause a TPABORT error independently of the transaction. For example, a TPABORT error caused by the system can occur when two processes are trying to modify data in the same block and one process fails.

  When a TPABORT error occurs, the system:

  1. Aborts the transaction.
  2. Rolls back the database to the values present when the transaction began.
  3. Restores the LOCK table to the values present when the transaction began.
  4. Tries to process the transaction again.

- In a DASL application, the operations that occur within a transaction must be restartable; that is, a transaction must be able to fail and then try again to commit. Because of this requirement, you can use only some DASL commands within transactions. Table 2–17 shows the DASL commands that you can use within transactions. All other DASL commands are illegal within transactions.

Table 2–17 DASL Commands Within Transactions

COUNT	EXIT	LOCK	TCOMMIT
DO	FILE	LOGOFF	TSTART
DONP	FILES	LOGON	UNLOCK
EVAL	HANG	MUMPS	XECUTE
EVALS	KILL	SET	

Note that you cannot use the following types of commands:

- Validations
- Commands that generate a READ from the terminal
- Commands that generate a WRITE to the terminal
- Commands that generate a GOTO (NXTFLD and NXTSCN)
- Commands that create a log file (LOG and LOGDMP)

The DASL software detects illegal commands embedded in a transaction only when it compiles the transaction. At compilation, the system generates an error message.

- On a multi-user system, VAX DSM does not provide an automatic locking mechanism for transaction processing. To ensure isolation and prevent two users from attempting to modify the same data simultaneously, for example, you must use the DASL LOCK action or MUMPS LOCK command outside the transaction when you evaluate values from the database.

  For example, in the DASL demonstration application, the screen ADBOOK uses the LOCK action in the NAME field. Whenever one user enters a name, the LOCK action ensures that the database entries for that name are not available to other users until that user has made changes to the database and released the lock with the UNLOCK action. The transaction occurs while the lock is in effect. The following example shows the actions in the NAME field.

```
-------------------- Actions ------------------------------
NXTFLD CLEAN:%RES="" ; EXIT EQ ; NXTFLD STREET:ID=""
LOCK/TIMEOUT=9 ^ADBK("DATA",ID,"DEMO") ; NXTFLD EVAL:%FND
ERROR "This record is currently locked by another user."
-----------------------------------------------------------
```

  Note that the next field that the screen branches to after the LOCK action is the EVAL field, which contains the transaction to file entries in the database.

- When you bracket transactions with the TSTART and TCOMMIT actions, the DASL software writes records to the journal file in the same manner as the LOGON and LOGOFF actions. Therefore, you do not need to use the LOGON and LOGOFF actions for journaling.

- You can also bracket transactions with the LOGON and LOGOFF actions for journaling purposes. However, the process in which data is stored, and the amount of data that you recover in the event of a hardware or media failure, differs from transactions bracketed with the TSTART and TCOMMIT actions.

When you use the LOGON and LOGOFF actions to journal transactions, the following events occur:

— When the application performs a SET action to a global, data is cached in memory, and written to a disk at a later time.

— In the event of a disk failure, you can perform a backup to restore the database to a previous known state and then dejournal all subsequent transactions.

— In the event of a CPU failure, any data that is cached in memory is lost and not recoverable. Therefore, you can lose the results of several transactions.

When you use the TSTART and TCOMMIT actions to bracket transactions:

— Each transaction permanently alters the database when it is committed. The system ensures that any committed data is recoverable.

— In the event of a CPU failure, you do not lose any data that has been committed to the database. Therefore, you can lose only the results of any incomplete transactions that were processing when the failure occurred.

In consequence, transactions bracketed with the TSTART and TCOMMIT actions provide a faster recovery in case of CPU failure and also do not require a backup procedure.

- Follw these guidelines when you design transactions:

  — Bracket commands tightly with as few commands as necessary within each transaction.

  — Do not use commands within transactions that can lead to an *open transaction* (a transaction that does not complete quickly).

    Open transactions can cause a potential bottleneck on your system. For example, you should not include a READ from the terminal within a transaction because users can respond slowly or leave the terminal without responding. You should not include a WRITE to the terminal because users can press Hold Screen and tie up the terminal screen indefinitely. In each of these cases, the transaction would remain open, preventing other users from accessing the data involved.

  — Use transactions only for commands that modify the database, such as the FILE and FILES actions.

  — Do not set a local variable that increments itself within a transaction.

When a failure occurs, the system rolls back the database to the values present when the transaction began, and also restores the LOCK table to the values present when the transaction began. However, the system does not restore the local symbol table, which retains the current values of variables. When you restart any partially completed transaction, in which a local variable has incremented itself before the failure occurred, you can cause an error in your database. For example, do not use the following type of code within your transaction:

```
TSTART
SET ^GLO=X
SET X=X+1
TCOMMIT
```

Related Sections

LOGOFF
LOGON
TCOMMIT
Transaction Processing
VAX DSM Database Operations Guide

UNLOCK

The UNLOCK action unlocks elements locked with a LOCK action.

Format

UNLOCK SPargument{:postcond}

where:

argument	is an argument identical in form to a VAX DSM LOCK – argument
postcond	is an optional postconditional expression

Comments

Keep the following points in mind when you use the UNLOCK action:

- You can abbreviate the UNLOCK action as ULK.
- The UNLOCK action is equivalent to the VAX DSM LOCK – command.

> **Note:** In previous versions of the DASL software, the UNLOCK action was equivalent to the VAX DSM ZDEALLOCATE command. Because of the change to the LOCK – command, the DASL Version 6.0 UNLOCK action is not completely compatible with previous versions of the UNLOCK action.

In previous versions of the DASL software, when you used the UNLOCK action, the system unlocked any locks that were set previously. For example, if you locked A twice and then specified an UNLOCK action, the UNLOCK action unlocked both locks on A. In the current version of the DASL software, the UNLOCK action decrements the LOCK action, so that you must specify an UNLOCK action for each occurrence of a LOCK action. If you lock A twice, the UNLOCK action now only unlocks one occurrence of the lock on A. The following example shows the difference in DASL Version 6.0 LOCK and UNLOCK actions:

```
          DASL Version 5.2                DASL Version 6.0
          ----------------                ----------------

(1ST)     LOCK A                          LOCK A
(2ND)     LOCK A                          LOCK A
          UNLOCK                          UNLOCK

          Unlocks both (1ST) and          Unlocks (2ND) lock only.
          (2ND) locks.                    (1ST) lock remains.
```

See the *VAX DSM Language Reference Manual* for more information about the LOCK + and LOCK – commands.

- The UNLOCK action with an argument unlocks all elements named in the argument. That is, the UNLOCK action removes those elements (global variables, local variables, and names) from the VAX DSM Lock Table.

  The UNLOCK action without an argument unlocks all elements locked by the current process. That is, an UNLOCK action without an argument removes all elements written in the VAX DSM Lock Table from the current process.

Related Section

LOCK

Examples

The following example unlocks the specified global variables.

```
--------------------------- Actions --------------------------

FILE NAME,STREET,CITY ; LOGOFF ; UNLOCK ^ADBK(ID) ; NXTFLD CLEAN
--------------------------------------------------------------
```

The following example unlocks all locked elements.

```
--------------------------- Actions --------------------------

FILE NAME,STREET,CITY ; LOGOFF ; UNLOCK ; NXTFLD CLEAN
--------------------------------------------------------------
```

XECUTE

The XECUTE action executes the line of VAX DSM statements you specify as an argument.

Format

XECUTE [SP]**argument{:postcond}**

where:

argument is a string of DSM code enclosed in quotation marks

postcond is an optional postconditional expression

Comments

Keep the following points in mind when you use the XECUTE action:

- The abbreviation for the XECUTE action is XE.
- You can use XECUTE as a screen action, a field action, or a validation. The XECUTE action executes the VAX DSM statements that are its argument. Control returns to the point immediately following the XECUTE argument.
- The MUMPS action also executes VAX DSM statements. Both the XECUTE and the MUMPS action insert MUMPS code in the compiled routine. However, there are some significant differences at run time.
  - — A DASL XECUTE action becomes an XECUTE statement at run time. The argument to the XECUTE statement is a quoted string. When you execute the routine, the XECUTE argument is executed in a separate stack frame from the routine.
  - — The argument to the MUMPS action is VAX DSM code that is inserted directly into the compiled routine, and is executed in the same stack frame. The argument to the MUMPS command is MUMPS code.
  - — If you want to use the VAX DSM NEW command to NEW a variable, specify the DASL MUMPS action rather than the XECUTE action. Using the NEW command with the MUMPS action NEWs the variable for the duration of the screen. Using the NEW command with the XECUTE action has no effect.
- You must set %MSG within the XECUTE argument to return a message for errors or events.
- When you use XECUTE as an action, you can append a postconditional expression to the XECUTE argument. The DASL software executes the statements in the argument only if the postconditional is true.

For more information on XECUTE as a validation, see the "XECUTE" section in this chapter.

Related Sections

DO
MUMPS
Screen Driver Variables
XECUTE

Display Designer

Using the interactive Display Designer, you can create new fields or edit existing fields for the current data screen. You can select the Display Designer at the prompt on the Define Data Screens Option.

Explanation

The Display Designer is a screen-oriented editor that you use to create or edit fields. You can use the Display Designer to perform any of the following tasks:

- View the screen as it appears to the user
- Move fields
- Delete fields
- Create fields
- Edit fields
- Create boxes
- Create lines
- Delete boxes
- Delete lines

Accessing the Display Designer

The Display Designer is a tool for refining the design of the data screens you created previously through the Define Data Screens Option. You can access the Display Designer from the first screen of the Define Data Screens Option when the DASL software displays the following prompt:

```
Display, Screen Edit, Field Edit, Compile, Batch, Utilities, or Quit:
```

Enter D to choose the Display Designer. The DASL software then displays the current data screen.

If the data screen is a new one, the DASL software displays a blank screen. If the data screen is an existing screen, the DASL software displays the current fields in their line and column positions.

The DASL software places the cursor at the home position (coordinates X=1 and Y=1) and displays the position of the cursor at the lower right corner of the screen in the form:

```
X:1 Y:1
```

Using the Display Designer Keypad

To edit the screen, you use the keys on the auxiliary keypad of Digital's VT-series terminals. Figure 2–4 shows the auxiliary keypad keys.

Figure 2–4 Display Designer Keypad

Table 2–18 describes the Display Designer keys.

Table 2–18 Display Designer Keys

Key	Action
Attach/Advance	If the cursor is currently within a field (between the right and left ends), attaches the cursor to the field. The cursor moves to the left end of left-justified fields, the right end of right-justified fields, or the center of center-justified fields. The cursor attaches to fields first, then boxes. If the cursor is currently on a line connecting box corners, attaches the cursor to the box. The cursor moves to the upper left corner of the box. If the cursor is not currently within a field or box, attaches the cursor to the next field or box to the right of the current position. If the cursor is at the right side of the screen, attaches the cursor to the next lower field or box at the left side of the screen. If the cursor is currently attached to a field or box, detaches the cursor from the current field or box, causes the DASL software to update the database with any changes you made during the attachment, and attaches the cursor to the next field or box to the right. If the cursor is at the right side of the screen, attaches the cursor to the next lower field or box at the left side of the screen. The DASL software displays the attached field name or box coordinates at the lower left screen corner.
Box	If the cursor is attached to a field, has no effect. If the cursor is currently attached to a box, moves the cursor to the next box corner. Cursor moves from upper left corner to upper right corner to lower right corner to lower left corner. If the cursor is not currently attached to a box, creates graphic box. (See the following sections for more information about creating boxes.)
Cancel	Terminates the current attachment and cancels any field movement or editing performed during the current attachment.
Create/Edit	If the cursor is currently attached to a field, lets you edit the values associated with a field. By default, you can perform a partial edit that allows you to change the justification, data name, prompt, field length, and attributes. You can also select a full edit to invoke the full field definition screen of the Define Data Screens Option. If the cursor is not currently in a field, lets you create a new field at the cursor position. As with an existing field, you can select a full or partial edit when you create a new field.

Table 2–18 (Cont.) Display Designer Keys

Key	Action
Delete	Deletes the currently attached field or box after you confirm the deletion. If no field or box is currently attached, attempts to perform an attachment at the current position. If there is no field or box at the current position, takes no action but displays an error message.
Detach	Detaches the cursor from a field or box and causes the DASL software to update the Screen Driver database with any changes you made during the attachment. Note, however, that pressing the Attach/Advance at an already attached field or box performs an implicit detachment from that field or box, an update of the database, and an attachment to the next field or box.
Pointers	Move one position in the direction of the arrow shown. Therefore, keypad 7 moves the cursor northwest and keypad 3 moves the cursor southeast. If a field is currently attached, moving the cursor moves the field. As the cursor moves, the DASL software updates the position numbers to match the current position of the cursor on the screen.

Moving a Field

Follow these steps to move a field:

1. Move the cursor to the field and press Attach/Advance.
2. Use the pointer keys to move the field to the new position. Because the field is attached to the cursor, it follows cursor movement.
3. Press Detach or Attach/Advance to release the field and save your changes.
4. Press Cancel to return a field to its original coordinates.

Moving Overlaid Fields

Overlaid fields are fields that occupy the same row and column position. Most overlaid fields are demand fields that appear in different processing situations. Overlaid fields are more difficult to move or access. You first move the cursor to the fields and press Attach/Advance. You are now attached to the field with the closest X and Y coordinates to the current overlaid position. If the two fields are at the same X and Y coordinates, the DASL software attaches to them in the alphabetic order of their field names.

To move one of the overlaid fields, you must move the fields away one by one until you get to the field you want. To delete or edit one of the overlaid fields, you repeatedly press Attach/Advance until the cursor attaches to the overlaid field you want. The DASL software displays the field name in the lower left corner of the screen.

Creating a Box or a Line

You can use the Display Designer to create a box or line, as follows:

1. Move the cursor to the position in which you want the first corner of the box or the left end of the line.
2. Press Box. A diamond appears on the screen at the position of the cursor.
3. Follow these steps to create a box:
   a. Move the cursor diagonally to the position of the opposite corner of the box. (For example, move from upper left to lower right.)
   b. Press Box. A box forms based on these two corner positions. The Display Designer returns the cursor to the position of the first corner.
4. Follow these steps to create a line:
   a. Move the cursor horizontally or vertically to the position at which you want the end of the line.
   b. Press Box. A line forms based on these two positions. The Display Designer returns the cursor to the position of the first line end.
5. Press Detach or Attach/Advance to detach and save the box.

Editing Boxes

You can use the Display Designer to move, stretch, or compress a box as follows:

1. Attach to the box by pressing Box. The DASL software attaches the cursor to the upper left corner of the box.
2. Use Box to move the cursor to the corner of the box you want to move. Each time you press Box, the DASL software moves the cursor clockwise to a different corner.
3. Use the pointer keys to move the cursor. As you move the cursor, you stretch or compress the box.
4. Press Detach or Attach/Advance to detach and save the box.

Deleting Fields and Boxes

You can delete fields and boxes as follows:

1. Attach to the field or box.
2. Press Delete.
3. Enter Y when the Display Designer prompts you to confirm the deletion.

Creating or Editing a Field

You can create a new field or edit an existing field as follows:

1. To create a new field, move the cursor to the desired field position. Press Create/Edit. Enter the new field name where prompted.

   To edit an existing field, attach to the desired field. Press Create/Edit.

2. When the DASL software prompts for a field justification, take one of the following steps:

   - To enter or edit only basic values (data name, prompt, length, or attributes), specify a justification of L (left), R (right), or C (center). The Display Designer then prompts you to enter or edit values.

     You can use the Display Designer to edit any data name associated with a field. If you enter DDN in response to the "Data Name:" prompt, the DASL software displays the Define Data Names Option screen. If you enter DDNB in response to the "Data Name:" prompt, the DASL software displays the Define Data Names (BRIEF) Option screen. When you finish, control returns to the Display Designer.

   - To enter or edit all field values, specify F (full). The Display Designer calls the second screen of the Define Data Screens Option where you can enter or edit all values for the field.

     When you finish, control returns to the Display Designer.

3. The cursor remains attached to the created or edited field. Press Detach or Attach/Advance. The Display Designer updates the Screen Driver database and the display when you finish.

Exiting from the Display Designer

Press Exit to exit from the Display Designer. Now, you must compile the screen to incorporate the Display Designer changes. **Make sure you press Detach to detach from the last field you create or edit before you exit.** The DASL software does not save any changes you make to a field to which you are still attached when you exit.

Related Sections

Define Data Screens Option
Fields

Scroll Regions

Scroll regions are windows you can create on the screen. Data within a scroll region can be made to scroll up and down, allowing the application user to access more information than is normally displayed in a single screen.

Explanation

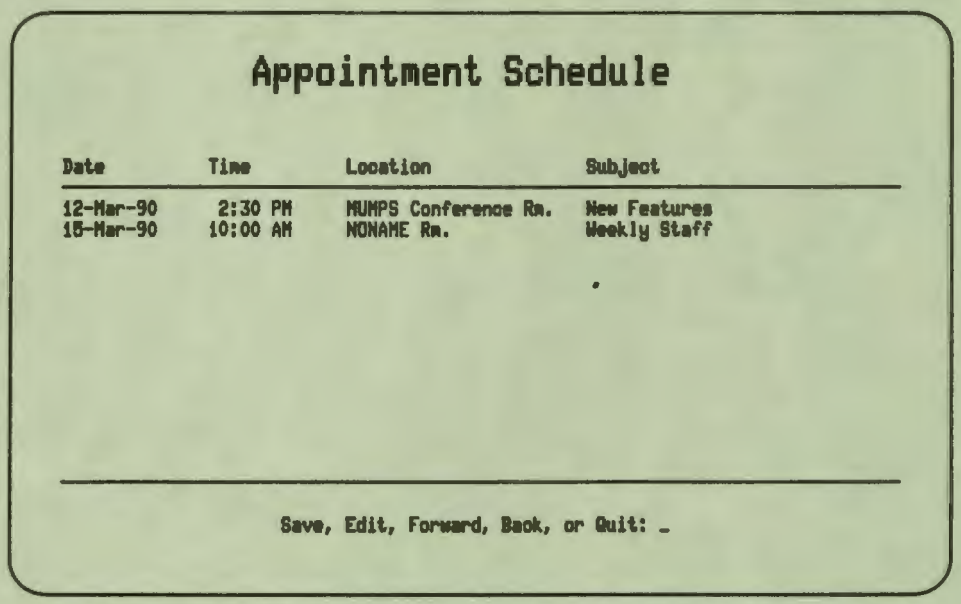
You use a scroll region whenever you need to capture or display an indefinite number of related data items or sets of data items.

Some scroll regions are interactive: they allow application users to add new values, delete values, or modify existing values for a set of data names.

Other scroll regions are display-only: they display values for a specified set of data names, but do not allow users to edit the displayed items.

Figure 2–5 shows a scroll region in the MEET screen of the demonstration system.

Figure 2–5 Scroll Region in MEET Screen



Creating Scroll Regions

A scroll region can take up an entire screen, or can share the screen with other nonscroll fields. However, you can create only one scroll region on any screen.

To create a scroll region, specify the SCROLL screen attribute in the first screen of the Define Data Screens Option. SCROLL specifies the size of the scroll region and, optionally, the array to use for data collected. The format of SCROLL is as follows:

SCROLL{/qualifier}{ SP first,last}

The arguments **first** and **last** are integers from 1 to 23 that specify the first and last line of the scroll region.

The **first** and **last** values are relative to the first line of the screen. Therefore, if the screen starts on line 3 and you specify a **first** value of 2 and a **last** value of 12, the scroll region starts on line 5 and ends on line 15 of the screen.

The qualifiers specify where the DASL software is to store scroll region data.

Building Default Scroll Arrays

When you use the SCROLL attribute without a qualifier, the DASL software stores all data in a scroll region in a local array of the same name as the screen containing the scroll region. The scroll array is a doubly linked list.

You can build a scroll array in the following ways:

- Use the EVALS action to direct the DASL software to build the entire array.
- Design a MUMPS routine to build part or all of the array.

The structure of a local scroll array is:

```
screen name(N)="forward pointer; backward pointer"

screen name(N,DATA NAME)="data"

screen name(N,DATA NAMEi)="internal field information"
```

where:

N	is an integer that represents a scroll region line
forward pointer	is a number pointing to the next node in the scroll array
backward pointer	is a number pointing to the previous node in the scroll array
DATA NAME	is a data name associated with a field in the scroll region
DATA NAMEi	represents a data name that has an internal node associated with it

Keep these points in mind about default scroll arrays:

- If insertion or deletion is allowed, the first subscript N can be a decimal number rather than an integer. With insertion of new data, new subscripts are generated so that the numeric collating sequence maintains the order of the linked list. This allows for simpler application processing of scroll data. The application programmer can use the VAX DSM $ORDER function to traverse the array.
- If there is no next or previous node, the DASL software assigns a pointer value of 0. Therefore, the first node in the array has a backward pointer of 0, and the last node in the array has a forward pointer of 0.
- Under certain conditions, scroll fields can have an internal node associated with them. The DASL software builds this node during the SCROLL/FILL action only if you need internal information as in the following cases:
  - — The field data name has both an internal and external representation, such as dates, times, or input and output transforms.
  - — The field is a DEMAND field.
  - — The field is erased or displayed using the ERASE and DISPLAY actions.

This internal node takes the form:

```
screen name(N,DATA NAMEi)= ext format $C(1) demand flag $C(1) display attributes
```

DATA NAMEi is the data name with a lowercase letter (i) appended.

Note: This node is subject to change. Never reference or modify this node in your application software.

Using SCROLL Attribute Qualifiers

You can add qualifiers to the SCROLL attribute to specify how the DASL software handles scroll arrays. The qualifiers available are:

- /ARRAY=name
- /GLOBAL
- /NOKILL
- /NORESET

/ARRAY=name

Use the /ARRAY qualifier to specify the name of the local or global array. The name you specify must be from 2 to 7 characters in length and follow the format of one uppercase alphabetic character followed by 2 to 6 uppercase alphanumeric characters.

You can use /ARRAY to rename the local or global array. However, the structure of the array is identical to those of the default local and global scroll arrays.

/GLOBAL

Use the /GLOBAL qualifier to specify a global scroll array. If you specify /GLOBAL alone, the DASL software creates the following global scroll array:

```
^DATG($J)
^DATG($J,"S")
^DATG($J,"S",screen name,N)="pointers"
^DATG($J,"S",screen name,N,data name)="data"
```

where:

^DATG	is a scratch global used for a number of different DASL operations
$J	is the VAX DSM $JOB special variable that contains the current process ID
"S"	is an uppercase S used to indicate a Screen Driver scratch global

Below this level, the structure of the global scroll array is identical to that of the default local scroll array. The DASL software handles EVALS actions, inserts, and deletes in the same way as it does the default local scroll array. To avoid allocation failures, use /GLOBAL for screens that handle large amounts of data.

/NOKILL

The /NOKILL qualifier directs the DASL software not to kill the scroll array on screen exit. The DASL software always kills the scroll array on screen entry.

/NORESET

The /NORESET qualifier directs the DASL software not to reset scroll region pointers when users return to a scroll region after working in an outside field. (Without this qualifier, the DASL software resets scroll region pointers by default.)

Scroll Fields

After you establish the scroll region as a screen attribute, you define the scroll fields. To include a field in a scroll region, you define the field on the first line of the scroll region.

Suppose you want users to enter lists of employee names, company site locations, and telephone numbers. Set up three fields containing the data names NAME, SITE, and PHONE on the first line of your scroll region, and give each field a different column position.

The DASL software recognizes only one active scroll line at a time. Use the SCROLL/INDEX command to change the active line. Each time SCROLL/INDEX is specified, the active line moves down one line.

For example, after the application user enters values for NAME, SITE, and PHONE on the first line, SCROLL/INDEX moves the active line to the second line to let the user enter additional data. You can use the NXTFLD action to move the cursor to the appropriate field.

The DASL software views the data you enter on the second line as a second set of values for the fields and DDNs already defined for the first line. This is the reason you need to define data fields only for the first line of the scroll region.

After the user has entered several lines of data, the scroll region appears as follows:

```
SMITH, JAMES    KLO1-1/E6        345--2346
JONES, SCOTT    RJ1-4/A8         876--0987
ALPERT, ALAN    MRO3-2/H7        231--4456
```

Using the KEY Field Attribute

You can allow users to edit or display scroll data with the scroll region function keys. To do so, give the scroll region fields the KEY field attribute followed by any combination of qualifiers.

The KEY field attribute enables the scroll region function keys and includes qualifiers that specify which keys to enable.

- The KEY/DELETE attribute enables the Remove key. When users press Remove, the DASL software deletes the current scroll region line. Data on any succeeding scroll lines moves up one line. The DASL software maintains the current scroll line as the *active line*.
- The KEY/FORWARDBACK attribute enables the Prev Screen and Next Screen keys in a scroll field or a nonscroll field such as a decision prompt. When users press Prev Screen or Next Screen, the DASL software moves the cursor up or down one window in the scroll region.
- The KEY/INSERT attribute enables the Insert Here key. When users press Insert Here, the DASL software moves the data associated with the current scroll line (and any succeeding scroll lines) down one line. The DASL software creates a new blank scroll line as the active line, and initializes all the scroll region data names to their default values.

- The KEY/UPDOWN attribute enables ⇑ and ⇓. When users press ⇑, the DASL software moves the cursor and the active line up one line. If the current line is the first line in the scroll region, the DASL software takes no action.

  When users press ⇓, the DASL software moves the cursor and the current active line down one line. If the current line is the last line of data in the scroll region, the DASL software takes no action.

  The KEY/UPDOWN attribute automatically enables KEY/FORWARDBACK.

Scroll Region Actions

Table 2–19 shows DASL actions that you specify for scroll regions.

Table 2–19 DASL Scroll Region Actions

Action	Use
EVALS	Retrieves values from the database and builds the scroll array
FILES	Stores values from the scroll array to the database
SCROLL	Controls action within a scroll region

See the "EVALS," "FILES," and "SCROLL" sections for more information about these actions.

DASL Scroll Region Variables

The DASL software maintains four variables to assist you in scroll region processing:

- %SL contains the number of lines in the scroll region.
- %SP contains the subscript value of the current scroll line, the active line. For example, in Screen(N), if N is equal to 1.5, then %SP contains the value 1.5.
- %SCX is the scroll region flag. %SCX is set to 1 if the current scroll line is the last scroll line. %SCX is set to 0 if the current scroll line is not the last scroll line. Setting %SCX is useful for next field branching logic. For example, you can define the following field action using %SCX.

  ```
  NXTFLD SEQ:%SCX
  ```

- %SN contains the current scroll line number relative to the first line in the scroll array. %SN is always an integer, while %SP can be a decimal number. For example, if %SP is equal to 1.5, then %SN is equal to 2.

Scroll Arrays at Run Time

At run time, the DASL software initializes the first line of the scroll region to the default field values. You can load data into the scroll region by using the EVALS action or by calling an application routine to build the data portion of the arrays. If you call an application routine, you must use a SCROLL/FILL action after the routine call, for example:

```
-------------------------Actions-----------------------------------

DO BUILD^ROUTINE ; SCROLL/FILL
NXTFLD EDIT
-------------------------------------------------------------------
```

The SCROLL/FILL action sets up the linked list pointers, performs output transforms, and handles date and time conversions.

Consider the following example. Suppose you create a scroll region in which users enter hospital patients with their dates of birth and sex. You define the scroll region and establish three fields on the first scroll line containing the data names:

- PNAME
- PDOB
- PSEX

After you use the EVALS action to load values for the data names into the array, or after the user enters values for the data names, the scroll array looks like the following:

```
SCREEN(1)="2;0"
SCREEN(1,"PNAME")="BUSCH, LESTER"
SCREEN(1,"PDOBi")="10-OCT-57"
SCREEN(1,"PDOB")=42651
SCREEN(1,"PSEX")="M"
SCREEN(2)="3;1"
SCREEN(2,"PNAME")="MILLER, DAISY"
SCREEN(2,"PDOBi")="22-JUN-48"
SCREEN(2,"PDOB")=39254
SCREEN(2,"PSEX")="F"
SCREEN(3)="0;2"
SCREEN(3,"PNAME")="VINCENT, ANDREW"
SCREEN(3,"PDOBi")="11-JAN-50"
SCREEN(3,"PDOB")=39822
SCREEN(3,"PSEX")="M"
```

> **Note:** Observe that the DASL software stores the DATE data type data for PDOB in internal and external date format. The DASL software also stores times in both external and internal time format.

If users then edit the scroll region and insert a new patient between Lester Busch and Daisy Miller, the DASL software adds decimal-subscripted nodes to insert the new node in the correct position.

The DASL software also updates the forward and backward pointers, as follows:

```
SCREEN(1)="1.5;0"
SCREEN(1,"PNAME")="BUSCH, LESTER"
SCREEN(1,"PDOBi")="10-OCT-57"
SCREEN(1,"PDOB")=42651
SCREEN(1,"PSEX")="M"
SCREEN(1.5)="2;1"
SCREEN(1.5,"PNAME")="CHANDLER, ROY"
SCREEN(1.5,"PDOBi")="21-JUN-84"
SCREEN(1.5,"PDOB")=52402
SCREEN(1.5,"PSEX")="M"
SCREEN(2)="3,1.5"
SCREEN(2,"PNAME")="MILLER, DAISY"
SCREEN(2,"PDOBi")="22-JUN-48"
SCREEN(2,"PDOB")=39254
SCREEN(2,"PSEX")="F"
SCREEN(3)="0;2"
SCREEN(3,"PNAME")="VINCENT, ANDREW"
SCREEN(3,"PDOBi")="11-JAN-50"
SCREEN(3,"PDOB")=39822
SCREEN(3,"PSEX")="M"
```

If a user later deletes Daisy Miller, then the DASL software kills SCREEN(2) and relinks the pointers, as follows:

```
SCREEN(1)="1.5;0"
SCREEN(1,"PNAME")="BUSCH, LESTER"
SCREEN(1,"PDOBi")="10-OCT-57"
SCREEN(1,"PDOB")=42651
SCREEN(1,"PSEX")="M"
SCREEN(1.5)="3;1"
SCREEN(1.5,"PNAME")="CHANDLER, ROY"
SCREEN(1.5,"PDOBi")="21-JUN-84"
SCREEN(1.5,"PDOB")=52402
SCREEN(1.5,"PSEX")="M"
SCREEN(3)="0;1.5"
SCREEN(3,"PNAME")="VINCENT, ANDREW"
SCREEN(3,"PDOBi")="11-JAN-50"
SCREEN(3,"PDOB")=39822
SCREEN(3,"PSEX")="M"
```

When exiting from the screen containing the scroll region, the DASL software kills the scroll array. You can use the SCROLL/NOKILL attribute to override this KILL action.

When you use the FILES action or call a VAX DSM routine to store scroll region data values in the database, the DASL software resequences the scroll index.

Related Sections

EVALS
FILES
KEY
Screen Driver Variables
SCROLL Action
SCROLL Attribute

Screen Driver Variables

DASL variables are local variables used by the Screen Driver and the Report Driver. These variables are used to pass information between the DASL software and the application software. All DASL variables are preceded by a percent sign (%). The name is a mnemonic for the information that the variable contains.

Screen Driver variables contain information about screen processing. Your screens can examine, test, and sometimes set these variables. Table 2–20 lists the Screen Driver variables and their contents. The table also specifies whether you can set the variable.

Table 2–20 DASL Screen Driver Variables

Variable	Contents	Set
%ATR	Optional list of device attributes, passed to %DADEV	Y
%DTI	Date in internal format	Y
%DTX	Date in external format	Y
%ERR	Parameter the programmer sets to correspond to an error code in ^DASY("ERR","CD") and then passes to LOG^%DAERRT	Y
%ERROR	Batch screen error message	N
%EX	Super Exit flag is 1 if Main Menu key is pressed	N
%EXT	The external value of data in an input or output transform	Y
%FND	1 or 0 set by EVAL, EVALS, LOCK, LOOKUP	N
	0 to n set by FILES, the number of items filed	N
%INP	User input before the DASL software performs validations	N
%INPUT	Batch screen file or input array name	Y
%INT	The internal value of data in an input or output transform	Y
%INT1	An internal value of data in a TRAN1 type cross-reference transform	Y
%INT2	An internal value of data in a TRAN2 type cross-reference transform	Y
%L(0)	Current screen name	N
%LOGFILE	Batch screen log file	Y
%MSG	Error message returned by validations	Y

Table 2–20 (Cont.) DASL Screen Driver Variables

Variable	Contents	Set
%OLD	Old field data value	N
%OPN	Name of current option	Y
%RES	New field data value	Y
%REST	Internal value of new field data used to perform pattern match validations for translation purposes	Y
%SCX	Scroll region flag 1 if current line is last line, 0 if current line is not the last line	N
%SL	Number of lines in scroll array	N
%SN	Number of lines between the current scroll line and the first line in the scroll array	N
%SP	Scroll array subscript of current line	N
%TMI	Time in internal format	Y
%TMX	Time in external format	Y
%USRID	Internal key for the user, having the same value as the logical name DSM$DASL_USRID	Y
%XS	LOOKUP command execute string	Y

Comments

Table 2–21 indicates sections in this chapter or in other chapters in this manual that contain more information about certain Screen Driver variables.

Table 2–21 References to Screen Driver Variables

Screen Driver Variables	Reference
%DTI,%DTX	DATE Validation
%ERROR,%INPUT,%LOGFILE	Batch Screens
%MSG,%FND,%XS	LOOKUP Validation
%MSG	Error/Event Code Dictionary Option, Chapter 5
%SCX,%SL,%SN,%SP	Scroll Regions
%INT,%INT1,%INT2,%EXT	Transforms, Chapter 1 Cross References, Chapter 1
%TMI,%TMX	TIME Validation
%OLD,%RES,%INP	Fields
%REST	DASL Language Utilities, Chapter 5
%EX	EXIT Action and Attribute

Related Sections

Fields
Scroll Regions

Batch Screens

A batch screen is a data screen that collects data noninteractively. A batch screen processes data from a VMS sequential file or local array.

Explanation

Most data screens in the DASL software are designed for interactive input. However, the DASL software can also handle batch input from a data file or local array. To handle batch input, you set up a batch screen by giving a screen the BATCH attribute. The batch screen validates data exactly as it would if the data were entered interactively.

Whenever you use a batch screen, the DASL software stores all data records retrieved and error messages generated by the batch operation in an internal log array. Use the LOG action to store a message in this array.

Use the LOGDMP action to write the array to a VMS file.

BATCH Attribute

The BATCH screen attribute specifies that a screen process data from a VMS sequential file or local array.

When you use BATCH, the DASL software receives input from a VMS sequential file or local array and stores this data in an internal area. BATCH also stores error messages in an internal area. You can write the contents of this area to a batch error log file.

The BATCH attribute has the following format:

BATCH{/DEBUG}{=n}

where:

/DEBUG	is an optional qualifier that causes write statements to be compiled into a generated routine and allows you to view the batch process on your screen
=n	is an optional modifier, an integer between 0 and 9 that causes the batch process to halt for 0 to 9 seconds after each transaction so that you can view the screen logic and errors on your screen

You can use the following three variations of the BATCH attribute:

- BATCH
- BATCH/DEBUG
- BATCH/DEBUG=n

When changing from one batch format to another; you must recompile the batch screen.

BATCH

When you use BATCH without a qualifier, the DASL software performs normal batch processing, but does no writes to the principal device. You cannot view the batch process at your terminal.

BATCH/DEBUG

When you use BATCH/DEBUG, the DASL software compiles write statements in the generated routine to aid in debugging. The DASL software displays the fields and data at your terminal as it reads data and displays error messages as errors occur.

BATCH/DEBUG=n

When you use BATCH/DEBUG=n, the DASL software compiles write statements in the generated routine and hangs for **n** seconds after each read field as a debugging tool.

BATCH/DEBUG causes write statements to be compiled into a generated routine. To remove these statements, edit the attribute to BATCH and recompile the screen.

DASL Variables for Batch Screens

The DASL software uses special variables related to batch screens. These variables are:

Variable	Meaning
%ERROR	Batch screen error message
%INPUT	Batch screen file or input array name
%LOGFILE	Batch screen log file

Input Data for Batch Screens

The two ways to provide input data for a batch screen are:

- Sequential data input from a VMS file, often used for initial database loads
- Buffered input from a local array, often used for links from other systems

For both types of input, you must set the internal variable %INPUT to an appropriate value.

In a sequential data input file, the value of %INPUT is the name of a data input file. The batch screen reads data sequentially from this file name.

The %INPUT file consists of any number of records. Each record contains one entry for each input field of the screen. Each of the field entries must occupy its own line in the input file. You can indicate null input with blank lines. The line following the input line to the last field of the record is by convention a single asterisk (*) to indicate the end of the record. Two consecutive lines of single asterisks indicate the end of the file.

Each field entry must pass any field validations. If input to any field does not pass the validation, the entire record is not added to the database. However, a copy of the record as well as the reason for the failed validation is stored in the log file %LOGFILE, which maintains a record of the errors to the batch process.

For buffered input based on a %INPUT array, you must provide the data in an array based on %INPUT. The DASL software reads the array by sequentially increasing the value of %INPUT by one and reading the data value contained in %INPUT(%INPUT).

Any error conditions cause immediate transfer to the EXIT field. The error message is recorded in %ERROR. Error conditions include end of file, invalid data length, and validations failure. You can also force an error condition by using the ERROR action.

Batch Log Files

Whenever you use a batch screen, the DASL software stores all data records retrieved and error messages generated by the batch operation in an internal log array. You can store a message in this array by using the LOG action.

If the batch job completes without error, the DASL software deletes the log array. If, however, errors occur, you can transcribe the log array into a VMS file for examination and correction.

The LOGDMP action transcribes that internal log array containing the data and any error messages (including those you can insert using the LOG action) into a VMS file. You specify the name of the file to use by setting the DASL variable %LOGFILE. (If %LOGFILE is undefined, or null, LOGDMP does not write the internal log array.)

After you edit the log file to remove the errors, you can rename the file and use it as input to the batch screen.

Preparing a Calling Screen

You must create a calling screen for the batch screen. In the calling screen, you specify actions related to the batch process.

For input from a sequential file, specify these actions:

1. Define the values of DASL variables %INPUT and %LOGFILE.
2. Open the variables %INPUT and %LOGFILE in the calling screen at the start of the batch process, and close them at the end of the batch process.
3. Call the batch screen to start the batch process.

For input from a %INPUT array, specify these actions:

1. Define the value of %LOGFILE, and open %LOGFILE.
2. In a loop, set the value of %INPUT(%INPUT), and call the batch screen. Repeat the loop until all values in the array have been read.
3. Close %LOGFILE.

You can then call the batch screen.

Designing a Batch Screen

Follow these steps to design a batch screen:

1. Begin with a normal data-entry screen. Copy the screen and assign the BATCH screen attribute to your copy.
2. Replace branching NXTFLD actions with single NXTFLD actions because batch screens contain linear logic. For example, an interactive screen can have a Save, Edit, or Quit field that uses branching logic; a batch screen goes directly to the FILE field.

3. Modify LOOKUP validations so that they do not produce a lookup list because batch screens cannot choose an item from a list. If you need a list to check errors, use the LOOKUP/EXACT validation. This validation generates an error for entries that are not unique. You can then treat non-unique entries as expected errors, and define an error message for them in %ERROR.

4. Create an exit field that the DASL software exits to on encountering an error.

   From this exit field, handle special cases or expected error conditions such as end-of-file. For example, you can add a skip field for error processing, and a clean field to clean up the local symbol table.

Related Sections

ERROR
LOG
LOGDMP
Screen BATCHDR in DASL Demo
Screen BATCH in DASL Demo
Screen Driver Variables

Examples

The following example is from the screen BATCHDR, the calling screen for the batch screen BATCH in the DASL demonstration system. BATCHDR contains one field, BATCH, which opens and closes the variables %INPUT and %LOGFILE.

```
Field Name: BATCH                Description: BATCH
Line: 5     Column: 40  Justify: C   Data Name:        Data Length: 0
Prompt: "...Working"
Attributes: BLINK

--------------------Actions---------------------------------------
!%INPUT, %LOGFILE must be defined, opened, and closed 'by hand'
SET %INPUT="phone.lis" ; SET %LOGFILE="logfile.lis"
XECUTE "O %INPUT,%LOGFILE:NEWV" ; DO BATCH ; XECUTE "C %INPUT,%LOGFILE"
------------------------------------------------------------------
```

The following example is the screen definition of the batch screen BATCH. Note the inclusion of the screen attribute BATCH, and the designation of the field ERROR as the batch exit field.

```
Description: Batch Load Address Book       Type: DATA    Version: 1.0
Group(s): DEMO                                        Created by:
Edited on ____ by ___                                 Compiled on ____
First Line: 1              Last Line: 23      Map Compiled Screen? Y
Attributes: BATCH ; EXIT ERROR
Actions: SET ^ADBK=0:'$D(^ADBK)#2
         SET ^XNAME="Cross Reference for Address Book":'$D(^XNAME)#2
         SET ID=0 ; NXTFLD NAME
```

The following example is from the field NAME on the batch screen BATCH. The LOOKUP/EXACT validation provides a lookup list in the log when an error is encountered.

```
Field: NAME      Description: Enter a new or old name.
Line: 5  Column: 5  Justify: L      Data Name: NAME   Data Length: (30)
Prompt: (Name:)
Help Text: (Enter a name as LAST, FIRST, [MI].)
Attributes: REQUIRED (LCASE)
Validations: LOOKUP/EXACT ; NAME (LOOKUP/VERIFY)
    ----------Actions----------
    NXTFLD STREET:'%FND
    LOCK/TO=9 ^ADBK(ID) ; NXTFLD EVAL:%FND
    NXTFLD LOCKED
    ---------------------------
```

The following example is the exit field ERROR that branches to the fields SKIP and CLEAN in the event of an error. SKIP skips fields on encountering an error in field data; CLEAN cleans up the local symbol table.

```
Field Name: ERROR        Description: Clean up the local symbol table.

    ----------Actions----------
    NXTFLD CLEAN:%ERROR["END OF FILE"
    LOG "--ERROR--   "_%ERROR  ; NXTFLD SKIP
    ---------------------------
```

The following example is the SKIP field that the ERROR field branches to with a NXTFLD.

```
Field Name: SKIP        Description: SKIP FIELDS ON ERROR
Line: 23     Column: 1   Justify: L    Data Name:   Data Length: 30
Prompt: "skip"
    ----------Actions----------
    NXTFLD SKIP:%RES'="*" ; LOGDMP
    NXTFLD SAME
    ---------------------------
```

Transaction Processing

Transaction processing is a type of computer processing in which:

- The user performs a limited, closed set of simple database updates called a *transaction*. Each transaction has a defined starting point and ending point.
- Many users can perform transactions concurrently.
- Transactions take place in *realtime*; that is, the result of each completed transaction is permanent and immediately visible to all users on the system.

You can use the TSTART and TCOMMIT actions to create transactions that allow for transaction processing in DASL applications.

For more information about transaction processing in VAX DSM, see the *VAX DSM Database Operations Guide*.

Explanation

A transaction is a set of updates to that database that exhibits the following properties:

- Atomicity

  Within each transaction, all individual operations must succeed or else all fail and the transaction is incomplete. Atomicity ensures that there are no partially completed transactions.

- Consistency

  Consistency occurs when a successful transaction changes the database from one consistent state to another consistent state.

- Isolation

  Transactions must not interfere with each other's database updates. This means that transactions can process concurrently with other transactions and that database updates do not occur until each transaction is committed. The results produced by processes running concurrently are the same as if the processes had run serially.

- Durability

  Once a transaction is committed, updates to the database that occur within the transaction are permanent. You cannot lose the results of a committed transaction because of a failure in a hardware component or in the storage media.

ACID is a mnemonic device that describes these properties. All transactions must exhibit the properties of ACID.

Looking at a Transaction

An update to the address book in the DASL application system is an example of a simple transaction. The actions that occur when you update the address book are:

1. You enter a name.
2. The system responds with more information about the address book entry if the name is an existing one, or accepts a new name.
3. You edit existing information or add new information to the address book entry.
4. The system validates the input data.
5. The system accesses the address book database, records the new or changed information, and updates the address book database.
6. The system notifies you that the address book database has been changed.

In the ADBOOK data-entry screen, you can define a transaction for step 5 of this procedure to record changes in the database. The following example shows some of the DASL actions that you can use within the FILE field of the ADBOOK data-entry screen to create a transaction:

```
------------------ Actions -------------------------------
     TSTART/TIMEOUT=99 "ADBOOK":COND=1
     COUNT IDNEXT:'ID ; SET ID=IDNEXT:'ID
     FILE NAME,STREET,CITY,STATE,ZIP,BTHDAY,PHHOME,PHWORK
     TCOMMIT ; NXTFLD SAME
----------------------------------------------------------
```

The ADBOOK transaction exhibits the properties of ACID because:

- There are several steps within the update transaction. If one step fails, then all steps fail and the database remains unchanged. Atomicity is preserved.
- Consistency occurs because a committed transaction permanently changes the database from one state to another state.
- Isolation occurs when no two users can update the same database information simultaneously. The programmer must use the DASL LOCK command to lock data so that no two users can update the same information. In the address book application, the screen ADBOOK uses the LOCK action in the NAME field. Whenever one user enters a name, the LOCK action ensures that the database entries for that name are not available to other users until that user has made changes to the database and released the lock with the UNLOCK action. The transaction in the FILE field occurs while the lock is in effect. Proper use of the LOCK and UNLOCK actions ensures isolation.

- Once an ADBOOK transaction has been committed to the database, it cannot be lost. The change to the database is permanent, exhibiting durability.

Using the TSTART and TCOMMIT Actions

You must bracket each transaction for transaction processing with a TSTART action and a TCOMMIT action.

The TSTART action marks the beginning of a transaction and takes the following syntax:

TSTART{/TIMEOUT=n} SP **"name"**

where:

/TIMEOUT=n	is an optional timeout that specifies that the system abort the transaction if the transaction is not committed within **n** seconds
"name"	is the name of the transaction that is written to the journal record

The TCOMMIT action marks the end of a transaction: it completes, or *commits*, the transaction. The TCOMMIT action takes the following syntax:

TCOMMIT{/qualifier}

where:

qualifier	determines the confirmational message that the system displays when a transaction is committed. The qualifiers are: **/NOMSG** — to suppress the display of the system default confirmational message **/MSG="argument"** — to display a message other than the system default message. **"argument"** can contain a MUMPS expression.

All operations that occur within the TSTART and TCOMMIT commands process as a whole. For example, if a transaction performs a debit from one bank account and a credit to another bank account, both the debit and credit must succeed or the entire transaction fails.

In most cases, if a transaction fails, the DASL software displays an error message, rolls back the database to the values present when the transaction began, and does not try to restart the transaction. However, the TPABORT error message represents a special case. A TPABORT error message can occur through no fault of the program within the transaction; the system can cause a TPABORT error independently of the transaction. For example, a TPABORT error caused by the system can occur when two processes are trying to modify the same block of data and one process fails.

When a TPABORT error occurs, the system aborts the transaction and rolls back the database to the values present when the transaction began. The DASL software, in this special case, tries to process the transaction again.

In a DASL application, the operations that occur within a transaction must be restartable; that is, a transaction must be able to fail and then try again to commit. Because of this requirement, you can use only some DASL commands within transactions. Table 2–17 in the "TSTART" section of this chapter shows the DASL commands that you can use within transactions. All other DASL commands are illegal within transactions.

See the "TSTART" section in this chapter for more information about the commands you can include within a transaction.

Comparing Transaction Processing and Journaling

When you bracket transactions with the TSTART and TCOMMIT actions, the DASL software marks the journal records in the same manner as the LOGON and LOGOFF actions. Therefore, you do not have to use the LOGON and LOGOFF actions for journaling purposes.

However, transactions that are bracketed with the LOGON and LOGOFF actions differ from transactions bracketed with the TSTART and TCOMMIT actions in the following areas:

- The process in which data is stored
- The amount of data that you recover in the event of a disk or CPU failure

When you use the LOGON and LOGOFF actions to journal transactions, the following events occur:

- When the application performs a SET to a global, data is cached in memory, and written to a disk at a later time.
- In the event of a disk failure, you can perform a backup to restore the database to a previous known state and then dejournal all subsequent transactions.
- In the event of a CPU failure, any data that is cached in memory is lost and not recoverable. Therefore, you can lose the results of several transactions.

When you use the TSTART and TCOMMIT actions to bracket transactions, the following events occur:

- Each transaction permanently alters the database when it is committed.
- In the event of a CPU failure, you do not lose any data that has been committed to the database. Therefore, you can lose only the results of any incomplete transactions that were processing when the failure occurred.

In consequence, transactions bracketed by the TSTART and TCOMMIT actions provide a faster recovery in case of CPU failure and also do not require a backup procedure. There is no difference in the recovery from media failure for transactions bracketed by the TSTART and TCOMMIT actions and transactions bracketed by the LOGON and LOGOFF actions.

Related Sections

LOGOFF
LOGON
TCOMMIT
TSTART
VAX DSM Database Operations Guide

Define Option Screens Option

Option screens provide the menu structure for your applications. You use option screens to create menus that logically group data screens and reports. You can also use DASL option screens as a menu system, without using DASL data screens or the DASL Data Dictionary.

Overview

Each option screen is a menu that lists user options. Each option can call any of the following:

- Data screen
- Option screen
- Text screen
- VAX DSM routine

You can identify each option by a number or a *keyword*.

An option's number represents the option's sequential position on the menu. The list of options displays in ascending numeric order from 1 to **n** where **n** is any number greater than 1.

Whenever an application user enters the number associated with a menu option, the DASL software transfers control to the screen or VAX DSM routine associated with the option. When the screen or VAX DSM routine completes execution, the DASL software returns control to the menu.

The *keyword* is an optional alternate way to choose menu options. Keywords are alphabetic strings that are parts of the option names. If you define keywords when you create an option screen, the DASL software displays the keyword string in uppercase characters. For example, if you enter the keyword **create** when you define an option you name "Create Employee Records", the DASL software displays the following option choice:

```
1. CREATE Employee Records
```

A user can then type in the shortest unique string in the keyword as an alternate method of choosing the option from the menu, for example:

```
Select Option:    CR Return
```

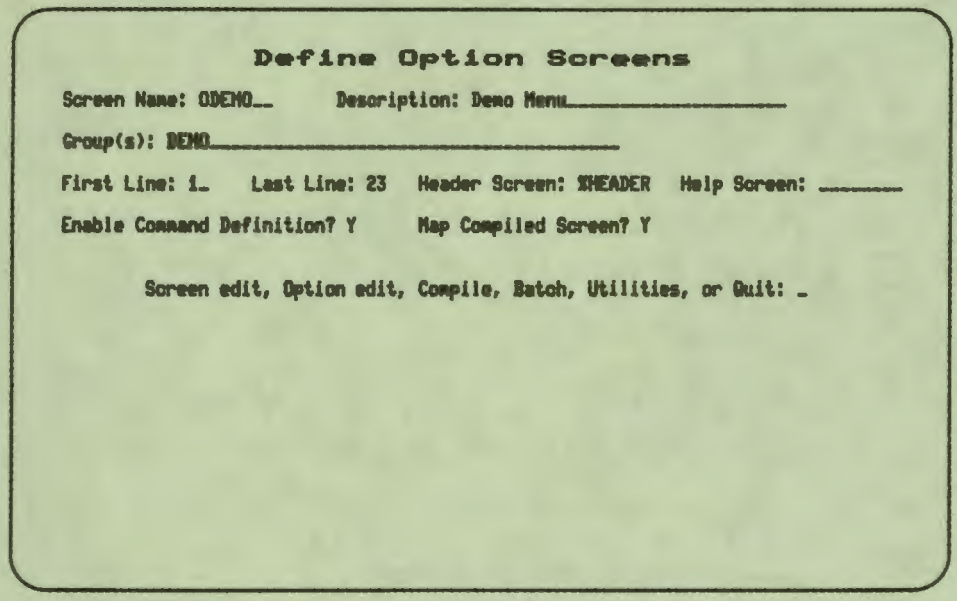
Creating Option Screens

When you create a new option screen, you must specify two types of information:

- Screen definition information
- Option information

Figure 2–6 shows the first screen you see when you choose the Define Option Screens Option. In this screen, you provide definitions that apply to the entire screen.

Figure 2–6 Define Option Screens Screen #1



Screen Definition Information for Option Screens

Table 2–22 lists the screen definition information you can specify for each option screen you create.

Table 2–22 Option Screen Definition Information

Item	Description
Screen Name	The name of the option screen, a standard DASL name of one alphabetic character followed by up to six alphanumeric characters.
Description	A text description of the option screen up to 35 characters in length that appears in lookup lists.
Group(s)	The names of the DASL groups to which the option screen belongs. The group name is a required item that the DASL software uses to classify screens by application or by package. Before you can assign an option screen to a group, you must define the group through the Group Dictionary Option of the Development Environment Menu.

Table 2–22 (Cont.) Option Screen Definition Information

Item	Description
First Line	A numeric value specifying the first line of the data screen display. The first-line value must be between 1 and 23.
Last Line	A numeric value specifying the last line of the data screen display. The last-line value must be between the first-line value and 23.
Header Screen	The name of the screen that specifies the appearance and wording of the option screen's header or title. The screen is normally a data screen with one or more display-only fields. The DASL software supplies a default header screen, %HEADER. The format of the option section is fixed. The option list starts one line below the last line specified in your header screen. The DASL software displays options on every other screen line, using a single or double column depending on the number of options available to the user at run time.
Help Screen	The name of a screen the DASL software is to call whenever a user presses Help when in the menu screen. Normally, the help screen explains how the options function or what they are used for.
Enable Command Definition	A flag used by the DASL software to determine if command definition is allowed. If the flag value is Y, the user can bypass menu operations by defining and using alphanumeric commands to access options. If the flag value is N, the user cannot define and enter alphanumeric commands to access options.
Map Compiled Screen	A flag used by the Build Mapped Section Utility of the Development Environment Menu. If the flag value is Y, the compiled data screen is included in a generated command procedure that interfaces with the VAX DSM Mapped Section Utility (^%RMBLD). If the flag value is N, the compiled data screen is not included.

Editing an Existing Screen

If the screen is an existing one, the DASL software fills in the screen information after you enter the screen name and displays the following menu prompt:

```
Screen edit, Option edit, Compile, Batch, Utilities, or Quit:
```

Use this prompt to perform any of the following:

- Edit the screen definition information.
- Access the second Define Option Screens screen to edit the option information.
- Compile the screen.
- Use the utilities to delete, copy, print, or add documenting comments to the screen.
- Quit and start over.

Editing the Screen Definition Information

You can change any of the screen definition information by entering S in response to the menu prompt. You can then change any of the specifications and save your changes.

Option Information

When you select Option edit at the prompt, the DASL software displays a second Define Options Screen screen where you specify option information.

Figure 2–7 shows the second screen of the Define Option Screens Option.

Figure 2–7 Define Option Screens Screen #2

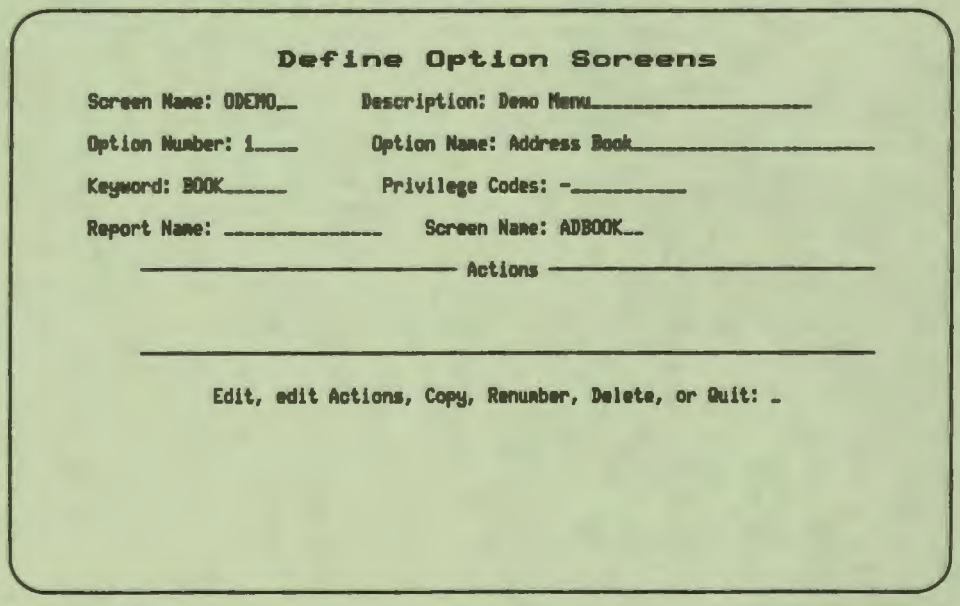


Table 2–23 shows the items you can specify for each option.

Table 2–23 Option Information

Item	Description
Screen Name	The name of the option screen.
Description	A text description of the option screen.
Option Number	The number of the option as it appears on the menu.
Option Name	The name of the option as it appears on the menu.
Keyword	A unique string of characters from the option name that the user can enter to select the option after the "Select Option:" prompt. The keyword appears in boldface uppercase letters on the menu.
Privilege Codes	A list of privilege codes, defined in the Privilege Code Dictionary Option of the Development Environment. In order for the option to be available and displayed to a user, at least one of the option's privilege codes must match one of the user's privilege codes. If you leave this field blank, the DASL software does not display the option.
Report Name	The name of a report called from the option.
Screen Name	The name of a screen called from the option.
Actions	Actions to be performed when the option is selected, such as calling a VAX DSM routine. Actions are entered in a scroll region. If you specify an entry reference, you must include the circumflex (^) before the routine name.

Editing an Existing Screen

If the screen is an existing one, the DASL software fills in the option information after you enter the option number, and then displays the following menu prompt:

```
Edit, edit Actions, Copy, Renumber, Delete, or Quit:
```

Use this prompt to do any of the following:

- Edit the option information.
- In the scroll region, edit the actions called by the option.
- Copy the screen.
- Renumber the option.
- Delete the option.
- Quit and start over.

Editing the Option Information

Enter O at the menu prompt to edit the option information. The DASL software displays the second screen of the Define Option Screens Option. You can then edit any option and save your changes.

To add options, give each new option a decimal number that indicates the position in which that option appears on the screen. For example, if the option screen consists of options numbered 1 to 4, you can add a new option numbered 1.1. Your new option 1.1 appears on the option screen between option 1 and option 2.

The maximum number of options you can define depends on the number of the lines occupied by the header screen.

Renumbering the Option

When you select Renumber at the decision prompt, the DASL software prompts you for a new option number as follows:

```
Renumber:
```

The DASL software then repositions the options by option number in ascending numeric order. For example, an initial lookup list of options appears as follows:

```
Option: *

      Name                        Key Word        Priv's        Screen

1. 1 ADDRESS BOOK                 BOOK             @            ADBOOK
2. 2 PRINT OPTIONS                PRINT            @            OPRINT
3. 3 USER UTILITIES               USER             @            %OAPP

Select Item Number:
```

If you renumber Option 1, Address Book, to Option 6, the lookup list appears as follows:

```
Option: *

      Name                        Key Word        Priv's        Screen

1. 2 PRINT OPTIONS                PRINT            @            OPRINT
2. 3 USER UTILITIES               USER             @            %OAPP
3. 6 ADDRESS BOOK                 BOOK             @            ADBOOK

Select Item Number:
```

The renumbered option also displays on the option screen in its new position.

At the Renumber prompt, if you enter an option number that is currently defined, the DASL software displays the following prompt:

```
OPTION 2 IS ALREADY DEFINED. OK TO OVERWRITE?
```

If you enter Y at this prompt, the DASL software deletes the previously defined option, assigns the option number to the current option, and resequences all options.

Compiling Option Screens

After you define a screen and all its options, you can return to the decision prompt on the first Define Options Screens screen to compile your screen. The compile operation generates a VAX DSM routine from your screen definition.

Enter C to compile the screen in your current process. The DASL software then compiles the screen and returns control to your terminal only when it completes the compilation.

Enter B to compile the screen in a batch process.

After you define an option screen, you can also compile it through the Compile Screens Option.

Using the Utilities

Enter U at the menu prompt to delete, copy, print, or add documenting comments to the screen. The DASL software then displays the following decision prompt:

```
Edit comments, Delete, Copy, or Print:
```

To add comments to document the screen, enter E. You then see a comments screen that enables you to specify the creator, editor, and version number of the screen, and also enter text (in a scroll region) to document the screen.

To delete the screen, enter D and verify your choice where prompted.

To make a copy of the screen under a different name, enter C and specify the new screen name. You can later edit the new screen name to change any of the copied values.

To print the screen, enter P. The DASL software displays a device selection screen that you use to produce the screen through a batch process or in your current process. If you choose to use your current process, you can display the screen on your terminal, enter .P to print it on the printer attached to your terminal, print it on any VMS device, or write it to a file.

Running Screens

Use the Run Screens Option of the Screen Driver Menu to run compiled screens.

Using the Print Option Structure Option

You can obtain a structured listing of the screens in your menu system by using the Print Option Structure Option of the Screen Driver Menu.

Using Option Screens in an Application

In an installed application, application users access screens and reports through menus. The DASL software determines which options to display by comparing the option privileges with the user's privileges. If more options are available than can be listed in one column, the DASL software displays the options in two columns.

When a user selects an option, the DASL software performs actions and calls the screen or report associated with the option. The name of the selected option is available in the DASL variable %OPN. The title of the screen you call is set to the value of %OPN. When the user completes the action, the DASL software returns control to the previous option screen.

Setting the Variables %OP(0) and %OPN

Application users usually access a menu screen from the login screen (which is tied to the application's Security System). In that case, the variables %OPN and %OP(0) are set by the login screen. The variable %OPN contains the title of the menu screen, and the variable %OP(0) contains the subtitle of the screen.

However, when you bypass the Security System or call a menu screen from a routine, you must set %OPN to the title of the screen and %OP(0) to the subtitle of the screen. By default, %OPN is set to "Main Menu."

As you select options from the main menu that call other screens, the title of the main menu screen becomes the subtitle of submenu option screens. The name of the option you select becomes the title of the submenu option screen.

The DASL software skips option screens that have only one available option, and invokes the option immediately.

Related Sections

Compile Screens Option
Print Screen Definitions Option
Privilege Code Dictionary Option, Chapter 5
Run Screens Option
Screen Driver Utilities
Screen Driver Variables
Screen ODEMO in DASL Demo

Examples

The following example shows the screen definition information for the Main Menu of the DASL demonstration system. You specify this information in the first screen of the Define Option Screens Option.

```
Description: Demo Menu                          Type: OPTION     Version:
Group(s): DEMO                                             Created by: GLP
Edited on 29-Mar-89 by GLP                              Compiled on 29-Mar-89
First Line: 1                      Last Line: 23  Map Compiled Screen? Y
Enable Command Definition? Y                      Header Screen: %HEADER
```

The following example shows the option information for Option 1 on the demonstration Main Menu. You specify this information in the second screen of the Define Option Screens Option.

```
1. Address Book                                    Keyword: BOOK
   Privilege Codes: -                           Report Name:
   Screen Name: ADBOOK
```

Command Mode

Command mode is an option for advanced users that you can allow in your applications. Command mode is a method of bypassing the menu structure and going directly to options.

Explanation

A command is a text string that you or application users have associated with a particular DASL option. When a user enters that text string after the "Select Option:" prompt on an option screen, the DASL software executes the option.

Command mode is similar to a VAX DSM DO command. When the DASL software completes the action specified by the command, the DASL software returns to the menu from which the user entered the command.

Command-mode commands are not menu specific. This means that you can enter any command-mode command on any menu in an application in which that command was defined. The DASL software then immediately performs the action, even if the action invokes an option that is several menu levels away.

Suppose you define the command ENROLL for option number 1, "Enroll Students", on an option screen titled "Student Records" that is part of a management information system for a school system. In the future, whenever a user enters ENROLL in response to any "Select Option:" prompt in the management information system, the application calls the screen or routine associated with the option titled Enroll Students.

Command mode also provides lookup capabilities. That is, a user can enter an asterisk (*) after the "Select Option:" prompt and see a lookup list of all available command-mode commands.

Command Types

The DASL software has the following types of command-mode commands:

- System commands
- User commands

System Commands

System commands are available to all application users who have the Security System privileges to use the option associated with the command. System commands can be defined by the manager of an application or by any user with system manager (+) privileges.

When you attempt to define a system command, the DASL software performs the following checks:

1. The DASL software checks the command definition flag set through the Site Parameters Option of the Development Environment Menu to make sure that system commands can be set in the application.
2. If system commands can be set, the DASL software checks the stored definition of the current option screen to make sure that the command definition flag is set to Y.
3. If the enable command definition flag is set to Y, the DASL software checks the user and device privileges to make sure that you have the @ (programmer) or + (system manager) privilege.

If all three checks are passed, you can define a system command.

User Commands

User commands are commands that individual users define for their own use. User commands are available only to the user who defines them.

Whenever you attempt to define a user command, the DASL software performs the following checks:

1. The DASL software checks the command definition flag set through the Site Parameters Option of the Development Environment Menu to make sure that user commands can be set in the application.
2. If user commands can be set, the DASL software checks the stored definition of the current option screen to make sure that the enable command definition flag is set to Y.

If both checks are passed, the DASL software allows you to define a user command.

Command Definition

You use a command definition screen to create system or user commands in the DASL software or in any applications you write through the DASL software. Press Command Mode (GOLD C on a VT100 keyboard or the F20 key on a VT200-series keyboard) after the "Select Option:" prompt on the option screen that contains the option you want to associate with the command. The DASL software then displays the following prompt:

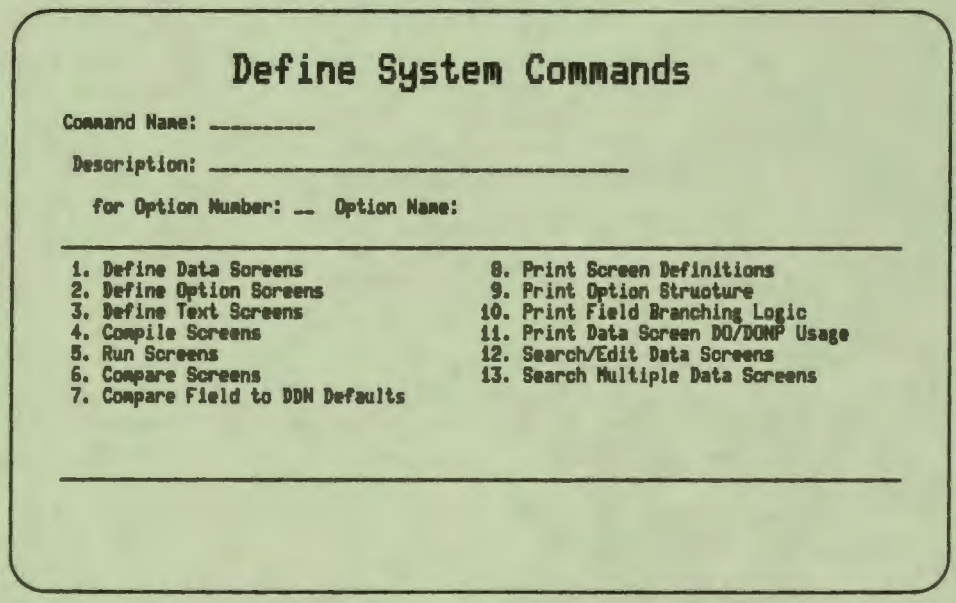
```
Define User or System Commands:
```

Enter U at this prompt to define a user command. The DASL software then displays a Define User Commands screen.

Enter S at this prompt to define a system command. The DASL software then displays a Define System Commands screen.

The Define User Commands screen and the Define System Commands screen are identical with the exception of their titles. Figure 2–8 shows a command definition screen.

Figure 2–8 Command Definition Screen



Note that the DASL software lists all the options for the option screen in a scroll region. You select the option number for the option for which you want to define a command from this list.

Table 2–24 lists the information you specify for each command you define.

Table 2–24 Command Definition Information

Item	Description
Command Name	Name by which the command can be invoked. When you enter a new name, the DASL software performs a lookup in the list of previously defined commands to make sure that the name is unique.
Description	Brief description of the command.
Option Number	Number of the option to associate with the command.

After you have entered command definition information, the DASL software displays the "Save, Edit, or Quit" prompt. You can then save new command definition information, edit existing information, or quit from the screen.

Using Command Mode

After you define a command, you can invoke that command from any option screen. At any "Select Option:" prompt, enter your defined command to move to a specified screen or VAX DSM routine. You can move easily to any screen or VAX DSM routine in your application using command mode.

When the DASL software completes the action specified by the command, the DASL software returns to the screen from which you entered the command. For example, you have called Screen B from Screen A, and then called Screen D from Screen B using command mode. When you exit from Screen D, the DASL software returns to Screen B. When you exit from Screen B, the DASL software returns to Screen A.

As noted previously, a user's ability to invoke a command depends on the Security System privileges of the user's process. If you change the privileges of an option for which you have already defined system commands or users have already defined user commands, the commands associated with that option are disabled. When the user invokes an illegal command, the terminal beeps and displays the following message:

```
Invalid Command Data. Command Disabled.
```

Related Sections

Define Option Screens Option
Security System Option, Chapter 6
Site Parameters Option, Chapter 5

Define Text Screens Option

Text screens display descriptive information one page at a time. You can use text screens as:

- Help screens when users request online help
- Menu options (when you create prototypes of screens)
- Online documentation

Explanation

The single line of help text you can associate with DASL data-screen fields often does not provide enough user information. If a particular option or field is complex, you can associate a help screen with the option or field. You can specify these help screens when you define fields in the Define Data Screens Option and options in the Define Option Screens Option. You can also call a text screen as a menu option.

You can use text screens to display multiple pages of information on the terminal screen. Whenever a user presses Help twice on an option screen or at a field associated with a text screen as a help screen, the DASL software clears the display between the specified first and last lines of the text screen as a scroll region, and fills the scroll region with text from the text screen.

For only one page of text, the DASL software displays the following prompt:

```
Press RETURN to exit.
```

For multiple pages of text, the DASL software displays the following prompt:

```
Page C of L; Press RETURN to continue.
```

where

C is the page number of the current page of text

L is the page number of the last page of text

On the last page of a help screen with multiple pages of text, the DASL software displays the following prompt:

```
Page L of L; Press RETURN to exit.
```

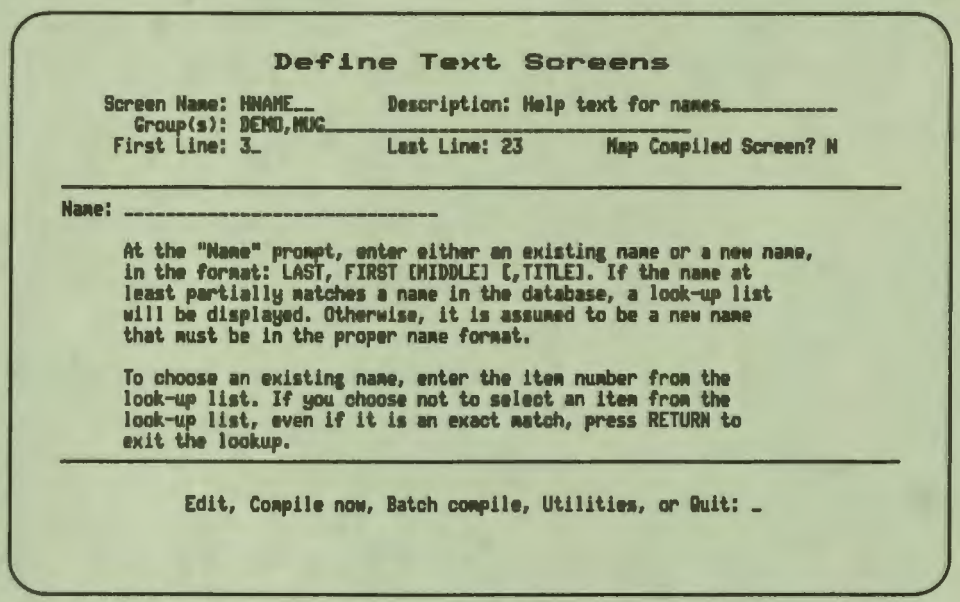
When there are multiple pages of text, you can enter B to go back a page and N to redisplay the page numbered N.

After the user presses Return, the DASL software displays any remaining text in the text screen, or returns to the point from which the user requested help.

Define Text Screens Screen

You define text screens for your applications through the Define Text Screens Option. Figure 2–9 shows the screen you see when you choose the Define Text Screens Option.

Figure 2–9 Define Text Screens Screen



Creating a New Screen

Enter a name (one letter followed by 0 to 6 alphanumeric characters) after the "Screen Name:" prompt to create a new text screen. The DASL software then prompts you to fill in screen definition information.

Table 2–25 lists the information you specify for each text screen you create.

Table 2–25 Text Screen Definition Information

Item	Description
Screen Name	The name of the text screen, a standard DASL name of one alphabetic character followed by up to six alphanumeric characters.
Description	A text description of the screen up to 35 characters in length that appears in lookup lists.
Group(s)	The names of the DASL groups to which the text screen belongs. The group name is a required item that the DASL software uses to classify screens by application or by package. Before you can assign a screen to a group, you must define the group through the Group Dictionary Option of the Development Environment Menu.
First Line	A numeric value specifying the first line of the data screen display. The first-line value must be between 1 and 23.
Last Line	A numeric value specifying the last line of the data screen display. The last-line value must be between the first-line value and 23.
Map Compiled Screen	A flag used by the Build Mapped Section Utility of the Development Environment Menu. If the flag value is Y, the compiled data screen is mapped: it is included in a generated command procedure that interfaces with ^%RMBLD, the DSM Mapped Section Utility. If the flag value is N, the compiled data screen is not mapped.
Text	The text to display on the screen, entered in a scroll region.

Keep the following comments in mind when you create text screens:

- You must enter the text in the format in which you want it displayed.
- You can enter text directly in a scroll region or use the EDT™ editor. If you use the EDT editor, you can create an external VMS text file for help text and include it into the text screen through EDT.
- You can use the $PAGE function to cause a page break.
- You can use Screen Driver variables within the text of the text screen. Enclose the variable within the (|) character as follows:

```
|%OPN|
```

Editing an Existing Screen

If the screen you specify at the "Screen Name:" prompt is an existing one, the DASL software fills in the screen specification information and displays the following menu prompt:

```
Edit, Compile now, Batch compile, Utilities, or Quit:
```

The following sections describe your choices at this prompt.

Editing Screen Definitions or Text

Enter E to edit the screen definitions and enter or edit text. The DASL software then displays the following menu prompt:

```
Specification edit, edit Text in scroll region, or Editor:
```

- Enter S to change any of the screen definition information again. You can then change any of the specifications and save your changes.
- Enter T to enter the text line by line into the scroll region. You then can enter the text as you enter any scroll region list.
- Enter E to use the VAX EDT editor to enter the text manually or read in a text file. The DASL software then places you in EDT. You can use any of the EDT capabilities to create or edit a file. When you finish, use the following procedure to exit from EDT and return to the DASL software:

  1. Press Ctrl/Z.
  2. Type exit.
  3. Press Return.

  You can then save your edits when you see a "Save, Edit, or Quit:" decision prompt.

Compiling and Running Text Screens

The compile operation generates an executable VAX DSM routine from your screen definition.

- Enter C to compile the screen in your current process. The DASL software then compiles the screen and returns control to your terminal only when it completes the compile operation.
- Enter B to compile the screen using a command procedure. The DASL software then submits a batch job and returns control to your terminal immediately.

After you define a text screen, you can also compile it through the Compile Screens Option.

Using the Utilities

Enter U to delete, copy, print, or add documenting comments to the screen. The DASL software then displays the following utilities decision prompt:

```
Edit comments, Delete, Copy, or Print:
```

- To add comments to document the screen, enter E. You then see a comments screen that you can use to specify the creator, editor, and version number of the screen, and also enter text (in a scroll region) to document the screen.
- To delete the screen, enter D and verify your choice where prompted.
- To make a copy of the screen under a different name, enter C and specify the new screen name. You can later edit the new screen name to change any of the copied values.
- To print the screen, enter P. The DASL software displays a device selection screen that you can use to produce the report through a batch process or in your current process. If you choose to use your current process, you can display the report on your terminal, enter .P to print it on the printer attached to your terminal, print it on any VMS device, or write the report to a file.

Starting Over

To start over, enter Q. The DASL software moves the cursor back to the "Screen Name:" prompt.

Related Sections

Define Data Screens Option
Define Option Screens Option

Screen Driver Utilities

The Screen Driver Menu includes several options that are user utilities. You can use these options to compile and run screens, to search and edit a screen, or to run various reports about screens you have defined. These options include:

- Compile Screens Option
- Run Screens Option
- Compare Screens Option
- Compare Field to DDN Defaults Option
- Print Screen Definitions Option
- Print Option Structure Option
- Print Field Branching Logic Option
- Print Data Screen DO/DONP Usage Option
- Search/Edit Data Screens Option
- Search Multiple Data Screens Option

The following sections contain examples of the reports that these options provide.

Specifying Screens in Utility Scroll Regions

When you use some Screen Driver utilities, the DASL software displays a scroll region so you can specify the screens to compile or include in a report.

You can list the screens by entering one or more of the following:

- Screen names
- An asterisk (*) for all screens
- A partial name preceded or followed by an asterisk (*)
- An at sign (@) followed by the name of a defined DASL group for all screens in that group
- A range of screens in the form name1-name2

If you choose a range in the form name1-name2, the DASL software includes all screens that fall within the specified range alphabetically by name.

If you specify a range of screens, you can verify the screens in the range. When the DASL software displays the "Continue, eXpand wildcards, Edit, or Quit:" prompt, enter X. The DASL software then displays all screens that fall within the range.

To exclude any of the listed screens, enter D in response to the "Continue, Delete, or Quit:" prompt that the DASL software displays after expanding the list. You then move back to the top of the scroll region. You can use the scroll region editing keys to change any of the entries.

> **Note:** The other choices at this prompt are C to continue to the device selection screen, E to edit the list of screens, or Q to start over again.

Choosing the Destination

When you finish choosing screens, the DASL software displays the device selection screen. Use the device selection screen to compile screens or produce reports through a batch process or in your current process. In the case of reports, the device selection screen also prompts you to select a device where you print the report.

If you use your current process, you can display the report on your terminal, enter .P to print it on the printer attached to your terminal, print it on any VMS device, or write the report to a file.

Compile Screens Option

You use the Compile Screens Option to convert data screen, option screen, and text screen definitions into executable VAX DSM routines.

You can use the Compile Screens Option to compile all screens or compile only the screens you specify. To compile all screens, enter Y in response to the "Compile all screens:" prompt. To compile specified screens, enter N.

Compiling Specified Screens

If you choose to compile specified screens, you then see a scroll region to list the screens to compile.

Choosing a Destination

When you finish specifying screens, the DASL software asks whether you want to compile the screens through a batch process or in your current process. If you are compiling more than one screen, compile through a batch process.

Using the ALL^%DASDCMP Entry Point

You can build a VMS command procedure that calls the DSM entry point ALL^%DASDCMP to compile all screens. The log file from the command procedure will contain the names of all screens compiled as well as error messages for screens that failed to compile.

Compiling Mapped Screens

When you compile a mapped screen, the screen may compile successfully, but DSM may display a warning message as follows:

```
%DSM-W-PGMMAPPED, Warning: ZSAVE on mapped routine
```

This message appears, in part or in full, at the current cursor position on the screen and disturbs the screen display. Press Ctrl/W to refresh the screen.

Related Sections

Define Data Screens Option
Define Option Screens Option
Define Text Screens Option
Run Screens Option

Run Screens Option

You use the Run Screens Option to run the VAX DSM routine associated with any compiled DASL screen.

Explanation

When you use the Run Screens Option, the DASL software prompts you for the screen name and then displays the following information:

- Description — brief description of the screen
- Type — data, option, or text screen

At the "Screen Name:" prompt, you can enter any of the following:

- An exact name
- An asterisk to view a lookup list
- A partial screen name

You can specify any previously compiled data screen, option screen, or text screen. After you verify your choice at the "Run or Quit:" decision prompt, the DASL software runs the screen you specify. If the screen you specify is not a compiled screen, the DASL software displays an error message.

Running Screens from the VAX DSM Environment

You can also run a screen from the VAX DSM environment. To run a screen from DSM, follow these steps:

1. Type D ^%DAS at the DSM prompt (>).
2. Press Return. The DASL software then prompts you for a screen name.
3. Enter the screen name and press Return.

Related Sections

Compile Screens Option
Define Data Screens Option
Define Option Screens Option
Define Text Screens Option

Compare Screens Option

You use the Compare Screens Option to run a report that lists any differences in the definitions of any two specified screens. You can compare any type of screens (data, text, or option). However, the two screens you compare must be of the same type.

Example 2–1 shows a section of a Screen Comparison Report for two data screens.

Example 2–1 Screen Comparison Report

```
2-Mar-89     1:58 PM                    DASL                          Page 1
                          DSM Application Software Library
                                Screen Comparison

Compare Screen: ADBOOK                    With Screen: BATCH
```

Screen Level:	Screen Level:
Description:	Description:
Address Book	Batch Load Address Book
Created By:	Created By:
Edit Date: 25-Feb-89	Edit Date: 27-Feb-89
Compile Date: 25-Feb-89	Compile Date: 27-Feb-89
Edit Time: 3:18 PM	Edit Time: 10:59 AM
Screen Attributes:	Screen Attributes:
EXIT CLEAN	BATCH; EXIT ERROR
Field: AGE	Not present
Field: BTHDAY	Field: BTHDAY
Action 1:	Action 1:
! DATE validation returns %DTI as date in internal ($H) format	NXTFLD PHHOME
Action 2:	Not present
Action 3:	Not present
Field: EDNAME	Not present
Field: EQ	Not present
Not present	Field: ERROR

As Example 2–1 shows, the Screen Comparison Report for data screens compares the screen definitions and all field definitions. If a particular field name is not present in both screens, the report lists the field name and notes that it is not present in the other screen.

Related Section

Screen Driver Utilities

Compare Field to DDN Defaults Option

You use the Compare Field to DDN Defaults Option to run a report that lists any differences in the default values defined at the template, data name, or field levels among the data screens you specify. You can use this option to avoid duplicating values unnecessarily.

Example 2–2 shows a section of a Compare Field to DDN Defaults Report created by the Compare Field to DDN Defaults Option.

Example 2–2 Compare Field to DDN Defaults Report

```
2-Mar-89     2:06 PM                    DASL                          Page 1
                          DSM Application Software Library
              Compare Data Screen Field Values to Data Name Defaults

-------------------------------------------------------------------------------

                              Screen: ADBOOK - Address Book

Field: EDNAME                 Data Name: NAME

Attribute
Field: REQUIRED
  DDN: LCASE
Validation
Field: NAME
  DDN: LOOKUP/VERIFY

    ---------------------------------------------------------------------------

Field: NAME                   Data Name: NAME

Validation
Field: LOOKUP/VERIFY ; NAME
  DDN: LOOKUP

    ---------------------------------------------------------------------------

Field: PHHOME                 Data Name: PHHOME
                              Template: PHONE

    ---------------------------------------------------------------------------
```

As Example 2–2 shows, the Compare Field to DDN Defaults Report lists all fields and data names in the specified screen in alphabetical order. The report also lists, where applicable, the templates on which the data name is based. The report also lists all differences among the fields, data names, and templates.

Related Section

Screen Driver Utilities

Print Screen Definitions Option

You use the Print Screen Definitions Option to run a report that lists all specifications for any data screens, option screens, or text screens you specify.

If you specify data screens, the report includes all field definitions listed in order of field name or field position on the screen. A report for a data screen ends with a rough image of the screen, showing the placement of all fields. Example 2–3 shows a Data Screen Definitions Report.

Example 2–3 Data Screen Definitions Report

```
25-Feb-89     2:05 PM                    DASL                          Page 1
                          DSM Application Software Library
                               Screen Definitions

                             Definition of Screen: ADBKSEL

_____________________________________________________________________________

Description: Print Selected Addresses                 Type: DATA      Version:
Group(s): DEMO, ADDRESS                                          Created by:
Edited on 7-Jan-89 by                                  Compiled on 25-Feb-89
First Line: 1                    Last Line: 23         Map Compiled Screen? N
Attributes: EXIT CLEAN
Actions: NXTFLD NAME

_____________________________________________________________________________

Field Name: TITLE    Description: Title
Line: 1   Column: 20   Justify: C   Data Name:              Data Length:
Prompt: "Print Address"
Attributes: WIDE ; BOLD
    _____________________________________________________________________
Field Name: NAME     Description: Ask name of person to print.
Line: 5   Column: 38   Justify: C   Data Name: NAME        Data Length: (30)
Prompt: (Name: )
Help text: (Enter a name as LAST, FIRST, [MI].)
Attributes: (LCASE)
Validations: (LOOKUP/VERIFY)
    -------------------------Actions-------------------------------------
    NXTFLD CLEAN: %RES="" ; NXTFLD CQ
    ---------------------------------------------------------------------
Field Name: CQ       Description: Continue or Quit.
Line: 8   Column: 40   Justify: C   Data Name:            Data Length: 1
Prompt: "Continue or Quit."
Help Text: "Enter "C" to continue or "Q" to quit."
Attributes: DEMAND ; REQUIRED ; EXIT CLEAN
Validations: TABLE C,Q
    -------------------------Actions-------------------------------------
    NXTSCN: %RES="Q" ; SET %RPNAME="ADBKONE" ; SET %PARM=ID
    DONP %REPDEV ; NXTSCN
    ---------------------------------------------------------------------

Field Name: CLEAN  Description: Kill off local variables used in this screen.

    -------------------------Actions-------------------------------------
    KILL NAME,ID
    ---------------------------------------------------------------------
```

Example 2–4 shows an Option Screen Definitions Report.

Example 2–4 Option Screen Definitions Report

```
2-Mar-89     2:07 PM                    DASL                          Page 1
                          DSM Application Software Library
                               Screen Definitions

                          Definition of Screen: OPRINT

Description: Print Options                          Type: OPTION     Version:
Group(s): DEMO                                            Created by:
Edited on 27-Feb-89 by                                 Compiled on 27-Feb-89
First Line: 1                    Last Line: 23         Map Compiled Screen? N
Enable Command Definition? Y                           Header Screen: %HEADER

                                  OPTIONS

1. Print Entire Address Book                          Keyword: ENTIRE
   Privilege Codes: @                              Report Name: ADBKALL
   Screen Name: %REPDEVH

2. Print Selected Addresses                           Keyword: SELECT
   Privilege Codes: @                              Report Name: ADBKONE
   Screen Name: ADBKSEL
```

Explanation

When you choose the Print Screen Definitions Option, the DASL software prompts you to specify how to list the screen's field definition as follows:

```
List by Alpha or Line and column order?
```

Although this prompt applies only to data screens, you have to respond to it even if you are listing only option and text screens. Enter either A or L.

Enter A to list data-screen fields in alphabetical order. Enter L to list data-screen fields in the order in which they appear on the screen, left to right, top to bottom.

Related Section

Screen Driver Utilities

Print Option Structure Option

You use the Print Option Structure Option to run a report listing all options called from the option screen you specify.

Example 2–5 is a section of a Option Structure Report created through the Print Option Structure Option.

Example 2–5 Option Structure Report

```
2-Mar-89   2:51 PM                    DASL                          Page 1
                        DSM Application Software Library
                          Option Structure from ODEMO

Option Name                                                    Privilege
                                                                String
-------------------------------------------------------------------------

Address Book (ADBOOK)                                             -

Print Options (OPRINT)                                            @

  Print Entire Address Book (%REPDEVH)                            @

  Print Selected Addresses  (ADBKSEL)                             @

User Utilities (%OAPP)                                            @

  Security System (%OSEC)                                         +=
     Classification Dictionary (%USRCLS)                          +

     User Dictionary (%USRFIL)                                    +

     Device Dictionary (%USRDEV)                                  +
```

As Example 2–5 shows, the Option Structure Report includes a listing of all privilege codes required to access the options. If any of the options is an option screen, the report also includes a list of the options called from that screen.

Related Sections

Define Option Screens Option
Screen Driver Utilities

Print Field Branching Logic Option

You use the Print Field Branching Logic Option to run a report listing the branching logic used by the screens you specify.

Example 2–6 shows a Field Branching Logic Report created through the Print Field Branching Logic Option. This report shows the fields and their NXTFLD branching logic.

Example 2–6 Field Branching Logic Report

```
2-Mar-89    2:55 PM                    DASL                          Page 1
                          DSM Application Software Library
                          Data Screen Field Branching Logic

------------------------------------------------------------------------------

                          Screen: ADBKSEL - Print Selected Addresses

Level 1:

        First Fld  ------>   NAME


Level 2:

        NAME       ------>   CLEAN:%RES=""
                             CQ

------------------------------------------------------------------------------
```

Related Sections

DO
DONP
Fields
NXTFLD
Screen Driver Utilities

Print Data Screen DO/DONP Usage Option

You use the Print Data Screen DO/DONP Usage Option to run a report listing any calls made by a specified screen to other screens or to VAX DSM routines.

Example 2–7 shows a DO/DONP Usage Report created through the Print Data Screen DO/DONP Usage Option.

Example 2–7 DO/DONP Usage Report

```
27-Aug-89  2:56 PM                    DASL                          Page 1
                        DSM Application Software Library
                          Data Screen DO/DONP Usage

Screen Name      Calls Screen/Routine  Field Name Tag      Component
-------------------------------------------------------------------------

SCOPT                  SCCOMTS         COMMENT             Action
                       UDEV            PRINT               Action
                       ^%DASDC         COMPILE             Action
                       ^%DASDUTL       COPY2       SCOPY   Action
                                       DEL         SDEL    Action
                       ^%DAUDEV        BATCH       BATCH   Action
```

As Example 2–7 shows, the DO/DONP Usage Report lists all the screens you specify. For each screen, the report includes the following:

- The routine or screen called
- The field at which the call was made
- Any tag (line label) specified as part of an entry reference
- Whether the call was an action or a validation

Related Sections

DO
DONP
Fields
Screen Driver Utilities

Search/Edit Data Screens Option

You use the Search/Edit Data Screens Option to search a screen for a specified string, edit any occurrences of the string, and then recompile the screen.

Explanation

The Search/Edit Data Screens Option is a quick way to inspect or edit a data screen. When you choose the Search/Edit Data Screens Option, you specify the following:

- Name of a data screen
- Character string

The DASL software then displays a scroll region containing all screen and field validations, attributes, actions, and data names on the specified data screen that contain the string.

You can then choose to enter the scroll region and edit any of the displayed items. After you finish, you can save your edits as part of the data screen definition.

You can also select the Search/Edit Data Screens Option by entering U at the prompt on the Define Data Screens screen. Then, enter S to select the Search/Edit Data Screens Option and search the data screen.

Related Sections

Define Data Screens Option
Screen Driver Utilities

Search Multiple Data Screens Option

You use the Search Multiple Data Screens Option to run a report listing all occurrences of a specified string in specified screens.

Example 2–8 shows a Search Multiple Data Screens Report created through the Search Multiple Data Screens Option.

Example 2–8 Search Multiple Data Screens Report

```
6-Oct-89  13:49                    DASL                          Page 1
                      DSM Application Software Library
                           Search Data Screens

    Searching for: "zip"

-------------------------------------------------------------------------

                              Screen: ADBOOK  Address Book
Field: EVAL
Action 1: EVAL STREET,CITY,STATE,ZIP,BTHDAY,AGE,PHHOME,PHWORK,PHOTH1

    ---------------------------------------------------------------------

Field: FILE
Action 6: FILE NAME,STREET,CITY,STATE,ZIP,BTHDAY,PHHOME,PHWORK,PHOTH1

    ---------------------------------------------------------------------

Field: STATE
Action 1: NXTFLD ZIP

    ---------------------------------------------------------------------

Field: ZIP
Data Name: ZIP
Help Text: (Enter a 5 or 9 digit zip code.)
Prompt: (Zip: )

    ---------------------------------------------------------------------
```

Explanation

When you choose the Search Multiple Data Screens Option, you specify the strings to search for. Then, the DASL software displays a scroll region where you specify which data screens to search.

Related Sections

Define Data Screens Option
Screen Driver Utilities