# VAX Common Data Dictionary

digital

Data Definition Language Reference Manual

Data Definition Language
Reference Manual

Order Number: AA–K085D–TE

94-003/049/06

# VAX Common Data Dictionary Data Definition Language Reference Manual

Order Number: AA-K085D-TE

August 1988

This manual describes the VAX Common Data Dictionary Data Definition Language Utility (CDDL) and the elements of a CDDL source file.

digital equipment corporation, maynard, massachusetts

CID #57493

The postpaid Reader's Comments forms at the end of this document request the user's critical evaluation to assist in preparing future documentation.

# Contents

# How to Use This Manual

This manual describes how the VAX Common Data Dictionary software, also referred to in this document as CDD, allows you to enter record definitions directly into the dictionary, using the Data Definition Language Utility (CDDL).

CDD is now a subset of the VAX CDD/Plus software, also referred to in this document as CDD/Plus. Many DIGITAL products, however, continue to function using Version 3.4 or earlier of CDD.

Version 3.4 and earlier of CDD use the DMU format for dictionary definitions, but not the CDO format. This manual is for dictionary users who need to use the DMU format for dictionary definitions.

If Version 3.4 or earlier of the VAX Common Data Dictionary is installed on your system, references in this manual to the "VAX Common Data Dictionary," "Common Data Dictionary," or "CDD" refer to the VAX Common Data Dictionary installed on your system.

If VAX CDD/Plus Version 4.0 or later is installed on your system, references in this manual to the "VAX Common Data Dictionary," "Common Data Dictionary," or "CDD" refer to the DMU format dictionary.

CDD/Plus supports dictionary definitions in two distinct formats:

- DMU **format** includes dictionary definitions that can be created and manipulated with the DMU, CDDL, and CDDV utilities, and other products that do not support the new features of CDD/Plus.
- CDO **format** includes dictionary definitions that can be created and manipulated with the CDO utility, the CDD/Plus call interface, and other supporting products.

## Intended Audience

The audience for this manual includes:

- The data administrator or system manager responsible for organizing the directory hierarchy and creating the record definitions to be stored in the CDD
- Programming supervisors responsible for maintaining portions of the hierarchy and the data definitions stored there
- Programmers responsible for storing new data definitions in the CDD

Before you read this manual, you should read Appendix A of the *VAX CDD/Plus User's Guide*.

## Operating System Information

For information on the compatibility of other software products with this version of CDD/Plus, refer to the System Support Addendum (SSA) that comes with the Software Product Description (SPD). You can use the SPD/SSA to verify which versions of your operating system are compatible with this version of CDD/Plus.

This manual consists of three chapters, five appendixes, and an index.

| | |
|---|---|
| Chapter 1 | Provides a brief overview of the hierarchical structure of the VAX Common Data Dictionary and introduces the use of the Data Definition Language Utility (CDDL). |
| Chapter 2 | Presents complete descriptions of the elements of a CDDL source file. |
| Chapter 3 | Presents a complete description of the CDDL compiler commands, including parameters and qualifiers. |
| Appendix A | Contains SOURCE.DDL, the source file for all the record definitions used as examples in this manual. |
| Appendix B | Contains a CDDL syntax skeleton. |
| Appendix C | Documents compiler error messages. |
| Appendix D | Documents the use of CDDL reserved words. |
| Appendix E | Contains changes made with Version 3.4 of CDD that were not described earlier. |

## Related Documents

For up-to-date references to further information on the topics covered in this manual, see the prefaces of the VAX CDD/Plus manuals in this documentation set.

## Conventions

This section explains the conventions for the syntax and symbols used in this manual:

| | |
|---|---|
| WORD | An uppercase word in a syntax format is a keyword. You must include it in the statement if the clause is used. |
| word | A lowercase word in a syntax format indicates a syntax element that you supply. |
| [ ] | Square brackets enclose optional clauses from which you can choose one or none. |
| { } | Braces enclose clauses from which you must choose one alternative. |
| [\| \|] | Bars in square brackets indicate that you can choose any combination of the enclosed options, but you can use each option only once. |
| <RET> | This symbol indicates the RETURN key. Unless otherwise indicated, end all user input lines in examples by pressing the RETURN key. |
| <CTRL/x> | This symbol tells you to press the CTRL (control) key and hold it down while pressing a letter key. |
| <GOLD-x> | This symbol indicates that you press the GOLD key and then a specified letter key consecutively. |
| . . . | A horizontal ellipsis means you can repeat the previous item. |
| .<br>.<br>. | A vertical ellipsis in an example means that information not directly related to the example has been omitted. |
| Color | Color in examples shows user input. |

## References to Products

CDD is a member of the VAX Information Architecture, a group of products that work with each other and with VAX languages conforming to the VAX calling standard to provide flexible solutions for information management problems.

VAX Information Architecture documentation explaining how these products interrelate is included with VAX CDD/Plus documentation. VAX Information Architecture documentation is also available separately. Contact your DIGITAL representative.

The CDD documentation to which this manual belongs often refers to products that are part of the VAX Information Architecture by their abbreviated names:

- VAX ACMS software is referred to as ACMS.
- VAX CDD/Plus software is referred to as CDD/Plus.
- VAX CDD software is referred to as CDD.
- VAX DATATRIEVE software is referred to as DATATRIEVE.
- VAX DBMS software is referred to as VAX DBMS.
- VAX Rdb/VMS software is referred to as Rdb/VMS.
- VAX TDMS software is referred to as TDMS.
- VIDA software is referred to as VIDA.

# The VAX Common Data Dictionary 1

A typical information management system today includes a combination of languages and language processors using the same data files and record definitions to perform different tasks. With information shared by various users, there is an obvious need to guarantee the accuracy and reliability of the data. Database management systems meet this need by providing central storage of and control over an organization's data.

The VAX Common Data Dictionary (CDD) performs a similar function, not for data, but for data definitions. The CDD is a central repository for data descriptions and definitions shared by VAX languages and by VAX information management processors.

## 1.1 The CDD Directory Hierarchy

With the CDD, you use a hierarchical directory structure to organize and arrange your data definitions. You collect related data definitions in dictionary directories in much the same way you use VAX/VMS directories to collect related files. Dictionary directories can own other directories, or they can own dictionary objects, which are the data descriptions stored in the dictionary. There are several types of CDD dictionary objects including VAX DBMS sets and realms, VAX DATATRIEVE domains and procedures, and VAX CDD record definitions.

The CDD also allows you to create special directories called subdictionary directories. Subdictionary directories function exactly like dictionary directories, but they are stored in separate physical files for security or accounting reasons. In practice, users are unaware of whether they are accessing dictionary directories or subdictionaries.

Figure 1-1 shows the hierarchical relationships in the sample dictionary that is the source of all the examples in this manual.

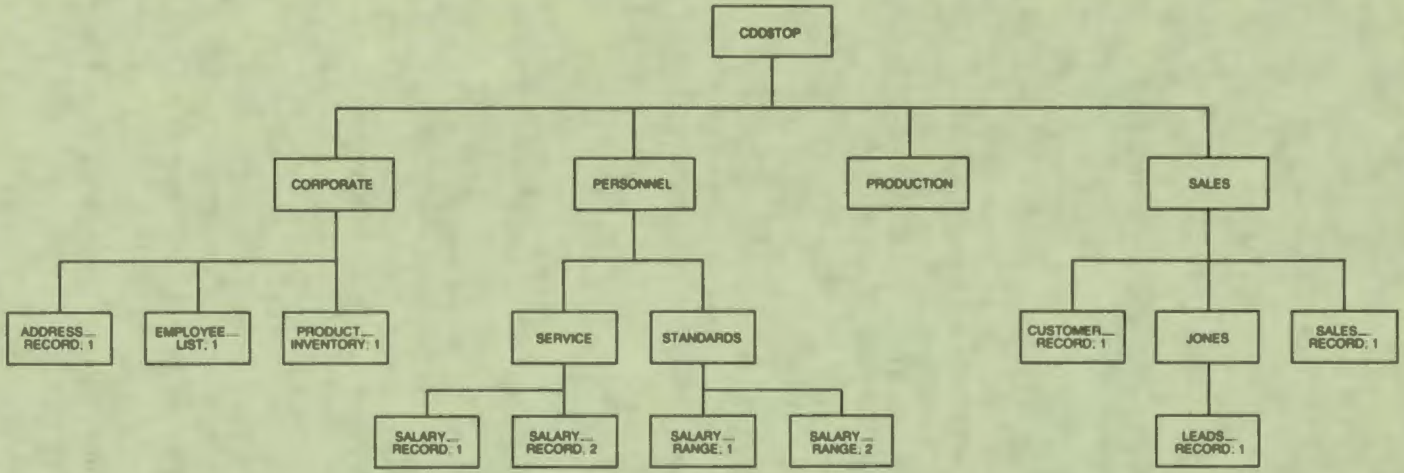


**Figure 1-1: Sample Dictionary Hierarchy**

As you can see from the figure, one dictionary directory, CDD$TOP, owns the entire dictionary structure. CDD$TOP is the permanently assigned name for this **root dictionary directory**. Below CDD$TOP are the directories CORPORATE, PRODUCTION, and SALES, and the PERSONNEL subdictionary. These dictionary and subdictionary directories organize the information stored in the CDD. You can use terms borrowed from family tree relationships to describe CDD hierarchical relationships:

- PRODUCTION has no children and no descendants.
- The record definitions ADDRESS_RECORD;1, EMPLOYEE_LIST;1, and PRODUCT_INVENTORY;1 are the three children of CORPORATE.
- PERSONNEL's two children are directories: SERVICE and STANDARDS. Each of these directories is the parent of two record definitions. SALARY_RECORD;1 and SALARY_RECORD;2 are children of SERVICE, and SALARY_RANGE;1 and SALARY_RANGE;2 are the children of STANDARDS.
- SALES is the parent of both a dictionary directory, JONES, and of two dictionary objects, CUSTOMER_RECORD;1 and SALES_RECORD;1. JONES, in turn, is the parent of LEADS_RECORD;1.
- LEADS_RECORD;1 is a descendant of SALES, and SALARY_RECORD;1 is a descendant of PERSONNEL.
- PERSONNEL is an ancestor of SALARY_RECORD;1, SALARY_RECORD;2, SALARY_RANGE;1, and SALARY_RANGE;2.

### 1.1.1 Full Path Names

To refer to a particular dictionary directory or object within the directory hierarchy, you use its dictionary path name. A full path name is the string of given names connecting CDD$TOP to the given name of the target dictionary directory or object. You separate given names from one another with periods, just as you do with VMS directories. The dictionary directory JONES, for example, has the following full path name:

```
CDD$TOP.SALES.JONES
```

Because the CDD can contain multiple versions of dictionary objects, each dictionary object has a version number associated with it. The version number is separated from the name of the object by a semicolon. For example, the full path name of version 2 of SALARY_RANGE is:

```
CDD$TOP.PERSONNEL.STANDARDS.SALARY_RANGE;2
```

You can specify the highest version of a dictionary object with an absolute version number or with no version number at all. For example, either of the following path names specifies the highest existing version of SALARY_RANGE, version 2:

```
CDD$TOP.PERSONNEL.STANDARDS.SALARY_RANGE
CDD$TOP.PERSONNEL.STANDARDS.SALARY_RANGE;2
```

### 1.1.2 Relative Path Names

If you define the logical name CDD$DEFAULT as the path name of a particular dictionary directory, that directory becomes your default dictionary directory. When you have defined a default directory, you can identify a dictionary directory or object by its relative path name, which includes only the names of those directories connecting the default directory to the target directory or object.

If, for example, your default dictionary directory were CDD$TOP.PERSONNEL, you could identify the dictionary object SALARY_RECORD;1 with either of the following path names:

```
CDD$TOP.PERSONNEL.SERVICE.SALARY_RECORD;1
SERVICE.SALARY_RECORD;1
```

## 1.2 History Lists

The CDD's optional history list feature allows you to document any operation on a dictionary directory or object. You can store documentary comments in the history list of the directory or object. You can use history lists to monitor CDD processing and to plan additions or changes to the dictionary hierarchy.

## 1.3 Access Control Lists

The CDD controls the security of the dictionary with access control lists associated with each dictionary directory or object. You can grant or deny nine CDD and four VAX DATATRIEVE access privileges to individual users or groups of users with CDD access control lists.

## 1.4 The Data Definition Language Utility (CDDL)

You can use the Data Definition Language Utility (CDDL) to define records and store them in the CDD. VAX programming languages and VAX Information Architecture products can then share these record definitions. Each language or product translates the generic definitions stored in the CDD into language- or product-specific definitions that it can use.

To enter a new data definition, or to modify an existing definition in the CDD, you first create a CDDL **source file**. This source file contains full data descriptions including field names, data types, and special attributes. Once the source file is complete, you use the CDDL compiler, either interactively or in batch mode, to insert the definitions into the CDD.

To use the CDDL compiler, you should define CDDL as a global symbol in your login command file:

```
$ CDDL:==$SYS$SYSTEM:CDDL
```

Chapter 2 of this manual describes CDDL source file statements and clauses. Chapter 3 describes CDDL compiler commands.

# CDDL Source File Description 2

You create a CDDL source file by using a text editor, like VAX EDT. This chapter describes the statements and clauses you use in a CDDL source file. When you have created the source file, you can store it in the CDD by using the CDDL compiler. Chapter 3 describes CDDL compiler commands.

CDDL source files consist of:

- DEFINE and END statements, which you use to define CDD records
- Field description statements, which you use to describe the general structure of records
- Field attribute clauses, which you use to describe characteristics of fields within CDD records

Use the following syntax guidelines to help create CDDL source files:

- Each line of CDDL source can be no longer than 255 characters.
- End each field description with a period.
- You can continue field attribute clauses from one line to the next without using a continuation character.
- Within each field attribute clause, the order of keywords is important. Refer to the format diagrams for proper keyword order of specific clauses.
- Within each field description, you can specify field attribute classes in any order you please.
- In elementary field descriptions, the DATATYPE clause is the only required field attribute clause.

The following sections contain descriptions of the statements and clauses you use to define records in the CDDL source file. These descriptions include:

- The syntax **format** you should use for each statement or clause
- **Syntax rules** to help you define records correctly
- **Usage notes** to show you how to use each statement or clause
- **Examples** to illustrate the use of each statement or clause

## 2.1 ALIGNED Field Attribute Clause

Aligns a field on a specified starting boundary relative to the start of the record.

### Format

```
                 ⎧ BIT       ⎫
                 ⎪ BYTE      ⎪
                 ⎪ WORD      ⎪
ALIGNED [ON]     ⎨ LONGWORD  ⎬   [BOUNDARY]
                 ⎪ QUADWORD  ⎪
                 ⎩ OCTAWORD  ⎭
```

### Usage Notes

- To satisfy hardware requirements, some languages and language processors have field alignment restrictions for data definitions. The ALIGNED clause enables you to control the starting boundaries of fields you define.
- Each field, except BIT fields, begins by default on the first byte following the last field. BIT fields begin on the bit immediately following the last field. You can modify this starting position with the ALIGNED clause.
- The ALIGNED clause aligns fields within a record relative to the start of the record, not relative to virtual memory locations. For example, if you specify LONGWORD alignment for a field, that field does not necessarily begin on a longword boundary in memory. Rather, the field begins some multiple of 32 bits beyond the start of the record. To correctly use the ALIGNED clause, you must know the memory alignment techniques of the language you use with the CDD.

**Note**

You should not use the ALIGNED clause in template records. When CDDL stores the template record, the position of an aligned field is fixed within the record and is not changed when the record is copied into another record definition. Therefore, the newly created field may not align properly in the new record definition.

### Example

In the following example, a LONGWORD field (QUANTITY) follows a BYTE field. The PRODUCT_NO field spans 64 bits, the DATE_ORDERED field spans 64 bits, and the STATUS_CODE field spans 8 bits. The ALIGNED clause causes the three bytes following STATUS_CODE to remain empty and aligns QUANTITY exactly 160 bits beyond the start of the record.

```
IN_STOCK STRUCTURE.
    PRODUCT_NO      DATATYPE IS TEXT
                    SIZE IS 8 CHARACTERS.
    DATE_ORDERED    DATATYPE IS DATE.
    STATUS_CODE     DATATYPE IS BYTE.
    QUANTITY        DATATYPE IS LONGWORD
                    ALIGNED ON LONGWORD.
    LOCATION        ARRAY 1:4
                    DATATYPE IS TEXT
                    SIZE IS 30 CHARACTERS.
    UNIT_PRICE      DATATYPE IS LONGWORD SCALE -2.
END IN_STOCK STRUCTURE.
```

## 2.2 ARRAY Field Attribute Clause

Declares multidimensional arrays or one-dimensional arrays. ARRAY is most useful when the subscript's lower bound is not equal to 1.

### Format

```
[ ROW_MAJOR    ]
[ COLUMN_MAJOR ] ARRAY [ n1 :] n2 [[ n3 :] n4 ] . . .
```

### Parameters

n1, n2, n3, n4

The upper and lower bounds of the subscripts.

### Syntax Rules

- Each dimension of the array is defined by a pair of *signed* integers.
- The first integer in each pair (n1, n3 . . .) specifies the subscript's lower bound. The default value for the lower bound is 1.
- The second integer in each pair (n2, n4 . . .) specifies the subscript's upper bound. The upper bound must be greater than or equal to the lower bound.

### Usage Notes

- With ARRAY, each subscript has an upper and lower bound defined by a pair of signed integers.
- In multidimensional arrays, ROW_MAJOR declares the rightmost subscript to be the fastest varying. COLUMN_MAJOR declares the leftmost subscript to be the fastest varying.
- If neither ROW_MAJOR nor COLUMN_MAJOR is specified, the default is ROW_MAJOR.

### Example

In the following example, the ARRAY clause declares 20 instances of SUPPLIER (from 0 to 19) where each instance is four 30-character strings.

```
SUPPLIER                        ARRAY 0:19 1:4
                                DATATYPE IS TEXT
                                SIZE IS 30 CHARACTERS.
```

## 2.3 BLANK WHEN ZERO Field Attribute Clause

Sets an entire field to blanks when you assign it a zero.

### Format

```
BLANK WHEN ZERO
```

### Usage Note

Only VAX COBOL supports this feature. All other processors ignore it.

### Example

In the following example, the field NEW is initially set to blanks when it is accessed by the VAX COBOL compiler.

```
ZIP_CODE STRUCTURE.
    NEW         DATATYPE IS UNSIGNED NUMERIC
                SIZE IS 4 DIGITS
                BLANK WHEN ZERO.
    OLD         DATATYPE IS UNSIGNED NUMERIC
                SIZE IS FIVE DIGITS.
END ZIP_CODE STRUCTURE.
```

## 2.4 COMPUTED BY DATATRIEVE Field Attribute Clause

Supplies expressions used by VAX DATATRIEVE to calculate the values of VIRTUAL fields.

### Format

$$\text{COMPUTED BY} \left\{ \begin{array}{l} \text{DTR} \\ \text{DATATRIEVE} \end{array} \right\} \text{AS quoted-string [quoted-string] . . .}$$

### Parameter

quoted-string

An expression used by VAX DATATRIEVE to calculate the field's value.

### Syntax Rule

The set of quoted strings must form a valid VAX DATATRIEVE virtual expression. The CDDL compiler does not check the virtual expression for correct syntax. It is your responsibility to provide a correct virtual expression.

### Usage Notes

- You must specify a VIRTUAL FIELD DATATYPE for a field using the COMPUTED BY DATATRIEVE clause.
- Only VAX DATATRIEVE can interpret VIRTUAL fields. Other processors either ignore their presence or refuse to process record descriptions containing them.
- If you are using a terminal of the VT200 family, you can use 8-bit characters in CDDL quoted strings.
- You cannot specify an INITIAL VALUE field attribute for a virtual field.
- You cannot specify a CONDITION NAME field attribute for a virtual field.
- You cannot use a virtual field as a tag variable in the VARIANTS OF field description statement.

### Example

In the following example, no storage has been allocated for TOTAL_PRICE. Instead, VAX DATATRIEVE uses the virtual expression "UNIT_PRICE QUANTITY" to calculate the value of TOTAL_PRICE at run time.

```
TOTAL_PRICE              DATATYPE IS VIRTUAL FIELD
                         COMPUTED BY DATATRIEVE AS
                             "UNIT_PRICE * QUANTITY".
```

## 2.5 CONDITION NAME Field Attribute Clause

Enables the VAX COBOL compiler to associate one or more condition names with specific values for a field.

### Format

```
CONDITION FOR COBOL [IS] condition-name
     [COBOL NAME [IS] quoted-string]

    {VALUE [IS]   }
    {VALUES [ARE] }

       { {    n1       } [THRU    {     n2      }]
       { {EXTERNAL e1  } [        {EXTERNAL e2  }]

                  [ {    n3      }  [  THRU  {     n4      } ] ] . . . }
                  [ {EXTERNAL e3 }  [        {EXTERNAL e4  } ] ]       }
```

### Parameters

condition-name

The condition.

quoted-string

A COBOL name for the condition.

n1, n2, n3, n4

Field values or ranges of field values associated with the condition name.

e1, e2, e3, e4

A quoted string containing a COBOL external name. See the *VAX COBOL Language Reference Manual* for information on the legal use of external names.

### Syntax Rules

- The condition name must be a string of up to 31 characters from the set A-Z, 0-9, _ , and $. The first character in the string must be a letter from A-Z, and the last character cannot be _ or $.

- Quoted strings or external names must be legal VAX COBOL names. However, the CDDL compiler does not check for correct syntax. It is your responsibility to provide a correct value expression.
- The values n1, n2, n3, and n4 can be fixed point numbers, floating point numbers, quoted strings, octal numbers, or hexadecimal numbers.
- The compiler ignores commas, but you can use them to make value specifications easier to read.

**Usage Notes**

- Only VAX COBOL supports this feature. Other language processors ignore the CONDITION NAME clause.
- Each CONDITION NAME clause defines one condition name. Each condition name can represent a discrete value, a range of values, or any combination of these.
- You can use the CONDITION NAME clause as many times as you wish within a field description.
- The values n1, n2, n3, and n4 must be legal values as defined by the data type declared for the field.
- The length of a literal you specify in a CONDITION NAME clause cannot exceed the length declared for the field.
- You can specify a fixed point number as the value of any field whose valid VAX COBOL data type is not DATE, TEXT, or UNSPECIFIED.
- You can specify a floating point number as the value of a field whose valid VAX COBOL data type is not DATE, TEXT, or UNSPECIFIED.
- You can specify a quoted string as the value only of a field whose valid VAX COBOL data type is DATE, TEXT, or UNSPECIFIED.
- If you are using a terminal of the VT200 family, you can use 8-bit characters in CDDL quoted strings.
- You can specify an octal number as the value of any valid VAX COBOL data type. In order to specify an octal number, place single quotation marks (') around the number and precede it with %O; for example,

```
VALUE IS %O'16'
```

- You can specify a hexadecimal number as the value of any valid VAX COBOL data type. In order to specify a hexadecimal number, place single quotation marks (') around it and precede it with %X; for example,

  ```
  VALUE IS %X'3E'
  ```

- Value n2 must be greater than or equal to value n1 in the field's collating sequence.
- Value n4 must be greater than or equal to value n3 in the field's collating sequence.

### Example

The following example defines three valid conditions and the values where the condition is invalid, according to the value of the field RECORD_IDENTIFIER.

```
RECORD_IDENTIFIER        DATATYPE IS TEXT
                         SIZE IS 1 CHARACTER
                         CONDITION FOR COBOL IS ON_HAND
                             COBOL NAME "ON-HAND"
                             VALUE IS "S"
                         CONDITION FOR COBOL BACKORDER
                             COBOL NAME "BACKORDER"
                             VALUE IS "B"
                         CONDITION FOR COBOL OUT_OF_STOCK
                             COBOL NAME "OUT-OF-STOCK"
                             VALUE IS "O"
                         CONDITION FOR COBOL IS INVALID
                             VALUES ARE "A", "C" THRU "N",
                             "P" THRU "R", "T" THRU "Z".
```

## 2.6 COPY Field Description Statement

Copies the description of an existing record (called a template record) into the description of a new field (the COPY field).

### Format

```
[ /* text */ ]
field-name        COPY [ FROM ] path-name [ ALIGNED clause ].
```

### Parameters

text

An explanation of the field.

field-name

The name of the COPY field.

path-name

The location of the template record definition within the CDD directory hierarchy.

### Syntax Rules

- You must assign a field name to a COPY field. This field name can be up to 31 characters from the set A-Z, 0-9, _, and $. The first character must be a letter from A-Z, and the last character cannot be _ or $. If you are using a terminal of the VT200 family, you can use alphabetic 8-bit characters in field names. Remember that other terminals cannot reproduce 8-bit characters.
- The path name can be a full or a relative path name, and it must be the path name of an existing CDD record description. You can specify an absolute version number with the path name.
- The COPY field description statement must terminate with a period.

## Usage Notes

- The copy operation takes place when the new record is compiled, not when it is copied into a program. When you modify a template record, you should recompile any record definitions that contain COPY field descriptions copying the modified template record.
- The COPY field description statement copies a complete record, a template record, into a single field.
- The COPY field description statement copies the description of the template record as the description of the COPY field. If the first subordinate field of the template record is a BIT field, the first subordinate field of the COPY field begins on the first bit immediately following the preceding field. Otherwise, the COPY field begins on the first byte immediately following the preceding field. You can modify this starting position with the ALIGNED clause (see Section 2.1).
- If you specify an absolute version number in the path name parameter, CDDL copies the version of the template record with that version number each time you compile the record description containing the COPY field. If you do not specify a version number, CDDL copies the highest version of the template record each time you compile the record description.
- In the COPY field, the field name you assign replaces the field name copied from the field description statement of the template record.
- When the CDDL compiler compiles a record definition containing a COPY field description, it automatically makes an entry documenting the copy operation in the history list of the template record, whether or not you use the /AUDIT qualifier.

## Example

CDD$TOP.CORPORATE.ADDRESS_RECORD defines the standard format for addresses. You can copy this field description into any new record requiring an address field.

```
ADDRESS                 COPY FROM
                        CDD$TOP.CORPORATE.ADDRESS_RECORD.
```

## 2.7 DATATYPE Field Attribute Clause

Declares the type and size of a field. Some valid CDD data types may not be supported by the languages or language processors you will be using with the CDD. It is your responsibility to make certain that the records you define are valid for the language processors you use.

### Format

```
DATATYPE [ IS ]
 ( DATE
 ( VIRTUAL [ FIELD ]
 ( BIT        [ FIELD ]                 [ SIZE [ IS ] ] n1
 ( UNSPECIFIED                          [ SIZE [ IS ] ] n1 [ BYTE [ S ]]
 (   { TEXT           }
 (   { VARYING STRING }                 [ SIZE [ IS ] ] n1 [ CHARACTER [ S ] ]
 ( POINTER [ TO path-name ]
 (
 (   { D_FLOATING               }       [          ]       [         ]
 (   { D_FLOATING COMPLEX       }       [          ]       [         ]
 (   { F_FLOATING               }       [          ]       [         ]
 (   { F_FLOATING COMPLEX       }       [ SCALE n1 ]       [ BASE n2 ]
 (   { G_FLOATING               }       [          ]       [         ]
 (   { G_FLOATING COMPLEX       }       [          ]       [         ]
 (   { H_FLOATING               }       [          ]       [         ]
 (   { H_FLOATING COMPLEX       }       [          ]       [         ]
 (   { [ UN ] SIGNED BYTE       }       [[ SIZE [ IS ] ] n1 [ DIGIT [ S ]   ]
 (   { [ UN ] SIGNED WORD       }       [          [ n2 FRACTION [ S ] ] ]]
 (   { [ UN ] SIGNED LONGWORD   }       [                                   ]
 (   { [ UN ] SIGNED QUADWORD   }       [SCALE n3                           ]
 (   { [ UN ] SIGNED OCTAWORD   }       [BASE n4                            ]
 (
 (   { PACKED DECIMAL            }       [ SIZE [ IS ] ] n1 [ DIGIT [ S ]
 (   { ZONED NUMERIC             }              [ n2 FRACTION [ S ] ] ]
 (   { UNSIGNED NUMERIC          }
 (   { LEFT SEPARATE NUMERIC     }      [SCALE n3]
 (   { LEFT OVERPUNCHED NUMERIC  }      [BASE n4 ]
 (   { RIGHT SEPARATE NUMERIC    }
 (   { RIGHT OVERPUNCHED NUMERIC }
```

### Syntax Rules

- DATE specifies that the field is a 64-bit VAX standard absolute date data type.
- VIRTUAL FIELD specifies that the field is a VAX DATATRIEVE virtual field. No space is allocated for virtual fields in a record. The COMPUTED BY DATATRIEVE clause determines the value of a virtual field at run time. A STRUCTURE field cannot contain the VIRTUAL FIELD datatype. See Section 2.4.
- BIT specifies that the field is a bit string. Indicate the number of bits in the field with an unsigned integer (n1).
- UNSPECIFIED declares that the field is a sequence of 8-bit unsigned bytes. Indicate the number of bytes in the field with an unsigned integer (n1).
- TEXT specifies that the field is a sequence of 8-bit ASCII bytes. Indicate the number of characters in the field with an unsigned integer (n1). CDDL accepts CHARACTER as a synonym for TEXT.
- VARYING STRING specifies that the field is a PL/I or PASCAL varying string. Indicate the number of characters in the field with an unsigned integer (n1). CDDL accepts VARYING TEXT as a synonym for VARYING STRING.
- POINTER specifies that the field contains the address of another field or record definition. In PL/I, for example, POINTER fields are used to access based variables and buffers allocated by the system. Although PL/I does not associate POINTER fields with a specified record structure, other languages do; the optional [TO path-name] lets you connect a POINTER to a structure.
- Floating point data types:
  - You can specify a SCALE as an implied exponent. The signed integer (n1) must be in the range -128 to 127. The SCALE specification indicates the number of places to shift the decimal point when the field is evaluated. Negative n1 indicates a shift of n1 places to the left, and positive n1 indicates a shift of n1 places to the right.
  - You can also specify the radix, or BASE, with an unsigned integer (n2). The BASE indicates the number system to be used when the field is evaluated. The default BASE is 10.
  - D_FLOATING specifies that the field is a 64-bit floating point number with precision to approximately 16 decimal digits.

- D_FLOATING COMPLEX specifies that the field consists of two 64-bit floating complex numbers, one for the real component and one for the imaginary. CDDL accepts D_FLOATING_COMPLEX as a synonym for D_FLOATING COMPLEX.
- F_FLOATING specifies that the field is a 32-bit floating point number with precision to approximately seven decimal digits.
- F_FLOATING COMPLEX specifies that the field consists of two 32-bit floating complex numbers, one for the real component and one for the imaginary. CDDL accepts FLOATING_COMPLEX, FLOATING COMPLEX, and F_FLOATING_COMPLEX as synonyms for F_FLOATING COMPLEX.
- G_FLOATING specifies that the field is an extended range 64-bit floating point number with precision to approximately 15 decimal digits.
- G_FLOATING COMPLEX specifies that the field consists of two extended range 64-bit floating complex numbers, one for the real component and one for the imaginary. CDDL accepts G_FLOATING_COMPLEX as a synonym for G_FLOATING COMPLEX.
- H_FLOATING specifies that the field is an extended range 128-bit floating point number with precision to approximately 33 decimal digits.
- H_FLOATING COMPLEX specifies that the field consists of two extended range 128-bit floating complex numbers, one for the real component and one for the imaginary. CDDL accepts H_FLOATING_COMPLEX as a synonym for H_FLOATING COMPLEX.

- Fixed point data types:
  - You can declare the total number of DIGITS (n1) and the number of those digits that are FRACTIONS (n2). The number of digits must be greater than 0 and less than 32. The number of fractions must not be greater than the number of digits. The default number of fractions is 0.
  - You can specify a SCALE as an implied exponent. The signed integer (n1) must be in the range -128 to 127. The SCALE specification indicates the number of places to shift the decimal point when the field is evaluated. Negative n1 indicates a shift of n1 places to the left, and positive n1 indicates a shift of n1 places to the right.

- You can also specify the radix, or BASE, with an unsigned integer (n2). The BASE indicates the number system to be used when the field is evaluated. The default BASE is 10.
- BYTE specifies that the field is an 8-bit byte. The BYTE can be SIGNED or UNSIGNED. If there is no sign specification, UNSIGNED is the default.
- WORD specifies that the field is a 16-bit word. The field can be SIGNED or UNSIGNED. If no sign is specified, UNSIGNED is the default.
- LONGWORD specifies that the field is a 32-bit longword. The LONGWORD can be SIGNED or UNSIGNED. If no sign is specified, UNSIGNED is the default.
- QUADWORD specifies that the field is a 64-bit quadword field. The field can be SIGNED or UNSIGNED. If no sign is specified, UNSIGNED is the default.
- OCTAWORD specifies that the field is a 128-bit octaword field. The field can be SIGNED or UNSIGNED. If no sign is specified, UNSIGNED is the default.

- Decimal string data types:
  - You must declare the total number of DIGITS (n1) in the field. You can also declare which of those digits are FRACTIONS (n2). The number of digits must be greater than 0 and less than 32. The number of fractions must not be greater than the number of digits. The default number of fractions is 0.
  - You can specify a SCALE as an implied exponent. The signed integer (n1) must be in the range -128 to 127. The SCALE specification indicates the number of places to shift the decimal point when the field is evaluated. Negative n1 indicates a shift of n1 places to the left, and positive n1 indicates a shift of n1 places to the right.
  - You can also specify the radix, or BASE, with an unsigned integer (n2). The BASE indicates the number system to be used when the field is evaluated. The default BASE is 10.
  - PACKED DECIMAL specifies that the field is a packed decimal numeric field. CDDL accepts PACKED NUMERIC as a synonym for PACKED DECIMAL.
  - UNSIGNED NUMERIC specifies that the field is an unsigned decimal string. You must use the UNSIGNED keyword.

- ZONED NUMERIC specifies that the field is a VAX ZONED NUMERIC data type. CDDL accepts SIGNED NUMERIC as a synonym for ZONED NUMERIC.
- LEFT SEPARATE NUMERIC specifies that the field is a left separate signed numeric decimal string. CDDL accepts [SIGNED] NUMERIC LEFT SEPARATE as a synonym for LEFT SEPARATE NUMERIC.
- LEFT OVERPUNCHED NUMERIC specifies that the field is a left overpunched signed numeric decimal string. CDDL accepts [SIGNED] NUMERIC LEFT OVERPUNCHED as a synonym for LEFT OVERPUNCHED NUMERIC.
- RIGHT SEPARATE NUMERIC specifies that the field is a right separate signed numeric decimal string. CDDL accepts [SIGNED] NUMERIC RIGHT SEPARATE as a synonym for RIGHT SEPARATE NUMERIC.
- RIGHT OVERPUNCHED NUMERIC specifies that the field is a right overpunched signed numeric decimal string. CDDL accepts [SIGNED] NUMERIC RIGHT OVERPUNCHED as a synonym for RIGHT OVERPUNCHED NUMERIC.

- CDDL accepts TYPE as a synonym for DATATYPE, but the compiler issues a warning when you use TYPE.

## Usage Notes

- The SCALE specification and the FRACTIONS specification have a similar function. They both indicate how many digits in a field are to the right of the decimal point. There are, however, two important differences:
  - The number of FRACTIONS you can specify is limited to the number of DIGITS you declare for the data type. With SCALE, there is no such limitation.
  - With SCALE, you can shift the decimal point to the right and to the left, but with FRACTIONS, you can shift the decimal point only to the left.
- Use a positive integer in the SCALE specification to move the decimal point to the right.
- Use a negative integer in the SCALE specification to move the decimal point to the left.

### Example

The STRUCTURE field BACK_ORDER contains examples of valid CDDL data declarations.

```
BACK_ORDER STRUCTURE.
    PRODUCT_NO      DATATYPE IS TEXT
                    SIZE IS 8 CHARACTERS.
    DATE_ORDERED    DATATYPE IS DATE.
    STATUS_CODE     DATATYPE IS BYTE.
    QUANTITY        DATATYPE IS LONGWORD
                    ALIGNED ON LONGWORD.
    SUPPLIER        ARRAY 1:4
                    DATATYPE IS TEXT
                    SIZE IS 30 CHARACTERS.
    UNIT_PRICE      DATATYPE IS LONGWORD SCALE -2.
END BACK_ORDER STRUCTURE.
```

## 2.8 DEFAULT_VALUE Field Attribute Clause

Sets a default value for a field accessed by VAX DATATRIEVE.

### Format

```
                      { DTR        }        { fixed-point-number }
DEFAULT_VALUE FOR     { DATATRIEVE } [ IS ] { quoted-string      }
```

### Parameters

fixed-point-number
quoted-string

A VAX DATATRIEVE expression that is the default value.

### Syntax Rules

- The quoted string or fixed point number must be a valid VAX DATATRIEVE expression. However, the CDDL compiler does not check for correct syntax. It is your responsibility to provide a correct default value expression.
- If you are using a terminal of the VT200 family, you can use 8-bit characters in CDDL quoted strings.

### Usage Notes

- The DEFAULT_VALUE clause, used with VAX DATATRIEVE, serves much the same purpose as the INITIAL_VALUE clause. The main difference between the two is that the INITIAL_VALUE clause assigns a value to a field when the field is first allocated, usually at compile time, but the DEFAULT_VALUE clause assigns a value to a field each time the record is stored without an explicit value assigned to it.
- Only VAX DATATRIEVE supports the DEFAULT_VALUE clause. Other language processors ignore it.

### Example

The following example assigns a DEFAULT_VALUE of 0 to the field TOTAL_PRICE.

```
TOTAL_PRICE                DATATYPE IS VIRTUAL FIELD
                           COMPUTED BY DATATRIEVE AS
                               "UNIT_PRICE * QUANTITY"
                           DEFAULT_VALUE FOR DATATRIEVE IS 0.
```

## 2.9 DEFINE and END Statements

The DEFINE statement begins each record definition in the source file. The END statement terminates each record definition.

### Format

```
DEFINE RECORD path-name

     [ DESCRIPTION [ IS ] /* text /* ].

     field-description-statement

END { path-name  } [ RECORD ].
    { given-name }
```

### Parameters

path-name

The location of the new record definition within the directory hierarchy.

text

Explanatory text describing the record definition.

field-description-statement

The field description for the entire record. This record field can be an elementary, STRUCTURE, COPY, or VARIANTS field.

given-name

The new record definition's name.

### Syntax Rules

- You must terminate the DEFINE statement and the END statement with periods.
- The path name can be a full or a relative path specification. The last given name in the path name is the name you assign to the new record definition. You can specify an absolute version number with the path name.

- The given name of the new record definition is a string of up to 31 characters from the set A-Z, 0-9, _, and $. The first character must be a letter from A-Z, and the last character must not be _ or $. If you are using a terminal of the VT200 family, you can use 8-bit alphabetic characters in path names. Remember that other terminals cannot reproduce 8-bit characters.
- Use the optional DESCRIPTION clause to include text in the CDD to document the record definition. You must use the keyword DESCRIPTION and the text delimiters /* and */ to insert one or more lines of text describing the record into the DEFINE statement. See Section 2.10.
- The field description statement defines the whole record. You can use an elementary, STRUCTURE, COPY, or VARIANTS field description statement, but you can use only one field description statement to define a record. STRUCTURE, COPY, and VARIANT field description statements are themselves subdivided and defined by subordinate field description statements. See Sections 2.6, 2.24, and 2.26.
- If you specify a path name in the END clause, it must match the path name in the corresponding DEFINE clause.
- If you specify a given name in the END clause, it must be the given name of the new data definition and match the final given name in the path specification of the corresponding DEFINE clause.

### Usage Notes

- If the path name contains dictionary directories that do not exist, the CDD automatically creates them.
- You can include passwords in the DEFINE statement path name, but this is not recommended because passwords included in the source can be displayed with the DMU LIST/ITEM=SOURCE command. Instead, use the /PATH qualifier with the compile command if passwords are required in the path name. See Chapter 3.
- The only way to assign a password to a record definition is to use the DMU SET PROTECTION or DMU SET PROTECTION/EDIT command after you compile the source file. You cannot assign a password to a record definition by including it in the DEFINE statement.

### Example

The following record description defines the dictionary object SALARY_RECORD in the sample CDD hierarchy. The field description statement for the record is SALARY STRUCTURE.

```
DEFINE RECORD CDD$TOP.PERSONNEL.SERVICE.SALARY_RECORD.
    SALARY STRUCTURE.
        EMPLOYEE_ID          DATATYPE IS UNSIGNED NUMERIC
                             SIZE IS 9 DIGITS.
        PAY STRUCTURE.
            JOB_CLASS        DATATYPE IS UNSIGNED NUMERIC
                             SIZE IS 3 DIGITS.
            INCR_LEVEL       DATATYPE IS UNSIGNED NUMERIC
                             SIZE IS 1 DIGIT.
            WEEKLY_SALARY    DATATYPE IS UNSIGNED NUMERIC
                             SIZE IS 6 DIGITS 2 FRACTIONS.
        END PAY STRUCTURE.
    END SALARY STRUCTURE.
END SALARY_RECORD RECORD.
```

## 2.10 DESCRIPTION Clause

Inserts text documenting record and field definitions into the CDD.

### Format

```
Within the DEFINE statement:

    DESCRIPTION [IS] /* text */

Preceding field description statements:

    /* text */
```

### Syntax Rules

- You must use the keyword DESCRIPTION and the text delimiters /* and */ to insert one or more lines of text describing the record into the DEFINE statement.
- You can also include text to describe individual fields within the record definition. Use the text delimiters /* and */ without the keyword DESCRIPTION, and place the text immediately before the field description statement of the field you want to document.
- If you are using a terminal of the VT200 family, you can use 8-bit characters in CDDL descriptions.

### Usage Notes

- This clause provides a convenient way to associate text with a record description. Proper use of this capability provides a tool for documenting the sources of data and for describing the field structure of a record. DESCRIPTION text can also aid in establishing record definition conventions and standards.
- You can also include comments in a CDDL source file by using the exclamation point (!) as a comment delimiter. Using the DESCRIPTION clause is preferable to including comment lines in the source file because DESCRIPTION text is stored directly in the CDD. CDDL/COPY_LIST copies DESCRIPTION clauses from template records, but does not copy comments preceded by an exclamation point. Also, you can display record-level DESCRIPTION text with DMU's

list/ITEM=DESCRIPTION command. DMU's EXTRACT/RECORD command extracts record- and field-level DESCRIPTION text but does not extract comments preceded by an exclamation point.

### Example

The following example contains DESCRIPTION text of the record definition as a whole, and of the elementary field ID.

```
DEFINE RECORD CDD$TOP.CORPORATE.EMPLOYEE_LIST
    DESCRIPTION /* This record contains the employee master list,
                   and it is the source from which employee fields
                   in other record descriptions are copied. */.
    EMPLOYEE STRUCTURE.
        /* An employee's ID number is his or her social
        security number. */
        ID                        DATATYPE IS UNSIGNED NUMERIC
                                  SIZE IS 9 DIGITS.
        NAME STRUCTURE.
            LAST_NAME             DATATYPE IS TEXT
                                  SIZE IS 15 CHARACTERS.
            FIRST_NAME            DATATYPE IS TEXT
                                  SIZE IS 10 CHARACTERS.
            MIDDLE_INITIAL        DATATYPE IS TEXT
                                  SIZE IS 1 CHARACTER.
        END NAME STRUCTURE.
        ADDRESS                   COPY FROM
                                  CDD$TOP.CORPORATE.ADDRESS_RECORD.
        DEPT_CODE                 DATATYPE IS UNSIGNED NUMERIC
                                  SIZE IS 3 DIGITS.
    END EMPLOYEE STRUCTURE.
END EMPLOYEE_LIST RECORD.
```

## 2.11 EDIT_CODE Field Attribute Clause

Provides a code that VAX RPG II follows when printing a field's value.

**Format**

```
EDIT_CODE FOR RPG [ IS ] quoted-string
```

**Parameter**

quoted-string

A VAX RPG II edit code or modifier.

**Syntax Rules**

- The quoted string must be a valid VAX RPG II edit code or modifier. However, the CDDL compiler does not check the quoted string for correct syntax.
- If you are using a terminal of the VT200 family, you can use 8-bit characters in CDDL quoted strings.

**Usage Note**

Only VAX RPG II supports the EDIT_CODE clause. Other language processors ignore it.

**Example**

In the following example, the edit code for the field ORDNUM is 3. Because the attribute's value is 3, a VAX RPG II program will suppress zeros when printing ORDNUM.

```
TRANSACTION STRUCTURE.
    ORDNUM          DATATYPE IS NUMERIC RIGHT OVERPUNCHED
                    SIZE IS 8 DIGITS
                    EDIT_CODE FOR RPG IS "3".
    AMOUNT          DATATYPE IS NUMERIC RIGHT OVERPUNCHED
                    SIZE IS 8 DIGITS 2 FRACTIONS
                    EDIT_WORD FOR RPG IS "$0   ,   .  CR".
END TRANSACTION STRUCTURE.
```

## 2.12 EDIT_STRING Field Attribute Clause

Provides a format that VAX DATATRIEVE follows when displaying a field's value.

### Format

$$\text{EDIT\_STRING FOR} \left\{ \begin{array}{l} \text{DTR} \\ \text{DATATRIEVE} \end{array} \right\} \text{[ IS ] quoted-string.}$$

### Parameter

quoted-string

A VAX DATATRIEVE edit string.

### Syntax Rules

- The edit string must be a valid VAX DATATRIEVE expression. However, the CDDL compiler does not check the quoted string for correct syntax.
- If you are using a terminal of the VT200 family, you can use 8-bit characters in CDDL quoted strings.

### Usage Note

Only VAX DATATRIEVE supports the EDIT_STRING clause. Other language processors ignore it.

### Example

In the following example, VAX DATATRIEVE displays the TRANS_DATE field as a series of three 2-digit numbers in the format month/day/year.

```
TRANSACTION STRUCTURE     OCCURS 1 TO 99 TIMES
                          DEPENDING ON TRANSACTION_COUNT.
    TRANS_DATE            DATATYPE IS DATE
                          EDIT_STRING FOR DATATRIEVE
                          IS "MM/DD/YY".
    ORDER_NUMBER          DATATYPE IS UNSIGNED NUMERIC
                          SIZE IS 10 DIGITS.
    AMOUNT                DATATYPE IS UNSIGNED NUMERIC
                          SIZE IS 8 DIGITS 2 FRACTIONS
                          INITIAL VALUE IS 0.
END TRANSACTION STRUCTURE.
```

## 2.13 EDIT_WORD Field Attribute Clause

Provides a format that VAX RPG II follows when printing a field's value.

### Format

```
EDIT_WORD FOR RPG [ IS ] quoted-string
```

### Parameter

quoted-string

A VAX RPG II edit word.

### Syntax Rules

- The quoted string must be a valid VAX RPG II edit word. However, the CDDL compiler does not check the quoted string for correct syntax.
- If you are using a terminal of the VT200 family, you can use 8-bit characters in CDDL quoted strings.

### Usage Note

Only VAX RPG II supports the EDIT_WORD clause. Other language processors ignore it.

### Example

In the following example, the EDIT_WORD attribute in the AMOUNT field specifies a monetary format:

```
TRANSACTION STRUCTURE.
    ORDNUM          DATATYPE IS NUMERIC RIGHT OVERPUNCHED
                    SIZE IS 8 DIGITS
                    EDIT_CODE FOR RPG IS "3".
    AMOUNT          DATATYPE IS NUMERIC RIGHT OVERPUNCHED
                    SIZE IS 8 DIGITS 2 FRACTIONS
                    EDIT_WORD FOR RPG IS "$0    ,    .  CR".
END TRANSACTION STRUCTURE.
```

## 2.14 Elementary Field Description Statement

Defines the characteristics of a field that is not subdivided into other fields.

### Format

```
[ /* text */ ]

{    *       }
{ field-name } field-attribute [ field-attribute ] . . . .
```

### Parameters

text

Explanatory text describing the field.

field-name

The field's name.

field-attribute

The field's characteristics, including data type.

### Syntax Rules

- The field name you assign can be up to 31 characters from the set A-Z, 0-9, _, and $. The first character must be a letter from A-Z, and the last character cannot be _ or $. If you are using a terminal of the VT200 family, you can use 8-bit alphabetic characters in field names. Remember that other terminals cannot reproduce 8-bit characters.
- If you use an asterisk (*) instead of a field name, you create an unnamed field.
- You must include the DATATYPE clause among the selected field attributes.
- You must terminate the elementary field description statement with a period.

### Usage Notes

- Unnamed fields are similar to FILLER fields in COBOL. You can use them to format print records or to reserve space in a record for future additions.

## Elementary

- Each field, except BIT fields, begins on the first byte following the preceding field. BIT fields begin on the bit immediately following the preceding field. You can modify this starting position with the ALIGNED clause. See Section 2.1.

### Example

NAME and ACCOUNT_NUMBER in the example below are elementary fields.

```
CUSTOMER STRUCTURE.
    NAME                    DATATYPE IS TEXT
                            SIZE IS 30 CHARACTERS.
    ACCOUNT_NUMBER          DATATYPE IS UNSIGNED NUMERIC
                            SIZE IS 7 CHARACTERS.
END CUSTOMER STRUCTURE.
```

## 2.15 INITIAL_VALUE Field Attribute Clause

Declares a field's value when CDDL first allocates the field.

### Format

```
                       ( complex-number          )
                       | fixed-point-number      |
                       | floating-point-number   |
INITIAL_VALUE [IS]    <  quoted-string            >
                       | hex-number              |
                       | octal-number            |
                       ( EXTERNAL quoted-string  )
```

### Usage Notes

- The value of the literal must fit into the space allocated for the field.
- You can specify a complex number as the INITIAL_VALUE only of a field whose data type is F_FLOATING COMPLEX, D_FLOATING COMPLEX, G_FLOATING COMPLEX, or H_FLOATING COMPLEX.
- You can specify a fixed point number as the INITIAL_VALUE of any field whose data type is not DATE, TEXT, UNSPECIFIED, VARYING STRING, or VIRTUAL FIELD.
- You can specify a floating point number as the INITIAL_VALUE of a field whose data type is not DATE, TEXT, UNSPECIFIED, VARYING STRING, or VIRTUAL FIELD.
- You can specify a quoted string as the INITIAL_VALUE only of a field whose data type is DATE, TEXT, UNSPECIFIED, or VARYING STRING.
- The quoted-string in the EXTERNAL subclause contains a legal VAX COBOL external name. See the *VAX COBOL Language Reference Manual* for legal EXTERNAL datatypes.
- If you are using a terminal of the VT200 family, you can use 8-bit characters in CDDL quoted strings.

- You can specify a hexadecimal number as the INITIAL_VALUE of a field with any data type except VIRTUAL FIELD. In order to specify a hexadecimal number, place single quotation marks (') around the number and precede it with %X; for example,

  ```
  INITIAL_VALUE IS %X'3E'
  ```

- You can specify an octal number as the INITIAL_VALUE of a field with any data type except VIRTUAL FIELD. In order to specify an octal number, place single quotation marks (') around the number and precede it with %O; for example,

  ```
  INITIAL_VALUE IS %O'16'
  ```

- A VIRTUAL FIELD cannot have an INITIAL_VALUE clause.
- Language processors that do not support the INITIAL_VALUE clause ignore it.
- If the base is not ten and scale is not zero, you can specify initial values only in hexadecimal or octal. Furthermore, before you translate the initial value to hexadecimal or octal, you should multiply it by the base raised to the value of the scale. For example, to specify an initial value of 1 for a field with base 2 and scale 5, first multiply the value by 2 raised to the fifth, yielding 32. Then convert 32 to its hexadecimal or octal equivalent, and store that value.

### Example

The following data declaration gives the field AMOUNT an INITIAL_VALUE of 0.

```
TRANSACTION STRUCTURE      OCCURS 1 TO 99 TIMES
                           DEPENDING ON TRANSACTION_COUNT.
    TRANS_DATE             DATATYPE IS DATE.
    ORDER_NUMBER           DATATYPE IS UNSIGNED NUMERIC
                           SIZE IS 10 DIGITS.
    AMOUNT                 DATATYPE IS UNSIGNED NUMERIC
                           SIZE IS 8 DIGITS 2 FRACTIONS
                           INITIAL_VALUE IS 0.
END TRANSACTION STRUCTURE.
```

## 2.16 JUSTIFIED RIGHT Field Attribute Clause

Truncates or fills a TEXT or UNSPECIFIED field from the left instead of from the right.

### Format

```
JUSTIFIED RIGHT
```

### Usage Notes

- Only VAX COBOL supports the JUSTIFIED RIGHT clause. Other language processors ignore it.
- Use this clause only on fields whose data type is TEXT or UNSPECIFIED.

## 2.17 MISSING_VALUE Field Attribute Clause

Specifies a value to indicate that a field has never been assigned a meaningful value.

### Format

```
                      { DTR         }        { fixed-point-number }
MISSING_VALUE FOR     { DATATRIEVE  } [ IS ] { quoted-string      }
```

### Parameters

fixed-point-number
quoted-string

The VAX DATATRIEVE missing value.

### Syntax Rules

- The quoted string or fixed point number must be a valid VAX DATATRIEVE expression for the field. The CDDL compiler does not check for correct syntax.
- If you are using a terminal of the VT200 family, you can use 8-bit characters in CDDL quoted strings.

### Usage Notes

- Only VAX DATATRIEVE supports the MISSING_VALUE clause. Other language processors ignore it.
- VAX DATATRIEVE treats a field containing a MISSING_VALUE as a special case; for example, VAX DATATRIEVE ignores fields containing a MISSING_VALUE when it performs statistical operations.

### Example

The following example assigns a missing value of 0 to the field PRICE. A PRICE of 0 indicates to VAX DATATRIEVE that the value for PRICE is missing; VAX DATATRIEVE ignores records with PRICE equal to 0 when it performs operations involving the PRICE field.

```
PRICE                         DATATYPE IS UNSIGNED NUMERIC
                              SIZE IS 8 DIGITS 2 FRACTIONS
                              MISSING_VALUE FOR DATATRIEVE IS 0.
```

## 2.18 NAME Field Attribute Clause

Declares a facility-specific name for a field. The specified language or language processor then recognizes only this name when you refer to the field.

### Format

```
           { BASIC }
NAME FOR   { COBOL }  [IS] quoted-string
           { PL/I  }
           { RPG   }
```

### Parameter

quoted-string

The facility-specific field name.

### Syntax Rules

- The quoted string must be a legal name for the specified language or language processor. The CDDL does not check the quoted string for validity or correct syntax.
- If you are using a terminal of the VT200 family, you can use 8-bit characters in CDDL quoted strings.

### Usage Notes

- You can use this clause only if you have specified a field name in the field declaration. You cannot specify a facility-specific name for unnamed fields.
- Once you have assigned a facility-specific name to a field, the facility no longer recognizes the field's original name.
- Be careful when you use the NAME clause because it enables you to assign completely different names to the same field. If you use it improperly, the NAME clause can become a source of confusion.
- Avoid assigning dissimilar names to the same field. The NAME clause is designed only to allow you to make field names seem native to applications languages.

### Example

The following example provides a VAX COBOL name and a VAX RPG II name for the ORDER_NUMBER field. Because of the NAME clause, VAX COBOL recognizes the field only by the name ORDER-NUMBER, and VAX RPG II recognizes the field only by the name ORDER#.

```
ORDER_NUMBER        DATATYPE IS UNSIGNED NUMERIC
                    SIZE IS 10 DIGITS
                    NAME FOR COBOL IS "ORDER-NUMBER".
                    NAME FOR RPG IS "ORDER#".
```

## 2.19 OCCURS Field Attribute Clause

Declares fixed-length, one-dimensional arrays.

### Format

```
OCCURS n1 [ TIME[S] ]
        [ INDEXED FOR COBOL BY quoted-string [,...] ]
```

### Parameters

n1

The number of occurrences of the array.

quoted-string

A VAX COBOL index name.

### Syntax Rules

- An unsigned integer (n1) declares the number of occurrences in one-dimensional, fixed-length arrays. This integer is the upper bound of the array.
- The number of occurrences must be greater than zero.

### Usage Notes

- The unsigned integer (n1) is the array's upper bound; the lower bound of an array declared with OCCURS is always 1. If you need to specify an array with a lower bound other than 1, use the ARRAY clause.
- Only VAX COBOL supports the INDEXED FOR COBOL BY optional field attribute clause. Other processors ignore it.
- You cannot use the INDEXED FOR COBOL BY optional field attribute clause with Version 3.0 of VAX COBOL or any earlier version. VAX COBOL supports INDEXED FOR COBOL BY in Version 3.1 and later.
- If you use a name as a COBOL index name, you cannot use that name as a field name or COBOL-specific name elsewhere in the record description.

## Example

In the following example, the OCCURS clause is used twice to declare 20 instances of SUPPLIER where each instance is four 30-character strings. Note that OCCURS clauses can be nested.

```
SUPPLIER STRUCTURE                OCCURS 20 TIMES.
    SUPPLIER                      OCCURS 4 TIMES
                                  DATATYPE IS TEXT
                                  SIZE IS 30 CHARACTERS.
END SUPPLIER STRUCTURE.
```

## 2.20 OCCURS . . . DEPENDING Field Attribute Clause

Declares a variable-length, one-dimensional array.

### Format

```
OCCURS n1 TO n2 [ TIME[S] ] DEPENDING [ ON ] field-name
        [ INDEXED FOR COBOL BY quoted-string [,...] ]
```

### Parameters

n1, n2

The range for the number of occurrences.

field-name

The tag variable field, whose value determines the actual number of occurrences.

quoted-string

A VAX COBOL index name.

### Syntax Rules

- Two unsigned integers (n1 and n2) specify a range for the number of occurrences; n1 specifies the minimum number of occurrences and must be greater than or equal to zero; n2 specifies the maximum number of occurrences and must be greater than or equal to n1.
- The actual number of occurrences varies according to the value of the named field.
- The field named in the DEPENDING clause must be an elementary field fixed in the record and not part of an array. Its field description must precede the array field description, and its value must never be less than n1 nor greater than n2. You must name the field.

- You must fully qualify the field name if it is not unique within the record. A fully qualified field name consists of an elementary field name preceded by the field names of as many of its direct ancestors as you need to specify the elementary field uniquely. You cannot omit the names of any of the ancestor fields within the fully qualified name, but once the elementary field name is identified uniquely, you can omit any remaining ancestors' field names. You must separate each element of a fully qualified field name from the next with a period.

### Usage Notes

- The unsigned integers n1 and n2 declare the range for the array's upper bound; the lower bound of an array declared with OCCURS . . . DEPENDING is always 1. If you need to specify an array with a lower bound other than 1, use the ARRAY clause.
- If the tag variable's name is not unique within the record, none of the ancestors in its fully qualified name can be unnamed fields.
- Only VAX COBOL supports the INDEXED FOR COBOL BY field attribute clause. Other processors ignore it.
- You cannot use the INDEXED FOR COBOL BY optional field attribute clause with Version 3.0 of VAX COBOL or any earlier version. VAX COBOL supports INDEXED FOR COBOL BY in Version 3.1 and later.
- If you use a name as a COBOL index name, you cannot use that name as a field name or COBOL-specific name elsewhere in the record description.

### Example

In the following example, a variable length array defines individual transactions within the STRUCTURE field SALES.

```
SALES STRUCTURE.
    TRANSACTION_COUNT                    DATATYPE IS UNSIGNED WORD
                                         VALID FOR DTR IF
                                             "TRANSACTION_COUNT > 0".
    TRANSACTION STRUCTURE                OCCURS 1 TO 99 TIMES
                                             DEPENDING ON
                                             TRANSACTION_COUNT.
        TRANS_DATE                       DATATYPE IS DATE.
        ORDER_NUMBER                     DATATYPE IS UNSIGNED NUMERIC
                                         SIZE IS 10 DIGITS.
        AMOUNT                           DATATYPE IS UNSIGNED NUMERIC
                                         SIZE IS 8 DIGITS 2 FRACTIONS
                                         PICTURE FOR COBOL IS "9(6)V99".
    END TRANSACTION STRUCTURE.
END SALES STRUCTURE.
```

The fully qualified field name of TRANSACTION_COUNT is SALES.TRANSACTION_COUNT.

## 2.21 PICTURE Field Attribute Clause

Declares a field's picture string for a specified language or language processor.

### Format

```
                 ⎧COBOL     ⎫
PICTURE FOR      ⎨DTR       ⎬    [IS] quoted-string
                 ⎨DATATRIEVE⎬
                 ⎩PLI       ⎭
```

### Parameter

quoted-string

The picture string for the specified language or language processor.

### Syntax Rules

- The quoted string must be a valid picture string for the specified language or language processor. The CDDL compiler does not check the quoted string for validity or correct syntax.
- If you are using a terminal of the VT200 family, you can use 8-bit characters in CDDL quoted strings.
- The data type implicit in the quoted string must be consistent with the data type you select with the DATATYPE clause.

### Usage Notes

- Each language that requires a picture string will construct a default picture string if you do not provide one. The default picture string provides a concise, efficient description of the field. You should use the default picture string whenever possible and avoid facility-specific PICTURE clauses.
- The CDDL compiler does not check picture strings for conformity with the DATATYPE clause. In most cases, therefore, the default picture string is probably the best option.
- You can use the EDIT_STRING field attribute clause to provide an edited picture string for VAX DATATRIEVE.

### Example

The following example contains a picture string for COBOL.

```
AMOUNT                DATATYPE UNSIGNED NUMERIC 8 DIGITS 2 FRACTIONS
                      PICTURE FOR COBOL IS "9(6)V99".
```

## 2.22 QUERY_HEADER Field Attribute Clause

Provides a label that VAX DATATRIEVE uses as a column heading for the field in printouts and reports.

### Format

$$\text{QUERY\_HEADER FOR} \left\{ \begin{array}{l} \text{DTR} \\ \text{DATATRIEVE} \end{array} \right\} \text{[ IS ] quoted-string [quoted-string] . . .}$$

### Parameter

quoted-string

The VAX DATATRIEVE query header.

### Syntax Rules

- The quoted string must be a valid VAX DATATRIEVE query header. The CDDL compiler does not check the quoted string for correct syntax.
- Quoted strings must be separated from each other by a space, tab, or comma.

### Usage Notes

- Only VAX DATATRIEVE supports the QUERY_HEADER clause. Other language processors ignore it.
- If you specify more than one quoted string, VAX DATATRIEVE stacks them. Use multiple quoted strings to specify long query headers.
- If you are using a terminal of the VT200 family, you can use 8-bit characters in CDDL quoted strings.

### Example

In the following example, the QUERY_HEADER clause specifies that VAX DATATRIEVE use TOTAL PRICE as the column header for the field TOTAL_PRICE.

```
TOTAL_PRICE                DATATYPE IS VIRTUAL FIELD
                           COMPUTED BY DTR AS
                               "UNIT_PRICE * QUANTITY"
                           QUERY_HEADER FOR DTR IS "TOTAL PRICE".
```

## 2.23 QUERY_NAME Field Attribute Clause

Provides VAX DATATRIEVE with an alternate reference name for a field. VAX DATATRIEVE allows you to refer to a field either by its field name or by a specified query name.

### Format

$$\text{QUERY\_NAME FOR } \left\{ \begin{array}{l} \text{DTR} \\ \text{DATATRIEVE} \end{array} \right\} \text{ [ IS ] quoted-string}$$

### Parameter

quoted-string

The VAX DATATRIEVE query name.

### Syntax Rule

The quoted string must be a valid VAX DATATRIEVE query name. The CDDL compiler does not check the quoted string for correct syntax.

### Usage Notes

- Only VAX DATATRIEVE supports the QUERY_NAME clause. Other processors ignore it.
- You can specify a query name for a field only if you have specified a field name for it in the field description statement. You cannot specify a query name for unnamed fields.
- QUERY_NAME is different from NAME because VAX DATATRIEVE recognizes both the query name and the original field name. With NAME, the facility-specific name is the only name recognized by the specified language or language processor.
- Choose a query name that is shorter and easier to remember than the actual field name. You can then choose a field name that is descriptive of the field's contents and purpose.
- If you are using a terminal of the VT200 family, you can use 8-bit characters in CDDL quoted strings.

### Example

In the following example, the QUERY_NAME clause specifies TP as an alternate DATATRIEVE name for the field TOTAL_PRICE.

```
TOTAL_PRICE                DATATYPE IS VIRTUAL FIELD
                           COMPUTED BY DTR AS
                              "UNIT_PRICE * QUANTITY"
                           QUERY_NAME FOR DTR IS "TP".
```

## 2.24 STRUCTURE Field Description Statement

Defines a field that is divided into one or more subordinate fields.

### Format

```
[ /* text */  ]

{     *      }
{ field-name } STRUCTURE [ field-attribute ] . . . .

    field-description-statement
    [field-description-statement] . . .

END [ field-name ] [ STRUCTURE ].
```

### Parameters

text

Explanatory text describing the field.

field-name

The field's given name.

field-attribute

The characteristics of the STRUCTURE field.

field-description-statement

The characteristics of a subordinate field.

### Syntax Rules

- The field name you assign can be up to 31 characters from the set A-Z, 0-9, _, and $. The first character must be a letter from A-Z, and the last character cannot be _ or $. If you are using a terminal of the VT200 family, you can use 8-bit alphabetic characters in field names. Remember that other terminals cannot reproduce 8-bit characters.
- If you use an asterisk (*) instead of a field-name, you create an unnamed field.

- If you do not specify a data type for the STRUCTURE, the CDDL assigns it the UNSPECIFIED data type.
- Subordinate field description statements describe contiguous portions of the field described by the STRUCTURE.
- There must be at least one subordinate field description statement. A subordinate field can be an elementary, a STRUCTURE, a COPY, or a VARIANTS field.
- CDDL accepts the keyword GROUP as a synonym for STRUCTURE, but the compiler issues a warning when you use GROUP.
- You must terminate the STRUCTURE and the END statements with periods.

### Usage Notes

- Unnamed fields are similar to FILLER fields in VAX COBOL. You can use them to format print records or to reserve space in a record for future additions.
- In a STRUCTURE field, you can use any field attribute clauses allowed in an elementary field. However, if you use the DATATYPE field description statement, you cannot create subordinate fields that exceed the length of the structure field.
- A STRUCTURE field cannot contain the VIRTUAL FIELD datatype.
- Although the CDDL compiler accepts data type specifications for STRUCTURE fields, the feature may not be supported by the language or language processor you use with the CDD. Make sure the definitions you store in the dictionary are valid for the processor that will use them.
- Each field, except BIT fields, begins on the first byte following the preceding field. BIT fields begin on the bit immediately following the preceding field. You can modify this starting position with the ALIGNED clause. See Section 2.1.

## Example

As this example shows, you can nest STRUCTURE field description statements. The STRUCTURE field ADDRESS, for example, has a subordinate STRUCTURE field, ZIP_CODE.

```
ADDRESS STRUCTURE.
    STREET                    DATATYPE IS TEXT
                              SIZE IS 30 CHARACTERS.
    CITY                      DATATYPE IS TEXT
                              SIZE IS 30 CHARACTERS.
    STATE                     DATATYPE IS TEXT
                              SIZE IS 2 CHARACTERS.
    ZIP_CODE STRUCTURE.
        NEW                   DATATYPE IS UNSIGNED NUMERIC
                              SIZE IS 4 DIGITS
                              BLANK WHEN ZERO.
        OLD                   DATATYPE IS UNSIGNED NUMERIC
                              SIZE IS 5 DIGITS.
    END ZIP_CODE STRUCTURE.
END ADDRESS STRUCTURE.
```

## 2.25 VALID FOR DATATRIEVE IF Field Attribute Clause

Causes VAX DATATRIEVE to validate value assignments to a field. VAX DATATRIEVE refuses to assign a value to a field if that value is not accepted by this validation expression.

### Format

$$\text{VALID FOR} \left\{ \begin{matrix} \text{DTR} \\ \text{DATATRIEVE} \end{matrix} \right\} \text{IF quoted-string [quoted-string]} \ldots$$

### Parameter

quoted-string

VAX DATATRIEVE source text forming a validation expression.

### Syntax Rule

The quoted strings must form a valid VAX DATATRIEVE validation expression. The CDDL compiler does not check the validation expression for correct syntax.

### Usage Notes

- Only VAX DATATRIEVE can interpret these validation expressions. Other language processors ignore their presence.
- If you are using a terminal of the VT200 family, you can use 8-bit characters in CDDL quoted strings.

### Example

In the following example, the VALID FOR DATATRIEVE IF clause causes VAX DATATRIEVE to accept only positive integers for the field TRANSACTION_COUNT.

```
TRANSACTION_COUNT       DATATYPE IS UNSIGNED WORD
                        VALID FOR DTR IF
                            "TRANSACTION_COUNT > 0".
```

## 2.26 VARIANTS Field Description Statement

Defines a set of two or more fields mapping the same portion of a record definition. VARIANT fields are similar to fields defined with the REDEFINES clause in VAX COBOL and VAX DATATRIEVE.

There are two formats for the VARIANTS field description statement:

- VARIANTS is almost identical to the REDEFINES clause. Within an application program, you can refer to any of the VARIANT fields.
- VARIANTS OF uses the value of a tag variable at run time to determine which of the VARIANT fields is the current VARIANT.

### 2.26.1 VARIANTS Field Description Statement

Defines two or more logical views of the same portion of a record definition. The VARIANTS statement functions like the REDEFINES clause in VAX COBOL and VAX DATATRIEVE.

#### Format

```
VARIANTS .
          VARIANT.
                 field-description-statement
                 [field-description-statement] . . .
          END [ VARIANT ].

          VARIANT.
                 field-description-statement
                 [field-description-statement] . . .
          END [ VARIANT ].
                              .
                              .
                              .
END [ VARIANTS ].
```

#### Parameters

field-description-statement

A definition of the field characteristics for each subordinate field of each VARIANT.

#### Syntax Rules

- The vertical ellipsis in the format indicates that the VARIANT field description block can be repeated.
- The VARIANTS, VARIANT, and END statements all must end with periods.

#### Usage Notes

- Each VARIANT begins on the same byte in the record, subject to individual alignment options (see Section 2.1). The length of the longest VARIANT in the collection determines the overall length of the VARIANTS field description.

- Be sure that the VARIANTS collection you define conforms to the requirements of the language or language processor that will access the record definition.
- VAX DATATRIEVE requires each VARIANT to contain a STRUCTURE field description statement at the top of the VARIANT.

## Example

The following example contains three VARIANT logical views of the same record. In an application program, you can refer to IN_STOCK, BACK_ORDER, or OUT_OF_STOCK depending on how you want to interpret the STOCK field.

```
STOCK STRUCTURE.
    VARIANTS.
        VARIANT.
            IN_STOCK STRUCTURE.
                PRODUCT_NO     DATATYPE IS TEXT
                               SIZE IS 8 CHARACTERS.
                DATE_ORDERED   DATATYPE IS DATE.
                STATUS_CODE    DATATYPE IS BYTE.
                QUANTITY       DATATYPE IS LONGWORD
                               ALIGNED ON LONGWORD.
                LOCATION       ARRAY 1:4
                               DATATYPE IS TEXT
                               SIZE IS 30 CHARACTERS.
                UNIT_PRICE     DATATYPE IS LONGWORD SCALE -2.
            END IN_STOCK STRUCTURE.
        END VARIANT.
        VARIANT.
            BACK_ORDER STRUCTURE.
                PRODUCT_NO     DATATYPE IS TEXT
                               SIZE IS 8 CHARACTERS.
                DATE_ORDERED   DATATYPE IS DATE.
                STATUS_CODE    DATATYPE IS BYTE.
                QUANTITY       DATATYPE IS LONGWORD
                               ALIGNED ON LONGWORD.
                SUPPLIER       ARRAY 1:4
                               DATATYPE IS TEXT
                               SIZE IS 30 CHARACTERS.
                UNIT_PRICE     DATATYPE IS LONGWORD
                               SCALE -2.
            END BACK_ORDER STRUCTURE.
        END VARIANT.
        VARIANT.
            OUT_OF_STOCK STRUCTURE.
                PRODUCT_NO         DATATYPE IS TEXT
                                   SIZE IS 8 CHARACTERS.
                DATE_LAST_SOLD     DATATYPE IS DATE.
            END OUT_OF_STOCK STRUCTURE.
        END VARIANT.
    END VARIANTS.
END STOCK STRUCTURE.
```

### 2.26.2 VARIANTS OF Field Description Statement

Names a tag variable whose value at run time determines which of the VARIANT fields is the current VARIANT.

**Format**

```
VARIANTS  OF field-name .

     VARIANT     {VALUE  [IS] }  n1 [THRU n2] [n3 [THRU n4] ] ... .
                 {VALUES [ARE]}
           field-description-statement
           [field-description-statement] ...
     END  [VARIANT] .

     VARIANT     {VALUE  [IS] }  n5 [THRU n6] [n7 [THRU n8] ] ... .
                 {VALUES [ARE]}
           field-description-statement
           [field-description-statement] ...
     END  [VARIANT] .
                            .
                            .
                            .
END [VARIANTS] .
```

**Parameters**

field-name

The tag variable field, whose value determines the selection of the current VARIANT at run time.

n1, n2, n3, n4, n5, n6, n7, n8

Values to be compared to the value of the tag variable at run time to determine the current VARIANT.

field-description-statement

A definition of the field characteristics for the subordinate fields of each VARIANT.

### Syntax Rules

- The vertical ellipsis in the format indicates that the VARIANT field description block can be repeated.
- The tag variable must be an elementary field fixed in the record and not part of an array, and it must precede the VARIANTS field description statement. It cannot be an unnamed field.
- You must fully qualify the tag variable's field name if it is not unique within the record. A fully qualified field name consists of an elementary field name preceded by the field names of as many of its direct ancestors as you need to specify the elementary field uniquely. You cannot omit the names of any of the ancestor fields within the fully qualified name, but once the elementary field name is identified uniquely, you can omit any remaining ancestors' field names. You must separate each element of a fully qualified field name from the next with a period. Furthermore, if the tag variable's name is not unique within the record, none of the ancestors in its fully qualified name can be unnamed fields.
- You must include a tag value clause for VARIANT.
- At run time, the values (n1, n2, n3, n4, . . .) you specify are compared to the value of the tag variable to determine the current VARIANT.
- Tag value specifications must conform to the following conditions:
  - The tag values n1, n2, n3, n4, . . . must be legal values as defined by the tag variable's data type. For example, if the tag variable is a TEXT field, the tag values must be quoted literals.
  - The values of n2, n4, n6, and n8 must be greater than or equal to the values of n1, n3, n5, and n7 in the collating sequence of the data type.
  - The range of values in one VARIANT must not overlap the range of values in any other VARIANT.
- The compiler ignores commas, but you can use them to make tag value specifications easier to read.
- The CDDL compiler does not check that the tag values you specify are legal. If you specify invalid values, you will receive error messages when you refer to the VARIANT field in an application.

### Usage Notes

- The tag value clause specifies a distinct value or set of values for the tag variable in each VARIANT of a VARIANTS field collection. The tag variable can then be used at run time to find the current VARIANT.
- Languages that do not support tag variables ignore the tag value clause. For more information on language support of CDD record definitions, refer to the documentation for the language you are using.
- Each variant begins on the same byte in the record, subject to individual alignment options (see Section 2.1). The length of the longest VARIANT in the collection determines the overall length of the VARIANTS field description.
- Be sure that the VARIANT collection you define conforms to the requirements of the language or language processor that will access the record definition.
- VAX DATATRIEVE requires each VARIANT to contain a STRUCTURE field description statement at the top of the VARIANT.

## Example

In the following example, RECORD_IDENTIFIER is the tag variable. The value of RECORD_IDENTIFIER at run time determines which VARIANT is current according to the translation table in the descriptive text.

```
STOCK STRUCTURE.
    /* RECORD_IDENTIFIER determines field type:
         S --> In-stock record.
         B --> Back-order record.
         O --> Out-of-stock record. */
    RECORD_IDENTIFIER       .   DATATYPE IS TEXT
                                SIZE IS 1 CHARACTER.
    VARIANTS OF RECORD_IDENTIFIER.
        VARIANT VALUE IS "S".
            IN_STOCK STRUCTURE.
                PRODUCT_NO      DATATYPE IS TEXT
                                SIZE IS 8 CHARACTERS.
                DATE_ORDERED    DATATYPE IS DATE.
                STATUS_CODE     DATATYPE IS BYTE.
                QUANTITY        DATATYPE IS LONGWORD
                                ALIGNED ON LONGWORD.
                LOCATION        ARRAY 1:4
                                DATATYPE IS TEXT
                                SIZE IS 30 CHARACTERS.
                UNIT_PRICE      DATATYPE IS LONGWORD SCALE -2.
            END IN_STOCK STRUCTURE.
        END VARIANT.
        VARIANT VALUE IS "B".
            BACK_ORDER STRUCTURE.
                PRODUCT_NO      DATATYPE IS TEXT
                                SIZE IS 8 CHARACTERS.
                DATE_ORDERED    DATATYPE IS DATE.
                STATUS_CODE     DATATYPE IS BYTE.
                QUANTITY        DATATYPE IS LONGWORD
                                ALIGNED ON LONGWORD.
                SUPPLIER        ARRAY 1:4
                                DATATYPE IS TEXT
                                SIZE IS 30 CHARACTERS.
                UNIT_PRICE      DATATYPE IS LONGWORD
                                SCALE -2.
            END BACK_ORDER STRUCTURE.
        END VARIANT.

        VARIANT VALUE IS "O".
            OUT_OF_STOCK STRUCTURE.
                PRODUCT_NO          DATATYPE IS TEXT
                                    SIZE IS 8 CHARACTERS.
                DATE_LAST_SOLD      DATATYPE IS DATE.
            END OUT_OF_STOCK STRUCTURE.
        END VARIANT.
    END VARIANTS.
END STOCK STRUCTURE.
```

# The CDDL Compiler Command Descriptions 3

After you create a source file containing record definitions and data descriptions, use the CDDL compiler to place those definitions into the CDD. There are two command formats:

- The CDDL command, which compiles source files
- The CDDL/RECOMPILE command, which locates existing CDD record definitions and recompiles them

Both command formats allow you to create history list entries and listings of the source text.

**Note**

Before you begin storing definitions in the CDD, your system manager or data administrator should carefully plan, create, and protect your CDD directory hierarchy. See the *VAX CDD/Plus User's Guide, Appendix A*, for information about planning, creating, protecting, and maintaining the CDD directory hierarchy.

The following sections contain complete descriptions of CDDL commands, parameters, and qualifiers. These descriptions include:

- The syntax **format** you should use for each command
- The **parameters** of each command
- Command **qualifiers** that modify the functions of each command
- **Restrictions** on the ways you can use commands

- **Usage notes** to show you how to use each command
- **Required privileges** for each command
- **Examples** from the sample dictionary (Figure 1-1) to illustrate the use of each command

## 3.1 CDDL Command

Compiles source files and places record definitions in the CDD.

### Format

```
CDDL file-specification

Qualifiers                              Defaults

/[NO] ACL                               /ACL
/AUDIT [ = (quoted-string
        [, quoted-string] ...)]
/AUDIT = file-specification
/NOAUDIT                                /NOAUDIT
/[NO]COPY_LIST                          /NOCOPY_LIST
/[NO]DIAGNOSTICS                        /NODIAGNOSTIC
/LISTING[=file-specification]           /LISTING
/NOLISTING
/PATH = path-name                       /PATH = CDD$DEFAULT
/[NO]REPLACE                            /NOREPLACE
/[NO]VERSION                            /NOVERSION
/V2
```

### Parameter

file-specification

The CDDL source file you want to compile. Each source file contains one or more CDD record definitions.

The file specification is a standard VMS file specification. The default file type is .DDL.

### Qualifiers

/ACL

Creates an access control list for each record definition in the source file. The contents of the access control list created vary, depending on the qualifiers used and whether the target directory already contains one or more versions of the record definition.

- If the directory in which CDDL stores the record definition does not already contain a record definition with the same name, CDDL creates the default access control list, which grants you, the creator, the following privileges: CONTROL, LOCAL_DELETE, DTR_EXTEND/EXECUTE, HISTORY, DTR_MODIFY, DTR_READ, SEE, UPDATE, and DTR_WRITE.
- If the directory contains one or more versions of a record definition with the same name as the definition CDDL is compiling and you use the /REPLACE qualifier, CDDL copies the contents of the access control list of the replaced version into the access control list of the new version.
- If the directory contains one or more versions of a record definition with the same name as the definition CDDL is compiling and you use the /VERSION qualifier, CDDL copies the contents of the access control list of the highest existing version into the access control list of the new version.

/NOACL

Prevents the creation of an access control list.

/AUDIT [= (quoted-string [, quoted-string]...)]
/AUDIT = file-specification

Creates a history list entry auditing the creation of each record definition.

You can include explanatory text in the history list entries in two ways:

- By including quoted strings. Enclose each quoted string in double quotation marks, and enclose the series of strings in parentheses. The parentheses are optional if you specify only one quoted string.
- By specifying a file whose contents are to be included in the history list entry. The file specification is a standard VMS file specification, and the default file type is .DAT.

  You can include no more than 64 input strings in a history list entry. The compiler ignores any excess.

If you are using a terminal of the VT200 family, you can use 8-bit characters in CDDL quoted strings.

/NOAUDIT

Prevents the creation of a history list entry.

/COPY_LIST

Expands in the listing file all template records included in a record description. The CDDL compiler creates CDDL source text from all template records and inserts that source text into the listing file. In the listing, the CDDL compiler inserts a "T" as the first character of each line that is part of an expanded template record.

/NOCOPY_LIST

Prevents the expansion of template records in the listing file.

/DIAGNOSTICS

Creates a diagnostics file that lists errors occurring during compilation. This qualifier is designed for use with the VAX Language-Sensitive Editor (LSE), and the diagnostics file is reserved for use by DIGITAL. LSE uses the file to display diagnostic messages and to position the cursor where a source error exists. The diagnostics file has the name of your source definition file and the default extension .DIA.

/NODIAGNOSTICS

Prevents creation of the diagnostics file. The default is /NODIAGNOSTICS.

/LISTING [=file-specification]

Writes an output file containing the command line, the source text, and CDDL messages. The file specification is a standard VMS file specification, and the default file type is .LIS.

/NOLISTING

Prevents creation of the listing file.

/PATH = path-name

Names a default directory from which to trace the path names in the CDDL source files. The CDDL uses your CDD$DEFAULT directory if you do not specify /PATH.

/REPLACE

Deletes an existing CDD record definition and inserts a new definition in its place. The new record definition copies the access control list and the history list of the record definition it replaces.

/NOREPLACE

Prevents the replacement of an existing record definition.

/VERSION

Creates an additional version of an existing CDD record. If you specify an absolute version number in the path name, the new record has that version number. If you do not, it has a version number one higher than the highest existing version.

If the logical name CDD$VERSION_LIMIT has been defined for your system, group, or process, the dictionary will store only the number of versions allowed by the quota CDD$VERSION_LIMIT specifies.

The newly created record definition copies the access control list of the highest existing version of the record definition. If no version exists, the newly created record definition contains the default access control list. You can prevent the creation of any access control list by using the /NOACL qualifier.

The newly created record definition copies the history list of the highest existing version.

/NOVERSION

Prevents the creation of a new version of an existing record definition.

/V2

The /V2 qualifier causes the compiler to use the CDD V2.0 defaults for the signs of the fixed point numbers. With this qualifier, BYTE, WORD, and LONGWORD data types are unsigned by default, but QUADWORD and OCTAWORD data types are signed.

If you do not specify the /V2 qualifier, all the fixed point data types are unsigned by default.

### Restrictions

- You cannot specify both /NOLISTING and /COPY_LIST.
- You cannot specify both /REPLACE and /VERSION.
- You cannot specify both /REPLACE and /NOACL. When you use the /REPLACE qualifier, the new definition always copies the access control list of the definition it replaces.

### Usage Notes

- You can include passwords in the DEFINE statement path name, but this is not recommended because passwords included in the source text can be displayed with the DMU LIST/ITEM=SOURCE command. Instead, use the /PATH qualifier with the compile command if passwords are required in the path name.

- If you use any qualifiers, you must type the source file specification on the same command line as the CDDL command. If you press RETURN before giving the file specification, CDDL does not prompt you for it.
- If you specify the version number of a record definition in the source file and a record definition with the same name and version number already exists, the compilation will fail if you use the /VERSION qualifier. To correct this problem, you can change or eliminate the version number in the source file or use /REPLACE instead of /VERSION.

### Required Privileges

- You need SEE and PASS_THRU access to the lowest existing ancestor of the dictionary object you are creating. You also need EXTEND unless you are using /REPLACE.
- If you use /REPLACE with the CDDL command, you need SEE, PASS_THRU, UPDATE, and either LOCAL_DELETE or GLOBAL_DELETE access to the dictionary object you are replacing.
- If you use /VERSION with the CDDL command, you need SEE, PASS_THRU, and UPDATE access to the highest version of the dictionary object you are compiling. Furthermore, if you use /NOACL with /VERSION, you must have CONTROL access to the highest existing version of the record definition.

### Examples

The following command inserts the record definition contained in a file named EMPLOYEE.DDL into the CDD. The new record definition has the default access control list and a history list entry containing the explanation, "Initial compile". A source file can contain any number of record definitions, but the CDDL creates only one listing file for each source file.

```
$ CDDL/AUDIT="Initial compile" EMPLOYEE.DDL
```

When the source file is compiled, CDDL creates a listing file, EMPLOYEE.LIS, in your default VMS directory. EMPLOYEE.LIS contains the command line you entered, the CDDL source file, and certain CDDL messages.

```
Command Line:  CDDL/AUDIT="Initial compile" EMPLOYEE.DDL        Source File:
DB3:[CASADAY.CDD]EMPLOYEE.DDL;1

       0001  DEFINE RECORD CDD$TOP.CORPORATE.EMPLOYEE_LIST
       0002    DESCRIPTION IS
       0003      /* This record contains the master list of all
       0004         employees */.
       0005    EMPLOYEE STRUCTURE.
       0006      /* An employee's ID number is his
       0007         or her social security number */
       0008      ID                  DATATYPE IS UNSIGNED NUMERIC
       0009                          SIZE IS 9 DIGITS.
       0010      NAME STRUCTURE.
       0011        LAST_NAME         DATATYPE IS TEXT
       0012                          SIZE IS 15 CHARACTERS.
       0013        FIRST_NAME        DATATYPE IS TEXT
       0014                          SIZE IS 10 CHARACTERS.
       0015        MIDDLE_INITIAL    DATATYPE IS TEXT
       0016                          SIZE IS 1 CHARACTER.
       0017      END NAME STRUCTURE.
       0018
       0019      ADDRESS             COPY FROM
       0020                          CDD$TOP.CORPORATE.ADDRESS_RECORD.
       0021      DEPT_CODE           DATATYPE IS UNSIGNED NUMERIC
       0022                          SIZE IS 3 DIGITS.
       0023    END EMPLOYEE STRUCTURE.
       0024  END EMPLOYEE_LIST RECORD.
                                                   1
%CDDL-S-RECORDCRE, record "CDD$TOP.CORPORATE.EMPLOYEE_LIST;1" created in the CDD
```

**Example 3-1: Sample CDDL Listing File**

The following example demonstrates how to create an additional version of a record definition.

The sample dictionary (Figure 1-1) contains two versions of the record definition CDD$TOP.PERSONNEL.STANDARDS.SALARY_RANGE. The file SALARY2.DDL contains the CDDL source file used to create CDD$TOP.PERSONNEL.STANDARDS.SALARY_RANGE;2:

```
$ TYPE SALARY2.DDL
DEFINE RECORD CDD$TOP.PERSONNEL.STANDARDS.SALARY_RANGE
    DESCRIPTION IS
        /* This record stores minimum salaries
        for the four incremental salary levels within
        each of 150 job classifications. It reflects
        Personnel Policy effective 1/1/84. */.
    SALARY_RANGE     ARRAY 150,4
                     DATATYPE IS UNSIGNED NUMERIC
                     SIZE IS 8 DIGITS 2 FRACTIONS.
END SALARY_RANGE RECORD.
```

The following example compiles SALARY2.DDL and adds it to a directory already containing SALARY_RANGE;1.

```
$ CDDL/VERSION/AUDIT SALARY2.DDL
```

The /VERSION qualifier causes the CDDL compiler to create a new version of SALARY_RANGE while retaining SALARY_RANGE;1. Because the source file does not specify the version number of the record definition, CDDL gives the new definition a version number of 2, one higher than the highest existing version. SALARY_RANGE;2 copies the access control list of SALARY_RANGE;1 because /ACL is the default. SALARY_RANGE;2 also copies the history list of SALARY_RANGE;1 and adds an entry documenting its creation.

## 3.2 CDDL/RECOMPILE Command

Recompiles CDD record definitions from the source text stored in the dictionary. Use CDDL/RECOMPILE to update record definitions containing a COPY field description statement after you modify the template record that the field copies. CDDL/RECOMPILE recompiles those record definitions and updates the definitions referred to in the COPY field description statements.

### Format

```
CDDL/RECOMPILE path-name [ , path-name ] ...


Qualifiers                            Defaults

/[ NO ] ACL                           /ACL
/AUDIT [ = (quoted-string
           [ , quoted-string ] ...) ]
/AUDIT = file-specification
/NOAUDIT                              /NOAUDIT
/[ NO ] COPY_LIST                     /NOCOPY_LIST
/LISTING [ = file-specification ]     /LISTING
/NOLISTING
/PATH = path-name                     /PATH = CDD$DEFAULT
/[ NO ] VERSION                       /NOVERSION
/V2
```

### Parameter

path-name

Specifies a record definition to be recompiled. You can use the absolute or the relative path name.

### Qualifiers

/ACL

Creates an access control list for the recompiled record definition. If you do not use the /VERSION qualifier, the recompiled definition copies the access control list of the version that you recompiled. If you use the /VERSION qualifier, the recompiled definition copies the access control list of the highest existing version of the record definition.

/NOACL

Prevents the creation of an access control list when it is used with /VERSION. Unless you specify both /NOACL and /VERSION, you cannot prevent the creation of an access control list.

/AUDIT [= (quoted-string [, quoted-string]...)]
/AUDIT = file-specification

Creates a history list entry auditing the recompilation of each record definition.

You can include explanatory text in the history list entries in two ways:

- By including quoted strings. Enclose each quoted string in double quotation marks, and enclose the series of strings in parentheses. The parentheses are optional if you specify only one quoted string.
- By specifying a file whose contents are to be included in the history list entry. The file specification is a standard VMS file specification, and the default file type is .DAT. You can include no more than 64 input strings in a history list entry. The compiler ignores any excess.

If you are using a terminal of the VT200 family, you can use 8-bit characters in CDDL quoted strings.

/NOAUDIT

Prevents the creation of a history list entry documenting the recompilation.

/COPY_LIST

Expands in the listing file all template records included in a record description. The CDDL compiler creates CDDL source text from all template records and inserts that source text into the listing file. In the listing, the CDDL compiler inserts a "T" as the first character of each line that is part of an expanded template record.

/NOCOPY_LIST

Prevents the expansion of template records in the listing file.

/LISTING [=file-specification]

Writes an output file containing the source text of the record definition and CDDL messages. The file specification is a standard VMS file specification, and the default file type is .LIS. /LISTING is the default; however, if you do not use /LISTING to explicitly name the listing file, CDDL creates a file called .LIS.

/NOLISTING

Prevents creation of the listing file.

/PATH = path-name

Names a default directory from which to trace the path names in the CDDL source files. The CDDL uses your CDD$DEFAULT directory if you do not specify /PATH.

/VERSION

Recompiles an existing record definition and creates an additional version of it. CDDL/RECOMPILE/VERSION recompiles the record you specify in the path name parameter, but does not replace it. Instead, it creates an additional version of the record definition with a version number one greater than the highest existing version.

/NOVERSION

Prevents the creation of an additional version of the recompiled definition.

/V2

Causes the compiler to use the CDD V2.0 defaults for the signs of the fixed point numbers. With this qualifier, BYTE, WORD, and LONGWORD data types are unsigned by default, but QUADWORD and OCTAWORD data types are signed.

If you do not specify the /V2 qualifier, all the fixed point data types are unsigned by default.

## Restrictions

- You cannot specify both /NOLISTING and /COPY_LIST.
- You must specify /VERSION if you specify /NOACL. Unless you are creating an additional version, the recompiled definition always copies the access control list of the definition it replaces.

### Usage Notes

- You can recompile CDD record definitions only if the CDDL compiler created the record definition. Source text created by other processors may be unavailable or incompatible with CDDL syntax.
- The CDDL compiler accepts password specifications for the given names of new record definitions, but passwords included in the source can be displayed with the DMU LIST/ITEM=SOURCE command.

### Required Privileges

- You need PASS_THRU and EXTEND access to the parent of the object you are recompiling. You also need SEE and PASS_THRU access to the object. If you are not using /VERSION, you need UPDATE and either LOCAL_DELETE or GLOBAL_DELETE access to the object.
- If you use /VERSION with the CDDL/RECOMPILE command, you need SEE, PASS_THRU, and UPDATE access to the highest version of the dictionary object you are compiling. Furthermore, if you use /NOACL with /VERSION, you must have CONTROL access to the highest existing version of the record.

### Examples

In the following example, the field ADDRESS, which is copied from the CORPORATE directory, is recompiled because of modifications made to the template record, CDD$TOP.CORPORATE.ADDRESS_RECORD.

```
CDD$TOP.SALES.CUSTOMER_RECORD
    CUSTOMER STRUCTURE.
        NAME        DATATYPE IS TEXT
                    SIZE IS 30 CHARACTERS.
        ADDRESS     COPY FROM
                    CDD$TOP.CORPORATE.ADDRESS_RECORD.
    END CUSTOMER STRUCTURE.

$ CDDL/RECOMPILE/AUDIT CDD$TOP.SALES.CUSTOMER_RECORD
```

In the following example, CDD$TOP.SALES.CUSTOMER_RECORD is again recompiled. In this case, however, the CDDL compiler creates an additional version because the /VERSION qualifier is used. The new version contains the same access control list and history list as the highest existing version of CDD$TOP.SALES.CUSTOMER_RECORD.

```
$ CDDL/RECOMPILE/VERSION/LISTING=CUSTOMER.LIS/COPY_LIST -
$_CDD$TOP.SALES.CUSTOMER_RECORD
```

When the above record definition is recompiled, CDDL creates a listing file, CUSTOMER.LIS, in your default VMS directory. CUSTOMER.LIS contains the command line you entered, the source text of the recompiled definition, and certain CDDL messages. Because /COPY_LIST is used, the CDDL compiler expands the definition to include source text created from the template record, CDD$TOP.CORPORATE.ADDRESS_RECORD.

```
VAX CDD Data Definition Language Utility  Version 3.0  5-MAR-1984  11:16:33:25  Page  1
Command Line: CDDL /RECOMPILE/VERSION/COPY_LIST CDD$TOP.SALES. CUSTOMER_RECORD
Source File:
  0001     DEFINE RECORD CDD$TOP.SALES.CUSTOMER_RECORD
  0002     DESCRIPTION IS
  0003          /* This record is of primary use to the marketing
  0004          department. It contains the names, addresses, and
  0005          phone numbers of all current customers. */.
  0006        CUSTOMER STRUCTURE.
  0007          NAME                     DATATYPE IS TEXT
  0008                                   SIZE IS 30 CHARACTERS.
  0009          ACCOUNT_NUMBER           DATATYPE IS UNSIGNED NUMERIC
  0010                                   SIZE IS 7 DIGITS.
  0011          ADDRESS
      T              ADDRESS STRUCTURE.
      T                  STREET          DATATYPE IS TEXT
      T                                  SIZE IS 30 CHARACTERS.
      T                  CITY            DATATYPE IS TEXT
      T                                  SIZE IS 30 CHARACTERS.
      T                  STATE           DATATYPE IS TEXT
      T                                  SIZE IS 2 CHARACTERS.
      T                  ZIP_CODE STRUCTURE.
      T                      NEW             DATATYPE IS UNSIGNED NUMERIC
      T                                      SIZE IS 4 DIGITS
      T                                      BLANK WHEN ZERO.
      T                      OLD             DATATYPE IS UNSIGNED NUMERIC
      T                                      SIZE IS 5 DIGITS.
      T                  END ZIP_CODE STRUCTURE
      T              END ADDRESS STRUCTURE.
  0012                                   CDD$TOP.CORPORATE.ADDRESS_RECORD.
  0013          TELEPHONE STRUCTURE.
  0014              AREA_CODE             DATATYPE IS UNSIGNED NUMERIC
  0015                                    SIZE IS 3 DIGITS.
  0016              NUMBER                DATATYPE IS UNSIGNED NUMERIC
  0017                                    SIZE IS 7 DIGITS.
  0018          END TELEPHONE STRUCTURE.
  0019        END CUSTOMER STRUCTURE.
  0020     END CUSTOMER_RECORD.
                                                                 1
%CDDL-S-RECOMPVER, record "CDD$TOP.SALES.CUSTOMER_RECORD;1" recompiled
and "CDD$TOP.SALES.CUSTOMER_RECORD;2" created in the CDD
```

# SOURCE.DDL: The Source File for Examples in This Manual A

```
DEFINE RECORD CDD$TOP.CORPORATE.ADDRESS_RECORD
    DESCRIPTION IS
        /* This record contains the standard format
        for addresses.  It provides the source from which all
        address fields in other record descriptions are copied. */.
    ADDRESS STRUCTURE.
        STREET                  DATATYPE IS TEXT
                                SIZE IS 30 CHARACTERS.
        CITY                    DATATYPE IS TEXT
                                SIZE IS 30 CHARACTERS.
        STATE                   DATATYPE IS TEXT
                                SIZE IS 2 CHARACTERS.
        ZIP_CODE STRUCTURE.
            NEW                 DATATYPE IS UNSIGNED NUMERIC
                                SIZE IS 4 DIGITS
                                BLANK WHEN ZERO.
            OLD                 DATATYPE IS UNSIGNED NUMERIC
                                SIZE IS 5 DIGITS.
        END ZIP_CODE STRUCTURE.
    END ADDRESS STRUCTURE.
END ADDRESS_RECORD.

DEFINE RECORD CDD$TOP.CORPORATE.EMPLOYEE_LIST
    DESCRIPTION IS
        /* This record contains the master list of all
        employees */.
    EMPLOYEE STRUCTURE.
        /* An employee's ID number is his
        or her social security number */
        ID                      DATATYPE IS UNSIGNED NUMERIC
                                SIZE IS 9 DIGITS.
        NAME STRUCTURE.
            LAST_NAME           DATATYPE IS TEXT
            .                   SIZE IS 15 CHARACTERS.
            FIRST_NAME          DATATYPE IS TEXT
                                SIZE IS 10 CHARACTERS.
```

(continued on next page)

```
            MIDDLE_INITIAL  DATATYPE IS TEXT
                            SIZE IS 1 CHARACTER.
        END NAME STRUCTURE.
        ADDRESS             COPY FROM
                            CDD$TOP.CORPORATE.ADDRESS_RECORD.
        DEPT_CODE           DATATYPE IS UNSIGNED NUMERIC
                            SIZE IS 3 DIGITS.
    END EMPLOYEE STRUCTURE.
END EMPLOYEE_LIST RECORD.

DEFINE RECORD CDD$TOP.CORPORATE.PRODUCT_INVENTORY
    DESCRIPTION IS
        /* This record is the primary location of inventory
        status information. */.
    INVENTORY STRUCTURE.
        STOCK STRUCTURE.
            /* RECORD_IDENTIFIER determines field type:
                S --> In-stock record.
                B --> Back-order record.
                O --> Out-of-stock record. */
            RECORD_IDENTIFIER       DATATYPE IS TEXT
                                    SIZE IS 1 CHARACTER
                                    CONDITION FOR COBOL IS ON_HAND
                                        COBOL NAME "ON-HAND"
                                        VALUE IS "S"
                                    CONDITION FOR COBOL BACKORDER
                                        COBOL NAME "BACKORDER"
                                        VALUE IS "B"
                                    CONDITION FOR COBOL OUT_OF_STOCK
                                        COBOL NAME "OUT-OF-STOCK"
                                        VALUE IS "O"
                                    CONDITION FOR COBOL IS INVALID
                                        VALUES ARE "A", "C" THRU "N",
                                        "P" THRU "R", "T" THRU "Z".
            VARIANTS OF RECORD_IDENTIFIER.
                VARIANT VALUE IS "S".
                    IN_STOCK STRUCTURE.
                        PRODUCT_NO      DATATYPE IS TEXT
                                        SIZE IS 8 CHARACTERS.
                        DATE_ORDERED    DATATYPE IS DATE.
                        STATUS_CODE     DATATYPE IS BYTE.
                        QUANTITY        DATATYPE IS LONGWORD
                                        ALIGNED ON LONGWORD.
                        LOCATION        ARRAY 1:4
                                        DATATYPE IS TEXT
                                        SIZE IS 30 CHARACTERS.
                        UNIT_PRICE      DATATYPE IS LONGWORD SCALE -2.
                    END IN_STOCK STRUCTURE.
                END VARIANT.
                VARIANT VALUE IS "B".
                    BACK_ORDER STRUCTURE.
                        PRODUCT_NO      DATATYPE IS TEXT
                                        SIZE IS 8 CHARACTERS.
                        DATE_ORDERED    DATATYPE IS DATE.
                        STATUS_CODE     DATATYPE IS BYTE.
                        QUANTITY        DATATYPE IS LONGWORD
                                        ALIGNED ON LONGWORD.
```

(continued on next page)

```
                    SUPPLIER        ARRAY 1:4
                                    DATATYPE IS TEXT
                                    SIZE IS 30 CHARACTERS.
                    UNIT_PRICE      DATATYPE IS LONGWORD
                                    SCALE -2.
                END BACK_ORDER STRUCTURE.
            END VARIANT.
            VARIANT VALUE IS "O".
                OUT_OF_STOCK STRUCTURE.
                    PRODUCT_NO      DATATYPE IS TEXT
                                    SIZE IS 8 CHARACTERS.
                    DATE_LAST_SOLD DATATYPE IS DATE.
                END OUT_OF_STOCK STRUCTURE.
            END VARIANT.
        END VARIANTS.
    END STOCK STRUCTURE.
  END INVENTORY STRUCTURE.
END PRODUCT_INVENTORY RECORD.

DEFINE RECORD CDD$TOP.PERSONNEL.STANDARDS.SALARY_RANGE
    DESCRIPTION IS
        /* This record stores minimum salaries
        for the five incremental salary levels within
        each of 200 job classifications.  */.
    SALARY_RANGE    ARRAY 200,5
                    DATATYPE IS UNSIGNED NUMERIC
                    SIZE IS 8 DIGITS 2 FRACTIONS.
END SALARY_RANGE RECORD.

DEFINE RECORD CDD$TOP.PERSONNEL.SERVICE.SALARY_RECORD
    DESCRIPTION IS
        /* This is the record containing salary
        information for all employees.  It is sensitive, and access
        is carefully restricted. */.
    SALARY STRUCTURE.
        EMPLOYEE_ID         DATATYPE IS UNSIGNED NUMERIC
                            SIZE IS 9 DIGITS.
        PAY STRUCTURE.
            JOB_CLASS       DATATYPE IS UNSIGNED NUMERIC
                            SIZE IS 3 DIGITS.
            INCR_LEVEL      DATATYPE IS UNSIGNED NUMERIC
                            SIZE IS 1 DIGIT.
            WEEKLY_SALARY   DATATYPE IS UNSIGNED NUMERIC
                            SIZE IS 6 DIGITS 2 FRACTIONS.
        END PAY STRUCTURE.
    END SALARY STRUCTURE.
END SALARY_RECORD RECORD.

DEFINE RECORD CDD$TOP.SALES.CUSTOMER_RECORD
    DESCRIPTION IS
        /* This record is of primary use to the
        marketing department.  It contains the names, addresses, and
        phone numbers of all current customers. */.
    CUSTOMER STRUCTURE.
        NAME                  DATATYPE IS TEXT
                              SIZE IS 30 CHARACTERS.
        ACCOUNT_NUMBER        DATATYPE IS UNSIGNED NUMERIC
                              SIZE IS 7 DIGITS.
```

(continued on next page)

```
        ADDRESS                 COPY FROM
                                CDD$TOP.CORPORATE.ADDRESS_RECORD.
        TELEPHONE STRUCTURE.
            AREA_CODE           DATATYPE IS UNSIGNED NUMERIC
                                SIZE IS 3 DIGITS.
            NUMBER              DATATYPE IS UNSIGNED NUMERIC
                                SIZE IS 7 DIGITS.
        END TELEPHONE STRUCTURE.
    END CUSTOMER STRUCTURE.
END CUSTOMER_RECORD.

DEFINE RECORD CDD$TOP.SALES.JONES.LEADS_RECORD
    DESCRIPTION IS
        /* This record is in the personal directory of
        Jones, a supervisor in the marketing department.  It contains
        information about prospective customers and the revenues that
        landing these customers might generate. */.
    LEADS_RECORD STRUCTURE.
        CONTACT_NAME          DATATYPE IS TEXT
                              SIZE IS 30 CHARACTERS.
        COMPANY               DATATYPE IS TEXT
                              SIZE IS 30 CHARACTERS.
        ADDRESS               COPY FROM
                              CDD$TOP.CORPORATE.ADDRESS_RECORD.
        TELEPHONE STRUCTURE.
            AREA_CODE         DATATYPE IS UNSIGNED NUMERIC
                              SIZE IS 3 DIGITS.
            NUMBER            DATATYPE IS UNSIGNED NUMERIC
                              SIZE IS 7 DIGITS.
        END TELEPHONE STRUCTURE.
        POTENTIAL_ANN_SALES     DATATYPE IS UNSIGNED NUMERIC
                                SIZE IS 8 DIGITS 2 FRACTIONS.
    END LEADS_RECORD STRUCTURE.
END LEADS_RECORD.

DEFINE RECORD CDD$TOP.SALES.SALES_RECORD.
    SALES STRUCTURE.
        CUSTOMER_NAME           DATATYPE IS TEXT
                                SIZE IS 30 CHARACTERS.
        ACCOUNT_NUMBER          DATATYPE IS UNSIGNED NUMERIC
                                SIZE IS 7 DIGITS.
        TRANSACTION_COUNT       DATATYPE IS UNSIGNED WORD
                                VALID FOR DATATRIEVE IF
                                    "TRANSACTION_COUNT > 0".
        TRANSACTION STRUCTURE   OCCURS 1 TO 99 TIMES
                                    DEPENDING ON TRANSACTION_COUNT.
            TRANS_DATE          DATATYPE IS DATE
                                EDIT_STRING FOR DATATRIEVE
                                IS "NN/DD/YY".
            ORDER_NUMBER        DATATYPE IS UNSIGNED NUMERIC
                                SIZE IS 10 DIGITS
                                NAME FOR COBOL IS "ORDER-NUMBER".
            AMOUNT              DATATYPE IS UNSIGNED NUMERIC
                                SIZE IS 8 DIGITS 2 FRACTIONS
                                INITIAL_VALUE IS 0
                                PICTURE FOR COBOL IS "9(6)V99".
                END TRANSACTION STRUCTURE.
    END SALES STRUCTURE.
END SALES_RECORD RECORD.
```

(continued on next page)

```
DEFINE RECORD _CDD$TOP.CDD$EXAMPLES.PERSONNEL.SERVICE.SALARY_RECORD
    DESCRIPTION IS
        /* This is the record containing salary
        information for all employees.  It is sensitive, and access
        is carefully restricted. Direct deposit information added
        5-JAN-1984.*/.
    SALARY STRUCTURE.
        EMPLOYEE_ID        DATATYPE IS UNSIGNED NUMERIC
                           SIZE IS 9 DIGITS.
        PAY STRUCTURE.
            JOB_CLASS      DATATYPE IS TEXT
                           SIZE IS 3 CHARACTERS.
            INCR_LEVEL     DATATYPE IS UNSIGNED NUMERIC
                           SIZE IS 1 DIGIT.
            WEEKLY_SALARY  DATATYPE IS UNSIGNED NUMERIC
                           SIZE IS 6 DIGITS 2 FRACTIONS.
            DIRECT_DEP     DATATYPE IS TEXT
                           SIZE IS 1 CHARACTER
                           VALID FOR DTR IF
                              DIRECT_DEP=Y OR DIRECT_DEP=N.
        END PAY STRUCTURE.
    END SALARY STRUCTURE.
END SALARY_RECORD RECORD.

DEFINE RECORD _CDD$TOP.CDD$EXAMPLES.PERSONNEL.STANDARDS.SALARY_RANGE
    DESCRIPTION IS
        /* This record stores minimum salaries
        for the four incremental salary levels within
        each of 150 job classifications.  It reflects
        Personnel Policy effective 1/1/84. */.
    SALARY_RANGE     ARRAY 150,4
                     DATATYPE IS UNSIGNED NUMERIC
                     SIZE IS 8 DIGITS 2 FRACTIONS.
END SALARY_RANGE RECORD.
```

# CDDL Syntax Skeleton B

## B.1 DEFINE...END

```
DEFINE RECORD  path-name

         [DESCRIPTION  [IS]  /* text */ ] .

         field-description-statement

         ┌path-name ┐
END      │          │   [RECORD] .
         └given-name┘
```

## B.2 Field Description Statements

### B.2.1 Elementary Field Description

```
 [ /* text */ ]

 {    *     }
 {field-name}     field-attribute [field-attribute] ... .
```

## B.2.2 STRUCTURE Field Description

```
[/* text */]

{    *     }
{field-name}    STRUCTURE [field-attribute] ... .

        field-description-statement
        [field-description-statement] ...

END [field-name] [STRUCTURE] .
```

## B.2.3 COPY Field Description

```
[/* text */]

field-name      COPY [FROM] path-name [ALIGNED clause] .
```

## B.2.4 VARIANT Field Description

**Format 1:**

```
VARIANTS .
     VARIANT .
          field-description-statement
          [field-description-statement] ...
     END [VARIANT] .

     VARIANT .
          field-description-statement
          [field-description-statement] ...
     END [VARIANT] .
                              .
                              .
                              .
END [VARIANTS] .
```

**Format 2:**

```
VARIANTS  OF field-name .

        VARIANT   {VALUE   [IS]  }  n1 [THRU n2] [n3 [THRU n4] ] ... .
                  {VALUES [ARE]}

            field-description-statement
            [field-description-statement] ...
        END [VARIANT] .

        VARIANT   {VALUE   [IS]  }  n5 [THRU n6] [n7 [THRU n8] ] ... .
                  {VALUES [ARE]}
            field-description-statement
            [field-description-statement] ...
        END [VARIANT] .
                        .
                        .
                        .
END [VARIANTS] .
```

## B.3 General Field Attributes

```
                   {BIT       }
                   {BYTE      }
                   {WORD      }
ALIGNED [ON]       {LONGWORD  }   [BOUNDARY]
                   {QUADWORD  }
                   {OCTAWORD  }

[ROW_MAJOR   ]
[COLUMN_MAJOR]   ARRAY [n1 :] n2  [ [ n3 :] n4] ...

OCCURS n1 [ TIME [ S ] ]
    [ INDEXED FOR COBOL BY quoted-string [ ,... ] ]

OCCURS n1 TO n2 [ TIME [ S ] ] DEPENDING [ ON ] field-name
     [ INDEXED FOR COBOL BY quoted-string [ ,... ] ]

                     {complex-number       }
                     {fixed-point-number   }
INITIAL_VALUE [IS]   {floating-point-number}
                     {quoted-string        }
                     {hex-number           }
                     {octal-number         }
```

```
DATATYPE [ IS ]
 / DATE                                                                          \
|  VIRTUAL [ FIELD ]                                                              |
|  BIT        [ FIELD ]                         [ SIZE [ IS ] ] n1                |
|  UNSPECIFIED                                  [ SIZE [ IS ] ] n1 [ BYTE [ S ]]  |
|    { TEXT           }                                                           |
|    { VARYING STRING }                         [ SIZE [ IS ] ] n1 [ CHARACTER [ S ] ]
|  POINTER [ TO path-name ]                                                       |
|                                                                                 |
|    { D_FLOATING              }                [          ]      [         ]     |
|    { D_FLOATING COMPLEX      }                [          ]      [         ]     |
|    { F_FLOATING              }                [          ]      [         ]     |
|    { F_FLOATING COMPLEX      }                [ SCALE n1 ]      [ BASE n2 ]     |
|    { G_FLOATING              }                [          ]      [         ]     |
|    { G_FLOATING COMPLEX      }                [          ]      [         ]     |
|    { H_FLOATING              }                [          ]      [         ]     |
|    { H_FLOATING COMPLEX      }                [          ]      [         ]     |
|    { [ UN ] SIGNED BYTE      }                [[ SIZE [ IS ] ] n1 [ DIGIT [ S ]  ]
|    { [ UN ] SIGNED WORD      }                [          [ n2 FRACTION [ S ] ] ]]
|    { [ UN ] SIGNED LONGWORD  }                [                                 ]
|    { [ UN ] SIGNED QUADWORD  }                [SCALE n3                         ]
|    { [ UN ] SIGNED OCTAWORD  }                [BASE n4                          ]
|                                                                                 |
|    { PACKED DECIMAL             }               [ SIZE [ IS ] ] n1 [ DIGIT [ S ]  |
|    { ZONED NUMERIC              }                        [ n2 FRACTION [ S ] ] ] |
|    { UNSIGNED NUMERIC           }                                                |
|    { LEFT SEPARATE NUMERIC      }               [SCALE n3]                       |
|    { LEFT OVERPUNCHED NUMERIC   }               [BASE n4 ]                       |
|    { RIGHT SEPARATE NUMERIC     }                                                |
 \   { RIGHT OVERPUNCHED NUMERIC  }                                               /
```

## B.4 Facility-Specific Field Attributes

```
BLANK WHEN ZERO

COMPUTED BY  { DTR        }  AS quoted-string  [quoted-string] ...
             { DATATRIEVE }

CONDITION FOR COBOL [IS] condition-name

    [COBOL NAME [IS] quoted-string]

    { VALUE  [IS]  }  n1 [THRU n2] [n3 [THRU n4] ] ...
    { VALUES [ARE] }
```

```
DEFAULT_VALUE FOR { DTR        } [IS] { fixed-point-number }
                  { DATATRIEVE }      { quoted-string      }

EDIT_CODE FOR   RPG  [IS]   quoted-string

EDIT_STRING FOR { DTR        } [IS]  quoted-string
                { DATATRIEVE }

EDIT_WORD FOR   RPG  [IS]  quoted-string

JUSTIFIED RIGHT

MISSING_VALUE FOR { DTR        } [IS]  fixed-point-number
                  { DATATRIEVE }       quoted-string

           { BASIC }
NAME FOR   { COBOL }       [IS] quoted-string
           { PL/I  }
           { RPG   }

              { COBOL      }
PICTURE FOR   { DTR        }  [IS] quoted-string
              { DATATRIEVE }
              { PL/I       }

QUERY_HEADER FOR { DTR        }  [IS] quoted-string [quoted-string]...
                 { DATATRIEVE }

QUERY_NAME FOR { DTR        }  [IS] quoted-string
               { DATATRIEVE }

VALID FOR { DTR        }  IF quoted-string [quoted-string]...
          { DATATRIEVE }
```

# CDDL Error Messages C

This appendix lists the error messages generated by the CDDL compiler. After each message, there is an explanation of the message and of the action you should take to correct the problem.

For example, if you specify a negative number as the number of DIGITS in a DATATYPE clause, you receive the following message:

```
%CDDL-E-ILLNODIG, illegal number of digits
```

A CDDL error message contains the following elements:

- The facility name preceded by a percent sign (%) or a hyphen (-) and followed by a hyphen (-), such as %CDDL- or -CDDL-.
- The severity code followed by a hyphen (-). Table C-1 lists severity codes in order of increasing severity.

**Table C-1: Explanation of Severity Codes**

| Code | Severity | Explanation |
|---|---|---|
| S | Success | Indicates that your command has executed successfully. |
| I | Information | Reports on actions taken by the software. |
| W | Warning | Indicates error conditions for which the compiler can take corrective action. You should check to make sure that this action has produced the results you want; the record definition placed in the dictionary may not be correct. |
| E | Error | Indicates conditions that are not fatal, but also are not correctable by the compiler. CDDL continues syntactic and semantic checking to detect as many errors as possible, but records with errors are not stored in the CDD. Therefore, you must correct any errors and compile any faulty record definitions again. |
| F | Fatal | Indicates that the compiler cannot continue syntactic checking and cannot store any definitions in the CDD. You must correct the error and compile the record definition again. |

- The diagnostic error message name followed by a comma (,). This name identifies the message. In the following list of error messages, the messages are alphabetized by diagnostic error message name.
- The diagnostic error message. The message is a brief description of the problem. Error messages may contain string substitutions identifying the particular file names or path names in question. These string substitutions are indicated by angle brackets (< >) within a message. For example:

```
unexpected <token> deleted
```

If you received this message, CDDL would substitute the actual token for <token>.

Normally, you can correct CDDL errors by checking the source file and compiling again. IF YOU RECEIVE A FATAL ERROR INDICATING AN INTERNAL CDDL PROBLEM, OR IF YOU RECEIVE A CDD OR UTILITIES ERROR MESSAGE THAT IS NOT DOCUMENTED, you should immediately notify your system manager or the person responsible for maintaining the dictionary. If your facility has a service contract with DIGITAL, your system manager should submit a Software Performance Report (SPR) on the forms provided by DIGITAL. Be sure to include a VMS BACKUP copy of the dictionary file and a listing of the source file and commands that produced the error.

**ABBREV, found <keyword> abbreviated as <abbreviation>**

| | |
|---|---|
| Explanation: | CDDL issues this message when you abbreviate a keyword. |
| User Action: | None. |

**ALLFAILED, none of the defined records was created**

| | |
|---|---|
| Explanation: | CDDL could not compile any of the record definitions in the source file. |
| User Action: | Refer to the other messages generated by the compiler for specific errors. |

**APPROXVAL, converted value is approximately <number>**

| | |
|---|---|
| Explanation: | You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for a record description and CDD calculated an approximate value for a field's literal value within the record. |
| User Action: | None. |

**ATTMLTDEF, field attribute conflicts with earlier declaration**

| | |
|---|---|
| Explanation: | You have defined a field attribute more than once (for example, by assigning two query names to one field). |
| User Action: | Correct the error and compile the record definition again. |

**ATTNOTDEF, DATATYPE clause not specified**

| | |
|---|---|
| Explanation: | You have omitted a DATATYPE clause from an elementary field description statement. |
| User Action: | Specify a data type and compile the record definition again. |

**AUDITFAIL, error occurred while creating audit entry**

| | |
|---|---|
| Explanation: | The COPY field description statement automatically adds an entry to the history list of template records as they are copied. If the audit fails, the compiler issues this message. The record definition, however, is stored in the CDD. |
| User Action: | None. |

**BADALIGN, illegal length of alignment filler field**

| | |
|---|---|
| Explanation: | You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for an invalid record description. |
| User Action: | Use the instructions at beginning of error message list in this appendix. |

**BADCHAR, illegal character**

| | |
|---|---|
| Explanation: | The parser has encountered an illegal character in the source file. If no other error messages appear, the parser has ignored the bad character and compiled the record definition successfully. |
| User Action: | If the record compilation was unsuccessful, refer to the documentation for the accompanying error messages for explanation and suggested action. |

**BADCLAUSE, clause may prompt diagnostics when compiled with CDDL**

| | |
|---|---|
| Explanation: | You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for a record description and the operation generated a clause that might cause the CDDL compiler to issue error messages. |
| User Action: | None. |

**BADCPYLST, unable to expand copied template record source in listing**

| | |
|---|---|
| Explanation: | CDDL tried unsuccessfully to expand a template record in the listing file. |
| User Action: | Check to make sure that the template record has not been deleted from the CDD. |

**BADDATTIM, invalid date/time value**

| | |
|---|---|
| Explanation: | You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for a record containing an invalid VMS absolute date and time value. |
| User Action: | None. |

**BADDIMENS, dimension lower bound must be less than upper bound**

Explanation: You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for an invalid record description.

User Action: Use the instructions at beginning of error message list in this appendix.

**BADEXTVAL, illegal CONDITION FOR COBOL VALUES — external values ignored.**

Explanation: You used DMU/EXTRACT or CDDL/COPY_LIST to generate CDDL source for a record containing an illegal EXTERNAL value.

User Action: Correct the EXTERNAL value, recompile the record definition, and reenter DMU/EXTRACT or CDDL/COPY_LIST.

**BADFIELD, syntax error in field name**

Explanation: You have entered a field name incorrectly.

User Action: Correct the error and compile the record definition again.

**BADFORMAT, does not conform to record description protocol version 4**

Explanation: You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for a record that does not conform to version 4 of the CDD record description protocol.

User Action: Use the instructions at beginning of error message list in this appendix.

**BADFRACTS, FRACTIONS clause conflicts with DIGITS clause**

Explanation: You have specified a number of FRACTIONS greater than the number of DIGITS.

User Action: Correct the error and compile the record definition again.

**BADIDENT, illegal identifier**

| | |
|---|---|
| Explanation: | The parser encountered an apparent identifier such as a keyword or a path name, but the identifier was not legal. If no other error messages appear, the parser has ignored the illegal identifier and compiled the record definition successfully. |
| User Action: | If the record compilation was unsuccessful, refer to the documentation for the accompanying error messages for explanation and suggested action. |

**BADLENGTH, length for structure is inconsistent with prior declaration**

| | |
|---|---|
| Explanation: | You have given a STRUCTURE field an explicit data type and length, but that length is less than the total length of the STRUCTURE's subfields. |
| User Action: | Correct the error and compile the record definition again. |

**BADOCCURS, maximum number of occurrences must be greater than 0**

| | |
|---|---|
| Explanation: | You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for an invalid record description. |
| User Action: | Use the instructions at beginning of error message list in this appendix. |

**BADOVRLAY, VARIANTS field does not have an OVERLAY data type**

| | |
|---|---|
| Explanation: | You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for an invalid record description. |
| User Action: | Use the instructions at beginning of error message list in this appendix. |

**BADPROTCL, <quoted string> is not a legal dictionary object**

| | |
|---|---|
| Explanation: | You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for an invalid record description. |
| User Action: | Use the instructions at beginning of error message list in this appendix. |

**BADTAGVAR, illegal tag variable**

| | |
|---|---|
| Explanation: | You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for an invalid record description. |
| User Action: | Use the instructions at beginning of error message list in this appendix. |

**BASEGTR1, base must be greater than 1**

| | |
|---|---|
| Explanation: | You have illegally specified a number with a base of 0 or 1. |
| User Action: | Correct the error and compile the record definition again. |

**BASENOT10, scale factor not applied to non-decimal values**

| | |
|---|---|
| Explanation: | You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for a record description containing literal values for fields with a base other than ten. The CDD did not apply any scaling factor to these fields. |
| User Action: | None. |

**CANTTRANS, unsupported language feature in record <given name>**

| | |
|---|---|
| Explanation: | You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for an invalid record description. |
| User Action: | Use the instructions at beginning of error message list in this appendix. |

**CDDERROR, error encountered while creating record**

| | |
|---|---|
| Explanation: | Another CDD message identifying the problem always follows this message. |
| User Action: | Refer to the documentation for the accompanying error message for explanation and user action. |

**CDDOBJERR, CDD error at object <given name>**

| | |
|---|---|
| Explanation: | You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for an invalid record description. |
| User Action: | Use the instructions at beginning of error message list in this appendix. |

**COBINDEX, index for COBOL illegal in this context — value ignored**

Explanation: You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for a record description containing an invalid attribute value. The CDD substituted a default value for the incorrect attribute value.

User Action: None.

**COMPBYDTR, COMPUTED BY DTR clause allowed only on VIRTUAL fields**

Explanation: You have illegally included the COMPUTED BY clause in the description of a field with a data type other than VIRTUAL FIELD.

User Action: Correct the error and compile the record definition again.

**COMPLEX, value conversion error on complex data type**

Explanation: CDDL could not convert an INITIAL VALUE specification to a complex number.

User Action: Check your INITIAL VALUE specification and make sure you have specified it correctly.

**CONVERR, value conversion error**

Explanation: CDDL could not convert a value.

User Action: Check your value specification and make sure you have specified it correctly.

**COPYERROR, error encountered while copying record <quoted string>**

Explanation: CDDL was unable to copy a template record. An accompanying message explains the problem.

User Action: If the accompanying message is a CDDL message, read the explanation of that message elsewhere in this appendix. If it is a CDD message, see Appendix D of the *VAX Common Data Dictionary Utilities Reference Manual*. If it is a system message, see the VMS documentation set.

**COPYNOLIS, the /COPY_LIST qualifier conflicts with /NOLISTING qualifier**

Explanation: You specified /COPY_LIST and /NOLISTING on the same CDDL command line.

User Action: Choose either /COPY_LIST or /NOLISTING and compile the record definition again.

**COPYNONAM, error encountered while copying record**

Explanation: CDDL was unable to copy a template record. An accompanying message explains the problem.

User Action: If the accompanying message is a CDDL message, read the explanation of that message elsewhere in this appendix. If it is a CDD message, see Appendix D of the *VAX Common Data Dictionary Utilities Reference Manual*. If it is a system message, see the VMS documentation set.

**DEFNOTSET, error encountered while setting default directory**

Explanation: An error occurred when CDD tried to set CDD$DEFAULT.

User Action: Check your definition of CDD$DEFAULT; it is illegal or points to an inaccessible directory or subdictionary.

**DEFVALUE, default value substituted for invalid CDD attribute value**

Explanation: You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for a record description containing an invalid attribute value. The CDD substituted a default value for the incorrect attribute value.

User Action: None.

**DEPENDING, depending item illegal in this context — value ignored**

Explanation: You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for an invalid record description.

User Action: Use the instructions at beginning of error message list in this appendix.

**DELETETOK, unexpected <token> deleted**

| | |
|---|---|
| Explanation: | The CDDL has deleted a superfluous keyword or token. |
| User Action: | None. |

**DEPINARRY, field named in DEPENDING clause is within an array**

| | |
|---|---|
| Explanation: | You have illegally defined the field that governs the number of occurrences in an OCCURS. . .DEPENDING clause as part of an array. |
| User Action: | Edit the source file to define this field as an elementary field, not as part of an array. Then compile the record definition again. |

**DEPNOTBEF, depending item not found before field definition**

| | |
|---|---|
| Explanation: | You must define the field governing the number of occurrences in an OCCURS. . .DEPENDING clause in the record before you declare the array. |
| User Action: | Correct the error and compile the record definition again. |

**DEPNOTELM, depending item is not an elementary field**

| | |
|---|---|
| Explanation: | You have illegally defined the field governing the number of occurrences in an OCCURS. . .DEPENDING clause as part of an array or a structure field. |
| User Action: | Correct the error and compile the record definition again. |

**DEPNOTFND, field named in DEPENDING clause not found**

| | |
|---|---|
| Explanation: | You have referred to a nonexistent field in an OCCURS. . .DEPENDING clause. |
| User Action: | Edit the source file to define the field governing the number of occurrences as an elementary field, not as part of an array. Then compile the record definition again. |

**DEPNOTUNQ, depending item field name is not unique**

| | |
|---|---|
| Explanation: | The specification for the field governing the number of occurrences in an OCCURS. . .DEPENDING clause matches more than one field. |
| User Action: | Edit the source file to make the specification unique, perhaps by fully qualifying the field name. Then compile the record definition again. |

**DSCNOTTRM, DESCRIPTION clause not terminated before EOF**

| | |
|---|---|
| Explanation: | You have omitted the */ terminator from a DESCRIPTION IS clause, and the parser has read the entire source file looking for it. |
| User Action: | Edit the file and insert */ at the appropriate place. Then compile the source file again. |

**DTYPEKWRD, use keyword DATATYPE rather than TYPE**

| | |
|---|---|
| Explanation: | You used TYPE instead of DATATYPE to specify a data type. CDDL compiled the definition successfully, but warns that DATATYPE is the preferred keyword. |
| User Action: | None. |

**DUPNAME, field name is not unique**

| | |
|---|---|
| Explanation: | Two or more fields within the same STRUCTURE have the same field name. This is a legal CDDL construction, but it may cause problems. Some language compilers, for example, do not allow name duplication within a STRUCTURE. |
| User Action: | Consider giving each field within the STRUCTURE a unique name. |

**ELEMFIELD, tag variable must be an elementary field**

| | |
|---|---|
| Explanation: | You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for a record description containing a tag variable that is not an elementary field. |
| User Action: | Use the instructions at beginning of error message list in this appendix. |

**EMPTYREC, a record must contain a field description**

| | |
|---|---|
| Explanation: | You have defined a record that does not contain a field description. Every record definition must contain at least one field description. |
| User Action: | Add at least one field description to the record definition and compile it again. |

**EMPTYSTRC, a structure must contain at least one sub-field description**

| | |
|---|---|
| Explanation: | You have used a STRUCTURE field description statement but have not included a subordinate field description. Legal subordinate fields include elementary, STRUCTURE, COPY, and VARIANTS fields. |
| User Action: | Add a subordinate field description to your STRUCTURE field description and compile the record definition again. |

**EXTRANATT, extraneous CDD attribute encountered — value ignored**

| | |
|---|---|
| Explanation: | You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for an invalid record description. |
| User Action: | Use the instructions at beginning of error message list in this appendix. |

**EXTTAGVAL, a tag value cannot be expressed as a COBOL EXTERNAL value**

| | |
|---|---|
| Explanation: | You have illegally used a COBOL EXTERNAL quoted string as a tag value. |
| User Action: | Re-enter a legal tag value. |

**FLTVALIMP, decimal representation of floating point value is imprecise**

| | |
|---|---|
| Explanation: | You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for a record description containing a floating point value. When converting the floating point value to its decimal equivalent, the CDD generated only an approximate value. |
| User Action: | None. |

**HEXCONERR, conversion error — value could not be converted to hex**

| | |
|---|---|
| Explanation: | You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for an invalid record description. |
| User Action: | Use the instructions at beginning of error message list in this appendix. |

**IGNORETAG, unable to use tag variable — tag values ignored**

| | |
|---|---|
| Explanation: | You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for an invalid record description. |
| User Action: | Use the instructions at beginning of error message list in this appendix. |

**ILLBOUND, lower bound cannot exceed upper bound**

| | |
|---|---|
| Explanation: | You have specified an array with a lower bound greater than the upper bound. |
| User Action: | Correct the error and compile the record definition again. |

**ILLCOBCND, illegal COBOL condition in template record**

| | |
|---|---|
| Explanation: | You have tried to copy into a record a template record with illegal COBOL condition attributes. |
| User Action: | Correct the attributes, recompile the definition, and copy the template record again. |

**ILLDTYPE, illegal or unsupported data type — code is <number>**

| | |
|---|---|
| Explanation: | You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for an invalid record description. |
| User Action: | Use the instructions at beginning of error message list in this appendix. |

**ILLEXTVAL, duplicate INITIAL_VALUE clauses — all but one ignored**

| | |
|---|---|
| Explanation: | You used DMU/EXTRACT or CDDL/COPY_LIST to generate CDDL source for a record containing duplicate INITIAL_VALUE clauses. The EXTERNAL clause was ignored. |
| User Action: | None. |

**ILLFQNAME, illegal fully qualified name**

| | |
|---|---|
| Explanation: | You have illegally included a password in a fully qualified name. |
| User Action: | Correct the syntax and try again. |

**ILLHEXOCT, illegal hex or octal number**

| | |
|---|---|
| Explanation: | The parser encountered an apparent hexadecimal or octal number because of the prefix (%X or %O), but a legal hexadecimal or octal string did not follow. If no other error message appears, the parser has ignored the illegal number and attempted to compile the record definition. |
| User Action: | If the record compilation was unsuccessful, refer to the documentation for the accompanying error messages for explanation and suggested action. |

**ILLINDXNM, index name defined elsewhere in record definition**

| | |
|---|---|
| Explanation: | You specified the same string value as a COBOL index name and as a field name or COBOL-specific field name in a record definition. You must use a unique string value for COBOL index names. |
| User Action: | Correct the error and compile the record definition again. |

**ILLINTVAL, if scale is not 0 and base is not 10, value must be expressed in hex or octal**

| | |
|---|---|
| Explanation: | You have entered an illegal initial value. If you have used a scale other than 0 and a base other than 10, you must identify the initial value as either an octal or hexadecimal number. |
| User Action: | Correct the error and compile the record definition again. |

**ILLLITERL, does not conform to record description protocol version 4**

| | |
|---|---|
| Explanation: | You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for an invalid record description. |
| User Action: | Use the instructions at beginning of error message list in this appendix. |

**ILLMINMAX, minimum cannot exceed maximum number of occurrences**

Explanation: In an OCCURS. . .DEPENDING clause, you have specified a minimum number of occurrences greater than the maximum number.

User Action: Correct the error and compile the record definition again.

**ILLNAMCHR, a field name contains a character other than A-Z, 0-9, $, or _**

Explanation: You have used an illegal character in a field name. A field name can contain only letters, integers, $, or _.

User Action: Correct the error and compile the record definition again.

**ILLNAMSIZ, a field name's length is 0 or greater than 31**

Explanation: The number of characters in a field name must be greater than 0 but less than 32.

User Action: Correct the error and compile the record definition again.

**ILLNCHAR, illegal character found in numeric input**

Explanation: During initial value, tag value, or condition value conversion, the parser encountered an illegal character, such as a period.

User Action: Correct the error and compile the record definition again.

**ILLNODIG, illegal number of digits**

Explanation: You have specified an illegal number of DIGITS. DIGITS specifications must be greater than 0 but less than 32.

User Action: Correct the error and compile the record definition again.

**ILLNUMBER, illegal number**

Explanation: You have entered an illegal number (for example, 2E with no scale specified). If no other error message appears, the parser has ignored the illegal number and compiled the record definition successfully.

User Action: If the record compilation was unsuccessful, refer to the documentation for the accompanying error messages for explanation and suggested action.

**ILLOCCURS, number of occurrences must be greater than zero**

| | |
|---|---|
| Explanation: | In an OCCURS clause, you have specified a number of occurrences less than or equal to 0. |
| User Action: | Correct the error and compile the record definition again. |

**ILLOVPNCH, illegal overpunched character**

| | |
|---|---|
| Explanation: | You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for an invalid record description. |
| User Action: | Use the instructions at beginning of error message list in this appendix. |

**ILLQUOSTR, illegal quoted string**

| | |
|---|---|
| Explanation: | The parser has encountered an illegal string in the source file. Make sure, for example, that all open quotations are subsequently closed and that quoted strings do not exceed a single line of CDDL source text. |
| User Action: | Correct the error and compile the record definition again. |

**ILLRELVER, CDDL does not support relative version numbers**

| | |
|---|---|
| Explanation: | You specified a relative version in a path name used by CDDL, but CDDL recognizes only absolute version numbers. To specify a version other than the highest to CDDL, you must use an absolute version number. |
| User Action: | Correct the error and compile the record definition again. |

**ILLSCALE, illegal scale value**

| | |
|---|---|
| Explanation: | You have specified an illegal SCALE. SCALE specifications cannot be greater than 127 nor less than -128. |
| User Action: | Correct the error and compile the record definition again. |

**ILLVERNUM, version number cannot be used on a directory name**

| | |
|---|---|
| Explanation: | You specified a version number with a directory's given name. Because CDD does not support multiple versions of dictionary directories, you cannot use a version number with directories. |
| User Action: | Correct the error and compile the record definition again. |

**ILLVFSTRC, STRUCTUREs cannot have the virtual field datatype**

| | |
|---|---|
| Explanation: | You have tried to specify a virtual field datatype in a STRUCTURE field description statement. |
| User Action: | Modify and recompile the record definition. |

**ILLVIRFLD, STRUCTUREs cannot be VIRTUAL FIELDS — UNSPECIFIED assumed**

| | |
|---|---|
| Explanation: | You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for an invalid record description. |
| User Action: | Use the instructions at beginning of error message list in this appendix. |

**INSERTDOT, inserted "." at the end of the previous line**

| | |
|---|---|
| Explanation: | CDDL has inserted a missing period at the end of a source line. |
| User Action: | None. |

**INSERTTOK, inserted <token> before <token>**

| | |
|---|---|
| Explanation: | CDDL has inserted a missing source token. |
| User Action: | None. |

**INSMATCH, <keyword> inserted to match <keyword> inserted earlier**

| | |
|---|---|
| Explanation: | You omitted a keyword. CDDL has inserted it. |
| User Action: | None. |

**INTERROR, internal CDDL error**

| | |
|---|---|
| Explanation: | This message indicates a problem with the compiler. |
| User Action: | Use the instructions at beginning of error message list in this appendix. |

**INV1STCHR, the first character of a field name is not A-Z**

| | |
|---|---|
| Explanation: | You have begun the name of a field with an illegal character. A field name must begin with a letter. |
| User Action: | Correct your error and compile the record definition again. |

**INVALLIT, invalid literal — length is zero**

| | |
|---|---|
| Explanation: | You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for an invalid record description. |
| User Action: | Use the instructions at beginning of error message list in this appendix. |

**INVALREC, record to be copied is invalid**

| | |
|---|---|
| Explanation: | You have tried to copy an incomplete template record definition. |
| User Action: | Examine the template, supply the missing attributes, and compile the record definition again. Then you can copy the complete template record definition with the COPY field description statement. |

**INVBITLEN, invalid length for bit field**

| | |
|---|---|
| Explanation: | The legal range for the length of a field with the BIT data type is from 1 to 65,535 bits. You have specified a BIT field outside of this range. |
| User Action: | Correct the error and compile the record definition again. |

**INVDATTIM, invalid date/time string — value set to 17-NOV-1858 00:00**

| | |
|---|---|
| Explanation: | You have specified an invalid quoted string as an INITIAL, CONDITION, or TAG value whose data type is defined as DATE. When a quoted string is invalid, CDDL sets the date or time field to binary zero. The field then yields the Smithsonian base date and time, 17-NOV-1858 00:00. |
| User Action: | Correct the error and compile the record definition again. |

**INVFLDLEN, field's length is invalid**

| | |
|---|---|
| Explanation: | You have specified an invalid length for a field. Text, unspecified, and varying string fields must have a length greater than 0 but less than 65,536 characters. |
| User Action: | Correct the error and compile the record definition again. |

**INVLENGTH, field length is not multiple of 8 bits — extra bits ignored**

| | |
|---|---|
| Explanation: | You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for an invalid record description. The generated CDDL source may not accurately reflect the record description. |
| User Action: | Examine the source text of the record description to ensure the correctness of the generated CDDL source. |

**INVLSTCHR, the last character of a field name is not A-Z or 0-9**

| | |
|---|---|
| Explanation: | You have ended a field name with an illegal character. A field name must end with a letter or an integer. |
| User Action: | Correct the error and compile the record definition again. |

**JUSTIFIED, JUSTIFIED RIGHT clause allowed only with TEXT or UNSPECIFIED fields**

| | |
|---|---|
| Explanation: | You have attempted to use the JUSTIFIED RIGHT clause in the description of a field whose data type is something other than TEXT or UNSPECIFIED. |
| User Action: | Correct the error and compile the record definition again. |

**KEYWDREPL, replaced <keyword> with <keyword>**

| | |
|---|---|
| Explanation: | You have used an incorrect keyword, and CDDL has replaced it with the correct one. |
| User Action: | None. |

**LANGNAMES, language specific names cannot be used with unnamed fields**

| | |
|---|---|
| Explanation: | You have attempted to specify a language-specific name for an unnamed field. Only named fields can have language-specific names. |
| User Action: | Correct the error and compile the record definition again. |

**LINETRUNC, output buffer overflow — line truncated**

Explanation: You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for a record description, and a line of the generated source exceeds 255 characters.

User Action: Examine the source text of the record description to learn what should be in the truncated line.

**MLTJUSTRT, JUSTIFIED RIGHT clause specified more than once**

Explanation: You have specified JUSTIFIED RIGHT more than once for a single field.

User Action: Correct the error and compile the record definition again.

**MULTARRAY, array clause specified more than once**

Explanation: You have specified more than one OCCURS. . .DEPENDING, ARRAY, or OCCURS clause for a single field.

User Action: Correct the error and compile the record definition again.

**MULTBLANK, BLANK WHEN ZERO clause specified more than once**

Explanation: You have defined a single field as BLANK WHEN ZERO more than once.

User Action: Correct the error and compile the record definition again.

**MULTDTYPE, DATATYPE clause specified more than once**

Explanation: You have specified more than one data type clause for a single field.

User Action: Correct the error and compile the record definition again.

**MULTSCALE, SCALE clause conflicts with earlier declaration**

Explanation: Within a single DATATYPE clause you have specified SCALE more than once, or you have specified both SCALE and FRACTIONS and their values are inconsistent.

User Action: Correct the error and compile the record definition again.

**MULTVALIS, VALUE IS clause specified more than once**

| | |
|---|---|
| Explanation: | You have illegally specified more than one VALUE IS or VALUES ARE clause within one VARIANT field description. |
| User Action: | Correct the error and compile the record definition again. |

**NEWVERCRE, new version <given name> created in the CDD**

| | |
|---|---|
| Explanation: | You created an additional version of a dictionary object. |
| User Action: | None. |

**NODEPITEM, depending item fieldname does not exist within the record**

| | |
|---|---|
| Explanation: | You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for an invalid record description. |
| User Action: | Use the instructions at beginning of error message list in this appendix. |

**NODTYPATT, field's data type attribute is missing**

| | |
|---|---|
| Explanation: | You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for an invalid record description. |
| User Action: | Use the instructions at beginning of error message list in this appendix. |

**NOLENGATT, field's length attribute is missing**

| | |
|---|---|
| Explanation: | You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for an invalid record description. |
| User Action: | Use the instructions at beginning of error message list in this appendix. |

**NOMATCH, name mismatch in "END" statement**

| | |
|---|---|
| Explanation: | The name you specified in the END statement does not match the name you specified in the DEFINE or STRUCTURE statement. |
| User Action: | Correct the error and compile the record definition again. |

**NOROOTATT, does not conform to record description protocol version 4**

Explanation: You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for an invalid record description.

User Action: Use the instructions at beginning of error message list in this appendix.

**NOTCDDL, record <quoted string> not created by CDDL, cannot be recompiled**

Explanation: You have tried to recompile a definition created by a facility (such as VAX DATATRIEVE) other than CDDL. CDDL does not support this operation.

User Action: None.

**NOTCDDREC, object is not a record**

Explanation: With the COPY field description statement, you can copy dictionary objects only if the CDD type is CDD$RECORD. You tried to use COPY with another type.

User Action: None.

**NOTCOMPLX, complex literal must be used with complex data type**

Explanation: You have incorrectly attempted to specify a complex number as the initial, tag, or condition value of a field whose data type is not one of the FLOATING COMPLEX data types.

User Action: Correct the error and compile the record definition again.

**NOTFIXED, fixed point literal not allowed in this context**

Explanation: You have specified a fixed point number as the initial, tag, or condition value of a field whose data type cannot be expressed as a fixed point number.

User Action: Edit the source file to change the initial, tag, or condition value or to define a compatible data type. Then compile the record definition again.

**NOTFLOAT, floating point literal not allowed in this context**

| | |
|---|---|
| Explanation: | You have specified a floating point number as the initial, tag, or condition value of a field whose data type cannot be expressed as a floating point number. |
| User Action: | Edit the source file to change the initial, tag, or condition value or to define a compatible data type. Then compile the record definition again. |

**NOTRECDEF, object <given name> is not a record definition**

| | |
|---|---|
| Explanation: | You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for a dictionary object that is not a CDD$RECORD. |
| User Action: | Be sure the path name you specify identifies a CDD record description and enter the command again. |

**NOTRECORD, object <quoted string> is not a record**

| | |
|---|---|
| Explanation: | The CDDL/RECOMPILE command allows you to recompile objects only if the type of the object is CDD$RECORD. You have tried to use the CDDL/RECOMPILE command with another type. |
| User Action: | None. |

**NOTREPLCE, object to be replaced is not a record**

| | |
|---|---|
| Explanation: | The /REPLACE qualifier to the compile command allows you to replace dictionary objects only if the type is CDD$RECORD. You tried to use /REPLACE with another type. |
| User Action: | None. |

**NOTSTRING, string literal not allowed in this context**

| | |
|---|---|
| Explanation: | You have specified an ASCII string as the initial, tag, or condition value of a field whose data type cannot be expressed as an ASCII string. |
| User Action: | Edit the source file to change the initial, tag, or condition value or to specify a compatible data type. Then compile the record definition again. |

**NUMCONERR, conversion error — number could not be converted to decimal**

| | |
|---|---|
| Explanation: | You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for a record description containing a value the CDD could not convert to a decimal equivalent. |
| User Action: | None. |

**OBJEXTUVS, another version already exists — record not created**

| | |
|---|---|
| Explanation: | You tried to create a record definition with the same path name as an already existing dictionary object, but you did not specify /VERSION. |
| User Action: | Choose a different given name, or use the /VERSION qualifier. |

**OBJEXTVNS, object already exists — not superseded**

| | |
|---|---|
| Explanation: | You tried to create a record definition with the same path name and version number as an already existing dictionary object. |
| User Action: | Choose a different given name or version number for the object. |

**OVERFLOW, value conversion overflow**

| | |
|---|---|
| Explanation: | You have specified a numeric value that is too large for the defined field. |
| User Action: | Specify a smaller value, or redefine the length of the field and recompile the record. |

**PSSWDSYN, syntax error in password specification**

| | |
|---|---|
| Explanation: | You supplied an invalid password specification. Passwords contain from 1 to 64 printable ASCII characters, including space and tab, but excluding open parenthesis [(], close parenthesis [)], and period [.]. |
| User Action: | Correct the syntax and compile the record definition again. |

**QUERYNAME, QUERYNAME clause cannot be used for unnamed fields**

Explanation: You have attempted to specify a VAX DATATRIEVE QUERY_NAME for an unnamed field. You can specify query names only for named fields.

User Action: Correct the error and compile the record definition again.

**RECNOTCRE, error in record definition — record not created**

Explanation: The record defined in the source file record was not created because of an error or errors.

User Action: Use the specific error messages you received to edit the source file.

**RECNOTFND, record to be replaced not found**

Explanation: The /REPLACE qualifier to the COMPILE command allows you to replace existing record definitions. This warning indicates that you tried to replace a record definition that does not exist. The new definition, however, has been stored in the CDD.

User Action: None.

**RECOMPACL, /VERSION qualifier must be specified to use /[NO]ACL qualifier**

Explanation: In order to use either /ACL or /NOACL with CDDL/RECOMPILE, you must use the version qualifier. Otherwise, the newly created record definition copies the access control list of the definition it replaces.

User Action: If you want the newly created object to have no access control list, use the /VERSION qualifier.

**RECOMPERR, error encountered while recompiling record <quoted string>**

Explanation: An error occurred when CDDL was trying to recompile a record. This message is followed by another that explains the problem.

User Action: Refer to the documentation for the accompanying error message for explanation and suggested action.

**RECOMPVER, record <given name> recompiled and <given name> created in the CDD**

Explanation: CDDL recompiled an existing record description, creating an additional version of it.

User Action: None.

**RECORDCRE, record <given name> created in the CDD**

Explanation: CDDL created a record description and inserted it in the dictionary.

User Action: None.

**RELVERNUM, relative version number on path name is being ignored**

Explanation: You included a relative version number as part of a path name, but CDDL recognizes only absolute version numbers. CDDL created the record definition with version number one higher than the previous highest.

User Action: If you want the version number you specified, you can change it with the DMU RENAME command.

**REPLACACL, the /[NO]ACL qualifier conflicts with the /REPLACE qualifier**

Explanation: When you use CDDL/REPLACE, the new object always copies the access control list of the object it replaces. You cannot specify /ACL or /NOACL with CDDL/REPLACE.

User Action: Reenter the command without the /NOACL qualifier.

**RMSERROR, RMS error in file <quoted string>**

Explanation: This message is followed by an RMS message explaining why CDDL was unable to open, close, read, or write a file.

User Action: Refer to the *VAX/VMS System Message and Recovery Procedures Manual*.

**SOMEFAIL, some of the defined records were not created**

Explanation: Some of the records defined in your source file were not compiled because of an error or errors.

User Action: Remove the successful definitions from the file, correct errors in those remaining, and compile the file again.

**SPELLCORR, identifier <keyword> replaced with <keyword> by the spelling corrector**

Explanation: You have misspelled a keyword, but CDDL has corrected the spelling.

User Action: None.

**STRCONERR, conversion error — string could not be converted to text**

Explanation: You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for a record description containing an invalid VMS absolute date and time value. The CDD displays the value in hexadecimal, not as an ASCII date and time specification.

User Action: None.

**STRMERGED, merged <string> and <string> to form <keyword>**

Explanation: You have left a space in the middle of a keyword, and CDDL has joined the two portions.

User Action: None.

**STRUCKWRD, use keyword STRUCTURE rather than GROUP**

Explanation: You used GROUP instead of STRUCTURE to specify a structure field. The CDDL compiled the definition successfully, but warns that STRUCTURE is the preferred keyword.

User Action: None.

**SYNTAX, syntax error near <quoted string>**

| | |
|---|---|
| Explanation: | You have a syntax error in the CDDL command line. An accompanying message explains the problem. |
| User Action: | Refer to the documentation for the accompanying error message for explanation and suggested action. |

**TAGINARRY, field named in tag variable clause is within an array**

| | |
|---|---|
| Explanation: | You have incorrectly defined the tag variable as part of an array. |
| User Action: | Edit the source file to make sure the tag variable is defined as an elementary field that is not part of an array. Then compile the record definition again. |

**TAGNOTALL, VALUE IS clause requires tag variable clause**

| | |
|---|---|
| Explanation: | You have not specified a tag variable in the VARIANTS statement, but you have specified a tag value for one of the variants. |
| User Action: | Edit the source file to make usage of the tag variable and the tag value consistent. Then, compile the record definition again. |

**TAGNOTBEF, tag variable not found before field definition**

| | |
|---|---|
| Explanation: | You must define the tag variable field in the record before you include the VARIANTS OF statement. |
| User Action: | Correct the error and compile the record definition again. |

**TAGNOTELM, tag variable is not an elementary field**

| | |
|---|---|
| Explanation: | You have either defined the tag variable as part of an array or you have defined it as a field that is not elementary. The tag variable field must be elementary and cannot be part of an array. |
| User Action: | Correct the error and compile the record definition again. |

**TAGNOTFND, field named in tag variable clause not found**

| | |
|---|---|
| Explanation: | You have referred to a nonexistent field in a tag variable clause. |
| User Action: | Edit the source file to make sure the tag variable is defined as an elementary field, not as part of an array. Then compile the record definition again. |

**TAGNOTUNQ, tag variable field name is not unique**

Explanation: Your tag variable specification matches more than one field in the record.

User Action: Edit the source file to make the specification unique, perhaps by fully qualifying the field name. Then compile the record definition again.

**TAGREQUIR, tag variable clause required**

Explanation: If you specify a tag variable with a VARIANTS OF statement, each variant must declare a value or values with the VALUE IS or VALUES ARE clauses. You have omitted one or more of these tag value clauses.

User Action: Correct the error and compile the record definition again.

**TOOFEWPAR, too few or missing parameter**

Explanation: You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for a dictionary object that is not a CDD$RECORD.

User Action: Be sure the path name you specify identifies a CDD record description and enter the command again.

**TOOMANPAR, too many or conflicting parameters**

Explanation: You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for a dictionary object that is not a CDD$RECORD.

User Action: Be sure the path name you specify identifies a CDD record description and enter the command again.

**TOOMANVAL, no more than two values permitted in a range**

Explanation: You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for an invalid record description.

User Action: Use the instructions at beginning of error message list in this appendix.

**TRUNCATIO, value conversion truncation error**

| | |
|---|---|
| Explanation: | The value you specified for an initial, tag, or condition value was too large to fit in the field as defined. |
| User Action: | Specify a smaller value, or redefine the field and recompile the record. |

**UNDERFLOW, value conversion underflow**

| | |
|---|---|
| Explanation: | The value you specified for an initial, tag, or condition value was too small to fit in the field as defined. |
| User Action: | Specify a different value, or redefine the field and recompile the record. |

**UNEXPTOK, found <token> when expecting one of <token>**

| | |
|---|---|
| Explanation: | CDDL is unable to determine what you were defining when it encountered a syntax error. |
| User Action: | Correct the error and compile the record definition again. |

**UNPRINTCH, text string contains one or more unprintable characters**

| | |
|---|---|
| Explanation: | You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for a record description containing an unprintable text string. The CDD displays the string's hexadecimal equivalent. |
| User Action: | None. |

**UNSPROTCL, object <object name> does not conform to record description protocol version 4**

| | |
|---|---|
| Explanation: | This object does not conform to the protocol used to describe CDD record definitions. |
| User Action: | Use the instructions at beginning of error message list in this appendix. |

**VALCONERR, conversion error — value could not be converted**

| | |
|---|---|
| Explanation: | You used DMU EXTRACT/RECORD or CDDL/COPY_LIST to generate CDDL source for an invalid record description. |
| User Action: | Use the instructions at beginning of error message list in this appendix. |

**VALUEOVFL, value overflow, value set to zero**

| | |
|---|---|
| Explanation: | An integer you specified in the source file is too large to fit into a 32-bit longword. CDDL converts all integers to longwords. |
| User Action: | None. |

**VALTRUNC, overflow occurred during conversion — value truncated**

| | |
|---|---|
| Explanation: | You have specified a value that is too large for the defined field. The hexadecimal or octal number has been truncated. |
| User Action: | None. |

**VERREPLAC, the /VERSION qualifier conflicts with the /REPLACE qualifier**

| | |
|---|---|
| Explanation: | You specified /VERSION and /REPLACE on the same CDDL command line. |
| User Action: | Correct the error and enter the command again. |

**VIRTFIELD, VIRTUAL field cannot have INITIAL or CONDITION value clauses**

| | |
|---|---|
| Explanation: | You have assigned an initial or condition value to a VIRTUAL FIELD. Virtual fields take up no space in the record and cannot store values. They are valid only when used with the COMPUTED BY DATATRIEVE clause. |
| User Action: | Correct the error and compile the record definition again. |

# CDDL Reserved Words D

Certain CDDL keywords are reserved. You cannot use CDDL reserved words in field names or a path name within a CDDL source file. CDDL reserved words are:

END
FROM
GROUP
IS
ON
RECORD
SIZE
STRUCTURE
VARIANTS

Although you cannot use these reserved words as path names in CDDL source files, you can use them as given names within path names. For example, the following statements are legal within CDDL source files because they use CDDL reserved words only as *parts* of longer path names:

```
DEFINE RECORD CDD$TOP.TEST.STRUCTURE.

END PAYROLL.SIZE.RECORD.

COPY FROM ON.ON.ON.
```

The following statements are illegal within CDDL source files because they use CDDL reserved words as the *entire* path name:

```
DEFINE RECORD FROM.

END SIZE.

COPY FROM ON.
```

# Additional CDDL Notes E

## E.1 Support of the VAX Language-Sensitive Editors (LSE)

Version 3.3 and later of CDD supports the VAX Language-Sensitive Editor (LSE). If the VAX Language-Sensitive Editor is installed on your system, you can use this feature to help write, compile, and debug CDDL definitions. The CDDL Language-Sensitive Editor provides templates and menus to walk you through CDDL options and syntax. It is especially useful for users unfamiliar with CDDL.

To invoke the Language-Sensitive Editor, type LSE at the DCL prompt. The following command, for example, creates a file ADDRESS.CDDL and displays a CDDL record definition template to guide you through the process of describing a CDDL record:

```
$ LSE ADDRESS.CDDL
```

When LSE compiles your source definition, it expects a file type of .CDDL. The CDDL compiler now recognizes both file types .CDDL and .DDL.

## E.2 /DIAGNOSTICS Qualifier for CDDL Command

The /DIAGNOSTICS qualifier with the CDDL command creates a diagnostics file that lists errors occurring during compilation. /DIAGNOSTICS is designed for use from the LSE environment. /DIAGNOSTICS lists errors in a file that has the default name of your definition file and the extension .DIA. The diagnostic file is reserved for use by DIGITAL. LSE uses the diagnostic file to display diagnostic messages and to position the cursor on the line and column where a source error exists.

You cannot use /DIAGNOSTICS with CDDL/RECOMPILE.

For complete information on using LSE, see the *VAX Language-Sensitive Editor User's Guide*.

## E.3 The CDDL ALIGNED Clause

Be careful when you use the CDDL ALIGNED clause.

- You should not use the ALIGNED clause in template records. When CDDL stores the template record, the position of an aligned field is fixed within the record and is not changed when the record is copied into another record definition. Therefore, the newly created field may not align properly in the new record definition.
- Records created with the ALIGNED clause using previous versions of CDDL may not have aligned fields properly. CDD Version 3.1 corrected this alignment problem. However, if you recompile the records using the ALIGNED clause, data already stored will no longer match the recompiled data definition.

# Index

In this index, a page number followed by a "t" indicates a table reference. A page number followed by an "e" indicates an example reference. A page number followed by an "f" indicates a figure reference.

# How to Order Additional Documentation

## Technical Support

If you need help deciding which documentation best meets your needs, call 800-343-4040 before placing your electronic, telephone, or direct mail order.

## Electronic Orders

To place an order at the Electronic Store, dial 800-DEC-DEMO (800-332-3366) using a 1200- or 2400-baud modem. If you need assistance using the Electronic Store, call 800-DIGITAL (800-344-4825).

## Telephone and Direct Mail Orders

| Your Location | Call | Contact |
|---|---|---|
| Continental USA, Alaska, or Hawaii | 800-DIGITAL | Digital Equipment Corporation<br>P.O. Box CS2008<br>Nashua, New Hampshire 03061 |
| Puerto Rico | 809-754-7575 | Local DIGITAL subsidiary |
| Canada | 800-267-6215 | Digital Equipment of Canada<br>Attn: DECdirect Operations KAO2/2<br>P.O. Box 13000<br>100 Herzberg Road<br>Kanata, Ontario, Canada K2K 2A6 |
| International | ——— | Local DIGITAL subsidiary or approved distributor |
| Internal[1] | ——— | SDC Order Processing - WMO/E15<br>*or*<br>Software Distribution Center<br>Digital Equipment Corporation<br>Westminster, Massachusetts 01473 |

[1]For internal orders, you must submit an Internal Software Order Form (EN-01740-07).

## Reader's Comments

**VAX Common Data Dictionary**
**Data Definition Language**
**Reference Manual**
AA–K085D–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

| **I rate this manual's:** | Excellent | Good | Fair | Poor |
|---|---|---|---|---|
| Accuracy (software works as manual says) | ☐ | ☐ | ☐ | ☐ |
| Completeness (enough information) | ☐ | ☐ | ☐ | ☐ |
| Clarity (easy to understand) | ☐ | ☐ | ☐ | ☐ |
| Organization (structure of subject matter) | ☐ | ☐ | ☐ | ☐ |
| Figures (useful) | ☐ | ☐ | ☐ | ☐ |
| Examples (useful) | ☐ | ☐ | ☐ | ☐ |
| Index (ability to find topic) | ☐ | ☐ | ☐ | ☐ |
| Page layout (easy to find information) | ☐ | ☐ | ☐ | ☐ |

I would like to see more/less ______________________________

______________________________

What I like best about this manual is ______________________________

______________________________

What I like least about this manual is ______________________________

______________________________

I found the following errors in this manual:

Page Description

______ ______________________________

______ ______________________________

______ ______________________________

Additional comments or suggestions to improve this manual:

______________________________

______________________________

______________________________

I am using **Version** ______ of the software this manual describes.

Name/Title ______________________________ Dept. ______________

Company ______________________________ Date ______________

Mailing Address ______________________________

______________________________ Phone ______________

Do Not Tear - Fold Here and Tape

digital™

No Postage
Necessary
if Mailed
in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01-3/J35
110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

Do Not Tear - Fold Here

Cut Along Dotted Line

## Reader's Comments

**VAX Common Data Dictionary**
**Data Definition Language**
**Reference Manual**
AA-K085D-TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

| **I rate this manual's:** | Excellent | Good | Fair | Poor |
|---|---|---|---|---|
| Accuracy (software works as manual says) | ☐ | ☐ | ☐ | ☐ |
| Completeness (enough information) | ☐ | ☐ | ☐ | ☐ |
| Clarity (easy to understand) | ☐ | ☐ | ☐ | ☐ |
| Organization (structure of subject matter) | ☐ | ☐ | ☐ | ☐ |
| Figures (useful) | ☐ | ☐ | ☐ | ☐ |
| Examples (useful) | ☐ | ☐ | ☐ | ☐ |
| Index (ability to find topic) | ☐ | ☐ | ☐ | ☐ |
| Page layout (easy to find information) | ☐ | ☐ | ☐ | ☐ |

I would like to see more/less ____________________________________________

______________________________________________________________________

What I like best about this manual is ____________________________________

______________________________________________________________________

What I like least about this manual is ___________________________________

______________________________________________________________________

I found the following errors in this manual:

| Page | Description |
|---|---|
| ____ | ____________________________________________ |
| ____ | ____________________________________________ |
| ____ | ____________________________________________ |

Additional comments or suggestions to improve this manual:

______________________________________________________________________

______________________________________________________________________

______________________________________________________________________

I am using **Version** ______ of the software this manual describes.

Name/Title ______________________________ Dept. ______________

Company ____________________________________ Date ____________

Mailing Address ______________________________________________

_____________________________________ Phone ______________

Do Not Tear - Fold Here and Tape

digital™

No Postage
Necessary
if Mailed
in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01-3/J35
110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

Do Not Tear - Fold Here

Cut Along Dotted Line

## Reader's Survey

**VAX Common Data Dictionary**
**Data Definition Language**
**Reference Manual**
AA-K085D-TE

1. How useful are the following methods for finding information in this manual?

| | Most | Very | Moderately | Not Very | Not at All |
|---|---|---|---|---|---|
| Table of contents | ☐ | ☐ | ☐ | ☐ | ☐ |
| Divider pages (if applicable) | ☐ | ☐ | ☐ | ☐ | ☐ |
| Index (circle: book or master) | ☐ | ☐ | ☐ | ☐ | ☐ |
| Other (specify) ______________ | ☐ | ☐ | ☐ | ☐ | ☐ |

2. What feature do you most want to see improved in this manual? Why?

______________________________________________

3. How helpful are these sources when you use the software this manual describes?

| | Most | Very | Moderately | Not Very | Not at All |
|---|---|---|---|---|---|
| Handbook or user's guide | ☐ | ☐ | ☐ | ☐ | ☐ |
| Introduction or overview | ☐ | ☐ | ☐ | ☐ | ☐ |
| Reference manual | ☐ | ☐ | ☐ | ☐ | ☐ |
| Quick reference guide | ☐ | ☐ | ☐ | ☐ | ☐ |
| Online help | ☐ | ☐ | ☐ | ☐ | ☐ |
| Online tutorial (if available) | ☐ | ☐ | ☐ | ☐ | ☐ |
| Other: colleague, telephone support services (specify) ______________ | ☐ | ☐ | ☐ | ☐ | ☐ |

4. What business tasks are you using the software described by this manual to solve (for example: billing, funds transfer, report writing)?

______________________________________________

5. Please estimate, if you can, how long the following VAX Information Architecture products have been used at your site:

| | | |
|---|---|---|
| VAX ACMS ________ | VAX CDD/Plus ________ | VAX DATATRIEVE ________ |
| VAX Data Distributor ________ | VAX DBMS ________ | VAX RALLY ________ |
| VAX Rdb/VMS ________ | VAX SQL ________ | VAX TEAMDATA ________ |
| VAX TDMS ________ | VIDA with IDMS/R ________ | |

6. This release of VAX Information Architecture documentation uses a 7x9 format for quick reference guides. Do you prefer such books in a 7x9 or a 4x8 pocket guide format? ______________________________

Thank you for your assistance.

May we contact you at work for further information? ☐ Yes ☐ No

Name ______________________________ Dept. ______________

Company ______________________________ Date ______________

Mailing Address ______________________________________

______________________________ Phone ______________

Do Not Tear - Fold Here and Tape

digital™

No Postage
Necessary
if Mailed
in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK02-2/N53
110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

Do Not Tear - Fold Here

Cut Along Dotted Line

DECLIT AA VAX K085D

VAX common data dictionary
data definition language
reference manual

DECLIT AA VAX K085D

VAX common data dictionary
data definition language
reference manual

| | | | |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

digital

digital

*0190133*

Printed in U.S.A.

X

# C.A.S. Delivery System

## System Manager's Guide

digital

software

94-003|049/14

# C.A.S. Delivery System
# System Manager's Guide

Order Number AA-K767C-TE

To order additional copies of this document, contact the Software Distribution Center, Digital Equipment Corporation, Maynard, Massachusetts 01754

First Printing, July 1982
Second Printing, July 1984
Third Printing, December 1985

| | | |
|---|---|---|
| DAL | Edusystem | RSTS |
| DEC | GIGI | RSX |
| DECnet | PDP | UNIBUS |
| DECtalk | PDT | VAX |
| DECwriter | ReGIS | VMS |
| DIBOL | RENAISSANCE | VT |
| | | digital™ |

8/6/0

# Contents

# Contents

## Tables

# Preface

Two types of system management functions are required for the Courseware Authoring System Delivery System (C.A.S. Delivery System).

The first type of system management function is performed by the VAX/VMS system manager and requires system privileges. The VAX/VMS system manager uses the AUTHORIZE utility to create accounts, and edits the SYSTARTUP.COM file to assign the system logical names needed by the C.A.S. Delivery System.

The second type of system management function is performed by the C.A.S. system manager using the C.A.S. Delivery System. The C.A.S. system manager, who does not require system privileges, creates and deletes groups, registers an instructor in a new group, and updates the information pertaining to all available lessons.

This manual explains both types of functions. Chapter 1 presents an overview of C.A.S. Chapter 2 details the considerations for assigning users to groups and for setting up accounts and directories for both users and groups. Chapter 3 explains C.A.S. files and file protections. Chapter 4 explains how the C.A.S. system manager uses the system.

## RELATED DOCUMENTS

This manual assumes that the reader is familiar with the *VAX/VMS System Management and Operations Guide* Order No. AA-M547A-TE.

The documentation set for C.A.S. consists of the following documents.

*C.A.S. Delivery System User's Guide*
Order No. AA-K764C-TE

*C.A.S. Delivery System Student Guide*
Order No. AV-K765C-TE

*VAX DAL Author's Guide*
Order No. AA-K763C-TE

*VAX DAL Pocket Reference Guide*
Order No. AV-DB13B-TE

*VAX DAL Reference Manual*
Order No. AA-K768C-TE

# 1

# Introduction

# 1 Introduction

This chapter describes the structure and operation of the Courseware Authoring System Delivery System (C.A.S. Delivery System). It also explains the relationship between the C.A.S. Delivery System and its companion software product, the Digital Authoring Language (VAX DAL).

## The C.A.S. Delivery System

The C.A.S. Delivery System consists of a software interface and a series of management reports. The delivery system, which will be referred to from this point as C.A.S., provides the capabilities needed to present computer-aided instruction (CAI) to students and track student progress. C.A.S. can deliver lessons written in any programming language.

## The DIGITAL Authoring Language

Related to C.A.S. is a companion software product, the DIGITAL Authoring Language (VAX DAL). VAX DAL is a structured, high-level language with features specifically designed for writing computer-aided instruction. A VAX DAL author writes source code with any VAX/VMS editor, compiles lessons with the VAX DAL compiler, and links them with the VAX/VMS linker.

Lessons written in VAX DAL can be executed from VAX/VMS command level, but some features of the language are fully implemented only when students execute these lessons as assignments through C.A.S.

## C.A.S. USERS

There are four types of C.A.S. users defined by their functions within the system.

- The C.A.S. system manager
- Authors
- Instructors
- Students

### The C.A.S. System Manager

The C.A.S. system manager has overall administrative responsibility for the C.A.S. system. With the VAX/VMS system manager, the C.A.S. system manager determines what VAX/VMS accounts and directories are needed for users and groups. After installation, the C.A.S. system manager is responsible for creating and deleting groups and for registering an instructor in each group. In addition, the system manager can perform the functions of the instructor for any group except for making and editing group assignments.

### Authors

Authors control availability of lessons within the C.A.S. Delivery System by publishing, republishing, and unpublishing lessons.

When authors publish completed lessons using C.A.S., the lessons become available to other C.A.S. users. Authors can then produce reports on lesson performance that assist them in lesson development.

### Instructors

Instructors have administrative responsibility for their groups. For a group of students, the instructor's responsibilities include registering students, assigning lessons to the group, and producing reports of student progress. For a group of authors, the instructor's only responsibility is registering authors in the group.

### Students

Students are responsible for taking lessons.

## C.A.S. GROUPS

Every C.A.S. user must be a member of a group; each group must include an instructor. Depending on site requirements, the other members of a group can be all students, all authors, or a mixture of students, authors, and instructors. Groups of students generally correspond to class groups. That is, students are grouped because they are studying the same subject and using the same C.A.S. lessons.

One group, the group defined by the logical name CAS$SYSGROUP, is required. The C.A.S. system manager is an instructor in CAS$SYSGROUP, and can register other instructors in the group. All instructors in this group have the same privileges as the system manager.

Chapter 2 discusses the considerations for grouping users.

## C.A.S. FILES

In addition to storing the interface, the reports, and the executable images of the compiler, C.A.S. uses a number of data files to store personal information about users, as well as information about groups,student performance, and the status of assignments.

Chapter 3 contains more information about the contents and locations of C.A.S. files.

## C.A.S. DATA BASES

C.A.S. includes two data bases, one for groups and one for lessons.

The group data base contains subdirectories for group and student files. It also includes the GRPLST.DAT file and the .INF file for the system group.

During the installation procedure, the C.A.S. system manager is prompted to specify a device and a top-level directory for storing the group data base. The logical name CAS$GROUPS points to the device and directory specified. The logical name CAS$GRPDEV also points to this device and directory specification, but as a rooted directory, which allows for automatic creation of subdirectories. Each time the C.A.S. system manager creates a new group with the CREATE option, a separate subdirectory of CAS$GROUPS is generated automatically on CAS$GRPDEV to store files for the new group. When a group is deleted, its subdirectory is automatically removed.

The lesson data base contains a subdirectory for each lesson published, as well as the LESSONAVL.DAT file. CAS$LESSONS points to the the top-level directory that contains the lesson data base.

When an author publishes a lesson, the system creates a subdirectory of CAS$LESSONS for the lesson. This subdirectory contains the executable image of the lesson and any auxiliary files specified by the author at the time of publication. The subdirectory also contains restart files and the student statistics file. Depending on the instructions in the lesson, it may also include a permanent variable file and a log file. The subdirectory is the default directory for any other files used by the lesson.

## PROTECTION OF DATA

C.A.S. data files and lessons contain two types of potentially sensitive information:

- Information useful to students, such as the answers to questions in the lessons
- Information that is by nature confidential, such as scoring information

Access to C.A.S. is restricted, as is access from C.A.S. to executable lessons and to the reports that read data files. C.A.S. files are protected from other VAX/VMS users with VAX/VMS file protections. C.A.S. users also can be prevented from accessing VAX/VMS command level.

### Access to C.A.S.

Access to C.A.S. requires a VAX/VMS user name and password. Depending on a site's system configuration, users can also be required to enter one or more of the following:

- Group
- C.A.S. name
- C.A.S. password

Users are identified by group and C.A.S. name. Each C.A.S. name must be unique within the group.

When a user accesses C.A.S., a flag is set to prevent another user from accessing C.A.S. at the same time using the same group and C.A.S. name. (See the description of the group information file in Chapter 3 for the location of the flag.) The restriction protects the integrity of the data files.

Command procedures can also place VAX/VMS users directly in C.A.S. groups after they log in, and log them out when they exit from C.A.S.

## Access to C.A.S. Functions

Access to lessons and to reports that read the C.A.S. data files is limited by group and by user type.

After an author publishes a lesson, the lesson name is displayed on a browse menu. Students, instructors, and authors can execute the lesson by selecting it from the browse menu.

Lessons can be restricted to one group when they are published. The C.A.S. system manager has access to the names of restricted lessons and the groups to which they are restricted. The system manager can execute any lesson. Instructors, authors, and students do not have access to the names of lessons restricted to groups other than their own. Instructors, authors, and students in other groups cannot execute restricted lessons.

Once lessons are assigned to a group, students in that group execute them only as assignments. Assigned lessons are scored the first time they are executed. Students can execute completed assignments again, but scores are not changed.

The C.A.S. system manager can access any report for any group. Only an instructor can access reports that show scores for individual students; these reports show only those students in the instructor's group.

## Access to C.A.S. Files

Access to C.A.S. files from VAX/VMS command level is restricted by the VAX/VMS file protections. VAX/VMS users with system accounts have full access to the files. For most files, other VAX/VMS users have no access from command level.

## C.A.S. LOGICAL NAMES

C.A.S. uses the following system logical names:

| | |
|---|---|
| CAS$SYSGROUP | Defines the C.A.S. system group. |
| CAS$GROUPS | Points to the top level directory for group files under which the directories for groups are created and maintained. |
| CAS$GRPDEV | Points to the same device and directory as CAS$GROUPS, but as a rooted directory that facilitates the creation of group subdirectories. |
| CAS$LESSONS | Points to the top level directory for lessons under which the directories for individual lessons are created and maintained. |
| SYS$SYSTEM | Points to a VMS system directory where the CAS.COM file and the C.A.S. report files reside. |

Two additional logical names are defined at VAX/VMS system startup for all C.A.S. users. These can be defined by the user in log-in files, and are as follows:

| | |
|---|---|
| CAS$GROUP | Identifies a C.A.S. group. If this logical name is defined, anyone accessing C.A.S. from this account is automatically placed in this group. |
| CAS$EXIT_FILE | Identifies a file of terminal control sequences created by the system manager or another user. If this logical name is defined, C.A.S. uses it to open the exit file to reset a user's terminal when the user exits from C.A.S. |

# 2
# C.A.S. Groups

# 2 C.A.S. Groups

This chapter discusses C.A.S. groups. It explains the possible combinations of users in groups, the relationship between C.A.S. groups and VAX/VMS accounts and directories, and the factors that affect the arrangement of C.A.S. groups.

## C.A.S. GROUPS

Every C.A.S. user is a member of a group. The group is the unit of control for the following system features.

- Lesson assignments. Lessons are assigned to the group, not to individual students.
- Reports. Data for reports is drawn from one group at a time. Reports show performance information for the students in the group, the lessons available to the group, the .INF file information for members of the group, and so on.
- Lesson restrictions. Lessons can be restricted so that the members of only one group can access them.
- The group environment. The instructor in each group determines whether passwords are required from group members and whether the members' VAX/VMS user names are translated to C.A.S. names when members access C.A.S.

Every C.A.S. group includes an instructor. The instructor registers other group members, defines the group environment, assigns lessons, and accesses reports that contain information on student performance. Instructors are restricted to their own groups, with the exception of instructors registered in CAS$SYSGROUP.

Instructors in CAS$SYSGROUP can access any group. There can be more than one instructor in a group. There are no restrictions on the number or combination of students and authors in a group. The three possible combinations of users in a group are: instructors and students, instructors and authors, and instructors and both students and authors.

### The System Group

The C.A.S. system manager is classified as an instructor in the group CAS$SYSGROUP. All instructors in this group have the same privileges as the system manager. In most cases, CAS$SYSGROUP should be limited to the system manager and one or two other instructors for backup.

### Directories for C.A.S. Groups

When the C.A.S. system manager uses the CREATE option to create a new group, the system automatically creates a subdirectory of CAS$GROUPS for the group's files. CAS$GROUPS is the top-level directory created during the installation of C.A.S. to store all group files.

The system uses a separate subdirectory of CAS$GROUPS for each group.

This simplifies system maintenance of student status (.STA) files. A group directory stores one .STA file for each student in the group directory. The file names are generated by C.A.S. and do not identify the student's group. Since each group has a separate subdirectory, the files in the subdirectory represent all files for the individual group, and therefore all files can be restored in one operation.

## VAX/VMS ACCOUNTS AND C.A.S. USERS

As a general rule, C.A.S. users who access the VAX/VMS system for purposes other than taking lessons should have separate accounts with separate default log-in directories.

The expected use of VAX/VMS mail through C.A.S. also affects decisions about separate accounts. VAX/VMS mail requires a VAX/VMS user name. Users who share an account can send mail to other users, but they can only receive mail as a group. If it is necessary for instructors to communicate with individual students using the personal mail utility (MAIL), then the students must have individual accounts.

When a user accesses C.A.S., the system can read the VAX/VMS user name entered during log in and translate it to the user's C.A.S. name. Whether or not name translation can be used depends on two factors. First, all members of the group using name translation must have separate accounts. Second, a VAX/VMS user cannot be registered as two or more C.A.S. users in one group. For example, an author might want to be registered as several different students in one group to test lessons during development. Even though the author has a separate account, name translation cannot be used for this group.

## Disk Quotas

The disk quota required for a C.A.S. user depends on the type of user. Generally, the system manager requires a large quota. If your installation has VAX DAL, authors who write as well as deliver lessons require quotas similar to those of other programmers on the system. Instructors require smaller quotas than authors, and students require minimal quotas.

The reasons for the quota requirements are included in the following discussions of the accounts for each type of user.

## The C.A.S. System Manager

The C.A.S. system manager must have a separate VAX/VMS account. When the C.A.S. system is installed, the program GROUPCRE (group create) is executed to create CAS$SYSGROUP and register the system manager in CAS$SYSGROUP. GROUPCRE prompts for the system manager's C.A.S. user name and enters it in the C.A.S. system group's .INF file.

After CAS$SYSGROUP is created, the system manager can access C.A.S. for the first time only from the account set up by GROUPCRE. In C.A.S. the system manager can change the requirements for access to CAS$SYSGROUP and register other instructors in CAS$SYSGROUP. Instructors in this group have the same access to all groups as the system manager.

The C.A.S. system manager is the owner of a number of C.A.S. files, including all published lesson files. If disk quotas are enforced on the device that contains the C.A.S. system directory, the system manager's quota must be large enough to accommodate the system manager's files. Chapter 3 contains more information about C.A.S. system files, including approximate sizes.

## Authors

If authors not only publish lessons but also develop them using VAX DAL, they usually require separate VAX/VMS accounts. From VAX/VMS command level, authors who develop lessons must access the VAX DAL compiler, the VAX/VMS linker, printing and file maintenance utilities, and editors for creating lessons.

It may also be useful for authors working on the same project to belong to the same VAX/VMS group. This facilitates file sharing during lesson development.

If the authors' log-in directories are on a device with enforced disk storage quotas, the quotas must be large enough to accommodate program development. Some of the C.A.S. reports produce listing files that are printed from VAX/VMS command level. The default directory for these files is the author's default directory. One author's report, the listing of the logging file, can create a large file. The estimated use of this report also affects the quota requirements.

The C.A.S. system manager owns the files in CAS$LESSONS for published lessons, but authors own any copies in their own directories of files for lessons that they publish.

The enqueue quota limit (ENQLM) for an author's account affects lesson publication. An ENQLM of at least 20 is strongly recommended. Depending upon other conditions, an ENQLM lower than 20 may not allow an author to publish lessons.

## Instructors

Instructors generally should have separate VAX/VMS accounts.

Some of the C.A.S. reports produce listing files that are printed from VAX/VMS command level. The instructor's default directory is the default directory for these files. Each report also uses a default file name. (The instructor has the option of specifying a directory and file name when generating the report.)

Instructors' quotas need to be large enough to accommodate the listing files.

## Students

Students do not need access to VAX/VMS command level to take lessons.

Students in a group generally can share an account. There are two situations, however, that require individual student accounts. If students must be able to receive individual mail messages, separate accounts are required. A student's subject matter might also make separate accounts necessary. For example, computer science students who take C.A.S. lessons and also use the computer system for other class work may need separate accounts.

There are no files that use the student's directory as the default directory.

The log-in directory for a group of students can contain a log-in command file. (See the sample command files at the end of this chapter.)

## CONSIDERATIONS FOR DEFINING C.A.S. GROUPS

The following sections discuss some of the factors that determine how to group C.A.S. users.

### Groups Of Students

Three main factors determine which groups to create and how to assign students to them:

- Lesson assignments
- Reports
- The position of the instructor

Of these factors, the way C.A.S. assigns lessons is most important.

#### Lesson Assignments

C.A.S. assigns lessons to groups, not to individual students. This means that a separate group is required for each different sequence of lessons assigned. Suppose that there are two sequences of lessons to supplement classroom instruction for a chemistry class. One sequence provides remedial help in mathematics, and the other does not. Because C.A.S. assigns lessons to groups, the two lesson sequences require two separate groups.

A lesson can be restricted to a single group. The restriction is made when the lesson is published, and can be changed by the author or by the system manager. All students who are taking a restricted lesson must be members of the group to which the lesson is restricted.

### Reports

Reports are designed to present figures for individual groups. Therefore, the way student groups are arranged affects how easily reports can be sent to the people who need them. For example, students might be enrolled in a physics class that requires lectures, labs, and C.A.S. lessons. If the lecturer receives reports, it is more reasonable to put all the students in one group. If the lab instructors receive the reports, having a separate group for each lab class is more efficient.

If authors need to check the lessons they publish, or if they are developing lessons using VAX DAL, they can use reports to obtain useful information for improving the lessons. Only the author who published or republished a lesson can obtain reports for the lesson. An author can print lesson data for a particular group, or for all groups.

### The Instructor

The position of the instructor and the physical distribution of terminals may affect how students are grouped. For C.A.S. purposes, an instructor's role is administrative rather than educational. At a site that has centrally located terminals, the instructors may be computer center employees who are responsible for a number of C.A.S. groups. In this case, it is reasonable to have more than one instructor in each group and to group students so that an instructor is available whenever students are using the system.

At a site where the classroom instructor acts as C.A.S. instructor, the instructor may find it convenient to place students in C.A.S. groups that correspond to class groups.

Classroom instructors sometimes write lessons for their students. In this case, a classroom instructor is registered in a group as both author and C.A.S. instructor. The students in the group are the same as the students in the class.

## Groups of Authors

Convenience is the main reason for creating a group consisting entirely of authors.

The author's reports, like the instructor's reports, contain information only for members of a single group, or for all groups. A VAX/VMS user, however, can be registered in more than one C.A.S. group, so authors can access information for several groups.

The lesson restriction defines the group whose members can execute the lesson. When authors publish a lesson, they can specify that the lesson be restricted to any group. The author does not need to be a member of the group to which the lesson is restricted.

## EXAMPLES OF GROUPS

The following examples show possible ways of setting up groups and the VAX/VMS accounts and directories required for the groups. Sample groups are included for each example.

1 Students taking a chemistry class that requires lectures, laboratory sessions, and C.A.S. lessons

2 Students taking C.A.S. lessons to ensure that they have the appropriate background for an algebra class

3 Authors developing lessons

### Example 1

Suppose that a chemistry class requires lectures, labs, and C.A.S. lessons. Three lab instructors teach one lab section each, and are responsible for student grades.

Terminals for student use are located near the lab instructors' offices. The lab instructors have arranged their office hours so that an instructor is available whenever the terminal room is open.

In this situation, it is reasonable to use three groups, one for each of the lab sections. The lab instructor for each group is also its C.A.S. instructor. Reports for each group show the students that are registered for the corresponding lab. The groups are named LAB1, LAB2, and LAB3.

Because there are three groups, there are three group directories. These are subdirectories of the top-level group directory that the logical name CAS$GROUPS points to, and are created automatically on CAS$GRPDEV, which points to the same device and directory as CAS$GROUPS, but as a rooted directory. To make it easy to associate the subdirectories with their corresponding groups, the subdirectory names are CAS$GRPDEV:[LAB1], CAS$GRPDEV:[LAB2], and CAS$GRPDEV:[LAB3].

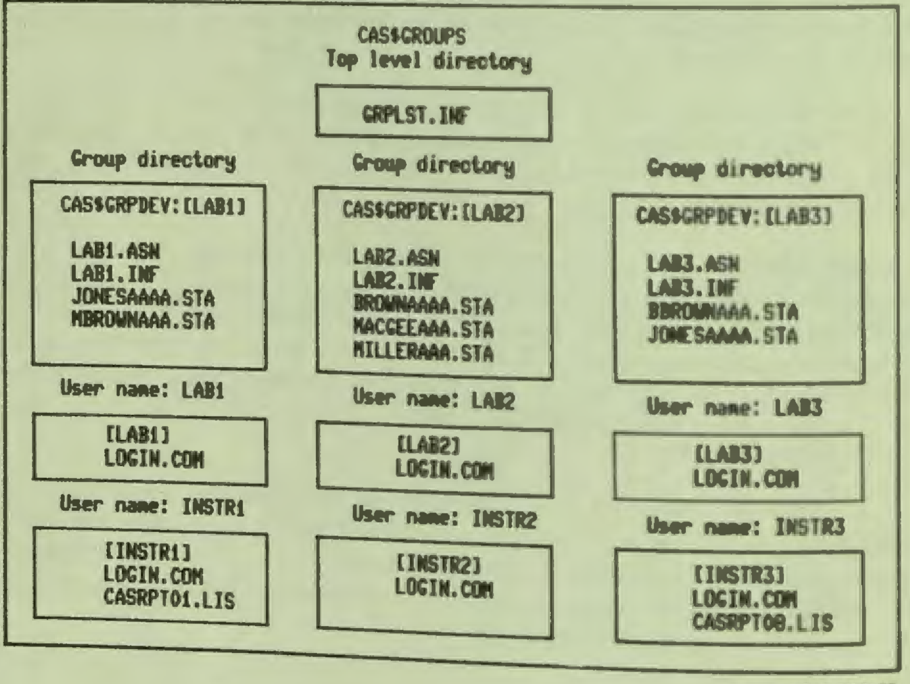
Although it is possible to use one account and one log-in directory for all students in all three groups, it is better to use three accounts. With three accounts, the process logical name CAS$GROUP, which identifies a C.A.S. group, can be defined differently in the log-in command file used by each account. Students can be placed automatically in the right C.A.S. group.

The three student accounts require quotas only large enough for a log-in command file in the default directory. Again, to make it easier to associate the VAX/VMS user names for the accounts with the log-in directories for the accounts and with the groups, the user names chosen are LAB1, LAB2, and LAB3. The log-in directories are [LAB1], [LAB2], and [LAB3].

It is possible to use one account and one log-in directory for all three instructors, but separate accounts and directories are better. The log-in command file in each directory can define CAS$GROUP so that instructors are automatically placed in their own groups. C.A.S. reports create files that are stored in the instructor's default directory. If instructors share an account, they are not able to use the default file names for reports without confusion.

The quotas for the instructors' accounts must be large enough to store report files. Because the size of report files depends upon many variables, including the number of students each instructor is responsible for, this quota varies widely.

Figure 2–1 shows the accounts and directories described above.



MR-S-3971-85

Figure 2–1
Example 1 Accounts and Directories

The VAX/VMS system manager authorizes the accounts for users and creates the six first-level directories shown in Figure 2–1.

The C.A.S. system manager creates the C.A.S. groups and registers the instructor in each group. At this point, the system manager can give control of the groups to the instructors.

Each instructor then accesses C.A.S., registers students, assigns lessons, and so on.

After students complete their lessons and their classes end, the groups are no longer needed. However, similar groups may be needed for the next term. To reinitialize a group by deleting all the students so that the group can be used for the next term, the system manager uses the INIT option in the group build/edit menu.

## Example 2

Suppose that a series of lessons is designed as a review of high school algebra. College students can take these lessons before taking a college algebra class. Students who want to take the review lessons must go to a learning center and ask to be registered. At the center are several people who can act as C.A.S. instructors for this group.

The group directory for the student group is a subdirectory of the top-level group directory, CAS$GROUPS, and it is created automatically on CAS$GRPDEV. One account and one log-in directory are needed for the group. The group directory is CAS$GRPDEV:[ALGEBRA]. The VAX/VMS user name for the account is ALGEBRA, and the VMS log-in directory is [ALGEBRA].

In this situation, the learning center employees who act as instructors probably perform other functions and have their own accounts. If they are instructors for more than one C.A.S. group, they cannot define CAS$GROUP, the logical name that points to a C.A.S. group, in their log-in command files.

Figure 2–2 shows the accounts and directories described above.

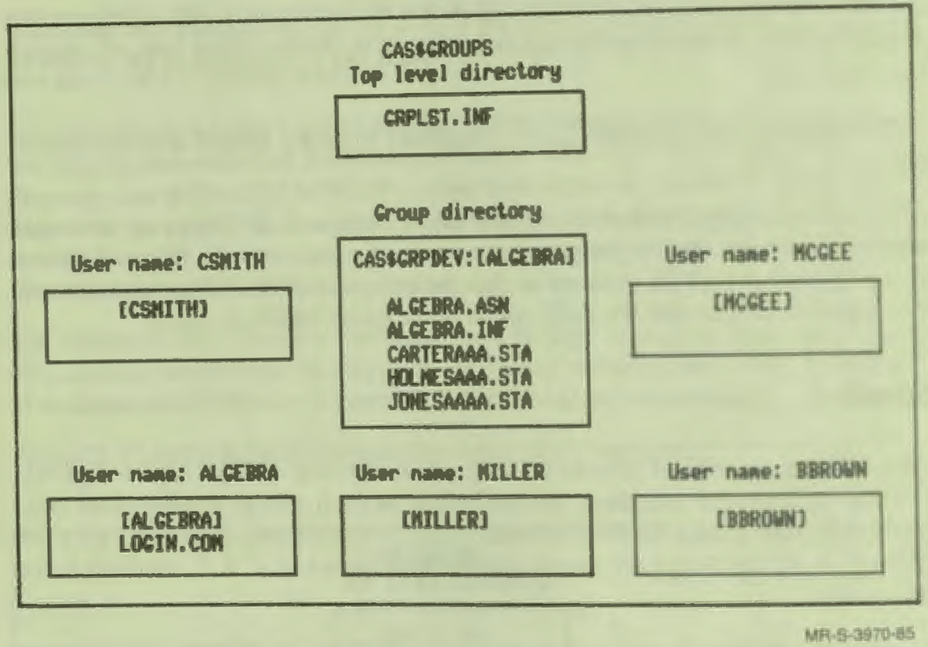


Figure 2–2
Example 2 Accounts and Directories

The VAX/VMS system manager authorizes the user with the name ALGEBRA, and creates the log-in directory for the account.

The C.A.S. system manager creates the group and registers an instructor. The instructor in turn registers other instructors and assigns the lessons to the group.

When students arrive to take lessons, any instructor registered in the group can register the students. Likewise, any instructor can check the status of student assignments and delete students who have completed the lessons.

It is possible to register a single instructor and request that everyone who is to act as an instructor log in with that user name and password. This means, however, that only one person at a time can access the system as the instructor. The system is designed to test user names, and recognizes when a second user attempts to log in with the same name as an active user.

## Example 3

Suppose that several authors are writing lessons. Although much of their work is done at VAX/VMS command level, the authors need access to C.A.S. to publish lessons and to execute the lessons as students. Authors are also likely to want to execute lessons written by other authors.

In a group designed for testing lessons, authors need to act as instructors and students as well as authors. For such a test group, it is probably best to register a single instructor, and tell the authors the user name and password. Then the authors can be completely responsible for the group. They can access C.A.S. as instructors to register themselves as students, and to assign lessons. They can also log in as authors to publish lessons and access author reports. Finally, they can access the system as students to take lessons.

Authors who publish lessons that are still being tested should restrict the lessons to the test group. Lessons that are not restricted can be executed with the BROWSE option by users in any group.

Lessons being tested are likely to contain the LOG instruction which generates log files (lesson.DLG files). Authors in a test group should be reminded to delete lesson.DLG files for their lessons as they complete testing so that the disk space can be reused.

All authors in a test group require separate VAX/VMS accounts and log-in directories. The group directory for a test group must be a subdirectory of CAS$GROUPS. The quotas for authors should be the same size as quotas for other programmers on the system.

A test group cannot use name translation. If the authors do not access any other group, they can define CAS$GROUP in their log-in command files. Accessing C.A.S. is more convenient if the instructor edits the group environment to change the requirement for a C.A.S. password.

## LOG-IN COMMAND FILES

There are several ways of using log-in command files with C.A.S. For example, a log-in command file can place students directly in C.A.S. and in the appropriate group. Students can also be logged out when they exit from C.A.S., which prevents them from reaching VAX/VMS command level.

Sites differ in the ways that they use log-in command files. Some sites have a system log-in command file that is executed for most users. In this case, the last command in the system log-in file generally executes a log-in command file in the user's log-in directory. Even when a site uses a system log-in command file, it is possible to specify different log-in command files for some accounts.

Consult the VAX/VMS system manager for the best way to handle log-in command files at your site.

## System Log-in Command Files

System log-in command files often display useful general information about the system, such as scheduled maintenance times or the status of peripheral devices. Often these files also define symbols that can be used in the same way as standard VMS commands.

If your site uses a system log-in command file, it should be executed for authors, and probably for instructors as well. For students, especially if they have no access to VAX/VMS command level, the system log-in command file may not need to be executed.

The system log-in command file should include the following command:

```
$ CAS :== @SYS$SYSTEM:CAS.COM
```

This command defines a symbol for running C.A.S.

## Log-in Command Files for Students

The following log-in command file places students directly in C.A.S. after they log in to the system, and logs them out when they exit from C.A.S. Because the process logical name CAS$GROUP is defined, C.A.S. places students in the appropriate group. The process logical name CAS$EXIT_FILE points to a file created at the site (called termreset.lis in the example that follows) of terminal control sequences. When a user exits the system, C.A.S. opens the file that CAS$EXIT_FILE points to, causing the control sequences to reset the terminal.

This log-in command file can be executed instead of the system log-in command file.

```
$! Ignore errors from the command procedure.
$ SET NOON
$! Turn off CTRL/Y and CTRL/C so students cannot
$! abort command procedure.
$ SET NOCONTROL_Y
$! Define logical name for C.A.S. group.
$! Change group name for each group.
$ DEFINE CAS$GROUP algebra
$! Open file termreset.lis to reset terminal when C.A.S. ends.
$ DEFINE CAS$EXIT__FILE termreset.lis
$! Redirect SYS$INPUT to the user terminal.
$ ASSIGN/USER 'F$LOGICAL("SYS$COMMAND")' SYS$INPUT
$! Execute C.A.S.
$ @SYS$SYSTEM:CAS.COM
$! Log out when C.A.S. ends.
$ LOGOUT
```

This log-in command file can be included in the log-in directory for accounts used by a number of students as long as the students are in the same group. The file can also be used in accounts for individual students.

## Log-in Command Files for Instructors and Authors

In most cases, instructors and authors have access to VAX/VMS command level and can edit log-in command files in their log-in directories.

The file for students can be used for instructors. If the last line is omitted, instructors are placed in C.A.S. after they log in, but are at VAX/VMS command level when they exit from C.A.S. When users exit, C.A.S. resets CTRL/Y and CTRL/C. The instructors described in example 1 above could use this log-in command file. If the instructors described in example 2 need to access several groups, they should define CAS$GROUP in their log-in command files.

If authors and instructors access only one C.A.S. group, they can define CAS$GROUP.

# 3

# C.A.S. Files

# 3
# C.A.S. Files

This chapter describes the files used by C.A.S. It provides information on file functions, locations, protection, ownership, and layout.

## DELIVERY SYSTEM FILES

The files used by C.A.S. include the following:

- The command file and the executable image files required for proper execution of the delivery system.
- The data files required by and maintained by the delivery system.
- The files required by and maintained by lessons.

Some of the delivery system files are provided as part of the software distribution kit, and others are created and maintained by the system. The installation creates the command file as well as executable images of the delivery system and reports. Some data files are created by the installation procedure, and some by C.A.S. Lesson files are automatically copied to a lesson subdirectory when an author publishes a lesson. After publication, maintenance of the lesson files is performed by or through C.A.S., depending on the file type.

### Delivery System Command File and Executable Images

The following files are required for C.A.S. They are located in the directory SYS$SYSTEM. Table 3–1 lists the delivery system files.

## Table 3–1. Delivery System Command File and Executable Images

| Directory | Files |
|---|---|
| SYS$SYSTEM: | INTERFACE.EXE |
| | CAS.COM |
| | CASRPT01.EXE |
| | CASRPT02.EXE |
| | CASRPT03.EXE |
| | CASRPT04.EXE |
| | CASRPT05.EXE |
| | CASRPT06.EXE |
| | CASRPT07.EXE |
| | CASRPT08.EXE |
| | CASRPT09.EXE |
| | CASRPT10.EXE |
| | CASRPT11.EXE |

The file INTERFACE.EXE contains the delivery system. When C.A.S. is installed with the software distribution kit, this file is installed in the directory SYS$SYSTEM as a known image with special privileges.

The file CAS.COM is an indirect command file that calls the delivery system, transfers control to the report programs and to lessons, and returns control to the delivery system when the report or lesson is finished. CAS.COM saves the status of CTRL/Y and CTRL/T and restores it when the user exits a report option or lesson. You should not change this command file in any way.

The rest of the executable images are for reports. When C.A.S. is installed from the software distribution kit, these files are installed in SYS$SYSTEM as known images with special privileges. When a user selects a report, control passes to the appropriate report program. After the report is completed, control returns to the delivery system.

Publishing a lesson creates a subdirectory of CAS$LESSONS and copies to the subdirectory the executable image of the lesson, and any additional files that the author specifies.

## Delivery System Data Files

The C.A.S. data files reside in the following locations:

- CAS$GROUPS – top-level directory of subdirectories for groups. Also contains the system group's .INF file and GRPLST.DAT.
- Group subdirectories – each group subdirectory contains files that pertain to an individual group. Some files contain information about the group, and some contain information about individual students in the group.
- CAS$LESSONS – top-level directory of subdirectories for individual lessons. Also contains the LESSONAVL.DAT file.
- Lesson subdirectories – each lesson subdirectory contains files that pertain to one lesson. These files include the student statistics file (student.STU) and the restart file (student.RST), which the system maintains for each student. Depending on the instructions in the lesson, the subdirectory also contains a permanent variable file (lesson.PRM) and the data log file (lesson.DLG).

  At the time of publication, the author specifies the name of the .EXE file containing the lesson, and the names of any other files the lesson requires to execute properly. The lesson subdirectory contains a copy of the specified .EXE file and any other files specified by the author when the lesson is published. C.A.S. does not modify files specified by the author, but lessons can contain instructions that modify files.

The nature of a C.A.S. file determines who owns the file; ownership is indicated by a user identification code (UIC). The owner of the C.A.S. data base owns all files except CAS$PUBL.LOG, the log file that is produced each time a lesson is published or republished. Each lesson's CAS$PUBL.LOG is owned by the author who published or republished the lesson. All lesson files are owned by the owner of the C.A.S. data base. The lesson start-up file passes the UIC of the C.A.S. data base to each lesson.

The approximate size of the data files is given as a guide to determining the disk space required for C.A.S. Table 3–2 does not list the lesson files that authors create since it is not possible to estimate the size of executable images for individual lessons, or the size or number of other files the lessons might require.

## Table 3–2. C.A.S. Data Files

| File | Location | Protection | Approx Size |
|---|---|---|---|
| LESSONAVL.DAT<br>1 file | CAS$LESSONS | SYSTEM:RWED<br>OWNER: R<br>GROUP: R<br>WORLD: R | 9 blocks allocated for approximately 30 lessons |
| GRPLST.DAT<br>1 file | CAS$GROUPS | SYSTEM:RWED<br>OWNER: none<br>GROUP: none<br>WORLD: none | 6 blocks allocated for first 5 groups; 1 block/5 groups thereafter |
| group.INF<br>1 file per group | group directory | SYSTEM:RWED<br>OWNER: none<br>GROUP: none<br>WORLD: none | < 20 members, 2 blocks/member; > 20 members, 1 block/member |
| group.ASN<br>1 file per group | group directory | SYSTEM:RWED<br>OWNER: none<br>GROUP: none<br>WORLD: none | 9 blocks for first 4 lessons; 1 block/4 lessons thereafter |
| name.STA<br>1 file per student | group directory | SYSTEM:RWED<br>OWNER: none<br>GROUP: none<br>WORLD: none | 1 block/4 lessons |
| lesson.STU<br>1 file per lesson | lesson directory | SYSTEM:RWED<br>OWNER: none<br>GROUP: none<br>WORLD: none | 1 block/student |
| name.RST<br>max of 1 file per student | lesson directory | SYSTEM:RWED<br>OWNER: none<br>GROUP: none<br>WORLD: none | 3 blocks (may be larger if lesson defines additional restart variables) |
| lesson.PRM<br>1 file per lesson | lesson directory | SYSTEM:RWED<br>OWNER: none<br>GROUP: none<br>WORLD: none | depends on number of permanent variables |

## Table 3–2. C.A.S. Data Files (Cont.)

| File | Location | Protection | Approx Size |
|---|---|---|---|
| lesson.DLG 1 file per lesson | lesson directory | SYSTEM:RWED OWNER: none GROUP: none WORLD: none | depends on LOG instruction in lesson |
| CAS$PUBL.LOG 1 file per publishing or republishing | author's default directory | author's default protection | depends on number of files specified during lesson publication |

Table 3–3 summarizes the interaction between C.A.S. functions and data files. The interaction is explained in the descriptions of each file that follow.

## Table 3–3. File and Function Interactions

| C.A.S. FUNCTION | FILE AFFECTED | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | lessonavl.dat | grplst.dat | group.INF | group.ASN | name.STA | lesson.STU | name.RST | lesson.PRM | lesson.DLG | CASSPUBL.LOG |
| Program GROUPCRE (during installation) | | CF | | | | | | | | |
| Create a group | | AR | CF | | | | | | | |
| Edit a group | | | ER | | | | | | | |
| Delete a group | | DR | DF | DF | DF | | | | | |
| Register any user | | | AR | | | | | | | |
| student only | | | | | CF | | | | | |
| Edit user information | | | ER | | | | | | | |
| Delete user | | | DR | | DF | | | | | |
| Assign first lesson | | | | CF | | | | | | |
| Assign lessons | | | | AR | | | | | | |
| Delete assignment | | | | DR | | | | | | |
| Delete last assignment | | | | DF | | | | | | |
| Publish first lesson | CF | | | | | | | | | |
| Publish lesson | AR | | | | | | | | | CF |
| Edit lesson information | ER | | | | | | | | DF? | |
| Republish lesson | | | | | | | DF? | DF? | | CF |
| Delete lesson | DR | | | | | DF | DF | DF | DF | |
| Execute lesson | | | | | | | | | | |
| Browse mode | | | | | | | | EF | AR | |
| Assigned – stopped | | | | | AR | | CF | EF | AR | |
| – completed | | | | | AR | AR | DF | EF | AR | |

KEY: CF Create File  AR Add Record  DF Delete File  DR Delete Record  EF Edit File  ER Edit Record

? indicates author's choice

MR-S-3964-85

**File Format**

C.A.S. contains random-access, ISAM (indexed-sequential-access mode), and sequential files. The user can check a file's type with the directory/full command through VMS. The log file (lesson.DLG) has variable length fields. All other files have fixed length fields. Fields containing characters are left justified and blank filled. All characters are uppercase. Integer fields and real number fields are long words (four bytes).

**System Data Files**

The file GRPLST.DAT, which is always stored in CAS$GROUPS, contains one record for each C.A.S. group. This file is created by the program GROUPCRE when CAS$SYSGROUP is created. Records are added to GRPLST.DAT as groups are created, and deleted as groups are deleted. GRPLST.DAT can be modified only by an instructor in CAS$SYSGROUP through the C.A.S. menu structure.

## Table 3–4. GRPLST.DAT File Format

| Field Name | Data Type | Description |
|---|---|---|
| Group_Name | 9 Chars | Group name. Entered by the C.A.S. system manager when the group is created. |
| Group_Dir | 63 Chars | Directory containing the group files. Entered by the system when the group is created. |

The file LESSONAVL.DAT, which is always stored in CAS$LESSONS, contains one record for every published lesson. This file is created when the first lesson is published, and a new record is added each time an author publishes a lesson. A record in LESSONAVL.DAT can be edited by either the author who published the lesson, or an instructor in CAS$SYSGROUP. When a lesson is deleted, by either its author or an instructor in CAS$SYSGROUP, the corresponding record in the LESSONAVL.DAT file is deleted as well.

The primary key of the LESSONAVL.DAT file is the lesson name.

## Table 3–5. LESSONAVL.DAT File Format

| Field Name | Data Type | Description |
|---|---|---|
| Lesson_Name | 9 Chars | The name of the lesson executable image file, without the .EXE extension. Entered by the author when publishing the lesson. (Entered as an auxiliary file when the lesson name is a .COM file preceded by the @ character.) |
| Aut_Name | 12 Chars | C.A.S. name of the author who published the lesson. Entered by the system during publication. |
| Les_Dscrptn | 60 Chars | Description of the lesson. Entered by the author when publishing the lesson. Description appears in all browse menus. |
| Aut_Group | 9 Chars | Author's group. Entered by the system during publication. |
| Restricted_Group | 9 Chars | Group that the lesson is restricted to. Entered by the author publishing the lesson. Contains spaces if no restriction. |
| Filename_To_Run | 38 Chars | Name of a command file to be executed when the user selects a lesson. Entered by the system if the lesson name begins with @. |

### Group Data Files

The files for a group are located in a subdirectory of CAS$GROUPS that is automatically created when the C.A.S. system manager creates the group. (The group files for CAS$SYSGROUP are always in CAS$GROUPS.) A group's files always include a group information file (group.INF).

When a group is assigned its first lesson, the group's assigned lesson file (group.ASN) is created. A student status file (student.STA) is created for each student as the student is registered. Records are added to this file as the student takes assigned lessons.

Table 3–6 lists the group data files.

## Table 3–6. Group Data Files

| Directory | Files | Description |
|---|---|---|
| group directory specified when group is created | group.INF | group information file |
| | group.ASN | assigned lessons |
| | studenta.STA<br>studentb.STA<br>studentc.STA<br>studentd.STA | student status files — one file per student |

The file group.INF contains a single record that defines the group environment. It also includes one record for each member registered in the group. The group.INF file is created when the group is created, and at that point it includes the group environment record. The record for the first instructor is added when the system manager registers an instructor in the group. Records are added, deleted, or modified as the instructor or the system manager registers group members, deletes them, or changes the information for them.

The system manager can modify all group.INF files through C.A.S. Instructors can modify the group.INF files for their own groups. The group.INF files are accessible from VAX/VMS command level only to users with system privileges.

The group information file has the following three record types:

- The group environment record, which defines the group environment.
- Student records, which contain student information.
- Author and instructor records, which contain information about authors and instructors.

These three record types have different formats. The tables that follow list the file formats for each type.

The key of the group environment record is 12 spaces. Student, author, and instructor records use the C.A.S. name as a primary key, and the VAX/VMS user name as a secondary key. Student, author, and instructor records are distinguished from one another by the STATUS character. For students, the STATUS field contains S. For authors, the STATUS field contains A. For instructors, the STATUS field contains G.

## Table 3–7. GROUP.INF File Format – Group Environment Record

| Field Name | Data Type | Description |
|---|---|---|
| CAS_Name | 12 Chars | blank |
| Vax_Name | 38 Chars | blank |
| Group_Desc | 60 Chars | Description of the group. Entered by the instructor. |
| Login_Disable | Byte | A value of 1 indicates that members of the group are not allowed to access C.A.S. Set if system manager marks group for deletion. |
| Pswrd_Req | Byte | A value of 1 indicates that users must supply a password to log in to C.A.S. Set by default. Modified by instructor. |
| Use_Vax_Name | Byte | A value of 1 indicates name translation from SYS$LOGIN to a C.A.S. name. Clear by default. Modified by instructor. |
| Numof_Assign | Integer | Number of lessons assigned to the group. Updated as records are added to or deleted from the group assignment file (.ASN). |

## Table 3–8. GROUP.INF File Format – Student Information Record

| Field Name | Data Type | Description |
|---|---|---|
| CAS_Name | 12 Chars | Primary key. C.A.S. name entered by instructor. |
| Vax_Name | 38 Chars | Secondary key. Used when name translation flag in the group environment is set. VAX/VMS user name as entered by instructor. |
| Status | 1 Char | Contains "S" for student. Entered when instructor registers the student. |

Table 3–8. GROUP.INF File Format – Student Information Record (Cont.)

| Field Name | Data Type | Description |
|---|---|---|
| Password | 9 Chars | The student's C.A.S. password. Used only if Pswrd_Req is set in the group environment record. Entered by student during first log in. Erased by instructor. |
| Real_Name | 20 Chars | The student's name. Entered by instructor. |
| Running_Lesson | Integer | Flag that indicates whether student is running a lesson. A nonzero value indicates that the student is running a lesson. |
| Sys_Tim | Real | Total number of hours spent in C.A.S. |
| Las_Date | 11 Chars | Last date student used C.A.S. |
| Num_Ses | Integer | Number of times student used C.A.S. |
| Ave_Ses_Ln | Real | Average time spent in one session. Derived by dividing the data in the Sys_Tim field by the data in Num_Ses. |
| Sesl_Dev | Real | Session length standard deviation. |
| Sum_SessIn_Sq | Real | The sum of the session lengths squared (used in the calculation of Sesl_Dev). |
| Las_Les_Usd | 9 Chars | Last lesson (assigned or browse mode) executed, but not necessarily completed, by the student. |
| Las_Asn_Usd | 9 Chars | Last assigned lesson executed. |
| Filler | Real | blank |
| Filler | Byte | blank |
| User_File_Name | 9 Chars | Name of the student's student status file (.STA). |

Table 3–9. GROUP.INF File Format – Instructor and Author Information Record

| Field Name | Data Type | Description |
|---|---|---|
| CAS_Name | 12 Chars | Primary key. C.A.S. name entered by instructor. |
| Vax_Name | 38 Chars | Secondary key. Used when name translation is set. VAX/VMS user name as entered by instructor. |
| Status | 1 Char | Contains "G" for instructor or "A" for author. Entered when instructor or author is registered. |
| Password | 9 Chars | The user's C.A.S. password. Used only if Pswrd_Req is set in the group environment record. Entered during first log in. Erased by instructor. |
| Real_Name | 20 Chars | The user's actual name. Entered when an instructor or an author is registered. |
| Running_Lesson | Integer | Flag that indicates whether student is running a lesson. A nonzero value indicates that the student is running a lesson. |
| Phone | 13 Chars | The user's phone number. Entered when an instructor or an author is registered. |
| Address | 40 Chars | The user's address. Entered when an instructor or an author is registered. |
| User_File_Name | 9 Chars | Not used. |

The file group.ASN contains a record for each assigned lesson. The file is created when the first lesson is assigned. Records are added or deleted when assignments are added or deleted.

The system manager can modify all group.ASN files through C.A.S. Instructors can modify the group.ASN files for their own groups.

The primary key of the group.ASN file is the assigned lesson; the secondary key is the next lesson assigned. Students take assigned lessons in either structured or unstructured order, depending upon the choice the instructor enters when making the group's first assignment. If the key of the first record in the file contains spaces, it indicates that assignments are structured.

## Table 3–10. GROUP.ASN File Format

| Field Name | Data Type | Description |
|---|---|---|
| Lesson | 9 Chars | Primary key. The name of the executable image. Entered by the instructor when making an assignment. |
| Next_Less_Name | 9 Chars | Secondary key. The name of the next lesson to be taken. This may be an executable image or a command file. Used only if lesson assignments are structured. The instructor specifies structured or unstructured order when making the first assignment to the group. |
| Less_Dscrptn | 60 Chars | Description of the assignment. Entered by the instructor when making the assignment. Displayed on list of assigned lessons. |
| Due_Date | 11 Chars | Date the lesson is due. Entered by the instructor when making the assignment. Displayed on list of assigned lessons, but not enforced. Used only for documentation. |

One student status file is created when each student is registered. The system generates the name of the file, and the extension is .STA. One record is added to a student's status file each time the student begins an assigned lesson. The record is modified if the student completes the lesson or ends the lesson without completing it. The records in a student's file are not modified if a lesson is removed from the assigned lesson file or the system.

The primary key for .STA files is the lesson name.

## Table 3–11. STUDENT.STA File Format

| Field Name | Data Type | Description |
|---|---|---|
| Lesson | 9 Chars | Primary key. File name of the executable image lesson file. |
| Comp_Status | 1 Char | "P"–Partially done "D"–Done Updated whenever a student ends a lesson. |
| Restart_file | 63 Chars | Full file name of this student's restart file for the lesson above. |

### Lesson Files

The lesson subdirectory that is automatically created when a lesson is published contains the executable image of the lesson and any auxiliary files specified by the author. The lesson subdirectory also contains a student statistics file for the lesson, and may contain restart files, permanent variable files, and logging files.

## Table 3–12. Lesson Files

| Directory | Files | |
|---|---|---|
| lesson subdirectory of CAS$LESSONS created when lesson is published | lesson.EXE<br>lesson.STU<br>lesson.ABS | required for publication<br>maintained by C.A.S.<br>specified by author |
| | studentaa.RST<br>studentab.RST<br>studentaf.RST<br>studentar.RST | exist for all users during execution of lesson; exist for students who have begun an assigned lesson but stopped the lesson before completing it |
| | lesson.PRM<br>lesson.DLG<br>xxxx.SHO<br>xxxx.PIC<br>xxxx.FNT<br>xxxx.DAT | specified by author |

The student statistics file (lesson.STU) contains one record for each student who has completed the lesson as an assignment. Each record contains summary statistics for the student. The .STU file is created when the first student in any group to which the lesson is assigned completes the lesson. Records are appended to the file for each student who completes the lesson. C.A.S. reports read this file.

The .STU file is accessible from VAX/VMS command level only to users with system privileges.

The lesson.STU file is sequential. Much of the information in this file is copied from the system variables when the lesson ends. The system variables are included under the description of each field in Table 3–13. For more information about system variables, refer to the *VAX DAL Reference Manual*.

## Table 3–13. LESSON.STU File Format

| Field Name | Data Type | Description |
|---|---|---|
| User_Group | 9 Chars | Group the student belongs to. |
| User_Dal_Name | 12 Chars | Student's C.A.S. name. |
| Les_Cmpt | Integer | The lesson is completed if this field is equal to 256. |
| NNO | Integer | The number of responses judged NO (NNO). |
| NOKFIRST | Integer | The number of first responses judged OK (NOKFIRST). |
| NOK | Integer | The number of responses judged OK (NOK). |
| FOK_Ratio | Floating Point | The number of first responses judged OK (NOKFIRST) divided by the number of responses judged NO (NNO). |
| N_Queries | Integer | The number of queries that the lesson executed (QUERIES). |
| Score | Floating Point | The student's total score (SCORE). |

**Table 3–13. LESSON.STU File Format (Cont.)**

| Field Name | Data Type | Description |
|---|---|---|
| Scoreable_Units | Integer | The number of units in the lesson that are scored. |
| Total_Time | Integer | The number of seconds the student spent in the lesson. |
| Goal | Integer | The goal at the end of the lesson (GOAL). |
| Scores_Array | Floating Point | Scores for each goal. This array is indexed by the integer goal number (SCORES(g)). |

A restart file (studentx.RST) is created each time a lesson is executed. During execution of the lesson, this file contains the current values of the system variables. If the lesson is executed in browse mode or if the lesson is completed, the file is deleted when the lesson ends. If the lesson is executed as an assignment and is stopped with the STOP instruction, the file is saved. When the student begins the lesson again, the saved restart file is used. The system generates a unique file name for each restart file. Authors can delete restart files for lessons they published. See the *VAX DAL Reference Manual* for the student.RST file format.

The permanent variable file (lesson.PRM) is created only if the lesson defines permanent variables. The file is created the first time the lesson is executed and updated during each subsequent execution. The .PRM file is accessible only by the lesson. Authors can delete the .PRM file for the lessons they publish. See the *VAX DAL Reference Manual* for the lesson.PRM file format.

The log file (lesson.DLG) is created only if the lesson executes a LOG instruction. The first time a lesson with a LOG instruction is executed, the file is created. Thereafter, a temporary file is created by each execution, and appended to the main file when the lesson ends. The .DLG file is read by C.A.S. reports.

The size of the lesson.DLG file depends on the type of logging, the number of units in the lesson, and the number of users who execute the lesson. Log files can become large enough to cause problems with disk quotas.

The system manager can delete the lesson.DLG file for any lesson. Authors can delete the lesson.DLG files for lessons they published.

When authors publish lessons with logging enabled, they should print reports and delete the lesson.DLG files frequently. When they no longer need to capture the information generated by the LOG instruction, they can edit the lesson source file to remove the instruction. The lesson can be compiled and linked again and republished.

The lesson.DLG file is a sequential file. The record types and their identifying numbers are shown in Table 3–14. Several records of the same type can occur in succession in a lesson.DLG file. For example, a student might enter a response that is judged wrong, then a response that is judged right. Both responses are logged as type 3 records.

## Table 3–14. LESSON.DLG File Format

| Record Number | Record Name | Data Type | Description |
|---|---|---|---|
| 0 | DAL_Name | 20 Chars | The user's C.A.S. name. |
| 1 | Unit_Name | 9 Chars | Name of unit. (Units that do not contain QUERY instructions are not logged.) |
| 2 | Date_Time | 23 Chars | The system date and time when the user entered the unit above. The format for the date and time is: DD-MMM-YYYY HH:MM:SS.HH |
| 3 | Actual Response | 500 Chars | What the user entered. C.A.S. reports truncate this field to 80 characters. |
| 4 | Latency | 4 Integers | Time that elapsed between display of the prompt and end of user's response. |
| 5 | Unanticipated Responses | 500 Chars | Any word not specified in RIGHT or WRONG instructions. Reports truncate this field to 80 characters. |

When users press the PF1 (detail) key in any menu, they receive a more detailed explanation of the current option. For most menus, these detailed explanations are part of C.A.S. The system cannot, however, provide lesson descriptions since the available lessons are different at every site.

The lesson.ABS file allows authors to provide descriptions of their lessons. Authors create this file in their directories, enter a lesson description into it, and specify it as an auxiliary file when publishing a lesson.

To display a lesson description, the user presses the PF1 (detail) key in the list of assigned lessons or the browse menu. If there is no lesson.ABS file, the PF1 key has no effect.

# 4 Using C.A.S.

# 4
# Using C.A.S.

The *C.A.S. Delivery System User's Guide* explains how to use C.A.S. The user's guide discusses how to select options from the menus, how to enter information in response to prompts, and how to use the auxiliary keypad. The user's guide also explains all instructor's menus and all author's menus.

This chapter explains how the functions of the system manager differ from the functions of instructors and authors, and contains detailed instructions for those functions that the system manager must perform.

The four functions of the system manager include:

- Creating new groups
- Registering the first instructor for a new group
- Editing the group environment and user information for the system group
- Deleting groups

The system manager is the only user who can display a list of all current groups.

The one function that the system manager cannot perform is publishing lessons. Only authors can publish lessons, and only the author who published a lesson can republish it.

Table 4–1 summarizes the C.A.S. functions and the users who can perform them.

**Table 4–1. C.A.S. Users and Functions**

| Function | System Manager | Instructor | Author | Student |
|---|---|---|---|---|
| Create groups | X | | | |
| Delete groups | X | | | |
| Register first instructor | X | | | |
| List all groups | X | | | |
| Register users | all groups | own group | | |
| Edit group information | all groups | own group | | |
| Delete users | all groups | own group | | |
| Reset running lesson flag | for all users at one time | for each user in own group | | |
| Assign lessons delete assignments | own group | own group | | |
| Publish and republish lessons | | | X | |
| Edit lesson information | all lessons | | own lessons | |

Table 4–1. C.A.S. Users and Functions (Cont.)

| Function | System Manager | Instructor | Author | Student |
|---|---|---|---|---|
| Access reports | all reports for all groups | instructor's reports for own group | author's reports for own group | |
| Take lessons | all lessons | depends on restrictions | depends on restrictions | depends on restrictions and assignments |

## THE PROGRAM GROUPCRE

The program GROUPCRE creates the group CAS$SYSGROUP and registers the system manager in the group. This program is executed during installation by VMSINSTAL.

The program GROUPCRE creates two files, GRPLST.DAT and the group.INF for CAS$SYSGROUP. Once GROUPCRE is executed, it is deleted to prevent its being executed again.

GROUPCRE creates the file GRPLST.DAT with one record identifying the system group. It also creates the group information file for CAS$SYSGROUP with two records, the group environment record and one instructor information record. The instructor information record contains the C.A.S. name entered in GROUPCRE and may contain a VAX/VMS user name read from the logical name SYS$LOGIN.

## INVOKING C.A.S.

The group CAS$SYSGROUP exists after the program GROUPCRE is executed by VMSINSTAL. You can access C.A.S. to create other groups and register an instructor in those groups.

Use the following procedure to invoke C.A.S. for the first time.

1 When the system prompt ($) is displayed, type CAS and press RET.

2 When prompted for the group, enter the name defined for CAS$SYSGROUP during installation.

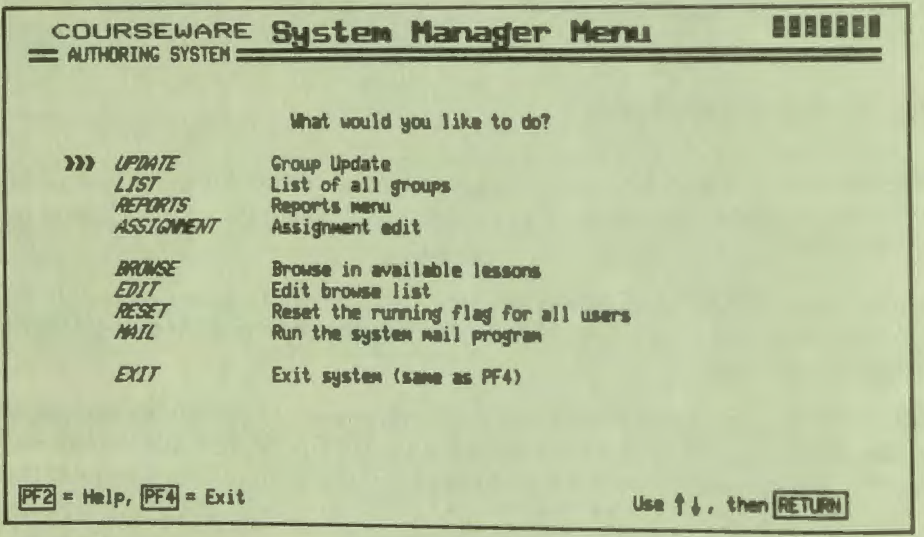
3 When prompted for the system manager's name, enter the C.A.S. account name for the system manager that you defined during installation.

4 The system prompts for a new password, then asks you to enter the password again as verification. Enter and verify your password.

5 The C.A.S. system manager menu appears.

## SYSTEM MANAGER MENU

The first menu displayed is the system manager menu shown in Figure 4–1.



```
COURSEWARE          System Manager Menu
== AUTHORING SYSTEM ==============================

                What would you like to do?

»» UPDATE       Group Update
   LIST         List of all groups
   REPORTS      Reports menu
   ASSIGNMENT   Assignment edit

   BROWSE       Browse in available lessons
   EDIT         Edit browse list
   RESET        Reset the running flag for all users
   MAIL         Run the system mail program

   EXIT         Exit system (same as PF4)


PF2 = Help, PF4 = Exit                    Use ↑↓, then RETURN
```

MR-S-3646-84

Figure 4–1
System Manager Menu

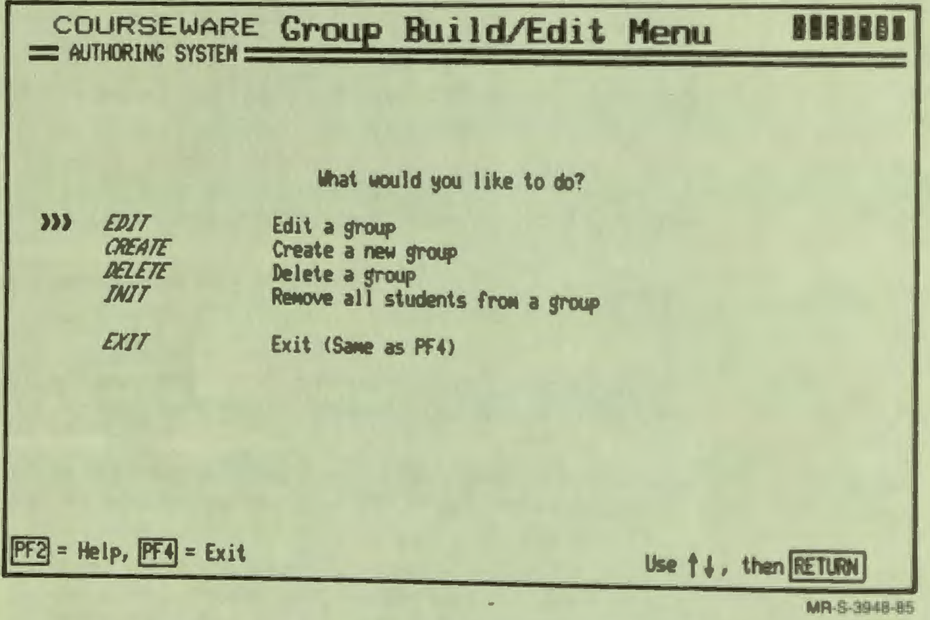
The options on the system manager menu are summarized in Table 4–2.

## Table 4–2. System Manager Menu Options

| Option | Description |
|---|---|
| UPDATE | displays the group build/edit menu for editing, creating, deleting, and re-initializing groups. Editing a group includes registering and deleting individual group members, changing information stored for each member, and changing the group environment. Instructors can edit their own groups. |
| LIST | displays the group name and directory of all C.A.S. groups. An asterisk marks any group for which log ins are disabled, indicating that the system manager can delete the group. |
| REPORTS | displays the menu of all C.A.S. reports. Both authors and instructors can access reports. |
| ASSIGNMENT | displays the group assignment menu for listing or altering the assignments for a group. Instructors can also assign lessons. |
| BROWSE | displays the list of published lessons. Lessons are executed by selecting them from this list. All C.A.S. users can browse. |
| EDIT | displays a lesson edit screen for editing the lesson description and the group restriction, deleting the lesson's log file (lesson.DLG), or changing the author's C.A.S. name and group to reassign the lesson to another author. Authors can edit the lessons they published. |
| RESET | resets the running lesson flag in all records in the group.INF file for all groups. This option displays the name of each group as it processes that group's group.INF file. Instructors can reset the running lesson flag for each user in their group. |
| MAIL | accesses the VAX/VMS personal mail utility (MAIL) for sending and receiving system mail. All C.A.S. users can access MAIL. |
| EXIT | exits from C.A.S. and returns to VAX/VMS command level. The PF4 key performs the same function. |

## Edit, Create, or Delete a Group

The UPDATE option from the system manager menu displays the group build/edit menu shown in Figure 4–2.



```
COURSEWARE Group Build/Edit Menu
AUTHORING SYSTEM

                    What would you like to do?

>>>  EDIT          Edit a group
     CREATE        Create a new group
     DELETE        Delete a group
     INIT          Remove all students from a group

     EXIT          Exit (Same as PF4)

PF2 = Help, PF4 = Exit
                                        Use ↑↓, then RETURN
```

MR-S-3948-85

Figure 4–2
Group Build/Edit Menu

Table 4–3 summarizes the options on the group build/edit menu.

**Table 4–3. Group Build/Edit Menu Options**

| Option | Description |
|---|---|
| EDIT | displays the group edit menu. Instructors in all groups access this menu to edit their groups. |
| CREATE | displays a screen requesting the name and directory specification for the group to be created. |
| DELETE | displays a screen requesting the name of the group to be deleted. |
| INIT | displays a screen requesting the name of the group to be re-initialized. INIT allows a user to delete a group's students and student restart and .STA files while retaining the group's instructors, authors, and assignments. |

## Edit a Group

The system manager can edit any group. Instructors in all groups except CAS$SYSGROUP can edit only their own groups. When you select the EDIT option, you are prompted for the name of the group to be edited. To edit CAS$SYSGROUP, press RET. To edit any other group, enter the name of that group. From this point, editing a group is identical for the system manager and for any instructor.

C.A.S. displays the group edit menu shown in Figure 4–3.

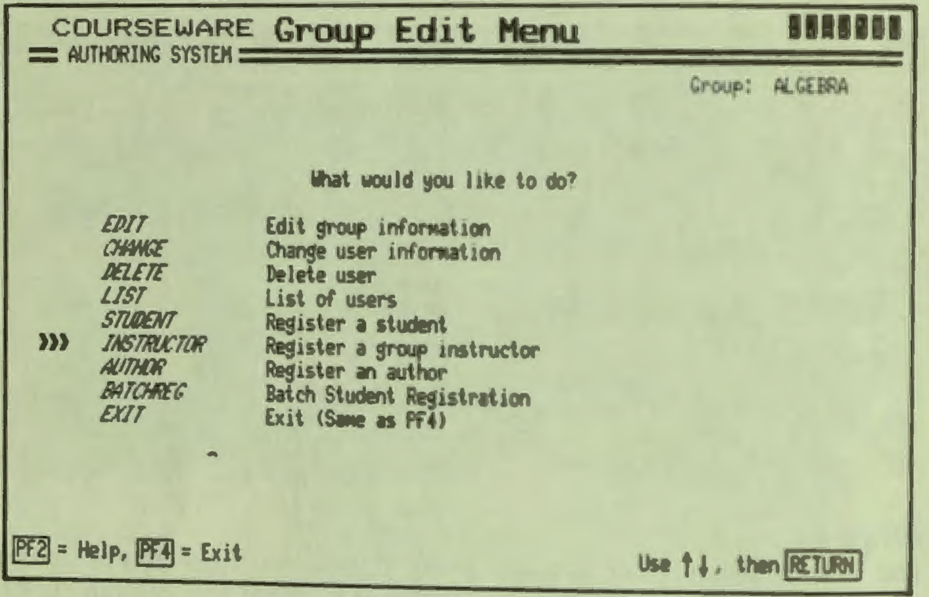

```
COURSEWARE Group Edit Menu
AUTHORING SYSTEM
                                          Group:  ALGEBRA

                    What would you like to do?

       EDIT          Edit group information
       CHANGE        Change user information
       DELETE        Delete user
       LIST          List of users
       STUDENT       Register a student
  )))  INSTRUCTOR    Register a group instructor
       AUTHOR        Register an author
       BATCHREG      Batch Student Registration
       EXIT          Exit (Same as PF4)

PF2 = Help, PF4 = Exit                     Use ↑↓, then RETURN
```
MR-S-3952-85

Figure 4–3
Group Edit Menu

Table 4–4 summarizes the options on the group edit menu.

**Table 4–4. Group Edit Menu Options**

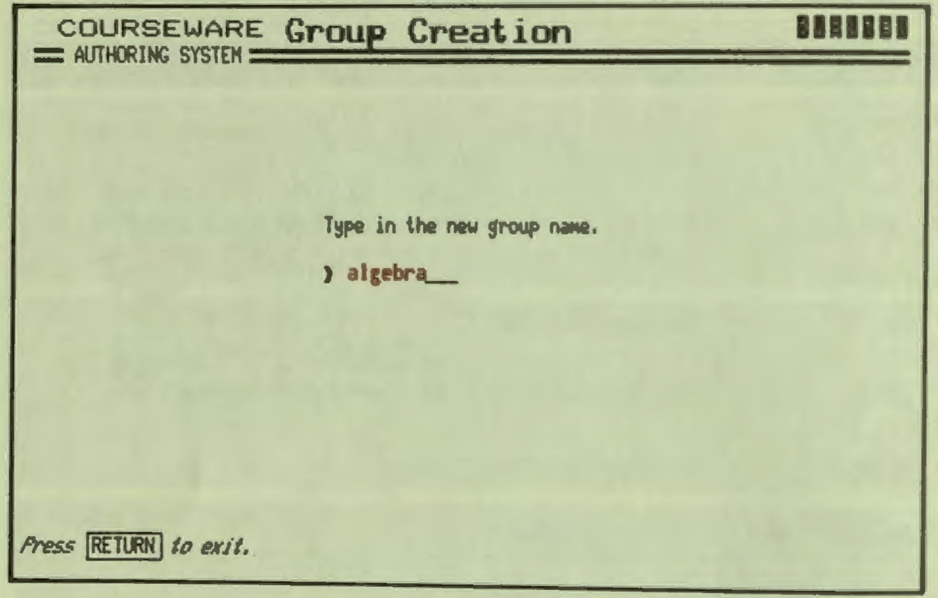
| Option | Description |
|---|---|
| EDIT | edits the information stored in the group environment record in the group information file. |
| CHANGE | edits the information entered when an instructor, author, or student was registered, or deletes a student's restart files. |
| DELETE | removes a user from C.A.S. |
| LIST | displays a list of the current members of the group. |
| STUDENT | registers a single student. |
| INSTRUCTOR | registers an instructor. |
| AUTHOR | registers an author. |
| BATCHREG | registers multiple students. |

**Create a Group and Register an Instructor**

Only the C.A.S. system manager can create groups and register the first instructor in a new group. That instructor can then register all other users, assign lessons, and so on.

When creating a new group, the C.A.S. system manager does not need to create a subdirectory for the group. The system automatically creates a new subdirectory of CAS$GROUPS using the rooted directory, CAS$GRPDEV.

Figure 4–4 shows the screen and the prompts to create a new group.



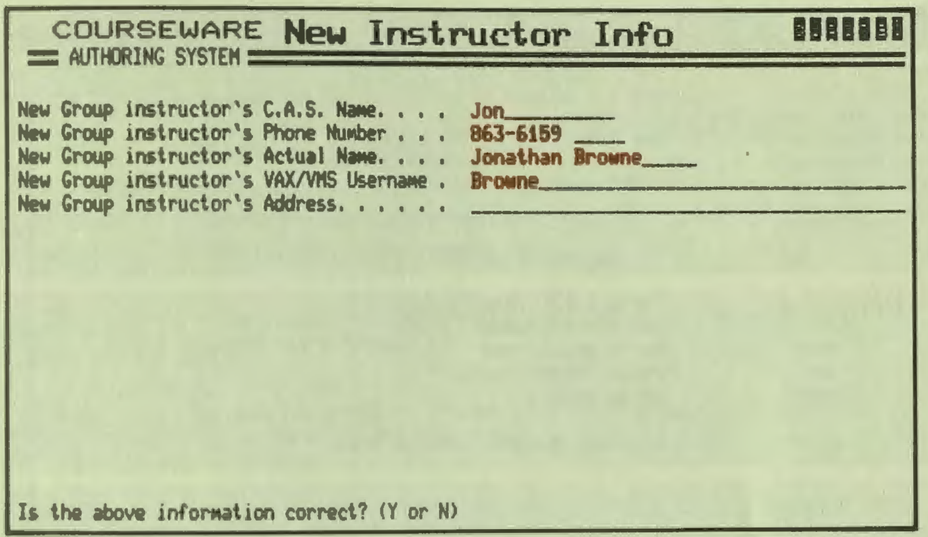
```
Type in the new group name.

) algebra__

Press [RETURN] to exit.
```

Figure 4–4
Group Creation Screen

After the new group is created, the group edit menu is displayed (see Figure 4–3) so you can register the first instructor in the new group.

The INSTRUCTOR option from the group edit menu displays the new instructor information screen. This screen contains prompts for the C.A.S. name, phone number, actual name, VAX/VMS user name, and address of the new instructor.

Figure 4–5 shows this screen with the information for a new instructor entered and with the confirmation message displayed at the bottom.


```
COURSEWARE New Instructor Info
AUTHORING SYSTEM

New Group instructor's C.A.S. Name. . . .  Jon
New Group instructor's Phone Number . . .  863-6159
New Group instructor's Actual Name. . . .  Jonathan Browne
New Group instructor's VAX/VMS Username .  Browne
New Group instructor's Address. . . . . .

Is the above information correct? (Y or N)
```

MR-S-3957-85

Figure 4–5
New Instructor Information Screen

Suppose that you type Y in response to the following confirmation message:

Is the above information correct?

If there is another user in the group with the same C.A.S. name, or if the C.A.S. name is not entered, you receive an error message.

If you type Y and receive an error message, or if you type N in response to the confirmation message, you receive the following prompt:

Do you want to edit the information? (Y or N)

If you answer N, you return to the group edit menu. If you answer Y, the new instructor information screen with change options, shown in Figure 4–6, is displayed.

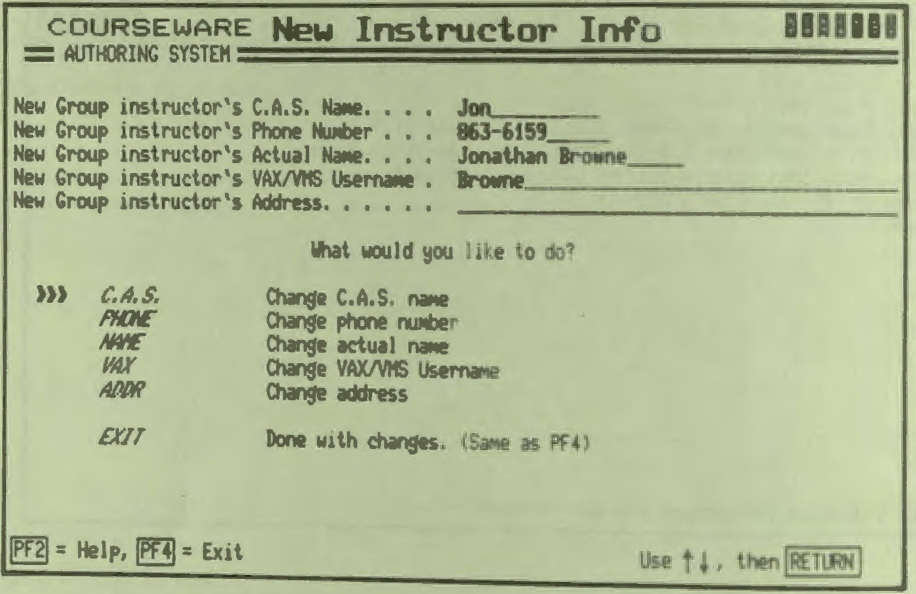

```
COURSEWARE New Instructor Info
AUTHORING SYSTEM

New Group instructor's C.A.S. Name. . . .  Jon________
New Group instructor's Phone Number . . .  863-6159_____
New Group instructor's Actual Name. . . .  Jonathan Browne____
New Group instructor's VAX/VMS Username .  Browne______________________________
New Group instructor's Address. . . . . .  ____________________________________

                          What would you like to do?

 >>>  C.A.S.           Change C.A.S. name
      PHONE            Change phone number
      NAME             Change actual name
      VAX              Change VAX/VMS Username
      ADDR             Change address

      EXIT             Done with changes. (Same as PF4)

PF2 = Help, PF4 = Exit                              Use ↑↓, then RETURN
```
MR-S-3958-85

Figure 4–6
New Instructor Information Screen with Change Options

Table 4–5 summarizes the options on the change screen.

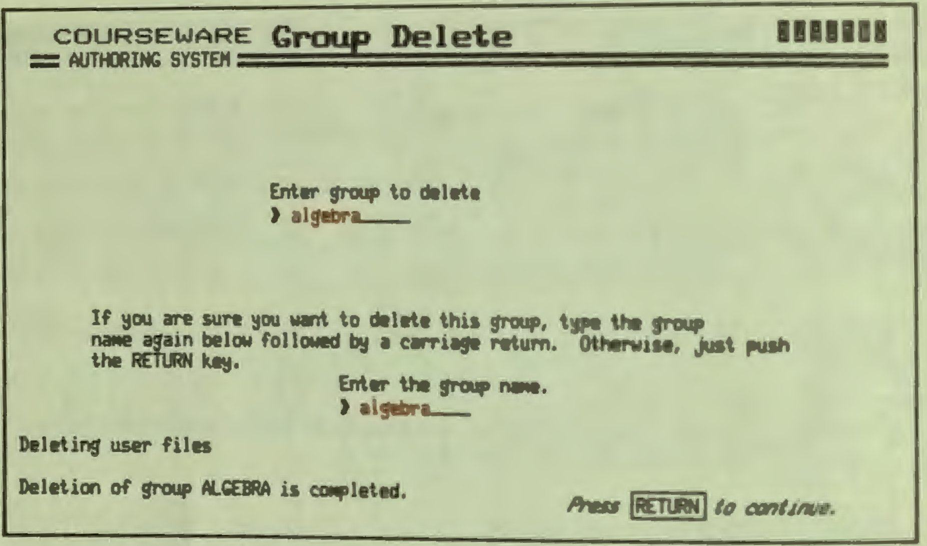
## Table 4–5. New Instructor Information Change Options

| Option | Description |
|---|---|
| C.A.S. | erases the old C.A.S. name, moves the cursor to the C.A.S. name line, and waits for a new name to be entered. (Figure 4–5 shows the old name deleted.) |
| PHONE | erases the old phone number, moves the cursor, and waits for a new number to be entered. |
| NAME | erases the old actual name, moves the cursor, and waits for a new name to be entered. |
| VAX | erases the old VAX/VMS user name, moves the cursor, and waits for a new name to be entered. |
| ADDR | erases the old address, moves the cursor, and waits for a new address to be entered. |
| EXIT | returns to the group edit menu. |

After a user is registered, the C.A.S. name cannot be changed since it is used in a number of files.

**Delete a Group**

To delete a group, select the DELETE option from the group build/edit menu. This displays the group delete screen shown in Figure 4–7.



```
                    Enter group to delete
                    > algebra___


      If you are sure you want to delete this group, type the group
      name again below followed by a carriage return.  Otherwise, just push
      the RETURN key.
                          Enter the group name.
                          > algebra___
Deleting user files

Deletion of group ALGEBRA is completed.
                                             Press [RETURN] to continue.
```

MR-S-3660-84

Figure 4–7
Group Delete Screen

You cannot delete a group while a group member is using C.A.S. If you attempt to do so, you are asked if the group should be marked for later deletion. Marking a group for deletion sets the Login_Disable flag in the group environment record of the group.INF file. Once you mark a group for deletion, users are not allowed to log in to the group, and the group can be deleted later. When the C.A.S. system manager lists the groups, the system displays an asterisk next to any group marked for deletion and a message asking if the system manager wants to delete the group.

Entering the name of a nonexistent group displays an error message.

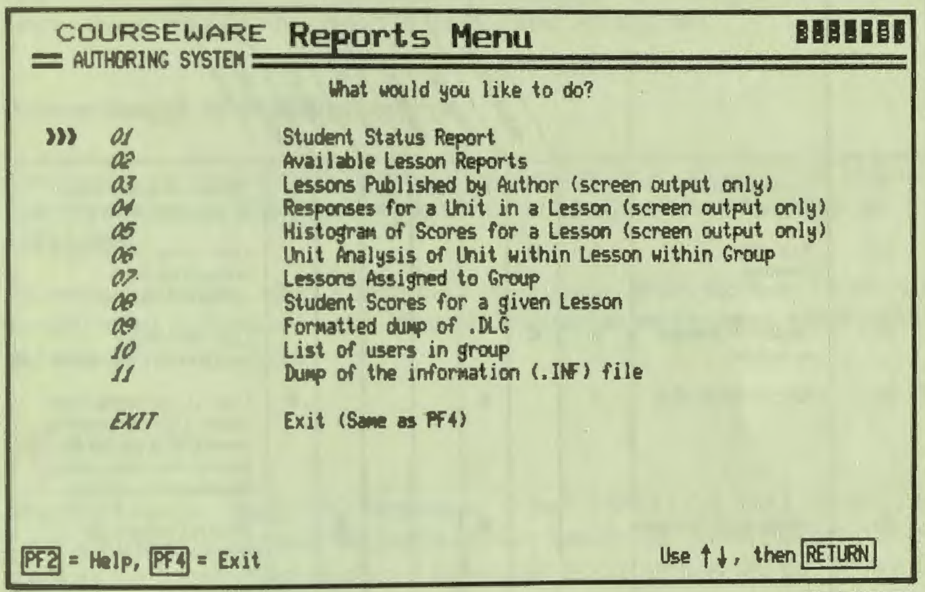
When you delete a group, the system removes the group files from the group directory, deletes the directory, and removes the entry for the group from the system data file GRPLST.DAT. The VAX/VMS account and log-in directories for group members are not affected.

## List C.A.S. Groups

To list all the C.A.S. groups, select the LIST option from the system manager menu.

## Select a Report

The REPORT option on the system manager menu displays the reports menu shown in Figure 4–8.



```
COURSEWARE          Reports Menu
AUTHORING SYSTEM
                    What would you like to do?

>>>  01       Student Status Report
     02       Available Lesson Reports
     03       Lessons Published by Author (screen output only)
     04       Responses for a Unit in a Lesson (screen output only)
     05       Histogram of Scores for a Lesson (screen output only)
     06       Unit Analysis of Unit within Lesson within Group
     07       Lessons Assigned to Group
     08       Student Scores for a given Lesson
     09       Formatted dump of .DLG
     10       List of users in group
     11       Dump of the information (.INF) file

     EXIT     Exit (Same as PF4)

PF2 = Help, PF4 = Exit                    Use ↑↓, then RETURN
```

MR-S-3950-85

Figure 4–8
Reports Menu

Table 4–6 summarizes the reports and shows the files that provide data. The reports do not modify any files.

## Table 4–6. Report Summary

| REPORT NUMBER | REPORT FILE | REPORT FOR AUTHOR OR INSTRUCTOR | FILES READ | | | | | | DESCRIPTION |
|---|---|---|---|---|---|---|---|---|---|
| | | | lessonavl.dat | group.INF | group.ASN | student.STA | lesson.STU | lesson.DLG | |
| 01 | Student Status Report | I | | ■ | ■ | ■ | | | Shows complete status of assigned lessons. |
| 02 | Available Lessons | A & I | ■ | | | | | | Lists name, description, and author of all published lessons. |
| 03 | Lessons Published by Author | A | ■ | | | | | | Lists all lessons published by one author. |
| 04 | Responses for Unit | A | | ■ | | | | ■ | Lists all responses from lesson.DLG file made by members of group. Also graphs five most common responses on screen. |
| 05 | Histogram of Scores | I | | ■ | | | ■ | | Displays percentage distribution of scores for lesson. |
| 06 | Analysis of Unit | A | | ■ | ■ | | | ■ | Lists information for checking overall student performance. |
| 07 | Assigned Lessons | I | | | ■ | | | | Lists name and description of assigned lessons. |
| 08 | Student Scores | I | | ■ | | | ■ | | Lists percent scores or standard scores, and calculates mean, standard deviation, and variance. |
| 09 | Formatted dump of lesson.DLG file | A | | ■ | | | | ■ | Formats and lists current .DLG file records. |
| 10 | List of Users | I | | ■ | | | | | Lists instructors, authors, and students in group. |
| 11 | Dump of Group Information File | I | | ■ | | | | | Lists all information except passwords in the group.INF file. |

MR-S-2104-82

The prompts displayed by the different reports are explained in the *C.A.S. Delivery System User's Guide*. The guide also shows examples of the reports.

Because the system manager can access reports for any group, the first prompt the system manager sees after selecting a report is a prompt for the group name. To print a report for the system group, press RET in response to the group prompt. To print a report for any other group, enter the name of the group.

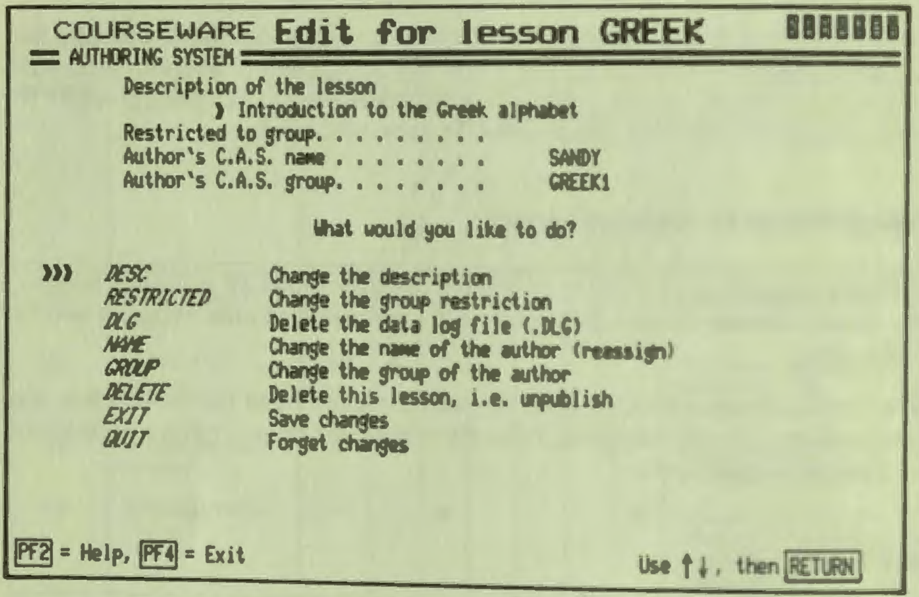
## Browse through All Published Lessons

The browse menu displayed for the system manager shows all published lessons. The browse menus displayed for other users are restricted both by group and by assignment.

To execute a lesson, select it from the browse menu. When the lesson ends, the browse menu is displayed again. Press PF4 to leave the browse menu and return to the system manager menu.

## Edit Browse List

The EDIT option changes the information in the LESSONAVL.DAT file and can be used to delete a lesson or assign it to another author. The system prompts for the name of the lesson, then displays a lesson edit screen such as the one shown in Figure 4–9.



```
COURSEWARE Edit for lesson GREEK
= AUTHORING SYSTEM
        Description of the lesson
              ) Introduction to the Greek alphabet
        Restricted to group. . . . . . . . .
        Author's C.A.S. name . . . . . . . .     SANDY
        Author's C.A.S. group. . . . . . . .     GREEK1

                     What would you like to do?

  )))  DESC          Change the description
       RESTRICTED    Change the group restriction
       DLG           Delete the data log file (.DLG)
       NAME          Change the name of the author (reassign)
       GROUP         Change the group of the author
       DELETE        Delete this lesson, i.e. unpublish
       EXIT          Save changes
       QUIT          Forget changes

PF2 = Help, PF4 = Exit                          Use ↑↓, then RETURN
```

MR-S-3951-85

Figure 4-9
Lesson Edit Screen

Table 4-7 summarizes the lesson edit options.

## Table 4–7. Lesson Edit Options

| Option | Description |
|---|---|
| DESC | erases the current description, moves the cursor, and waits for a new description to be entered. The description is displayed on all browse lists. The instructor enters the description for assigned lessons when making assignments. |
| RESTRICTED | erases the old restriction, moves the cursor, and waits for a new restriction to be entered. Press RET to delete the lesson restriction. |
| DLG | deletes the log file in the lesson subdirectory. This option requests confirmation before deleting the file. |
| NAME | erases the author's C.A.S. name, moves the cursor, and waits for the user to enter a new author's name. This option effectively reassigns the lesson. |
| GROUP | erases the author's C.A.S. group, moves the cursor, and waits for the user to enter a new C.A.S. group for the author. Use this when reassigning a lesson if the new author is in a different C.A.S. group. |
| DELETE | deletes the lesson, including all files in the lesson subdirectory as well as the subdirectory itself. The record for the lesson in the system file LESSONAVL.DAT is also deleted. |

# Index

# Index

## READER'S COMMENTS

**Note:** This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. Problems with software should be reported on a Software Performance Report (SPR) form. If you require a written reply and are eligible to receive one under SPR service, submit your comments on an SPR form.

Did you find errors in this manual? If so, specify by page.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Is there sufficient documentation on associated system programs required for use of the software described in this manual? If not, what material is missing and where should it be placed?

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Non-programmer interested in computer concepts and capabilities

Name ______________________ Date ______________

Organization ______________________

Street ______________________

City ______________ State ______________ Zip Code or Country ______________

Please cut along this line.

Fold here

DECLIT AA VAX K767C

C.A.S. delivery system :
  system manager's guide

DECLIT AA VAX K767C

C.A.S. delivery system :
  system manager's guide

|  |  |  |  |
|---|---|---|---|
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Do Not Te

No Postage
Necessary
If Mailed In The
United States

BUSINES
FIRST CL

Postage

200 Forest Street MRO1-2/L12
Marlborough, Massachusetts 01752

AA-K767C-TE
Printed in U.S.A.

# X

# C.A.S. Delivery System

User's Guide

digital

software

94-003104911.3

# C.A.S. Delivery System User's Guide

ORDER NUMBER AA-K764C-TE

To order additional copies of this document, contact the Software Distribution Center, Digital Equipment Corporation, Maynard, Massachusetts 01754

First Printing, July 1982
Second Printing, July 1984
Third Printing, December 1985

| | | |
|---|---|---|
| DAL | Edusystem | RSTS |
| DEC | GIGI | RSX |
| DECnet | PDP | UNIBUS |
| DECtalk | PDT | VAX |
| DECwriter | ReGIS | VMS |
| DIBOL | RENAISSANCE | VT |
| | | digital™ |

81611

## Contents

# Contents

# Chapter 2
# The Instructor

## Tables

# PREFACE

This manual is intended for instructors and authors. It explains how to use the Courseware Authoring System Delivery System (C.A.S.) to provide computer-aided instruction (CAI) for students.

In the context of C.A.S., authors publish CAI lessons, students take the lessons, and instructors manage records and assignments for groups composed of all three types of users.

Chapter 1 is an introduction. It describes C.A.S. users and groups in detail, and explains how to enter C.A.S., define a password, and send and receive mail. It also explains how C.A.S. interprets the keys on the keyboard and keypad.

Chapter 2 is intended for instructors. It describes the role of the instructor and contains samples and descriptions of the instructor's menus, screens, and reports.

Chapter 3 is for authors. It describes the role of the author and includes samples and descriptions of the author's menus, screens, and reports.

Chapter 4 is intended for both instructors and authors. It describes the role of the student and contains samples and descriptions of the student screen and menus.

Instructors and authors should read the introduction and the chapter that particularly relates to them. To familiarize themselves with student use of C.A.S., they should also read Chapter 4. Students should consult the *C.A.S. Delivery System Student Guide* as their reference.

## REFERENCES

This manual references the following books of the VAX/VMS Documentation Set:

- *Introduction to VAX/VMS*
  Order No. AA-Y500A-TE
- *VAX/VMS DCL Dictionary*
  Order No. AA-Z200A-TE
- *VAX/VMS System Messages And Recovery Procedures Manual*
  Order No. AA-Z800A-TE
- *VAX/VMS Mail Utility Reference Manual*
  Order No. AA-Z421A-TE

## RELATED DOCUMENTS

| | |
|---|---|
| *VAX DAL Author's Guide* Order No. AA-K763C-TE | This manual is a guide to the DIGITAL Authoring Language (VAX DAL) for new CAI authors or experienced authors who are new to VAX DAL. |
| *VAX DAL Pocket Reference Guide* Order No. AV-DB13B-TE | This document is a quick reference guide to the commands and features of VAX DAL. |
| *VAX DAL Reference Manual* Order No. AA-K768C-TE | This manual is the complete reference guide to VAX DAL. |
| *C.A.S. Delivery System Student Guide* Order No. AV-K765C-TE | This document is a guide for students using the C.A.S. Delivery System. It explains how to access assigned lessons, browse through lessons, use the menus and keypad, and send and receive mail. |
| *C.A.S. Delivery System System Manager's Guide* Order No. AA-K767C-TE | This manual explains concepts and requirements of C.A.S. for the person who has overall administrative responsibility for the system. |

## CONVENTIONS USED IN THIS MANUAL

This manual uses the following conventions:

- These symbols represent pressing a key on the main keypad or auxiliary keypad:

| | |
|---|---|
| RET | PF3 |
| SHIFT | PF4 |
| PF1 | , |
| PF2 | ENTER |

- This symbol indicates holding down the CTRL key while pressing the main keypad key O:

  CTRL/O

- Main keypad and auxiliary keypad key names that are unique to C.A.S. are capitalized (for example, SELECT).
- Quotation marks enclose some of the messages that C.A.S. displays on the monitor.
- In examples of commands entered by users, anything that a user types is printed in color.
- Italicized words used in examples indicate information that changes according to what the user enters.

# 1 Getting Started with C.A.S.

# 1 Getting Started with C.A.S.

The Courseware Authoring System Delivery System (C.A.S. Delivery System) is a menu-driven system that delivers computer-aided instruction to students. The C.A.S. Delivery System, which will be referred to from this point as C.A.S., includes an interface and a reporting package that are designed to support lesson delivery, scoring, and administration. C.A.S. can deliver lessons written in any computer language.

VAX DAL, which is a C.A.S. companion product, consists of a compiler for the DIGITAL Authoring Language and a computer-aided course in using the language. Authors can use VAX DAL to write computer-aided instruction (CAI).

This manual discusses how to use C.A.S. to deliver lessons to students. Lesson delivery involves the following activities:

- Creating groups, which are collections of instructors, authors, and students.
- Registering instructors, authors, and students in groups.
- Publishing, republishing, and deleting lessons. (Publishing a lesson means making it available to users.)
- Assigning lessons to students.
- Generating reports that record the progress of individual students and show information about lessons and groups.

## C.A.S. USERS

The users of C.A.S. include the C.A.S. system manager, instructors, authors, and students. Each type of user performs specific functions, although all users can browse through lessons and send and receive system mail.

After C.A.S. is installed, the VAX/VMS and C.A.S. system managers determine which user groups to create and what to name them, whether accounts are to be shared or separate, and who the instructor is for each group.

The C.A.S. system manager also adds users to the initial system group; creates groups and registers the first instructor in them; deletes groups; and edits the browse list. (The C.A.S. system manager's role is discussed in detail in the *C.A.S. Delivery System System Manager's Guide*.)

Instructors change the group environment; register authors, students, and any additional instructors in the group; assign published lessons to the group; and obtain reports.

Authors control the availability of lessons within the C.A.S. Delivery System by publishing, republishing, and unpublishing lessons. After an author publishes a lesson, the lesson may be assigned to students. Users can browse through lessons once they are published.

Authors produce reports on lesson performance which assist them in improving lessons. Some authors may also be involved in writing lessons.

Students do published lessons.

Table 1–1 illustrates the functions performed by each type of C.A.S. user.

**Table 1–1. Functions Performed By C.A.S. Users**

| USER TYPE | FUNCTION |
|---|---|
| C.A.S. System Manager | Add users to the initial system group<br>Create and edit groups<br>Delete groups<br>Register instructors<br>Edit the browse list<br>Browse<br>Use MAIL utility |
| Instructors | Change the group environment<br>Register group members<br>Assign lessons<br>Obtain reports<br>Browse<br>Use MAIL utility |
| Authors | Publish lessons<br>Republish lessons<br>Delete lessons<br>Obtain reports<br>Browse<br>Use MAIL utility |
| Students | Do assignments<br>Browse<br>Use MAIL utility |

## THE C.A.S. GROUP

A C.A.S. group is a collection of one or more instructors, authors, and students. All C.A.S. users are members of groups. Each group has one or more instructors, and can contain any number of authors and students, who are registered in the group by instructors.

Each C.A.S. group has a group environment. When a group is created, it has a default environment which the group instructor can change. In the default environment, there is no group description, passwords are required, and VAX/VMS user name translation, which is explained in this chapter under "Shared and Separate Accounts," is disabled.

## Group Attributes

Groups have the following attributes:

- Group environments that consist of a group description, a password flag, and a VAX/VMS user name translation flag.
- Records of group members that are maintained by instructors.
- Performance, usage, and personal data of group members that are stored together in group files.
- Assignments that are shared by the students in a group.
- Lessons that are restricted to only one group.
- Reports that show information for the group.

## Kinds Of C.A.S. Groups

Each installation can organize users into groups as it chooses. Possible groups include the following:

- A group of students, with at least one instructor. Groups of students can be separated by subject matter or by class level. For example, there can be single chemistry, math, and history groups; or there can be several groups for each subject, each representing a different class level.
- A group solely of authors, either with a separate instructor, or with one of the authors in the group also registered as the instructor.
- A group of both students and authors, with at least one instructor.

Figure 1–1 illustrates typical C.A.S. groups.

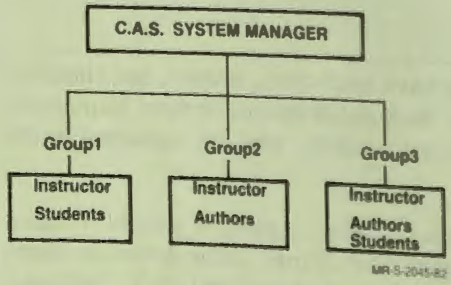


Figure 1–1
Typical C.A.S. Groups

## RUNNING C.A.S.

Before running C.A.S., users must log in to their VAX/VMS accounts. Logging in to a VAX/VMS account requires a user name and a password. The VAX/VMS user name and password identify the user to the VAX/VMS system. (The *Introduction to VAX/VMS* manual contains information on the VAX/VMS log-in procedure.)

After logging in to their VAX/VMS accounts, users can run C.A.S. by typing the word CAS and pressing RET in response to the VAX/VMS prompting symbol, the dollar sign ($), as follows:

```
$ CAS RET
```

Alternatively, they can execute a command file that includes this statement.

### User Identification

Users identify themselves by group, C.A.S. name, and password. A group and a C.A.S. name can be specified in several ways; the user does not necessarily have to type them. Users do not have to type a group if their group name is defined as a logical name (CAS$GROUP) in a log-in command file in their VAX/VMS account. Likewise, users do not have to type their C.A.S. names if VAX/VMS user name translation is enabled in their group environments.

### Defining A Password

If the instructor requires a password for the group, group members must define a password the first time they run C.A.S. A password assignment screen appears and prompts for a password of up to nine characters. A second password prompt appears to ensure that the user remembers the password. If the password is retyped correctly, the user accesses the first menu. Whenever the second password is typed incorrectly, a message advises the user that the two passwords do not match. If this message appears, the user should press RET to repeat the password assignment procedure.

To ensure confidentiality, passwords are not echoed on the screen as they are typed. Users should be careful to choose passwords that are easy to remember.

## Automatic C.A.S. Entry And VAX/VMS Logout

It is possible to set up log-in command files so that users automatically enter C.A.S. after logging in to their VAX/VMS accounts, and automatically log out of their VAX/VMS accounts after exiting from C.A.S. Such log-in command files are normally used to prevent students from accessing VAX/VMS at the command level. Examples of log-in command files of this type are shown in the *C.A.S. Delivery System System Manager's Guide*. The C.A.S. system manager should provide instructors and authors with information on log-in command files at their site.

## Shared Or Separate Accounts

VAX/VMS accounts can be set up so that users in a group can either share one VAX/VMS account or have separate VAX/VMS accounts. Students are likely to share accounts; instructors and particularly authors are likely to have separate accounts.

Users of a shared account have the same VAX/VMS user name and the same VAX/VMS password. Therefore, groups in which all members share one VAX/VMS account cannot use VAX/VMS user name translation, which automatically associates a C.A.S. name with each VAX/VMS user name. User name translation is available only with separate accounts and only where a VAX/VMS user is not registered as a different C.A.S. user in the same group. (For example, VAX/VMS user name translation cannot be used where an author is also registered as an instructor.) Users who share an account are more likely to enter C.A.S. passwords than users with separate accounts.

## User Logical Names

The following two logical names can be defined in the user's log-in command file:

- CAS$GROUP – Defines the group the user belongs to.
- CAS$EXIT_FILE – Identifies the file of terminal control sequences that C.A.S. opens to reset the terminal when the user exits the system.

CAS$EXIT_FILE can also be defined in the group or system logical name table instead of the user's log-in command file.

## C.A.S. TERMINALS

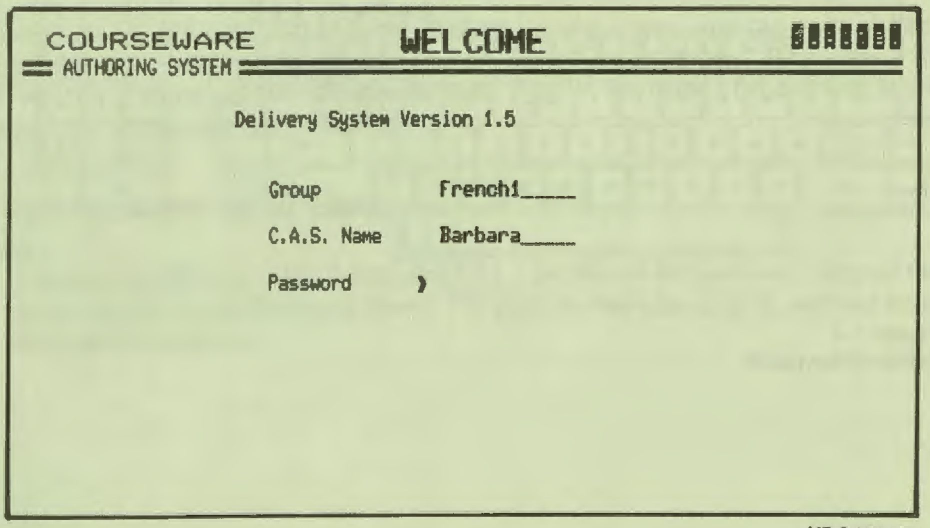
C.A.S. requires a terminal that is attached to a color or black-and-white monitor. The terminal and the monitor are connected to a VAX/VMS computer system. The terminal must support the Remote Graphics Instruction Set (ReGIS).

### C.A.S. Menus And Screens

Users communicate with C.A.S. by selecting options from menus that are displayed on the monitor and by typing answers in response to prompts.

The menus contain options that perform functions or access other menus or screens. Screens display information, prompt for information, and contain options for changing the information shown. There are separate sets of menus for instructors, authors, and students.

Unless their log-in command files allow them to access C.A.S. directly, users see the welcome screen shown in Figure 1–2 after logging in to their VAX/VMS accounts and typing "CAS". The welcome screen includes a password prompt if passwords are required. After typing the information that the welcome screen prompts for, users access the menus for an instructor, author, or student, depending on their function in C.A.S.



```
Delivery System Version 1.5

   Group          French1____

   C.A.S. Name    Barbara_____

   Password      ›
```

MR-S-3990-85

Figure 1–2
Welcome Screen

## Terminal Keypads

Each C.A.S. terminal has a main keypad and an auxiliary keypad; some terminals feature an additional editing keypad.

The main keypad contains the standard character and numeric keys, the symbol keys, the arrow keys, the RET key, and so on. The auxiliary keypad, located on the right side of the keyboard, contains the PF keys, the ENTER key, and several others. The editing keypad is located between the main and auxiliary keypads on terminals that have this feature. Figure 1–3 shows the two keyboards commonly used for C.A.S.

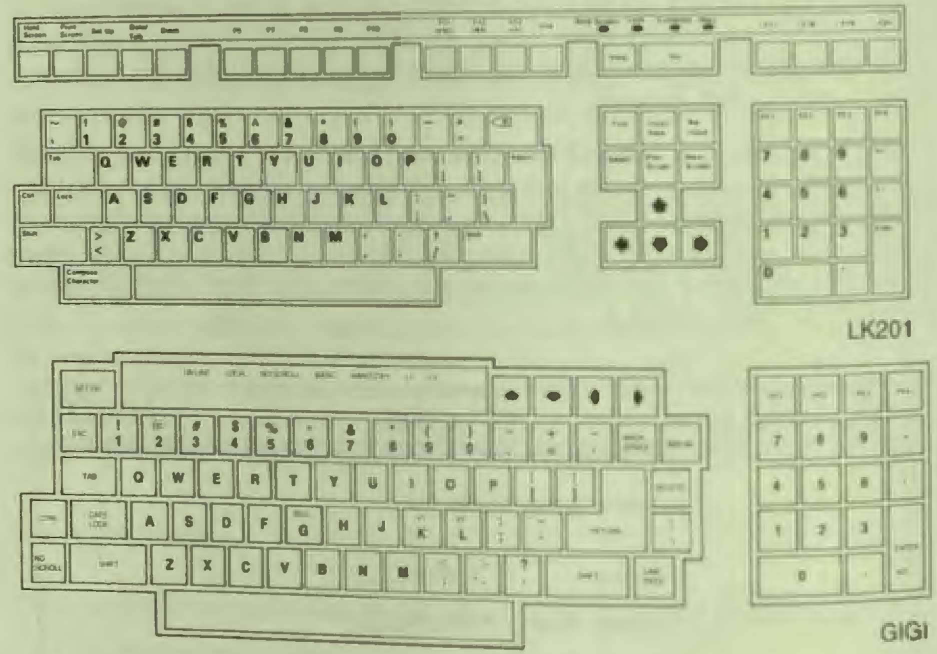


Figure 1–3
Terminal Keypads

## Selecting Options and Lessons

You use the auxiliary keypad keys for manipulating screens, selecting options or lesson names from menus, and getting on-line help.

The arrow keys are located on the main keypad of some terminals; on other terminals, they are part of the editing keypad. When used in combination with the RET key, the arrow keys select an option or a lesson name from a menu. The arrow keys move the selection arrow up and down on the monitor so that it points to the desired option or lesson name. When the selection arrow is pointing to an option or lesson name, press RET on the main keypad or the ENTER key on the auxiliary keypad, or the SELECT key on the editing keypad to invoke that option or lesson.

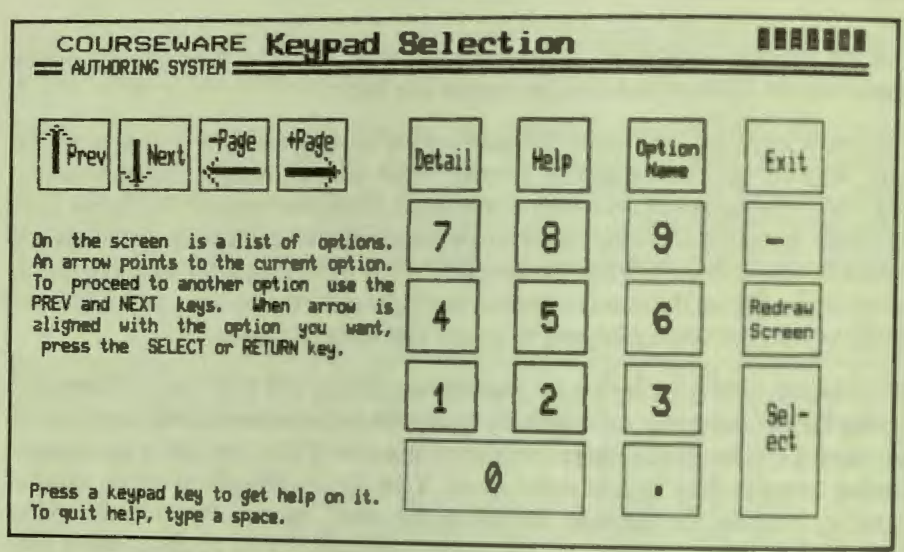
You can also select an option by pressing the PF3 key (OPTION NAME key) and typing the option name, or simply by typing the option name without pressing the key first. Option names entered with or without the PF3 key can either be abbreviated or typed as they appear in the menu. If the full or abbreviated option name is typed incorrectly, the message "Invalid option name" appears. Retyping the option name clears the error message. You select assignments from an assignments list, lessons from a browse menu, and reports from a report menu, in the same way as other options.

To back up one level in the menu structure, press PF4. This is helpful when you forget the name of an option and want to enter it directly.

If a menu is in the process of being displayed on the screen, you can press any key to stop the display and produce the prompt "Enter Option Name:". The echo of the key you pressed appears after the prompt. Type in the name of the option at this prompt, typing after the echoed character.

## Auxiliary Keypad Key Functions

Pressing the PF2 key (HELP key) displays a picture of the auxiliary keypad that looks like the screen displayed below. The picture shows the C.A.S.-defined arrow keys and keypad keys.



On the screen is a list of options. An arrow points to the current option. To proceed to another option use the PREV and NEXT keys. When arrow is aligned with the option you want, press the SELECT or RETURN key.

Press a keypad key to get help on it. To quit help, type a space.

Figure 1–4
Keypad Selection Screen

The functions that are invoked by pressing the arrow keys and the auxiliary keypad keys are described below:

- The PREV key is the up arrow key. Pressing PREV moves the selection arrow up to the previous option on the screen; if the selection arrow is at the top option, it skips to the bottom option.
- The NEXT key is the down arrow key. In a menu, pressing NEXT moves the selection arrow down to the next option; in a list that continues on more than one screen, pressing NEXT displays the next screen. If a list contains only one screen, and the selection arrow is at the bottom option, pressing NEXT moves the arrow back up to the top option.
- The -PAGE key is the left arrow key. Pressing -PAGE displays the previous screen of options when the options fill more than one screen. (The PREV SCREEN key on the editing keypad has the same effect as -PAGE.)
- The +PAGE key is the right arrow key. Pressing +PAGE displays the next screen of options when the options fill more than one screen. (The NEXT SCREEN key on the editing keypad has the same effect as +PAGE.)

- The DETAIL key is the PF1 key. Pressing DETAIL displays a description of the option that the selection arrow is pointing to. Pressing DETAIL for assigned lessons and in browse mode produces a description of the highlighted lesson if the author who published a lesson provided a text file (lesson.ABS) containing the lesson description.
- The HELP key is the PF2 key. Pressing HELP displays the picture of the C.A.S. auxiliary keypad that is shown here in the keypad selection screen. When the keypad selection screen is displayed, pressing a key produces more information about that key.

  To exit from the keypad selection screen, the user must press the space bar.
- The OPTION NAME key is the PF3 key. Pressing OPTION NAME produces a prompt for an option name. To select the option, the user types its full or abbreviated name.
- The EXIT key is the PF4 key. The user presses it to exit from the current menu and return to the previous menu.
- The REDRAW SCREEN key is the ☐ key. The user presses it to redraw the current screen.
- The SELECT key is the ENTER key. Pressing SELECT is equivalent to pressing RET. (Pressing the SELECT key on the editing keypad has the same effect as pressing RET or ENTER.)

**Note:** In C.A.S. the user must terminate typed input by pressing the RET or ENTER key.

## MAIL

C.A.S. offers a MAIL option in the instructor, author, and student main menus. The MAIL option accesses the VAX/VMS MAIL utility. The MAIL utility is discussed in the instructor and author chapters in this manual as well as in *VAX/VMS Mail Utility Reference Manual*.

## WHAT YOU NEED TO KNOW

You need the following information in order to log in to your VAX/VMS account, run C.A.S., and perform the tasks described above:

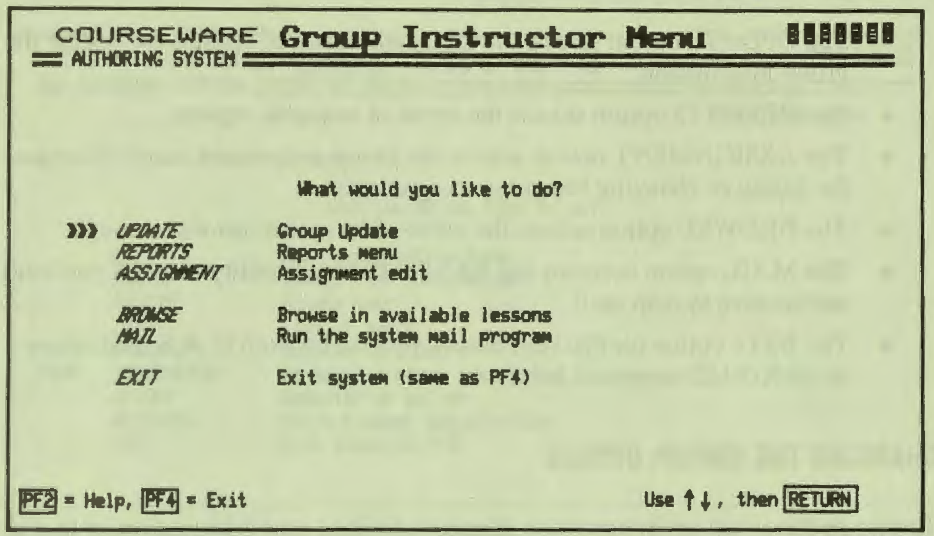
- Your VAX/VMS user name and password.
- Your C.A.S. name.
- The name of your group and whether the accounts of group members are shared or separate.

  Before you are registered in the group, the VAX/VMS or C.A.S. system manager assigns the group name and then determines whether accounts are to be shared or separate. Whether accounts are shared or separate influences the types of changes you make to your group environment. Since VAX/VMS user name translation cannot be used with shared accounts, it must remain disabled if students in your group share an account. For a shared account you probably want to require passwords; if so, do not disable the default password requirement.
- The user information for the authors and students to be registered in your group.

  When you register authors, you need to know their C.A.S. names. If VAX/VMS user name translation is used, you also need to know the authors' VAX/VMS user names. In addition, you may want to know their phone numbers, actual names, and addresses. When you register students, you need to know their C.A.S. names. You may also want to know their actual names and VAX/VMS user names.
- The names of lessons that students should take.

## GROUP INSTRUCTOR MENU

When you run C.A.S., you access menus that allow you to perform the activities described previously. The first menu displayed is the group instructor menu shown in Figure 2–1.



```
COURSEWARE      Group Instructor Menu
AUTHORING SYSTEM

                         What would you like to do?

 >>> UPDATE          Group Update
     REPORTS         Reports menu
     ASSIGNMENT      Assignment edit

     BROWSE          Browse in available lessons
     MAIL            Run the system mail program

     EXIT            Exit system (same as PF4)

PF2 = Help, PF4 = Exit                          Use ↑↓, then RETURN
```

MR-S-3628-84

Figure 2–1
Group Instructor Menu

By selecting options from the group instructor menu, you access other menus with options for changing the group information and assignments and for selecting reports and lessons. Figure 2–2 shows where you are placed after selecting options from the group instructor menu.

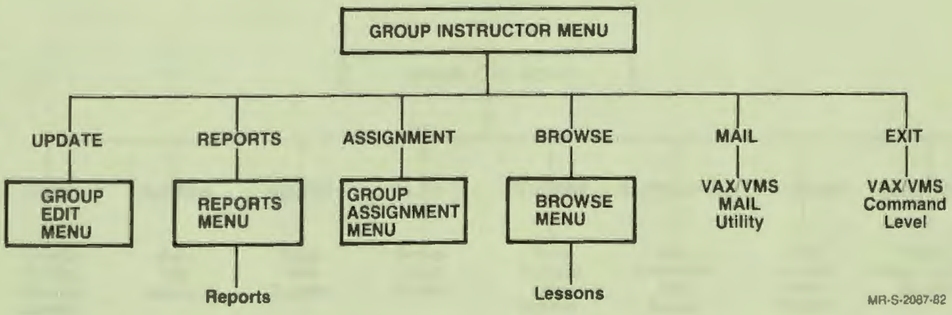


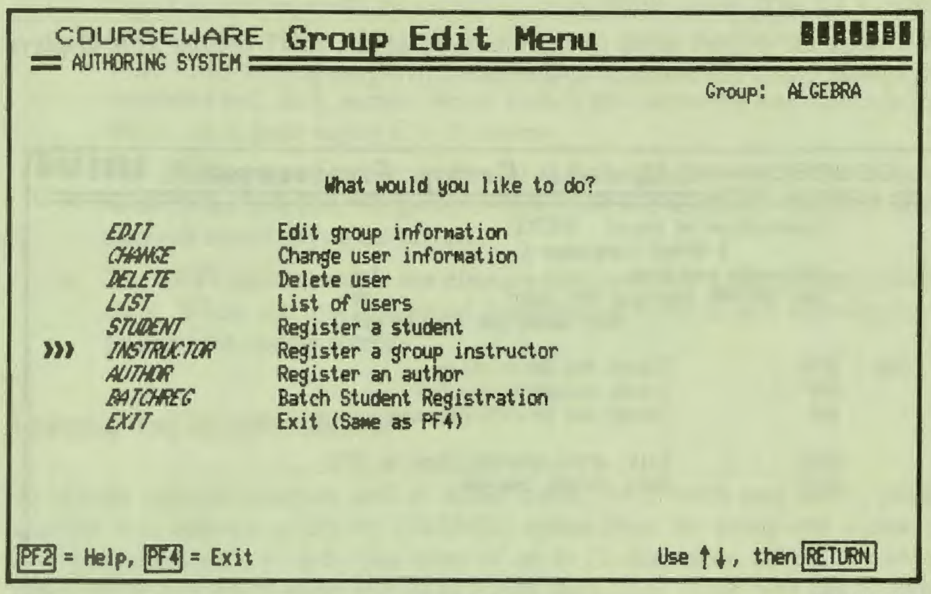
Figure 2–2
Group Instructor Menu And Options

The functions of the options in the group instructor menu are as follows:

- The UPDATE option selects the group edit menu of options for editing the group information.
- The REPORTS option selects the menu of available reports.
- The ASSIGNMENT option selects the group assignment menu of options for listing or changing the group assignments.
- The BROWSE option selects the menu of lessons to browse through.
- The MAIL option accesses the VAX/VMS MAIL utility that lets you send and receive system mail.
- The EXIT option (or PF4 key) allows you to exit from C.A.S. and return to VAX/VMS command level.

## CHANGING THE GROUP: UPDATE

To change the group environment, to change individual user information, or to register group members, select the UPDATE option from the instructor menu. This displays the group edit menu shown in Figure 2–3.



```
COURSEWARE Group Edit Menu
AUTHORING SYSTEM
                                                   Group:  ALGEBRA

                        What would you like to do?

        EDIT          Edit group information
        CHANGE        Change user information
        DELETE        Delete user
        LIST          List of users
        STUDENT       Register a student
  >>>   INSTRUCTOR    Register a group instructor
        AUTHOR        Register an author
        BATCHREG      Batch Student Registration
        EXIT          Exit (Same as PF4)

PF2 = Help, PF4 = Exit                          Use ↑↓, then RETURN
```

MR-S-3952-85

Figure 2–3
Group Edit Menu

Options from the group edit menu display screens that allow you to make specific changes. Figure 2–4 shows the screens that you access by selecting options from the group edit menu.

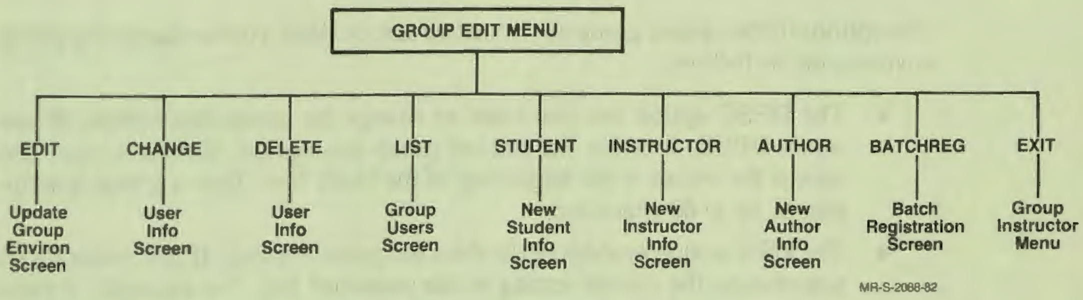
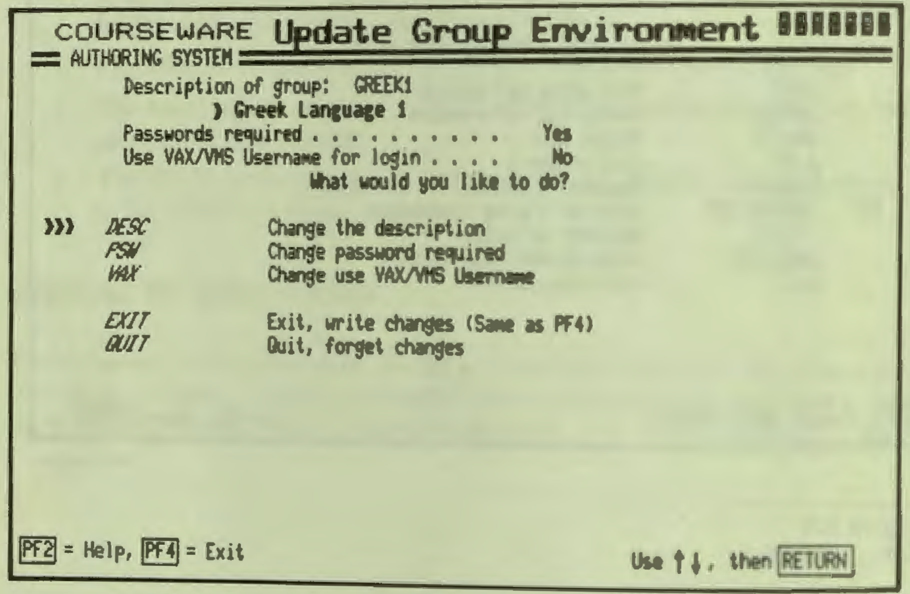


Figure 2–4
Group Edit Menu Options And Screens

To change the current group environment, select the EDIT option. This displays the update group environment screen shown in Figure 2–5.



```
COURSEWARE Update Group Environment
AUTHORING SYSTEM
        Description of group:  GREEK1
               ) Greek Language 1
        Passwords required . . . . . . . . . .    Yes
        Use VAX/VMS Username for login . . . .    No
                               What would you like to do?

 )))  DESC          Change the description
      PSW           Change password required
      VAX           Change use VAX/VMS Username

      EXIT          Exit, write changes (Same as PF4)
      QUIT          Quit, forget changes




PF2 = Help, PF4 = Exit                        Use ↑↓, then RETURN
```

MR-S-3953-85

Figure 2–5
Update Group Environment Screen

The options in the update group environment screen allow you to change the group environment as follows:

- The DESC option lets you enter or change the group description. If you select DESC, it erases the current group description, if there is one, and moves the cursor to the beginning of the blank line. Type a group description of up to 60 characters.
- The PSW option enables or disables the password flag. If you select PSW, you reverse the current setting of the password flag. For example, if passwords are required, selecting PSW disables the flag so that group members are no longer prompted for a password.

- The VAX option enables or disables the VAX/VMS user name translation flag. This option works the same way as the PSW option. The VAX option reverses the setting of the VAX/VMS user name translation flag. When VAX/VMS user name translation is enabled, VAX/VMS user names are translated to C.A.S. names. When VAX/VMS user name translation is disabled, users must supply C.A.S. names.
- The EXIT option retains any changes that you made to the group environment. When you exit, the group environment is changed to match the environment displayed on the screen.
- The QUIT option cancels any changes that you made to the group environment. When you quit, the group environment remains as it was when you selected the update screen.

## Changing User Records: CHANGE

To change user information such as actual name, VAX/VMS user name, phone number, and address, select the CHANGE option from the group edit menu. A prompt appears for a C.A.S. user name of up to 12 characters. (If you decide to return to the group edit menu instead of typing the C.A.S. name, you can do so by pressing the RET key.) After you enter the C.A.S. name, the screen displays information about the instructor, author, or student you selected. (NOTE: You cannot change information for a user while that user is running C.A.S.)

Figure 2–6 shows a screen of information about the instructor. The author information screen is similar. The student information screen includes an additional option, DELRST, for deleting student restart files.

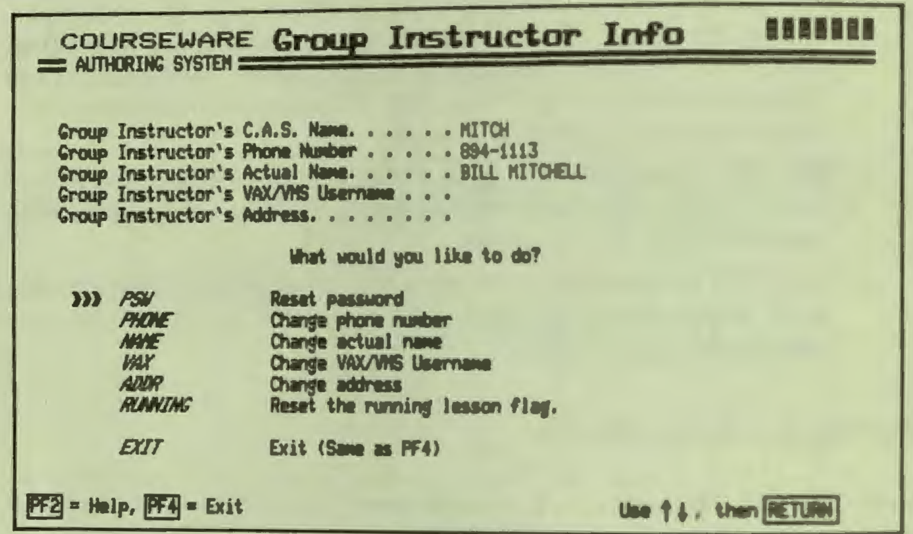

```
COURSEWARE Group Instructor Info
== AUTHORING SYSTEM ==

Group Instructor's C.A.S. Name. . . . . . MITCH
Group Instructor's Phone Number . . . . . 894-1113
Group Instructor's Actual Name. . . . . . BILL MITCHELL
Group Instructor's VAX/VMS Username . . .
Group Instructor's Address. . . . . . . .

                      What would you like to do?

 ))) PSW              Reset password
     PHONE            Change phone number
     NAME             Change actual name
     VAX              Change VAX/VMS Username
     ADDR             Change address
     RUNNING          Reset the running lesson flag.

     EXIT             Exit (Same as PF4)

PF2 = Help, PF4 = Exit                         Use ↑↓, then RETURN
```
MR-S-3631-84

Figure 2–6
Group Instructor Information Screen

The instructor and author information screens display the user's C.A.S. name, phone number, actual name, VAX/VMS user name, and address, with options to change any of these except the C.A.S. name. The student information screen displays the student's C.A.S. name, actual name, and VAX/VMS user name, with options to change the actual name and VAX/VMS user name. The student screen also displays a DELRST option to delete the student's restart files.

You cannot change a C.A.S. name after registration. Change the VAX/VMS user name entry only if there is a discrepancy between the user's actual VAX/VMS user name and the name that is shown. A discrepancy can result if the VAX/VMS system manager changes the VAX/VMS user name, or if a name is typed incorrectly.

If you select DELRST in the student information screen, you are prompted for a lesson name. You can enter the name of a specific lesson for which you want to delete the student's restart file, or an asterisk (*) to delete all of the student's restart files. To exit the DELRST option without deleting any restart files, simply press RET at the lesson name prompt.

In addition to the options for changing information in instructor, author, and student records, the PSW and RUNNING options are available for changing the password and resetting the running lesson flag.

When you select PSW, users are prompted to enter new passwords the next time they run C.A.S. Select RUNNING if you want to disable the running lesson flag so that the user can recover from a system failure and continue running C.A.S. (The running lesson flag is enabled when users run lessons and reports.)

**Note:** Two users in the same group who share the same C.A.S. name cannot run C.A.S. at the same time.

To change an item of information in an instructor, author, or student record, select the option that corresponds to the information that you want to change. For example, if you want to change an author's phone number, select the PHONE option. The current phone number is erased from the screen, and the cursor is positioned at the beginning of the blank phone number line. Type a new phone number. The same thing happens if you select any of the other options: the current information is erased, and the cursor is positioned at the beginning of the blank line so that you can enter the new information.

The maximum number of characters allowed is:

| | |
|---|---|
| • Phone number | - 13 |
| • Actual name | - 20 |
| • VAX/VMS user name | - 38 |
| • Address | - 40 |

If you select the EXIT option or press PF4, you return to the group edit menu.

## Deleting A User: DELETE

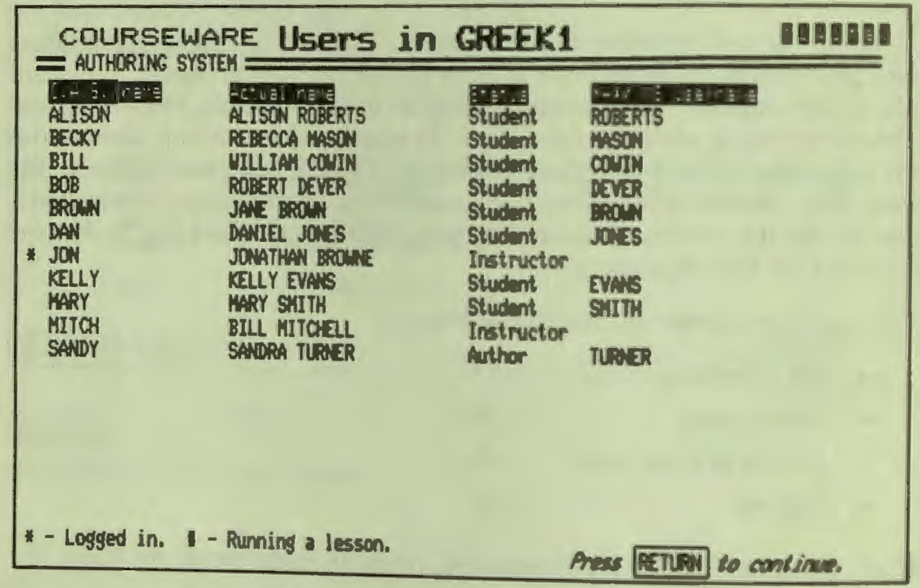
To delete a user from the group, select the DELETE option from the group edit menu. You receive a prompt for the user's C.A.S. name. Type the name and press RET. (If you decide to return to the group edit menu instead of typing a C.A.S. name, press the RET key.) The information screen for the instructor, author, or student with that C.A.S. name appears. At the bottom of the information screen, the system requests confirmation:

```
Ok to delete? (Y or N)
```

If you type Y, the user is deleted; and you return to the group edit menu. If you type N, the user is not deleted; and you return to the group edit menu.

## Listing Group Members: LIST

To list all the members of a group and their C.A.S. names, actual names, status (user type), and VAX/VMS user names, select the LIST option from the group edit menu. A typical group users screen is shown in Figure 2–7.



```
COURSEWARE Users in GREEK1
= AUTHORING SYSTEM ==========================================

 C.A.S. name    Actual name        Status      VAX/VMS username
 ALISON         ALISON ROBERTS     Student     ROBERTS
 BECKY          REBECCA MASON      Student     MASON
 BILL           WILLIAM COWIN      Student     COWIN
 BOB            ROBERT DEVER       Student     DEVER
 BROWN          JANE BROWN         Student     BROWN
 DAN            DANIEL JONES       Student     JONES
* JON           JONATHAN BROWNE    Instructor
 KELLY          KELLY EVANS        Student     EVANS
 MARY           MARY SMITH         Student     SMITH
 MITCH          BILL MITCHELL      Instructor
 SANDY          SANDRA TURNER      Author      TURNER




* - Logged in.  ‡ - Running a lesson.
                                      Press RETURN to continue.
```

MR-S-3954-85

Figure 2–7
Group Users Screen

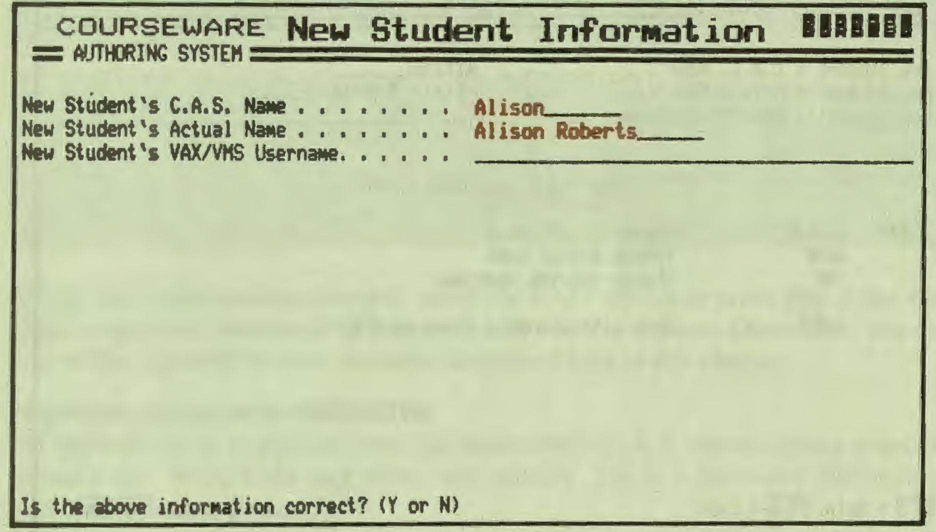
If the list fills more than one screen, press RET to continue the display on subsequent screens. (Pressing the NEXT and + PAGE keys also displays subsequent screens.) When the list ends, press RET or PF4 to return to the group edit menu.

## Registering New Group Members

To register an individual student, instructor, or author in a group, select the STUDENT, INSTRUCTOR, or AUTHOR option from the group edit menu. To register several students at one time, select the BATCHREG option. Depending on the option you select, a new information screen appears for student, instructor, author, or batch student registration.

## Registering A Student: STUDENT

To register a new student individually, enter the student's C.A.S. name, actual name, and VAX/VMS user name. A C.A.S. name is required for successful registration. An actual name, while not required, is nevertheless useful since many reports include this field. You must supply a VAX/VMS user name if your group is using VAX/VMS user name translation. The new student information screen is shown in Figure 2–8.



```
New Student's C.A.S. Name . . . . . . . .  Alison______
New Student's Actual Name . . . . . . . .  Alison Roberts______
New Student's VAX/VMS Username. . . . . .  ______________________________________

Is the above information correct? (Y or N)
```

MR-S-3955-85

Figure 2–8
New Student Information Screen

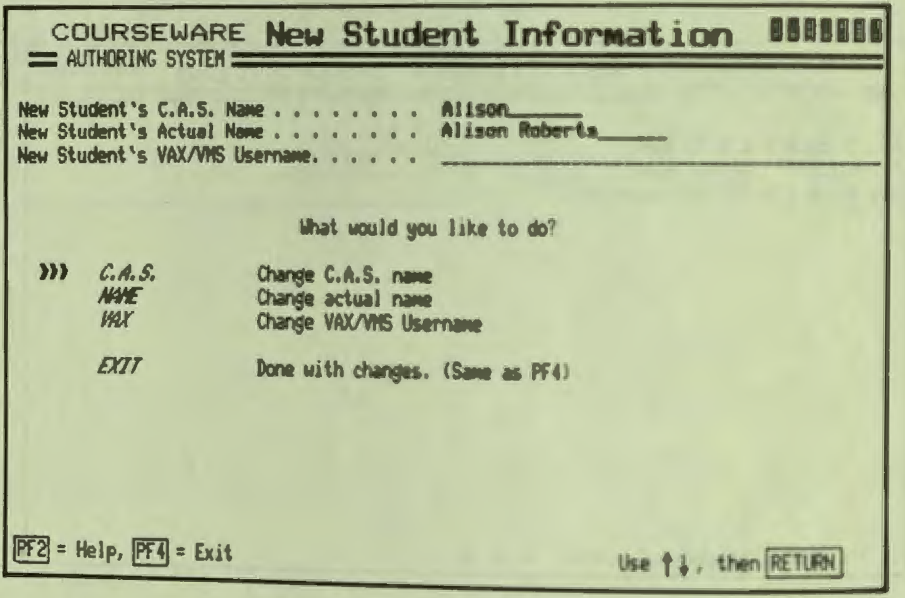
Type the C.A.S. name, actual name, and VAX/VMS user name. The maximum number of characters allowed is:

| | |
|---|---|
| • C.A.S. name | - 12 |
| • Actual name | - 20 |
| • VAX/VMS user name | - 38 |

Press RET after each entry. When you finish, you see the following prompt:

```
Is the above information correct? (Y or N)
```

If you type Y to confirm that the information is correct, the system either registers the student and redisplays the group edit menu, or displays one of the registration error messages described later in this chapter. If you type N to indicate that the information is incorrect, additional options appear that allow you to correct the information. The new student information screen with change options is shown in Figure 2–9.



```
COURSEWARE New Student Information
AUTHORING SYSTEM

New Student's C.A.S. Name . . . . . . . . .  Alison
New Student's Actual Name . . . . . . . . .  Alison Roberts
New Student's VAX/VMS Username. . . . . .

                    What would you like to do?

 >>>  C.A.S.       Change C.A.S. name
      NAME         Change actual name
      VAX          Change VAX/VMS Username

      EXIT         Done with changes. (Same as PF4)

PF2 = Help, PF4 = Exit                        Use ↑↓, then RETURN
```

MR-S-3956-85

Figure 2–9
New Student Information Screen With Change Options

Each of the options for changing the student information is described below.

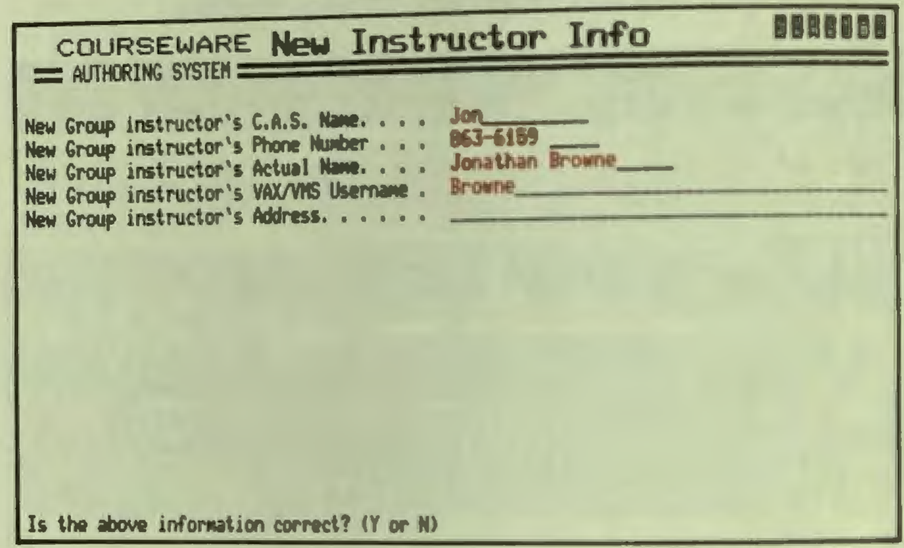
## Table 2–1. New Student Information Change Options

| C.A.S. OPTION | CHANGE ITEM | C.A.S. RESPONSE (What you should type/enter) |
|---|---|---|
| C.A.S. | C.A.S. Name | Current name is erased. Enter new C.A.S. name. |
| NAME | Actual Name | Current name is erased. Enter new student name. |
| VAX | VAX/VMS User Name | Current user name is erased. Enter new VAX/VMS user name. Use to correct a discrepancy between the actual VAX/VMS user name and the user name shown. |

When you finish making changes, select the EXIT option or press PF4. If the student is registered successfully, the group edit menu reappears. Otherwise, you see one of the registration error messages described later in this chapter.

**Registering An Instructor: INSTRUCTOR**

To register a new instructor, enter the instructor's C.A.S. name, phone number, actual name, VAX/VMS user name, and address. The new instructor information screen is shown in Figure 2–10.


```
COURSEWARE New Instructor Info
AUTHORING SYSTEM

New Group instructor's C.A.S. Name. . . .  Jon
New Group instructor's Phone Number . . .  863-6159
New Group instructor's Actual Name. . . .  Jonathan Browne
New Group instructor's VAX/VMS Username .  Browne
New Group instructor's Address. . . . . .

Is the above information correct? (Y or N)
```

MR-S-3957-85

Figure 2–10
New Instructor Information Screen

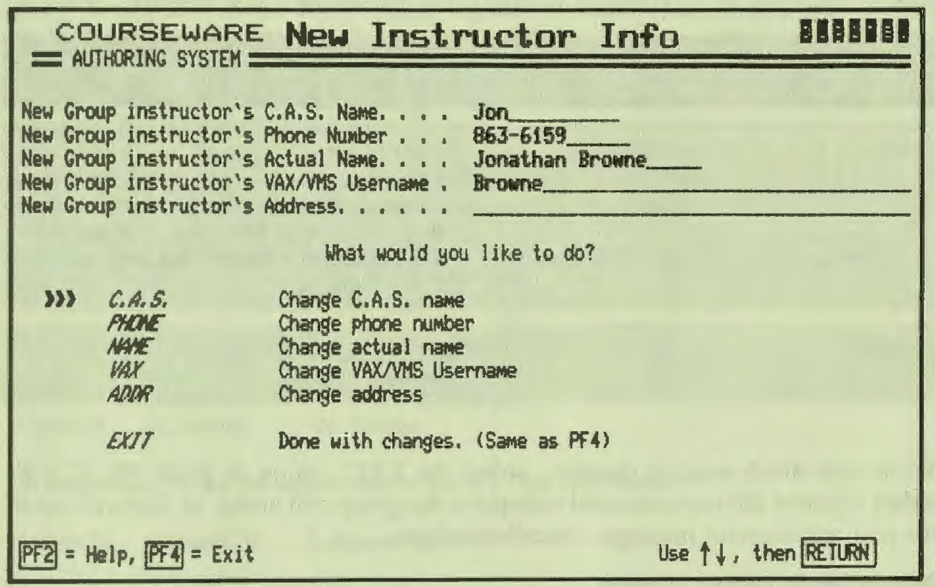
Type the C.A.S. name, phone number, actual name, VAX/VMS user name, and address. The maximum number of characters allowed is:

| | |
|---|---|
| • C.A.S. name | - 12 |
| • Phone number | - 13 |
| • Actual name | - 20 |
| • VAX/VMS user name | - 38 |
| • Address | - 40 |

Press RET after each entry. When you press RET after typing the address, you are prompted to confirm the information:

Is the above information correct? (Y or N)

If you type Y to indicate that the information is correct, the system either registers the instructor and redisplays the group edit menu, or displays one of the registration error messages described later in this chapter. If you type N to indicate that the information is incorrect, additional options appear to allow you to correct the information. The new instructor information screen with change options is shown in Figure 2–11.



```
COURSEWARE New Instructor Info
AUTHORING SYSTEM

New Group instructor's C.A.S. Name. . . .  Jon
New Group instructor's Phone Number . . .  863-6159
New Group instructor's Actual Name. . . .  Jonathan Browne
New Group instructor's VAX/VMS Username .  Browne
New Group instructor's Address. . . . . .

                    What would you like to do?

 >>>  C.A.S.          Change C.A.S. name
      PHONE           Change phone number
      NAME            Change actual name
      VAX             Change VAX/VMS Username
      ADDR            Change address

      EXIT            Done with changes. (Same as PF4)

PF2 = Help, PF4 = Exit                            Use ↑↓, then RETURN
```

MR-S-3958-85

Figure 2–11
New Instructor Information Screen With Change Options

Each of the options for changing the instructor information is described below.

## Table 2–2. New Instructor Information Change Options

| C.A.S. OPTION | CHANGE ITEM | C.A.S. RESPONSE (What you should type/enter) |
|---|---|---|
| C.A.S. | C.A.S. Name | Current name is erased. Enter new C.A.S. name. |
| PHONE | Phone Number | Current phone number is erased. Enter new phone number. |
| NAME | Actual Name | Current name is erased. Enter new instructor name. |

Table 2–2. New Instructor Information Change Options (Cont.)

| C.A.S. OPTION | CHANGE ITEM | C.A.S. RESPONSE (What you should type/enter) |
|---|---|---|
| VAX | VAX/VMS User Name | Current user name is erased. Enter new VAX/VMS user name. Change the VAX/VMS user name entry only if there is a between the actual VAX/VMS user name and the user name that is shown. |
| ADDR | Address | Current address is erased. Enter new address. |

When you finish making changes, select the EXIT option or press PF4. C.A.S. either registers the instructor and redisplays the group edit menu, or displays one of the registration error messages described below.

**Registering An Author: AUTHOR**

Refer to the preceding section on registering instructors. The screens and the registration procedure for instructors and authors are the same.

**Registration Error Messages**

After you type the information required for registering a student, instructor, or author, and then type Y in response to the confirmation message described above, you may see one of the following error messages:

- You must specify a C.A.S. name.
  Change the above information? (Y or N)

  This message appears if you fail to supply a C.A.S. name. If you type Y in response to the message to indicate that you want to supply a name, the change options are added to the user information screen. You can then select the C.A.S. option and type a C.A.S. name. If you type N, no registration occurs, and you return to the group edit menu.

- Duplicate C.A.S. name. Registration unsuccessful.
  Change the above information? (Y or N)

  This message is displayed if you enter a C.A.S. name already assigned to another member of the group. If you type Y in response to the message, the change options are added to the user information screen. You can then select the C.A.S. option and type another C.A.S. name. If you type N, no registration occurs, and you return to the group edit menu.

**Registering a List of Students: BATCHREG**

The BATCHREG option allows you to register several students at one time. Before you can use BATCHREG, you must use an editor to create a file that lists the students in the format required by the BATCHREG option.

The required record format for the file is:

| | |
|---|---|
| C.A.S. name | = 12 characters maximum, beginning in column 1, followed by a space or tab character |
| VAX/VMS user name | = 38 characters maximum, followed by a space or tab character |
| Real name | = 20 characters maximum |

```
student20   student20    Marianne Johnson
student21   student21    Jon Richter
!
! A single tab character separates the fields in the entry below.
!
student22	studentl22	Nathaniel Brown
```

You must use a single tab character or space to separate the fields in an entry. An entry can be a maximum of 72 spaces long, including the tab characters or spaces used to delimit the fields. Since spaces delimit the fields, the C.A.S. name and VAX/VMS user name cannot contain spaces. The real name can include spaces.

The only required field is the C.A.S. name. This can contain only alphanumeric characters, up to 12 in a name. If the format of a C.A.S. name is invalid, the student will not be registered.

The VAX/VMS user name and real name are optional fields. If the VAX/VMS user name is longer than 38 characters, the system truncates the field and produces a warning. Entering a real name that is longer than 20 characters produces the same result.

You can include comments and blank lines in the file. The first character of a blank line or comment line must be an exclamation point (!).

When you choose BATCHREG in the group edit menu, you are prompted for the name of the file that contains the student list. If the file is in your default directory, you enter the file name only. If the file is in another directory, you must enter the complete file specification, including disk and directory names, as shown in the sample screen in Figure 2–12.

The system registers all students who are listed in the file with correctly formatted and valid entries.

The BATCHREG screen includes the following question:

Do you want a listing file?

If you type a Y, BATCHREG produces an output file that indicates which registrations were successful. The output file uses the same format as the input file, but includes a letter code in columns 71–72 indicating the registration status for each record as follows:

1 G = Good; registration successful
2 D = Duplicate record
3 N = Not registered; error within this record
4 T = Truncated; VAX/VMS user name was longer than 38 characters, and/or real name was longer than 20 characters

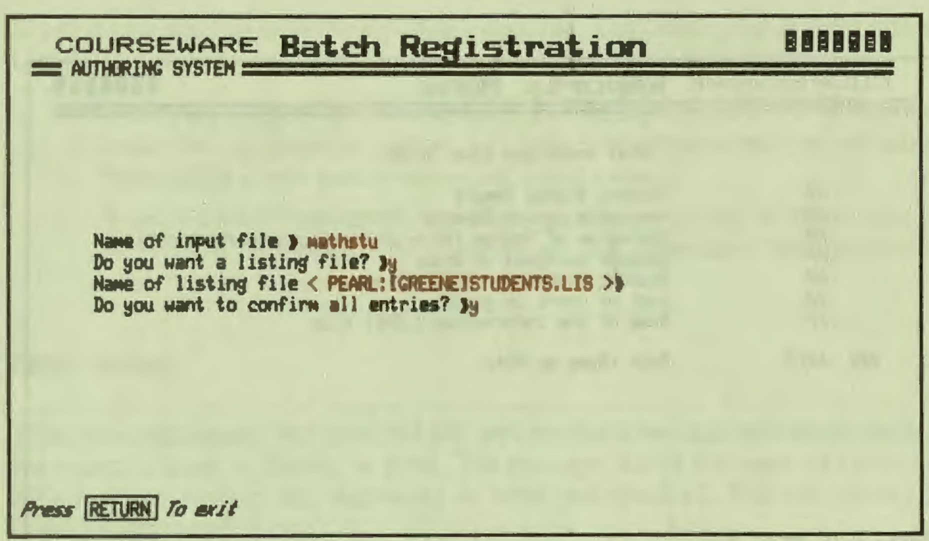
If there are any invalid entries, edit the output file to correct them. You can then rerun the BATCHREG option, submitting the edited output file as your input file.

The BATCHREG screen also allows you to display confirmed registrations on the screen. You see the following question:

Do you want to confirm all entries?

Type Y if you want valid registrations to be confirmed with screen output. A summary report that records successful and unsuccessful responses appears when batch registration is complete.

The batch registration screen is shown in Figure 2–12.



```
Name of input file > mathstu
Do you want a listing file? >y
Name of listing file < PEARL:[GREENE]STUDENTS.LIS >>
Do you want to confirm all entries? >y
```

MR-S-3627-84

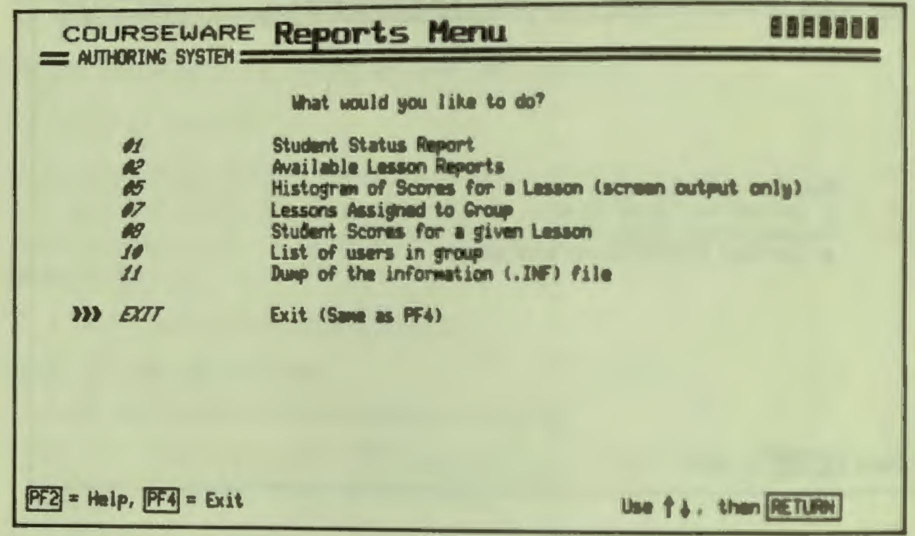
Figure 2–12
Batch Registration Screen

### Ending The Group Edit: EXIT

To exit from the group edit menu and return to the group instructor menu, select the EXIT option or press PF4.

## SELECTING A REPORT: REPORTS

To display or print a report, select the REPORTS option from the group instructor menu. This displays the reports menu shown in Figure 2–13. To exit from the reports menu and return to the group instructor menu, select the EXIT option or press PF4.



```
COURSEWARE Reports Menu
AUTHORING SYSTEM

                 What would you like to do?

      #1          Student Status Report
      #2          Available Lesson Reports
      #5          Histogram of Scores for a Lesson (screen output only)
      #7          Lessons Assigned to Group
      #8          Student Scores for a given Lesson
      10          List of users in group
      11          Dump of the information (.INF) file

  >>> EXIT        Exit (Same as PF4)

PF2 = Help, PF4 = Exit                          Use ↑↓, then RETURN
```

MR-S-3636-84

Figure 2–13
Reports Menu

## Report File Specification

When you select a report, you must supply the necessary report parameters in response to the system prompts. You must also specify where you want to store the report. For every report, you receive the following file specification prompt:

WHERE SHOULD THE REPORT GO? (DEFAULT: *default*.LIS)

You can store the report in the default file indicated in the prompt or in a file that you specify. Indicate where to store the report as follows:

- To store the report in the default file, press RET in response to the file specification prompt. (The system places the default file containing your report in your default directory.)
- To store the report in another file, type the file specification and press RET. If you do not specify a file extension, the system provides the extension .LIS.

If you do not want to store the report in a disk file, you can display or print it from C.A.S. as follows:

- To display a report on your monitor, type the logical name TT: in response to the file specification prompt. TT: defines which terminal you are using. Press CTRL/O if you want to terminate report output.
- To print a hard-copy report, type the logical name for any of the printers on the system. Although logical names are site dependent, the line printer often has the logical name LPA0:.

## Report Output

After you store a report in a specified file, you receive a message informing you that the report is ready to display or print. The message shows the name of either the default file or another file, depending on what you specified. You can display or print the report as follows:

- To print a copy of a report that is displayed only on the monitor (report 05), press the SHIFT and PF1 keys simultaneously (or the SHIFT AND PRINT SCREEN keys on VT240 series terminals) while the report is displayed on your monitor. (You can do this only if your terminal is connected to a hard-copy graphics printer such as an LA34VA.)
- To display the report on your monitor, issue a TYPE command at VAX/VMS command level after you exit from C.A.S.
- To print a hard-copy of the report, issue a PRINT command containing the specified file name at VAX/VMS command level after you exit from C.A.S.

## Report Descriptions

Reports show data only for the group you belong to. The reports that are available to you are described below.

## Table 2–3: Report Descriptions

| REPORT NAME | OPTION/REPORT NUMBER | DESCRIPTION |
|---|---|---|
| Student Status | 01 | Shows the lesson completion status (P = partially done, D = done, N = not started) of all students in the group, for one or all lesson assignments. |
| Available Lessons | 02 | Lists the name, description, and author of all published lessons that are available to your group. |
| Histogram of Scores For a Lesson | 05 | Shows a histogram of the percentage distribution of scores and the number of students receiving them, for a lesson. |
| Lessons Assigned To Group | 07 | Shows the name and description of every lesson assigned to your group. A comment line at the end of the report indicates whether lessons are taken in structured or unstructured order. |
| Student Scores For A Given Lesson | 08 | Shows student scores for a lesson. |
| List Of Users In Group | 10 | Shows the actual name, C.A.S. name, VAX/VMS user name, and user type of every user in your group. |
| List Of The Information (.INF) File | 11 | Shows a formatted list of the group information file. The group information file contains the group environment and user information. |

## Report Parameters

The following pages contain examples of what is displayed on the monitor when you select a report. Each example shows the parameters that C.A.S. prompts for, and is followed by a sample report.

The system displays a cursor, usually a rectangle, to prompt for information. When a prompt for a lesson or unit name appears, type the name and press RET. When offered the ALL or QUIT options, type ALL or QUIT in full. ALL selects all lessons or units; QUIT cancels the report and redisplays the reports menu. When you are prompted for a numbered option, type the number and press RET.

### (01) Student Status Report

When you select this report, you produce the following display:

```
DEFINE SELECTION CRITERIA FOR STUDENT STATUS REPORT
SELECT SORT OPTION
 1. STUDENT NAME
 2. C.A.S. NAME
 3. STATUS, STUDENT NAME
 4. EXIT FROM REPORT
1 RET
ENTER LESSON NAME (9 CHARS) OR "QUIT" OR "ALL"
ALL RET
STATUS RECORDS BEING SORTED

WHERE SHOULD THE REPORT GO? (DEFAULT: CASRPT01.LIS)
RET
STUDENT STATUS REPORT IS READY FOR PRINTING – CASRPT01.LIS

PRESS "RETURN" TO CONTINUE
```

```
RPT01                          STUDENT  STATUS  REPORT                    PAGE:   1
DATE:  3/30/82                  FOR  GROUP: ENG1                    TIME: 07:34:19
                                   LESSON: SQUARS

          STUDENT NAME                      C.A.S.  NAME       STATUS
        BARRY BROWN                         BARRY                N
        BILL BROWN                          BILL                 N
        DAVE GREEN                          DAVE                 N
        DEANNA KNAUF                        DEANNA               D
        ELIZABETH                           ELIZABETH            N

        JANET                               JANET                N
        JEFFERY                             JEFFREY              N
        JULIE NIELSEN                       JULIE                D
        MIKE BROWN                          MIKE                 D
        RICH                                RICH                 N

        TIM SMITH                           TIM                  N

              STATUS                                COUNTS
         D = DONE                                      3

         P = PARTIALLY DONE

         N = NOT STARTED                               8

                                                                      MR-0-2058-82
```

Figure 2–14
(01) Student Status Report

**(02) Available Lessons Report**

When you select this report, you see the following:

```
BEGINNING SELECTION OF AVAILABLE LESSONS FOR GROUP groupname
SORT LESSON INFORMATION BY:
 1. LESSON NAME
 2. AUTHOR
 3. EXIT FROM THE REPORT
PLEASE ENTER OPTION DESIRED
1 (RET)

WHERE SHOULD THE REPORT GO? (DEFAULT: CASRPT02.LIS)
(RET)
AVAILABLE LESSONS LIST IS READY TO BE PRINTED – CASRPT02.LIS

PRESS "RETURN" TO CONTINUE
```

```
RPT02                               LESSONS AVAILABLE TO ENGLISH                                   PAGE:   1
DATE:   9/23/85                                                                             TIME: 16:09:37

LESSON NAME          LESSON DESCRIPTION                                          AUTHOR NAME  &  GROUP
ANALOGIES  WORD RELATIONSHIPS                                                     CAROL          ENGLISH
ANTONYMS   WORD OPPOSITES                                                         CAROL          ENGLISH
APOS       APOSTROPHE USAGE                                                       SCOTT          ENGLISH
COMPLEX    THE COMPLEX SENTENCE                                                   CAROL          ENGLISH
DALCAI     INTRODUCTION TO DAL (VERSION 1.5)                                      DIGITAL        SYSTEM

GRAMMAR    GRAMMAR AND USAGE                                                      CAROL          ENGLISH
JUDGER     DEMONSTRATES JUDGING CRITERIA                                          SCOTT          ENGLISH
LETTER     ALPHABET TEST                                                          SCOTT          ENGLISH
MODIFIER   MISPLACED MODIFIERS                                                    CAROL          ENGLISH
NEWWORDS   WORD MATCHING EXERCISE                                                 DARRELL        ENGLISH

PARTS      PARTS OF SPEECH                                                        CAROL          ENGLISH
PREFIX     WORD PREFIXES                                                          CAROL          ENGLISH
SIMPLE     THE SIMPLE SENTENCE                                                    CAROL          ENGLISH
SPELLING   SPELLING DRILL                                                         LIZ            ENGLISH
SUFFIX     WORD SUFFIXES                                                          CAROL          ENGLISH

VERB       STUDY OF VERB TENSES                                                   LIZ            ENGLISH
```
MR-S-3989-85

Figure 2–15
(02) Available Lessons Report

**(05) Histogram Of Scores For A Lesson**

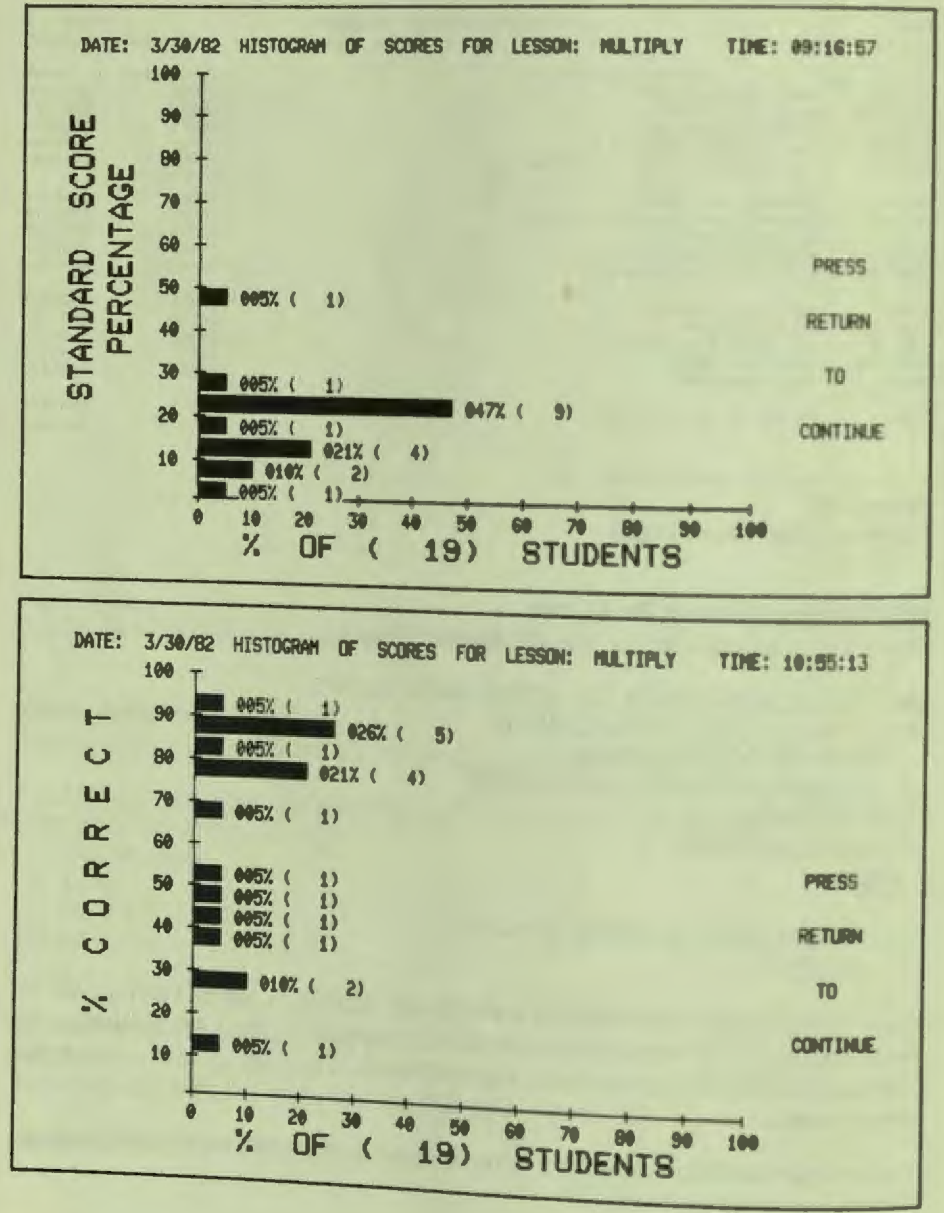
When you select this report, you display the following:

```
DEFINE SELECTION CRITERIA FOR HISTOGRAM OF SCORES
SELECT TYPE OF SCORE TO BE CHARTED
 1. STANDARD SCORES (WEIGHTED)
 2. PERCENT OF CORRECT OVER QUERIES
 3. BOTH (OPTION 1 & 2)
 4. EXIT FROM REPORT
3 (RET)

ENTER LESSON NAME (9 CHARS) OR "QUIT"
MULTIPLY (RET)
```

If you select a report showing both kinds of score (option 3), the first display shows the histogram of the standard score percentage (option 1). Press (RET) to display the histogram of the correct score percentage (option 2). Press (RET) again to return to the reports menu.

If you request the report and there are no scores recorded, you receive the message "There are no scores to report."

This report is displayed on the monitor only.



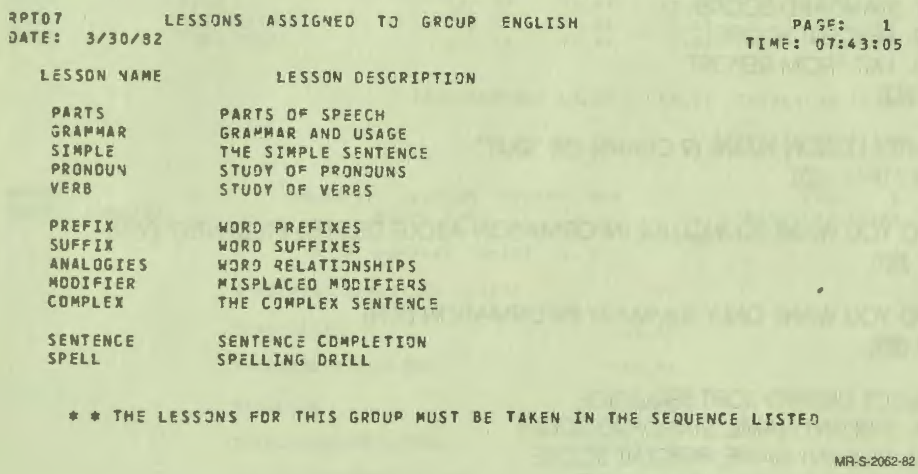
MR-S-2060-82

Figure 2–16
(05) Histogram Of Scores For A Lesson

**(07) Lessons Assigned To Group**

When you select this report, you see the following:

```
BEGINNING REPORT OF LESSONS ASSIGNED TO YOUR GROUP

ENTER "QUIT" TO EXIT FROM THE REPORT OR PRESS "RETURN" TO
CONTINUE
RET

WHERE SHOULD THE REPORT GO? (DEFAULT: CASRPT07.LIS)
RET

LIST OF LESSONS ASSIGNED TO GROUP groupname READY —
CASRPT07.LIS

PRESS "RETURN" TO CONTINUE
```



```
RPT07         LESSONS  ASSIGNED  TO  GROUP  ENGLISH                PAGE:   1
DATE:  3/30/82                                                TIME: 07:43:05

   LESSON NAME          LESSON DESCRIPTION

    PARTS              PARTS OF SPEECH
    GRAMMAR            GRAMMAR AND USAGE
    SIMPLE             THE SIMPLE SENTENCE
    PRONOUN            STUDY OF PRONOUNS
    VERB               STUDY OF VERBS

    PREFIX             WORD PREFIXES
    SUFFIX             WORD SUFFIXES
    ANALOGIES          WORD RELATIONSHIPS
    MODIFIER           MISPLACED MODIFIERS
    COMPLEX            THE COMPLEX SENTENCE

    SENTENCE           SENTENCE COMPLETION
    SPELL              SPELLING DRILL

      * * THE LESSONS FOR THIS GROUP MUST BE TAKEN IN THE SEQUENCE LISTED
```

MR-S-2062-82

Figure 2–17
(07) Lessons Assigned To Group

**(08) Student Scores For A Given Lesson**

When you select this report, you select either standard scores or percent scores. Based on the kind of scores you select, the report shows a mean percent score, a standard deviation figure, and a variance figure. The formula for each of these follows:

1 Mean = the accumulation of scores (either % right or weighted) divided by the number of students

2 Variance = $\frac{\Sigma (X^2)}{n}$

Where: x = each student's score minus the mean
n = number of students

3 Standard Deviation = $\sqrt{\frac{\Sigma (X^2)}{n}}$

Where: x = each student's score minus the mean
n = number of students

When you select this report, you display the following:

```
BEGINNING SELECTION OF STUDENT SCORES
SELECT SCORE TO BE USED FOR RANK AND CALCULATIONS
 1. STANDARD SCORE
 2. PERCENT SCORE
 3. EXIT FROM REPORT
1 (RET)

ENTER LESSON NAME (9 CHARS) OR "QUIT"
MULTIPLY (RET)

DO YOU WANT TO INCLUDE INFORMATION ABOUT DELETED STUDENTS? (Y/N)
Y (RET)

DO YOU WANT ONLY SUMMARY INFORMATION (Y/N)
N (RET)

SELECT DESIRED SORT SEQUENCE
 1. STUDENT NAME, STANDARD SCORE
 2. STUDENT NAME, PERCENT SCORE
 3. CLASS RANK, STUDENT NAME
 4. RESPONSE TIME, PERCENT SCORE
1 (RET)

WHERE SHOULD THE REPORT GO? (DEFAULT: CASRPT08.LIS)
(RET)

DO YOU WANT ONLY 1 STUDENT PER PAGE?
 N (RET)

STUDENT SCORES REPORT READY – CASRPT08.LIS

PRESS "RETURN" TO CONTINUE
```

```
RPT08                    STUDENT  SCORES  FOR  LESSON: MULTIPLY              PAGE:   1
DATE:  4/02/82                     GROUP: ENG1                          TIME: 09:32:54
    C.A.S.                                                    %       STD    CLASS   RESPONSE
    NAME              STUDENT NAME         RIGHT/ASKED      SCORE    SCORE    RANK     TIME
                   * *  DELETED  * *          11/ 28        39.29    10.00     12    0:01:32
                   * *  DELETED  * *           8/ 28        28.57     7.00     13    0:01:16
BARBARA            BARBARA WHITE              25/ 28        89.29    24.00      2    0:03:47
BARRY              BARRY BROWN                26/ 28        92.86    25.00      1    0:01:48
CAROLYN            CAROL LYNN BARRON          25/ 28        89.29    24.00      2    0:02:10

DAVE               DAVE GREEN                 15/ 28        53.57    14.00      9    0:02:08
ELIZABETH          ELIZABETH                  21/ 28        75.00    20.00      7    0:01:40
GARY               GARY SWIFT                 19/ 29        65.52    18.00      8    0:01:47
JANE               JANE STONE                 47/ 54        87.04    46.00      3    0:03:03
JEFFREY            JEFFERY                     8/ 28        28.57     7.00     13    0:01:07

DOC                JIM SMITH                  25/ 28        89.29    24.00      2    0:01:57
JOHN               JOHN COBB                  13/ 28        46.43    12.00     10    0:01:06
JULIE              JULIE NIELSEN              25/ 30        83.33    23.00      5    0:02:28
MARY               MARY POTTER                 4/ 28        14.29     3.00     14    0:01:09
MIKE               MIKE BROWN                 24/ 28        85.71    23.00      4    0:01:55

REGINALD           REGINALD GOOD              22/ 28        78.57    21.00      6    0:01:24
RICH               RICH                       22/ 28        78.57    21.00      6    0:01:41
SUE                SUE BLACK                  21/ 28        75.00    20.00      7    0:01:19
TONY               TONY DICE                  12/ 28        42.86    11.00     11    0:01:15

                                       MEAN SCORES: 65.42    18.57   TOTAL: 00:34:32

                                                                   AVERAGE: 00:01:49

RPT08                        MULTIPLY   LESSON  REPORT  FOR                  PAGE:   1
DATE:  4/02/82                     GROUP: ENG1                          TIME: 09:32:54

                              * *  SUMMARY  SHEET  * *

                      STUDENTS COMPLETING LESSON               019

                      MEAN SCORE                             65.42

                      STANDARD DEVIATION                    105.89

                      VARIANCE                              214.38

                      TOTAL RESPONSE TIME                 00:34:32

                      AVERAGE RESPONSE TIME               00:01:49
```

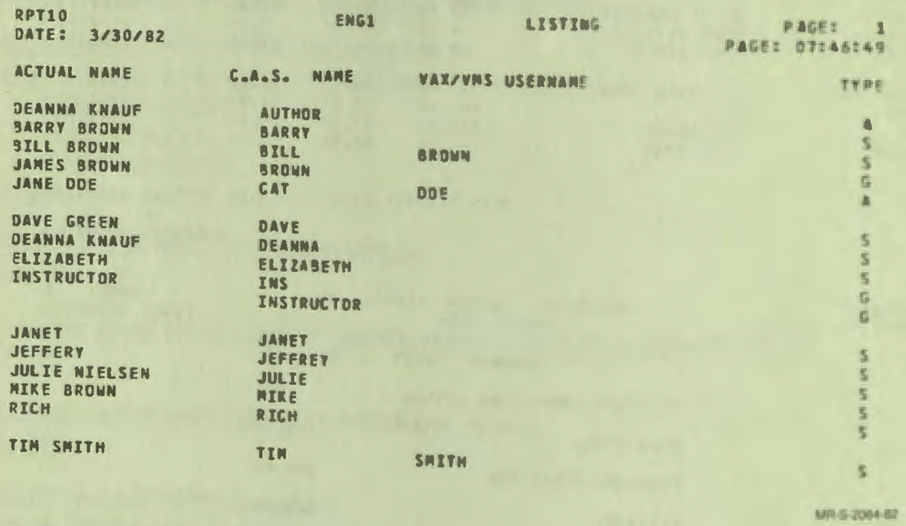
MR-S-2063-82

Figure 2–18
(08) Student Scores For A Given Lesson

**(10) List Of Users In Group**

When you select this report, you display the following:

```
GROUP LISTING REPORT BEGINNING

ENTER "QUIT" TO EXIT FROM REPORT OR PRESS "RETURN" TO
CONTINUE
RET
WHERE SHOULD THE REPORT GO? (DEFAULT: CASRPT10.LIS)
RET
groupname      GROUP MEMBER LISTING READY – CASRPT10.LIS

PRESS "RETURN" TO CONTINUE
```



```
RPT10                         ENG1            LISTING                    PAGE:   1
DATE:  3/30/82                                                    PAGE: 07:46:49

ACTUAL NAME           C.A.S. NAME      VAX/VMS USERNAME                     TYPE

DEANNA KNAUF            AUTHOR                                                A
BARRY BROWN             BARRY                                                 S
BILL BROWN              BILL            BROWN                                 S
JAMES BROWN             BROWN                                                 G
JANE DOE                CAT             DOE                                   A

DAVE GREEN              DAVE                                                  S
DEANNA KNAUF            DEANNA                                                S
ELIZABETH               ELIZABETH                                             S
INSTRUCTOR              INS                                                   G
                        INSTRUCTOR                                            G

JANET                   JANET                                                 S
JEFFERY                 JEFFREY                                               S
JULIE NIELSEN           JULIE                                                 S
MIKE BROWN              MIKE                                                  S
RICH                    RICH                                                  S

TIM SMITH               TIM             SMITH                                 S
```

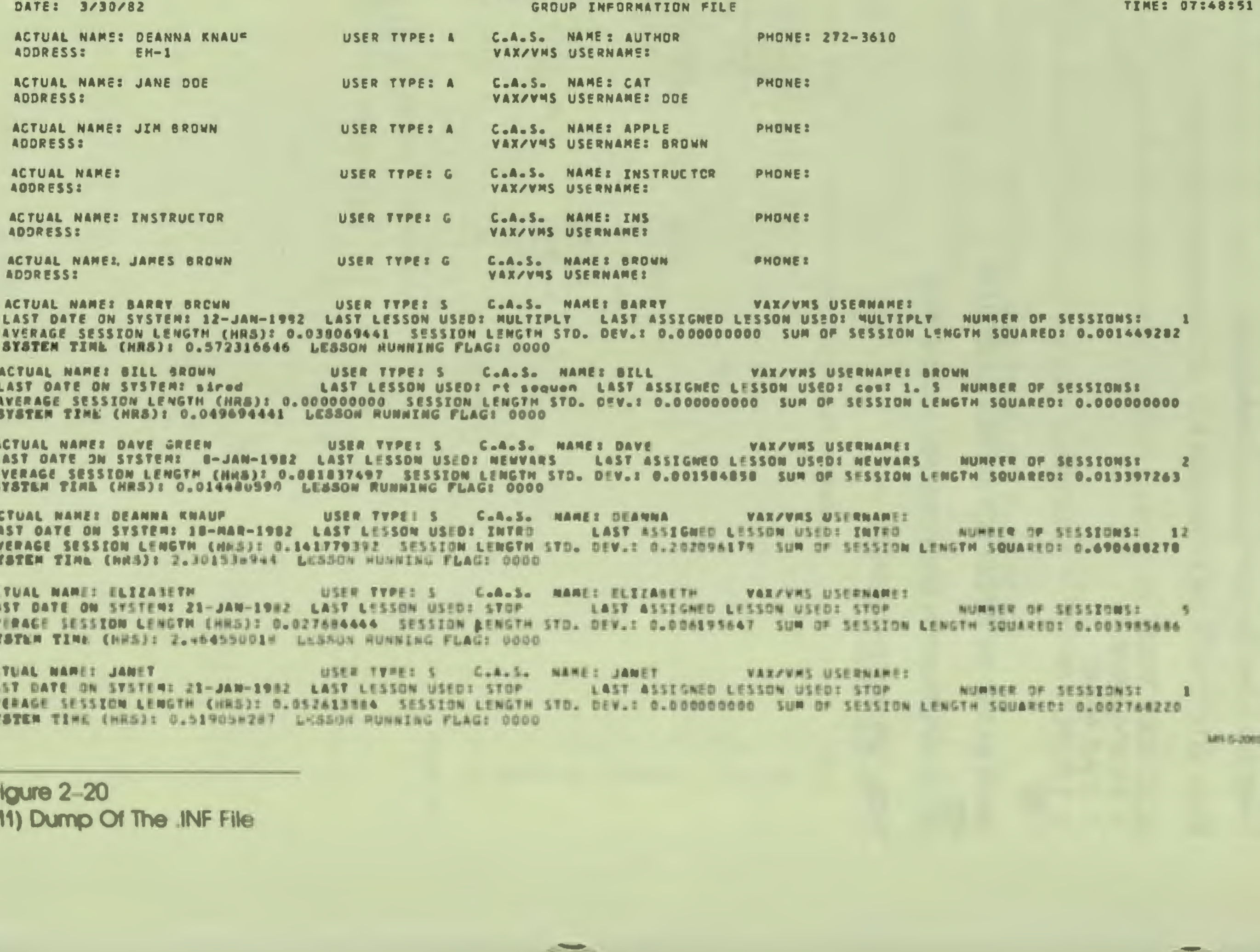
MR-S-2064-82

Figure 2–19
(10) List Of Users In Group

**(11) Dump Of The Information (.INF) File**

When you select this report, you see the following:

```
BEGIN SELECTION OF GROUP MEMBERS FOR INFORMATION REPORT

SELECT SORT OPTION
 1. STUDENT NAME
 2. C.A.S. NAME
 3. EXIT FROM REPORT
1 (RET)

WHERE SHOULD THE REPORT GO? (DEFAULT: CASRPT11.LIS)
(RET)
GROUP INFORMATION REPORT IS READY — CASRPT11.LIS

PRESS "RETURN" TO CONTINUE
```



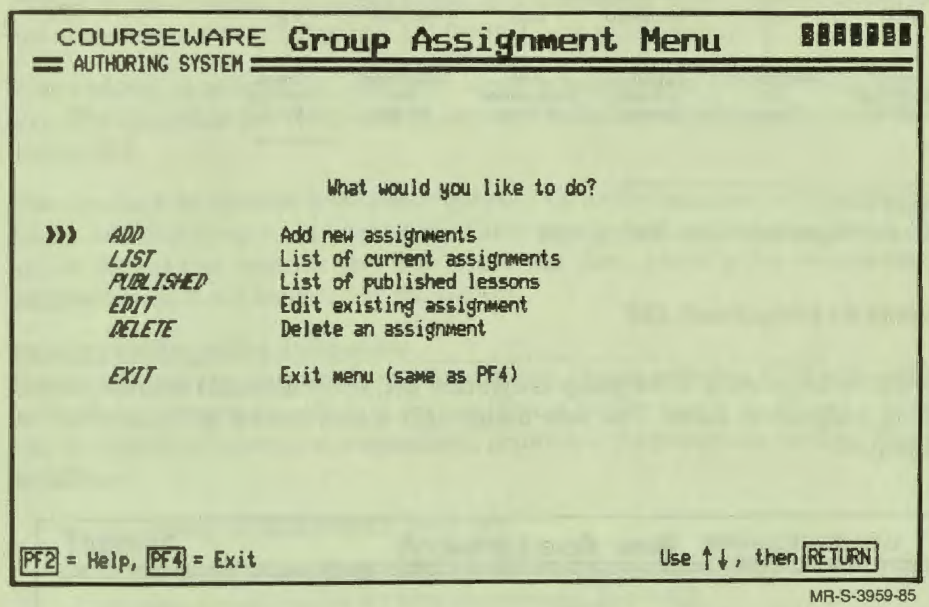
```
RPT11                                             FORMATTED PRINTOUT OF ENG1                                          PAGE:   1
DATE:  3/30/82                                        GROUP INFORMATION FILE                                       TIME: 07:48:51

ACTUAL NAME: DEANNA KNAUF             USER TYPE: A    C.A.S.  NAME: AUTHOR          PHONE: 272-3610
ADDRESS:      EM-1                                    VAX/VMS USERNAME:

ACTUAL NAME: JANE DOE                 USER TYPE: A    C.A.S.  NAME: CAT             PHONE:
ADDRESS:                                              VAX/VMS USERNAME: DOE

ACTUAL NAME: JIM BROWN                USER TYPE: A    C.A.S.  NAME: APPLE           PHONE:
ADDRESS:                                              VAX/VMS USERNAME: BROWN

ACTUAL NAME:                          USER TYPE: G    C.A.S.  NAME: INSTRUCTOR      PHONE:
ADDRESS:                                              VAX/VMS USERNAME:

ACTUAL NAME: INSTRUCTOR               USER TYPE: G    C.A.S.  NAME: INS             PHONE:
ADDRESS:                                              VAX/VMS USERNAME:

ACTUAL NAME: JAMES BROWN              USER TYPE: G    C.A.S.  NAME: BROWN           PHONE:
ADDRESS:                                              VAX/VMS USERNAME:

ACTUAL NAME: BARRY BROWN              USER TYPE: S    C.A.S.  NAME: BARRY           VAX/VMS USERNAME:
LAST DATE ON SYSTEM: 12-JAN-1992  LAST LESSON USED: MULTIPLY   LAST ASSIGNED LESSON USED: MULTIPLY   NUMBER OF SESSIONS:    1
AVERAGE SESSION LENGTH (HRS): 0.038069441  SESSION LENGTH STD. DEV.: 0.000000000  SUM OF SESSION LENGTH SQUARED: 0.001449282
SYSTEM TIME (HRS): 0.572316646  LESSON RUNNING FLAG: 0000

ACTUAL NAME: BILL BROWN               USER TYPE: S    C.A.S.  NAME: BILL            VAX/VMS USERNAME: BROWN
LAST DATE ON SYSTEM: sired          LAST LESSON USED: rt sequen  LAST ASSIGNED LESSON USED: cos: 1. S  NUMBER OF SESSIONS:
AVERAGE SESSION LENGTH (HRS): 0.000000000  SESSION LENGTH STD. DEV.: 0.000000000  SUM OF SESSION LENGTH SQUARED: 0.000000000
SYSTEM TIME (HRS): 0.049694441  LESSON RUNNING FLAG: 0000

ACTUAL NAME: DAVE GREEN               USER TYPE: S    C.A.S.  NAME: DAVE            VAX/VMS USERNAME:
LAST DATE ON SYSTEM:  8-JAN-1982  LAST LESSON USED: NEWVARS    LAST ASSIGNED LESSON USED: NEWVARS    NUMBER OF SESSIONS:    2
AVERAGE SESSION LENGTH (HRS): 0.081837497  SESSION LENGTH STD. DEV.: 0.001584858  SUM OF SESSION LENGTH SQUARED: 0.013397263
SYSTEM TIME (HRS): 0.014480590  LESSON RUNNING FLAG: 0000

ACTUAL NAME: DEANNA KNAUF             USER TYPE: S    C.A.S.  NAME: DEANNA          VAX/VMS USERNAME:
LAST DATE ON SYSTEM: 18-MAR-1982  LAST LESSON USED: INTRO        LAST ASSIGNED LESSON USED: INTRO      NUMBER OF SESSIONS:   12
AVERAGE SESSION LENGTH (HRS): 0.141779392  SESSION LENGTH STD. DEV.: 0.202098179  SUM OF SESSION LENGTH SQUARED: 0.690480278
SYSTEM TIME (HRS): 2.301533944  LESSON RUNNING FLAG: 0000

ACTUAL NAME: ELIZABETH                USER TYPE: S    C.A.S.  NAME: ELIZABETH       VAX/VMS USERNAME:
LAST DATE ON SYSTEM: 21-JAN-1982  LAST LESSON USED: STOP         LAST ASSIGNED LESSON USED: STOP       NUMBER OF SESSIONS:    5
AVERAGE SESSION LENGTH (HRS): 0.027684444  SESSION LENGTH STD. DEV.: 0.008195647  SUM OF SESSION LENGTH SQUARED: 0.003985686
SYSTEM TIME (HRS): 2.464550018  LESSON RUNNING FLAG: 0000

ACTUAL NAME: JANET                    USER TYPE: S    C.A.S.  NAME: JANET           VAX/VMS USERNAME:
LAST DATE ON SYSTEM: 21-JAN-1982  LAST LESSON USED: STOP         LAST ASSIGNED LESSON USED: STOP       NUMBER OF SESSIONS:    1
AVERAGE SESSION LENGTH (HRS): 0.052613884  SESSION LENGTH STD. DEV.: 0.000000000  SUM OF SESSION LENGTH SQUARED: 0.002768220
SYSTEM TIME (HRS): 0.519058287  LESSON RUNNING FLAG: 0000
```

MR-S-2065-82

Figure 2–20
(11) Dump Of The .INF File

## ASSIGNING LESSONS TO THE GROUP: ASSIGNMENT

To assign lessons to the group, select the ASSIGNMENT option from the group instructor menu. The group assignment menu shown in Figure 2–21 is displayed.



```
COURSEWARE        Group Assignment Menu
AUTHORING SYSTEM

                    What would you like to do?

>>>  ADD            Add new assignments
     LIST           List of current assignments
     PUBLISHED      List of published lessons
     EDIT           Edit existing assignment
     DELETE         Delete an assignment

     EXIT           Exit menu (same as PF4)

PF2 = Help, PF4 = Exit                          Use ↑↓, then RETURN
```

MR-S-3959-85

Figure 2–21
Group Assignment Menu

Options from the group assignment menu allow you to add, list, edit, or delete assignments, and list the published lessons that are available for assignment. Figure 2–22 shows the screens that you access by selecting options from the group assignment menu.

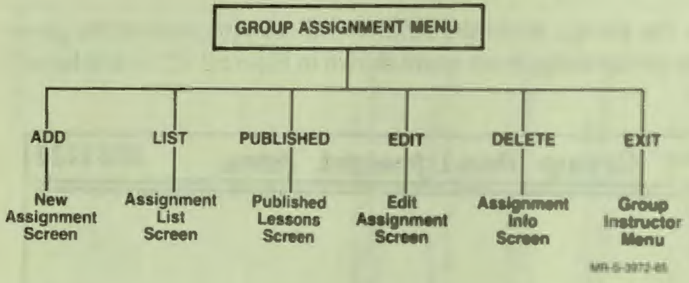
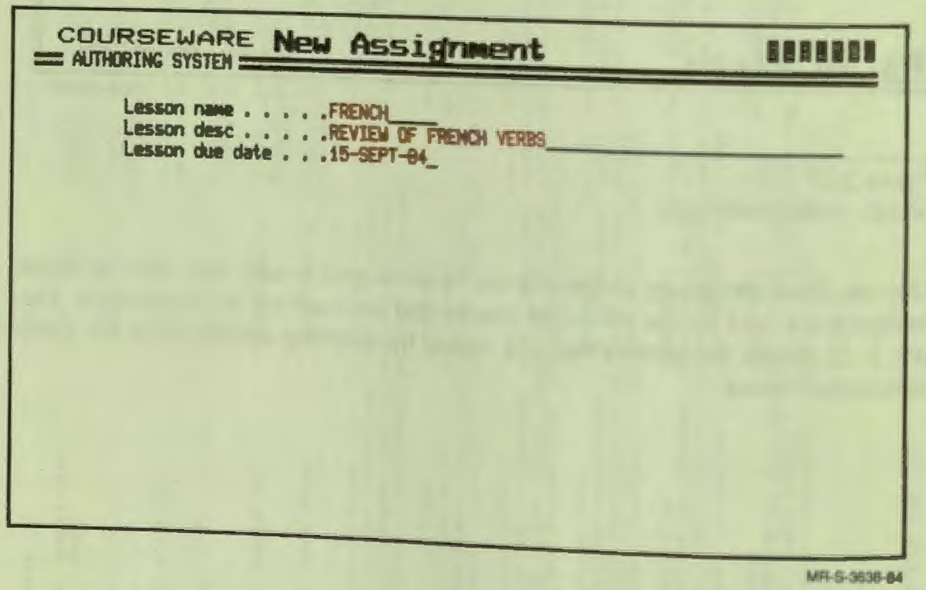


Figure 2–22
Group Assignment Menu And Options

### Adding An Assignment: ADD

To add an assignment to the group assignment list, select the ADD option from the group assignment menu. The new assignment screen shown in Figure 2–23 is displayed.



```
Lesson name . . . . .FRENCH____
Lesson desc . . . . .REVIEW OF FRENCH VERBS____________________
Lesson due date . . .15-SEPT-84_
```

Figure 2–23
New Assignment Screen

### Selecting Structured Assignments

When you add the first assignment, you are asked if you want assignments to be structured. If you want students to do assignments in the order that you specify, type Y to select structured. If you want students to be able to do assingnments in any order, type N to select unstructured. (If, after you select structured, you want to change to unstructured, you must delete all assignments from the assignment list and select unstructured when you add the first new assignment.)

When adding an assignment, you must supply a lesson name of up to nine characters. The name that you enter must be identical to the lesson name in the published lessons list.

You can enter an optional lesson description of up to 60 characters and an optional lesson due date of up to 11 characters. These appear both in your assignment list and in the list that students see. The lesson due date, which is for documentary purposes only, is not enforced by the system.

### Inserting and Appending Assignments

If you selected the structured option, you can choose whether to insert a new assignment between two others in the assignment list, or simply to append it to the list. To append or insert new assignments, respond to the prompt for the next lesson as follows:

- To append an assignment, press RET.
- To insert an assignment, type the name of the assignment currently on the list that should follow the new assignment. Press RET.

### Changing An Assignment

When you finish entering the lesson information, you receive a prompt for verification:

```
Is the above information correct? (Y or N)
```

If you type N, change options are added to the screen to let you correct the information. If you type Y, the lesson is assigned to the group and you return to the group assignment menu unless one of the following error messages appears:

-
  ```
  The lesson has not been published
  Re-edit the assignment? (Y or N)
  ```

  This message indicates that there is no published lesson with the name you entered, or that you tried to assign a published lesson that is restricted to another group.

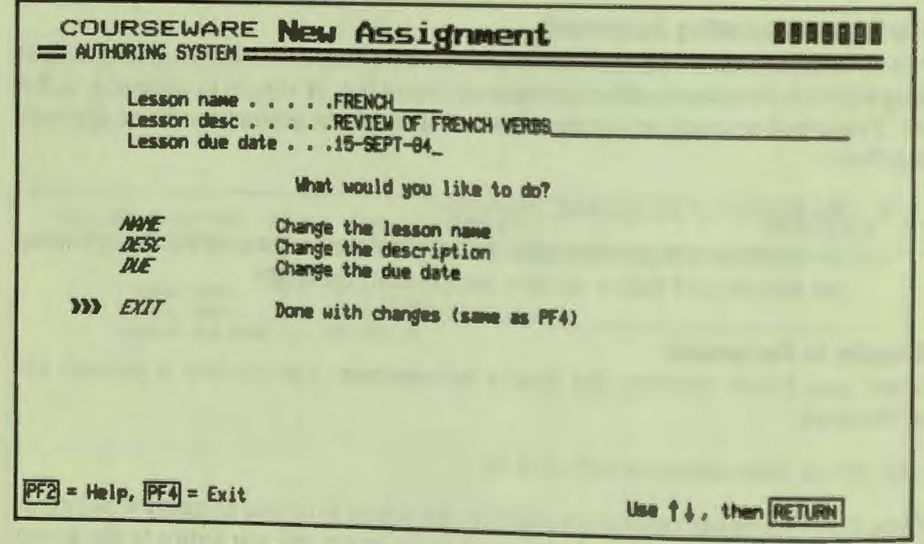
- This lesson is already assigned to this group
  Re-edit the assignment? (Y or N)

  This message appears if the lesson is already assigned to the group.

- The lesson name is invalid
  Re-edit the assignment? (Y or N)

  This message indicates that the lesson name contains spaces or nonalphanumeric characters.

If you type Y in response to any of the above messages, change options are added to the screen. If you type N, the lesson is not added and you return to the group assignment menu. The new assignment screen with change options is shown in Figure 2–24.



```
COURSEWARE New Assignment
AUTHORING SYSTEM

      Lesson name . . . . .FRENCH____
      Lesson desc . . . . .REVIEW OF FRENCH VERBS______________________
      Lesson due date . . .15-SEPT-84_

                     What would you like to do?

      NAME          Change the lesson name
      DESC          Change the description
      DUE           Change the due date

  >>> EXIT          Done with changes (same as PF4)


PF2 = Help, PF4 = Exit
                                          Use ↑↓, then RETURN
```

MR-S-3639-84

Figure 2–24
New Assignment Screen With Change Options

Each of the options for changing the assignment information is described below.

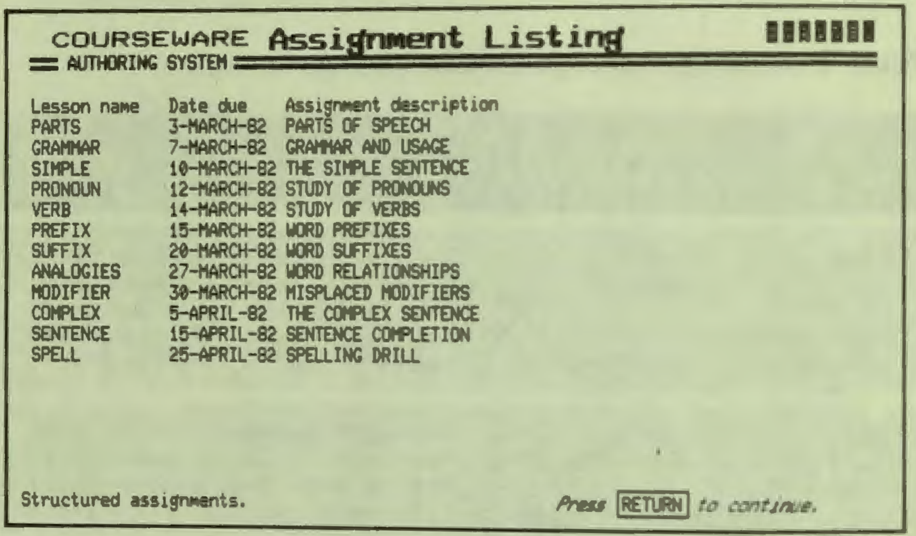
**Table 2–4: Assignment Information Change Options**

| C.A.S OPTION | CHANGE ITEM | C.A.S. EDIT PROCESS (What the system does, and what you should type/enter) |
|---|---|---|
| NAME | Lesson Name | The current name is erased. Enter correct lesson name. The lesson name that you enter must be identical to the lesson name appearing in the published lessons list. |
| DESC | Lesson Description | The current description is erased. Enter new description. The lesson description that you enter here appears in the assignment list. |
| DUE | Due Date | The current due date is erased. Enter new due date. The due date that you enter here appears in the assignment list. |
| NEXT | Next Lesson | The current next lesson is erased. Enter new next lesson. *This option appears only when assignments are structured. |

When you finish making changes, select the EXIT option or press PF4.

## Listing Group Assignments: LIST

To list all the current assignments for the group, select the LIST option from the group assignment menu. A typical assignment list containing lesson names, due dates, and assignment descriptions is shown in Figure 2–25. Press RET or PF4 when you finish viewing the list to return to the group assignment menu.



COURSEWARE **Assignment Listing**
AUTHORING SYSTEM

```
Lesson name   Date due      Assignment description
PARTS         3-MARCH-82    PARTS OF SPEECH
GRAMMAR       7-MARCH-82    GRAMMAR AND USAGE
SIMPLE        10-MARCH-82   THE SIMPLE SENTENCE
PRONOUN       12-MARCH-82   STUDY OF PRONOUNS
VERB          14-MARCH-82   STUDY OF VERBS
PREFIX        15-MARCH-82   WORD PREFIXES
SUFFIX        20-MARCH-82   WORD SUFFIXES
ANALOGIES     27-MARCH-82   WORD RELATIONSHIPS
MODIFIER      30-MARCH-82   MISPLACED MODIFIERS
COMPLEX       5-APRIL-82    THE COMPLEX SENTENCE
SENTENCE      15-APRIL-82   SENTENCE COMPLETION
SPELL         25-APRIL-82   SPELLING DRILL
```

Structured assignments.                    *Press* RETURN *to continue.*

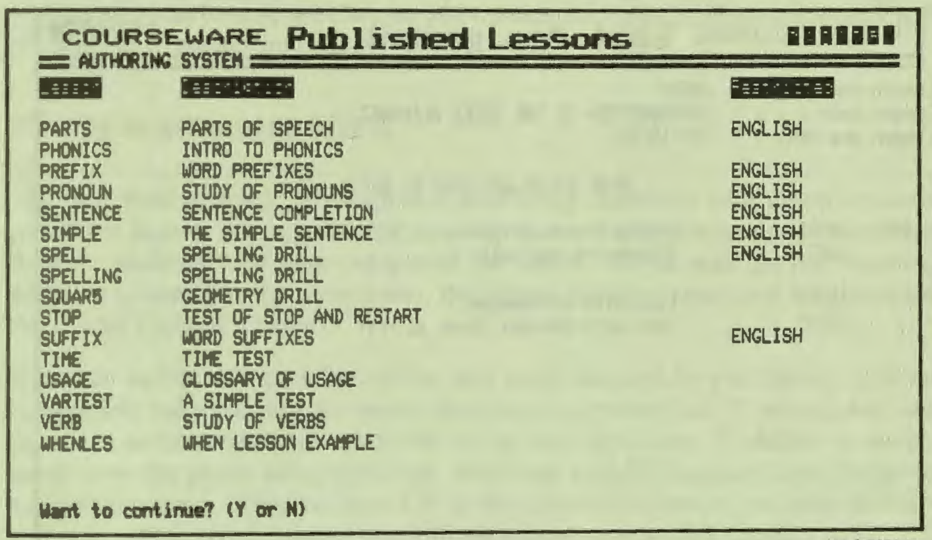
MR-S-3657-84

Figure 2–25
Assignment List Screen

## Listing Published Lessons: PUBLISHED

The published lessons list contains the names of lessons that are available for assignment.

To see this list, select the PUBLISHED option from the group assignment menu. A typical published lessons screen containing lesson names, descriptions, and group restrictions is shown in Figure 2–26.



```
COURSEWARE Published Lessons                                  [illegible]
AUTHORING SYSTEM
[illegible]     [illegible]                                  [illegible]

PARTS           PARTS OF SPEECH                               ENGLISH
PHONICS         INTRO TO PHONICS
PREFIX          WORD PREFIXES                                 ENGLISH
PRONOUN         STUDY OF PRONOUNS                             ENGLISH
SENTENCE        SENTENCE COMPLETION                           ENGLISH
SIMPLE          THE SIMPLE SENTENCE                           ENGLISH
SPELL           SPELLING DRILL                                ENGLISH
SPELLING        SPELLING DRILL
SQUAR5          GEOMETRY DRILL
STOP            TEST OF STOP AND RESTART
SUFFIX          WORD SUFFIXES                                 ENGLISH
TIME            TIME TEST
USAGE           GLOSSARY OF USAGE
VARTEST         A SIMPLE TEST
VERB            STUDY OF VERBS
WHENLES         WHEN LESSON EXAMPLE

Want to continue? (Y or N)
```

MR-S-3656-84

Figure 2–26
Published Lessons Screen

When a list is too long to be displayed on one screen, the following prompt appears at the bottom of the first section:

Want to continue? (Y or N)

If you want to view the next section of the published lessons list, type Y. Type N if you want to return to the group assignment menu. If the list does not continue onto another screen, press RET or PF4 when you finish viewing the screen to return to the group assignment menu.

## Editing An Assignment: EDIT

You can change any of the information about an assignment except the assignment name, which corresponds to the name of a published lesson. To change an assignment's description, due date, or position in a structured list of assignments, select the EDIT option from the group assignment menu. A prompt for the lesson name is added to the screen. After entering the name of the lesson you want to edit, you see the edit assignment screen. A typical edit assignment screen is shown in Figure 2–27.

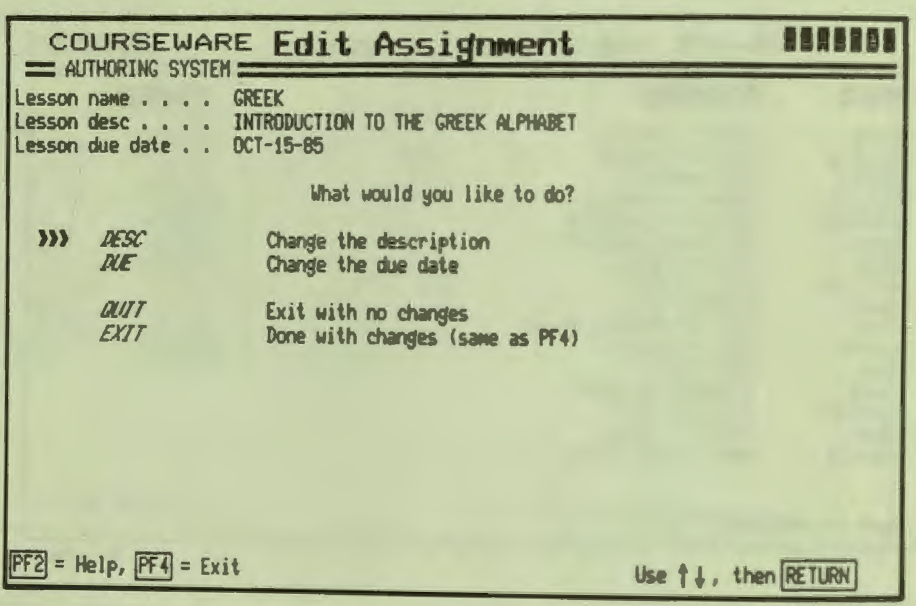

```
COURSEWARE Edit Assignment
= AUTHORING SYSTEM =
Lesson name . . . .  GREEK
Lesson desc . . . .  INTRODUCTION TO THE GREEK ALPHABET
Lesson due date . .  OCT-15-85

                         What would you like to do?

 >>>  DESC          Change the description
      DUE           Change the due date

      QUIT          Exit with no changes
      EXIT          Done with changes (same as PF4)




PF2 = Help, PF4 = Exit                          Use ↑↓, then RETURN
```
MR-S-3961-85

Figure 2–27
Edit Assignment Screen

Each of the options for editing the assignment information is described below.

## Table 2–5: Assignment Information Edit Options

| C.A.S. OPTION | EDIT ITEM | C.A.S. RESPONSE (What you should type/enter) |
|---|---|---|
| DESC | Lesson Description | The current description is erased. Enter new description. The lesson description that you enter here appears in the assignment list. |
| DUE | Due Date | The current due date is erased. Enter new due date. The due date that you enter here appears in the assignment list. |
| NEXT ○ | Next Lesson | The current next lesson is erased. Enter new next lesson. This option appears only when assignments are structured. |

When you finish editing the assignment, select the EXIT option or press PF4. (If you do not want to save the changes you made, select the QUIT option to exit with no changes to the assignment.)

## Deleting An Assignment: DELETE

You may want to delete an assignment after every student in your group has completed the lesson. When you delete an assignment, the system retains performance data for students who have completed the lesson. If you reassign the lesson or assign a lesson with the same name, the lesson remains completed for those students who finished it earlier.

When an author deletes a lesson that you have assigned to your group, you are responsible for removing the lesson from the assignment list. If you do not, students see an error message when they try to take the lesson. To delete an assignment from the group assignment list, select the DELETE option from the group assignment menu. When prompted, type the name of the lesson you want to delete and press RET to get the assignment information screen for that lesson.

A typical delete assignment screen showing the lesson name, description, and due date is shown in Figure 2–28.

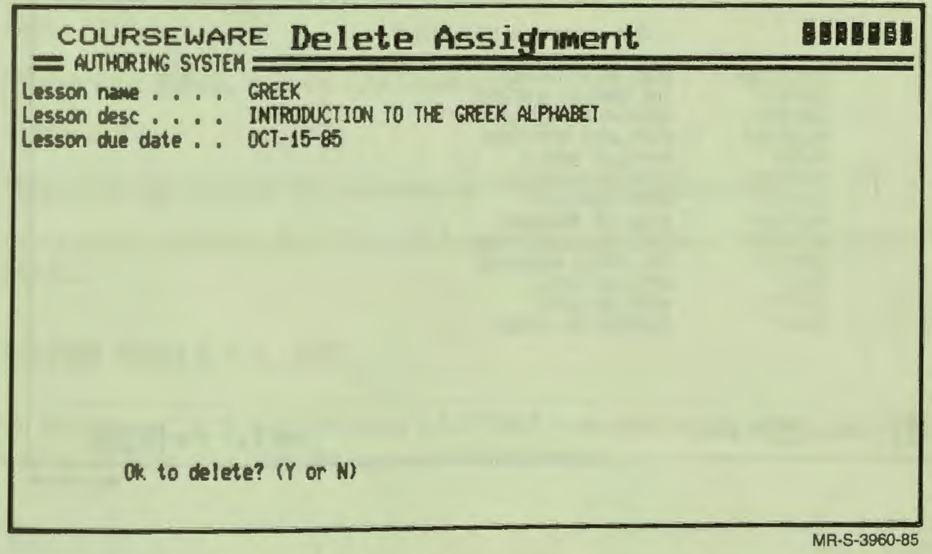

```
COURSEWARE  Delete Assignment
AUTHORING SYSTEM
Lesson name . . . .   GREEK
Lesson desc . . . .   INTRODUCTION TO THE GREEK ALPHABET
Lesson due date . .   OCT-15-85



          Ok to delete? (Y or N)
```
MR-S-3960-85

Figure 2–28
Delete Assignment Screen

A prompt for confirmation appears at the bottom of the screen:

Ok to delete? (Y or N)

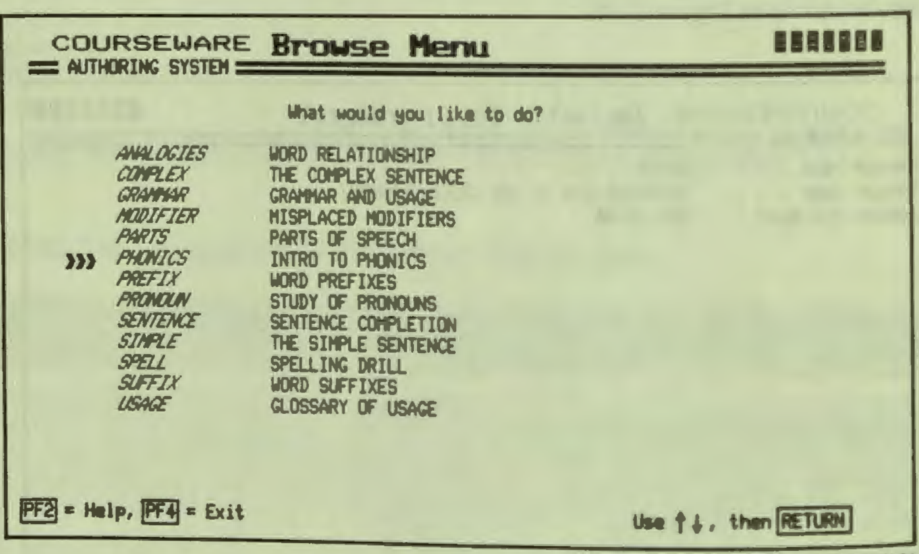
To delete the assignment from the group assignment list, type Y. To leave the assignment in the list, type N. After responding to the prompt, you return to the group assignment menu.

### Ending The Assignment Update: EXIT

To exit from the group assignment menu and return to the group instructor menu, select the EXIT option or press PF4.

## BROWSING THROUGH LESSONS: BROWSE

To browse through published lessons that are not restricted to other groups, select the BROWSE option from the group instructor menu. A typical browse menu is shown in Figure 2–29.



```
COURSEWARE  Browse Menu
AUTHORING SYSTEM

                         What would you like to do?

          ANALOGIES      WORD RELATIONSHIP
          COMPLEX        THE COMPLEX SENTENCE
          GRAMMAR        GRAMMAR AND USAGE
          MODIFIER       MISPLACED MODIFIERS
          PARTS          PARTS OF SPEECH
     >>>  PHONICS        INTRO TO PHONICS
          PREFIX         WORD PREFIXES
          PRONOUN        STUDY OF PRONOUNS
          SENTENCE       SENTENCE COMPLETION
          SIMPLE         THE SIMPLE SENTENCE
          SPELL          SPELLING DRILL
          SUFFIX         WORD SUFFIXES
          USAGE          GLOSSARY OF USAGE

PF2 = Help, PF4 = Exit                              Use ↑↓, then RETURN
```

MR-S-3656-84

Figure 2–29
Browse Menu

When students take an assigned lesson, performance statistics are collected. Depending on the lesson, students may be able to stop a lesson and later restart it where they left off. In contrast, in browse mode no performance statistics are collected, and lessons always start at the beginning.

When a lesson ends, the browse menu is redisplayed. Press PF4 to leave the browse menu and return to the group instructor menu.

## SENDING AND RECEIVING MAIL: MAIL

The VAX/VMS MAIL utility allows you to send and receive system mail. To run the MAIL utility, select the MAIL option from the group instructor menu. The program displays the prompt:

```
MAIL>
```

You can then issue the HELP command to obtain a list of commands and topics that are available with MAIL:

```
MAIL> HELP
```

To obtain more information about a particular command or topic, type HELP, followed by the command or topic name:

```
MAIL> HELP topic
```

To exit from MAIL, type the EXIT command:

```
MAIL> EXIT
```

When you exit from MAIL, you return to the group instructor menu.

Refer to the *VAX/VMS Mail Utility Reference Manual* for more information about MAIL.

## EXITING FROM C.A.S.: EXIT

To exit from C.A.S. and return to VAX/VMS command level, select the EXIT option (or press PF4) from the group instructor menu.

# 3 The Author

# 3 The Author

If you are an author, your function in C.A.S. is to publish lessons to make them available to C.A.S. users. You may also be engaged in developing lessons, which involves using a text editor to create source files of computer language instructions, compiling the source code to produce object code, and linking the object code to produce an executable image.

When you publish a lesson, the name of the lesson is added to the browse menus of all C.A.S. users, or to the browse menus of users in the group to which the lesson is restricted. (The lesson name is not added to the student assignment list, however, until the instructor assigns the lesson.)

When you publish a lesson, you must specify the executable image file (lesson.EXE) and any auxiliary files used by the lesson. C.A.S. copies these files to a lesson subdirectory from the directory you use when you publish the lesson. You also list the output files used by the lesson; you must create the output files in your default directory before you publish the lesson. There is a separate subdirectory for each lesson published in C.A.S.

In addition to publishing lessons, your functions as an author include:

- Republishing lessons after modifying them.
- Executing lessons to test them.
- Deleting lessons to make them unavailable to other users.
- Browsing through published lessons.
- Obtaining reports on available lessons and on student responses and performance for particular lessons.

As an author, you can publish only the lessons that are in your default directory. You can republish or delete only the lessons you yourself have published. The C.A.S. system manager can delete any lesson, and change lesson descriptions and group restrictions.

You may want to test a completed lesson for educational validity using a test group consisting of students from the target population. Once the test group is created and students are registered in it, you can publish the lesson and restrict it to the test group. After the students take the lesson, you can get reports on student responses and modify the lesson based on the report information. You can then recompile, relink, and republish the lesson without the restriction.

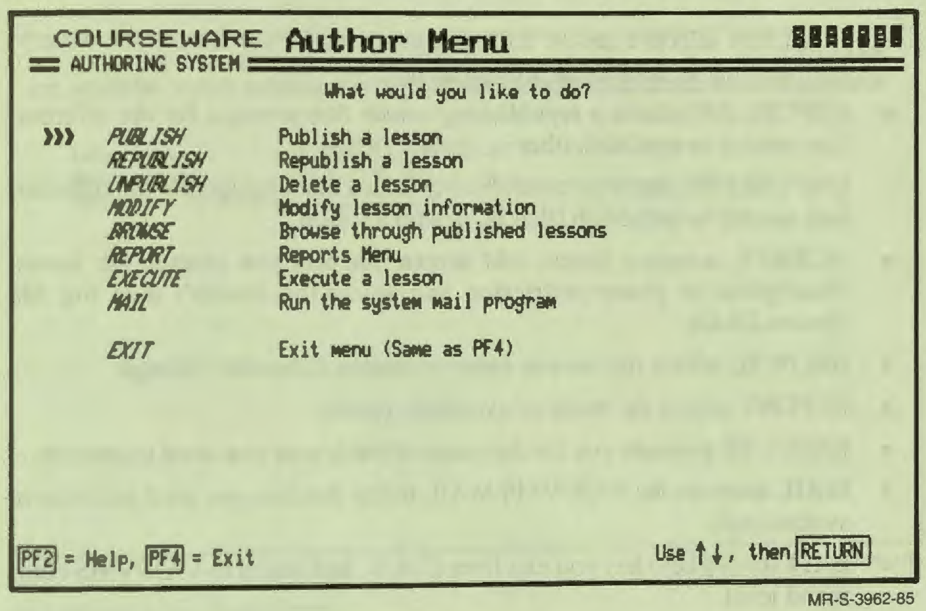
## WHAT YOU NEED TO KNOW

You need the following information in order to log in to your VAX/VMS account and run C.A.S.:

- Your VAX/VMS user name and password
- Your group name and C.A.S. name

## AUTHOR MENU

When you run C.A.S., you access menus that let you perform the activities described above. The first menu you view is the author menu shown in Figure 3–1.



```
COURSEWARE Author Menu
AUTHORING SYSTEM
                              What would you like to do?

>>>  PUBLISH          Publish a lesson
     REPUBLISH        Republish a lesson
     UNPUBLISH        Delete a lesson
     MODIFY           Modify lesson information
     BROWSE           Browse through published lessons
     REPORT           Reports Menu
     EXECUTE          Execute a lesson
     MAIL             Run the system mail program

     EXIT             Exit menu (Same as PF4)

PF2 = Help, PF4 = Exit                              Use ↑↓, then RETURN
```

MR-S-3962-85

Figure 3–1
Author Menu

By selecting options from the author menu, you access screens that enable you to publish, republish, execute, or delete lessons, or change the lesson information. You also access menus that let you select reports or browse through lessons. Figure 3–2 below shows where C.A.S. places you after you select options from the author menu.

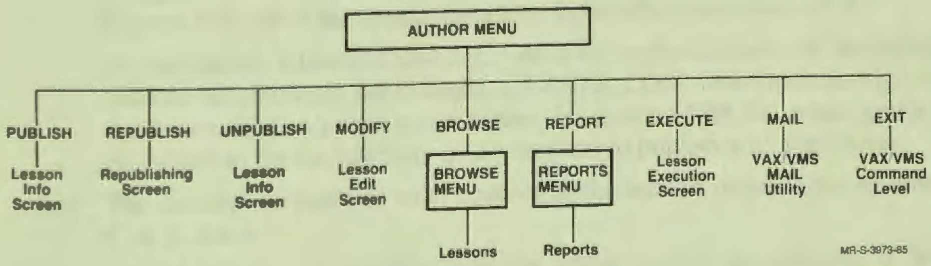


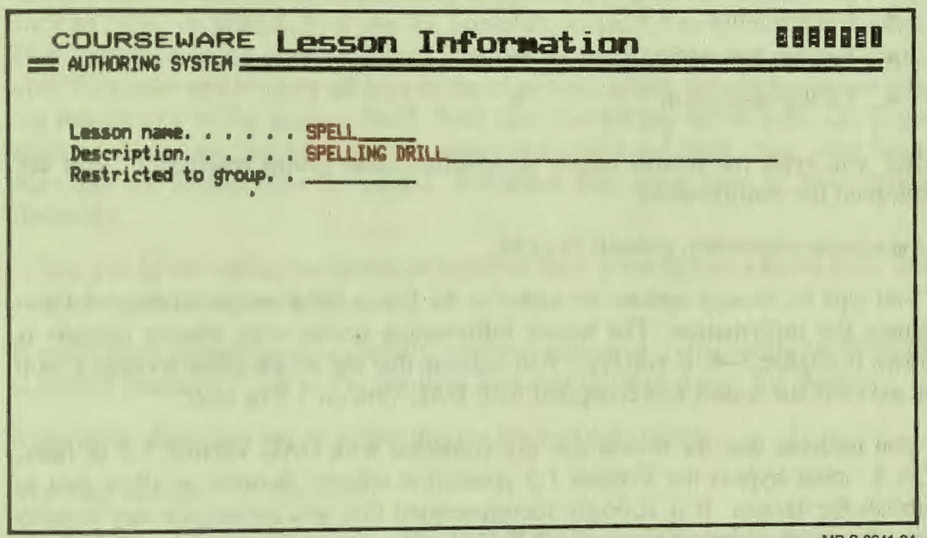
Figure 3–2
Author Menu And Options

The functions of the options in the author menu are described below:

- PUBLISH selects a lesson information screen for the lesson you want to publish (that is, add to the browse menu).
- REPUBLISH selects a republishing screen that prompts for the information needed to republish (that is, replace) a lesson.
- UNPUBLISH selects an unpublishing screen that prompts for the information needed to unpublish (that is, delete) a lesson.
- MODIFY selects a lesson edit screen that lets you change the lesson description or group restriction and delete the lesson's data log file (lesson.DLG).
- BROWSE selects the browse menu of lessons to browse through.
- REPORT selects the menu of available reports.
- EXECUTE prompts you for the name of the lesson you want to execute.
- MAIL accesses the VAX/VMS MAIL utility that lets you send and receive system mail.
- EXIT (or PF4 key) lets you exit from C.A.S. and return to VAX/VMS command level.

## PUBLISHING A LESSON: PUBLISH

To publish a lesson, select the PUBLISH option from the author menu. The lesson information screen shown in Figure 3–3 is displayed.



```
Lesson name. . . . . . SPELL
Description. . . . . . SPELLING DRILL
Restricted to group. .
```

MR-S-3641-84

Figure 3–3
Lesson Information Screen

Enter a lesson name, a lesson description, and the name of the group that the lesson is restricted to, if there is a restriction. As you type this information, you should be aware of the following:

- The lesson name must be the name of the lesson.EXE file containing the executable image, or the name of a lesson.COM file that you want executed whenever a student chooses the lesson. The lesson.COM and/or lesson.EXE file must be in your default directory. You can publish only the lessons that are in the default directory from which you run C.A.S.

  If you specify a lesson.COM file, you must begin the name of the lesson with the @ character, for example, @MATH.COM. You must also specify the lesson.EXE file that accompanies the lesson.COM file when you are prompted to list the auxiliary files you want to publish with the lesson.

- The description that you enter appears in the browse menus seen by other C.A.S. users.

- Lessons can be restricted to only one group, or left unrestricted. If you want the lesson to be available to all groups, simply press RET when prompted to specify which group the lesson should be restricted to.

The maximum number of characters allowed is:

|  |  |
|---|---|
| • Lesson name | – 9 |
| • Lesson description | –60 |
| • Group restriction | – 9 |

After you type the lesson name, description, and group restriction, you are prompted for confirmation:

```
Is the above information correct? (Y or N)
```

If you type N, change options are added to the lesson information screen to let you change the information. The lesson information screen with change options is shown in Figure 3–4. If you type Y to indicate that the information is correct, you are asked if the lesson was compiled with DAL Version 1.5 or later.

If you indicate that the lesson was not compiled with DAL Version 1.5 or later, C.A.S. must bypass the Version 1.5 protection scheme in order to allow you to publish the lesson. It is strongly recommended that you recompile any lessons compiled with older versions of VAX DAL. If you publish an older VAX DAL lesson without recompiling it, you cannot take advantage of the file protection that Version 1.5 provides, and you therefore risk a breach of security.

The next prompt to appear asks you to supply the names of any auxiliary files that the lesson requires:

```
Type in the files (one per line) that this lesson requires.
>________________________
```

Auxiliary files that commonly accompany the lesson.EXE are the abstract file (.ABS), text files (.TXT), picture files (.PIC), tray files (.SHO), and alternate character set files (.FNT). If you specify a lesson.COM file that begins with the character @ as your lesson name, you must enter the lesson.EXE file as an auxiliary file when publishing the lesson. Although no other auxiliary files are required for publishing a lesson, the published lesson will not run correctly if you omit an auxiliary file needed by the lesson.

Each lesson.EXE should be accompanied by an abstract file (lesson.ABS) that contains a description of the lesson. The lesson.ABS file, which is an ASCII text file, provides the lesson description that is displayed when you press the DETAIL key in browse mode. The lesson.ABS file should have the same name as the lesson. You can use any editor to create it. Each line in the file can be 64 characters wide, and you can display up to 15 lines on one screen. If the description in the file exceeds one screen on the monitor, it continues on the subsequent screen.

When prompted for the auxiliary files, type one file name per line and press RET, or use wildcard characters to enter multiple files of the same file name or file type, such as *.PIC or SPELL.*. If you try, however, to use *.* to specify the auxiliary files, you receive an error message. This occurs because the process that copies your files attempts to copy all files in the directory, which now includes the error log file created by the process itself. Note also that for any file of type .OUT, you must explicitly specify the file name rather than use a wildcard. You must list all files that the lesson uses for output, and these files must exist in your default directory.

When you finish typing the names of required files, press RET on a blank line. This starts the publishing process.

If publishing completes successfully, you are prompted to list, without using the wildcard character *, all of the auxiliary files that the lesson uses for output:

Type in the files (one per line) that this lesson uses for output.

You may not use "*" character.

The system assigns write access to the files that you list. The lesson then can write information to the output files when a user executes it.

When you finish entering the lesson's output files, you return to the author menu. The browse list now contains the name and description of the lesson you published.

## ERROR MESSAGES

After you type the lesson name, description, and group restriction, and type Y in response to the confirmation message above, C.A.S. normally publishes the lesson. You may, however, receive one of the messages described below:

```
Lesson not found
Edit the information? (Y or N)
```

This message indicates that the lesson you specified does not exist in your default directory.

```
Lesson has already been published
Edit the information? (Y or N)
```

This message is displayed if a lesson with the same name is already published. If there is a published lesson with the name that you typed, it indicates one of two things:

- Another author has published a lesson with the same name. If this happens, you should exit from C.A.S., rename your lesson, and try again.
- You have already published an earlier version of your lesson. If this is the case, republish the lesson using the REPUBLISH option described in the next section.

```
Invalid lesson name
Edit the information? (Y or N)
```

This message appears if the lesson name contains spaces or nonalphanumeric characters.

(If you type N in response to any of the above messages, you return to the author menu. If you type Y, change options are added to the lesson information screen to let you change the lesson information. The lesson information screen with the change options is shown in Figure 3–4.)

```
An error has occurred while transferring the necessary files for this lesson. For a specific
error message, please refer to the output file CAS$PUBL.LOG.
```

This message appears when the system is unable to copy one or more of the auxiliary files you specified during publication. Check the file CAS$PUBL.LOG, which the system writes to your default directory each time you publish a lesson. The log file lists the files that were successfully copied to the lesson directory, and indicates what problems occurred when a file could not be copied. Common problems are misspelled file names, incorrect file protection, and insufficient disk quotas.

**To be able to read the log file CAS$PUBL.LOG, you must own the lesson files as well as the directory from which you publish the lesson.**

**If you have a number of large lessons, publish each one from a separate subdirectory so that you can easily identify the correct CAS$PUBL.LOG for an individual lesson.**

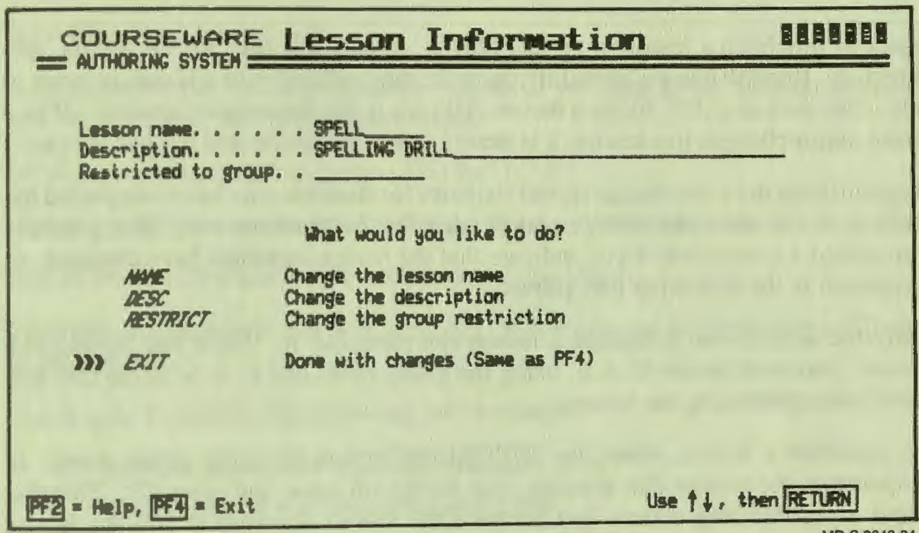

```
COURSEWARE Lesson Information
== AUTHORING SYSTEM

   Lesson name. . . . . . SPELL
   Description. . . . . . SPELLING DRILL
   Restricted to group. .

                         What would you like to do?

      NAME               Change the lesson name
      DESC               Change the description
      RESTRICT           Change the group restriction

   >>> EXIT              Done with changes (Same as PF4)

PF2 = Help, PF4 = Exit                        Use ↑↓, then RETURN
```
MR-S-3642-84

Figure 3–4
Lesson Information Screen With Change Options

The options for changing the lesson information are described below.

## Table 3–1: Lesson Information Change Options

| C.A.S. OPTION | CHANGE ITEM | C.A.S. RESPONSE (What you should type/enter) |
|---|---|---|
| NAME | Lesson Name | Current name is erased. Enter new lesson name. |
| DESC | Lesson Description | Current description is erased. Enter new description. |
| RESTRICT | Group Restriction | Current restriction is erased. Enter new restriction. |

When you finish making changes, select the EXIT option or press PF4. When prompted, type in the necessary file names to publish your lesson.

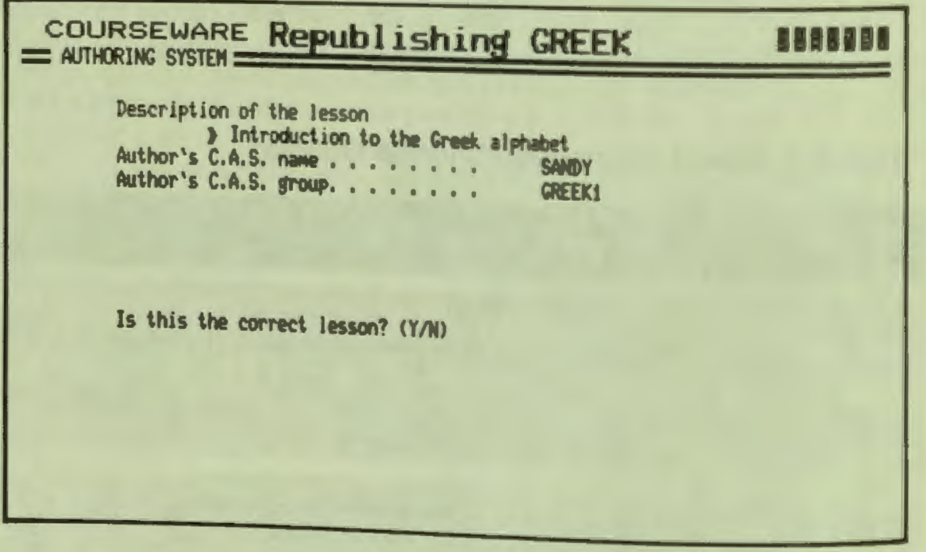
## REPUBLISHING A LESSON: REPUBLISH

You can republish a lesson to update files in, or add new files to, the lesson subdirectory. Republishing is useful if you make minor changes to a lesson or want to add a file such as a .PIC file or a lesson.ABS file to the lesson subdirectory. (If you make major changes to a lesson, it is best to delete the lesson and publish it again.)

Republishing does not change stored statistics for students who have completed the lesson. It also does not change restart variables for students who have partially completed a lesson unless you indicate that the restart variables have changed, as explained in the following paragraphs.

Only the author who published a lesson can republish it. When you republish a lesson, you must access C.A.S. using the group name and C.A.S. name that you used when publishing the lesson.

To republish a lesson, select the REPUBLISH option from the author menu. In response to the prompt that appears, type the lesson name and press RET. This displays a republishing screen that contains the lesson description and the lesson author's C.A.S. name and C.A.S. group. A lesson republishing screen is shown in Figure 3–5.



```
Republishing GREEK

        Description of the lesson
               > Introduction to the Greek alphabet
        Author's C.A.S. name . . . . . . . .      SANDY
        Author's C.A.S. group. . . . . . . .      GREEK1


        Is this the correct lesson? (Y/N)
```

MR-S-3963-85

Figure 3–5
Lesson Republishing Screen

You are prompted for confirmation of the lesson:

Is this the correct lesson? (Y/N)

If you type N, you return to the author menu. If you type Y, the message is erased and replaced with the following message:

Do you want to update the image (.EXE) file? (Y/N)

If you type N, you receive a prompt for the names of changed or new auxiliary files:

Type in the files (one per line) to update/add to the lesson.

Type one file name per line and press RET. When you are finished, press RET on a blank line.

If you type Y, you see the following two messages:

- Have the permanent variables changed? (Y/N)
- Have the restart variables changed? (Y/N)

These messages are displayed to allow you to delete the permanent variables file (.PRM) and the restart files (.RST) associated with the lesson. After you type Y or N in response to the permanent variables message, the restart variables message appears. Type Y in response to either message if you have changed the number or data type of the permanent variables or user-defined restart variables. This deletes the appropriate files and updates the lesson.EXE file. If you type N in response to either message, the permanent variables file and the restart files are not deleted.

After you respond to the permanent variables and restart variables messages, the following prompt appears:

Type in the files (one per line) to update/add to the lesson.

Type the names of any filesthat you have changed or added that go with the lesson. Type one file name per line and press RET. When you are finished, press RET on a blank line to republish the lesson. When the republishing process is finished, you return to the author menu.

## DELETING THE LESSON: UNPUBLISH

To delete a lesson so that it is no longer available to users, select the UNPUBLISH option from the author menu. In response to the prompt that appears, type the name of the lesson you want to delete. You are prompted for confirmation:

```
Ok to delete lesson? (Y or N)
```

If you type Y, the system does the following:

- Removes the lesson name from the browse menu. (The lesson name remains, however, on the assignment lists that instructors and students see. Since the lesson cannot be run after it is deleted, students trying to run the deleted lesson receive error messages. Therefore, instructors should remove the names of deleted lessons from assignment lists.)
- Deletes the lesson files and lesson subdirectory.
- Returns you to the author menu.

If you type N, the lesson is not deleted, and you remain in the unpublish lesson screen.

## MODIFYING A LESSON: MODIFY

If you want to change a group restriction or lesson description in the browse menus, or delete a lesson's data log file (lesson.DLG), select the MODIFY option from the author menu.

A lesson can be modified only by the C.A.S. system manager or the author who published the lesson. To modify a lesson, you must use the group name and C.A.S. name that you used when publishing the lesson.

When you select the MODIFY option, you are prompted for the lesson name. Type a lesson name of up to nine characters. A lesson edit screen, such as the one shown in Figure 3–6, displays the lesson description, the group restriction, the author's C.A.S. name, and the author's C.A.S. group. Change options shown at the bottom of the screen allow you to change the lesson description or group restriction, or to delete the lesson data log file.

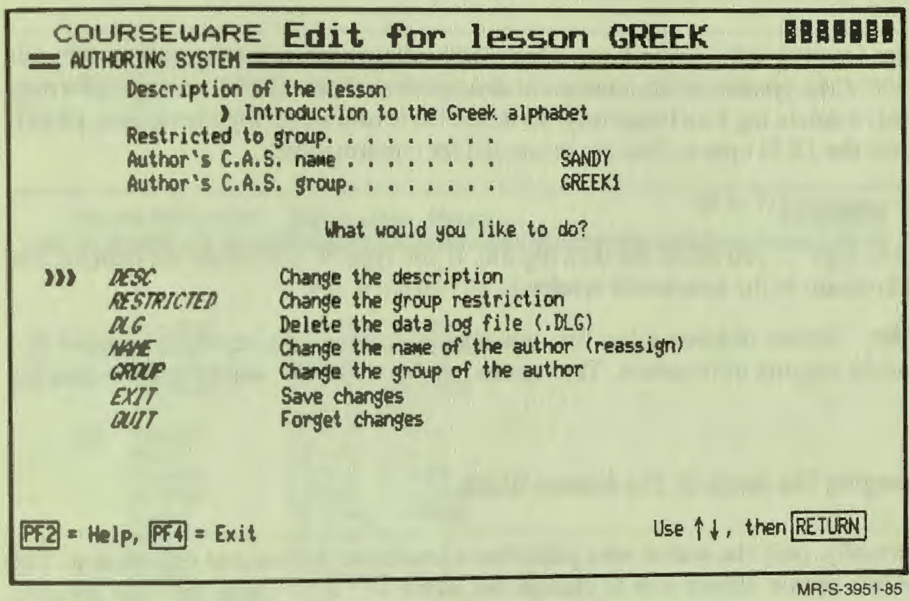

```
COURSEWARE Edit for lesson GREEK
AUTHORING SYSTEM
          Description of the lesson
                  ) Introduction to the Greek alphabet
          Restricted to group. . . . . . . . .
          Author's C.A.S. name . . . . . . . .      SANDY
          Author's C.A.S. group. . . . . . . .      GREEK1

                         What would you like to do?

  >>>   DESC             Change the description
        RESTRICTED       Change the group restriction
        DLG              Delete the data log file (.DLG)
        NAME             Change the name of the author (reassign)
        GROUP            Change the group of the author
        EXIT             Save changes
        QUIT             Forget changes


PF2 = Help, PF4 = Exit                           Use ↑↓, then RETURN
```
MR-S-3951-85

Figure 3–6
Lesson Edit Screen

## Changing The Lesson Description: DESC

To change the lesson description, select the DESC option. This erases the current description, leaving a blank line with the cursor positioned at the beginning of the line. Type a new description of up to 60 characters.

This lesson description appears in the browse menus. Assigned lessons have a different description that the instructor enters when assigning the lesson.

## Changing The Group Restriction: RESTRICT

To change the group restriction for a lesson, select the RESTRICT option. It erases the name of the group to which the lesson is restricted so that you can specify a new group restriction. Type a group name of up to nine characters.

To cancel the restriction rather than specify a new one, press RET on the blank line.

## Deleting The Data Log File: DLG

Data logging can create large files. Depending on the configuration of your VAX/VMS system and the amount of disk space available for file storage, you may need to delete log files frequently. To delete the lesson's data log file (lesson.DLG), select the DLG option. You are prompted for confirmation:

```
Are you sure? (Y or N)
```

If you type Y, you delete the data log file. If you type N, you retain the data log file and remain in the lesson edit screen.

**Note:** Before deleting a log file, you should generate any reports you need that contain logging information. The reports numbered 04, 06, and 09 use the data log file.

## Changing The Name Of The Author: NAME

Normally, only the author who publishes a lesson can update and republish it. The NAME option allows you to change the author's C.A.S. name for your lessons, effectively reassigning the lesson.

To assign a lesson to a different author, select the NAME option. This option erases the author's C.A.S. name. Type the C.A.S. name of the author to whom you are assigning the lesson.

The selection arrow moves to the GROUP option. If the new author of the lesson is in a separate group, select the GROUP option and see the instructions in the next section for changing the group. Otherwise, select the EXIT option to save your changes.

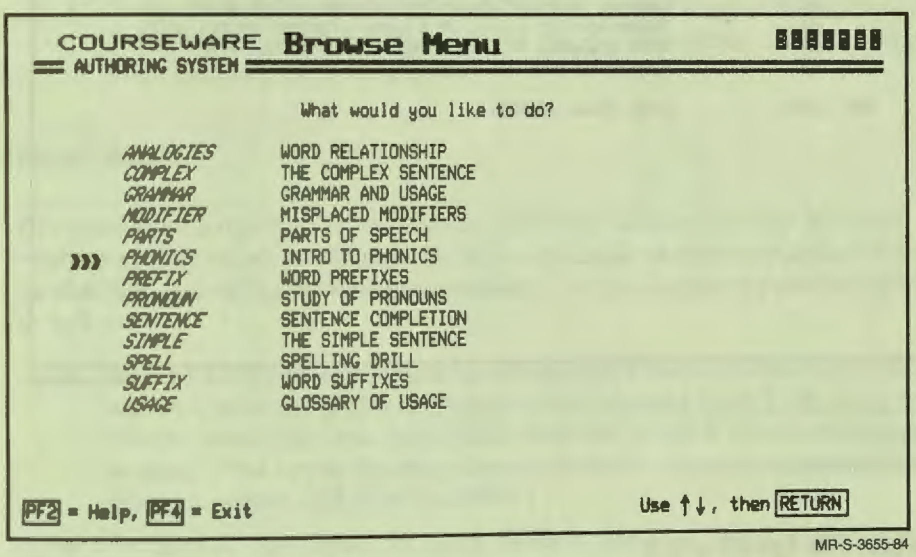
## Changing The Group Of The Author: GROUP

When you reassign a lesson by changing the author's name, you must also change the author's group if the new author is in a different C.A.S. group.

To change an author's group, select the GROUP option. It erases the author's C.A.S. group. Type the name of the C.A.S. group to which the new author of the lesson belongs.

## BROWSING THROUGH PUBLISHED LESSONS: BROWSE

To browse through published lessons that are not restricted to other groups, select the BROWSE option from the author menu. A typical browse menu is shown in Figure 3–7.



```
What would you like to do?

    ANALOGIES  WORD RELATIONSHIP
    COMPLEX    THE COMPLEX SENTENCE
    GRAMMAR    GRAMMAR AND USAGE
    MODIFIER   MISPLACED MODIFIERS
    PARTS      PARTS OF SPEECH
>>> PHONICS    INTRO TO PHONICS
    PREFIX     WORD PREFIXES
    PRONOUN    STUDY OF PRONOUNS
    SENTENCE   SENTENCE COMPLETION
    SIMPLE     THE SIMPLE SENTENCE
    SPELL      SPELLING DRILL
    SUFFIX     WORD SUFFIXES
    USAGE      GLOSSARY OF USAGE

PF2 = Help, PF4 = Exit                    Use ↑↓, then RETURN
```

Figure 3–7
Browse Menu

When students take an assigned lesson, performance statistics are collected. Depending on the lesson, students may be able to stop a lesson and restart it where they left off. In browse mode, no performance statistics are collected and lessons always start at the beginning.

When a lesson ends, the browse menu is redisplayed. Press PF4 to leave the browse menu and return to the author menu.

## SELECTING A REPORT: REPORT

To display or print a report, select the REPORT option from the author menu. This displays the reports menu shown in Figure 3–8. To exit from the reports menu and return to the author menu, select the EXIT option or press PF4.

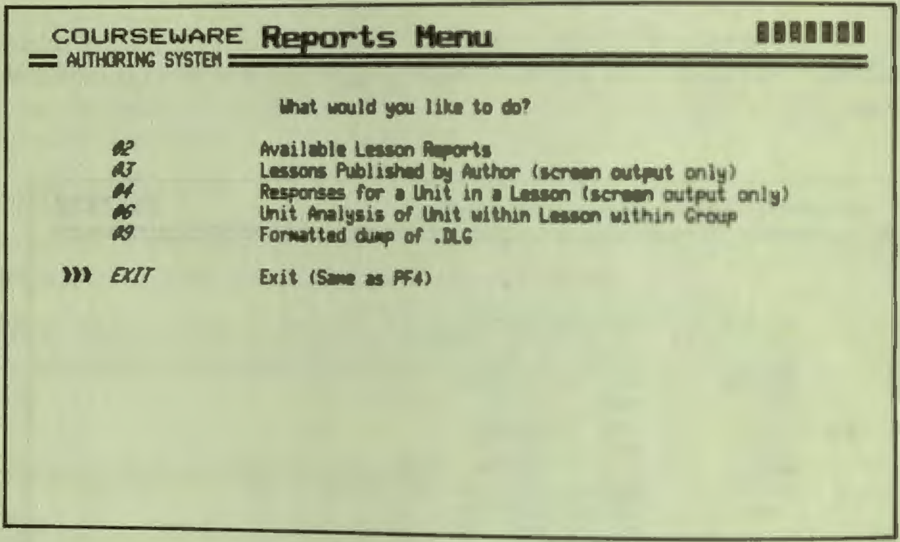

```
COURSEWARE  Reports Menu
AUTHORING SYSTEM

                    What would you like to do?

        A2          Available Lesson Reports
        A3          Lessons Published by Author (screen output only)
        A4          Responses for a Unit in a Lesson (screen output only)
        A6          Unit Analysis of Unit within Lesson within Group
        A9          Formatted dump of .DLC

    ))) EXIT        Exit (Same as PF4)
```

Figure 3–8
Reports Menu

## Report File Specification

When you select a report, you must supply the necessary report parameters in response to the prompts you receive. You must also specify where you want to store the report. For every report, you must answer the following file specification prompt:

WHERE SHOULD THE REPORT GO? (DEFAULT: *default*.LIS)

You can store the report either in the default file indicated in the prompt or in a file that you specify. Indicate where to store the report as follows:

- To store the report in the default file indicated in the file specification prompt, press RET. This places the default file in your default directory.
- To store the report in another file, type the file specification and press RET. If you do not specify a file extension, the system supplies the extension .LIS.

If you do not want to store the report in a disk file, you can display or print it from C.A.S. as follows:

- To display a report on your monitor, type the logical name TT: in response to the file specification prompt. TT: defines the terminal that C.A.S. is running on. Pressing CTRL/O terminates report output.
- To print a hard-copy report, type the logical name for any of the printers on the system. Although logical names are site dependent, the line printer often has the logical name LPA0:.

## Report Output

After you store a report in a specified file, a message informs you that the report is ready to display or print. The message shows the name of either the default file or another file, depending on what you specified. You can display or print the report, as follows:

- To print a copy of a report that is displayed only on the monitor (reports 03 and 04), press the SHIFT and PF1 keys simultaneously (on VT240 series terminals, press SHIFT and PRINT SCREEN) while the report is displayed on your monitor. (You can do this only if your terminal is connected to a hard-copy graphics printer such as an LA34VA.)
- To display the report on your monitor, issue a TYPE command at VAX/VMS command level after you exit from C.A.S.
- To print a hard copy of the report, issue a PRINT command at VAX/VMS command level after you exit from C.A.S.

## Report Descriptions

Reports show data only for the group from which you are running C.A.S. The five reports available to you are described below.

**Table 3–2: Report Descriptions**

| REPORT NAME | OPTION/REPORT NUMBER | DESCRIPTION |
|---|---|---|
| Available Lessons | 02 | Lists the name, description, and author of all published lessons that are available to the group. |
| Lessons Published By Author | 03 | Shows an alphabetical listing of all the lessons published by an author. |
| Responses For A Unit In A Lesson | 04 | Shows a tabular ranking of the five most popular actual responses for a lesson unit. Also displays a graph of the percent comparison of the response frequencies. |
| Unit Analysis Of Unit Within Lesson Within Group | 06 | Shows number of tries, total response time, and average response time of students, for a particular unit or all units, in a lesson. |
| Formatted Dump Of .DLG | 09 | Shows a formatted dump of the data log file for a particular lesson unit. |

## Report Parameters

The following pages contain examples that show the parameters each report requires. Each example displays exactly what appears on the monitor when you select the report. A report sample follows each example of the screen prompts.

The system displays a cursor, usually a rectangle, to prompt for information. When you are prompted for the name of a lesson or unit, type the name and press RET. When you choose the ALL or QUIT options, type ALL or QUIT in full. ALL selects all lessons or units; QUIT cancels the report and redisplays the reports menu. When prompted for a numbered option, type the number and press RET.

**(02) Available Lessons Report**

When you select this report, you produce the following display:

```
BEGINNING SELECTION OF AVAILABLE LESSONS FOR GROUP groupname
SORT LESSON INFORMATION BY:
 1. LESSON NAME
 2. AUTHOR
 3. EXIT FROM THE REPORT
PLEASE ENTER OPTION DESIRED
1 RET

WHERE SHOULD THE REPORT GO? (DEFAULT: CASRPT02.LIS)
RET
AVAILABLE LESSONS LIST IS READY TO BE PRINTED – CASRPT02.LIS

PRESS "RETURN" TO CONTINUE
```

This report is also available to instructors.

```
RPT02                          LESSONS AVAILABLE TO ENGLISH                    PAGE:   1
DATE:  9/23/85                                                            TIME: 16:09:37

LESSON NAME         LESSON DESCRIPTION                              AUTHOR NAME  &  GROUP
ANALOGIES   WORD RELATIONSHIPS                                        CAROL         ENGLISH
ANTONYMS    WORD OPPOSITES                                            CAROL         ENGLISH
APOS        APOSTROPHE USAGE                                          SCOTT         ENGLISH
COMPLEX     THE COMPLEX SENTENCE                                      CAROL         ENGLISH
DALCAI      INTRODUCTION TO DAL (VERSION 1.5)                         DIGITAL       SYSTEM

GRAMMAR     GRAMMAR AND USAGE                                         CAROL         ENGLISH
JUDGER      DEMONSTRATES JUDGING CRITERIA                             SCOTT         ENGLISH
LETTER      ALPHABET TEST                                             SCOTT         ENGLISH
MODIFIER    MISPLACED MODIFIERS                                       CAROL         ENGLISH
NEWWORDS    WORD MATCHING EXERCISE                                    DARRELL       ENGLISH

PARTS       PARTS OF SPEECH                                           CAROL         ENGLISH
PREFIX      WORD PREFIXES                                             CAROL         ENGLISH
SIMPLE      THE SIMPLE SENTENCE                                       CAROL         ENGLISH
SPELLING    SPELLING DRILL                                            LIZ           ENGLISH
SUFFIX      WORD SUFFIXES                                             CAROL         ENGLISH

VERB        STUDY OF VERB TENSES                                      LIZ           ENGLISH
                                                                               MR-S-3969-85
```

Figure 3–9
(02) Available Lessons Report

**(03) Lessons Published By Author**

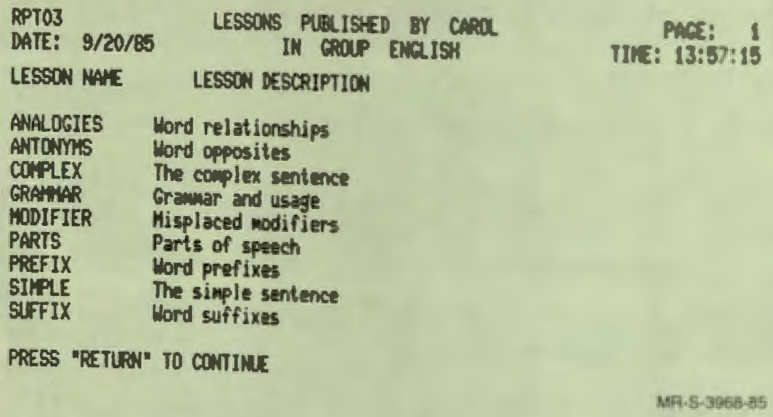
When you select this report, you display the following:

SELECTION BEGINNING FOR LESSONS PUBLISHED BY AUTHOR

ENTER AUTHOR'S C.A.S. NAME OR "QUIT"
CAROL $\boxed{\text{RET}}$

ENTER AUTHOR'S C.A.S. GROUP NAME OR "QUIT"
ENGLISH $\boxed{\text{RET}}$

This report is displayed on the monitor only.



```
RPT03                LESSONS  PUBLISHED  BY  CAROL                  PAGE:  1
DATE:  9/20/85             IN  GROUP  ENGLISH                 TIME: 13:57:15
LESSON NAME        LESSON DESCRIPTION

ANALOGIES       Word relationships
ANTONYMS        Word opposites
COMPLEX         The complex sentence
GRAMMAR         Grammar and usage
MODIFIER        Misplaced modifiers
PARTS           Parts of speech
PREFIX          Word prefixes
SIMPLE          The simple sentence
SUFFIX          Word suffixes

PRESS "RETURN" TO CONTINUE
```

MR-S-3968-85

Figure 3–10
(03) Lessons Published By Author

**(04) Responses For A Unit In A Lesson**

When you select this report, you see the following:

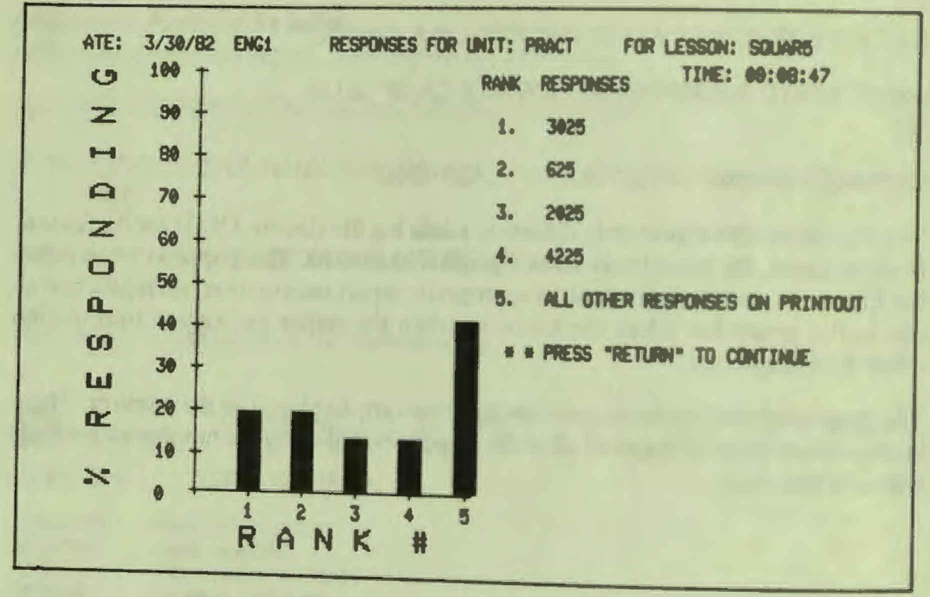
BEGINNING SELECTION FOR RESPONSE RANKING REPORT

ENTER LESSON NAME (9 CHARS) OR "QUIT"
SQUAR5 $\boxed{\text{RET}}$
YOU CAN RETRIEVE LOGGED DATA FOR ALL GROUPS OR A SPECIFIC GROUP.
ENTER NAME OF GROUP (9 CHARS) OR "ALL" OR "QUIT"
ENG1
ENTER UNIT NAME (9 CHARS)
PRACT $\boxed{\text{RET}}$

```
ALL OF THE RESPONSES WILL BE REPORTED ON A PRINTOUT

WHERE SHOULD THE REPORT GO? (DEFAULT: CASRPT04.LIS)
RET

RESPONSE RANKING REPORT READY – CASRPT04.LIS
```

You can create this report only if there is a data log file (lesson.DLG) for the lesson. In some cases, the report may show a graph with no data. This happens when a data log file exists, but does not contain appropriate report information, such as when no one in the group has taken the lesson or when the author has logged information other than responses.

The graph and the five most common responses are displayed on the monitor. There is also a hard-copy printout of all of the responses ranked by the number of students who entered them.



```
ATE:  3/30/82  ENG1      RESPONSES FOR UNIT: PRACT      FOR LESSON: SQUAR5
                                   RANK  RESPONSES           TIME: 00:08:47

                                    1.   3025

                                    2.   625

                                    3.   2025

                                    4.   4225

                                    5.   ALL OTHER RESPONSES ON PRINTOUT

                                     * * PRESS "RETURN" TO CONTINUE
```

```
DATE:   3/30/82   ENG1     RESPONSES FOR UNIT: PRACT      FOR LESSON: SQUAR5     T

RANK     CNT   RESPONSE

  1       3   3025

  2       3   625

  3       2   2025

  4       2   4225

  5       1   1225

  6       1   5625

  7       1   234

  8       1   4324

  9       1   225

 10       1   1225

 11       1   7225
```

MR-S-2081-82

Figure 3–11
(04) Responses For A Unit In A Lesson

**(06) Unit Analysis Of Unit Within Lesson Within Group**
When you select this report, you see the following:

```
ITEM ANALYSIS SELECTION BEGINNING

ENTER LESSON NAME (9 CHARS) OR "QUIT"
SQUAR5 (RET)
YOU CAN RETRIEVE LOGGED DATA FOR ALL GROUPS OR A SPECIFIC GROUP.
ENTER NAME OF GROUP (9 CHARS) OR "ALL" OR "QUIT" OR
PRESS <RETURN> TO USE DATA FROM YOUR CURRENT GROUP groupname
ENG1
ENTER UNIT NAME OR "ALL" (9 CHARS)
PRACT (RET)

SELECT SORT OPTION
 1. STUDENT NAME
 2. RESPONSE TIME
 3. AVG. RESPONSE TIME
1 (RET)

WHERE SHOULD THE REPORT GO? (DEFAULT: CASRPT06.LIS)
(RET)
ITEM ANALYSIS REPORT FOR LESSON: lesson name READY – CASRPT06.LIS

PRESS "RETURN" TO CONTINUE
```

You can create this report only if there is a data log file (lesson.DLG) for the lesson. If the lesson.DLG file for the selected lesson does not contain any records for students in the group you are reporting on, you get an error message and no report is generated. The error message, "ZERO RECORDS WERE SELECTED FOR THE REPORT – THE REPORT WAS NOT GENERATED", is displayed instead of the message telling you that the report is ready.

```
RPT06                  GROUP: ENG1      ITEM  ANALYSIS                    PAGE:    1
DATE:  3/30/82         LESSON: SQUARS     UNIT: PRACT                TIME: 07:51:52

                                                 TOTAL
                                       #     RESPONSE TIME         AVG RESPONSE
           STUDENT NAME              TRIES    PER STUDENT               TIME

        JULIE                          8         1:24                   0:11
        MIKE                          15         0:41                   0:03


                 * * * O V E R A L L   A V E R A G E * * *

              NUMBER OF           TOTAL               AVERAGE
                TRIES         RESPONSE TIME        RESPONSE TIME

                 11               01:02                00:05
```

Figure 3–12
(06) Unit Analysis Of Unit Within Lesson Within Group

## (09) Formatted Dump Of .DLG

When you select this report, you see the following:

DEFINE SELECTION CRITERIA FOR LOGGING REPORTS
ENTER LESSON NAME (9 CHARS) OR "QUIT"
PHONICS (RET)

ENTER UNIT NAME (9 CHARS) OR "ALL"
ALL (RET)

ENTER FROM DATE (MM/DD/YY) OR "ALL" FOR ALL DATES
03/25/85 (RET)

ENTER THRU DATE (MM/DD/YY) OR PRESS "RETURN" FOR CURRENT DATE
(RET)

WHERE SHOULD THE REPORT GO? (DEFAULT: CASRPT09.LIS)
(RET)

LOGGING REPORT IS READY FOR PRINTING – CASRPT09.LIS

PRESS "RETURN" TO CONTINUE

When a prompt appears for a beginning date (FROM DATE), type the date from which you want the report data to begin, or type ALL to get data for all dates. If you type a beginning date, you are prompted for an ending date (THRU DATE). If you type ALL instead of a beginning date, you are not prompted for an ending date.

Type beginning and ending dates in the MM/DD/YY format and include leading zeros (for example, 03/03/84).

You can create this report only if there is a data log file (lesson.DLG) that contains data for the specified unit.

```
RPT09          LOGGING FILE INFORMATION FOR LESSON: SQUARS          PAGE:    2
DATE: 03/30/82      FOR GROUP: ENG1        UNIT:                TIME: 10:17:56
                          (ALL DATES INCLUDED)
         UNEXPECTED: 7225

         RESPONSE:   3025
         LATENCY:       3
         UNEXPECTED: 3025

         RESPONSE:   7225
         LATENCY:       4
         UNEXPECTED: 7225




      UNIT: PRACT                DATE: 25-MAR-1982            TIME: 13:39:41.60
         RESPONSE:    625
         LATENCY:       3
         UNEXPECTED: 625

         RESPONSE:   2025
         LATENCY:       4
         UNEXPECTED: 2025

         RESPONSE:    625
         LATENCY:       3
         UNEXPECTED: 625

         RESPONSE:   2025
         LATENCY:       5
         UNEXPECTED: 2025

         RESPONSE:   2025
         LATENCY:       2
         UNEXPECTED: 2025
   STUDENT NAME: MIKE
      UNIT: PRACT                DATE: 25-MAR-1982            TIME: 13:41:27.11
         RESPONSE:    625
         LATENCY:       2
         UNEXPECTED: 625

         RESPONSE:    234
         LATENCY:       2
         UNEXPECTED: 234



      UNIT: PRACT                DATE: 25-MAR-1982            TIME: 13:41:37.02
         RESPONSE:   4324
         LATENCY:       2
         UNEXPECTED: 4324
```

MR-S-2083-82

Figure 3–13
(09) Formatted Dump Of .DLG

## EXECUTING A LESSON: EXECUTE

Executing a lesson is a way of testing it in C.A.S. without publishing it. When you execute a lesson, the lesson.EXE file must be in your default directory.

To execute a lesson, select the EXECUTE option from the author menu. When prompted for the name of the lesson you want to execute, type the lesson name and press RET. The lesson then executes as if you had selected it from the browse menu. When the lesson is finished executing, press RET to return to the author menu.

## SENDING AND RECEIVING MAIL: MAIL

The VAX/VMS MAIL utility allows you to send and receive system mail. To run the MAIL utility, select the MAIL option from the author menu. The program displays the prompt:

```
MAIL>
```

You can then issue the HELP command to obtain a list of commands and topics that are available with MAIL:

```
MAIL> HELP
```

To obtain more information about a particular command or topic, type HELP, followed by the command or topic name:

```
MAIL> HELP topic
```

To exit from MAIL, type the EXIT command:

```
MAIL> EXIT
```

When you exit from MAIL, you return to the author menu.

Refer to the *VAX/VMS Mail Utility Reference Manual* for more information about MAIL.

## EXITING FROM C.A.S.: EXIT

To exit from C.A.S. and return to VAX/VMS command level, select the EXIT option (or press PF4) from the author menu.

# 4

# The Student

# 4 The Student

The student's primary role is to take the lessons assigned by the instructor. Students can also browse through published lessons that are not restricted to other groups, and send and receive mail.

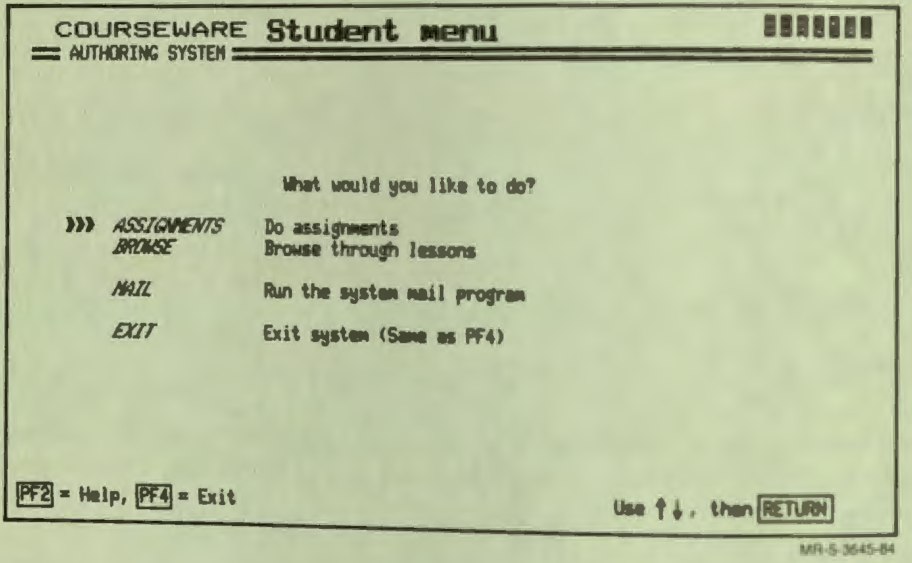
## WHAT STUDENTS NEED TO KNOW

The instructor provides students with the following information, which they need in order to log in to their VAX/VMS accounts and run C.A.S.:

- Instructions on how to log in to their VAX/VMS account, and possibly on how to run C.A.S. and log out. (In most cases, commands in the log-in command files of students place them automatically in C.A.S. and log them out of their VAX/VMS accounts. Refer to the *C.A.S. Delivery System System Manager's Guide* for information on log-in command files.)
- VAX/VMS user names and passwords.
- Group names and C.A.S. names, depending on their log-in command files.

## STUDENT MENU

When students run C.A.S. the student menu is displayed. Students can choose to do an assignment, browse, or send and receive mail. The student menu is shown in Figure 4–1.



```
COURSEWARE Student menu
AUTHORING SYSTEM

                         What would you like to do?

  ››› ASSIGNMENTS   Do assignments
      BROWSE        Browse through lessons

      MAIL          Run the system mail program

      EXIT          Exit system (Same as PF4)

PF2 = Help, PF4 = Exit                       Use ↑↓, then RETURN
```

Figure 4–1
Student Menu

The options from the student menu access a list of assignments, a menu of lessons to browse through, or the VAX/VMS MAIL utility. Figure 4–2 below shows where C.A.S. places students after they select options from the student menu.

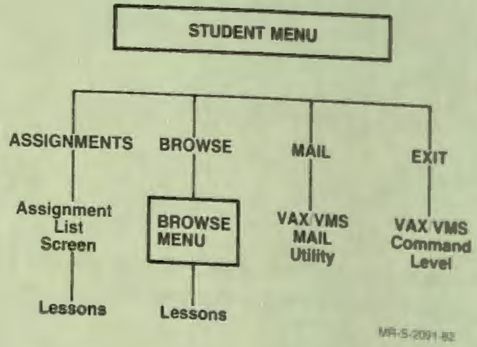


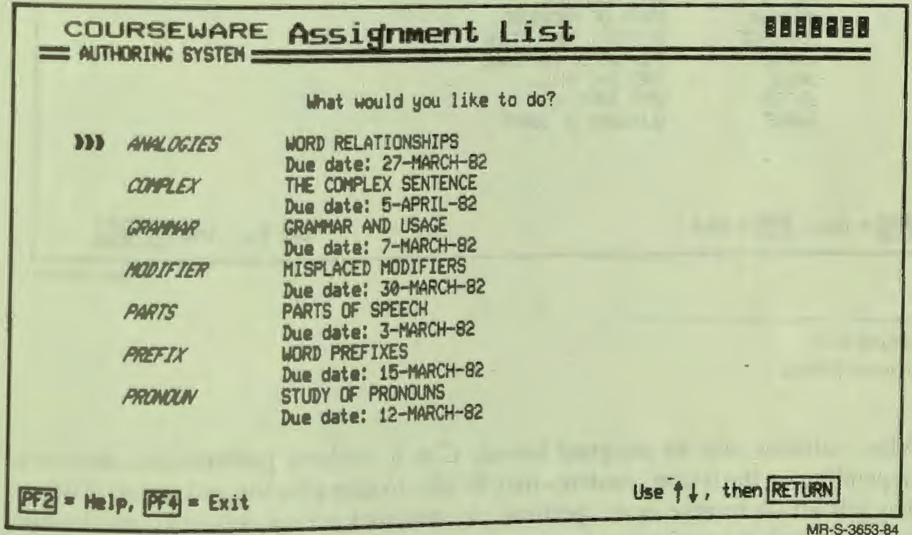
Figure 4–2
Student Menu And Options

To do assignments, students select the ASSIGNMENTS option from the student menu. Assignments are either structured or unstructured, depending on what the instructor specified when entering the first assignment for the group. This specification affects the way students access assignments. If assignments are structured, a student who selects the ASSIGNMENTS option is placed back into a partially completed lesson or into the next assignment. If assignments are unstructured, a student is shown a list of assignments to choose from.

Each assignment in the assignment list shows:

- The lesson name.
- A brief assignment description that the instructor entered when making the assignment.
- A status of done, partially completed (if the lesson contains restart logic), or not attempted.
- A due date for the assignment. (The due date is for documentary purposes only, and is not enforced by the system.)

As students do assignments, C.A.S. collects performance statistics. Figure 4–3 shows a sample assignment list. Students press PF4 to exit from the assignment list and return to the student menu.



```
COURSEWARE  Assignment List
AUTHORING SYSTEM

                    What would you like to do?

 ))) ANALOGIES      WORD RELATIONSHIPS
                    Due date: 27-MARCH-82
     COMPLEX        THE COMPLEX SENTENCE
                    Due date: 5-APRIL-82
     GRAMMAR        GRAMMAR AND USAGE
                    Due date: 7-MARCH-82
     MODIFIER       MISPLACED MODIFIERS
                    Due date: 30-MARCH-82
     PARTS          PARTS OF SPEECH
                    Due date: 3-MARCH-82
     PREFIX         WORD PREFIXES
                    Due date: 15-MARCH-82
     PRONOUN        STUDY OF PRONOUNS
                    Due date: 12-MARCH-82

PF2 = Help, PF4 = Exit                    Use ↑↓, then RETURN
```

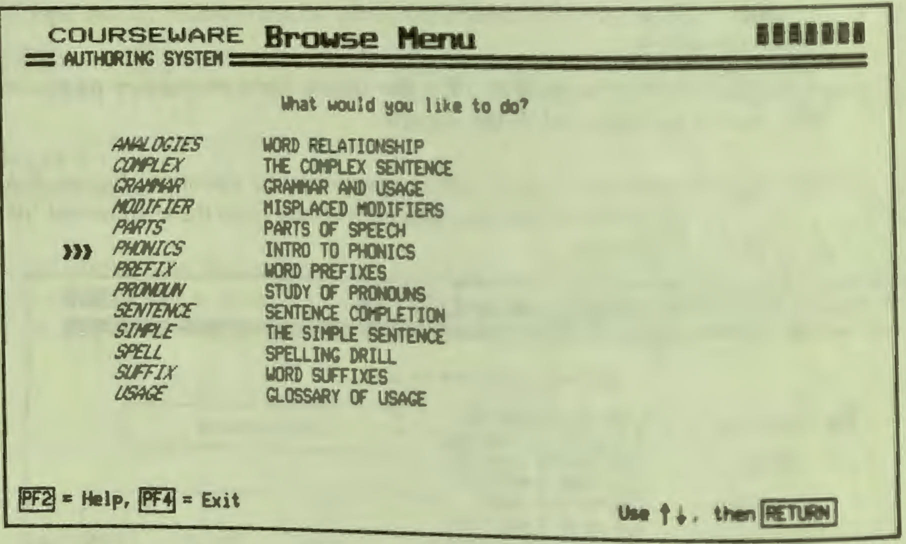
MR-S-3653-84

Figure 4–3
Assignment List Screen

Students cannot browse through assignments until they complete the assignments. Once C.A.S. collects performance information for unstructured assignments, students can browse through the assignments by selecting them from the assignment list. Students cannot browse through structured assignments, however, until the assignments are removed from the assignment list. They can then browse through the assignments by using the browse menu.

## BROWSING THROUGH PUBLISHED LESSONS: BROWSE

Students can browse through published lessons that are not assigned to their group and are not restricted to another group. The BROWSE option from the student menu displays a menu of published lessons. A typical browse menu is shown in Figure 4–4.



```
COURSEWARE  Browse Menu
AUTHORING SYSTEM

                 What would you like to do?

       ANALOGIES     WORD RELATIONSHIP
       COMPLEX       THE COMPLEX SENTENCE
       GRAMMAR       GRAMMAR AND USAGE
       MODIFIER      MISPLACED MODIFIERS
       PARTS         PARTS OF SPEECH
   >>> PHONICS       INTRO TO PHONICS
       PREFIX        WORD PREFIXES
       PRONOUN       STUDY OF PRONOUNS
       SENTENCE      SENTENCE COMPLETION
       SIMPLE        THE SIMPLE SENTENCE
       SPELL         SPELLING DRILL
       SUFFIX        WORD SUFFIXES
       USAGE         GLOSSARY OF USAGE

PF2 = Help, PF4 = Exit
                                   Use ↑↓, then RETURN
```

MR-S-3655-84

Figure 4–4
Browse Menu

When students take an assigned lesson, C.A.S. collects performance statistics. Depending on the lesson, students may be able to stop a lesson and restart it where they left off. In browse mode, performance statistics are not collected, and lessons always start at the beginning.

When a lesson ends, the browse menu is redisplayed. A student presses PF4 to leave the browse menu and return to the student menu.

## SENDING AND RECEIVING MAIL: MAIL

The VAX/VMS MAIL utility allows students to send and receive system mail. Refer to the instructor and author chapters for information about the MAIL utility. Also, the *VAX/VMS Mail Utility Reference Manual* contains detailed information about MAIL. Students can refer to the *C.A.S. Delivery System Student Guide* for information about MAIL.

When students exit from MAIL, they return to the student menu.

## EXITING FROM C.A.S.: EXIT

To exit from C.A.S. students select the EXIT option from the student menu (or press PF4). In most cases, students do not return to VAX/VMS command level after exiting from C.A.S., but are automatically logged out of their VAX/VMS accounts by commands in their log-in command files.

# GLOSSARY

# GLOSSARY

**browse list:** a list of all the published lessons, regardless of group restrictions and assignments, that the C.A.S. system manager can access.

**browse menu:** a list of published lessons that varies from one group to another, depending on group restrictions and assignments.

**C.A.S. name:** the identification of a user to the C.A.S. system. A C.A.S. name is assigned by the instructor at registration. A user need not enter a C.A.S. name to run C.A.S. if VAX/VMS user name translation is set.

**default directory:** the directory that the user is working from. The system automatically looks in the default directory for files unless the user includes the name of a different directory in the file specification.

**directory:** a file used to locate a set of files on a disk volume. It contains a list of file names (including type and version number) and their unique internal identifications.

**group:** a collection of at least one instructor, as well as authors and/or students.

**group environment:** group attributes that are changed by the group instructor. Group attributes include a group description, a password flag, and a VAX/VMS user name translation flag.

**menu:** a list displayed on the monitor that contains options for performing C.A.S. functions such as accessing displays, lessons, reports, the MAIL utility, and other menus.

**password:** a confidential code used to protect a user's account from being accessed by unauthorized persons.

**publish:** a C.A.S. function that moves copies of the lesson-executable image file and its associated auxiliary files from the author's account to a lesson subdirectory.

**published lesson:** a lesson that is available to C.A.S. users.

**screen:** a picture on the monitor that can show information, display prompts for information, and offer options for changing the information shown. Each C.A.S. screen displays and prompts for information related to C.A.S. functions.

**selection arrow:** an indicator that users move up and down on the monitor to select options.

**structured assignments:** assignments done in the order prescribed by the instructor.

**subdirectory:** a directory file catalogued in a higher-level directory that lists additional files belonging to the owner of the directory.

**unstructured assignments:** assignments done in the order selected by each student.

**VAX/VMS account:** an account created by the VAX/VMS system manager to collect information on a user's demands on VAX resources such as disk space. Users must supply a VAX/VMS user name and a password to log in to a VAX/VMS account.

**VAX/VMS Username:** identification of a user to the VAX/VMS system.

# Index

# Index

## READER'S COMMENTS

**Note:** This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. Problems with software should be reported on a Software Performance Report (SPR) form. If you require a written reply and are eligible to receive one under SPR service, submit your comments on an SPR form.

Did you find errors in this manual? If so, specify by page.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Is there sufficient documentation on associated system programs required for use of the software described in this manual? If not, what material is missing and where should it be placed?

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Non-programmer interested in computer concepts and capabilities

Name ______________________ Date ______________________

Organization ______________________

Street ______________________

City ______________ State ______________ Zip Code or Country ______________

Please cut along this line.

Fold here

Do Not Tear – Fold Here and Staple

No Postage
Necessary
If Mailed In The
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD, MA

Postage Will Be Paid by:

digital

Software Publications
200 Forest Street MRO1–2/L12
Marlborough, Massachusetts 01752

DECLIT AA VAX K764C

C.A.S. delivery system :
user's guide

DECLIT AA VAX K764C

C.A.S. delivery system :
user's guide

| | | | |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

AA-K784C-TE
Printed in U.S.A.

X

PCSA

digital

# VMS Services for PCs Release Notes Version 3.0

Order Number: AA-LB64D-TE

94-003\049/12

PCSA

# VMS Services for PCs Release Notes, Version 3.0

Order Number AA–LB64D–TE

December 1989

| | |
|---|---|
| **Revision/Update Information:** | This document supersedes the *VAX/VMS Services for MS-DOS Release Notes*, Version 2.2, order number AA–LB64C–TE. |
| **Software Version:** | PCSA Version 3.0 or greater |

digital equipment corporation
maynard, massachusetts

**First Published, May 1988**
**Revised August 1988, April 1989, December 1989**

81423

# Contents

## Tables

# VMS Services for PCs Release Notes, Version 3.0

The following sections contain information about:

- Upgrading your server to PCSA V3.0 from PCSA V2.1 or earlier
- Known problems
- PCSA Manager Commands and Privileges
- PCDISK Error Messages
- Documentation
- Corrections

## Upgrading Your Server to PCSA V3.0 from PCSA V2.1 or Earlier

Before you can upgrade your server to PCSA Version 3.0, you have to decide whether or not you want to retain your current service database.

If you want to retain your service database, proceed to the section "Upgrading Your Server and Retaining Your Server Database."

If you do want to retain your service database, proceed to the section "Upgrading Your Server and Losing Your Server Database."

### Upgrading Your Server and Retaining Your Service Database

To upgrade your server to PCSA V3.0 and retain your current service database, you must apply all of the intermediate versions of the server up to and including PCSA V2.2.

### Upgrading Your Server and Losing Your Service Database

To upgrade your server to PCSA V3.0 and lose your current service database, you can install PCSA V3.0 directly onto your current server.

## Increasing File Server Performance

There are two methods you can use to increase the performance of the file server:

- Open file caching
- Read and write raw operations

### Open File Caching

The server uses a form of caching refered to as Open file caching, which is disabled by default. When enabled, Open file caching delays the closing of a file on the server, although the server tells the client the file has been closed. The server holds the file open for a time specified by the logical PCFS$CLOSE_DELAY. The default for PCFS$CLOSE_DELAY is 15 seconds.

Open file caching provides significant performance gains when running in enviroments where files are constantly being opened and closed, as in database manipulation.

There is a side effect on VMS interactive users and any applications that may be running under VMS and sharing files with the DOS system. If an application or VMS user tries to access the file inside the time limit set by PCFS$CLOSE_DELAY, the following message is displayed:

```
file locked by another user, unable to open
```

When enabling PCFS$CACHE_OPEN_FILES, you must know what type of sharing is taking place on your server.

You can also tune the server's basic file data caching parameters to your systems configuration. For information, see *Server Administration with Commands*.

### Read and Write Raw Operations for PCSA V3.0 Clients

The PCSA V3.0 server can use the Read and Write raw modes of the LAN Manager protocol. This feature increases your file server performance if you are in an environment that is making large record transfers. You can only use this feature on a server that is servicing PCSA V3.0 clients.

To enable the Read and Write raw feature, edit SYS$STARTUP:PCFS_LOGICALS.COM and add the following lines:

```
DEFINE/SYSTEM/EXECUTIVE/NOLOG PCFS$READBRAW_OVER_BUFFER TRUE
DEFINE/SYSTEM/EXECUTIVE/NOLOG PCFS$WRITEBRAW_OVER_BUFFER TRUE
```

Restart the file server by entering:

```
$ @SYS$STARTUP:PCFS_STARTUP.COM
```

## Known Problems

This section describes the known problems and anomalies in the current release.

### Starting LASTDRIVER with Insufficient NPAGEDYN

Make sure you have sufficient amounts of NPAGEDYN and NPAGEVIR before attempting to install the server (see Chapter 2 of the *Installation Guide: VMS Services for PCs*). If a very low amount is available on the system when LASTDRIVER is started, a fatal bugcheck may occur. This could result in a system crash, depending on the state of the BUGCHECKFATAL system parameter.

### Naming Devices

Certain device names are reserved for use by PCSA, and should not be used as device labels for disk volumes or other devices. The list of PCSA reserved device names is:

- LASTx
- LADx
- LACx

Where:

| | |
|---|---|
| x | Is an optional digit that may be appended to the device name. |

Do not define device labels that begin with LAST, LAD or LAC. Instead, you can prefix disk names with DISK$.

### Creating Print Queues on Mixed Cluster Systems

On mixed cluster systems (VMS V5.x with VMS V4.7), all physical and generic print queues should be created on the 5.x system if they will be offered as PC print services. This is due to a difference in the VMS system service which is used to obtain queue information. For customers who are using mixed cluster environments but cannot move the print queues, a patch is available through Atlanta. The patch should only be considered if relocating the queue is not possible.

### PCDISK and Special Characters in DOS File Specifications

PCDISK returns an error when the following special characters are used within a DOS file specification.

| Character | Situation |
|---|---|
| ? | Is the first character. |
| ^ | Is the first character. |
| ( or ) | Is anywhere in the file specification. |

| Character | Situation |
|---|---|
| ! | Anywhere ignores all characters to the right of it. |

To avoid this problem, enclose the DOS file specification in quotation marks. For example:

```
A:\>dir "???.bat"
A:\>attrib "^abc.dat" /read-only
A:\>copy hoolay.dat "tim.(m)"
A:\>rename qwan.dat "ab!cd!e.x!y"
```

### File Server (PCFS_SERVER) Default State

The startup file has been modified to set the default server state to accept connection requests from workstations that are not registered in the DECnet node data base.

### Secondary Passwords

PCSA does not support secondary passwords. Attempting to connect to a dual password account will fail with an invalid service or password error.

### Password Expiration

VMS passwords can expire and a PC user may not be aware of it because there is no way for VMS to notify the PC client that the password is due to expire, as it does for interactive logins. It is recommended that the system administrator set all password expirations to the same date, thus all passwords on the system will expire simultaneously, and PC users can be notified in advance.

### Purging Old Log Files

Periodically, it is recommended that you purge the SYS$SPECIFIC:[PCSA] and SYS$COMMON:[PCSA] directories. This will remove old PCSA log files. You can use the /KEEP=*n* qualifier on the PURGE command, where *n* is the number of versions of each file that should be retained.

### The PCSA MENU Utility and the DEFAULT UAF Entry

When the PCSA MENU utility creates a user account, it uses the UAF entry DEFAULT to determine what the account should look like. This includes all flags and restrictions that can be applied to an account. If the DEFAULT UAF entry has been altered to meet the needs of the system, those changes will be reflected in any accounts created using the PCSA Manager.

## PCSA Manager Commands and Privileges

Table 1 contains a list of PCSA_MANAGER commands and the privileges required to use each command. This list supersedes the privileges listed in *Server Administration with Commands*.

**Table 1 PCSA Manager Commands**

| Command | Object | Privileges Required |
|---|---|---|
| ADD | CLIENT_OS | VOLPRO, LOG_IO, PHY_IO, OPER, SYSPRV |
| | NODE | OPER, SYSPRV, BYPASS |
| | SERVICE | OPER, SYSPRV |
| | TEMPLATE | OPER, SYSPRV, BYPASS |
| | USER | OPER, SYSPRV, BYPASS |
| | WORKSTATION | OPER, SYSPRV, BYPASS |
| BROADCAST | | OPER, SYSPRV |
| CLOSE | FILE_SERVER, FILE | OPER, SYSPRV |
| CREATE | DISK | OPER, SYSPRV |
| DELETE | DISK | OPER, SYSPRV |
| DENY | | OPER, SYSPRV |
| DISMOUNT | DISK | OPER, SYSPRV |
| GRANT | | OPER, SYSPRV |
| ADMINISTRATE (MENUS) | | No privileges required. |
| MODIFY | DISK | OPER, SYSPRV |
| | USER | OPER, SYSPRV, BYPASS |
| | WORKSTATION | OPER, SYSPRV, BYPASS |
| MOUNT | DISK | OPER, SYSPRV |
| REMOVE | CLIENT_OS | VOLPRO, LOG_IO, PHY_IO, OPER, SYSPRV |
| | NODE | OPER, SYSPRV |
| | SERVICE | OPER, SYSPRV |
| | TEMPLATE | OPER, SYSPRV, BYPASS |

**Table 1 (Cont.) PCSA Manager Commands**

| Command | Object | Privileges Required |
|---|---|---|
| | USER | OPER, SYSPRV, BYPASS |
| | WORKSTATION | OPER, SYSPRV, BYPASS |
| SET | CHARACTERISTICS | OPER, SYSNAM, SYSPRV |
| | DISK_SERVER CHARACTERISTICS | OPER, SYSNAM, SYSPRV |
| | DISK_SERVER SERVICE | OPER, SYSPRV |
| | FILE_SERVER CHARACTERISTICS | OPER, SYSPRV |
| | FILE_SERVER SERVICE | OPER, SYSPRV |
| SHOW | CLIENT_OS | No privileges required. |
| | DISK_SERVER CHARACTERISTICS | No privileges required. |
| | DISK_SERVER CONNECTIONS | No privileges required. |
| | DISK_SERVER COUNTERS | No privileges required. |
| | DISK_SERVER SERVICES | No privileges required. |
| | FILE_SERVER CHARACTERISTICS | No privileges required. |
| | FILE_SERVER CONNECTIONS | No privileges required. |
| | FILE_SERVER COUNTERS | No privileges required. |
| | FILE_SERVER OPEN_ FILES | No privileges required. |
| | FILE_SERVER SERVICES | No privileges required. |
| | FILE_SERVER SESSIONS | No privileges required. |
| | FILE_SERVER STATUS | No privileges required. |
| | TEMPLATES | No privileges required. |
| | USERS | OPER, SYSPRV |
| | VERSION | No privileges required. |

**Table 1 (Cont.) PCSA Manager Commands**

| Command | Object | Privileges Required |
|---|---|---|
| | WORKSTATIONS | No privileges required. |
| START | DISK_SERVER CONNECTIONS | OPER, SYSPRV |
| | FILE_SERVER CONNECTIONS | OPER, SYSPRV |
| | FILE_SERVER LOGGING | OPER, SYSPRV |
| STOP | DISK_SERVER CONNECTIONS | OPER, SYSPRV |
| | FILE_SERVER CONNECTIONS | OPER, SYSPRV |
| | FILE_SERVER LOGGING | OPER, SYSPRV |
| ZERO | DISK_SERVER COUNTERS | OPER, SYSPRV |

## PCDISK Error Messages

The following PCDISK error messages supersede those in *Server Administration with Commands*.

**PCDISK-E-ALOUTRANG, ALLOCATION quantity must be within 'min' to 'max' blocks**

*Explanation:* PCDISK could not create the specified virtual disk file with the /ALLOCATION size specified.

*User Action:* Use an allocation quantity within the identified range for that size disk. For further assistance, enter:

```
HELP CREATE /SIZE
```

The information displayed contains a table of supported virtual disk sizes along with their valid allocation quantity ranges.

**PCDISK-E-EACCDRV, Error accessing drive 'drv:'**

*Explanation:* PCDISK could not access the DOS drive.

*User Action:* Check the extended error message displayed on the next line for further explanation. This is the emulation of the DOS "Not ready error reading drive X:" "Abort, Retry, Fail?". It will only occur on connections to a physical diskette and PCSA virtual disk service connections.

**PCDISK-E-EATTRIB, Error setting attributes on file(s) 'file-spec'**

*Explanation:* PCDISK could not set the DOS file attributes on the specified files.

*User Action:* Check the extended error message displayed on the next line for further explanation.

**PCDISK-E-ECONN, Error connecting drive 'drv:'**

*Explanation:* PCDISK could not connect to the specified drive letter.

*User Action:* Check the extended error message displayed on the next line for further explanation.

**PCDISK-E-ECONNECT, Error connecting drive 'drv:' as 'DOS-dev-spec'**

*Explanation:* PCDISK could not connect the drive letter to the specified DOS device.

*User Action:* Check the extended error message displayed on the next line for further explanation.

**PCDISK-E-ECHDIR, Error changing directory to 'directory-spec'**

*Explanation:* PCDISK could not change default directories.

*User Action:* Check the extended error message displayed on the next line for further explanation.

**PCDISK-E-ECOPY, Error copying 'file-spec' to 'file-spec'**

*Explanation:* PCDISK could not copy the file.

*User Action:* Check the extended error message displayed on the next line for further explanation.

**PCDISK-E-ECREATE, Error creating 'virtual-disk-file'**

*Explanation:* PCDISK could not create the specified virtual disk file.

*User Action:* Check the extended error message displayed on the next line for further explanation.

**PCDISK-E-EDELETE, Error deleting file 'file-spec'**

*Explanation:* PCDISK could not delete the file.

*User Action:* Check the extended error message displayed on the next line for further explanation.

**PCDISK-E-EDINUSE, Specified drive is currently in use**

*Explanation:* This message is displayed when you specify the FORMAT command with a /DEVICE=diskette_device and a drive letter that are currently connected to something other then the specified device.

*User Action:* Either use a free drive letter or specify the correct device for that letter.

Subsequent formats of a diskette device that was initially specified with the /DEVICE qualifier need only specify the drive letter.

**PCDISK-E-EDISCONNECT, Error disconnecting drive 'drv:'**

*Explanation:* PCDISK could not disconnect the specified drive letter.

*User Action:* Check the extended error message displayed on the next line for further explanation.

**PCDISK-E-EEXPORT, Error exporting 'DOS-file-spec' to 'VMS-file-spec'**

*Explanation:* PCDISK could not copy the DOS file to VMS.

*User Action:* Check the extended error message displayed on the next line for further explanation.

**PCDISK-E-EFORMAT, Error formatting drive 'drv:' disk 'DOS-dev-spec'**

*Explanation:* PCDISK could not format the DOS device.

*User Action:* Check the extended error message displayed on the next line for further explanation.

**PCDISK-E-EIMPORT, Error importing 'VMS-file-spec' to 'DOS-file-spec'**

*Explanation:* PCDISK could not copy the VMS file to DOS.

*User Action:* Check the extended error message displayed on the next line for further explanation.

**PCDISK-E-EINDRV, Invalid drive specification**

*Explanation:* PCDISK found an invalid DOS drive specification. You may have:

- Omitted a required DOS drive command parameter.
- Specified an unconnected drive where a connection is required.
- Specified a connected drive where a free drive is required.
- Used an invalid syntax.

*User Action:* The correct syntax is an upper or lower case alphabetic letter followed by a colon. For example:

```
A:
b:
Z:
a:
d:
z:
```

**PCDISK-E-EMKDIR, Error creating subdirectory 'dir-spec'**

*Explanation:* PCDISK could not create the subdirectory.

*User Action:* Check the extended error message displayed on the next line for further explanation.

**PCDISK-E-ENOCONN, No established connections**

*Explanation:* There are currently no established connections associating a DOS drive with a DOS device.

*User Action:* Use the USE command to establish a connection. Enter the following command to display information about supported DOS devices that may be connected:

```
HELP DOS_devices
```

**PCDISK-F-ENOMEM, Insufficient memory for allocation of file or device buffers**

*Explanation:* PCDISK got an error when attempting to allocate memory for required data structures or buffers to perform the specified command.

*User Action:* Contact your system manager.

**PCDISK-E-ERDONLY, Specified drive is connected read only**

*Explanation:* You tried to write to a drive that is connected read-only.

*User Action:* Disconnect the drive, reconnect it with write access, and reenter the command.

**PCDISK-E-ERENAME, Error renaming 'file-spec' to 'file-spec'**

*Explanation:* PCDISK could not rename the file.

*User Action:* Check the extended error message displayed on the next line for further explanation.

**PCDISK-E-ERMDIR, Error removing subdirectory 'dir-spec'**

*Explanation:* PCDISK could not remove the subdirectory.

*User Action:* Check the extended error message displayed on the next line for further explanation.

**PCDISK-E-ETYPE, Error typing file 'file-spec'**

*Explanation:* PCDISK could not type the file.

*User Action:* Check the extended error message displayed on the next line for further explanation.

**PCDISK-E-EVOLUME, Error setting drive 'drv:' volume label to 'label'**

*Explanation:* PCDISK could not set the volume label on the specified drive.

*User Action:* Check the extended error message displayed on the next line for further explanation.

**PCDISK-E-EXCOPY, Error performing XCOPY from 'file-spec' to 'file-spec'**

*Explanation:* PCDISK could not perform the XCOPY command specified.

*User Action:* Check the extended error message displayed on the next line for further explanation.

**PCDISK-E-FNF, File not found**

*Explanation:* PCDISK could not find the specified file.

*User Action:* Check your default path, or your path specification, and verify that the device, directory, file name, and file type were all specified correctly.

**PCDSHR-E-BADBPB, Non-DOS disk, or partitioned hard disk format**

*Explanation:* The DOS device has an invalid BIOS parameter block.

*User Action:* If it is a virtual disk file, try connecting it using the /HARD_DISK qualifier. It may be in hard disk format.

**PCDSHR-E-BADFAT, Invalid FAT format**

*Explanation:* PCDISK detected an invalid file allocation table. The virtual disk is probably corrupted.

*User Action:* Try connecting to it with a DOS PC through PCSA and run CHKDSK.

**PCDSHR-E-BADPTBL, Invalid partition table entry**

*Explanation:* The virtual hard disk partition table has an invalid entry.

*User Action:* Try mounting the virtual disk without the /HARD_DISK qualifier. It may be in floppy format.

**PCDSHR-E-CONNREJ, LAD connect rejected**

*Explanation:* This message is displayed when connecting to a PCSA supported virtual disk service. The connection was rejected because of an invalid nonexistent password.

*User Action:* Reenter the command and specify a valid password.

**PCDSHR-E-COPYSELF, File cannot be copied onto itself**

*Explanation:* You tried to copy a file to itself.

*User Action:* The input and output specifications in a COPY operation must be different.

**PCDSHR-E-CTRLC, CTRL/C pressed, operation canceled**

*Explanation:* This is an internal error message which will never be issued by the CLI. The PCDISK shareable image has canceled the current operation as a result of operator interruption, and has done any cleanup required. Cancel interruption is supported and this error is returned when it occurs.

*User Action:* No user action is required.

**PCDSHR-E-DEVALLOC, Device is allocated to another user**

*Explanation:* You tried to mount a diskette device that is allocated to another user.

*User Action:* Enter the following command to determine who has the device mounted:

```
SHOW DEV/FU devx:
```

**PCDSHR-E-DEVMOUNT, Device is already connected**

*Explanation:* You tried to mount a diskette device that is already mounted.

*User Action:* Enter the following command to assist in recovery of the mounted device:

```
SHOW DEV/FU devx:
```

**PCDSHR-E-DINUSE, Specified drive is currently in use**

*Explanation:* You tried to connect a virtual disk to a drive letter that currently has a DOS device connected.

*User Action:* Reenter the command and specify an unused drive letter or handle.

**PCDSHR-E-DNF, Directory not found**

*Explanation:* PCDISK could not find the specified subdirectory.

*User Action:* Make sure your path is specified correctly. Make sure you specified a directory.

**PCDSHR-E-DNSUPP, Device type not supported**

*Explanation:* You specified a device type that is not supported.

*User Action:* Specify a supported device. PCDISK supports:

- RX33s at 1.2MB
- RX23s at 1.4MB

**PCDSHR-E-EACCES, Permission denied on file operation**

*Explanation:* You tried one of the following:

- To write to a read-only file
- To do a file operation to a subdirectory

*User Action:* Do not attempt to write to a read-only file. If you want to delete a subdirectory, use the RMDIR command.

**PCDSHR-E-EBADF, Invalid file handle**

*Explanation:* An invalid file handle was passed to a PCDISK callable routine.

*User Action:* Check for programming errors.

**PCDSHR-E-EBUFTOSM, Buffer is to small to fit requested information.**

*Explanation:* A PCDISK routine was called with an insufficient buffer size to contain the requested information.

*User Action:* Check for programming errors.

**PCDSHR-E-EDEXST, Directory already exists**

*Explanation:* You tried one of the following:

- To create a subdirectory that already exists
- To create a subdirectory with the same name as an existing file

*User Action:* Make sure the subdirectory does not already exist. Do not attempt to create a subdirectory with the same name as an existing file.

**PCDSHR-E-EDNOEMP, Directory is not empty**

*Explanation:* PCDISK cannot remove a directory unless it is empty.

*User Action:* Delete the files within the directory and reenter the command.

**PCDSHR-E-EDRONLY, Target drive is connected read only**

*Explanation:* You tried to write to a drive that is connected read-only.

*User Action:* Check for programming errors. Disconnect the drive and reconnect it with write access.

**PCDSHR-E-EDSELCT, Cannot remove currently selected directory**

*Explanation:* PCDISK cannot remove the current default directory.

*User Action:* Use the CHDIR command to move to an upper level directory or the root directory. Reenter the command.

**PCDSHR-E-EFEXST, File already exists**

*Explanation:* You tried to do a non-supersede file create when the specified file already exists. This usually occurs on a RENAME command.

*User Action:* Do not attempt to rename a file to a name that is the same as an existing file.

**PCDSHR-E-EFRONLY, Attempted write access to a read only file**

*Explanation:* You tried to write to a file that has the DOS read-only attribute set.

*User Action:* Use the ATTRIB or SET FILE command to modify the file attribute. Reenter the command.

**PCDSHR-E-EINDIR, Invalid directory specification**

*Explanation:* PCDISK found an invalid directory specification.

*User Action:* Check delimiters in the directory specification.

**PCDSHR-E-EINDRV, Invalid drive specification**

*Explanation:* PCDISK found an invalid DOS drive specification. You have specified an unconnected drive where a connected drive is required.

*User Action:* Check the delimiters in the drive specification.

**PCDSHR-E-EINFIL, Invalid file specification**

*Explanation:* PCDISK found an invalid file specification.

*User Action:* Check delimiters in file specification.

**PCDSHR-E-EINPAT, Invalid path specification**

*Explanation:* PCDISK found an invalid path specification. This message is displayed if the specified path is greater than 63 characters.

*User Action:* Use the CHDIR command to move closer to the object directory. This requires a smaller path specification.

**PCDSHR-E-EINVAL, Invalid argument**

*Explanation:* An invalid argument was passed to a PCDISK callable routine.

*User Action:* Check for programming errors. This message is currently displayed only by the seek routines.

**PCDSHR-E-EMFILE, Too many open files**

*Explanation:* You tried to open more files than PCDISK allows.

*User Action:* Check for programming errors. You can open 64 DOS files and 10 RMS files.

**PCDSHR-E-EMTPTBL, Empty partition table**

*Explanation:* The virtual disk has an empty partition table.

*User Action:* Try mounting the virtual disk without the /HARD_DISK qualifier. It may be in floppy format.

**PCDSHR-E-ENOMEM, Insufficient memory for allocation of file or device buffers**

*Explanation:* An error occurred while attempting to allocate memory for required data structures or buffers to perform the specified function.

*User Action:* Check user and system parameters limiting memory resources.

**PCDSHR-E-ENOSPC, No space left on drive**

*Explanation:* You tried to allocate space on a drive that is full.

*User Action:* Use a larger virtual disk, multiple diskettes or purge the current disk.

**PCDSHR-E-ENOTCONN, Drive not connected**

*Explanation:* You tried to disconnect a drive that is not connected.

*User Action:* Make sure your drive specification is correct.

**PCDSHR-E-ERECSIZE, Fixed length record size must be even**

*Explanation:* You specified a record size that is not an even number.

*User Action:* When exporting a file to a fixed length record format, you must specify the size as an even number. The default is 512 and the range is 2 to 32766.

**PCDSHR-E-ESRONLY, Attempted to remove a read only subdirectory**

*Explanation:* You tried to remove a subdirectory that has the DOS read-only attribute set.

*User Action:* Use the ATTRIB or SET FILE command to modify the subdirectory attribute. Reenter the command.

**PCDSHR-E-EXDEV, Cannot rename a file to another device**

*Explanation:* You tried to rename a file across a drive.

*User Action:* Use the COPY and DEL commands to move a file to a different drive.

**PCDSHR-E-FNF, File not found**

*Explanation:* PCDISK could not find the specified file.

*User Action:* Check your default path, or your path specification. Make sure you specified the device, directory, file name, and file type correctly.

**PCDSHR-E-HDINUSE, Specified drive, or sequence drive for partition, is in use**

*Explanation:* PCDISK requires that virtual disks in hard disk format have their partitions mounted in sequence from the initially specified drive letter. This is done so that the programmer may assume that the third partition is mounted two drive letters after the specified drive letter within the mount command.

*User Action:* Check for specified, or sequence drive that may be currently mounted.

**PCDSHR-E-HDSKERR, LAD Server disk error**

*Explanation:* This message is displayed when connecting to a PCSA supported virtual disk service. A hard disk error has occurred on the server.

*User Action:* Do one of the following:

- Disconnect and try to reconnect.
- Contact the system manager for the server offering the virtual disk service.

**PCDSHR-E-IMPWLDOUT, Import file is incompatible, output spec cannot have wildcards**

*Explanation:* When importing a file whose name is incompatible with DOS file naming conventions, you cannot use wildcards within the output file specification.

*User Action:* You must specify a valid DOS file name for output.

**PCDSHR-E-INCPVER, Incompatible DAD protocol version**

*Explanation:* This message is displayed when there is an incompatibility between protocols when connecting to a PCSA supported virtual disk service.

*User Action:* Contact your system manager.

**PCDSHR-E-INSUFCSP, Insufficient contiguous space on device**

*Explanation:* There is insufficient contiguous space on the specified device. Creating a virtual disk file by default is "contiguous-best-try".

*User Action:* If a contiguous file is required, you can purge or do a VMS backup and restore to create more contiguous space for your virtual disk file.

**PCDSHR-E-IVSIZE, Unrecognized or ambiguous size specification**

*Explanation:* Either the /SIZE= specification within the CREATE virtual disk command is invalid, or it needs more letters to make it unique.

*User Action:* Enter the following command for a list of valid keywords:

```
HELP CREATE /SIZE
```

**PCDSHR-E-LADINUS, Writable LAD service is in use**

*Explanation:* This message is displayed when connecting to a PCSA supported virtual disk service. Another user is already connected to the requested service.

*User Action:* Only one person at a time can connect and use a writable virtual disk service.

**PCDSHR-E-LADUNER, Unexpected LAD error: 'error #'**

*Explanation:* This message is displayed by the server when connecting to a PCSA supported virtual disk service.

*User Action:* Contact your system manager.

**PCDSHR-E-MAXCONX, Maximum connections exceeded**

*Explanation:* This message is displayed when connecting to a PCSA supported virtual disk service. The maximum number of connections allowed by the PCSA server has been exceeded.

*User Action:* Contact your system manager, who can increase the maximum number of allowable connections.

**PCDSHR-E-NDAVAIL, No device table entries available**

*Explanation:* You tried to mount a drive, but the maximum number of drives has been exceeded.

*User Action:* Check for programming errors. You cannot mount more than 26 drives (the number of letters in the alphabet).

**PCDSHR-F-NOMEM, Insufficient memory for allocation of file or device buffers**

*Explanation:* This message is displayed while attempting to allocate memory for required data structures or buffers to perform the specified function.

*User Action:* Check user and system parameters limiting memory resources.

**PCDSHR-E-NPHYDEV, Format may only be issued to a physical (non-virtual) device**

*Explanation:* You tried to do a low level physical device format to a non-physical device. This should never happen when using the CLI.

*User Action:* Check for programming errors.

**PCDSHR-E-OBSOLETE, Obsolete shareable image routine called, relink required**

*Explanation:* The routines which allocated data structures when importing and exporting files have been incorporated within other main routines. The entries have been left in the shareable image transfer vector table to maintain compatibility across versions.

*User Action:* Check for programming error.

**PCDSHR-E-SVCNOFF, LAD service not offered**

*Explanation:* The specified PCSA virtual disk service is not currently offered on the network.

*User Action:* Make sure you spelled the node and service specifications correctly.

## Documentation

The differences between VMS Services for PCs, Version 3.0 documentation and the documentation for earlier versions of VAX/VMS Services for MS-DOS are:

- Several books have new names, as shown in the following table:

| Former Name | New Name |
|---|---|
| *VAX/VMS Services for MS-DOS Installation Guide* | *Installation Guide: VMS Services for PCs* |
| *Setup and Management Using PCSA Manager Menu* | *Server Management with the Menu* |
| *VAX/VMS Services for MS-DOS Administration Guide* | *Server Administration with Commands* |

  These three books, plus the *Network Troubleshooting Guide* make up the documentation set and are packaged together in two binders.

- The *Server Management with the Menu* has been reorganized to reflect system administration tasks.
- Adding a workstation is now done by the Netsetup utility. Since you can no longer add a workstation from the PCSA Manager Menu, this task is not discussed in *Server Management with the Menu*.

  To add a workstation, see *Installation and Configuration Guide: DECnet PCSA Client for DOS (VMS Media)* in the documentation for DECnet PCSA Client for DOS software.

## Corrections

### SYSMWCNT Value

On Page 1-12 of *Installation Guide: VMS Services for PCs*, the following sentence is incorrect:

```
The required value of SYSMWCNT in this example is 755.
```

The sentence should read:

```
The required value of SYSMWCNT in this example is 756.
```

### Calculating NPAGEDYN for the File Server

The file server requires 310,000 bytes (NPAGEDYN) to run properly. If you have a prior version of PCSA installed, you have already allocated that space. Disregard the calculation in step 6, on page 1-14, of the *Installation Guide: VMS Services for PCs*.

**NOTE**
To tune the disk server cache, see *Server Administration with Commands*.

### NPAGEDYN Value

On Page 1-14 of *Installation Guide: VMS Services for PCs*, the following sentence is incorrect:

```
For example, if the current value of NPAGEDYN is 626,280:
```

The sentence should read:

```
For example, if the current value of NPAGEDYN is 326,280:
```

### NCP SHOW EXECUTOR command

On Page 1-15 of *Installation Guide: VMS Services for PCs*, the following sentence is incorrect:

```
NCP> NCP SHOW EXECUTOR
```

The sentence should read:

```
NCP> SHOW EXECUTOR
```

### Installing PC DECWindows on Server Not Running PCSA

In *Server Administration with Commands*, Chapter 1 describes how to install PC DECwindows on a server that is not running PCSA. On page 1-11, the NFT commands are incorrect. The following information is correct:

1. Use the Network File Transfer (NFT) utility to copy the program from the PCSA system device to the remote system. On a VAX system, enter the following text on **one** command line and then press the Return key:

   ```
   C:\>NFT COPY/IMAGE d:\XSERVER\REMOTE\PCX_VAX.ULT nodename
   "username password"::/usr/bin/pcx_server Return
   ```

   On a DECstation system, enter the following text on **one** command line and then press the Return key:

   ```
   C:\>NFT COPY/IMAGE d:\XSERVER\REMOTE\PCX_DECS.ULT nodename
   "username password"::/usr/bin/pcx_server Return
   ```

## Incorrect Table Title

In *Server Administration with Commands*, the title of Table 9-1 should be PCSA MANAGER Commands, not PCSA MANAGER File Server Commands.

DECLIT AA VAX LB64D
PCSA : VMS services for PCs : release notes : version 3.0

DECLIT AA VAX LB64D

PCSA : VMS services for PCs
: release notes : version
3.0

| | | | |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

*0206159*

Printed in U.S.A.

X

# VAX RALLY

digital

Release Notes

Order Number: AA-KZ26A-TE

94-003(049/05

# VAX RALLY Release Notes

Order Number: AA-KZ26A-TE

September 1988

VAX RALLY is an application generator that provides a menu and forms-based environment for creating, modifying, and executing applications. This manual summarizes new features, changes, and known restrictions, limitations, and problems with VAX RALLY V2.0.

| | |
|---|---|
| **OPERATING SYSTEM:** | VMS |
| **SOFTWARE VERSION:** | VAX RALLY Version 2.0 |

digital equipment corporation, maynard, massachusetts

LID# 76834

# Contents

## Tables

## How to Use This Manual

The VAX RALLY software, also referred to as RALLY, is an application generator that provides a menu- and forms-based environment for creating, modifying, and executing applications. This manual describes new features in this version of RALLY and warns you about known restrictions and problems.

This document serves as the release notes for both VAX RALLY and the VAX RALLY Run-Time Option.

VAX RALLY includes both the VAX RALLY Definition System and the VAX RALLY Run-Time System.

The VAX RALLY Run-Time Option includes only the Run-Time System. You can use this option to run existing RALLY applications, but you cannot use it to create new RALLY applications or modify existing ones.

### Intended Audience

This manual is intended for all RALLY installers and application definers. This manual is not intended for users who are using RALLY to run applications created for them by others.

## Operating System Information

Information about the versions of the operating system and related software that are compatible with this version of RALLY is included in the RALLY media kit, in the Installation Guide, or in the Before You Install letter.

For information on the compatibility of other software products with this version of RALLY refer to the System Support Addendum (SSA) that comes with the Software Product Description (SPD). You can use the SPD/SSA to verify which versions of your operating system are compatible with this version of RALLY.

## Structure

This manual has four chapters and one appendix.

| | |
|---|---|
| Chapter 1 | Describes the new and changed features in RALLY Version 2.0. |
| Chapter 2 | Discusses the problems fixed in this version of RALLY. |
| Chapter 3 | Explains the problems and restrictions in this version of RALLY and provides additional notes. |
| Chapter 4 | Provides information for the existing A-TO-Z Application Generator customer who is now using RALLY. |
| Appendix A | Lists the new terminology used in V2.0. |

## Related Documents

Refer to the following manuals for more information about RALLY:

- *Introduction to VAX RALLY*—Provides an overview of RALLY.
- *VAX RALLY Application User's Guide*—Provides step-by-step instructions on using a RALLY application at run time.
- *VAX RALLY Command Reference Manual*—Describes RALLY commands.
- *VAX RALLY Installation Guide*—Provides information about installing RALLY.
- *VAX RALLY ADL User's Guide*—Describes the RALLY Application Development Language (ADL).
- *VAX RALLY Definition System User's Guide*—Provides step-by-step instructions on defining a RALLY application.
- *VAX RALLY Reference Manual and Master Index*—Explains major RALLY terms and concepts.

- *VAX RALLY Guide to Application Development*—Provides application development guidelines within the framework of a suggested development methodology.

## Conventions

This manual uses the following conventions:

| Convention | Meaning |
|---|---|
| RETURN | Named keys appear in uppercase letters. |
| KPn | Key names that begin with KP indicate keys on the numeric keypad on the right side of the terminal keyboard. |
| GOLD-x | The hyphen in key names means that you press the two keys in the order listed. In this case, press the GOLD key (PF1) and then the character key, represented here by x. |
| CTRL/x | The slash in the key name means that you press the two keys simultaneously. In this case, press the CTRL key (control) and, while holding down the CTRL key, press the character key, represented here by x. |
| 'delete word' | RALLY command names appear in lowercase letters inside single quotation marks. |
| *Edit Application* | Names of Definition System menus and form/reports appear in italics. |
| $ | The dollar sign is used to indicate the DCL prompt. This prompt can be different on your system. |
| Menu paths | Menu paths are indicated in text by menu entry numbers, followed by keywords from the menu entry in parentheses. For example, take menu path 3 6 2 (application message edit). RALLY accepts input in either form, so you can choose to navigate by entering numbers or by entering text. You need only enter enough characters from each keyword to make it unique. Use whichever method makes it easy for you to navigate and recall paths. |

## References to Products

The RALLY documentation to which this manual belongs refers to the following products by their abbreviated names:

- VAX CDD/Plus software is referred to as CDD/Plus.
- VAX DATATRIEVE software is referred to as DATATRIEVE.
- VAX DBMS software is referred to as VAX DBMS.
- VAX Rdb/VMS software is referred to as Rdb/VMS.
- VAX TEAMDATA software is referred to as TEAMDATA.
- VAX ALL-IN-1 software is referred to as ALL-IN-1.

# New and Changed Features in Version 2.0 1

This chapter summarizes the features which have been added for Version 2.0 (V2.0). It also includes significant changes in the behavior of Version 1.1 (V1.1) features.

The RALLY Run-Time System is now available as a separate orderable option. This document includes information for both the the Run-Time System and the Definition System.

## 1.1 Improved Usability of Definition System

Extensive modifications were made to the RALLY Definition System with the explicit goal of improving usability (ease of use and ease of learning). A summary follows:

- New menus and menu trees.

  The new menus include several menus with objects and operations in different columns (you make one choice from each column on such menus). Where features have been moved to different places in the menu tree, the appropriate forms have been reorganized as well.

- Direct object manipulation operates on a number of Definition System forms.

  In V1.1, you were able to create and delete menu choices using RALLY's form/report commands, such as 'insert record' and 'delete record'. Manipulating objects in this manner is referred to as direct object manipulation. In V2.0, this style of interface is used for editing most RALLY subobjects, including DSD fields and form/report fields.

- The SELECT key ('local_function' command) can be used to navigate through your application by following connected objects.
- Creating and editing form/report groups is easier. The visual representation of group relationships in a form/report is clearer.
- View DSDs, foreign key links, data source keys, and break-up DSDs are hidden from the definer. RALLY automatically makes the necessary connections between groups when you specify certain types of relationships, and RALLY automatically modifies the hidden objects when you modify the relationship using a Definition System form.
- Terminology has been revised. Many of the terms used to refer to RALLY features have been changed. Refer to Table A-1, Terminology Changes, for a listing of the old terms with their corresponding new terms.
- Location coordinates are easier to specify and understand. The "advanced location options" menu has been replaced by a pop-up subform you can invoke from the location coordinates form.

The following sections describe the major usability improvements in more detail.

### 1.1.1 Direct Object Manipulation

In RALLY V2.0, you define subobjects of form/reports and DSDs using a new technique, called direct object manipulation. In this technique, the subobjects are represented on the screen by a list of records. For example, the fields in a particular data source definition are shown as a list. To create a new subobject (for example, a new DSD field), you use the regular RALLY form/report commands to insert a new record into the list to represent the new field. To delete a subobject, you delete the record representing that subobject. To edit a subobject, you move the cursor to the name of the subobject and use 'local_function' by pressing the SELECT key. To rename a subobject, you move the cursor to the name of the subobject and type the new name.

Direct object manipulation form/reports are used in the Definition System for the following:

- Fields in a DSD
- Groups in a form/report
- Objects in a form/report group
- Conditional next field definitions in a form/report
- Conditional lists of values in a form/report

- Form/report packets for a form/report
- Fields in a relation

In most of these cases, direct object manipulation replaces the menus formerly used for creating, editing, deleting, and renaming these subobjects. For form/report packets, you can use either direct object manipulation or the object/operation menu.

### 1.1.2 Local Functions

The Definition System uses RALLY's own "local function" feature to permit you to edit an object from a context in which the object name appears. To use this feature, move the cursor to the name of the desired object, then use the 'local_function' command (press the SELECT key) to edit that object. For example:

- To edit a form/report group, go to the *Groups in this Form/Report* form, move the cursor to the desired group name, then press SELECT.
- If you select option 6 (data) on the *Edit a Form/Report Group* form, the DSD name and the field names are displayed. You can edit the DSD from this context by moving the cursor to the DSD name and pressing SELECT.

There is no limit to the number of times that 'local_function' can be used successively. For example, you could begin by editing a task and follow references to a menu called by the task, to a form/report packet called as a choice from the menu, to the form/report called by that form/report packet, to a parameter packet used to validate a form/report field, and to the ADL procedure called by the parameter packet.

The local function capability applies only for certain fields on Definition System forms. Legends on the forms indicate which fields this feature applies to and how to invoke the local function.

### 1.1.3 Indentation Used to Show Group Hierarchical Relationships

In the list of records representing the groups in a form/report, the vertical and horizontal position of the group name reflects the place the group occupies in the form/report hierarchy. When creating a new group, you create a new record below the parent group and indent the name of the new group so that it is further to the right than the parent group but aligned with any sibling groups.

### 1.1.4 New Ways to Specify Group Links

View DSDs, foreign key links, DSD keys, and breakup DSDs no longer appear in the Definition System. When creating a new data group, you must specify the link to connect the new group to any parent and child data groups. Each link can be either a "form/report join," corresponding to the former usage of view DSDs and foreign key links, or a "control break," corresponding to the former usage of breakup DSDs. Form/report joins and control breaks can be combined in the same form/report, with some restrictions. After a group is created, you can edit the links between groups by taking option 9 (parent) and option 10 (child) on the *Edit a Data Group* menu.

Regardless of whether the application was created using RALLY V1.1 or V2.0, you can display and edit the links between groups using the RALLY V2.0 Definition System.

### 1.1.5 Terminology Changes

Many RALLY V1.1 terms have been changed for V2.0. The goal in each case has been to select a term that most accurately reflects the function or feature, yet is also as succinct as possible. Sample changes include substituting "Definition System" for "Dialog," substituting "application file" for "AFILE," and substituting "RALLY Report Utility" for "Script Writer." The list in Appendix A, Terminology Changes, shows the old (V1.1) terminology and the new (V2.0) equivalents.

Also note the following changes:

- Several choices on Definition System forms have been rephrased from a negative form to a positive form of the statement. For example, in the *Record Operation Options for a Group* form, "Do not commit changes when leaving record" has been changed to "Commit when leaving record". The default behavior has not changed from V1.1 to V2.0, just the terminology.
- The word "use" has been dropped from certain RALLY command names. For example, the V1.1 commands 'use local_function' and 'use macro' are now called 'local_function' and 'macro number'. Many of the commands that start with "use" are still being recognized by RALLY V2.0. However, you are strongly encouraged to use the V2.0 commands, because support for the remaining V1.1 commands might be removed in a future release of RALLY if required, in order to support future commands that might be added to the product.

### 1.1.6 Location Coordinates Specified as Start Point and Object Size

In V1.1, location coordinates for an object (field, group, text area, and so on) were specified in terms of a start point and an end point. For example, for a group three lines long and forty columns wide, you might have specified starting coordinates of 10 and 20 (line 10, column 20) and ending coordinates of 12 and 59.

In V2.0, location coordinates are specified in terms of a start point and a size (number of lines and columns). Using the previous example, the location coordinates would be expressed as 10 and 20 (line 10, column 20), and 3 and 40 (3 lines, 40 columns). The text and legends on Definition System forms for changing location coordinates clearly indicate what information to enter for each item.

To specify information about object location options, type "Y" in the "Opt" field on the *Location Coordinates and Options* form. This information appeared on the *Advanced Location Attributes* menu in the V1.1 Dialog.

### 1.1.7 Additional Usability Improvements

In addition to the usability improvements previously described, RALLY V2.0 includes the following:

- The form/report screen editor is available directly when you edit an individual group or text area. When you invoke the screen editor from a group or text area, the image editor state is automatically set up for that group or text area.

- Creation of list of values groups has been simplified.

- Most list of values have been enlarged to 80 columns. To invoke this feature, press CTRL/E or use 'go_to end_of_line' while the cursor in the LOV rests on a field which extends off the screen.

- Legends have been revised. They now include LK201 key definitions as well as RALLY command names.

- An option has been added to delete objects despite reference to them from other objects. When you use this option, the object you indicated that you wanted to delete is replaced with a placeholder object.

  If you later change the placeholder back into a regular object, the new object inherits the relationships of the deleted object.

- The form/report screen editor has been changed so that deleting a field while in the screen editor prevents displaying of the field. The field can be redisplayed in the usual way: that is, by editing its location options (to change "Displayed on:" from NONE to something else) and restoring it to the group's visitation order.

- A 'delete previous_word' command has been added to the Definition System screen editor and to the Run-Time System. This command is mapped to the LINEFEED key (F13 on LK keyboards). Previously, this key was mapped to the 'delete line' command. (In V2.0, GOLD-LINEFEED is mapped to 'undelete word'.)
- The behavior of the 'delete word', 'previous word', and 'next word' commands has been changed to increase compatibility with DIGITAL editors such as EDT and TPU. These commands now recognize common word delimiters such as space, colon, and end-of-line.

## 1.2 Automatic Application Upgrading

To accommodate the many enhancements to RALLY V2.0, the format of RALLY application files (formerly referred to as AFILEs) has been changed.

To minimize any inconvenience that might result from this change, both the V2.0 Run-Time System and the V2.0 Definition System have the ability to work with V1.1 applications. When you use the V2.0 Run-Time System with a V1.1 application, RALLY will upgrade the application in memory as it executes, and the application will not be modified. However, the first time that you edit your application using the V2.0 Definition System, RALLY will create a new version of your application file that uses the new format. After your application file has been upgraded in this way, you will not be able to run or edit the file with V1.1.

RALLY will not let you use the /NONEW qualifier with the DCL command RALLY EDIT when you are upgrading an application to use the new format.

See Chapter 3, Problems, Restrictions, and Other Notes, for additional information on application file upgrades.

## 1.3 VAX RMS File Support

VAX RMS has been added to the list of data sources supported by VAX RALLY. Data source definitions for RMS files are created by reference to descriptions of the files in CDD/Plus. These descriptions are usually created by using the CDD/Plus utility CDO; the descriptions cannot be created from within RALLY.

RALLY supports multi-user read and write access to sequential and indexed files and to relative files in sequential mode. Form/reports can combine data from all access methods, for example, from a relational database and from an RMS file.

VAX DATATRIEVE is not used to implement RMS support and is not required for use of RMS in RALLY. Read-only access to RMS files through DATATRIEVE is still available, and might still be desirable for solving some application problems.

Some new RALLY features do not apply to RMS file access; among such features are parameters for record selection expression and extended QBE (query-by-example).

## 1.4 Enhancements to the Utilities

RALLY V2.0 includes many changes and enhancements to the utilities. These improvements are described in the following sections.

### 1.4.1 Improvements to the Application Reporter

The V1.1 "Script Writer" has been replaced with the RALLY Report Utility. The Report Utility has the following capabilities:

- Application reports are available interactively from anywhere in the Definition System, including while editing an ADL procedure. The Report Utility is brought up in a separate RALLY task, so that you can switch back and forth between the report and your editing session. This capability is invoked by pressing GOLD-R.
- The formatting and clarity of the reports has been substantially improved.
- The report facility hides view DSDs, break-up DSDs, foreign key links, data source keys, and access-method-dependent objects. The information associated with a hidden object is now included with the description of the form/report group that is associated with the hidden object.
- You can select from a variety of reports, including a report on the entire application, on all objects of a specified type, on a specific named object, or a database cross-reference report.
- You have the option of including cross reference information in your reports, so that, for each application object, you get a list of the application objects that reference that object.
- A new report is available that shows all of the data sources referenced by your application.
- You can cut and paste information from an interactive report to the ADL editor.

### 1.4.2 RALLY Update Utility

This new utility compares the data descriptions for your data sources with the information stored in your RALLY application file and changes the application file as required to make the file consistent with the data. For example, if you change the data type of a field from integer to string, you can correct your application to reflect the change with the Update Utility.

The reserving list of the transaction parameter block for Rdb/VMS data source definitions is now modified by the Update Utility and not by the Compact Utility, as with V1.1. The reserving list specifies the relations that will be reserved when RALLY attaches to the database.

### 1.4.3 DCL Commands for Utilities

The RALLY DCL command line now provides DCL interfaces to the following RALLY utilities: Compact, Report, Merge, Update, Verify, Load/Unload Messages, Load/Unload Data, Define Keys, and Integrate. Type HELP RALLY at the DCL prompt for information about the available commands.

### 1.4.4 Simplification of Forms for Using Utilities

With V1.1, when the Verify Utility is used from its forms interface, the user does not have to name the application to be verified—RALLY implicitly verifies the current application.

In V2.0, the Verify Utility is used as the model for all utilities. The utilities implicitly operate on the application being edited, and, when you leave the utility, you will find yourself editing the application created in the utility, not the application used as input for the utility.

## 1.5 Procedural Programming Enhancements

Procedural programming enhancements include changes to the callable interface, extensions to external program links, and improvements to ADL.

### 1.5.1 Callable Interface for Run-Time System

There is now a callable interface to the RALLY Run-Time System. In many cases, you will be able to use the "simple" interface (calling RALLY$RALLY); in some cases you may need to use the "full" interface (consisting of several routines). See the *VAX RALLY Definition System User's Guide* and the *VAX RALLY Reference Manual and Master Index* for detailed information on the callable interface.

### 1.5.2 Extensions to External Program Links

Significant enhancements to RALLY's interface to 3GL programs are supported in this version. In summary:

- A new type of external program link is available. This new link type allows you to invoke routines that conform to the VAX Calling Standard. Some restrictions apply, but most of the popular combinations of data type, argument passing mechanism, and descriptor class are supported.
- The 3GL routines called by a RALLY application need no longer be in a single shareable image. Each external program link names the shareable image and routine to be called.
- You can use this new type of external program link to invoke VAX/VMS Runtime Library routines. You can also use this link to invoke the VAX/VMS utilities that have callable interfaces.

### 1.5.3 Enhancements to ADL

Several enhancements have been made to RALLY's Application Development Language (ADL), including the following:

- A series of lists of values (LOV) for object names are now available in the ADL screen editor. The lists display the names of the application objects that you might want to reference within an ADL procedure. You invoke the LOVs by pressing GOLD-SELECT and selecting an object type. When you select an object, its name will be copied into your ADL source code at the current cursor position.
- The CALL_CMD built-in function has been extended to accept RALLY commands, as well as command aliases. Commands (including any parameters or qualifiers) are passed as a string argument to CALL_CMD.

  For example, you can now specify CALL_CMD ('insert record next_group') or CALL_CMD (finsnext); both produce the same result.

  Use of aliases will continue to be supported, however, documentation focuses on the new methods. Use of aliases should be discouraged in new applications.
- Three new built-in functions have been added to assist with manipulation of character strings: SEARCH, LENGTH, and UPCASE.
- DEFINE_LOGICAL, DELETE_LOGICAL, and TRANSLATE_LOGICAL functions have also been added.

- A new function, GET_MODE, has been added. It returns the current mode of the form/report from which the ADL procedure was called.
- A built-in constant DB_EOS has been defined. The ADL built-in procedures DB_GET_FIRST and DB_GET_NEXT return this value when there are no more records to read.
- A new function, SET_TRACE(), writes each line of ADL code to the log as the procedure executes when you specify the /TRACE_LOG option on the command line.
- DEBUG_LOG, a new function, writes a string to the log file if you specify the /TRACE_LOG option. The DEBUG_LOG syntax is:

  ```
  DEBUG_LOG(char_exp);
  ```

  Char_exp represents the string you want to write to the source file.
- Arguments (actual parameters) can be passed to an invoked external program link (thus eliminating the need to create a parameter packet in such a case).

Descriptions of the new logical name-related functions are found in the *VAX RALLY ADL User's Guide.*

## 1.6 Rdb/VMS Specific Changes

Changes specific to Rdb/VMS include: parameterized restrictions for DSDs, logical OR in record selection expressions, and extended QBE.

### 1.6.1 Parameterized Restrictions for DSDs

You can now specify a restriction in the record selection expression of a data source definition (DSD) that includes a parameter that is not resolved until run time. For example, uses include:

- Selecting records corresponding to unpaid invoices that were issued more than two months before the current date.
- Selecting records where the USERNAME field is equal to the USERNAME of the current user.
- Subsetting a list of values to show only values that match a pattern string entered by the user; such as RO*GERS to find both ROGERS and RODGERS.

- Selecting records where the values of the key fields match the values of the corresponding fields of another form/report. This can be used to implement applications where the user starts out in a directory report, moves the cursor to the item of interest, then presses SELECT to go to a detail form that describes the item.

Using the feature involves two steps:

1. Adding a restriction to the record select expression in the DSD. In the right-most field where you would normally put a field name or a value, put a name without a dot in it.

2. Using option 7 (parameters) on the *Edit a Data Group* menu when editing the form/report group. Specify the application object from which the actual value for the parameter will be taken at run time.

Parameterized restrictions are only supported for use with Rdb/VMS.

### 1.6.2 Logical OR in Record Selection Expression

You can specify logical ORs as part of the restrictions in a record selection expression for an Rdb-based DSD. You can construct complex Boolean expressions by combining the use of the already existing AND feature with the new OR feature.

There is no way to give special precedence, such as you might attempt with the use of parentheses: AND will always have a higher precedence than OR.

### 1.6.3 Extended QBE for Rdb/VMS

Extended query is a new feature for use with form/report fields based on Rdb/VMS data sources. To extend a query, move your cursor to the field on which you wish to perform the extended query; press FIND (invoking the new 'extend query' command); type the query expression at the prompt (for example, MATCHES foo*bar); press RETURN to move to another field or press ENTER to perform the query.

Extended query support in V2.0 has not been provided for access methods other than Rdb/VMS.

## 1.7 Command and Key Definition Changes

There have been a number of changes to the set of commands that you use to run and define RALLY applications. Some commands have been changed to accept parameters and qualifiers, new commands have been added, names of existing commands have been changed, and key definitions have changed. There are also fewer key definition files.

### 1.7.1 Command Parameters and Qualifiers

A number of commands have been modified so that they can accept parameters and qualifiers on the command line. For example:

- 'set highlight=(bold,underline)'
- 'macro number 7'
- 'write report disk:[dir]name'
- 'write report disk:[dir]name /nohighlighting'

### 1.7.2 New Commands

VAX RALLY V2.0 provides the following new commands:

- 'write' commands

  Five new commands have been added for writing out text to files. The new commands parallel the 'print' commands:

  - 'write screen'
  - 'write report'
  - 'write rest_of_report'
  - 'write page'
  - 'write menu'

  You can use a qualifier, /nohighlighting, with these commands. These commands then ignore highlighting information and convert any line drawings to dashes, bars, and plus signs. These commands also accept an optional file specification parameter (for example, 'write screen july_sales.dat').

- 'delete previous word'

The 'delete previous word' command has been added to the text editing commands. By default, this key is mapped to the LINEFEED key (F13 on LK keyboards).

- 'extend query'

  The 'extend query' command is used for invoking the extended query feature, which is a new feature for V2.0.

- 'fork report_utility'

  The 'fork report_utility' command has been added to the commands in the key definition source file. This command brings up the Report Utility while you are working in the Definition System.

- 'adl lov'

  The 'adl lov' command has been added to the commands in the key definition source file. This command brings up the list of values (LOV) in the ADL editor.

- 'coordinates'

  The 'coordinates' command has been added to the commands in the key definition source file. This command brings up the coordinates form from within the form/report screen editor.

### 1.7.3 CALL_CMD ('ignore') Cancels Previous Function Key

A new use has been added for the ADL built-in CALL_CMD. CALL_CMD ('ignore') can now be used in ADL procedures to ignore an unprocessed form/report command. See the *VAX RALLY ADL User's Guide* for additional information on CALL_CMD and the *VAX RALLY Command Reference Manual* for information on the 'ignore' command.

### 1.7.4 Changes to Command Names

The names of several RALLY commands have been changed. Changes range from minor spelling alterations to completely new terms. Some changes were made to reflect the function more clearly, some to eliminate unnecessary words (for example, "use" in many command names), and others to avoid conflicts with new commands. The following table lists the old command names and their new equivalents.

**Note**

Most of the V1.1 command names are recognized by RALLY V2.0. However, you are strongly encouraged to use the V2.0 names, because support for the remaining V1.1 names might be removed in a future release of RALLY if required, in order to support future names that might be added to the product.

**Table 1–1: New Command Names**

| V1.1 Command | V2.0 Command |
|---|---|
| delete previous_character | delete previous character |
| read one_macro | read named_macro |
| use command | application_command |
| use cursor_roam | first choice |
| use file_macro | macro name |
| use function_key_choice | function_key_choice |
| use local_function | local_function |
| use macro | macro number |
| use prompt_line | response area |
| use roam last | last choice |
| write one_macro | write named_macro |
| write macros | write macros |

### 1.7.5 Changes to Default Key Definitions

In RALLY, keys can have different meanings in different situations. V2.0 takes greater advantage of this capability than V1.1. For example:

- The SELECT key now means 'local_function' in most circumstances, but in a list of values it still means 'select value', just as it did in V1.1.
- NEXT SCREEN now means:
  - 'next screen' when viewing help, so you can use it to flip through long help messages

  - 'next scroll_area' when you are in a scrolling area, so you can use it to navigate within lists of values
  - 'next page' in most other situations

- The FIND key now means 'set query' when the user is in update mode and 'extend query' (a new command for V2.0) when the user is in query mode. You now exit from query mode (thereby executing a query) by pressing the ENTER key.

- GOLD-Q means 'previous group' when the user is in a list of values, so that you can now use it to exit from a list of values without selecting a value. However, it still means 'quit action' in most other situations.

- The key mappings for performing queries and using lists of values have been changed. In query mode, the ENTER key is used to execute the query; the FIND key invokes the new extended query feature. In a list of values (LOV), the ENTER key selects the value that the cursor is resting on. To leave an LOV without selecting a value, use GOLD-Q, GOLD-K, or F8, all of which are mapped to the 'previous group' command in the context of an LOV.

It is suggested that you look at the Definition System legends or online help information if you have any doubt as to which key sequences map to which commands.

The assignment of different mappings depending on context was accomplished by adding new key areas to the key definition files. However, V1.1 key definitions will work with V2.0. If you have modified your key definitions and would like to upgrade to the V2.0 key definitions, then you should look at RALLY$WPS.RGC and RALLY$EDT.RGC, both of which are in SYS$EXAMPLES. In V1.1, we required that the key areas appear in the same fixed sequence. In V2.0, the key areas are named and can appear in any sequence.

The changed key definitions are listed in Table 1-2.

**Table 1-2: New Key Definitions**

| Key | Command |
|---|---|
| **General** | |
| LINEFEED | 'delete previous word' |
| **Help** | |
| NEXT SCREEN | 'next page' |
| PREV SCREEN | 'previous page' |
| GOLD-NEXT SCREEN | 'last page' |
| GOLD-PREV SCREEN | 'first page' |
| **ADL Screen Editor** | |
| GOLD-SELECT, GOLD-KP. | 'adl lov' |
| **Form/Report Screen Editor** | |
| GOLD-A | 'coordinates' |

**Table 1-2: New Key Definitions (Cont.)**

| Key | Command |
|---|---|
| **Form/Report Keys** | |
| *Update, Insert and Read Only Modes* | |
| FIND | 'set query' |
| SELECT, KP. | 'local_function' |
| GOLD-SELECT, GOLD-KP. | 'list_of_values' |
| *Query Mode* | |
| FIND | 'extend query' |
| ENTER, GOLD-F, F10, CTRL-Z | 'execute query' |
| *Scrolling Regions (all modes)* | |
| NEXT SCREEN | 'next scroll_area' |
| PREV SCREEN | 'prev scroll_area' |
| GOLD-NEXT SCREEN | 'last scroll_area' |
| GOLD-PREV SCREEN | 'first scroll_area' |
| *List of Values* | |
| ENTER, GOLD-F, F10, SELECT, CTRL-Z, KP. | 'select value' |
| GOLD-Q, GOLD-K, F8 | 'previous group' |

When you use RALLY you will get the WPS version of the new key definitions by default. If you would like to use EDT-compatible key definitions, use the qualifier /KEY_DEFINITIONS=RALLY$EDT or define the RALLY$KEY_DEF logical name to be RALLY$EDT.

The V1.1 key mappings are still available. If you would like to use V1.1 key definitions, use the qualifier /KEY_DEFINITIONS=RALLY_EDT or /KEY_DEFINITIONS=RALLY_WPS, or define the RALLY$KEY_DEF logical to be RALLY_EDT or RALLY_WPS. (Note the underscore in the equivalence name instead of the dollar sign.) The V1.1 key definition files are included with RALLY V2.0, but might be removed in a later release. If you have defined the logical name RALLY$KEY_DEF, you must change that definition to point to the new key definitions, or you will continue using the V1.1 definitions.

### 1.7.6 Fewer Key Definition Files

Only V1.1 versions of the _132 and _VAXSTATION key definition files (for example, RALLY_WPS_132) are included with RALLY V2.0. These special key definition files are not needed for V2.0. When RALLY is initialized, it queries VMS to find out the size of your terminal, instead of getting this information from the key definition file. To use the RALLY Definition System in 132 column mode, use the DCL command "SET TERMINAL /WIDTH=132" before invoking RALLY. This will give you a wider list of values area and is useful for editing 132-column printed reports.

Applications which you create will continue to use the default window size (24 by 80) unless you change the application's task(s). To do so, select 3 1 2 (application task edit) and edit the window size.

### 1.7.7 Use Command Names in Key Definition Files

Previous releases of RALLY required the use of ADL aliases for commands in key definition source files. Command names can now be used in key definition files. This is particularly convenient for defining keys for macros, application commands, menu function keys, and so on. Previously, the rules for defining keys for these commands were irregular. Now, you simply use the same text that you would use on the command line; for example, "macro number 9," "application command 3," "function_key_choice 7".

The example key definition source files included in the kit use command names, rather than ADL aliases, to bind key sequences to commands. Included in these files are key bindings for three new commands that are only supported by the Definition System:

- 'fork report_utility'

  Brings up the Report Utility from the Definition System.

- 'adl lov'

  Brings up LOV in ADL editor.

- 'coordinates'

  Brings up the coordinates form from within the form/report screen editor.

If you have developed your own key definition files, and use them with the Definition System, then you should add these commands to these files.

The syntax used in key definition files is otherwise unchanged. Use of aliases is still supported.

### 1.7.8 Key Definition Source Code Change

The example key definition source code files provided with the Definition System have been changed to use application mode cursor keys and 8-bit control codes. These changes were made to improve compatibility with VAX DEC/Test Manager (DTM). The change to 8-bit control codes will also improve performance when RALLY is used with newer terminals, such as VT200 and VT300 series terminals and VAXstations.

You do not need to upgrade existing key definition files or key definition source code files to follow these new conventions. The Define Keys Utility and the Run-Time System are both capable of interpreting both the old and new control sequences. Recompilation of key definition source code is required to receive the benefit of the performance improvement previously mentioned.

## 1.8 Form/Report Enhancements

RALLY V2.0 improvements include many new form/report features. Some of these features include: new class of reports, running aggregates stored in data sources, printing of subforms, and lock timing option, among others.

### 1.8.1 Sequential Sibling Reports Supported

Say that you want to create a report that presents two lists of records for each parent record. For example, say that the parent group is projects and you want to list all of the personnel (people) assigned to each project and all of the equipment assigned to each project. You would like these lists formatted so that all of the people are listed first, and then all of the equipment is listed. In V2.0, this report can be created by following these instructions:

1. Put the second child group (equipment) in a format group.
2. Set the format group to appear only on last pages.

3. Set the format group to be displayed below the first sibling group (personnel).

The resulting group structure is:

```
MAIN    Main Group
DATA        PROJECTS_DG
DATA            PERSONNEL_DG
FORMAT          EQUIPMENT_FG
DATA                EQUIPMENT_DG
```

### 1.8.2 'Next_group' Now Based on Visitation Order

The RALLY commands 'insert record next_group', 'next group', 'previous group', 'last group', and 'first group' in V2.0 determine the order of sibling groups on a different basis from that used in V1.1. In V1.1, sibling groups were visited in the order in which the groups appeared on the output order of their parent group. In V2.0, the visitation order (formerly called "input order") determines the order in which sibling groups are visited.

### 1.8.3 Running Aggregates and Computed Fields Can Be Stored in Data Source

In V2.0, running aggregates can be stored in a data source. This was not allowed in V1.1. Also, computed fields that use a field from a parent group as an input to their computation expression can now be stored in a data source.

### 1.8.4 User Can Leave LOV When Cursor Enters LOV Automatically

When the cursor goes directly to a list of values, the user is now allowed to exit from the list of values without selecting a value by using the 'previous group' command. Any value the user then types is validated against the list of values, as if the "validate against list of values" feature were in effect.

### 1.8.5 Display of Fields in Form/Reports Not at the Top of the Action Stack

In RALLY V1.1, when an ADL procedure changed the value of a field in a form/report, the change was not displayed until the user returned to that form/report. For example, if a form/report called a menu that called an ADL procedure that changed the field's value, the change was not visible until the menu was exited.

In V2.0, such changes can be displayed even in a form/report that is not at the top of the action stack. In the previous example, the new value of the field can be displayed while the menu is still active, providing that the menu does not obscure the field on the screen.

### 1.8.6 Printing of Subforms

When a form/report that has subforms is printed, only the main part of the form/report is printed; the subforms are not printed. In previous versions, each subform was printed on a separate page.

### 1.8.7 Lock Timing Option

In RALLY V1.1, under SHARED WRITE, locks are acquired by RALLY the first time that a user makes a change to a record. This introduces the possibility that locks might be held for several seconds, or even longer, while the user thinks about and makes several changes to the record. This is often the desirable behavior, because it ensures that once a record is touched by a user, no other user will make changes to the same record.

RALLY V2.0 preserves this behavior as an option, but it further adds the option to "Lock records at UPDATE". If this option is selected, RALLY defers acquisition of locks until after the user has entered all changes for that record. (If in the meantime some other user has changed the record, an error message will be displayed.) This eliminates the possibility of holding locks across user think time, thus improving multi-user performance.

### 1.8.8 Legend for No Records

Form/reports can display a legend when the cursor is not in a field, for example, when a form/report is in UPDATE mode and there are no records displayed.

The default for the "Legend to display when cursor is not in a field" on the *Attributes of a Form/Report Packet* form, is 30,000. This default is applied when you create a form/report packet. The run-time help file provides a legend with this number. This legend explains how to insert records when there are none displayed. If you want to use this legend in existing applications, edit your form/report packets to specify 30,000 as the legend to display when the cursor is not in a field.

### 1.8.9 Packet Option to Keep User in Insert Mode

"User may not leave insert mode" on the *Attributes of a Form/Report Packet* form, is the default when you create a new form/report packet. This option disables the 'set query' and 'set update' commands, preventing the user from reading or modifying existing data. This option has no effect on packets with initial modes other than insert.

### 1.8.10 Command 'insert record after' Can Be Used to Insert First Record

With all previous releases of VAX RALLY, the 'insert record next_group' command provided the only means for inserting the first record into a group that did not contain any records, even with a top-level data group. In V2.0, the 'insert record after' command (INSERT HERE) can be used in this situation to insert the first record.

### 1.8.11 New Initial Modes for Form/Report Packets

V2.0 includes the following new initial modes for form/report packets:

- READ ONLY

  This initial usage mode allows users to display data from the data source, but not to insert, delete, or modify data. Users can query the data source.

- PRINT REPORT

  In this mode, RALLY creates the form/report and sends it to a printer, not to the screen or to a disk file. By default, the form/report goes to the system printer.

- WRITE REPORT

  In this mode, RALLY creates the form/report and sends it to a file on disk, not to the screen or to a printer. By default, the file name is the form/report name, and the default file type is .LIS.

### 1.8.12 Form/Report Joins and Control Breaks Can Be Combined in the Same Form/Report

In V1.1, a single form/report could use form/report joins or control breaks, but not both. In V2.0, these can be combined within the same form/report. For example, in the RALLY$TAPES application, a form/report could join PEOPLE and TAPES, and display TAPES by categories and artists.

### 1.8.13 SET_FAILURE() in Action Sites

If an ADL procedure called at an after-visiting action site calls the ADL built-in procedure SET_FAILURE, the user's cursor is returned to the current field.

If an ADL procedure called at an after-form/report action site calls the ADL built-in procedure SET_FAILURE, the form/report is not exited.

### 1.8.14 Additional Form/Report Changes

Additional changes include:

- A new option has been added for form/report groups to have the 'next record' and 'previous record' commands maintain the current field. This means that in a form/report which repeats records down the screen, pressing the Down Arrow key will result in vertical motion only, eliminating the horizontal motion to the beginning of the next record.
- An option has been added to reread a list of values each time it is used.
- Field justification has been changed so that it is now an attribute of the form/report field. In V1.1, it was an attribute of number character set and date format objects.

## 1.9 Miscellaneous Functionality Changes

This section describes a number of miscellaneous functionality changes. These changes include:

- A number of action sites have been added for use with menus.
- Legends are now available for use with menu choices and the menu response area, as well as form/report fields. When you associate a legend with a menu choice and the user moves the cursor to that choice, the legend is displayed.
- The Definition System has been changed to use the Rdb convention for describing the scale factor to be applied to fields. The old convention is now used only in ADL.

### 1.9.1 RALLY EDIT /TRACE

V2.0 allows use of the /TRACE qualifier with the RALLY EDIT command. The /TRACE qualifier causes RALLY to generate a trace log file when you run your application within the Definition System.

### 1.9.2 Changed Behavior of RETURN_TO Call Type

The behavior of the RETURN_TO call type has been changed to be consistent with the behavior of the 'first menu' command. (The 'first menu' command does the same thing as typing "top" on a menu.) In V1.1, these differed with respect to intermediate ADL procedures on the stack: such ADL procedures were exited in mid-stream for RETURN_TO, but run to completion for 'first menu'. In V2.0, intermediate ADL procedures are run to completion for RETURN_TO as well as for 'first menu'.

### 1.9.3 Dynamic Activation of RALLYEXT

The implementation of the buffer-method has been changed so that the RALLYEXT image is dynamically activated at run time. Because this image is now only activated with customer-created images created for use of buffer-method external program links, there is no longer any need for a stub RALLYEXT image in the kit. This will have a positive (although small) impact on application startup time for most applications, since the image activation overhead for RALLYEXT will be paid only when a call is made to a buffer method external link.

These changes will be transparent to V1.1 users. If a V1.1 user is using external links and has already performed one of the following then the external link should continue to work without change:

1. Installed the image
2. Placed the image in SYS$SHARE under the name RALLYEXT, or
3. Defined the logical RALLYEXT to point to the image

If the user is not currently using BUFFER method external links, then the shareable image will never be activated.

### 1.9.4 Verification of Individual Objects

Verification of individual objects is available and a message has been added to give users feedback that the verify was indeed successful. (Verify is choice 6 under Operations on the *Edit Application* menu.)

### 1.9.5 List of Values Available for Messages

In V2.0, RALLY provides a list of values for the messages in an application. The LOV is displayed when you edit a message. The message type (help, error, or legend) is displayed in the LOV, as well as the number.

### 1.9.6 Message Numbering Guidelines

In V2.0, the number range 1 to 10,000 is reserved for use by RALLY applications. No RALLY Run-Time System messages will use numbers in this range. It is recommended that you only use numbers in this range. You will be given a warning message if you use messages outside of this range; however you will be allowed to use any number up to 32,767.

It is also recommended that you use unique message numbers for all messages, regardless of their type. In many cases, messages with the same numbers will not interfere with each other, but re-use of numbers does open up possibilities for future conflicts.

To help you avoid unintentionally re-using message numbers, the Definition System will now give you a warning message when you create a message with a number that already appears somewhere else in the help or error chain. A future release might provide a similar warning when messages are loaded using the Load/Unload Message Utility.

### 1.9.7 Character Sets

The ability to create and edit character set definitions was removed from the Definition System. As implemented in past versions, character set definitions have little or no impact on applications. The only change that you could make to the character set that had any effect was to change the uppercase value for a character. This change was made to facilitate closer integration with the VMS National Character Set Utility in future releases of RALLY. In all cases, applications will continue to behave as they did under V1.1.

### 1.9.8 Only One Character Required for Selecting Menu Options

In previous releases of RALLY, you were required to enter two or more characters on keyword menus, even when a single character would have been unambiguous. In V2.0, a single character is sufficient in these instances.

### 1.9.9 ADL Editor Now Scrolls Instead of Page Flipping

In previous releases of RALLY, all editors moved forward or backward a screen at a time whenever the Up Arrow and Down Arrow keys were used to leave the area currently displayed. This behavior has been changed in the message viewer and the ADL editor. In V2.0, these editors will scroll one line when you use the Up Arrow and Down Arrow keys to move your cursor outside of the area currently displayed. The behavior of the menu, form/report, and message editors remains unchanged.

### 1.9.10 Process Quota Check

Each time that you use RALLY V2.0, RALLY checks the quotas associated with your process. If any of your process quotas are below the values specified in the *VAX RALLY Installation Guide* then a warning error message will be displayed for each quota that is low. The message will give the current value and the minimum value and include a request to notify the system manager. The user must acknowledge the message but will not be prevented from proceeding. This check is made both for running and editing applications.

## 1.10 CDD/Plus Integration

RALLY V2.0 integrates with CDD/Plus. The RALLY Definition System shares two kinds of information with CDD/Plus: information about your data sources and information about your application objects.

### 1.10.1 Data Sources

With all supported data sources (except FDX), RALLY can read definitions of your data sources from the CDD/Plus data dictionary. With RMS and DATATRIEVE, you must specify your data source by supplying the path name of a CDD/Plus definition of the data source. Although DIGITAL recommends that you always integrate your databases with CDD/Plus, dictionary integration is optional for Rdb/VMS databases. You can specify an Rdb/VMS database by providing its file name or its CDD/Plus path name.

When you edit your DSDs or verify your application, RALLY will notify you if the definition of your database has changed (for example, if the data type of a field is changed from numeric to text). When such changes occur, you can use the Update Utility to change your application to respond to the changes.

You can use RALLY to define and modify the structure of Rdb/VMS databases. If you elect to have your databases integrated with the dictionary, RALLY will create and maintain dictionary definitions for your databases.

### 1.10.2 Application Objects

RALLY has the ability to write entries into the CDD/Plus dictionary that represent your application and its DSDs. RALLY also creates relationships between these entries and the dictionary entries for the data sources that are referenced by your application. This makes your RALLY application show up when the interactive dictionary operator utility (CDO) is used for impact analysis.

If you are editing an application file in place, you can create and update the dictionary entries for your application by using the /DICTIONARY qualifier with the RALLY CREATE and RALLY EDIT commands. You can also use the Integrate Utility to create or update dictionary entries for your application. Use the Integrate Utility if you edit a local copy of an application file, and you want to delay updating the dictionary entries for the application until you update the master copy of the application (for example, by copying the local copy of the application file into a master directory).

### 1.10.3 CDD/Plus Proxies For Application Objects

RALLY provides the option of writing proxy objects into the dictionary that represent your RALLY application and its DSDs. If your data sources have been integrated with CDD/Plus, RALLY will also record in the dictionary relationships between your DSDs and the data sources that they reference. These relationships make it possible to use the CDO utility to see which RALLY applications reference a given database.

### 1.10.4 CDD/Plus V4.0

For the Base system only, the installation procedure now requires that CDD/Plus V4.0 is present on the system on which RALLY is being installed.

## 1.11 Start Up Command Procedure

System managers should note that the name of the RALLY startup file has been changed from RALLY_STARTUP to RALLY$STARTUP.

In V2.0, we have changed the saveset mnemonics for the VAX RALLY installation kits:

1. From RALLY to RALBO, for the VAX RALLY base option. This kit includes both the Definition System and the Run-Time System.

2. From RLYED to RALTBI, for the Text Based Instruction installation kit. This kit is provided with the base option kit at no additional cost.

The saveset mnemonic for the VAX RALLY Run-Time Option is RALRTO.

## 1.12 Changes to RALLY System File Names

Many files in the RALLY Definition System and Run-Time System have been renamed to start with RALLY$, to be consistent with a naming scheme being adopted by many products. For example, the sample application that accompanies the *VAX RALLY Definition System User's Guide* is named RALLY$COMMERCE.RGA (not RALLY_COMMERCE.RGA).

The *VAX RALLY Installation Guide* contains a list of the system files.

## 1.13 New and Revised Documentation

The following documentation is supplied with VAX RALLY V2.0:

- *Introduction to VAX RALLY*—revised version of the current manual.
- *VAX RALLY Definition System User's Guide*—revised manual; title changed to reflect the terminology change ("Definition System" for V2.0, replacing "Dialog" for V1.1). Refers often to the sample application RALLY$COMMERCE.RGA.
- *VAX RALLY Reference Manual and Master Index*—new reference manual.
- *VAX RALLY Guide to Application Development*—new manual; includes a suggested methodology for RALLY application development. Refers often to the new sample application RALLY$SALES_INFO.
- *VAX RALLY Command Reference Manual*—revised version of the current manual.
- *VAX RALLY ADL User's Guide*—revised version of the current manual.
- *VAX RALLY Application User's Guide*—revised version of the current manual.
- *VAX RALLY Installation Guide*—revised for V2.0.
- *VAX RALLY Release Notes, Version 2.0*
- Poster showing the Definition System menu structure

The V1.1 *VAX RALLY Dialog Reference Manual* has been dropped, because it merely provided information contained in the RALLY Definition System help file. This information is still available through the use of the 'help' command. The new *VAX RALLY Reference Manual and Master Index* provides better organized and more meaningful reference information.

### 1.13.1 Online Version of *VAX RALLY Application User's Guide*

The *VAX RALLY Application User's Guide* is supplied as a printed manual. In addition, an online version of the manual is provided in the VAX DOCUMENT source file SYS$EXAMPLES:RALLY$APPLICATION_USERS_GUIDE.SDML, to assist you in providing customized documentation for run-time users of RALLY applications.

See the comments at the beginning of that file for information on processing the file and for important copyright information.

### 1.13.2 Sample Applications

Several sample applications are provided in the SYS$EXAMPLES directory. The following are of special interest:

- RALLY$COMMERCE.RGA illustrates many features described in tutorial sections of the *VAX RALLY Definition System User's Guide*. It is essentially the same as the V1.1 application named RALLY_COMMERCE_ADVANCED.RGA. (The V1.1 application named RALLY_COMMERCE.RGA has been dropped, as RALLY$COMMERCE.RGA contains the features of both V1.1 applications.)

  This application uses the database files RALLY$COMMERCE.RDB and RALLY$COMMERCE.SNP.

- RALLY$SALES_INFO.RGA accompanies the new manual *VAX RALLY Guide to Application Development*. It is a new application for V2.0.

  This application uses the database files RALLY$SALES_INFO.RDB and RALLY$SALES_INFO.SNP.

# Problems Fixed in Version 2.0 2

RALLY Version 2.0 (V2.0) fixes a number of problems that appeared in RALLY Version 1.1 (V1.1). This chapter describes the most significant problems and the steps taken to address these problems.

## 2.1 Database Names Normalized Before Comparisons

In previous releases of RALLY, the Rdb/VMS access method compared the names of databases as entered by the definer when determining how to share attaches between DSDs. The result was that RALLY would sometimes treat relations in the same database as if they were in separate databases. This introduced seemingly random performance penalties (because of extra attaches) and opportunities for introduction of data inconsistencies (because Rdb/VMS guarantees consistency within a database attach, but not across multiple simultaneous attaches).

## 2.2 Subforms Display Correct Data If Multiple Records Per Page

In V1.1, subforms sometimes displayed incorrect data if more than one record was displayed per page. This problem does not occur with V2.0.

## 2.3 LOV Lookup Changes

In V1.1, LOV lookup (Copy other fields from list of values) had side effects that limited its usefulness. To rectify this problem, the way LOV lookup translation works has been changed for V2.0.

In V2.0, LOV lookup will change the value of variable fields, but not data fields, when data is read from the data source in UPDATE mode. If a data field uses the option to copy other fields from the list of values, the data field will be looked up in the list of values, with the result of the lookup filling in any "other fields" that are variable fields. If a variable field uses the option to copy other fields from the list of values and the "other fields" are data fields, then the "other fields" will be looked up in the list of values, with the result of the lookup filling in the original variable field.

In V1.1, data fields could be overwritten as a side effect of being read from the data source. This does not happen in V2.0.

These changes allow you to use an LOV to do lookups both when entering data and when displaying data read from a data source.

## 2.4 Menus No Longer Disappear When Partially Occluded

In all previous releases of RALLY, partially occluded menus were blanked out. This made it difficult to implement nested menus, such as the two-column menus used in the RALLY Definition System. This problem has been fixed in V2.0.

## 2.5 Empty LOV Does Not Result in Form/Report Termination

In V1.1, when a required field was validated against a list of values, but the list of values was empty, the form/report was terminated. In V2.0, the form/report is not terminated if the field is in a data group. This avoids the possibility of accidentally rolling back previous records entered by the user.

## 2.6 Trailing Blanks No Longer Trimmed for Varying Text Fields

In previous releases of RALLY, the values of fields that were defined as VARYING TEXT in an Rdb/VMS database were always trimmed of trailing blanks before writing them to the database. In V2.0, blanks are trimmed only if the delete trailing blanks input option has been selected for the corresponding form/report field.

## 2.7 Compiler Warning in Shareable Images

V1.1 could not be used with external program links to shareable images that included compiler warning. VAX RALLY will now trap the compilation warning signal from LIB$FIND_IMAGE_SYMBOL. If the /TRACE switch was used, the warning message will be written to the trace log and execution of the application will continue. If the /TRACE switch was not used, the warning signal will be ignored.

## 2.8 Problems with Handling of Signals from External Links

In V1.1, if an external program routine signaled SS$_DEBUG, a stack dump occurred. This problem occurred because VAX RALLY is linked /NODEBUG/NOTRACEBACK. As a result of receiving a SS$_DEBUG signal, an attempt was made to pass control to the debugger and the debugger was not there to catch the attempt. VAX RALLY will now attempt to intercept the SS$_DEBUG signal. If the debugger is not active, VAX RALLY will display an error message indicating the signal was received and attempt to return to the external routine which signaled. If the debugger is active, it will receive the signal and the user will be placed in the debugger in the routine which signaled.

## 2.9 Problem with Large Temporary Files

In all previous releases of RALLY, a bug check resulted whenever a temporary file grew larger than 9000 blocks. These temporary files were produced only for long form/reports. This problem has been fixed in V2.0.

## 2.10 GET_CMD Now Returns 'Finish Action'

In V1.1, the ADL built-in function GET_CMD returned incorrect values for the commands 'finish action' (exitaction), 'finish task' (exittask), 'finish application' (exitappl), 'quit action' (abortaction), 'quit task' (aborttask), and 'quit application' (abortappl). In V2.0, GET_CMD returns the correct value. ADL procedures can detect which command was used by testing for the corresponding command alias.

## 2.11 'Quit Application' Now Works As Documented

'Quit application' did not work properly when you are in an infinite loop of error messages, either because of definer error or RALLY development error. Up until now, the only way out of such an infinite loop of error messages was CTRL/C or CTRL/Y. Now, 'quit application' gets you out of RALLY.

## 2.12 Changed Behavior of Before/After-Commit/Rollback

The following action sites are now invoked less often than in V1.1: before-commit, before-rollback, after-commit, after-rollback. In V2.0, these action sites are invoked, as their names suggest, before and after commits and rollbacks. In V1.1, these action sites, especially before- and after-commit, were often invoked when there was actually no commit or rollback.

For example, in V2.0, the before-commit action site is not invoked in a form/report having no data groups. In V1.1, the before-commit action site could have been used to perform cross-field validation on variable fields in the main group. To perform such validation in V2.0, use the after-form/report action site.

## 2.13 "After Value Change" Action Site

In V2.0 the after-value change action site is invoked every time the user, an ADL procedure, or an external program sets a form/report field's value. In V1.1, the after-value change action site was not invoked the first time an ADL procedure or external program set a field's value.

However, in V2.0 as in V1.1, the after-value change action site is not called the first time a computed field is computed.

## 2.14 Improved Handling of Failed Queries

Failed queries are now treated as errors more consistently than in V1.1. A failed query is one that finds no records. In V1.0, this was not treated as an error; instead, the form/report was displayed with no records, usually resulting in a mostly blank screen. In V1.1, failed queries were treated as errors, but not consistently. If a form/report had a list of values, the list of values was included in the criteria for deciding whether a query failed. Therefore, in the usual case where a list of records had one or more records, all queries were deemed successful and were displayed as in V1.0. In V2.0, lists of values are ignored in considering whether a query succeeds or fails.

Many applications built in V1.0 need to suppress the error for failed queries, because these applications use queries performed by ADL procedures to restrict the records that are displayed on a form/report. In V1.1, some of these applications have been modified to include lists of values, on the basis that V1.1 would consider all queries on such a form/report to be successful. In V2.0, two approaches are available:

1. Modify the application to use run-time variables in its record selection expressions, rather than queries performed by ADL procedures.

2. Add a data group to the form/report that will always have at least one record.

## 2.15 Autocommit and "Commit on Query"

The determination of whether or not a commit happens when a user does a query has changed from V1.1. In V2.0, this is controlled entirely by the "commit on query" option in the form/report packet. In V1.1, if a group or its underlying data source definition used the autocommit option (that is, commit when leaving record), a commit would happen even if the "commit on query" option were not selected.

## 2.16 Spawn Disabled for Captive Accounts

RALLY V2.0 does not allow the 'spawn' and 'interrupt' commands to be used to spawn DCL subprocess from captive user accounts. A captive user account is an account created with the AUTHORIZE qualifier /FLAGS=(CAPTIVE).

# Problems, Restrictions, and Other Notes 3

This chapter describes known problems and restrictions with RALLY Version 2.0 (V2.0) and suggests ways of working around these problems. This chapter also includes notes regarding RALLY, for example, upgrading V1.1 applications.

## 3.1 Upgrading V1.1 Applications

This section is intended for customers who developed applications under RALLY V1.1 and will be running or modifying them using RALLY V2.0.

### 3.1.1 Use RALLY EDIT to Upgrade Application Files

For each RALLY V1.1 or A-to-Z AG V1.1 application that you wish to modify in any way (including compacting), you must use the DCL command RALLY EDIT specifying the application file *before* you use the RALLY Compact Utility or any other RALLY utility that can modify that application file.

You will not be allowed to use the /NONEW qualifier when upgrading application files, so V2.0 will always create a new copy of your application file when upgrading it and will not modify your V1.1 application files. However, you should make backup copies of all V1.1 application files that you intend to use with V2.0 before you use RALLY V2.0 with the files.

When you use the V2.0 Run-Time System with a V1.1 application, RALLY will upgrade the application in memory as it executes, and the application will not be modified.

### 3.1.2 Upgrade Required for Some Help Messages

In previous versions of RALLY, the message screen editor erroneously marked both help and error messages as error messages. Because the type designation for a message had very little significance in V1.1, most people were not aware of this problem. V2.0 makes greater use of the type of a message, so DIGITAL recommends that you correct the type of the affected messages using the procedure described below. This problem will not impact the execution of applications and will cause only minor annoyances when editing, for example:

- The affected help messages will be designated as error messages in the list of values that is displayed when you copy, edit, or delete a message.
- You will not be able to edit the affected help messages by using the 'local_function' command when using the *Edit a Form/Report Packet* form.
- The UNLOAD MESSAGES /TYPE=HELP command will not unload the impacted help messages but the UNLOAD MESSAGES /TYPE=ERROR will unload the messages.

If you created your messages using the message screen editor and keep these messages in the same application file as your other application objects (for example, forms and menus), use the following procedure to correct this problem:

- Use the DCL command RALLY UNLOAD MESSAGES /TYPE=ERROR to extract the messages from your application file.
- Use a text editor to edit the message source text file created by the previous command and change the type designation for the help messages from $e$ to $h$.
- Use the DCL command RALLY EDIT to edit the application file and delete all of the help messages.
- Use the RALLY LOAD MESSAGES /MERGE command to reload the messages back into the application file.

If you created your messages using the message screen editor but keep them in a single application file separate from the application file that contains your other objects (for example, forms and menus), use the following procedure to correct this problem:

- Use the command RALLY UNLOAD MESSAGES /TYPE=ERROR to extract the messages from your application file.
- Use a text editor to edit the message source text file created by the above command and change the type designation for the help messages from $e$ to $h$.

- Use the RALLY LOAD MESSAGES command to reload the messages. This command will create a new version of the application file that contains the old error messages and the corrected help messages.

If you created your messages using the message screen editor but keep them in two different application files, one for help messages and one for error messages, use the following procedure to correct this problem:

- Unload the messages from the help message file using the DCL command RALLY UNLOAD MESSAGES /TYPE=ALL.
- Use a text editor to edit the message source text file created by the previous command and change the type designation for the help messages from $e$ to $h$.
- Use the RALLY LOAD MESSAGES command to reload the messages. This command will create a new version of the application file that contains only the corrected help messages.

If you created your messages using the Load/Unload Message Utility, you will not be affected by this problem. However, because of a minor problem with the way that the V1.1 Load Messages Utility stored messages, you will not be able to use the V2.0 Definition System to copy such messages until they have been reloaded using the Load/Unload Message Utility. This problem will not impact most users.

### 3.1.3 Several Objects Now Hidden

In V1.1, you were required to manipulate keys, foreign key links, view DSDs, and break-up DSDs. In V2.0, these objects are hidden from your view, and are maintained solely by RALLY.

You can now consider the information that was previously associated with these objects to be stored with the data groups in your form/reports. Thus, changes you make to the control breaks or foreign keys for a form/report will only affect the form/report that you modify, even if you shared the underlying hidden objects with other form/reports when you created the application under V1.1.

### 3.1.4 Primary Key Functionality No Longer Supported

In V1.1, key objects had an option called a "primary key." Changes to the on-disk format used to store RALLY applications made it prohibitively difficult to continue this feature under V2.0. V1.1 applications that took advantage of this option will continue to run as they did under V1.1, until they are edited under V2.0. After that, the option will be reset. To implement similar functionality under V2.0, define an index with duplicates not allowed or use the DB_QUERY ADL built-in procedure to verify that there are no records in the data source with the same field values.

### 3.1.5 DB_RELATED_GROUPS No Longer Supported

V1.1 applications that use the ADL built-in function DB_RELATED_GROUPS will continue to work under V2.0. However, because view DSDs are now hidden from your view, you cannot use this feature with newly created applications. Use a while loop and DB_QUERY instead.

### 3.1.6 Character Set Definitions No Longer Supported

In V1.1, you were given the opportunity to define alternate character set definitions. These character sets were used only to determine the uppercase value of a character when the "convert to uppercase" option was set for a form/report field.

The ability to define alternate character sets within RALLY has been removed from RALLY for V2.0. This change was made to facilitate closer integration with the VMS National Character Set Utility in future versions of RALLY. Both in V1.1 and V2.0, string comparisons are made on the basis of byte-by-byte comparison of character code values.

### 3.1.7 Menus No Longer Disappear When Partially Occluded

In all previous releases of RALLY, partially occluded menus were blanked out. This made it difficult to implement nested menus, such as the two-column menus used in the RALLY Definition System. This problem has been fixed in V2.0.

You might find that applications created before V2.0 can contain form/reports that are less than 24 lines long. When such a form/report is executed, part of the previous menu can appear at the bottom of the screen. To correct this problem, edit the form/report and change the size of its main group to 24.

### 3.1.8 Changed Behavior of RETURN_TO Call Type

The behavior of the RETURN_TO call type has been changed to be consistent with the behavior of the 'first menu' command. (The 'first menu' command does the same thing as typing "top" on a menu.) In V1.1, these differed with respect to intermediate ADL procedures on the stack: such ADL procedures were exited in mid-stream for RETURN_TO, but run to completion for 'first menu'. In V2.0, intermediate ADL procedures are run to completion for RETURN_TO as well as for 'first menu'.

This change will not impact most applications. However, if you use the RETURN_TO call type in your application, you should check your application to determine if the behavior of your ADL procedures will be adversely affected by this change. If so, you can restore the previous behavior by using a global variable to control when the affected ADL procedures exit.

### 3.1.9 Changed Behavior of Before/After-Commit/Rollback Action Sites

The following action sites now are invoked less often than in V1.1: before-commit, before-rollback, after-commit, after-rollback. In V2.0 these action sites are invoked, as their names suggest, before and after commits and rollbacks. In V1.1, these action sites, especially before- and after-commit, were often invoked where there was actually no commit or rollback.

For example, in V2.0 the before-commit action site is not invoked in a form/report having no data groups. In V1.1, the before- commit action site could have been used to perform cross-field validation on variable fields in the main group. To perform such validation in V2.0, use the after-form/report action site.

You should check your form/reports to see if they will be impacted by this change. You will be able to restore the V1.1 behavior by using the after-form/report action site where you previously used the before-and-after-commit action sites.

### 3.1.10 Changed Behavior of 'Next Group'

The RALLY commands 'insert record next_group', 'next group', 'previous group', 'last group', and 'first group' in V2.0 determine the order of sibling groups on a different basis from that used in V1.1. In V1.1, sibling groups were visited in the order in which the groups appeared on the output order of their parent group. In V2.0, the visitation order (formerly called "input order") determines the order in which sibling groups are visited.

To maintain the behavior from V1.1, edit the visitation order to list the groups in the desired order. Conditional next field definitions can be used in V2.0 to control which sibling groups are visited and in what order.

### 3.1.11 GET_CMD Now Returns 'Select_Value'

In V1.1, GET_CMD returned the alias for 'next field' when the user selected a value from a list of values. In V2.0, GET_CMD returns 'select value'. If you trapped selection from an LOV by looking for 'next field', you must modify your application to trap 'select_value' instead.

### 3.1.12 GET_CMD Now Returns 'Finish Action'

In V1.1, the ADL built-in function GET_CMD returned incorrect values for the commands 'finish action' (exitaction), 'finish task' (exittask), 'finish application' (exitappl), 'quit action' (abortaction), 'quit task' (aborttask), and 'quit application' (abortappl). In V2.0, GET_CMD returns the correct value. ADL procedures can detect which command was used by testing for the corresponding command alias.

Applications built in RALLY V1.1 that tested for the incorrect values should be changed to test for the correct values.

### 3.1.13 Key Definition Changes

As described in Chapter 1, New and Changed Features in Version 2.0, V2.0 includes a number of key definition changes. These changes can impact applications in subtle ways, so you should examine the changes carefully to determine if there will be any impact on your application. For example, some V1.1 applications use GET_CMD to trap use of the 'select_value' key and dispatch a field- specific action. Since the keys that were bound to 'select_value' are now bound to 'local_function' instead, these applications will require modification to allow use with the V2.0 key definitions. Until such changes are made, you can continue using them with the V1.1 key definitions, which are supplied with the V2.0 kits.

## 3.2 Definition System

This section describes aspects of the Definition System that are not in keeping with the documentation or that vary from the behavior that you might otherwise expect to find.

### 3.2.1 'Local_Function' Not Always Available

Within the Definition System, you can often jump from editing one object to editing a related object by using the 'local_function' command. However, there are a number of places where you might expect to be able to do this and cannot. For example, you cannot use 'local_function' when you are creating a group and would like to create a DSD to base the group on. Likewise you cannot use 'local_function' to jump from editing a field to editing the field's LOV group. When in doubt, read the legend. In all cases where a local function is available, the legend will indicate this and describe the action that is invoked.

### 3.2.2 Changes Made With 'Local_Function' May Not Appear Immediately

There are a number of cases in which, if you use 'local_function' to jump from editing one object to editing another object and then make a change to the related object, that change will not be visible when you return to the first object. For example, this is the case when you jump from editing a data group, use 'local_function' to edit the DSD for the data group, add a field to the DSD, then return to editing the data group. You must exit from editing the form/report before the new DSD field will be included in the LOV of fields in that DSD that appears when you edit the data group.

### 3.2.3 Builder Tools Cannot Create Form/Report Over Placeholders

You cannot use the form/report option of the Builder Tools to create a form/report when there is already an object in your application with the name that you specified for the form/report, even if the object is just a placeholder. An error message will be displayed if you attempt to do this. This situation comes up most frequently when you delete a form/report that is referenced by a form/report packet, and then go back to recreate the form/report using the form/report option of the Builder Tools. In this situation, you can remove the references to the placeholder or create the new form/report using the *Edit Application* menu instead of the Builder Tools.

### 3.2.4 Message Text Not Displayed in Trace Log

The message text for RALLY error messages recorded by the debugger trace log is not readily available. Usually, you can obtain the text of an error message by writing a small application (or modifying an existing application) that asks for an error number and then uses the ADL function ERROR to display the message for that number.

This technique will not work with message numbers 12287, 19871, 29019, and 31743. These message numbers are used to display messages that have variable contents, such as file names and messages returned by other VAX products such as Rdb/VMS, CDD/Plus, or DATATRIEVE.

### 3.2.5 Limit of 32 Fields Copied from LOV

When the list of values option "copy other fields from list of values" is used, a maximum of 32 fields can be copied from the list of values.

### 3.2.6 Use INTEGRATE /REPLACE after COMPACT

If you have integrated your RALLY application into the CDD/Plus dictionary, you should always invoke the Integrate Utility using the "replace" option after compacting the application.

### 3.2.7 Utilities Do Not Default Output File Names

The DCL commands for the RALLY utilities do not default output file names from input file names, as described in the documentation. Instead, the user is prompted for the output file names when they are not stated explicitly. For example, the documentation states that you can invoke the Compact Utility with the command line:

```
$ COMPACT RALLY$TAPES
```

In fact, the name of the output application file is required. If you use the above command line, you will be prompted for the name of the output application file.

### 3.2.8 Utilities Do Not Report Warnings at DCL Level

When RALLY utilities are invoked from DCL level, any warnings that are reported by the utilities do not affect the status that is returned by the utility at DCL level. For example, you might expect to be able to use the DCL command ON WARNING to trap the condition where the Verify Utility finds integrity errors in your application; however, this is not the case. As long as the Verify Utility is able to complete the verification, the return status will be set to success after the verification is completed.

### 3.2.9 Compact Utility Does Not Release Virtual Memory

When you invoke a RALLY utility from the *Utilities* menu while editing an application, the amount of virtual memory used by RALLY increases significantly and the extra virtual memory is not released until you exit from the Definition System. To avoid this problem, use the DCL command RALLY COMPACT to compact large application files rather than compacting them from within the Definition System.

By default the Merge Utility compacts the output application file after completing the merge, so this problem can impact the merging of applications as well. To avoid this problem when merging, either suppress compaction of the output file or use the DCL command RALLY MERGE.

### 3.2.10 Resolving Conflicts in the Merge Utility

By default, the Merge Utility resolves conflicts in non-global objects of the same name and type by using the information from the copy file. The master application file always provides the global information.

In most cases, if the merge encounters two objects of the same name but different types, RALLY issues an informational message and does not merge the two objects. The rest of the merge continues. However, if the merge encounters two global variables of the same name but different types, the entire merge fails.

### 3.2.11 /FIELD, /GLOBAL_INFORMATION, and /OUTPUT Qualifiers in the Report Utility

From the DCL level, use the REPORT/DATABASE_CROSS_REFERENCE/FIELD command if you want field-level information about data source information corresponding to data source definition fields in your application.

From the DCL level, use the /GLOBAL_INFORMATION qualifier with the /ALL_OBJECTS qualifier to get the application's global information. Global information includes the default number format, the default number character set, default character and date values, default menu information, help and error characteristics, global variables, and application commands.

From the DCL level use the REPORT/OUTPUT= command to specify the destination of the report (to a file or to your terminal).

## 3.3 Other Problems and Restrictions

This section describes various other problems and restrictions that apply to the definition and use of RALLY applications.

### 3.3.1 Online Release Notes

If there are any discrepancies between the hard copy release notes and the online release notes, the online release notes are more current.

### 3.3.2 VAX RALLY Not Supported During VAXcluster Rolling Upgrade

RALLY V2.0 is not supported during a "rolling upgrade" of VMS. If RALLY V2.0 is used in a VMS V5.0 VAXcluster environment, all nodes in the VAXcluster must be running VMS V5.0 before you install RALLY V2.0.

A similar restriction applies to the VAX RALLY Run-Time Option. Use of the VAX RALLY V2.0 Run-Time Option with RALLY applications that are based on Rdb/VMS is not supported during a rolling upgrade of VMS. This restriction does not apply to applications that do not use Rdb/VMS. Use of the Run-Time Option is supported during a rolling upgrade of VMS when the applications to be executed use only VAX Record Managment System (RMS) files and VAX DATATRIEVE domains.

### 3.3.3 CALL_CMD() Not Available in Validation ADL Procedures

An ADL procedure that validates a form/report field cannot call the ADL built-in procedure CALL_CMD. A run-time error message will result.

### 3.3.4 RMS NULL_KEY Attribute Not Supported

In V2.0, RMS DSDs do not support the alternate key NULL_KEY attribute. This includes both RMS file creation and record storage.

### 3.3.5 Limit of Five Active VAX DATATRIEVE DSDs

VAX DATATRIEVE imposes a limit of five DATATRIEVE streams open at any given time. RALLY uses one DATATRIEVE stream for each DATATRIEVE DSD. As a result you cannot have more than five DATATRIEVE DSDs opened at the same time. A run-time error message will result.

### 3.3.6 Output Text Parameters Padded with Blanks

In V2.0, parameters passed to 3GL links are always padded with spaces to make them as long as the declared parameter length. Thus, if a RALLY field has the value "abc" but the external data type allows up to six characters, the value "abc   " will be passed to external links.

## 3.4 Internal Errors and System Problems

The following section discusses what you should do should you encounter a RALLY internal error and how to recover from system problems.

### 3.4.1 If You Encounter a RALLY Internal Error

If RALLY has an internal error while you are using it, you should submit a SPR (Software Problem Report) to DIGITAL describing how to reproduce the error. It is best if you can reproduce the problem with one of the example applications. Otherwise, please send DIGITAL a tape with all files required by your application; for example, the application file, help files, error files, and external program code. If you are using Rdb/VMS, include a backup of your database (the RBR file), not the database itself (the RDB file).

### 3.4.2 Recovering from System Problems

If, for any reason, you are unable to exit successfully from the Definition System after making changes to your application, you should:

1. Delete the output application file.
2. Compact the previous version of the file.
3. Edit the compacted file to reenter your changes.

This applies when your work is interrupted by routine system problems, such as loss of power or communications problems, and when there is an internal error within RALLY.

## 3.5 Addition to Files in Appendix A of Installation Guide

In addition to the files listed the *VAX RALLY Installation Guide* in Appendix A, Files Installed, the following file is also installed if you install the Definition System examples: SYS$EXAMPLES:RALLY$TAPES_DEFINE.RGM

# Considerations for the A-to-Z Application Generator (AG) Customer 4

In general, A-to-Z Application Generator (AG) applications will run under RALLY without modification. However, there are several items to be considered when moving an AG application to RALLY.

## 4.1 Bundling with A-to-Z

VAX RALLY is not available bundled with A-to-Z. The VAX RALLY Run-Time System is available separately. The VAX RALLY Run-Time System is equivalent to the part of AG that was bundled with the A-to-Z Base System.

## 4.2 When Does an Application Become a RALLY Application?

An AG application can be run under RALLY and will remain an AG application, until the first time it is edited using RALLY. Once an application is edited using RALLY, it is becomes a RALLY application and is no longer runnable or editable using AG.

After an application has been upgraded, it is still possible to copy individual objects from other AG applications. However, certain items accessible under AG, such as printer definition file names, are no longer accessible.

## 4.3 Method of Invoking the Definition System Interface

RALLY applications do not install on A-to-Z menus. If you desire, you can install RALLY on an A-to-Z menu as you would any other product not specifically designed to layer on top of A-to-Z. Normally, the RALLY Definition System is invoked directly from DCL.

## 4.4 Method of Invoking an Application

AG applications are generally invoked using a command file where macro files, keyboard definition files, and error logging files to be used with the application are determined using parameters.

RALLY applications can be invoked using DCL commands or command files, and items such as keyboard definition files are determined using DCL command switches.

For example, an application is invoked in AG using the following:

```
$ dev_name:ATOZAG$RUNTIMES1 -
    dev_name:ATOZAG$VTA2Z_RUN.ATOZAG$TERMINAL -
    dev_name:appl_name.ATOZAG$APPLICATION -
    dev_name:ATOZAG$ATOZMACS.ATOZAG$TERMINAL -
    "MAIN_TASK" -
    dev_name:DEBUG.LOG
```

The same application might be invoked under RALLY using the following:

```
$ RALLY RUN/TRACE=dev_name:DEBUG.LOG-
     /KEY_DEFINITION=dev_name:ATOZAG$VTA2Z_RUN.ATOZAG$TERMINAL-
     /MACRO_DEFINITION=dev_name:ATOZAG$ATOZMACS.ATOZAG$TERMINAL-
     dev_name:appl_name.ATOZAG$APPLICATION
```

## 4.5 Keyboard Definitions

AG keyboard definition files will continue to function under RALLY V2.0. They can be invoked for use with an application as shown in the previous example.

It can be beneficial to rename existing keyboard definition files to use the VAX RALLY naming conventions. File name correspondences are:

**Table 4–1: Naming Keyboard Definition Files**

| A-to-Z Keyboard Definition Files | VAX RALLY Keyboard Definition Files |
|---|---|
| ATOZAG$DECVT_RUN.ATOZAG$KDF | *.RGC |
| ATOZAG$DECVT_RUN.ATOZAG$TERMINAL | *.RGK |
| ATOZAG$TERMCAP.ATOZAG$TERMINAL | RALLY$TERMINALS.TXT |

Existing keyboard definition files can be modified and rebuilt using the RALLY Define Keys Utility.

## 4.6 Help and Error Files

AG help and error files will continue to function under RALLY V2.0. However, the error file chain should be modified so the last file ATOZAG$ERRORS.ATOZAG$ERROR is replaced by RALLY$ERRORS.RGE.

Developers can choose to retain the AG help files depending on such matters as whether or not the A-to-Z keyboard mappings are retained.

## 4.7 Printer Definition Files

AG printer definition files will continue to function under RALLY V2.0. However, they cannot be modified using RALLY nor can new ones be created using RALLY. Moreover, it will be necessary either to redirect the file name SYS$LIBRARY:RALLY_PRINTER.RGP to the AG printer definition file or to replace the RALLY_PRINTER.RGP file with the AG printer definition file.

## 4.8 3GL Exits

Existing AG 3GL exits should continue to function under RALLY V2.0 without modification.

## 4.9 RMS Support

RMS file support under RALLY V2.0 is integrated with CDD/Plus. RMS DSDs defined under AG will upgrade to RALLY RMS DSDs. However, to modify these DSDs, the record definitions must first be placed into CDD/Plus manually, and any further modifications must be performed using the dictionary facilities rather than directly through RALLY.

Support for AG "Starts With" functionality is available with RALLY V2.0.

## 4.10 Interrupt Key Support

Interrupt key support is available through the RALLY callable interface.

## 4.11 Kit Building Procedures

No kit building facilities are provided with RALLY V2.0 nor will the facilities provided with AG be sufficient to build kits for RALLY applications.

## 4.12 A-to-Z Data Manager

RALLY V2.0 is not integrated with the A-to-Z Data Manager. All integration is through CDD/Plus. Both RALLY V2.0 and the A-to-Z Data Manager V2.1 can read from CDD/Plus. Therefore, metadata stored in CDD/Plus will be available to both RALLY and the A-to-Z Data Manager.

## 4.13 Application File Porting Tool

The application file porting tool is not supported under RALLY V2.0.

## 4.14 A-to-Z Business Graphics Support

AG provided predefined 3GL exits for calling A-to-Z Business Graphics. These 3GL exits will continue to function without modification under RALLY. However, new RALLY applications will not be created containing these 3GL exits.

## 4.15 A-to-Z AG "Built-In" External Links

AG provided predefined 3GL exits for several functions. Many of these 3GL exits will continue to function without modification under RALLY. They can be recreated manually or copied to new RALLY applications. However, new RALLY applications will not be created containing these 3GL exits. The following table lists the AG built-in external links and the level of support under RALLY.

**Table 4–2: AG Built-In External Links and VAX RALLY Support**

| Number | Name | Support |
|---|---|---|
| 63997 | INTERRUPT | Not Supported - See Sections on Callable Interface for Run-Time System and Interrupt Key Support |
| 63993 | TOGGLE_NUMERIC_KEYPAD | Supported |
| 63990 | TOGGLE_STARTS_WITH | Supported |
| 63995 | UC_GRAPH | Supported |
| 63991 | UC_SET_EDIT | Supported |
| 63992 | UC_SET_NUMERIC | Supported |
| 63988 | UC_STARTS_WITH_OFF | Supported |
| 63989 | UC_STARTS_WITH_ON | Supported |

## 4.16 AG Starts-With for RMS

The A-TO-Z Application Generator provided a limited mechanism for using starts-with as a query operator with RMS DSDs. RALLY V2.0 provides run-time support for use of this feature in applications that were created using the A-TO-Z Applications Generator.

# Terminology Changes A

The list shows the old (Version 1.1) terminology and the new (Version 2.0) equivalents.

**Table A-1: Terminology Changes**

| Old Term Or Phrase | New Term Or Phrase |
|---|---|
| Accepts no blanks | Must fill field with non-blank characters |
| Access method diagnostic options | Debugging options (For actions) |
| Action Diagnostic options | Debugging options (For actions) |
| Addable only | Visitable only when inserting new record |
| AFILE | Application file |
| After-events | After-action sites |
| After field | After visiting field |
| Arrow highlighting | Arrow choice indicator |
| Auto skip field | Automatic 'next field' when full |
| Basic controls | Date formatting strings - Date formatting options |
| Before field | Before visiting field |
| Before-events | Before-action sites |

(Continued on next page)

Table A–1: Terminology Changes (Cont.)

| Old Term Or Phrase | New Term Or Phrase |
|---|---|
| Blank character positions not allowed | Must fill field with non-blank characters |
| Browse/update/delete (mode) | Update (mode) |
| Call packet | Form/report packet |
| Conditional next field | Conditional next field definition |
| Constrain to target | Use size of targets to constrain intermediate results |
| Cursor roam | Cursor movement |
| Cursor roam position | Cursor position |
| Data source field | Data source definition field |
| Default value of null field | Null value |
| Defer update of computed fields | Defer calculation of computed fields |
| Delete dependent records | Delete record when parent record is deleted |
| Dialog | Definition System |
| Display format | Output format |
| Display number character set | Default number character set for output |
| Display number format | Default number format for output |
| Edit a display image | Screen editor |
| Edit form/report image | Screen editor |
| Enterable only | Modifiable only when inserting new record |
| External link | External program link |
| External link storage types | External data types |
| External storage type | External data type |
| FIX storage types | FIX data types |
| Field after-event | After visiting field |
| Field before-event | Before visiting field |
| Field for the page number | PAGE NUMBER (field type - main group) |
| Field for today's date | CURRENT DATE (field type - main group) |
| Field for total number of pages | NUM OF PAGES (field type - main group) |
| Field whose value is copied | Source of copy (for COPY field type) |

**Table A-1: Terminology Changes (Cont.)**

| Old Term Or Phrase | New Term Or Phrase |
|---|---|
| File to print to | Name of output file |
| Follow rest of input list | Edit rest of visitation order |
| Form/report call packet | Form/report packet |
| Form/report inheritable options | Record operation options |
| Form/report options | Record operation options |
| Format type | External data types |
| Ignore null records | Ignore null records when reading from database |
| Image editor | Screen editor |
| Image editor's state | (Appear as options for screen editing.) |
| Inheritable form/report options (DSD) | (See *Record Operation Options for a Group* form.) |
| Inheritable form/report options (DSD field) | (Options are on the *Validation for a DSD Field* and *Input Options for a DSD Field* forms.) |
| Initially active | Initially displayed |
| Input format of field | Input format |
| Input order | Visitation order |
| Lines horizontally scroll | Horizontal scrolling |
| Look only field | Display only |
| Make this a non-displayed field | Displayed on [which pages]: NONE |
| Minimal match | Abbreviation allowed |
| Mixed start on roam | Cursor initially on first choice |
| Name of field to apply function to | Field to be aggregated |
| Next field generates new record | Automatic 'insert record after' |
| No key out | Disallow all application commands when in this task |
| No toggle in | Disallow moving into this task from another task |
| No toggle out | Disallow moving into another task |
| No window keys | Disallow window manipulation commands when in this task |
| Options for computed data | Options for computations |

(Continued on next page)

**Table A-1: Terminology Changes (Cont.)**

| Old Term Or Phrase | New Term Or Phrase |
|---|---|
| Output format of field | Output format |
| Output order | Fields, text areas, and child groups |
| Page location of group | Displayed on [which pages]: FIRST, MIDDLE, LAST, ... |
| Page number to display | Number of page to display |
| Page total field | Aggregate on each page |
| Pick value automatically if only... | Select value automatically if only... |
| Precision of number | Scale factor |
| Previous field generates new | Automatic 'insert record before' record |
| Prompt area | Response area |
| Roam position | Cursor position |
| Roam area | Cursor highlight area |
| Round digits | Round during computations |
| Running field | Running aggregate |
| Runtime diagnostic options | Debugging options |
| Script writer | RALLY Report Utility |
| Security AFILE item | Password object |
| Security for global information | Security for editing this application file |
| Storage type | Data type |
| Task area | Window area |
| Today's date (type of field) | CURRENT DATE |
| Total number of pages (type of field) | NUM OF PAGES |
| Up, down, and newline can exit field | Up arrow, down arrow, or RETURN can exit field |
| Upper case each character | Convert to uppercase |
| Use system date | Variable's value is current date |
| Value change after-event | After value change |
| Value change before-event | Before value change |

**Table A–1: Terminology Changes (Cont.)**

| Old Term Or Phrase | New Term Or Phrase |
|---|---|
| Width of number | Number of digits |
| Word conflict check | Check menus for word conflicts |

# How to Order Additional Documentation

## Technical Support

If you need help deciding which documentation best meets your needs, call 800-343-4040 before placing your electronic, telephone, or direct mail order.

## Electronic Orders

To place an order at the Electronic Store, dial 800-DEC-DEMO (800-332-3366) using a 1200- or 2400-baud modem. If you need assistance using the Electronic Store, call 800-DIGITAL (800-344-4825).

## Telephone and Direct Mail Orders

| Your Location | Call | Contact |
|---|---|---|
| Continental USA, Alaska, or Hawaii | 800-DIGITAL | Digital Equipment Corporation<br>P.O. Box CS2008<br>Nashua, New Hampshire 03061 |
| Puerto Rico | 809-754-7575 | Local DIGITAL subsidiary |
| Canada | 800-267-6215 | Digital Equipment of Canada<br>Attn: DECdirect Operations KAO2/2<br>P.O. Box 13000<br>100 Herzberg Road<br>Kanata, Ontario, Canada K2K 2A6 |
| International | ———— | Local DIGITAL subsidiary or approved distributor |
| Internal[1] | ———— | SDC Order Processing - WMO/E15<br>*or*<br>Software Distribution Center<br>Digital Equipment Corporation<br>Westminster, Massachusetts 01473 |

[1]For internal orders, you must submit an Internal Software Order Form (EN-01740-07).

## Reader's Comments

**VAX RALLY**
**Release Notes**
**AA-KZ26A-TE**

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

| I rate this manual's: | Excellent | Good | Fair | Poor |
|---|---|---|---|---|
| Accuracy (software works as manual says) | ☐ | ☐ | ☐ | ☐ |
| Completeness (enough information) | ☐ | ☐ | ☐ | ☐ |
| Clarity (easy to understand) | ☐ | ☐ | ☐ | ☐ |
| Organization (structure of subject matter) | ☐ | ☐ | ☐ | ☐ |
| Figures (useful) | ☐ | ☐ | ☐ | ☐ |
| Examples (useful) | ☐ | ☐ | ☐ | ☐ |
| Index (ability to find topic) | ☐ | ☐ | ☐ | ☐ |
| Page layout (easy to find information) | ☐ | ☐ | ☐ | ☐ |

I would like to see more/less ______

What I like best about this manual is ______

What I like least about this manual is ______

I found the following errors in this manual:

| Page | Description |
|---|---|
| | |
| | |
| | |

Additional comments or suggestions to improve this manual:

______

I am using **Version** ______ of the software this manual describes.

Name/Title ______ Dept. ______

Company ______ Date ______

Mailing Address ______

______ Phone ______

Do Not Tear - Fold Here and Tape

digital™

No Postage
Necessary
if Mailed
in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01-3/J35
110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

Do Not Tear - Fold Here

Cut Along Dotted Line

# Reader's Comments

**VAX RALLY**
**Release Notes**
**AA-KZ26A-TE**

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

| **I rate this manual's:** | Excellent | Good | Fair | Poor |
|---|---|---|---|---|
| Accuracy (software works as manual says) | ☐ | ☐ | ☐ | ☐ |
| Completeness (enough information) | ☐ | ☐ | ☐ | ☐ |
| Clarity (easy to understand) | ☐ | ☐ | ☐ | ☐ |
| Organization (structure of subject matter) | ☐ | ☐ | ☐ | ☐ |
| Figures (useful) | ☐ | ☐ | ☐ | ☐ |
| Examples (useful) | ☐ | ☐ | ☐ | ☐ |
| Index (ability to find topic) | ☐ | ☐ | ☐ | ☐ |
| Page layout (easy to find information) | ☐ | ☐ | ☐ | ☐ |

I would like to see more/less ________________________________________

______________________________________________________________

What I like best about this manual is ___________________________________

______________________________________________________________

What I like least about this manual is ___________________________________

______________________________________________________________

I found the following errors in this manual:

Page   Description

_____   ______________________________________________________

_____   ______________________________________________________

_____   ______________________________________________________

Additional comments or suggestions to improve this manual:

______________________________________________________________

______________________________________________________________

______________________________________________________________

I am using **Version** _______ of the software this manual describes.

Name/Title ______________________________ Dept. ____________

Company ________________________________ Date ____________

Mailing Address _________________________________________________

______________________________________ Phone ____________

Do Not Tear - Fold Here and Tape

digital™

No Postage
Necessary
if Mailed
in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01-3/J35
110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

Do Not Tear - Fold Here

Cut Along Dotted Line

## Reader's Survey

**VAX RALLY**
**Release Notes**
**AA-KZ26A-TE**

1. How useful are the following methods for finding information in this manual?

| | Most | Very | Moderately | Not Very | Not at All |
|---|---|---|---|---|---|
| Table of contents | ☐ | ☐ | ☐ | ☐ | ☐ |
| Divider pages (if applicable) | ☐ | ☐ | ☐ | ☐ | ☐ |
| Index (circle: book or master) | ☐ | ☐ | ☐ | ☐ | ☐ |
| Other (specify) ______ | ☐ | ☐ | ☐ | ☐ | ☐ |

2. What feature do you most want to see improved in this manual? Why?

______

3. How helpful are these sources when you use the software this manual describes?

| | Most | Very | Moderately | Not Very | Not at All |
|---|---|---|---|---|---|
| Handbook or user's guide | ☐ | ☐ | ☐ | ☐ | ☐ |
| Introduction or overview | ☐ | ☐ | ☐ | ☐ | ☐ |
| Reference manual | ☐ | ☐ | ☐ | ☐ | ☐ |
| Quick reference guide | ☐ | ☐ | ☐ | ☐ | ☐ |
| Online help | ☐ | ☐ | ☐ | ☐ | ☐ |
| Online tutorial (if available) | ☐ | ☐ | ☐ | ☐ | ☐ |
| Other: colleague, telephone support services (specify) ______ | ☐ | ☐ | ☐ | ☐ | ☐ |

4. What business tasks are you using the software described by this manual to solve (for example: billing, funds transfer, report writing)?

______

5. Please estimate, if you can, how long the following VAX Information Architecture products have been used at your site:

| | | |
|---|---|---|
| VAX ACMS ______ | VAX CDD/Plus ______ | VAX DATATRIEVE ______ |
| VAX Data Distributor ______ | VAX DBMS ______ | VAX RALLY ______ |
| VAX Rdb/VMS ______ | VAX SQL ______ | VAX TEAMDATA ______ |
| VAX TDMS ______ | VIDA with IDMS/R ______ | |

6. This release of VAX Information Architecture documentation uses a 7x9 format for quick reference guides. Do you prefer such books in a 7x9 or a 4x8 pocket guide format? ______

Thank you for your assistance.

May we contact you at work for further information? ☐ Yes ☐ No

Name ______ Dept. ______

Company ______ Date ______

Mailing Address ______

______ Phone ______

Do Not Tear - Fold Here and Tape

digital™

No Postage
Necessary
if Mailed
in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK02-2/N53
110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

Do Not Tear - Fold Here

Cut Along Dotted Line

DECLIT AA VAX KZ26A

VAX RALLY release notes

DECLIT AA VAX KZ26A

VAX RALLY release notes

•Ø198 3•

Printed in U.S.A.