CA IDMS™

IDD DDDL Reference Guide

Release 18.5.00, 2nd Edition

CA Technologies Product References

This document references the following CA Technologies products:

- CA ADS™
- CA ADS™ Alive
- CA IDMS™/DB
- CA IDMS™/DC (DC)
- CA IDMS™/DC Sort
- CA IDMS™/DC or CA IDMS™ UCF (DC/UCF)
- CA IDMS™ Dictionary Module Editor (CA IDMS DME)
- CA IDMS™ UCF (UCF)

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Documentation Changes

The following documentation updates were made for the 18.5.00 release of this documentation:

- DISPLAY/PUNCH ALL Statement (see page 116)—A description of the new RECursive parameter that appends "AS SYNTAX." or "AS COMMENT." to each generated line of output was added.
- DISPLAY/PUNCH Examples (see page 124)—An example of using the RECursive parameter was added.
- Entity-Type Syntax (see page 127)—The syntax diagrams of all statements (except RECORD SYNONYM) have been updated with the RECursive parameter.

Contents

Chapter 1: Introduction 11

Syntax Diagram Conventions 11

Chapter 2: Coding Considerations 15

Overview 16
Syntax Format 18
Character Set Restrictions 21
Keywords and User-supplied Names 23
Input Column Range 24
Comment Text, Source Statements, and Edit Instructions 24
- Comment Text 25
- Source Statements 26
- Edit Instruction Text 26

Batch Considerations 26
- User Comments 27
- Carriage Control Statements 27

Chapter 3: DDDL Compiler Options 29

Overview 29
SIGNON Statement 30
SIGNOFF Statement 33
SET OPTIONS Statement 34
- SET OPTIONS Functions 34
- SET OPTIONS Syntax 36
- SET OPTIONS Defaults and Overrides 59
- SET OPTIONS Security 62

DISPLAY/PUNCH OPTIONS Statement 63
INCLUDE Statement 64
COMMIT Statement 66

Chapter 4: General DDDL Syntax Options 67

Overview 67
Identifying Entity Occurrences 68
- NAME Clause 68
- VERSION Clause 69

Additional Qualifiers 71
Securing the Dictionary 72
PREPARED/REVISED BY Clause 73
AUTHORITY Clause 75
USER Clause 76
PUBLIC ACCESS Clause 79
Documenting Entity Occurrences 81
DESCRIPTION Clause 81
COMMENTS Clause 83
TEXT Clause 87
Copying and Editing Entity Occurrences 88
SAME AS Clause 88
COPY Clause 91
EDIT Clause 93
INSERT Instruction of the EDIT Clause 96
ERASE Instruction of the EDIT Clause 98
REPLACE Instruction of the EDIT Clause 99
LIST Instruction of the EDIT Clause 101
SEQUENCE Instruction of the EDIT Clause 101
SHOW Instruction of the EDIT Clause 102
Associating Entity Occurrences 103
Relational Keys 104
Attribute/Entity Relationships 109
Displaying Entity Occurrences 112
DISPLAY/PUNCH Statement 113
DISPLAY/PUNCH ALL Statement 116
WHERE Clause (Conditional Expressions) 118
DISPLAY/PUNCH Examples 124

Chapter 5: Entity-Type Syntax 127

Overview 128
Considerations for Syntax Presentation 129
ATTRIBUTE 131
CLASS 138
DESTINATION 144
ELEMENT 151
ELEMENT SYNONYM 167
ENTRY POINT 169
FILE 174
FILE SYNONYM 184
LINE 186

LOAD MODULE 192
LOGICAL TERMINAL 198
MAP 205
MESSAGE 211
MODULE (PROCESS/QFILE/TABLE) 219
PANEL (SCREEN) 234
PHYSICAL TERMINAL 239
PROCESS 247
PROGRAM 255
QFILE 272
QUEUE 281
RECORD (REPORT/TRANSACTION) 287
    RECORD Statement 288
    RECORD ELEMENT Substatement 307
    COBOL Substatement 322
    REMOVE ALL Substatement 331
    VIEW ID Substatement 332
RECORD SYNONYM 333
SYSTEM (SUBSYSTEM) 334
TABLE 341
TASK 351
USER 358
USER-DEFINED ENTITY 380

Chapter 6: Online DDDL Compiler 387

Overview 387
Screen Format 388
Online Sessions 389
    Beginning a Session 390
    Conducting an Online Session 390
    Terminating a Session 392
    Recovering a Session 393
Online Commands 393
    Top-line Commands 394
    Line Commands 395
Program Function Keys Assigned to Operations 396

Chapter 7: IDD Menu Facility 399

Overview 399
Screen Formats 400
    Fixed Screens 401

Pageable Screens....403
Using Menu Facility Screens....405
Predefined Control Keys....405
Cursor Positioning....407
Message Display and Field Highlighting....407
Default Value Assignment....408
Help Screens....409
Online Commands....409
Top-line Commands....410
Line Commands....411
Conducting a Menu Facility Session....411
Beginning a Session....412
Navigating Screens....413
Displaying Entity Occurrences....415
Adding Entity Occurrences....416
Modifying Entity Occurrences....417
Deleting Entity Occurrences....418
Terminating a Session....418
Descriptions of IDD Menu Facility Screens....419
Entry and Processing Screens....420
Screens Common to All Entity Types....421
ATTRIBUTE Entity Screens....422
CLASS Entity Screens....423
ELEMENT Entity Screens....423
FILE Entity Screens....425
MESSAGE Entity Screens....426
MODULE Entity Screens....426
PROCESS Entity Screens....427
PROGRAM Entity Screens....428
QFILE Entity Screens....430
RECORD Entity Screens....431
SYSTEM Entity Screens....432
TABLE Entity Screens....433
USER Entity Screens....434
Sample Session....436

Appendix A: DDDL Compiler Batch Execution JCL 443

IDMSDDDL Under z/OS....443
IDMSDDDL Under z/VSE....445
IDMSDDDL Under z/VM....453

Appendix B: Syntax Converters for COBOL and PL/I 455

IDMSIDDC (COBOL Converter) 455
IDMSIDDP (PL/I Converter) 455

Appendix C: Data Transfer Between Dictionaries 457

Overview 458
Steps for Data Transfer 459
Example of Transferring Data Between Dictionaries 460
Completing the Data Transfer 461
Transferring in Batch Mode 461

Appendix D: Default Version Number Conventions 463

Appendix E: IDD User-Exit Program 465

When a User Exit Is Called 466
Rules for Writing the User-exit Program 467
Control Blocks and Sample User-exit Programs 469
Sample IDD User-exit Program 470

Appendix F: Using the DDDL Compiler as a Subprogram 477

IDMSDB--Overview 477
Compiler Interface Parameter List 479
Work-area File 479
Sample Program that Calls IDD 480

Appendix G: Double-Byte Character Set (DBCS) Strings 487

Overview 488
Coding DBCS Strings 489

Chapter 1: Introduction

The CA IDMS IDD DDDL Reference Guide serves as the primary source for using the Data Dictionary Definition Language (DDDL) to populate and maintain dictionaries.

The purpose of this manual is to provide information on DDDL syntax, coding considerations, and DDDL compiler options. The manual also includes information on entering statements using the online compiler and the Integrated Data Dictionary (IDD) menu facility.

This manual is intended for anyone who uses the dictionary or who is responsible for dictionary administration.

This section contains the following topics:

Syntax Diagram Conventions (see page 11)

Syntax Diagram Conventions

The syntax diagrams presented in this guide use the following notation conventions:

`UPPERCASE OR SPECIAL CHARACTERS`

Represents a required keyword, partial keyword, character, or symbol that must be entered completely as shown.

`lowercase`

Represents an optional keyword or partial keyword that, if used, must be entered completely as shown.

`italicized lowercase`

Represents a value that you supply.

`lowercase bold`

Represents a portion of the syntax shown in greater detail at the end of the syntax or elsewhere in the document.

Points to the default in a list of choices.

Indicates the beginning of a complete piece of syntax.

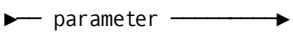
Indicates the end of a complete piece of syntax.

Indicates that the syntax continues on the next line.

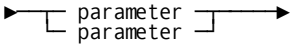
Indicates that the syntax continues on this line.

Indicates that the parameter continues on the next line.

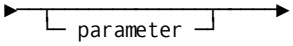
Indicates that a parameter continues on this line.



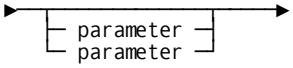
Indicates a required parameter.



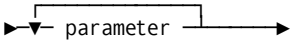
Indicates a choice of required parameters. You must select one.



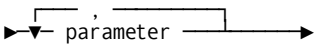
Indicates an optional parameter.



Indicates a choice of optional parameters. Select one or none.



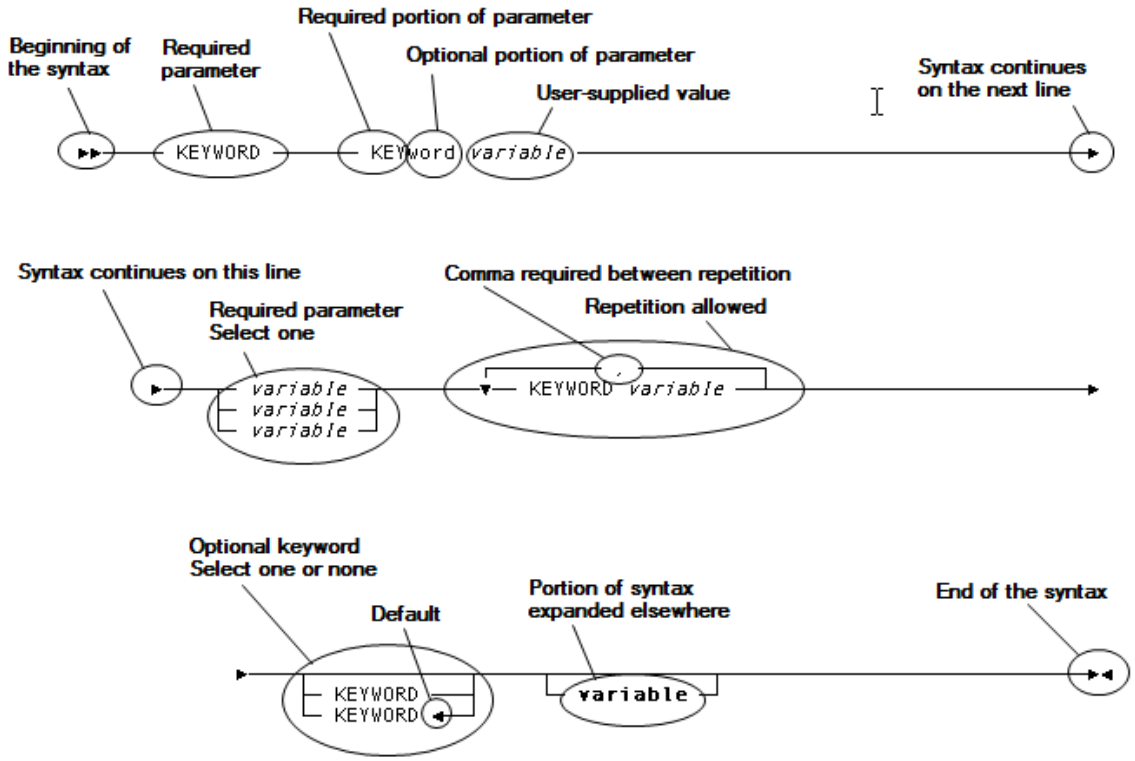
Indicates that you can repeat the parameter or specify more than one parameter.



Indicates that you must enter a comma between repetitions of the parameter.

Sample Syntax Diagram

The following sample explains how the notation conventions are used:



Chapter 2: Coding Considerations

This section contains the following topics:

Overview (see page 16)
Syntax Format (see page 18)
Character Set Restrictions (see page 21)
Keywords and User-supplied Names (see page 23)
Input Column Range (see page 24)
Comment Text, Source Statements, and Edit Instructions (see page 24)
Batch Considerations (see page 26)

Overview

The Data Dictionary Definition Language (DDDL) is the source language used to maintain data resource components in the dictionary. Input to the DDDL compiler consists of source statements; the compiler processes the input and populates the data dictionary. Once added to the dictionary, data resource definitions can be modified, replaced, deleted, displayed, and punched by using DDDL syntax.

You can submit input to the DDDL compiler in online or batch mode.

Note: Identical syntax rules apply to online and batch input to the DDDL compiler.

Online Statement Entry

Online, you can submit statements in three ways:

- **Submit freeform source statements directly to the DDDL compiler.** A text editor writes the input to and output from the DDDL compiler to a work file, the contents of which are displayed at the terminal. At the end of each online session, the DDDL compiler displays a Transaction Summary. The summary lists the DDDL statements submitted and the number of error messages issued.

  More information: For more information on using the DDDL compiler online, see Online DDDL Compiler (see page 387).

- **Submit information through the IDD menu facility using standard, fixed-format screens.** Because syntax parameters are presented on each screen, the user need not be familiar with DDDL syntax and rules. Definitions can be displayed, added, modified, or deleted from the dictionary.

  More information: For more information on using the IDD menu facility, see IDD Menu Facility (see page 399).

- **Submit source statements using the command facility.** You can submit source statements, optionally identifying any ambiguous entity types (for example, DDDL FILE as opposed to DDL FILE) using the keyword IDD.

  More information: For more information on using the command facility to enter DDDL statements, refer to the *CA IDMS Common Facilities Guide* document.

Batch Statement Entry

As an alternative to online statement entry, you can submit freeform source statements to the DDDL compiler in batch mode. Output from the DDDL compiler consists of the Integrated Data Dictionary Activity List, which lists each source input statement and a Transaction Summary (as described above under online entry). Error messages issued during the batch run appear on the line immediately following the problem statement.

The following figure shows a sample Integrated Data Dictionary (IDD) Activity List.

```
IDMSDDDL     nn.n                                CA                          DATE          TIME          PAGE
                              INTEGRATED DATA DICTIONARY ACTIVITY LIST        mm/dd/yy      08463000      0001

0001            SET OPTIONS INPUT COLUMNS ARE 1 THRU 71.
0002     MODIFY ENTITY SYSTEM
0003        USER DEFINED NEST IS 'SIMILAR FILE'.
0004     MODIFY ENTITY SYSTEM
0005        USER DEFINED NEST IS 'RELATED FILE'.
0006     ADD CLASS ENTITY-STATUS
0007        DELETION LOCK IS ON
0008        ATTRIBUTES ARE MANUAL SINGULAR.
0009     ADD CLASS ENTITY-TYPE
0010        DELETION LOCK IS ON
0011        ATTRIBUTES ARE SINGULAR.
0012     ADD ATTRIBUTE PRODUCTION
0013        WITHIN CLASS ENTITY-STATUS
0014        DELETION LOCK IS ON
0015        COMMENTS 'DESIGNATES PRODUCTION OCCURRENCES'.
0016     ADD SYSTEM ORDER-CONTROL.
0017     ADD SYSTEM BACK-ORDER.
0018     ADD SYSTEM INVENTORY
0019        WITHIN SYSTEM ORDER-CONTROL
0020        ENTITY-STATUS PRODUCTION
0021        'SIMILAR FILE' BACK-ORDER.



IDMSDDDL     nn.n              COMPUTER ASSOCIATES INTERNATIONAL, INC.       DATE          TIME          PAGE
                              INTEGRATED DATA DICTIONARY ACTIVITY LIST        mm/dd/yy      08463000      0002

              ** TRANSACTION SUMMARY **
ENTITY                 ADD MODIFY REPLACE DELETE DISPLAY
...................... ... ...... ....... ...... .......

ENTITY                   0      2       0      0       0
ATTRIBUTE                1      0       0      0       0
CLASS                    2      0       0      0       0
SYSTEM                   3      0       0      0       0

NO ERRORS OR WARNINGS ISSUED FOR THIS COMPILE
```

The sections in this chapter describe the general rules for preparing statements for input to the DDDL compiler.

Syntax Format

DDDL compiler input consists of statements arranged in a prescribed syntactical order. These statements reflect the logical organization of the dictionary by supporting standard IDD entity types, entity-type synonyms, entity types that support CA IDMS functions, and user-defined entity types.

All DDDL entity-type statements include the following five components:

- Verb
- Entity-type
- Entity-occurrence
- Optional clauses
- Period

Descriptions of these components follow.

Verb

The verb designates the requested function. One of the following verbs must accompany each DDDL entity-type statement:

- **ADD** establishes a new entity occurrence in the dictionary (see the table below for information about acceptable synonyms).
- **MODIFY** updates an existing entity occurrence with user-supplied options (see the table below for information about acceptable synonyms).
- **DELETE** removes an existing entity occurrence from the dictionary (see the table below for information about acceptable synonyms).

  For compatibility with CA IDMS SQL, the following verbs have been included as acceptable synonyms for ADD, MODIFY, and DELETE.

Verb	Synonym
ADD	CREATE
MODIFY	ALTER
DELETE	DROP

- **REPLACE** replaces an existing entity occurrence but preserves relationships established through other entity-type syntax.
- **DISPLAY** displays one or more existing entity occurrences, as follows:
  - In an online session, DISPLAY lists the requested definitions at the terminal. The user can edit the output and resubmit it to the DDDL compiler.
  - In batch mode, DISPLAY prints the requested definitions on the Integrated Data Dictionary Activity List.
- **PUNCH** lists one or more existing entity occurrences, as follows:
  - In an online session, PUNCH lists the requested definitions at the terminal. The user can edit the output and resubmit it to the DDDL compiler.
  - In batch mode, PUNCH writes the requested definitions to the SYSPCH output file or to an IDD module that has been defined as the PUNCH destination.

Entity Type

The entity type identifies the type of data that is the object of the specified verb. The DDDL compiler supports the following standard IDD entity types, entity-type synonyms, and CA IDMS components:

ATTRIBUTE	PROGRAM
CLASS	QFILE
DESTINATION	QUEUE
ELEMENT	RECORD
ELEMENT SYNONYM	RECORD SYNONYM
ENTRY POINT	REPORT
FILE	SUBSYSTEM
FILE SYNONYM	SYSTEM
LINE	TABLE
LOAD MODULE	TASK
LOGICAL-TERMINAL	TRANSACTION
MAP	USER
MESSAGE	User-defined entity
MODULE	PROCESS
PANEL	PHYSICAL-TERMINAL

Entity Occurrence

The entity occurrence identifies a specific occurrence of the named entity type, entity synonym, or CA IDMS component. An entity occurrence consists of a name and an optional version number and language (attribute within the system-supplied class LANGUAGE), which permits the user to assign one name to multiple entity occurrences.

Optional Clauses

Optional clauses provide qualifying data for each entity occurrence. Once an entity occurrence has been defined, the user can extend its basic definition with optional data and comments and can associate the entity with other occurrences of the same entity type.

Period

A period signifies the end of the statement and is required in all DDDL statements. The period can directly follow the last word in the statement, can be separated from the last word by blanks, or can appear on a separate line. If specified in the SET OPTIONS statement (see SET OPTIONS Statement (see page 34)), a semicolon can also be used as an end-of-statement character.

Note: The end-of-statement character is not shown in the syntax diagrams for DDDL statements.

Order of Components

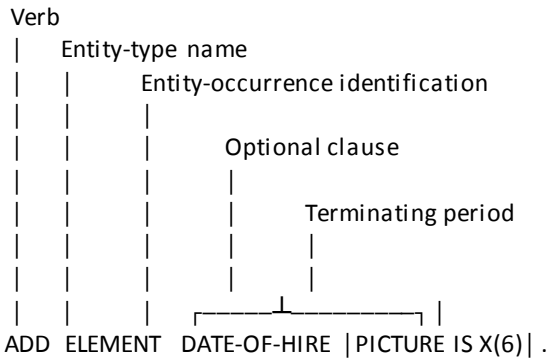
The verb, entity-type name, and entity-occurrence identification, which are required in all DDDL statements, must be specified in the order described above.

Optional clauses follow the entity-occurrence identification and can be specified in any order.

The period (or alternate character) must terminate each statement.

Example of Statement Components

The following example graphically illustrates the components of a typical DDDL statement:



Exceptions to the syntax format rules are indicated in the syntax presentation.

Character Set Restrictions

To accommodate a broad range of applications, both automated and manual, the DDDL compiler enforces a minimum number of character set restrictions. These restrictions are shown in the following table.

Restriction for:	Description
Statement terminator	Use a period to terminate each DDDL statement. Assume the use of the terminating period when you read the syntax in this document. It will not be shown in syntax diagrams. If the period is omitted, the DDDL compiler executes the statement from which the period has been omitted, issues a warning message, and flags all subsequent statements with an error message. **Note:** You can establish recognition of the semicolon as an alternative end-of-statement character if you specify the SET OPTIONS clause SEMICOLON ALTERNATE END OF SENTENCE IS ON. For more information, see SET OPTIONS Statement (see page 34).
Delimiters	Use one or more blanks as a delimiter. Commas, semicolons, and colons are treated as blanks. **Note:** You cannot use the semicolon as a delimiter if you've defined it as a statement terminator. Throughout this manual, commas have been included in numbers to enhance readability (for example, where the manual uses 32,767, you should use 32767).

Restriction for:	Description
Null strings	Use two single quotation marks with no intervening space to nullify existing values. Note, however, that comment text cannot be nullified in this manner.
Quotation marks	You must use a quotation mark (or a special character designated as the site-standard quote character) to enclose user-supplied names containing one or more embedded delimiters (blanks, commas, periods, semicolons, apostrophes, parentheses, colons, and quote characters). **Note:** You cannot use the semicolon as a delimiter if you've defined it as a statement terminator. The DDDL compiler interprets any word enclosed in quotation marks as a user-supplied value, even if the word is a DDDL keyword. For example: add element <--- **DDDL keyword** name is 'element' <--- **user-supplied name** pic X(9). The IDD installation procedure establishes the single quotation mark (') as the default quote character. However, the user can define a site-standard quote character by using the QUOTE IS clause of the SET OPTIONS statement (see SET OPTIONS Statement (see page 34)). If you want to include the site-standard quote character in a user-supplied name, code that character twice. For example, assuming that the single quotation mark (') is the site-standard quote character, the name MARY'S PROGRAM must be input as 'MARY''S PROGRAM'.

Keywords and User-supplied Names

Keywords

Keywords are predefined names or special characters that are either required (shown as uppercase in syntax diagrams) or optional (shown as lowercase). You can enter the full keyword or abbreviate each keyword to a minimum of three characters, provided that no other keyword in the same syntactical position is abbreviated identically. The keywords ELEMENT and VERSION are exceptions to the three-character minimum; they can be abbreviated to EL and V, respectively. Keyword abbreviations that require more than three characters are noted in the syntax.

User-supplied Names

User-supplied names are names that you define. In addition to the character set restrictions, observe these points when you define names:

- Names must be unique.
- Names must not duplicate any of the reserved words of a specific compiler or assembler, and they must observe the compiler's character set and word-length restrictions.
- Names must be 1 to 32 characters (with noted exceptions).
- Valid characters for names:
  - Letters (A through Z) (uppercase or lowercase)
  - Digits (0 through 9)
  - At sign (@)
  - Dollar sign ($)
  - Pound sign (#)
  - Hyphen (-) (the first and last character in a name can't be a hyphen)
  - Underscore (_)
- A name must include at least one nonnumeric character.
- When you assign names to user-defined nests, comment keys, and alternative picture keywords, the DDDL compiler classifies each word and places it in the first appropriate category, as follows:
  1. DDDL keyword
  2. User-defined comment key
  3. User-defined nest

4. User-defined nest inverse key
5. Alternative picture keyword (ELEMENT entity type only)
6. Class name

For example, if a user-defined comment key that is not enclosed in quotation marks is the same as a DDDL keyword, the DDDL compiler interprets the comment key as a DDDL keyword.

Input Column Range

You can code DDDL source statements on a single line or on multiple lines, in columns 1 through 80. However, the IDD installation procedure establishes these default input ranges:

- Batch compiler— 1 through 72
- Full-screen mode— 1 through 79
- Line mode— 1 through 80

You can override input range defaults by using the INPUT COLUMNS ARE clause of the SET OPTIONS statement (see SET OPTIONS Syntax (see page 36)).

Comment Text, Source Statements, and Edit Instructions

This section describes special rules and considerations for coding these variables:

- *comment-text*
- *source-statement*
- *edit-instruction*

Comment Text

Comment text is represented in the DDDL statements as *comment-text*. Guidelines for entering comment text are as follows:

- Enter comment text in columns 1 through 80. Include the quote character in the input column count when determining the number of characters per line.
- Comments can consist of any number of lines.
- Each line must begin with the site-standard quote character.
- Each line must end with the site-standard quote character, unless it is being continued on the next line.
- To continue comment text beyond one line, code a hyphen character on all subsequent lines (the hyphen is included in the number of characters per line). The hyphen can appear anywhere within the specified input column range, provided it is the first character on the continued line.

  Note: The DDDL compiler does not process lines of commenttext that begin with an asterisk (*).
- To concatenate lines of comment text into one line containing a maximum of 80 characters, code a plus sign (+) as the first character on the line to be concatenated.

  Note: You must include a closing quotation mark on a line that is to be concatenated.

Example of Continuation and Concatenation

The following example illustrates lines of comment text that are to be continued (as represented by the hyphen) and concatenated (as represented by the plus sign). Note that spaces coded before the closing quotation mark are included in concatenated lines:

```
'lengthy input can be concatenated'
-'onto one line '
+'containing up to 80 characters.'
-'code a + as the first character on a line'
-'that is to be concatenated.'
-'the plus sign and quotation mark'
-'are not included in the '
+'80-character count.'
```

The comment text as it would appear on a batch report:

```
LENGTHY INPUT CAN BE CONCATENATED
ONTO ONE LINE CONTAINING UP TO 80 CHARACTERS.
CODE A + AS THE FIRST CHARACTER ON A LINE
THAT IS TO BE CONCATENATED.
THE PLUS SIGN AND QUOTATION MARK
ARE NOT INCLUDED IN THE 80-CHARACTER COUNT.
```

Source Statements

Module, process, and q-file source statements are represented in DDDL statements as *source-statement*. Source statements can consist of any number of input lines. The DDDL compiler reads text in columns 1 through 80 and places it in the requested module, process, or q-file.

Source input is terminated when the DDDL compiler encounters an MSEND instruction.

Example of Source Statements

The following example shows the statements associated with the MODULE ADDRESS1 in the dictionary:

```
add module address1
    module source follows
       900010 move cust-name to name-line.
       900020 move street-no to no-line.
       900030 move street-name to str-line.
       .
       .
       .
       900080 move zip-code to z-no.
       900090 write label-rec.
     msend.
```

More information: For more information on defining modules, processes, and q-files, see MODULE (PROCESS/QFILE/TABLE) (see page 219).

Edit Instruction Text

An edit instruction is represented in a DDDL statement as *edit-instruction*. EDIT instruction text can consist of any number of input lines. The DDDL compiler reads the contents of columns 1 through 80 and associates it with the text that is the object of the EDIT instruction.

Valid keywords in an edit instruction are INSERT, REPLACE, ERASE, LIST, SEQUENCE, and SHOW. CEND terminates the INSERT and REPLACE edit instructions.

More information: For more information on edit instructions, see EDIT Clause (see page 93).

Batch Considerations

Coding for user comments and coding for carriage control statements are described in this section.

User Comments

Use *+ or -- anywhere on an input line to indicate that the remainder of the line is a user comment. If you code these characters in the first two positions, the line will not be echoed. If you want the comment line to be echoed, use the character * and a space in the first two positions.

Carriage Control Statements

The SKIP and EJECT statements are used to format the Integrated Data Dictionary Activity List. These control statements are not printed and do not affect the operation of the DDDL compiler.

SKIP Statement

The SKIP statement inserts one, two, or three blank lines between any two DDDL source statements. SKIP can also be used in an online session to create a single blank line.

Syntax for the SKIP statement is SKIP*count*. *Count* is the number 1, 2, or 3 (for example, SKIP2).

SKIP1/2/3 specifies that the DDDL compiler is to insert 1, 2, or 3 blank lines in the Integrated Data Dictionary Activity List following the line on which the SKIP statement appears.

The following rules apply to the SKIP statement:

- The specified integer cannot be separated from the keyword SKIP. SKIP1 is valid; SKIP 1 is invalid.
- The keyword SKIP must be by itself on the line.
- If the keyword SKIP appears within module source code that is the object of an INSERT or REPLACE instruction, the DDDL compiler interprets SKIP as part of the module source code.

EJECT Statement

The EJECT statement specifies advancement of the paper to the top of a new page before printing the next source statement. Typically, EJECT is used to format the Integrated Data Dictionary Activity List by entity type.

Syntax for the EJECT statement is simply EJECT. The following rules apply to the EJECT statement:

- The keyword EJECT must be by itself on the line.
- If the keyword EJECT appears within module source code that is the object of an INSERT or REPLACE instruction, the DDDL compiler interprets EJECT as part of the module source code.

Chapter 3: DDDL Compiler Options

This section contains the following topics:

Overview (see page 29)
SIGNON Statement (see page 30)
SIGNOFF Statement (see page 33)
SET OPTIONS Statement (see page 34)
DISPLAY/PUNCH OPTIONS Statement (see page 63)
INCLUDE Statement (see page 64)
COMMIT Statement (see page 66)

Overview

You can direct processing in a DDDL compiler session using the statements shown in the following table.

Statement	What it does
SIGNON	Begins an online session or batch run of the DDDL compiler.
SIGNOFF	Ends an online session or batch run of the DDDL compiler.
SET OPTIONS	Supplies the default processing options for the current dictionary or DDDL compiler session.
DISPLAY/PUNCH OPTIONS	Supplies display/punch defaults for the current dictionary or DDDL compiler session.
INCLUDE	Retrieves the source statements associated with an IDD module.
COMMIT	Writes a checkpoint to the journal file.

These statements are described in this chapter.

SIGNON Statement

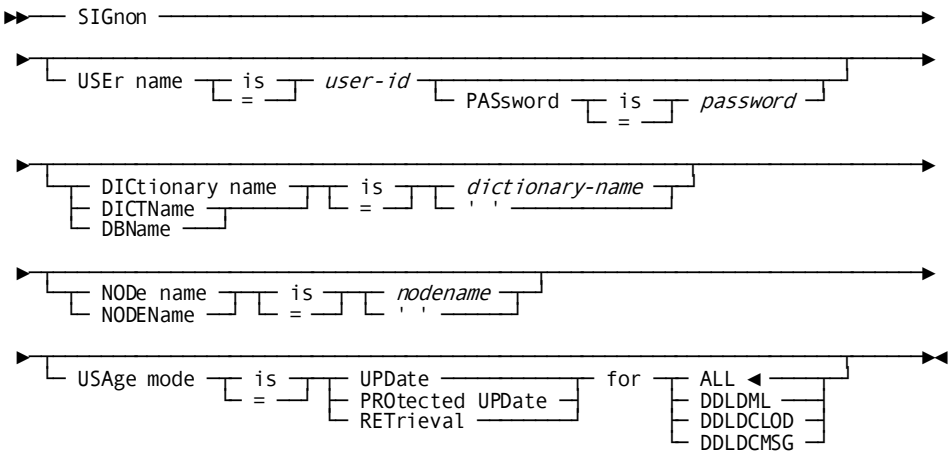
The SIGNON statement permits users to identify themselves to the DDDL compiler and to describe the environment in which the compiler is to execute.

If IDD SECURITY is ON in the dictionary, you must already be assigned the IDD authority through the AUTHORITY clause of the USER statement (see USER (see page 358)).

Note: You can also prevent unauthorized access to the dictionary using the central security facility. For information on the central security facility, see *CA IDMS Security Administration Guide*.

Syntax

SIGNON Statement



Parameters

USEr name is *user-id*

Specifies the ID of the user signing on to the DDDL compiler. If the SECURITY clause of the SET OPTIONS statement specifies that security for IDD is on, *user-id* must be the ID of a user authorized (in the USER clause) for DDDL compiler access. *User-id* must be a 1- to 32-character value and must be enclosed in quotation marks if it contains embedded blanks or delimiters.

PASsword is *password*

Specifies the password of the user signing on to the DDDL compiler.

DICtionary name is *dictionary-name*

Specifies the dictionary to be accessed by the DDDL compiler. If *dictionary-name* is blanks enclosed by quotes, it indicates the default dictionary for the local mode runtime environment or the target node if running under the central version.

NODe name is *nodename*

Specifies the name of the node that controls the dictionary to be accessed. *Nodename* identifies a node in the network. If *nodename* is blanks enclosed in quotes, it indicates the local node (the node at which the online compiler is executing or the DC/UCF system accessed by the batch compiler running under the central version).

USAge mode is

Specifies the manner in which the DDDL compiler can access dictionary areas. This clause overrides the usage mode defined during system generation by means of the IDD statement (see *CA IDMS System Generation Guide*).

UPDate

Specifies that the current user and all other users can update the dictionary concurrently. The DDDL compiler automatically prevents deadlock conditions or situations in which users must wait for commands issued by other users to be processed. This is the default, unless overridden during system generation, and is also the suggested usage mode for the DDDL compiler.

PROtected UPDate

Specifies that only the current user can update the dictionary. Other users are restricted to performing retrieval operations. During an online session, the current user has exclusive control for update only if the DDDL compiler has been invoked. Between terminal interactions, the areas can be updated by other users.

RETrieval

Specifies that the current user can only perform retrieval operations against the dictionary. This usage mode does not restrict other users from accessing the dictionary in update or protected update mode.

FOR ALL

Indicates that the usage mode applies to all areas. ALL is the default.

FOR DDLDML

Indicates that the usage mode applies only to the DDLDML area.

FOR DDLDCLOD

Indicates that the usage mode applies only to the DDLDCLOD area.

FOR DDLDCMSG

Indicates that the usage mode applies only to the DDLDCMSG area.

Usage

When to specify USER and PASSWORD in SIGNON

If *you are identified to the environment in which the compiler is executing and you do not hold the necessary authorities to perform the intended actions*, you must use the USER clause of SIGNON. In this case, you would specify the ID of a user who holds the necessary authorities (providing USER SIGNON OVERRIDE IS ALLOWED is specified in the SET OPTIONS statement). If the user ID you specify has been assigned a password in the dictionary being accessed, you must also supply that password in the SIGNON statement.

If *you are not identified to the execution environment and IDD SECURITY is ON*, you must use the USER parameter of SIGNON. In this case, the user ID and password you specify are verified by the central security facility. If verified, you will be known to both the execution environment and the compiler. The user ID must hold the appropriate IDD authority in the dictionary you are accessing as well as the authority to sign on to the DC/UCF system (if you are executing online). If the user ID you specify has been assigned a password in the central security facility, that password must be specified in the SIGNON statement.

In all other cases, the USER parameter is not required and should not be specified.

Note: For more information on the central security facility, refer to the *CA IDMS Security Administration Guide* document.

Identifying the dictionary to be accessed

The DICTIONARY and NODENAME clauses together identify the dictionary to be accessed by the compiler. If only one is specified, the other is derived.

Dictionary-name, if specified, must identify a DBNAME or segment accessible at the target node or local mode runtime environment. If *dictionary-name* is not specified, but *nodename* is specified, then the dictionary is the default dictionary at the specified node.

In local mode, *nodename* has no meaning and is ignored. When running under the central version, *nodename*, if specified, identifies the node at which the target dictionary resides. If not specified, the location of the dictionary is determined from the resource table associated with the local DC/UCF system.

If neither *dictionary-name* nor *nodename* is specified, they will be established from:

- The TCF specification, if running under TCF (for more information, refer to *CA IDMS Common Facilities Guide*)
- Session attributes as established by DCUF, SYSIDMS, system or user profiles
- The default dictionary associated with the local runtime environment

Readying several areas

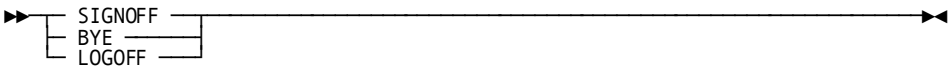
The USAGE MODE clause can be repeated to ready different areas in different usage modes. For example, to add or delete a load module from an area on a system in which the DDLDML area is available for retrieval only, specify USAGE MODE IS RETRIEVAL FOR DDLDML.

SIGNOFF Statement

When issued during an online session, SIGNOFF signs off the user from the DDDL compiler and deletes the default session options. In batch mode, the SIGNOFF statement terminates the DDDL compiler.

Syntax

SIGNOFF Statement



SET OPTIONS Statement

The SET OPTIONS statement controls DDDL compiler processing options. The user can supply default processing options for the current dictionary or for the current DDDL session. This section has the following information about the SET OPTIONS statement:

- Functions of the statement
- Syntax and parameter descriptions
- Default options and overrides
- Security considerations

SET OPTIONS Functions

The SET OPTIONS statement allows you to perform the functions shown in the following table.

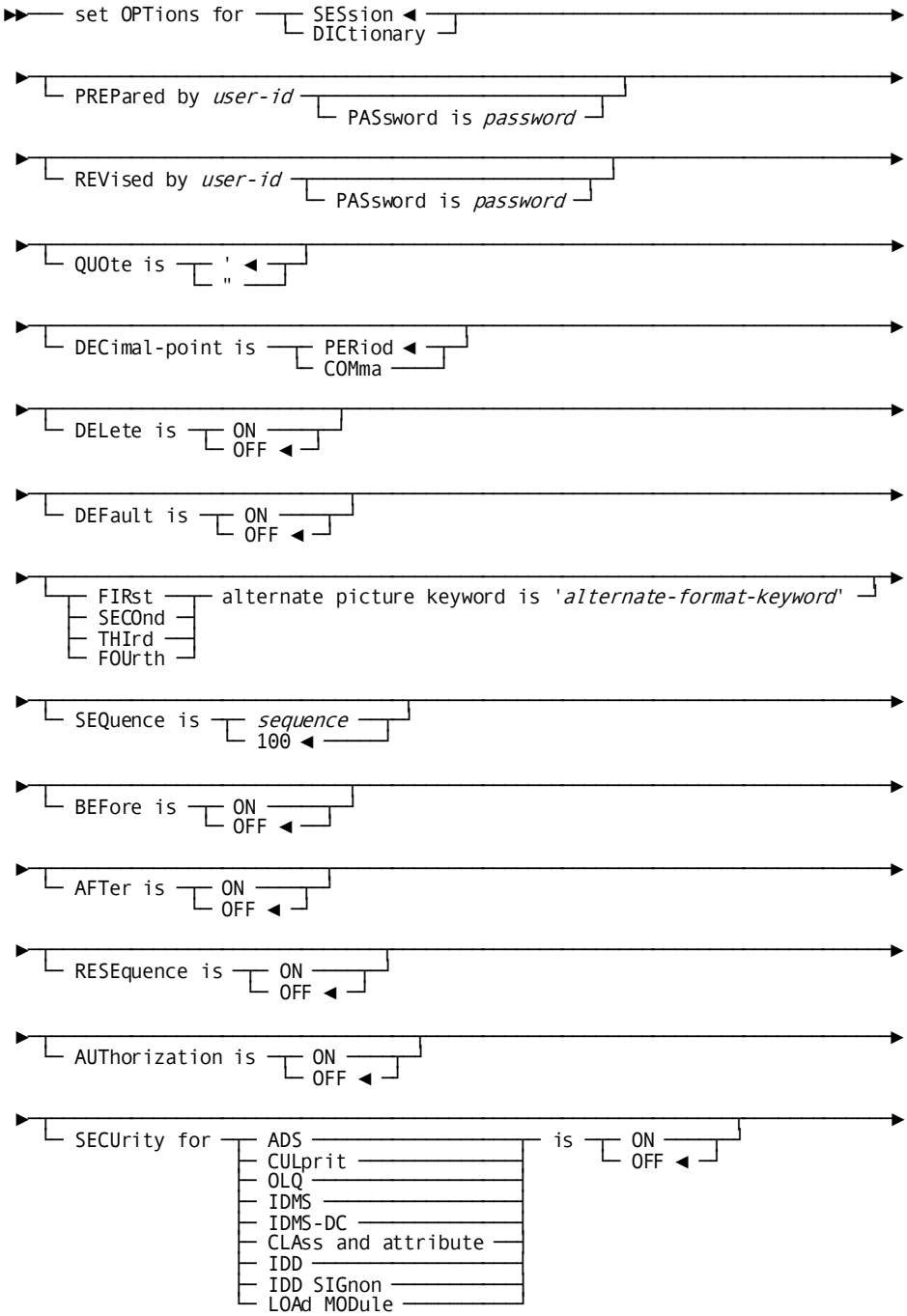
To do this	Use this SET OPTIONS clause
Specify dictionary security options	PREPARED BY REVISED BY SECURITY FOR PASSWORD SECURITY OVERRIDE REGISTRATION OVERRIDE USER SIGNON OVERRIDE
Set a site-standard quote character	QUOTE
Set a site-standard decimal point character	DECIMAL-POINT
Specify maintenance conventions for entity-occurrences	DELETE DEFAULT
Establish keywords that identify alternative formats(up to four) for element occurrences	ALTERNATE PICTURE KEYWORD
Direct the text editor of the batch DDDL compiler	SEQUENCE BEFORE AFTER RESEQUENCE

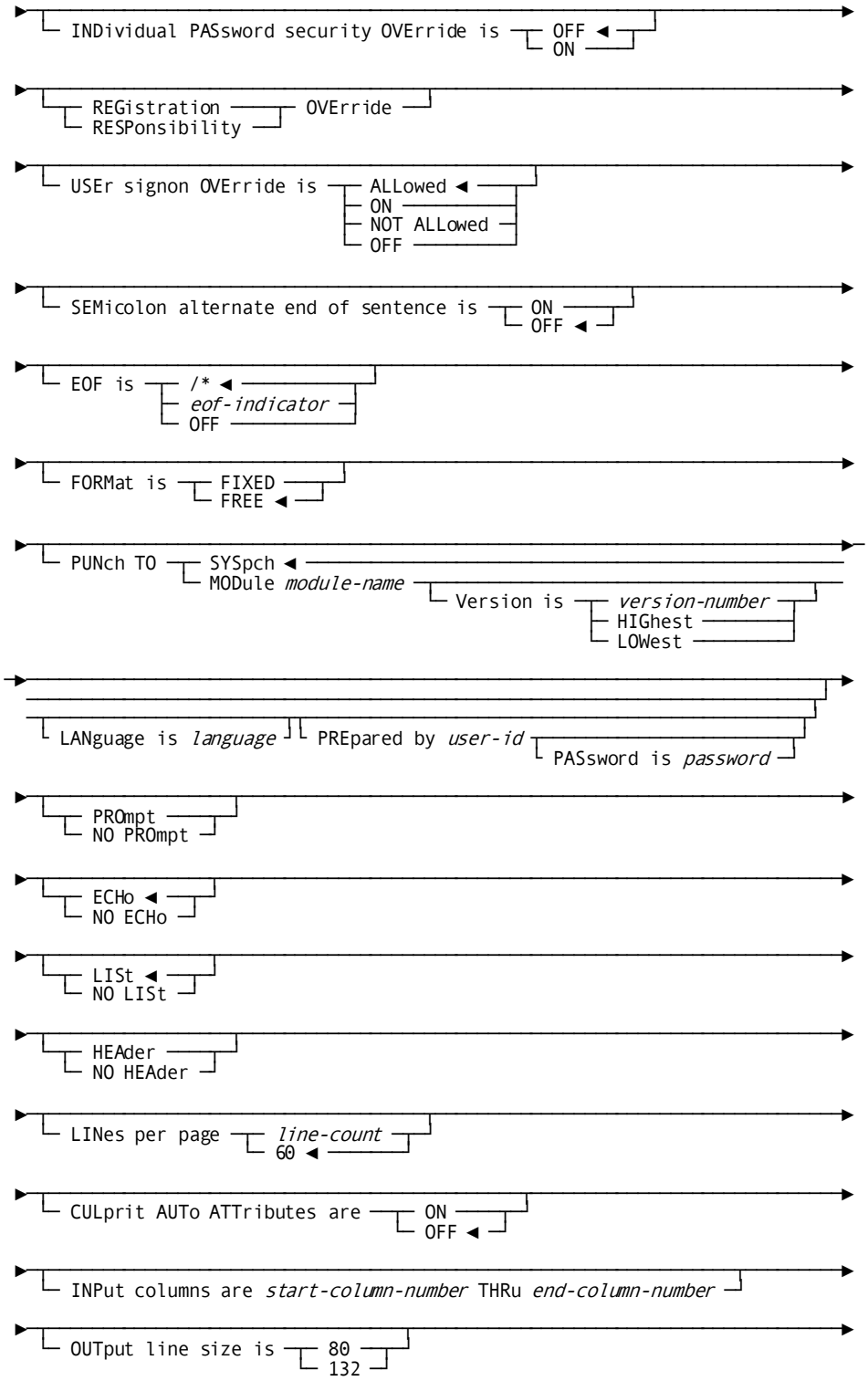
To do this	Use this SET OPTIONS clause
Set input and output formats for online and batch processing	PROMPT/NO PROMPT ECHO/NO ECHO LIST/NO LIST HEADER/NO HEADER LINES PER PAGE INPUT COLUMN OUTPUT LINE SIZE JCL CODE
Specify whether DML precompilers are to accept undefined programs at runtime	AUTHORIZATION IS ON/OFF
Specify whether CULPRIT is to copy file definitions from the dictionary at runtime	CULPRIT AUTO ATTRIBUTES
Specify default level numbers for elements that participate in record-element structures	LEVEL NUMBERS
Establish default version numbers	DEFAULT FOR NEW VERSION DEFAULT FOR EXISTING VERSION
Control DISPLAY/PUNCH output	PUNCH TO FORMAT DISPLAY ALL LIMIT INTERRUPT COUNT DISPLAY WITH/ALSO WITH/WITHOUT DISPLAY AS SYNTAX/COMMENTS DISPLAY VERB
Establish a default end-of-file indicator for use with the online DDDL compiler	EOF
Establish recognition of an alternative end-of-statement character	SEMICOLON ALTERNATE

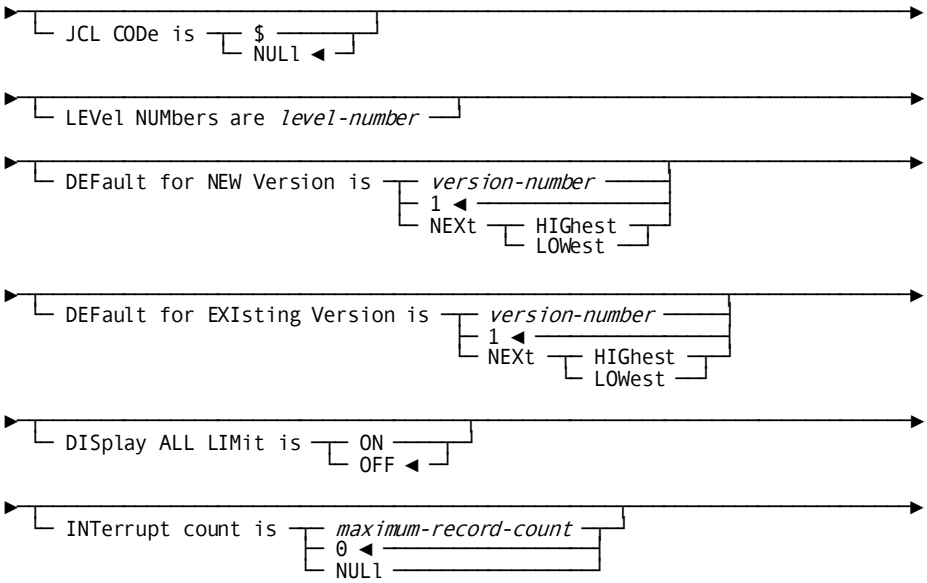
SET OPTIONS Syntax

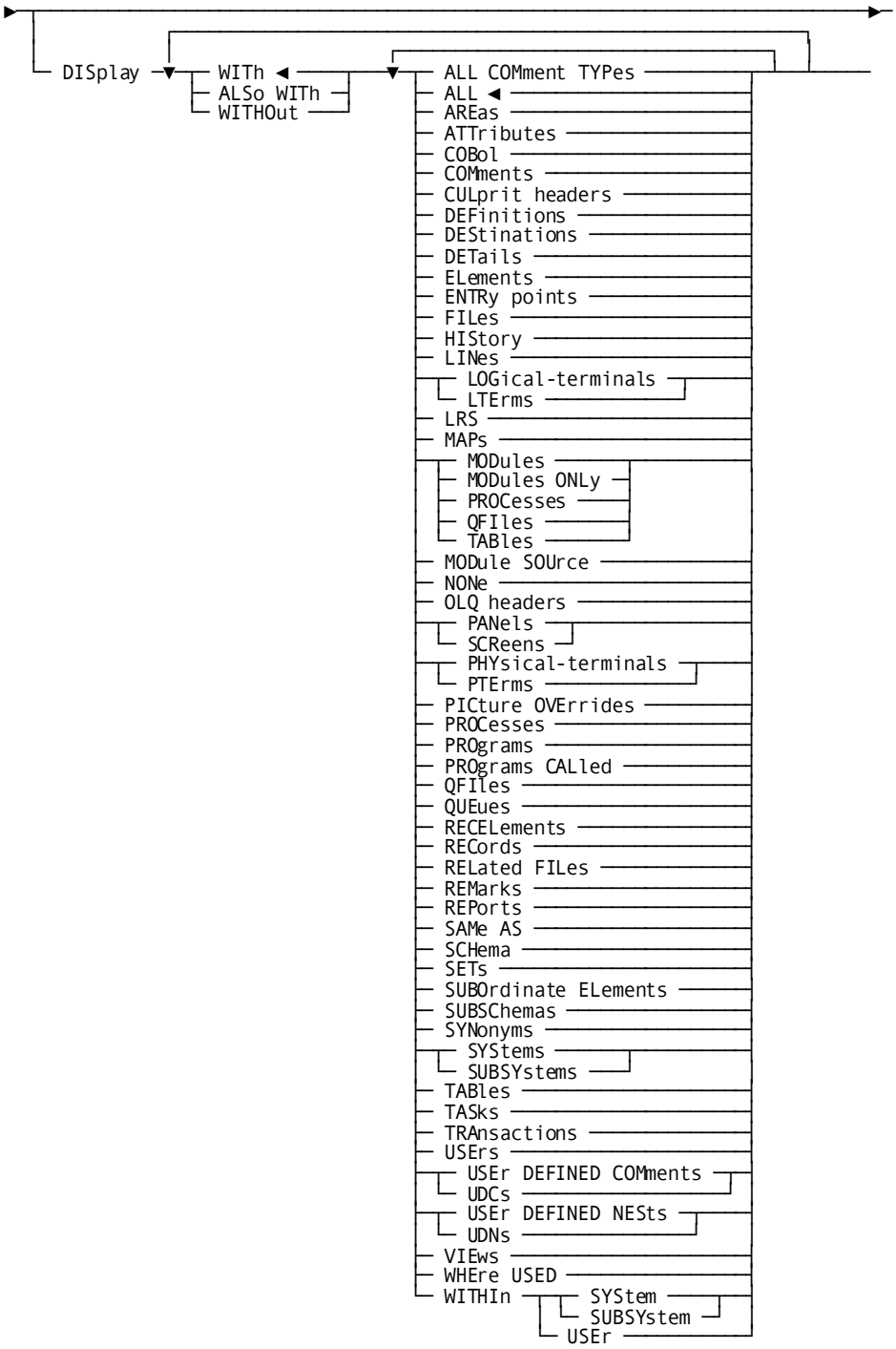
Syntax

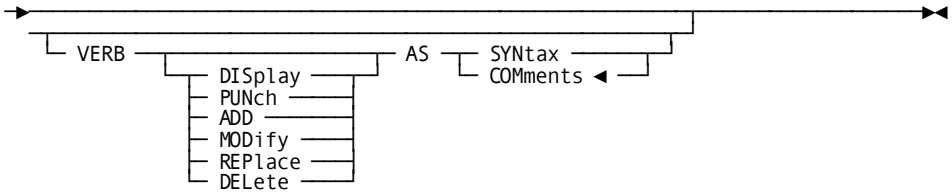
SET OPTIONS Statement




INDividual PASsword security OVErride is
OFF
ON
REGistration
RESPonsibility
OVErride
USEr signon OVErride is
ALLowed
ON
NOT ALLowed
OFF
SEMicolon alternate end of sentence is
ON
OFF
EOF is
/*
eof-indicator
OFF
FORMat is
FIXED
FREE
PUNch TO
SYSpch
MODule module-name
Version is
version-number
HIGhest
LOWest
LANguage is language
PREpared by user-id
PASsword is password
PROmpt
NO PROmpt
ECHo
NO ECHo
LISt
NO LISt
HEAder
NO HEAder
LINes per page
line-count
60
CULprit AUTo ATTributes are
ON
OFF
INPut columns are start-column-number THRu end-column-number
OUTput line size is
80
132


JCL CODe is
$
NULl
LEVel NUMbers are level-number
DEFault for NEW Version is
version-number
1
NEXt
HIGhest
LOWest
DEFault for EXIsting Version is
version-number
1
NEXt
HIGhest
LOWest
DISplay ALL LIMit is
ON
OFF
INTerrupt count is
maximum-record-count
0
NULl


DISplay
WITh
ALSo WITh
WITHOut
ALL COMment TYPes
ALL
AREas
ATTributes
COBol
COMments
CULprit headers
DEFinitions
DESTinations
DETails
ELements
ENTRy points
FILes
HIStory
LINes
LOGical-terminals
LTErms
LRS
MAPs
MODules
MODules ONLy
PROCesses
QFIles
TABles
MODule SOUrce
NONe
OLQ headers
PANels
SCReens
PHYsical-terminals
PTErms
PICture OVErrides
PROCesses
PROgrams
PROgrams CALled
QFIles
QUEues
RECELements
RECords
RELated FILes
REMarks
REPorts
SAMe AS
SCHema
SETs
SUBOrdinate ELements
SUBSChemas
SYNonyms
SYStems
SUBSYstems
TABles
TASks
TRAnsactions
USErs
USEr DEFINED COMments
UDCs
USEr DEFINED NESts
UDNs
VIEws
WHEre USED
WITHIn
SYStem
SUBSYstem
USEr



Parameters

set OPTions for DICtionary

Establishes new default processing options for all DDDL compiler sessions that access the current dictionary; installation defaults or defaults established with a previous SET OPTIONS FOR DICTIONARY statement are overridden. To issue SET OPTIONS FOR DICTIONARY statements, the user must be assigned AUTHORITY FOR UPDATE IS ALL (see USER (see page 358)). When the issuing user has not been assigned the proper authority, the PREPARED BY or REVISED BY clause must accompany this statement.

set OPTions for SESsion

Establishes temporary default processing options for a single DDDL compiler session. Any user can issue the SET OPTIONS FOR SESSION statement. However, some clauses require user authority (see SET OPTIONS Security (see page 62)). When the issuing user has not been assigned the proper authority, SET OPTIONS statements including such clauses must also include PREPARED BY or REVISED BY specifications.

PREpared by *user-id*

Establishes a default PREPARED BY specification for all entity-type statements issued in the current DDDL session and assigns the user authority to specify secured clauses of the SET OPTIONS statement. If this clause is not specified, the default PREPARED BY specification is the user ID supplied in SIGNON statement or the user ID known to the execution environment. PREPARED BY is ignored if NOT ALLOWED is specified in the USER SIGNON OVERRIDE clause. For more information about the PREPARED BY clause, see Securing the Dictionary (see page 72).

PASsword is *password*

Identifies the password assigned to the authorized user specified in the PREPARED BY clause.

REVised by *user-id*

Establishes a default REVISED BY specification for all entity-type statements issued in the current DDDL session and assigns the issuing user authority to specify secured clauses of the SET OPTIONS statement. If this clause is not specified, the default REVISED BY specification is the user ID supplied in the SIGNON statement or the user ID known to the execution environment. PREPARED BY is ignored if NOT ALLOWED is specified in the USER SIGNON OVERRIDE clause. For more information about the REVISED BY clause, see Securing the Dictionary (see page 72).

QUOte is '

Defines the site-standard quote as a single quotation mark (').

QUOte is ''

Defines the site-standard quote as a double quotation mark ('').

DECimal-point is

Specifies the site-standard decimal-point character to be used in ELEMENT, RECORD ELEMENT, and COBOL PICTURE and VALUE specifications.

PERiod

Establishes the period (.) as the default decimal-point character. The DDDL compiler interprets periods as decimal points.

COMma

Establishes the comma (,) as the default decimal-point character. The DDDL compiler interprets commas as insertion characters.

DELete is

Specifies whether elements are deleted or retained when the only record occurrence in which they participate is deleted.

ON

Deletes elements when the only record occurrence in which they participate is deleted. The DDDL compiler, however, does not automatically delete elements that participate in another record, have associated description text, have been modified and have a nonblank date-last-updated field, or are associated with users, attributes, ranges, other elements through nesting or any comment type.

OFF

Retains elements when the only record occurrence in which they participate is deleted. To delete such elements, the user must issue individual DELETE ELEMENT statements.

DEFault is

Specifies whether ADD statements that identify existing entity occurrences are accepted or rejected.

ON

Accepts ADD statements that identify existing entity occurrences. The DDDL compiler interprets such statements as MODIFY statements for the entity occurrence and issues the message ADD CHANGED TO MODIFY.

OFF

Rejects ADD statements that identify existing entity occurrences.

FIRst/SECond/THIrd/FOUrth ALTERNATE PICTURE KEYWORD IS '*alternate-format-keyword*'

Establishes up to four keywords that can be used within ELEMENT and RECORD statements to identify alternative formats for the object element or for all elements within the object record. *Alternate-format-keyword* is a 1 to 16 character user-defined keyword enclosed in quotation marks that characterizes the desired format (for example, 'NUMERIC EDITED' or 'ZONED DECIMAL'). To issue this clause, the user must be assigned AUTHORITY FOR UPDATE IS ALL.

SEQuence is *sequence*

Establishes the starting and increment values for line numbers associated with entries in record-element structures, comment text, and module source. *Sequence* must be a 1 to 5 digit integer.

BEFore is

Specifies whether text is to be printed before it is erased by an EDIT clause.

ON

Specifies that text to be erased or replaced by an EDIT clause instruction is to be printed before it is erased.

OFF

Specifies that text to be erased or replaced by an EDIT clause instruction is *not* to be printed before it is erased. The user can include a SHOW instruction within individual EDIT clauses to override the SET OPTIONS BEFORE value.

AFTer is

Specifies whether to print new text after it is inserted or replaced by an EDIT clause instruction.

ON

Specifies that the new text to be inserted or replaced by an EDIT clause instruction is to be printed after it is inserted or replaced.

OFF

Specifies that the new text to be inserted or replaced by an EDIT clause instruction is *not* to be printed after it is inserted or replaced.

The user can include a SHOW instruction within individual EDIT clauses to override the SET OPTIONS AFTER value.

RESEquence is

Specifies whether all instructions in an EDIT clause are resequenced after modification.

ON

Specifies that text to be modified by an EDIT clause instruction is to be resequenced after all instructions within the requested EDIT clause have been completed.

OFF

Specifies that text to be modified by an EDIT clause instruction is **not** to be resequenced after all instructions within the requested EDIT clause have been completed.

You can include a SEQUENCE instruction within individual EDIT clauses to override the SET OPTIONS RESEQUENCE value.

AUThorization is

Specifies guidelines for accepting or rejecting programs based on whether they are defined in the dictionary.

ON

Directs CA IDMS DMLO precompilers to accept only programs defined in the dictionary (those represented by occurrences of the PROGRAM entity type in the dictionary).

OFF

Directs CA IDMS DMLO precompilers to accept any program.

SECUrity for

Specifies whether security is to be enabled for IDD, CA IDMS, and CA IDMS/DC system entity types, the DDDL compiler, the CA IDMS schema compiler, the CA IDMS subschema compiler, and for CA ADS, CA Culprit, and CA OLQ operations. The SECURITY FOR clause is repeatable.

ADS

Specifies that only users with ADS authority can access CA ADS.

CULprit

Specifies that only users with CULPRIT authority can authorize other users to access files and subschemas to run CA Culprit reports. CA Culprit runs will access only authorized files and subschemas. CA Culprit security can also be used to restrict a user's ability to change record layouts and file definitions and to restrict access to DDR reports. To change record layouts and file definitions, the user must be assigned the CULPRIT OVERRIDES ARE ALLOWED option; to access DDR reports, the user must be assigned the CULPRIT OVERRIDES ARE ALLOWED option and must be authorized to access subschema IDMSNWKA of schema IDMSNTWK version 1. For more information about the CULPRIT OVERRIDES option, see USER (see page 358).

OLQ

Specifies that only users with CA OLQ authority can code USER statement clauses that pertain to CA OLQ. Additionally, if SECURITY FOR OLQ IS ON is specified, CA OLQ release 3.1 and later will enforce subschema and q-file restrictions. See the *CA OLQ Reference Guide* for further details.

IDMs

Specifies that only users with IDMS authority can register programs with subschemas and use the CA IDMS schema compiler and/or the CA IDMS subschema compiler.

IDMS-DC

Specifies that only users with IDMS-DC authority can access occurrences of the DESTINATION, LINE, LOGICAL-TERMINAL, MAP, MESSAGE, PANEL, PHYSICAL-TERMINAL, QUEUE, and TASK entity types.

CLAss and attribute

Specifies that only users with CLASS AND ATTRIBUTE authority can access occurrences of the ATTRIBUTE, CLASS, and user-defined entity types.

IDD

Specifies that only users with IDD authority can access occurrences of the ELEMENT, FILE, MODULE, QFILE, PROCESS, PROGRAM, RECORD, SYSTEM, TABLE, and USER entity types.

IDD SIGnon

Specifies that only users with IDD SIGNON authority can sign on to the DDDL compiler.

LOAd MODule

Specifies that only users with AUTHORITY FOR UPDATE IS LOAD MODULE can access a load module in the dictionary. To issue this clause, the user must be assigned AUTHORITY FOR UPDATE IS LOAD MODULE.

is ON

Specifies (as part of the SECURITY FOR clause) that user authorization is required to access (ADD, MODIFY, REPLACE, DELETE, DISPLAY, PUNCH) secured entity types or perform secured operations. If the authorized user has been assigned a password, that password must be supplied in the accompanying PREPARED BY/REVISED BY specification. User authority is established with the USER statement (see USER (see page 358)).

is OFF

Specifies (as part of the SECURITY FOR clause) that user authorization is not required to access entity types specified in the SECURITY FOR clause.

INDividual PASsword security OVErride is

Specifies whether users will be allowed to modify their own passwords.

OFF

Specifies that users cannot modify their own passwords unless they are assigned AUTHORITY FOR UPDATE IS PASSWORD and, if the SET OPTIONS statement specifies SECURITY FOR IDD IS ON, AUTHORITY FOR UPDATE IS IDD.

ON

Specifies that users can modify their own passwords. To issue this clause, the user must be assigned AUTHORITY FOR UPDATE IS ALL.

REGistration OVErride

Turns off entity-occurrence security for the DDDL compiler session. The user cannot revoke this security override for the duration of the session. To issue this clause, the user must be assigned AUTHORITY FOR UPDATE IS ALL. For a detailed discussion of entity-occurrence security, see Securing the Dictionary (see page 72). RESPONSIBILITY is a synonym for REGISTRATION.

SEr signon OVErride is

Indicates whether CA IDMS/DB will allow users to specify a different user ID in a SIGNON statement from the one known to the environment in which the compiler is executing (the DC/UCF system for online, the batch environment for batch).

ALLowed

Users may sign on to the compiler with a different user ID from the ID known to the execution environment and *user-specification* clauses may be used to override the default user ID. ALLOWED is the default. ON is a synonym for ALLOWED.

NOT ALLowed

CA IDMS/DB will not allow the user ID to be changed. Users who are already known to the environment cannot specify a different user ID in the SIGNON statement. Additionally, *user-specification* clauses cannot be used to change the default user ID. OFF is a synonym for NOT ALLOWED.

SEMicolon alternate end of sentence is

Indicates whether the semicolon is to be recognized as an alternative end-of-statement character.

ON

Specifies that both semicolons and periods are to be recognized as end-of-statement characters.

OFF

Specifies that the semicolon is *not* recognized as an alternative end-of-statement character. OFF is the default.

EOF is

Overrides the default logical end-of-file indicator established at IDD installation.

/*

Indicates the default logical end-of-file indicator.

eof-indicator

Specifies an end-of-file indicator.

OFF (online IDD only)

Specifies that there is no active end-of-file indicator.

FORMat is

Establishes the default format for DISPLAY/PUNCH verb output.

FIXED

Lists DISPLAY/PUNCH output in columnar format.

FREE

Lists DISPLAY/PUNCH output as running text.

PUNch TO

Specifies the default destination for PUNCH verb output.

SYSpch

Directs PUNCH verb output to the SYSPCH file. SYSPCH is the default destination established during IDD installation.

MODULE *module-name*

Directs PUNCH verb output to an IDD module. *Module-name* must be the 1- through 32-character name of a module that has been defined in the dictionary with a MODULE statement (see MODULE (PROCESS/QFILE/TABLE) (see page 219)). The following rules apply to the module named as the PUNCH verb destination:

- Once the module has been named as the destination of the PUNCH command, it cannot be modified, replaced, or deleted.
- A module cannot be punched to itself.

If module source code is already associated with the named module, the DDDL compiler adds the PUNCH verb output to the end of the existing source. If module source does not exist, the DDDL compiler generates a header, which contains the date and time that the initial punched output was created; the punched output follows this header.

LANguage is *language*

Specifies a language to be associated with the named module.

PREpared by *user-id*

Identifies the user who defined the module.

PASsword is '*password*'

Specifies the password of the identified user; mandatory if a password is associated with the user.

PROmpt

Specifies use of the word ENTER to prompt users for input. The PROMPT option is useful for local TSO z/VM operations or with dial-up devices.

NO PROmpt

Specifies no user prompt.

ECHo

Redisplays each input line the compiler reads. This is useful when DDDL statements are input one line at a time (for example, under TSO or z/VM, or from a hard-copy terminal).

NO ECHo

Specifies no redisplay of input lines even if the line contains an error. Suppresses execution of the EJECT and SKIP carriage control statements.

LISt

Redisplays each line read by the compiler.

NO LISt

Specifies no redisplay of input lines unless a line contains errors. Suppresses execution of the EJECT and SKIP carriage control statements.

HEAder

Specifies that the header lines that identify the DDDL compiler are to be printed on the IDD Activity List.

NO HEAder

Specifies that the header lines that identify the DDDL compiler are *not* to be printed on the IDD Activity List. This is useful when DDDL statements are input one line at a time (for example, under TSO or z/VM, or from a hard-copy terminal).

LINes per page *line-count*

Specifies the number of lines per page as a SET OPTIONS FOR SESSION option. The acceptable range for *line-count* is 10 through 60.

CULprit AUTo ATTributes are

For CA Culprit users only, this parameter determines whether file definitions are copied from the dictionary by CA Culprit at runtime.

ON

Specifies that the file description, including such information as block size, record size, recording mode, file and device types, or input module name, is to be copied from the dictionary at runtime.

OFF

Specifies that the file description is *not* copied from the dictionary at runtime.

INPut columns are *start-column-number* THRu *end-column-number*

Specifies the starting and ending columns for DDDL compiler input (and output with the exception of error messages). The maximum input column range is 1 through 80 for batch and line mode and 1 through 79 for full-screen mode. The default column range established at installation is 1 through 72 for batch mode, 1 through 80 for line mode, and 1 through 79 for full-screen mode. The continuation character (+) need not be coded in column 1; it can appear anywhere, provided that it is the first entry on the line. The user can select any value within the allowable range for *start-column-number* and *end-column-number*; the minimum number of characters allowed between low and high columns is ten.

OUTput line size is

Specifies an output line size for error messages. The line size for all other DDDL output is determined by the INPUT COLUMNS ARE clause.

80

Specifies an error message line size of 80 columns for the online compiler. The DDDL compiler does not list the line numbers of erroneous lines when it issues error messages; the error message, however, always appears on the line immediately below the erroneous line.

132

Specifies an error message line size of 132 columns for the batch compiler.

JCL CODe is

Specifies whether a dollar sign ($) in the first column of module source or EDIT clause input will be recognized as JCL or input data.

$

Specifies that a dollar sign in column one of module source or EDIT clause input is to be translated to a slash (/) when it is stored in the dictionary. Typically, this option is used to ensure that the operating system does not interpret input data as JCL.

NULl

Specifies that a dollar sign in column one is to be treated as input data; that is, the dollar sign is not translated to a slash when it is stored in the dictionary.

LEVel NUMbers are

Specifies the values to be associated with corresponding hierarchical depths in record-element structures.

level-number

Specifies a two-digit integer in the range 02 through 49; used with the LEVEL NUMBERS clause. In order to request a range of level numbers, the entire sequence of numbers must be explicitly coded; up to 48 level numbers can be specified in ascending order. If fewer than 48 level numbers are coded, 49 is the default. The LEVEL NUMBERS specification does not modify level numbers in existing record elements.

DEFault for NEW Version is

Establishes a default version number for the VERSION parameter of the NAME clause in ADD statements.

version-number

Specifies that the DDDL compiler is to assign a new entity occurrence the specified version number; *version-n* must be an integer in the range 1 through 9999.

NEXt HIGhest

Specifies that the DDDL compiler is to assign a new entity occurrence the highest version number associated with the specified entity-occurrence name, plus 1. If NEXT is the only keyword coded, NEXT HIGHEST is assumed. Because NEXT HIGHEST creates a new version, if the requested entity occurrence exists in the dictionary, the DDDL compiler *does not* issue the ADD CHANGED TO MODIFY message.

NEXt LOWest

Specifies that the DDDL compiler is to assign a new entity occurrence the lowest version number associated with the specified entity-occurrence name, minus 1. Because NEXT LOWEST creates a new version, if the requested entity occurrence exists in the dictionary, the DDDL compiler *does not* issue the ADD CHANGED TO MODIFY message.

DEFault for EXIsting Version is

Establishes a default version number to be used in any statement or clause that references an existing entity occurrence.

version-number

Specifies use of the specified version number; *version-number* must be an integer in the range 1 through 9999.

HIGhest

Specifies use of the highest version number associated with the specified entity-occurrence name.

LOWest

Specifies use of the lowest version number associated with the specified entity-occurrence name.

DISplay ALL LIMit is

Indicates whether the DDDL compiler will limit the number of records to be retrieved using a DISPLAY ALL statement.

ON

Specifies that the number of records retrieved by a DISPLAY ALL statement will be limited to the number of records specified in the INTERRUPT COUNT clause.

OFF

Specifies that the number of records retrieved by a DISPLAY ALL statement is *not* limited. OFF is the default.

INTerrupt count is

Specifies the maximum number of records to be retrieved using a DISPLAY ALL statement when the DISPLAY ALL LIMIT is ON.

maximum-record-count

Specifies a maximum number for INTERRUPT COUNT. *Maximum-record-count* can be any number, including 0.

NULl

Sets the *maximum-record-count* to 0.

DISplay

Supplies default values for DISPLAY/PUNCH clauses. This clause is positional; it must be coded as the last clause in a SET OPTIONS statement. You can select one or more entity options for display, but you cannot repeat an option.

WITh

Lists the requested information. All options specified in previously issued DISPLAY WITH and DISPLAY ALSO WITH statements are replaced.

ALSo WITh

Lists the requested information in addition to any information requested in previously issued DISPLAY WITH and DISPLAY ALSO WITH statements.

WITHOut

Excludes the specified information from the information requested in previously issued DISPLAY WITH and DISPLAY ALSO WITH statements.

ALL

Specifies the display of all of the information associated with the requested entity occurrence. ALL is the default.

ALL COMment TYPes

Specifies the display of all comment entries (COMMENTS, DEFINITIONS, ELEMENT DEFINITIONS, CULPRIT HEADERS, OLQ HEADERS, REMARKS, and user-defined comment keys) associated with the requested entity occurrence.

AREas

Specifies the display of all database areas associated with the requested entity occurrence.

ATTributes

Specifies the display of all attributes associated with the requested entity occurrence.

COBol

Specifies the display of record elements associated with the requested record occurrence displayed in COBOL format. This parameter applies only to RECORD entities.

COMments

Specifies the display of all comments associated with the requested entity occurrence.

CULprit headers

Specifies the display of all RIT headers associated with the requested record element. This parameter applies to record elements only.

DEFinitions

Specifies the display of all definitions associated with the requested entity occurrence.

DEStinations

Specifies the display of all destinations associated with the requested entity occurrence.

DETails

Specifies the display of entity-specific descriptions; for example, the length of a record or the block size of a file.

ELements

Specifies the display of all elements associated with the requested entity occurrence.

ENTRy points

Specifies the display of all entry points associated with the requested entity occurrence.

FILes

Specifies the display of all files associated with the requested entity occurrence.

HIStory

Specifies the display of the chronological account of an entity's existence, including PREPARED/REVISED BY specifications, date created, and date last updated. For programs, HISTORY also includes the number of times the program has been compiled and the date of the last compilation.

LINes

Specifies the display of all lines associated with the requested entity occurrence.

LOGical-terminals

Specifies the display of all logical terminals associated with the requested entity occurrence.

LRS

Specifies the display of all of the logical records associated with the requested program. This parameter only applies to PROGRAM entities.

MAPS

Specifies the display of all maps associated with the requested entity occurrence.

MODules

Specifies the display of all modules, processes, q-files, or tables associated with the requested entity occurrence.

MODules ONLy

Optionally limits the list to modules with a language specification other than PROCESS, OLQ, or TABLE.

MODule SOUrce

Specifies the display of the source statements associated with the requested module, process, or q-file. This parameter only applies to MODULE, PROCESS, and QFILE entities.

NONE

Specifies the display of the name of the requested entity occurrence. NONE is meaningful only when WITH is specified.

OLQ headers

Specifies the display of all CA OLQ headers associated with the requested record element. This parameter applies to record elements only.

PANels

Specifies the display of all panels (screens) associated with the requested entity occurrence. SCREENS is a synonym for PANELS.

PHYsical-terminals

Specifies the display of all physical terminals associated with the requested entity occurrence. PTERMS is a synonym for physical-terminals.

PICture OVErrides

Specifies the display of the PICTURE, USAGE, VALUE, JUSTIFY, SIGN, BLANK WHEN ZERO, LINE IS, SUBORDINATE ELEMENT REDEFINES, and SUBORDINATE ELEMENT OCCURS specifications associated with the requested record element. This parameter only applies to RECORD entities.

PROCesses

Specifies the display of all processes associated with the requested entity occurrence.

PROgrams

Specifies the display of all programs associated with the requested entity occurrence.

PROgrams CALled

Specifies the display of all of the subprograms associated with the requested program. This parameter applies to PROGRAM entities only.

QFIles

Specifies the display of all q-files associated with the requested entity occurrence.

QUEues

Specifies the display of all queues associated with the requested entity occurrence.

RECELems

Specifies the display of record detail information and all record elements associated with the requested record occurrence. This parameter applies only to RECORD entities.

RECords

Specifies the display of all records associated with the requested entity occurrence.

RELated FILes

Specifies the display of all of the relationships created with the RELATED FILES ARE clause of the FILE statement for the requested file.

REMarks

Specifies the display of all remarks associated with the requested program. This parameter applies to PROGRAM entities only.

REPorts

Specifies the display of all reports associated with the requested entity occurrence.

SAMe AS

Specifies the display of all of the relationships that exist between the entities that are the source and target of a SAME AS clause; for information on the SAME AS clause, see Copying and Editing Entity Occurrences (see page 88).

SCHemas

Specifies the display of all schemas associated with the requested entity occurrence.

SETs

Specifies the display of all sets associated with the requested entity occurrence.

SUBOrdinate ELements

Specifies the display of all of the subordinate elements associated with the requested group element. This parameter applies to ELEMENT entities only.

SUBSChemas

Specifies the display of all subschemas associated with the requested entity occurrence.

SYNonyms

Specifies the display of all of the synonyms associated with the requested entity occurrence. If the requested entity occurrence is an attribute, record synonyms associated with that attribute apply. This parameter only applies to ELEMENT, FILE, and RECORD entities.

SYStems

Specifies the display of all systems or subsystems associated with the requested entity occurrence. SYSTEMS and SUBSYSTEMS are synonyms.

TABles

Specifies the display of all tables associated with the requested entity occurrence.

TASks

Specifies the display of all tasks associated with the requested entity occurrence.

TRAnsactions

Specifies the display of all transactions associated with the requested entity occurrence.

USErs

Specifies the display of all users associated with the requested entity occurrence.

USEr DEFINED COMments

Specifies the display of all user-defined comment keys associated with the requested entity occurrence. This parameter only applies to entities with user-defined comment keys. UDCS is a synonym for USER DEFINED COMMENTS.

USEr DEFINED NESts

Specifies the display of all of the user-defined nests associated with the requested entity occurrence. This parameter only applies to entities with relational keys. UDNS is a synonym for USER DEFINED NESTS.

WHEre USED

Specifies the display of all relationships in which the requested entity occurrence participates as a subordinate element, program called, related file, attribute, user-defined nest, or user or system within another user or system.

This parameter only applies to ATTRIBUTE, ELEMENT, FILE, MODULE, PROGRAM, RECORD, SYSTEM, USER, and user-defined entities.

WITHin SYStem

Specifies the display of all system (subsystem) occurrences related to the requested system/subsystem by means of the WITHIN SYSTEM clause. This parameter applies to SYSTEM/SUBSYSTEM entities only. SUBSYSTEM and SYSTEM are synonyms and can be used interchangeably.

WITHIn USEr

Specifies the display of all users with whom the requested user has been related by means of the WITHIN USER clause. This parameter applies to USER entities only.

VERB DISplay/PUNch/ADD/MODify/REPlace/DELete

Specifies the default verb to accompany DISPLAY/PUNCH verb output.

AS SYNtax

Specifies that information listed in response to a DISPLAY/PUNCH request appears as DDDL syntax. By displaying entity-occurrence definitions as syntax, the user can edit existing definitions and resubmit them to the DDDL compiler.

AS COMments

Specifies that DISPLAY/PUNCH output appears as comments (which are ignored by the DDDL compiler). Each line is preceded by an asterisk and a plus sign (*+) in the first two columns.

Example

The following SET OPTIONS statement establishes default processing options for one session. Specifications are given that instruct the DDDL compiler to list DISPLAY/PUNCH output in syntax format and accept input in columns 2 through 65.

```
set options for session
   input columns are 2 thru 65
   display as syntax.
```

Usage

Order of SET OPTIONS parameters

The parameters of the SET OPTIONS statement can be coded in any order, with the exception of the DISPLAY clause, which must appear as the last clause in a SET OPTIONS statement.

Considerations for alternate picture keywords

Subsequently issued SET OPTIONS statements can change existing alternate picture keywords. Note, however, that all elements and record elements that have been assigned alternative formats retain those formats unless the element or record-element definition is explicitly changed. For example, if an element definition specifies PICTURE IS 'NUMERIC EDITED', the format remains unchanged, regardless of whether a SET OPTIONS FIRST ALTERNATE PICTURE KEYWORD statement establishes a new keyword.

DELETE IS ON usage and cautions

DELETE IS ON provides a convenient means of deleting record elements that have been added to the dictionary with the COBOL substatement. Because the COBOL substatement automatically associates elements with records, elements associated with deleted records need not be maintained.

More information: For a detailed description of the COBOL substatement, see RECORD (REPORT/TRANSACTION) (see page 287).

To avoid the inadvertent deletion of elements, select DELETE IS ON only on an as-needed basis; immediately thereafter, specify DELETE IS OFF.

Overriding PREPARED BY and REVISED BY clauses

You can override the default PREPARED BY specification by including a PREPARED BY clause in individual ADD, MODIFY, REPLACE, DELETE, and DISPLAY/PUNCH statements.

You can override the default REVISED BY specification by including a REVISED BY clause in individual ADD, MODIFY, REPLACE, DELETE, and DISPLAY/PUNCH statements.

Overrides to PREPARED BY and REVISED BY clauses are recognized only if ALLOWED is specified in the USER SIGNON OVERRIDE clause.

Overriding SEQUENCE values

The user can include the following instructions within individual EDIT clauses to override the SET OPTIONS SEQUENCE value:

- SEQUENCE overrides sequence default for all EDIT clause instructions.
- INCREMENT BY in an INSERT or REPLACE instruction overrides, for that instruction only, the default established by the SET OPTIONS statement or by the SEQUENCE instruction.

Overriding the default PUNCH destination

The user can include a TO SYSPCH/MODULE clause within individual PUNCH statements to override the default PUNCH destination.

WHERE USED guidelines

WHERE USED must accompany a request to display SUBORDINATE ELEMENTS, PROGRAMS CALLED, RELATED FILES, ATTRIBUTES, USER-DEFINED NESTS, or WITHIN USER or WITHIN SYSTEM.

If WHERE USED is not specified, a request for SUBORDINATE ELEMENTS, PROGRAMS CALLED, RELATED FILES, ATTRIBUTES, USER-DEFINED NESTS, or WITHIN USER or WITHIN SYSTEM displays the programs called by, elements subordinate to, files, attributes, and user-defined nests related to, and users and systems within the requested entity occurrence.

For example, DISPLAY PROGRAM PAYROLL WITH PROGRAMS CALLED lists the programs called by the program named PAYROLL; DISPLAY PROGRAM PAYROLL WITH PROGRAMS CALLED WHERE USED lists the programs that call the program PAYROLL.

Overriding DISPLAY/PUNCH options

You can include these clauses in individual DISPLAY/PUNCH statements to override the specified options:

- WITH/ALSO WITH/WITHOUT
- VERB
- AS SYNTAX/COMMENTS

SET OPTIONS Defaults and Overrides

The IDD installation procedure establishes defaults for most of the DDDL compiler processing options. These defaults remain in effect until they are explicitly changed for one of the following:

- **Dictionary**—you can specify a SET OPTIONS FOR DICTIONARY statement to establish default options for all DDDL compiler sessions that access the current dictionary.

  Note: Any option you can specify under SET OPTIONS FOR DICTIONARY you can also specify under SET OPTIONS FOR SESSION.

- **Single session**—you can specify a SET OPTIONS FOR SESSION statement to establish default options for the current DDDL session only.

  Note: If you try to use SET OPTIONS FOR DICTIONARY when only SET OPTIONS FOR SESSION is permitted, IDD applies the option to your current session.

- **Single statement**—you can specify optional clauses in an entity-type statement to establish processing options for that statement only.

The following table lists DDDL compiler processing options, their installation default values, and the ways you can change the defaults. Installation defaults are for batch and online processing, unless otherwise noted.

SET OPTIONS clause	Default	Overrides possible for:		
		Dictionary	Session	Statement
AFTER IS	OFF	X	X	X
AUTHORIZATION IS	OFF	X		
BEFORE IS	OFF	X	X	X
CULPRIT AUTO ATTRIBUTES ARE	OFF	X		
DECIMAL- POINT IS PERIOD/COMMA	PERIOD	X	X	
DEFAULT FOR EXISTING VERSION IS	1	X	X	X
DEFAULT FOR NEW VERSION IS	1	X	X	X
DEFAULT IS	OFF	X	X	
DELETE IS	OFF	X	X	
DISPLAY ALL LIMIT	OFF	X	X	

DISPLAY AS SYNTAX/ COMMENTS	COMMENTS		X	X
DISPLAY WITH/ ALSO WITH/ WITHOUT	WITH ALL		X	X
DISPLAY VERB	ADD		X	X
ECHO/NO ECHO	ECHO		X	
EOF IS	/* (batch)	X	X	
FIRST/SECOND/ THIRD/FOURTH ALTERNATE PICTURE KEYWORD	No defaults	X	X	
FORMAT IS FIXED/FREE	FREE		X	X
HEADER/ NO HEADER	HEADER (batch) NO HEADER (online)		X	
INPUT COLUMNS ARE	1-72 (batch) 1-80 (online)		X	
INTERRUPT COUNT IS	0	X	X	X
JCL CODE IS	NULL		X	
LEVEL NUMBERS ARE	02-49	X	X	
LINES PER PAGE	60		X	
LIST/NO LIST	LIST		X	
OUTPUT LINE SIZE IS	132 (batch) 80 (online)		X	
PASSWORD SECURITY OVERRIDE	OFF	X		
PREPARED BY/ REVISED BY	No defaults		X	X

PROMPT/ NO PROMPT	NO PROMPT (batch) NO PROMPT (online; 3270) PROMPT (online; line mode)		X	
PUNCH TO	SYSPCH		X	X
QUOTE IS	'	X	X	
RESEQUENCE IS	OFF	X	X	
REGISTRATION OVERRIDE	No default		X	
SECURITY FOR	OFF	X	X	
SEMICOLON ALTERNATE	OFF	X	X	
SEQUENCE IS	100	X	X	X
USER SIGNON OVERRIDE	ON	X	X	

Overriding Clauses

The following DDDL clauses override defaults established by the SET OPTIONS FOR SESSION statement:

- The PREPARED/REVISED BY clause of ADD/MODIFY/REPLACE/DELETE/DISPLAY/PUNCH
- The SHOW instruction of the EDIT clause
- The SEQUENCE instruction of the EDIT clause
- The INCREMENT BY parameter of the INSERT and REPLACE instruction of the EDIT clause
- The WITH/ALSO WITH/WITHOUT clause of DISPLAY/PUNCH
- The VERB clause of DISPLAY/PUNCH
- The AS SYNTAX/COMMENTS clause of DISPLAY/PUNCH
- The TO clause of PUNCH
- The VERSION clause of ADD/MODIFY/REPLACE/DELETE/DISPLAY/PUNCH

SET OPTIONS Security

You must have explicit update authority to submit certain SET OPTIONS clauses. The following table lists the applicable clauses and the required authority.

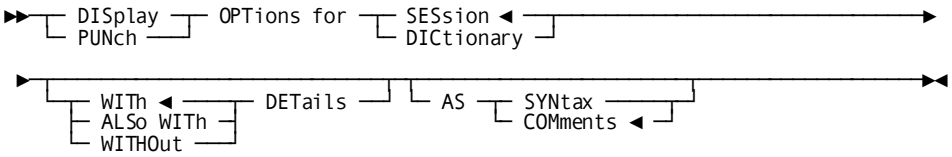
SET OPTIONS clause	Required UPDATE authority
SET OPTIONS FOR DICTIONARY	ALL
FIRST/SECOND/THIRD/FOURTH ALTERNATE PICTURE KEYWORD	ALL
CULPRIT AUTO ATTRIBUTES	CULPRIT
REGISTRATION OVERRIDE	ALL
PASSWORD SECURITY OVERRIDE	ALL
SECURITY FOR ADS	ALL
SECURITY FOR CLASS AND ATTRIBUTE	CLASS AND ATTRIBUTE
SECURITY FOR CULPRIT	CULPRIT
SECURITY FOR IDD	IDD
SECURITY FOR IDD SIGNON	IDD SIGNON
SECURITY FOR IDMS	IDMS
SECURITY FOR IDMS-DC	DC
SECURITY FOR LOAD MODULE	ALL
SECURITY FOR OLQ	OLQ
USER SIGNON OVERRIDE	ALL
DISPLAY ALL LIMITS	ALL
INTERRUPT COUNT	ALL

DISPLAY/PUNCH OPTIONS Statement

The DISPLAY/PUNCH OPTIONS statement lists, as a SET OPTIONS statement, the default processing options in effect for the current DDDL compiler session or dictionary.

Syntax

DISPLAY/PUNCH OPTIONS Statement



Parameters

DISplay/PUNch OPTions for

Lists the default processing options established with the SET OPTIONS statement or during IDD installation.

SESsion

Specifies that the options in effect for the current DDDL session are listed along with signon information, including user name, dictionary name, node name, and usage mode. This is the default.

DICtionary

Specifies that the options in effect for the current dictionary are listed along with the date the dictionary was created and the date of its most recent update.

WITh DETails

Specifies that all default processing options in effect for the session or for all sessions are listed.

ALSo WITh DETails

Specifies that all default processing options in effect for the session or for all sessions are listed.

WITHOut DETails

Specifies that only the statement SET OPTIONS FOR SESSION/DICTIONARY is listed.

AS SYNtax

Specifies that DISPLAY/PUNCH OPTIONS output is to be formatted as syntax (meaning that you can edit and resubmit the statements).

AS COMments

Specifies that DISPLAY/PUNCH OPTIONS output is to be formatted as comments (meaning that the DDDL compiler ignores the statements).

Example

The following example illustrates the output associated with a DISPLAY OPTIONS statement.

```
display options for session.
*+   set options for session
*+       dictionary name is prod
*+       usage mode is update
*+       default for existing version is 1
*+       quote is '
*+       eof is '/*'
*+       default is off
*+       sequence is 100
*+       no prompt
*+       echo
*+       list
*+       header
*+       input columns are 1 thru 80
*+       output line size is 80
```

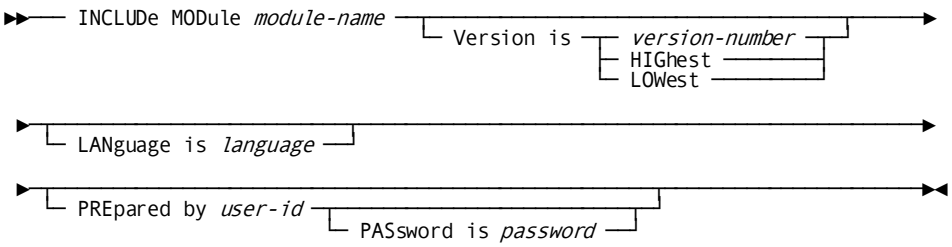
INCLUDE Statement

The INCLUDE statement temporarily suspends input to the batch or online DDDL compiler and retrieves, as input to the compiler, source statements associated with an existing IDD module. Modules are defined in the dictionary using the MODULE statement (see MODULE (PROCESS/QFILE/TABLE) (see page 219)). The module source can contain any number of DDDL statements.

When all the module source has been included, the DDDL compiler continues processing with the source statement immediately following the INCLUDE statement.

Syntax

INCLUDE Statement



Parameters

INCLUDe MODule *module-name*

Specifies that the DDDL compiler is to include in the current input file the source statements associated with the named module. *Module-name* must be the name of an existing IDD module.

Version is

Qualifies nonunique module names.

version-number

Specifies a specific version number for the module.

HIGhest

Specifies that the DDDL compiler is to use the highest version number associated with the specified module.

LOWest

Specifies that the DDDL compiler is to use the lowest version number associated with the specified module.

LANguage is *language-name*

Qualifies the module name by language. This parameter is required if the module has been defined with a language in the dictionary.

PREpared by *user-name*

Specifies the name of the user requesting the INCLUDE operation. For a detailed description of the PREPARED BY clause, see Securing the Dictionary (see page 72).

PASsword is *password*

Specifies the password of the user requesting the INCLUDE operation.

Usage

Restrictions on INCLUDE

The following restrictions apply to the INCLUDE statement:

- INCLUDE statements cannot appear within the module source; that is, INCLUDE statements cannot be nested.
- The requested module cannot update its own module source.

If the module source being included contains a SIGNON statement to another dictionary, the DDDL compiler terminates the INCLUDE operation and continues processing with the statement immediately following the INCLUDE.

Example

The following example shows a DDDL compiler session in which the user includes source statements associated with the module INCLUDE-TEST version 1 in the current DDDL input file. The definition of the module INCLUDE-TEST is shown.

```
add module include-test version 1
   prepared by wmc
   module source follows
      display all modules where name contains '-wmc'.
      signon dict=b
      modify user wmc.
      ...
   msend.
```

The sample session follows:

```
signon dict=a
include module include-test version 1.
display file xyz version 2.
 .
 .
 .
signoff
```

Because module INCLUDE-TEST contains a SIGNON statement, the DDDL compiler terminates the INCLUDE operation without executing the MODIFY USER WMC statement; processing continues with the DISPLAY FILE XYZ statement.

COMMIT Statement

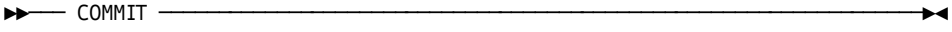
COMMIT is useful in the following situations:

- When journaling in local mode, COMMIT writes a COMT checkpoint to the journal file.
- Under the central version in batch mode, COMMIT releases update and exclusive locks.

COMMIT facilitates recovery during DDDL compiler runs that process large amounts of data. For more information about CA IDMS/DB backup and recovery procedures, see *CA IDMS Database Administration Guide*.

Syntax

COMMIT Statement



Chapter 4: General DDDL Syntax Options

This section contains the following topics:

Overview (see page 67)
Identifying Entity Occurrences (see page 68)
Securing the Dictionary (see page 72)
Documenting Entity Occurrences (see page 81)
Copying and Editing Entity Occurrences (see page 88)
Associating Entity Occurrences (see page 103)
Displaying Entity Occurrences (see page 112)

Overview

This chapter describes the DDDL syntax options that are common to all or many DDDL entity-type statements. Because these general options are functionally the same, regardless of the entity to which they apply, the rules presented below are not repeated in the detailed syntax presentations in Chapter 4.

Functions and associated syntax options are shown in the following table.

Function	Syntax option
Identify entity occurrences	NAME and VERSION clauses
Secure entity occurrences	PREPARED BY and REVISED BY clauses AUTHORITY clause (USER statement) USER clause PUBLIC ACCESS clause
Document entity-occurrences	DESCRIPTION clause COMMENTS clause TEXT clause
Manipulate entity-occurrences	SAME AS clause COPY clause EDIT clause
Establish or change entity-occurrence relationships	Relational keys Attribute/entity relationships
Display entity occurrences	DISPLAY/PUNCH statement

These syntax options are described in this chapter.

Identifying Entity Occurrences

Each entity occurrence in the dictionary must be unique. Specific qualifying clauses for each entity type allow you to make entity occurrences unique:

- All entity occurrences must be identified by *name* and, optionally, by *version number*.
- Some entity occurrences require (or allow) additional qualifiers.

The NAME and VERSION clauses and additional qualifiers are described separately in this section.

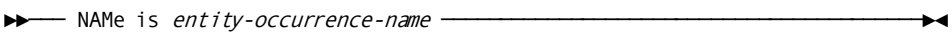
NAME Clause

Each entity-type statement must include a name that identifies the object entity occurrence in the dictionary. This name clause:

- Follows the verb and entity-type name
- Precedes all optional clauses

Syntax

NAME Clause



Parameters

NAMe is

Identifies either a new entity occurrence to be added to the dictionary or an existing entity occurrence to be modified, replaced, deleted, displayed, or punched.

entity-occurrence-name

Uniquely identifies the object entity in the dictionary; if the name includes embedded blanks or delimiters, it must be enclosed in site-standard quote characters. If the specified name is not unique, it must be qualified by a version number (and/or additional qualifier).

Note: For more information, see VERSION Clause (see page 69) and Additional Qualifiers (see page 71).

Example

In the following statement, the NAME clause assigns the name PAYROLL to an occurrence of the SYSTEM entity type.

```
add system name is payroll.
```

VERSION Clause

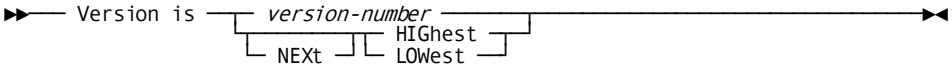
IDD supports the use of version numbers to accommodate entity occurrences that are identically named but different in usage or format. For example, when designing and testing a new application, the user can maintain several occurrences of the same entity by assigning a unique version number to each occurrence. When the final definition is approved, the user can retain the appropriate version in the dictionary, deleting all other versions.

Note: DDDL syntax does not support version number identification for the CLASS, ATTRIBUTE, ENTRY POINT, MESSAGE, USER, and user-defined entity types.

The VERSION clause permits the user to specify an explicit version number or the next higher or lower number. The user can also specify default version numbers for the current session or dictionary by using the SET OPTIONS DEFAULT FOR NEW VERSION and SET OPTIONS DEFAULT FOR EXISTING VERSION statements (see SET OPTIONS Syntax (see page 36)).

If a version number is not specified within an entity-type statement, the version number conventions described in Default Version Number Conventions, apply.

Syntax: VERSION Clause



Parameters

Version is *version-number*

Specifies a unique integer in the range 1 through 9999.

NEXt

When added to HIGHEST (NEXT HIGHEST), specifies the highest version number associated with the entity occurrence, plus 1. When added to LOWEST (NEXT LOWEST), specifies the lowest version number associated with the entity occurrence, minus 1. If *only* NEXT is specified, NEXT HIGHEST is assumed. This parameter is used only with the ADD statement or with the NEW NAME or NEW VERSION clauses.

HIGhest

Specifies the highest version number associated with the object entity occurrence. If NEXT HIGHEST is specified and the object entity occurrence does not exist in the dictionary, the DDDL compiler assigns a version number of 1.

LOWest

Specifies the lowest version number associated with the object entity occurrence. If NEXT LOWEST is specified and the object entity occurrence does not exist in the dictionary, the DDDL compiler assigns a version number of 9999.

Examples

Assuming that versions 3, 8, 9, 11, and 23 of element ACCT-NUMBER exist in the dictionary, this statement implicitly requests version 23.

```
modify element account-number
     version is highest.
```

Assuming that versions 420, 440, and 460 of record EMP-NAME exist in the dictionary, this statement implicitly assigns version 461.

```
add record emp-name
     version is next highest.
```

Additional Qualifiers

Some entity types require or allow qualifiers in addition to those specified using the NAME and VERSION clauses. These entity types and their corresponding qualifiers are shown in the following table.

Entity type	Additional qualifier	Notes
ATTRIBUTE	WITHIN CLASS	Required when ADD/CREATE is specified. Required for other verbs when the attribute that is specified is *not* unique in the dictionary.
LOAD MODULE	MODULE TYPE	Optional; if MODULE TYPE is specified, the compiler makes sure the load module named in a MODIFY/ALTER or DELETE/DROP statement is of the same type.
MAP	WITHIN PANEL	Required when ADD/CREATE is specified.
MODULE	LANGUAGE	Required if a version of HIGHEST or LOWEST is specified or defaulted to.
	TYPE	Required when ADD/CREATE is specified.

More information: For more information about the qualifiers shown in the previous table, see the syntax and parameter descriptions for the corresponding entity types in Entity-Type Syntax (see page 127).

Securing the Dictionary

IDD provides security features that facilitate the protection of the data resource from unauthorized access, modification, or deletion, as follows:

- **Entity-type security**

  Allows the data administrator to secure access to the CLASS, ATTRIBUTE, and LOAD MODULE entities and to one or more CA IDMS/DB, CA IDMS/DC, and IDD entities. The data administrator can also restrict access to the DDDL compiler, CA IDMS/DB and CA ADS system components, and CA OLQ and CA Culprit operations. Entity-type security is controlled by the SET OPTIONS statement SECURITY FOR clause described under SET OPTIONS Syntax (see page 36).

  If the SET OPTIONS statement specifies a SECURITY IS ON option, only a user with the proper authority can access the secured entity or entity group or can perform the secured operation. If the authorized user has been assigned a password, that password must be provided. User authority is established with the USER statement AUTHORITY clause, which defines the entity group or entity types to which the user has access and specifies the type of access permitted (that is, the verbs the user can issue). For a description of the AUTHORITY clause, see USER (see page 358). If the SET OPTIONS statement specifies SECURITY IS OFF, user authority is not required; however, the data administrator can secure individual entity occurrences, as described below.

- **Entity-occurrence security**

  Controls user access to individual entity occurrences. The data administrator can apply entity-occurrence security to occurrences of all entity types *except* CLASS, LOAD MODULE, MESSAGE, and USER. The data administrator controls entity-occurrence security by means of the USER and PUBLIC ACCESS clauses within individual entity-type statements.

- **Password protection**

  Prohibits a user from adding or changing passwords for other users and from assigning other users the authority to access secured entity types or to perform secured operations. Password authority is established with the AUTHORITY clause of the USER statement. Typically, only one user has password authority; that user will control all passwords. However, the data administrator can activate a password security override to allow users to modify their own passwords. If the SET OPTIONS statement specifies INDIVIDUAL PASSWORD SECURITY OVERRIDE IS ON, users need no authority to modify their own passwords; the INDIVIDUAL PASSWORD SECURITY OVERRIDE clause is described under SET OPTIONS Syntax (see page 36).

The DDDL clauses in the following table govern security. Each of these clauses is described separately in this section.

This clause	Governs security by:

This clause	Governs security by:
PREPARED/ REVISED BY	Supplying additional user names and passwords to be used in IDD security
AUTHORITY	Assigning users authority to access secured entity types and perform secured operations
USER	Registering users with an entity occurrence and establishing the extent to which users can access or update the named entity occurrence
PUBLIC ACCESS	Specifying the extent to which unregistered users can access or update an entity occurrence

PREPARED/REVISED BY Clause

The PREPARED/REVISED BY clause supplies a user ID and optionally, a password. The DDDL compiler uses this information to determine whether the requested user is authorized to perform secured operations or access a secured entity type or entity occurrence.

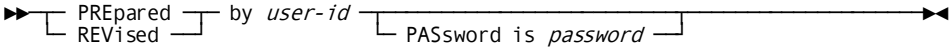
The PREPARED/REVISED BY clause within an individual entity-type statement overrides, for that statement only, the default PREPARED/REVISED BY specification, if present. The default PREPARED/REVISED BY specification is determined by the user ID supplied by one of the following, in order of precedence:

1. PREPARED/REVISED BY clause in the SET OPTIONS statement
2. DDDL compiler signon procedure
3. System signon procedure

A PREPARED/REVISED BY clause can appear in any ADD, MODIFY, REPLACE, DELETE, and DISPLAY/PUNCH statement associated with any entity type. PREPARED BY can be used when a new comment key definition is added to the dictionary; REVISED BY can be used when a comment key is changed.

The DDDL compiler reads in the entity-occurrence identification, then performs a security check. The compiler checks the user specified in the default PREPARED/REVISED BY clause and in any additional PREPARED/REVISED BY clauses. If neither of the requested users is authorized, the DDDL compiler rejects the entire statement. If at least one user is authorized, the DDDL compiler processes the statement.

Syntax: PREPARED/REVISED BY Clause



Parameters

PREpared by

Supplies a PREPARED BY specification for the named entity occurrence.

REVised by

Supplies a REVISED BY specification for the named entity occurrence.

user-id

Identifies the user requesting the ADD, MODIFY, REPLACE, DELETE, DISPLAY, or PUNCH operation. *User-id* must be a 1- through 32-character value and must be enclosed in quotation marks if it contains embedded blanks or delimiters. The specified ID must correspond to the ID of a user in the dictionary. If the requested entity occurrence is secured, the named user must be authorized to perform the requested operation. The DDDL compiler adds the authority of the user specified in the PREPARED/REVISED BY clause to the authority of the signed-on user to validate user-entity authority.

PASsword is *password*

Specifies the password associated with the named user. *Password* must be a valid 1- through 8-character password and must be enclosed in quotation marks if it contains embedded blanks or delimiters. If the named user has not been assigned a password, this parameter is invalid. DDDL suppresses the password when it echoes the command.

Example

In the following statement, user DGS adds the system named ACCOUNTING to the dictionary; DGS becomes the PREPARED BY specification for ACCOUNTING, overriding the default PREPARED BY specification established at signon or in a SET OPTIONS statement.

```
add system name is accounting
     prepared by user dgs.
```

Usage

Using PREPARED/REVISED BY for security

When you use this clause for security purposes, the PREPARED/REVISED BY clause must immediately follow the entity-occurrence identification.

SET OPTIONS may impact PREPARED/REVISED BY

If the USER SIGNON OVERRIDE clause of SET OPTIONS is set to OFF (or NOT ALLOWED), the PREPARED/REVISED BY clause is ignored and a warning message is displayed.

AUTHORITY Clause

The AUTHORITY clause of the USER statement defines a user in the dictionary and assigns the specified user authority to access secured entity types and perform secured operations.

Each user definition must include an AUTHORITY clause to grant the named user the authority to access each entity type, entity group, and product that has been secured by means of a SET OPTIONS SECURITY IS ON statement. The AUTHORITY clause also specifies the verbs (ADD, MODIFY, DELETE, REPLACE, DISPLAY, PUNCH) that the user is authorized to issue; this feature allows the data administrator to grant a user the authority to modify some entity types yet only display other entity types.

The syntax for the AUTHORITY clause appears with the USER statement (see USER (see page 358)).

Examples

In the following example, user DDA can use *any verb* to access *any secured entity type* and can perform *any secured operation*; typically, update authority is only assigned to the data administrator.

```
add user name is dda
     include authority for update is all.
```

In the following example, user WMC can modify and display all entity types in the IDD entity group *except* USER.

```
add user name is wmc
     include authority for modify is idd
     exclude authority for modify is user.
```

In the following example, user WMC can issue *all* USER statement clauses that require CA ADS, CA Culprit, and CA OLQ update authority, *all verbs* for all IDD entity types (*except* ADD QFILE and ADD PROCESS), and DISPLAY/PUNCH verbs for *all* entity types; however, user WMC cannot issue USER statements that require PASSWORD update authority.

```
add user name is wmc
     include authority for update is (ads olq culprit idd)
     exclude authority for add is (process qfile)
     include authority for display is all
     exclude authority for update is password.
```

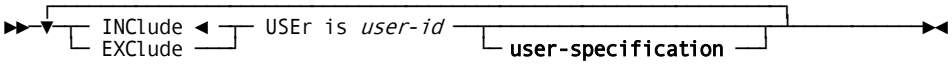
USER Clause

The USER clause is valid in all entity-type statements *except* CLASS, LOAD MODULE, MESSAGE, and USER. The USER clause:

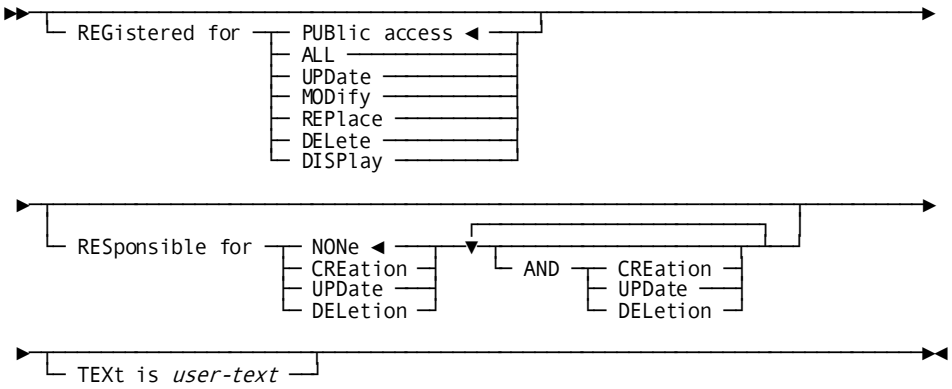
- Associates one or more existing users with the requested entity occurrence
- Registers each user to perform operations (MODIFY, REPLACE, DELETE, DISPLAY/PUNCH) for the requested entity occurrence, or establishes a registration option of public access.
- Assigns each user responsibility for the creation, update, and/or deletion of the requested entity occurrence.

Each iteration of the USER clause associates one user with the named entity, specifies a registration option and one or more responsibilities, and optionally supplies descriptive text.

Syntax: USER Clause



Expansion of *user-specification*



Parameters

USEr is *user-id*

Associates (INCLUDE) a user with or disassociates (EXCLUDE) a user from the requested entity occurrence. *User-id* must correspond to a 1- through 32-character user ID in the dictionary. If the specified ID includes embedded blanks or delimiters, it must be enclosed in site-standard quote characters.

user-specification

See the following descriptions for the REGISTERED FOR, RESPONSIBLE FOR, and TEXT parameters.

REGistered for

Registers the named user with the requested entity occurrence and specifies the functions the user can perform for the entity.

PUBlic access

Specifies that the PUBLIC ACCESS clause (described later in this chapter) controls the functions that the user can perform. This is the REGISTERED FOR default.

ALL

Specifies that the user is registered to perform all functions; the user can issue MODIFY, REPLACE, DELETE, and DISPLAY/PUNCH verbs, and can change the REGISTERED FOR options for other users and the PUBLIC ACCESS specification.

UPDate

Specifies that the user is registered to perform update functions; the user can issue MODIFY, REPLACE, DELETE, and DISPLAY/PUNCH verbs but cannot change the REGISTERED FOR and PUBLIC ACCESS specifications.

MODify

Specifies that the user is registered only to issue MODIFY and DISPLAY/PUNCH verbs.

REPlace

Specifies that the user is registered only to issue REPLACE and DISPLAY/PUNCH verbs.

DELete

Specifies that the user is registered only to issue DELETE and DISPLAY/PUNCH verbs.

DISPlay

Specifies that the user is registered only to issue DISPLAY/PUNCH verbs.

RESponsible for

Documents responsibility for the named user. The options named with RESPONSIBLE FOR do not have any impact on entity-occurrence security.

NONe

Specifies that no responsibility is documented for the named user. NONE is the default.

CREation

Documents creation responsibility for the named user.

UPDate

Documents update responsibility for the named user.

DELetion

Documents deletion responsibility for the named user.

AND CREation/UPDate/DELetion

Documents additional creation, update, or deletion responsibilities for the user. You can repeat this clause.

TEXt is *user-text*

Associates 1 through 40 characters of documentation text with the user/entity relationship. If the text includes special characters or embedded blanks, it must be enclosed in quotation marks. For more information about the TEXT clause, see Documenting Entity Occurrences (see page 81), later in this chapter.

Examples

The following examples illustrate four forms of the USER clause.

In the following example, user WMC can perform all functions for the CUSTOMER record (issue MODIFY, REPLACE, DELETE, and DISPLAY verbs, change the REGISTERED FOR specification for other users, and change the PUBLIC ACCESS specification); user WMC is also assigned documentation responsibility for creating, updating, and deleting the record.

```
add record name is customer
   include user wmc
     registered for all
     responsible for creation
       and update and deletion.
```

In the following example, user WMC can modify, replace, delete, and display the requested entity occurrence but *cannot* change the REGISTERED FOR specifications for other users or the PUBLIC ACCESS specification of the requested entity occurrence.

```
include user wmc
     registered for update.
```

In the following example, user WMC can modify and display *only* the requested entity occurrence.

```
include user wmc
     registered for modify.
```

In the following example, user WMC can only display the requested entity occurrence.

```
include user wmc
     registered for display.
```

Usage

USER clause rules

The following rules apply to the USER clause:

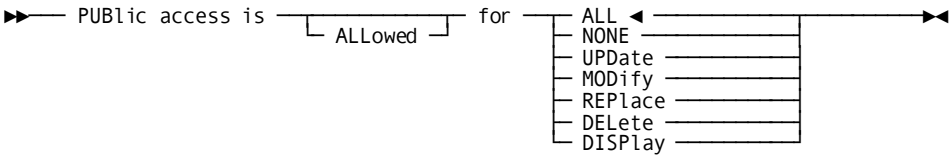
- The clause can be repeated as needed to define multiple users for each entity occurrence.
- To assign a value other than ALL to the PUBLIC ACCESS clause (that is, to override the default), at least one user of the entity occurrence must be assigned the REGISTERED FOR ALL option; see PUBLIC ACCESS Clause (see page 79) for additional details.
- An EXCLUDE request that names the last user assigned the REGISTERED FOR ALL option will not be processed unless PUBLIC ACCESS IS ALLOWED FOR ALL has been specified; see PUBLIC ACCESS Clause (see page 79) below for further details.
- The REGISTERED FOR parameter overrides any previously specified registration options for the named user.

PUBLIC ACCESS Clause

PUBLIC ACCESS specifications control entity-occurrence security by identifying the extent to which unregistered users can access and/or update the requested entity occurrence. If the PUBLIC ACCESS clause is not specified in an ADD statement, any user with the proper entity-type authority can update and display the requested entity occurrence.

Note: The optional PUBLIC ACCESS clause is valid in all entity-type statements *except* CLASS, LOAD MODULE, MESSAGE, and USER.

Syntax: PUBLIC ACCESS Clause



Parameters

PUBlic access is ALLOwed for

Supplies the public access specification for the requested entity occurrence.

ALL

Specifies that unregistered users are allowed to issue all verbs and perform all secured operations. ALL is the default.

NONe

Specifies that unregistered users are not allowed to access the entity occurrence.

UPDate

Specifies that unregistered users are allowed to issue all verbs (MODIFY, REPLACE, DELETE, and DISPLAY/PUNCH).

MODify

Specifies that unregistered users are allowed to issue only MODIFY and DISPLAY/PUNCH verbs.

REPlace

Specifies that unregistered users are allowed to issue only REPLACE and DISPLAY/PUNCH verbs.

DELete

Specifies that unregistered users are allowed to issue only DELETE and DISPLAY/PUNCH verbs.

DISplay

Specifies that unregistered users are allowed to issue only DISPLAY/PUNCH verbs.

Examples

In the following example, any user can modify, replace, delete, and display the requested entity occurrence and change the REGISTERED specification.

```
public access is allowed for all.
```

In the following example, unregistered users cannot update or display the requested entity occurrence.

```
public access is allowed for none.
```

In the following example, unregistered users can modify, replace, delete, and display *only* the requested entity occurrence.

```
public access is allowed for update.
```

Usage

Overriding PUBLIC ACCESS is NONE

When the first user assigned the REGISTERED FOR ALL option is associated with an entity occurrence, the DDDL compiler automatically sets the PUBLIC ACCESS specification to NONE in order to prohibit unregistered users from accessing the entity. To override the PUBLIC ACCESS specification, the data administrator must submit the PUBLIC ACCESS clause immediately after the REGISTERED FOR ALL specification. For example:

```
add record cust-rec
     user jmc registered for all
     public access is display.
```

Later, any user who has been registered for all can change the PUBLIC ACCESS specification.

PUBLIC ACCESS clause with option other than ALL

The DDDL compiler will not process a PUBLIC ACCESS clause that specifies an option other than ALL unless at least one user associated with the requested entity occurrence is assigned the REGISTERED FOR ALL option. This feature ensures that each entity occurrence has at least one user who can change the REGISTERED FOR specification.

Documenting Entity Occurrences

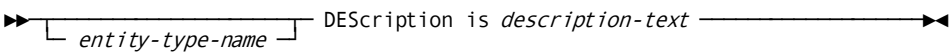
The DESCRIPTION, COMMENTS, and TEXT clauses are used to document entity-occurrence definitions. These clauses are described separately in this section.

DESCRIPTION Clause

The DESCRIPTION clause associates up to one line of documentation text with an entity occurrence. Typically, descriptions clarify entity-occurrence identifications or briefly explain the expected use of an entity. This clause functions as follows:

- In an *ADD* statement, DESCRIPTION establishes a user-specified description for the entity occurrence.
- In a *MODIFY* or *REPLACE* statement, DESCRIPTION replaces an existing description in its entirety, or, if no description exists, establishes the specified description.

Syntax: DESCRIPTION Clause



Parameters

entity-type-name

Identifies the entity type with which the description is being associated. If specified, *entity-type-name* must be a standard IDD entity type.

DEScription is *description-text*

Assigns 1 through 40 (64, with element occurrences) characters of description text to the requested entity occurrence. *Description-text* must be coded on one line and, if the text contains embedded blanks or delimiters, must be enclosed in site-standard quote characters. To remove existing description text, specify a null string ('').

Examples

The following examples illustrate the use of the DESCRIPTION clause.

The following clause associates documentation text with the FILE occurrence named BILLING.

```
add file billing
     description is 'outstanding accounts receivable'.
```

The following clause nullifies an existing DESCRIPTION clause.

```
modify system payroll
     system description is ''.
```

COMMENTS Clause

Comments are used to store lengthy descriptions of entities. For each entity type (except LOAD MODULE), IDD permits an unlimited number of user-supplied comment entries. The user can associate any number of lines of text with each entry; no restrictions apply.

Comment Text Is Identified by Comment Keys

Comment text is identified by *predefined* or *user-defined* comment keys, which can be associated with any entity occurrence to separately document design, operational, or usage considerations for the named entity. For example, the user might associate the user-defined comment key RECOVERY PROCEDURE with a program; text associated with that comment key contains instructions directed to the operator for use if the program terminates abnormally.

Predefined Comment Keys

The DDDL compiler supports the predefined comment keys shown in the following table.

Comment key	Identifies
COMMENTS	General comments
DEFINITION	A full description of the use or purpose of the entity occurrence
CULPRIT HEADER	An alternative column header for use in CA Culprit reports. The length and number of these headers are governed by CA Culprit conventions, as described in the *CA Culprit User Guide*. This comment key is valid only with RECORD statements and RECORD ELEMENT substatements.
OLQ HEADER	An alternative column header for use in CA OLQ reports. This comment key is valid only with RECORD statements and RECORD ELEMENT substatements.
REMARKS	Descriptive text for programs. This comment key can appear in PROGRAM statements only.

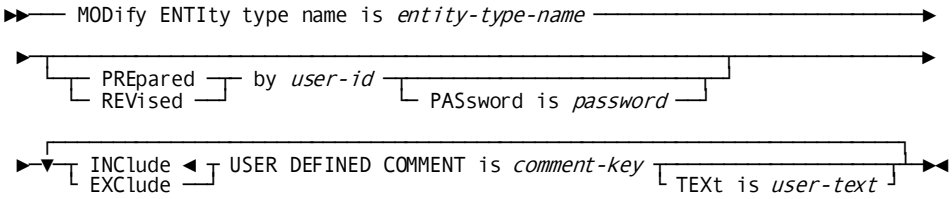
User-defined Comment Keys

You can define *additional* comment keys.

To define a new comment key, you must issue a MODIFY ENTITY statement to modify the standard ENTITY definition established during IDD installation. You must be assigned ATTRIBUTE authority to define comment keys (see AUTHORITY Clause (see page 75)).

Note: Do not use the MODIFY ENTITY statement to add user-defined entities to the dictionary; the result of such use is unpredictable.

Syntax: MODIFY ENTITY Statement (for user-defined comments)



Parameters

ENTIty type name is *entity-type-name*

Specifies the entity type that is the object of the modification. *Entity-type-name* can be any standard IDD entity-type name; however, several entity types cannot appear in this clause. A list of the substitute names to be used for these entity types follows:

Entity type	Substitute name
ENTRY POINT	PROGRAM
PROCESS	MODULE
QFILE	MODULE
REPORT	RECORD
SCREEN	PANEL
SUBSYSTEM	SYSTEM
TABLE	MODULE
TRANSACTION	RECORD
User-defined entity	ATTRIBUTE

PREpared/REVised by *user-id*

Identifies the user requesting the operation. PREPARED BY can be used when a new comment key definition is added to the dictionary; REVISED BY can be used when a comment key is changed. If the named user has been assigned a password, the PASSWORD parameter must be specified. See Securing the Dictionary (see page 72) earlier in this chapter for the rules pertaining to the PREPARED/REVISED BY clause.

PASsword is *password*

Specifies the password of the user named in the PREPARED/REVISED BY clause. If *password* contains embedded blanks or delimiters, it must be enclosed in site-standard quote characters.

USER DEFINED COMMENT is *comment-key*

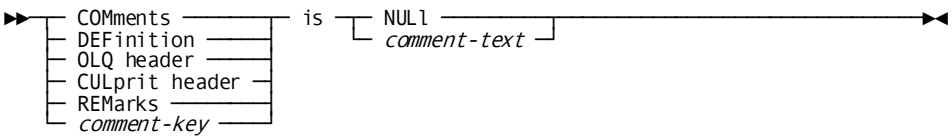
Identifies the comment key to be associated with (INCLUDE) or dissociated from (EXCLUDE) the requested entity type; INCLUDE is the default. *Comment-key* must be a unique 1- through 40-character value. Values that contain embedded blanks or special characters or that duplicate a keyword from the DDDL syntax must be enclosed in site-standard quote characters. Note that a keyword defined as a relational key (see Associating Entity Occurrences (see page 103), later in this chapter) for the requested entity cannot be defined as a comment key for the same entity. This clause can be repeated to add any number of comment keys.

Note: Use the EXCLUDE option with care. When a comment key is excluded from an entity type, relationships between occurrences of the entity that are based on the excluded comment key cannot be deleted, reported on, or reestablished with the INCLUDE option. First, delete the comment text from all entity occurrences with which it is associated; then exclude the comment key.

TEXt is *user-text*

Associates documentation text with the comment key. Text must be 1 through 40 characters in length and, if it includes delimiters or embedded blanks, must be enclosed in site-standard quote characters.

Syntax: COMMENTS Clause



Parameters

COMments/DEFinition/OLQ header /CULprit header/REMarks/*comment-key* is

Identifies the predefined (COM/DEF/OLQ/CUL/REM) or user-defined (*comment-key*) comment key to which the comment text applies. *Comment-key* must be a user-defined key previously established in the dictionary through the MODIFY ENTITY statement. If *comment-key* includes delimiters or embedded blanks, or if it duplicates a DDDL keyword, it must be enclosed in site-standard quote characters. Because the DDDL compiler recognizes comment keys as keywords, the specified comment key can be abbreviated.

NULl

Removes existing text from the comment key.

comment-text

Specifies the comment text to be associated with the comment key. *Comment-text* can consist of multiple input lines. Each line following the first line must begin with the continuation character (-) followed by the site-standard quote character; the closing quotation mark is optional. Once defined, comment text can be edited (see EDIT Clause (see page 93), later in this chapter).

Usage

Associating text with a comment key

After using the MODIFY ENTITY statement to include a comment key for an entity type, you can use the COMMENTS clause to associate text with a predefined or user-defined comment key.

Include a COMMENTS clause in the applicable entity-type statement. If a comments clause appears in a MODIFY or REPLACE statement, the DDDL compiler edits, replaces, or removes existing comment text.

Disassociating comment text from a comment key

To delete a comment key, remove the comment text associated with a specified entity (using the NULL parameter of the COMMENTS clause). If the comment is user-defined, issue the MODIFY ENTITY statement specifying the EXCLUDE USER DEFINED COMMENT option.

Examples

The following statement establishes the comment key SPECIAL CONSIDERATIONS for the SYSTEM entity type.

```
modify entity system
    revised by j-user
    include user defined comment is 'special considerations'.
```

The following statement associates the comment text VACATION PAY INCLUDED IN JUNE 30 CHECKS with the comment key SPECIAL CONSIDERATIONS for the system PAYROLL.

```
modify system payroll
     'special considerations' is 'vacation pay included'
     -'in june 30 checks'.
```

The following statement deletes the comment key from the system PAYROLL.

```
modify system payroll
    'special considerations' is null.
```

The following statement excludes the comment key from the entity type.

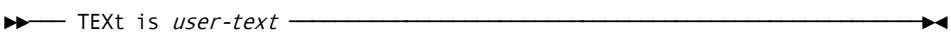
```
modify entity system
    revised by j-user
    exclude user defined comment is 'special considerations'.
```

TEXT Clause

The TEXT clause associates documentation text with the following:

- User to entity relationships
- Relational keys
- Relational-key to entity-occurrence structures
- Attribute to entity relationships
- Language to module structures
- File-type/VSAM-type/input-module/device-type/file to file relationships
- Module/program/system/user to system relationships
- Entry point/module/program to program relationships
- Record to record relationships
- User/file to user relationships

Syntax: TEXT Clause



Parameters

TEXt is *user-text*

Specifies 1 to 40 characters of documentation text. If *user-text* includes embedded blanks or delimiters, it must be enclosed in site-standard quote characters.

Copying and Editing Entity Occurrences

The DDDL compiler syntax includes three clauses that are used to manipulate entity-occurrence definitions, as shown in the following table. Each of the clauses is described separately in this section.

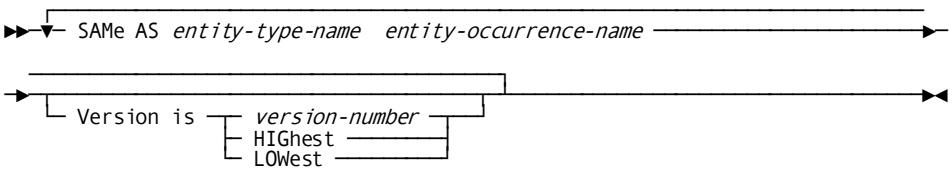
Clause	What it does
SAME AS	Copies the entire definition associated with an existing entity occurrence into the object entity-occurrence definition
COPY	Copies selected options from an existing entity occurrence into the object entity-occurrence definition
EDIT	Modifies lines in comment text and module source by performing add, replace, or delete functions, as specified

SAME AS Clause

Use the optional SAME AS clause in ELEMENT, FILE, MODULE, QFILE, PROCESS, PROGRAM, RECORD (REPORT) (TRANSACTION), SYSTEM (SUBSYSTEM), TABLE, and USER statements to reduce the amount of coding needed to define multiple entities of the same type.

SAME AS copies options associated with an existing entity occurrence into another entity occurrence. Except as noted in individual entity-type statements in Chapter 4, all options are copied.

Syntax: SAME AS Clause



Parameters

SAMe AS *entity-type-name*

Specifies the name of the object entity type; valid names are ELEMENT, FILE, MODULE, PROCESS, QFILE, TABLE, PROGRAM, RECORD, REPORT, TRANSACTION, SYSTEM, SUBSYSTEM, and USER.

entity-occurrence-name

Specifies the name of the entity-occurrence definition to be copied. The specified name must be the entity's primary name; it cannot be a synonym. DDDL makes sure that the entity occurrences specified (for the module, qfile, process, and table entities) are the same entity type.

Version is *version-number*/HIGhest/LOWest

Qualifies nonunique entity-occurrence names for the SAME AS clause.

Usage

Considerations for using SAME AS

The following considerations apply to using the SAME AS clause:

- When a definition is copied into an existing entity occurrence, the copied definition is merged with the existing definition.
- You must have authority to access the entity occurrence from which the definition is to be copied.
- When the SAME AS clause is used with an entity type that supports USER REGISTERED FOR, the compiler copies all registered users from the original entity occurrence to the named entity occurrence. The compiler also cross-references the registration in the applicable USER entity occurrence.
- VERSION HIGHEST *cannot* be specified for the entity-occurrence to be copied if:
  - The SAME AS clause references the same entity-occurrence name as the entity-occurrence being added or modified

    and

  - A version number, VERSION HIGHEST or NEXT HIGHEST establishes the entity-occurrence being added or modified as the highest version of the entity-occurrence

  For example, assume you are starting with version 1 of the program PAYROLL. NEXT HIGHEST in the ADD PROGRAM statement (below) produces a version 2. The following SAME AS clause recalls the newly established version 2 (instead of the intended version 1), causing the error message ATTEMPTED RECURSIVE CONNECTION.

```
add program payroll version next highest
     same as program payroll version highest.
```

Example

The following example adds the elements MODEL-DATE, PROMISE-DATE, and SHIP-DATE to the dictionary, copying the definition of MODEL-DATE for PROMISE-DATE and SHIP-DATE.

```
add element model-date
    sub elements are
        month
        day
        year.

add element promise-date
    same as element model-date
    comments 'items will be shipped before promise date'
    -'when possible'.

add element ship-date
    same as element model-date.
```

The DISPLAY statement lists the resulting definition for SHIP-DATE.

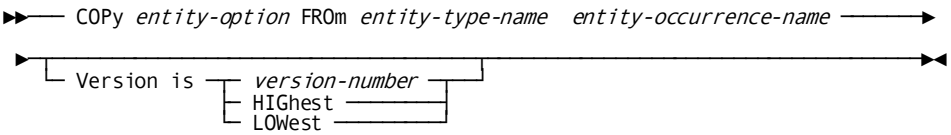
```
display element ship-date prep by j-user.
        *+   add
        *+   element name is ship-date
        *+      date created is      mm/dd/yy
        *+      prepared by j-user
        *+      subordinate elements are
        *+           month    version is 1
        *+           day      version is 1
        *+           year     version is 1
        *+      .
```

COPY Clause

Use the COPY clause to copy selected options from one entity-occurrence definition to another and to merge the copied options into the target definition. Options that exist within both entity definitions are not copied. The COPY clause is valid in any entity-type statement that supports the SAME AS clause.

Note: To use the COPY clause, the user must have authority to access the entity occurrence from which the definition is to be copied. The secured entity must allow the user a minimum of DISPLAY access through either a PUBLIC ACCESS clause or a REGISTERED clause.

Syntax: COPY Clause



Parameters

COPy *entity-option*

Specifies the portion of the object entity definition to be copied. For the valid syntax options for each entity type, see Entity-Type Syntax (see page 127).

FROm *entity-type-name*

Specifies the name of the source entity type; valid names are ELEMENT, FILE, MODULE, PROCESS, QFILE, TABLE, PROGRAM, RECORD, REPORT, TRANSACTION, SYSTEM, SUBSYSTEM, or USER.

entity-occurrence-name

Specifies the name of the existing entity occurrence from which the option is to be copied. DDDL makes sure that the entity occurrences specified (module, qfile, process, and table) are the same entity type. Source text prevents copying source text from one entity type to an unrelated entity type.

Version is

Qualifies nonunique entity-occurrence names.

Example

The following example adds programs STCKUPDT and INVCTRL to the dictionary. All modules associated with STCKUPDT are copied to INVCTRL. Because modules ONORD, REORD, and NEWORD exist in both programs, those modules are not copied.

```
add program stckupdt
     module used is onord language is assembler
     module used is reord language is assembler
     module used is neword language is assembler
     module used is stat language is assembler
     module used is recov language is assembler.

add program invctrl
     module used is reord language is assembler
     module used is onord language is assembler
     module used is neword language is assembler.

modify program invctrl
     copy modules from program stckupdt.
```

The DISPLAY statement lists the resulting definition for INVCTRL.

```
display program invctrl.
        *+   add
        *+   program name is invctrl
        *+   date created is   mm/dd/yy
        *+   prepared by j-user
        *+   module used is onord version is 1 language is assembler
        *+   module used is reord version is 1 language is assembler
```

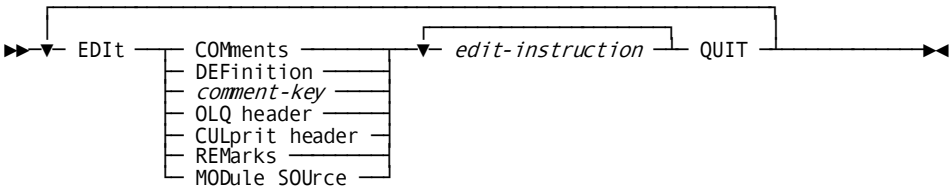
```
*+   module used is neword version is 1 language is assembler
*+   module used is stat version is 1 language is assembler
*+   module used is recov version is 1 language is assembler.
```

EDIT Clause

Use an EDIT clause within entity-type statements to update comment text and the source statements that comprise modules, processes, and qfiles. The EDIT clause is intended for use in batch mode or with a dial-up device; in full-screen mode, users can employ the online text editor described in Online DDDL Compiler (see page 387).

Each comment line and source statement has a unique line number by which it can be referenced during editing. The DDDL compiler automatically generates these numbers, incrementing each line by 100 or by the default value defined in the SEQUENCE clause of the SET OPTIONS statement. Unless overridden in a SEQUENCE instruction (described in this chapter), the default value is referred to as the current increment.

Syntax: EDIT Clause



Parameters

EDIt

Specifies the object of the edit operation. EDIT and the object of EDIT (for example, EDIT COMMENTS) must be coded on a line by itself.

COMments

Specifies that text associated with the predefined comment key COMMENTS is to be edited.

DEFinition

Specifies that text associated with the predefined comment key DEFINITION is to be edited.

comment-key

Specifies that text associated with a user-defined comment key is to be edited. The specified comment key must exist in the dictionary and must either be abbreviated to one word that does not duplicate a DDDL keyword or be enclosed in quotation marks if it includes embedded blanks or delimiters or duplicates a DDDL keyword.

OLQ header

Specifies that text associated with the predefined comment key OLQ HEADER is to be edited.

CULprit header

Specifies that text associated with the predefined comment key CULPRIT HEADER is to be edited.

REMarks

Specifies that text associated with the predefined comment key REMARKS is to be edited.

MODule SOUrce

Specifies that text associated with the named module, process, or qfile source is to be edited.

edit-instruction

Specifies the edit operation to be performed; valid keywords are INSERT, REPLACE, ERASE, LIST, SEQUENCE, and SHOW. Multiple edit instructions can appear between the EDIT and QUIT keywords; however, a single input line can contain only one edit instruction.

QUIT

Terminates the EDIT clause. This keyword must appear on a separate input line following the last edit instruction. If QUIT is omitted, the DDDL compiler attempts to interpret subsequent DDDL source statements as edit instructions and may incorrectly modify the entity occurrence to which the EDIT statement applies.

Example

The following figure shows an Integrated Data Dictionary Activity List containing EDIT instructions that insert text in the module IDMS-STATUS.

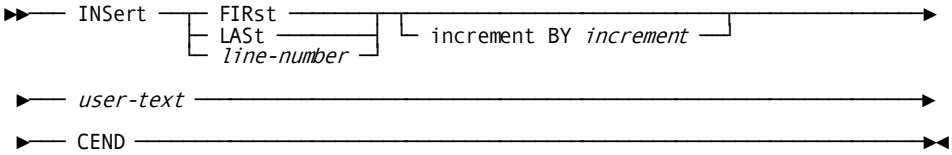
```
IDMSDDDL     nn.n                          CA, INC.                              DATE          TIME          PAGE
                                INTEGRATED DATA DICTIONARY ACTIVITY LIST          mm/dd/yy      12393315      0001

0001             SET OPTIONS INPUT COLUMNS ARE 1 THRU 71.
0002     MODIFY MODULE IDMS-STATUS
0003         EDIT MODULE SOURCE
0004         SHOW ON
0005         SEQUENCE 100
0006         LIST FIRST TO LAST
     100)       ****************************************************************
     200)        IDMS-STATUS                                              SECTION.
     300)       ****************************************************************
     400)                 IF DB-STATUS-OK GO TO ISABEX.
     500)                 PERFORM IDMS-ABORT.
     600)                 DISPLAY '************************'
     700)                         ' ABORTING - ' PROGRAM-NAME
     800)                         ', '           ERROR-STATUS
     900)                         ', '           ERROR-RECORD
    1000)                         ' **** RECOVER IDMS ****'
    1100)                        UPON CONSOLE.
    1200)                 DISPLAY 'PROGRAM NAME ------ ' PROGRAM-NAME.
    1300)                 DISPLAY 'ERROR STATUS ------ ' ERROR-STATUS.
    1400)                 DISPLAY 'ERROR RECORD ------ ' ERROR-RECORD.
    1500)                 DISPLAY 'ERROR SET ---------- ' ERROR-SET.
    1600)                 DISPLAY 'ERROR AREA -------- ' ERROR-AREA.
    1700)                 DISPLAY 'LAST GOOD RECORD -- ' RECORD-NAME.
    1800)                 DISPLAY 'LAST GOOD AREA ---- ' AREA-NAME.
    1900)                 DISPLAY 'DML SEQUENCE--------' DML-SEQUENCE.
    2000)                 ROLLBACK.
    2100)                 CALL 'ABORT'.
    2200)            ISABEX. EXIT.
0007         INSERT 1950
0008                      DISPLAY 'DBKEY____________' DBKEY.
0009         CEND
           THE FOLLOWING WERE INSERTED:
    1950)                 DISPLAY 'DBKEY____________' DBKEY.
0010         LIST 1800 TO LAST
    1800)                 DISPLAY 'LAST GOOD AREA ---- ' AREA-NAME.
    1900)                 DISPLAY 'DML SEQUENCE--------' DML-SEQUENCE.
    1950)                 DISPLAY 'DBKEY____________' DBKEY.
    2000)                 ROLLBACK.
    2100)                 CALL 'ABORT'.
    2200)            ISABEX. EXIT.
0011         QUIT.
```

Syntax and parameter descriptions for each of the EDIT instructions follow.

INSERT Instruction of the EDIT Clause

INSERT adds one or more text lines before or after existing text or at a specified line in the existing text. If the SET OPTIONS statement specifies the AFTER IS ON option or if a SHOW instruction (described under a later instruction heading) precedes the current INSERT instruction, the results of the insert operation are listed at the terminal or on the Integrated Data Dictionary Activity List.

Syntax: INSERT Instruction of EDIT



Parameters

INSert

Inserts the specified text.

FIRst

Adds the new text before the existing text; the starting line number is equal to the current increment.

LASt

Adds the new text after the existing text, beginning at the last line number plus the current increment.

line-number

Adds the first line of new text at the specified unused line number within the existing text; *line-number* must be a 1- through 8-digit integer.

increment BY *increment*

Specifies the starting line number and the interline increment to be applied to new lines. *Increment* must be a 1- through 8-digit integer. If used, this clause must appear on the same input line as the INSERT keyword. For this clause only, the specified increment value becomes the current line increment value, overriding the SET OPTIONS SEQUENCE default or the temporary default established by the SEQUENCE instruction.

The RESEQUENCE option of the SET OPTIONS statement affects INSERT operations, as follows:

- If RESEQUENCE IS OFF is specified, all lines must be inserted between two existing text lines because no resequencing will occur. Therefore, *increment-number* must be small enough to accommodate all new lines.
- If RESEQUENCE IS ON is specified, any number of lines can be inserted in existing text because resequencing will occur after all edit instructions are processed.

user-text

Specifies one line of text to be inserted, beginning in column 1. Each additional line of text must be coded on a separate input line.

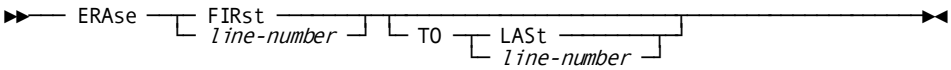
CEND

Terminates the INSERT instruction. If CEND is omitted, the DDDL compiler issues the END OF FILE BEFORE QUIT message.

ERASE Instruction of the EDIT Clause

ERASE removes the specified text lines from existing comment text or module source. If the SET OPTIONS statement specifies the BEFORE IS ON option or if a SHOW instruction precedes the current ERASE instruction, the DDDL compiler will list the erased text at the terminal or on the Integrated Data Dictionary Activity List.

Syntax: ERASE Instruction of EDIT



Parameters

ERASE

Specifies an erase operation.

FIRst

Erases the first line of existing text or begins the erase operation at the first line of existing text.

line-number

Specifies an existing text line to be erased or begins the erase operation at the specified line.

TO LASt

Continues the ERASE function through the last line of existing text.

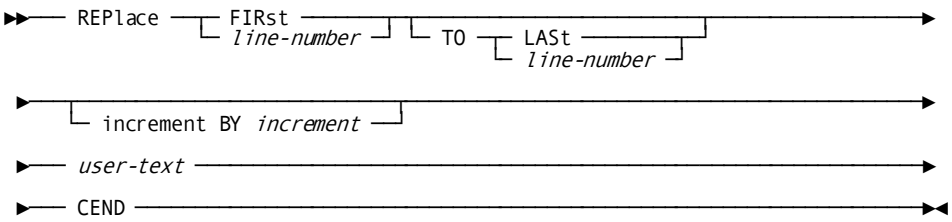
TO *line-number*

Continues the ERASE function through the specified line.

REPLACE Instruction of the EDIT Clause

REPLACE removes the specified text lines from existing comment text or module source and adds new text beginning at the line vacated by the first removed line. If the SET OPTIONS statement specifies the AFTER IS ON or BEFORE IS ON option or if a SHOW ON instruction precedes the current REPLACE instruction, the DDDL compiler will list the results of the REPLACE operation (both the text removed and the existing text with additions) at the terminal or on the Integrated Data Dictionary Activity List.

Syntax: REPLACE Instruction of EDIT



Parameters

REPlace

Specifies a REPLACE operation.

FIRst

Removes the first line of existing text or begins the removal at the first line within the specified range of lines. The new text is added in place of the first deleted line.

line-number

Removes the specified line of existing text or begins the removal at the first text line within the specified range of lines. The first line of new text is added at the specified location.

TO LASt

Continues the removal and replacement of existing text through the last existing line.

TO *line-number*

Continues the removal and replacement of existing text through the specified line.

increment BY *increment*

Specifies the starting line number for the REPLACE FIRST instruction and the interline increment to be applied to replaced lines. *Increment* must be a 1- through 8-digit integer. If used, this optional clause must appear on the same input line as the REPLACE keyword. For this clause only, *increment* becomes the current line-increment value, overriding the SET OPTIONS SEQUENCE default or the temporary default established by the SEQUENCE instruction.

The RESEQUENCE option of the SET OPTIONS statement affects REPLACE operations as follows:

- If RESEQUENCE IS OFF is specified, all lines must be inserted between two existing text lines because no resequencing will occur. Therefore, *increment-number* must be small enough to accommodate all required replacement lines.
- If RESEQUENCE IS ON is specified, any number of replacement lines can be inserted in existing text because resequencing will occur after all edit instructions are processed.

user-text

Specifies one line of text to be inserted, beginning in column 1. Each additional line of text must be coded on a separate input line.

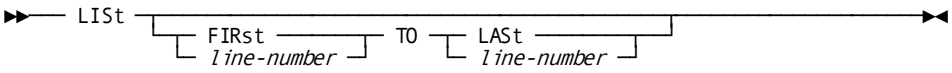
CEND

Terminates the REPLACE instruction. If CEND is omitted, the DDDL compiler will issue the END OF FILE BEFORE QUIT message.

LIST Instruction of the EDIT Clause

LIST requests the DDDL compiler to list the specified line or range of lines from existing comment text or module source at the terminal or on the Integrated Data Dictionary Activity List.

Syntax: LIST Instruction of EDIT



Parameters

LISt

Lists all lines of existing text.

LISt FIRst

Lists the first line of text or initiates the LIST function at the first line of existing text.

LISt *line-number*

Lists the specified line of text or initiates the LIST function at the specified line or first existing line after the specified line.

TO LASt

Continues the LIST function through the last line of text.

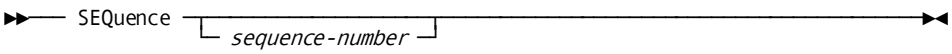
TO *line-number*

Continues the LIST function through the line identified by *line-number* or the last existing line in the specified range. The ending line number must be greater than the beginning line number.

SEQUENCE Instruction of the EDIT Clause

SEQUENCE is used to resequence comment text or module source.

Syntax: SEQUENCE Instruction of EDIT



Parameters

SEQuence

Requests that the DDDL compiler resequence existing text by using the default line increment value specified in the SET OPTIONS statement SEQUENCE clause.

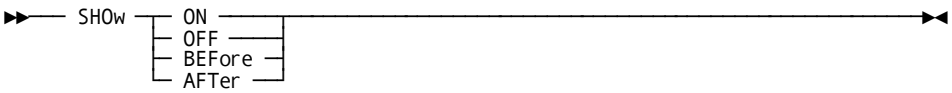
sequence-number

Specifies the sequence number that SEQUENCE is to use as the starting line number, the interline increment, and the current increment for all subsequent INSERT and REPLACE instructions of the current EDIT clause.

SHOW Instruction of the EDIT Clause

SHOW requests or suppresses the listing of the results of subsequent INSERT, ERASE, or REPLACE instructions at the terminal or on the Integrated Data Dictionary Activity List. SHOW overrides SET OPTIONS statement BEFORE and AFTER clause specifications and remains in effect for all INSERT, ERASE, and REPLACE instructions until the DDDL compiler encounters another SHOW instruction or until the EDIT clause is terminated.

Syntax: SHOW Instruction of EDIT



Parameters

SHOw

Specifies a SHOW operation.

ON

Selects both the AFTER IS ON and BEFORE IS ON defaults for the INSERT, ERASE, and REPLACE instructions in the current EDIT clause.

OFF

Selects both the AFTER IS OFF and BEFORE IS OFF defaults for the INSERT, ERASE, and REPLACE instructions in the current EDIT clause.

BEFore

Selects only the BEFORE IS ON default for the ERASE and REPLACE instructions in the current EDIT clause.

AFTer

Selects only the AFTER IS ON default for the INSERT and REPLACE instructions in the current EDIT clause.

Associating Entity Occurrences

IDD supports relationships between entity occurrences to enable the dictionary to correctly represent relational facts about the data resource. An example of such a relationship is the association between a user and a system.

Standard Relationships

The DDDL compiler establishes standard entity-occurrence relationships through the clauses shown in the following table.

More information: For more information on these clauses, see Entity-Type Syntax (see page 127).

Clause	What it does
USER	Defines the relationship between an entity occurrence and its users. This clause is valid in all entity types except CLASS, LOAD MODULE, MESSAGE, and USER and is described under Securing the Dictionary (see page 72), earlier in this chapter.
WITHIN SYSTEM	Defines the relationship between a destination, line, logical terminal, map, module, physical terminal, process, program, qfile, queue, table, or task and a system or subsystem. Syntax for the WITHIN SYSTEM clause appears in the individual entity-type syntax in Chapter 4.
Nesting clauses	Expresses hierarchical relationships between two users, systems, files, elements, or programs.

Standard Nesting Clauses

The standard nesting clauses are as follows:

Entity type	Clause
ELEMENT	SUBORDINATE ELEMENT
FILE	RELATED FILE
PROGRAM	PROGRAM CALLED
SYSTEM	WITHIN SYSTEM
USER	WITHIN USER

These clauses are described in detail within the applicable entity-type statement in Chapter 4.

User-defined Relationships

The DDDL compiler also supports user-defined entity-occurrence relationships through the following:

- **User-defined nests**

  Express relationships between entities of the same type in terms that are meaningful within the user environment. IDD supports user-defined nests through *relational keys*.

- **Class/attribute structures**

  Relate documentation characteristics known as *attributes* to entity occurrences.

Relational keys and attribute/entity relationships are discussed separately in this section.

Relational Keys

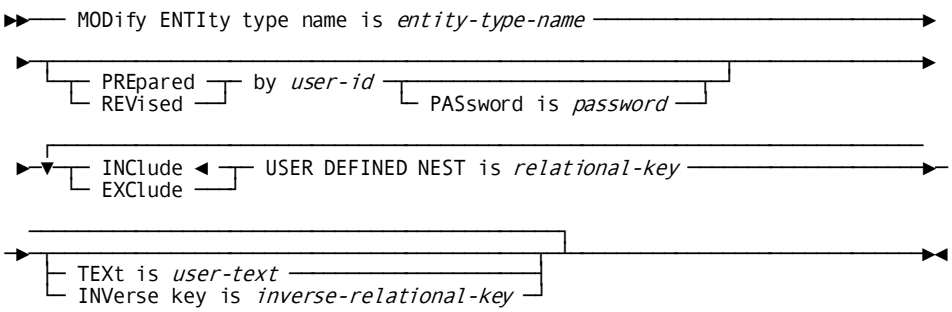
Relational keys are user-defined keywords that relate entities of the same type. The user can associate any number of relational keys with occurrences of the ATTRIBUTE, ELEMENT, FILE, MODULE, PROGRAM, RECORD, SYSTEM, USER, and user-defined entity types by including a *relational-key clause* within the applicable entity-type statement. Relational-key clauses are functionally similar to standard DDDL nesting clauses; however, the use of a relational key allows the user to express the relationship in more precise terms.

Defining Relational Keys

To define a relational key, the user must issue a MODIFY ENTITY statement to modify the standard ENTITY definition established during IDD installation.

Note: Do not use the MODIFY ENTITY statement to add user-defined entities to the dictionary; the result of such use is unpredictable.

Syntax: MODIFY ENTITY Statement (for user-defined nests)



Parameters

ENTIty type name is *entity-type-name*

Specifies the entity type that is the object of the modification. *Entity-type-name* can be any standard IDD entity-type name; however, several entity types cannot appear in this clause. A list of the substitute names to be used for these entity types follows:

Entity type	Substitute name
ENTRY POINT	PROGRAM
PROCESS	MODULE
QFILE	MODULE
REPORT	RECORD
SCREEN	PANEL
SUBSYSTEM	SYSTEM
TABLE	MODULE
TRANSACTION	RECORD
User-defined entity	ATTRIBUTE

PREpared/REVised by *user-id*

Identifies the user requesting the operation. The PREPARED BY clause can be used when a new comment key definition is added to the dictionary; REVISED BY can be used when a comment key is changed. For the rules pertaining to the PREPARED/REVISED BY clause, refer to Securing the Dictionary (see page 72), earlier in this chapter.

PASsword is *password*

Specifies the password of the user named in the PREPARED BY/REVISED BY clause. If the named user has been assigned a password, this parameter must be specified.

USER DEFINED NEST is *relational-key*

Identifies the relational key to be associated with (INCLUDE) or dissociated from (EXCLUDE) the object entity type; INCLUDE is the default. *Relational-key* must be a unique 1- through 40-character value. Values that contain embedded blanks or delimiters, or that duplicate a keyword from the DDDL syntax must be enclosed in site-standard quote characters. The same relational key can be defined for multiple entity types; however, a keyword defined as a comment key for the object entity cannot be defined as a relational key for the same entity (see COMMENTS Clause (see page 83), earlier in this chapter). This parameter can be repeated to add any number of relational keys.

Note: Use the EXCLUDE option with care. If a relational key is excluded from an entity type, relationships between occurrences of that entity that are based on the excluded relational key cannot be deleted, reported on, or reestablished with the INCLUDE option. First, delete the relationship from all entity occurrences; then exclude it from the ENTITY definition.

TEXt is *user-text*

Associates documentation text with the relational key. *User-text* must be 1 through 40 characters in length and, if it includes delimiters or embedded blanks, must be enclosed in site-standard quote characters.

INVerse key is *inverse-relational-key*

Associates a second relational key with the primary relational key. *Inverse-relational-key* is a unique 1- through 40-character value. Values that contain embedded blanks or delimiters or that duplicate a keyword from the DDDL syntax must be enclosed in site-standard quote characters.

When two entity occurrences are associated with the primary relational key, the DDDL compiler automatically maintains the logical connections implied by the secondary (inverse) key as well as those associated with the primary key. The DDDL compiler also maintains primary and secondary connections when two entity occurrences are associated with an inverse relational key. The user can modify the inverse relational key without affecting all occurrences of the primary relational key.

Example

The following 3-step example illustrates the use of relational keys and inverse relational keys.

1. The following statement defines a relational key for the USER entity type with an inverse relational key.

```
modify entity user
     revised by j-user
     include user defined nest is 'manages'
         inverse key is 'works for'.
```

2. Three USER definitions are added to the dictionary. JOE is added without the use of relational keys. BOB is added to the dictionary, and his relationship with ANN is documented using the inverse relational key. ANN is added to the dictionary, and her relationship with JOE is documented using the primary relational key.

```
add user joe.
add user ann
     'manages' joe.
add user bob
     'works for' ann.
```

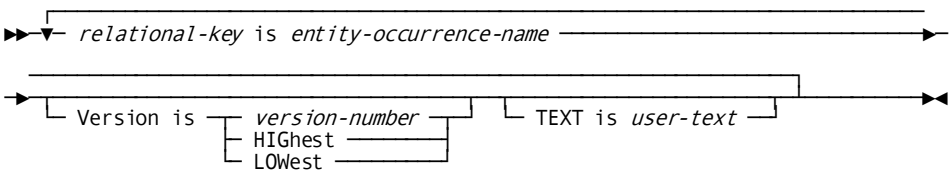
3. The resulting definitions are displayed.

```
display user joe.
        *+   add
        *+   user name is joe
        *+      'works for' is ann
        *+   .
display user bob.
        *+   add
        *+   user name is bob
        *+      'works for' is ann
        *+   .
display user ann.
        *+   add
        *+   user name is ann
        *+      'manages' is joe
        *+      'manages' is bob
        *+   .
```

Using Relational-key Clauses

The user can include a *relational-key* clause within the applicable entity-type statement to associate an entity occurrence with an occurrence of the same entity type. The relational-key clause can be repeated using the same or different relational keys to associate the entity occurrence with additional occurrences of the same entity type. The DDDL compiler rejects any relational-key clauses that attempt to duplicate existing relationships.

Syntax: RELATIONAL-KEY Clause



Parameters

***relational-key* is**

Names an existing relational key. The specified value must be enclosed in site-standard quote characters if it contains embedded blanks or delimiters, or if it duplicates a DDDL compiler keyword. Because the DDDL compiler recognizes relational keys as keywords, the specified relational key can be abbreviated.

entity-occurrence-name

Names the entity occurrence to which the object entity occurrence is being related. If *entity-occurrence-name* is qualified by multiple versions, the optional VERSION clause must be specified.

Note: The user can supply a LANGUAGE parameter to uniquely identify occurrences of the MODULE entity type in relational-key clauses (see MODULE (PROCESS/QFILE/TABLE) (see page 219)).

Version is *version-number*/HIGhest/LOWest

Qualifies nonunique entity-occurrence names for the relational-key clause.

TEXt is *user-text*

Associates 1 through 40 characters of documentation text with the nested structure being defined. If the text contains embedded blanks or delimiters, it must be enclosed in site-standard quote characters.

Examples

The following statement associates the previously-defined file WEEKLY-SALES with the new file, INVOICES, by means of the relational key SIMILAR FILE.

```
add file invoices
     'similar file' is weekly-sales.
```

The following statements establish a relationship between users. Both departments and individuals are documented as users.

```
modify entity type name is user
     user defined nest is department-number.

add user name is 122.

add user name is wmc
     department-number is 122.
```

Attribute/Entity Relationships

Attributes are characteristics that can be assigned to entities.

Classes are categories of attributes.

For example, the attributes COBOL, Assembler, and PL/I are assignable to programs and are grouped into a class called LANGUAGE.

Note: For more information on the rules for defining attributes and classes, see the ATTRIBUTE (see page 131) and CLASS (see page 138) sections.

A Class Must Exist in the Dictionary

A class must exist in the dictionary in order for attributes within that class to be related to entity occurrences. Each class definition contains qualifiers that determine how attributes within the class are added to the dictionary and govern how many attributes can be related to each entity occurrence. These qualifiers are described in the following table.

Qualifiers for Attributes

To do this	Use these qualifiers
Specify how attributes are added to the dictionary	**Manual** Attributes within classes assigned the manual qualifier must be defined in the dictionary explicitly with ADD ATTRIBUTE statements before being associated with an entity occurrence. Typically, the manual qualifier applies to classes having a limited number of attributes that can be easily predefined. For example, the class SEX has only two attributes, MALE and FEMALE. These attributes must exist in the dictionary before they can be related to occurrences of the USER entity.
	Automatic Attributes within classes assigned the automatic qualifier are added to the dictionary automatically. The automatic qualifier applies to classes having an unlimited number of attributes that would be difficult to predefine. For example, the class BIRTH DATE has unlimited attributes. These attributes are added to the dictionary automatically when they are related to occurrences of the USER entity.

To do this	Use these qualifiers
Specify how many attributes can be related to each entity occurrence	**Singular** Only one attribute can be related to each entity occurrence. For example, if attributes within the class LANGUAGE are to be related to programs, LANGUAGE should be assigned the singular qualifier because only one language (for example, COBOL) is valid for a single program.
	Plural An unlimited number of attributes can be related to each entity occurrence. For example, if attributes within the class LANGUAGE are to be related to users, LANGUAGE should be assigned the plural qualifier because a user could be proficient in several languages.

Standard Classes - LANGUAGE and MODE

The Integrated Data Dictionary automatically creates two standard classes; these classes and their qualifiers are as follows:

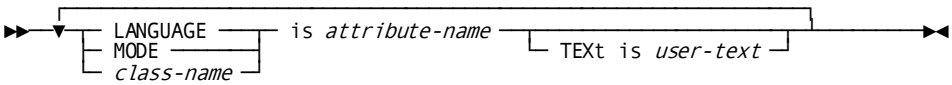
- **LANGUAGE** class— Qualifiers are MANUAL PLURAL.
- **MODE** class— Qualifiers are AUTOMATIC PLURAL.

The IDD installation procedure assigns attributes to the LANGUAGE class (for example, OLQ, CULPRIT, COBOL).

Class/Attribute Clause

The repeatable *class/attribute clause*, valid in all entity-type statements, is used to establish attribute/entity relationships.

Syntax: Class/attribute Clause



Parameters

LANGUAGE/MODE/*class-name* is

Specifies the class in which the named attribute participates. Specify LANGUAGE or MODE to relate an attribute within the predefined class LANGUAGE or MODE to the requested entity occurrence. Specify *class-name* to relate an attribute within a user-defined class to the requested entity occurrence. The name must be 1 through 20 characters in length, must reference a class that has been defined in the dictionary with an ADD CLASS statement, and must be coded on one input line. *Class-name* cannot be abbreviated.

Note: The specification of LANGUAGE or MODE affects the processing of other CA IDMS data-management components and should be used with care.

attribute-name

Specifies the attribute to be related to the named entity. The named attribute must exist in the dictionary if the named class is assigned the manual qualifier. If *attribute-name* includes embedded blanks or delimiters, it must be enclosed in site-standard quote characters. The specified attribute name must be unique within the named class but need not be unique within the dictionary.

TEXt is *user-text*

Associates 1 through 40 characters of documentation text with this attribute/entity relationship. If the text includes embedded blanks or delimiters, it must be enclosed in site-standard quote characters.

Examples

Assuming that class DATE-OF-HIRE has been defined with the automatic qualifier, the following statement adds user JCD and attribute 2/25/87 to the dictionary and relates this attribute to both user JCD and class DATE-OF-HIRE.

```
add user jcd
     date-of-hire is 2/25/87.
```

Using the predefined class LANGUAGE, the following statement associates the predefined attribute COBOL with the program BILLING.

```
modify program billing
     language is cobol.
```

Displaying Entity Occurrences

You can list one or more entity-occurrence definitions by using the DISPLAY/PUNCH statement, which functions as follows:

- **DISPLAY**— Lists all or selected portions of the requested entity occurrences at the terminal or on the Integrated Data Dictionary Activity List. During an online session, the user can edit DISPLAY verb output and resubmit it as input to the DDDL compiler.
- **PUNCH**— When used online, functions in the same way as DISPLAY. When used in batch mode, writes the requested information to the SYSPCH file or to an IDD module defined as the destination for PUNCH verb output.

Optional DISPLAY/PUNCH statement clauses allow the user to specify, for the current DISPLAY/PUNCH statement only, the entity-type options to be listed, whether these options are to appear as syntax or comments, the verb to accompany the DISPLAY/PUNCH output, and, for PUNCH only, the destination for the punched output. If the DISPLAY/PUNCH statement requests multiple occurrences of an entity type, the user can supply a conditional expression that specifies criteria to be used by the DDDL compiler in selecting the requested entities.

Two Output Formats

The format of DISPLAY/PUNCH verb output is governed by the SET OPTIONS statement FORMAT IS FIXED/FREE specification. A FREE format appears as *running text*, for example:

```
display next map.
*+   display map name is linda version is 1
*+          within panel linda-olmpanel version is 1
display prior program.
*+   display program name is chs02 version is 1 .
```

A FIXED format appears in a *columnar* presentation, for example:

```
display first 2 maps.
*+   display
*+   map name                        mkmap2
*+    version                         0000000001
*+    within panel                   mkmap2-olmpanel
*+     version                        0000000001
*+     .
*+   display
*+   map name                        mkmap1
*+    version                         0000000001
*+    within panel                   mkmap1-olmpanel
*+     version                        0000000001
*+     .
```

Columnar format facilitates access to DDDL compiler output by online CA IDMS applications.

Requesting Single or Multiple Occurrences

There are two DISPLAY/PUNCH statements:

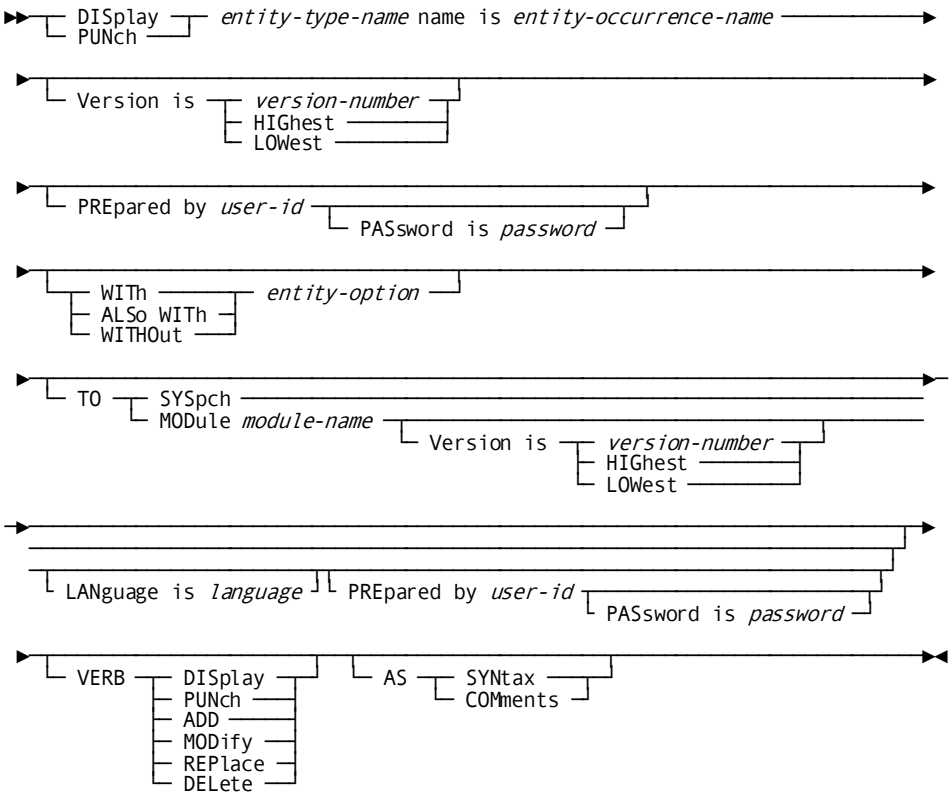
- DISPLAY/PUNCH for requesting a single entity occurrence
- DISPLAY/PUNCH ALL for requesting multiple occurrences

The syntax for each statement is presented separately in the following two subsections. Note that parameter descriptions that apply to both DISPLAY/PUNCH and DISPLAY/PUNCH all appear following the DISPLAY/PUNCH syntax.

DISPLAY/PUNCH Statement

The DISPLAY/PUNCH statement allows you to display or punch options for a *single* entity occurrence.

Syntax: DISPLAY/PUNCH (for a single entity occurrence)



Parameters

DISplay/PUNch *entity-type-name*

Specifies that the DDDL compiler is to display or punch the information associated with a single entity-occurrence definition. *Entity-type-name* must be a valid IDD entity type.

name is *entity-occurrence-name*

Specifies an existing occurrence of the specified entity type.

Version is *version-number*/HIGhest/LOWest

Qualifies nonunique entity names.

PREpared by *user-id*

Identifies the user requesting the DISPLAY operation. *User-id* must be a 1- through 32-character value and must be enclosed in quotation marks if it contains embedded blanks or delimiters.

PASsword is *password*

Specifies the password associated with the user named in the PREPARED BY parameter.

WITh *entity-option*

Specifies that the DDDL compiler is to replace the options specified in the SET OPTIONS DISPLAY WITH and DISPLAY ALSO WITH statements with the specified options for this DISPLAY/PUNCH request only.

ALSo WITh

Specifies that the DDDL compiler is to add the specified options to the default options specified in the SET OPTIONS DISPLAY WITH and DISPLAY ALSO WITH statements for this DISPLAY/PUNCH request only.

WITHOut

Specifies that the DDDL compiler is to exclude the specified options from the default options specified in the SET OPTIONS DISPLAY WITH and DISPLAY ALSO statements for this DISPLAY/PUNCH request only.

entity-option

Specifies an entity-specific option that is the object of the WITH/ALSO WITH/WITHOUT specification. All *entity-option*s you can specify are described in detail under the DISPLAY clause of the SET OPTIONS statement (see SET OPTIONS Syntax (see page 36)). Individual syntax diagrams in Chapter 4 list the valid options for each entity type.

TO

Specifies the destination for punched output (used with PUNCH only).

SYSpch

Specifies that the DDDL compiler is to direct PUNCH verb output to the SYSPCH file.

MODule *module-name*

Specifies that the DDDL compiler is to direct PUNCH verb output to the named module. *Module-name* must be the 1- through 32-character name of a module defined in the dictionary through the MODULE statement (see MODULE (PROCESS/QFILE/TABLE) (see page 219)). The following rules apply to the named module:

- Once the module has been named as the destination of the PUNCH command, it cannot be modified, replaced, or deleted.
- A module cannot be punched to itself.
- The PUNCH verb cannot name a module that is the object of an INCLUDE statement.

If module source is already associated with the object module, the DDDL compiler adds the PUNCH verb output to the end of the existing module. If module source does not exist, the DDDL compiler generates a header which contains the date and time that the initial punched output was created.

The specified destination overrides the default destination established in the SET OPTIONS PUNCH statement.

Version is *version-number*/HIGhest/LOWest

Qualifies the named module with a version number.

LANguage is *language*

Qualifies the named module with a language.

VERB DISplay/PUNch/ADD/MODify/REPlace/DELete

Specifies the verb that is to accompany DISPLAY/PUNCH output. This parameter overrides the default verb established in the SET OPTIONS VERB statement.

AS SYNtax

Specifies that the text output by the DISPLAY/PUNCH verb is to appear as syntax. In an online session, text displayed as syntax can be edited and resubmitted to the DDDL compiler. If the PUNCH command is issued in batch mode, the DDDL compiler directs the output to the SYSPCH file or to an IDD module, where it can be edited and subsequently resubmitted.

This parameter overrides the default format established in the SET OPTIONS statement.

AS COMments

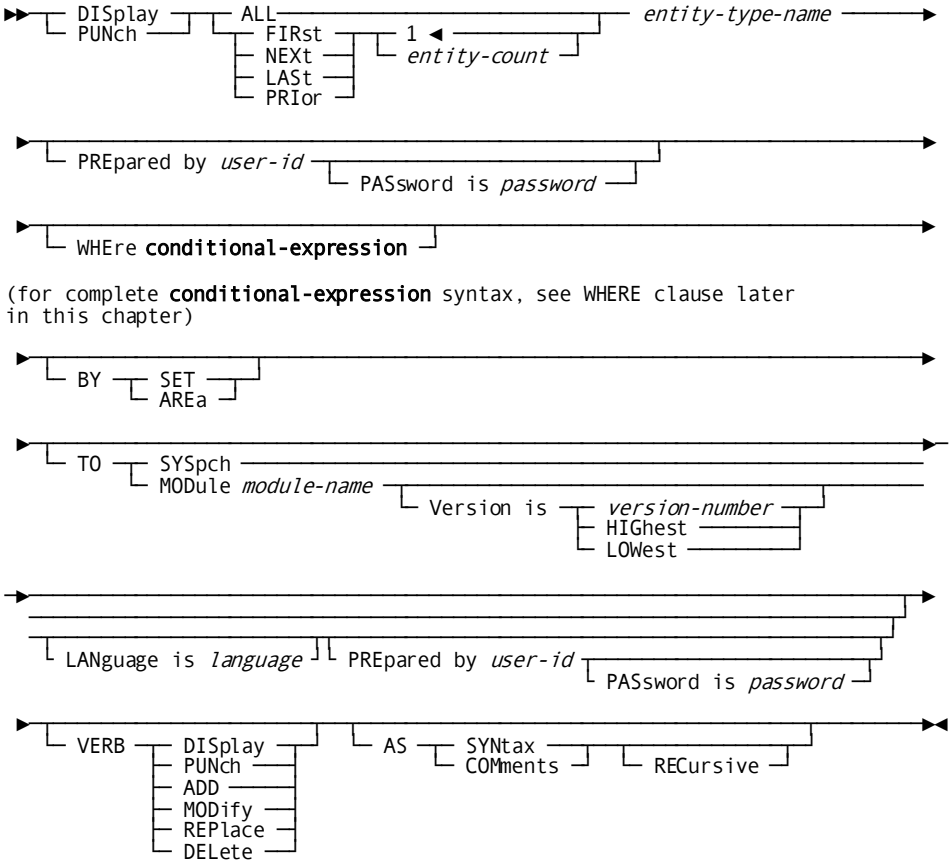
Specifies that the text output by the DISPLAY/PUNCH verb be formatted as compiler comments; comments are preceded by *+ and are ignored by the DDDL compiler. This parameter overrides the default format established in the SET OPTIONS statement.

DISPLAY/PUNCH ALL Statement

The DISPLAY/PUNCH ALL statement allows you to display or punch options for *multiple* entity occurrences.

Note: The parameter descriptions that apply to both the DISPLAY/PUNCH and the DISPLAY/PUNCH ALL statements appear after the DISPLAY/PUNCH syntax.

Syntax: DISPLAY/PUNCH ALL (for multiple entity occurrences)



Parameters

Parameters specific to DISPLAY/PUNCH ALL

DISplay/PUNch ALL/FIRst/NEXt/LASt/PRIOr

Specifies that the DDDL compiler is to display or punch multiple entity occurrences. The output consists only of the information necessary to execute a DISPLAY/PUNCH *entity* request for each entity occurrence. For example, RECORD occurrences are displayed with their name and version, MODULE occurrences with their name and language, and ATTRIBUTE occurrences with their name and class. In an online session, the user can execute the displayed statements by pressing ENTER. This two-step process allows the user to scan the contents of the dictionary for the desired entity-occurrence definitions without generating unneeded output.

ALL

Lists all occurrences of the requested entity type that the current user is authorized to display. With a large number of entity occurrences, ALL may slow online response time. You can use the DISPLAY ALL LIMIT and INTERRUPT COUNT clauses of the SET OPTIONS statement (see SET OPTIONS Statement (see page 34)) to limit DISPLAY.

FIRst/NEXt/LASt/PRIOr

Lists the first, next, last, or prior occurrences of the named entity type.

entity-count

Specifies the number of occurrences displayed or punched. 1 is the default.

entity-type-name

Identifies the entity type or entity synonym that is the object of the DISPLAY/PUNCH ALL request.

WHEre *conditional-expression*

Specifies criteria to be used by the DDDL compiler in selecting occurrences of the requested entity type. *Conditional-expression* is described in detail under WHERE Clause (Conditional Expressions) (see page 118), later in this chapter.

BY SET

Retrieves all entity occurrences by the set relationship between the OOAK-012 record and the entity record. BY SET is most efficient when the dictionary contains relatively fewer occurrences than there are pages in the dictionary. BY SET can be applied to any entity type except MESSAGE or LOAD MODULE. SET is the default for all entity types except ELEMENT and ELEMENT SYNONYM.

BY AREa

Retrieves all entity occurrences by sweeping the DDLDML area of the dictionary. BY AREA is most efficient when the dictionary contains relatively numerous occurrences of the requested entity-type, generally more occurrences than there are pages in the dictionary. BY AREA can be applied to any entity type except MESSAGE or LOAD MODULE. AREA is the default for the ELEMENT and ELEMENT SYNONYM entity types.

RECursive

Appends "AS SYNTAX." or "AS COMMENT." to each generated line of output.

WHERE Clause (Conditional Expressions)

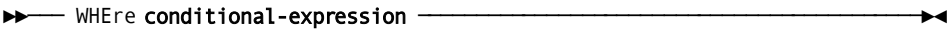
The WHERE clause of a DISPLAY/PUNCH ALL statement defines a condition. The outcome of a test for the condition determines which occurrences of the named entity type the DDDL compiler selects for display.

The WHERE clause can contain a single condition, or two or more conditions combined with the logical operators AND or OR. The logical operator NOT specifies the opposite of the condition. The DDDL compiler evaluates operators in a WHERE clause one at a time, from left to right, in order of precedence. The default order of precedence is as follows:

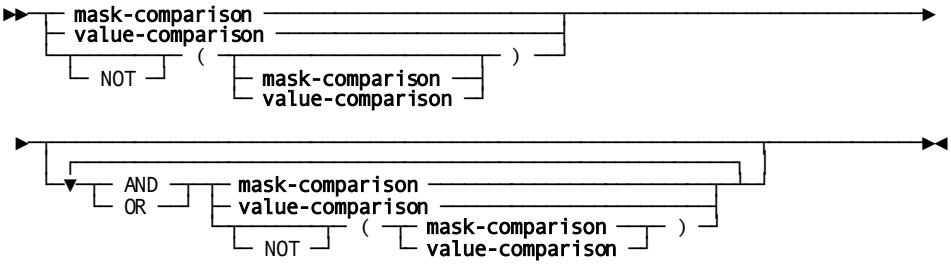
- MATCHES or CONTAINS keywords
- EQ, NE, GT, LT, GE, LE operators
- NOT
- AND
- OR

If parentheses are used to override the default order of precedence, the DDDL compiler evaluates the expression within the innermost parentheses first.

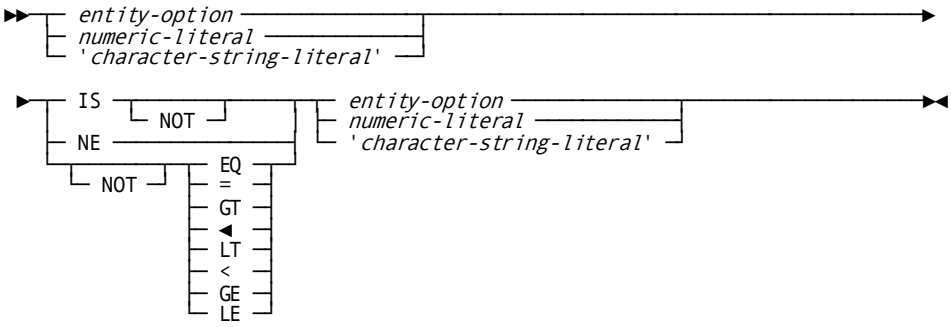
Syntax: WHERE Clause (for conditional expressions)



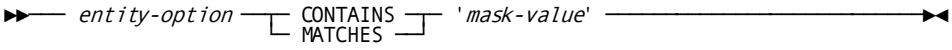
Expansion of *conditional-expression*



Expansion of value-comparison



Expansion of mask-comparison



Parameters

NOT

Specifies that the opposite of the condition fulfills the test requirements; if NOT is specified, the condition must be enclosed in parentheses.

AND

Specifies a logical operator to accompany multiple conditions. The expression is true only if the outcome of both test conditions is true.

OR

Specifies a logical operator to accompany multiple conditions. The expression is true if the outcome of either one or both test conditions is true.

value-comparison

Compares values represented in the left and right-side operands based on the specified comparison operator.

entity-option

Identifies a syntax option associated with the named entity type; valid options for each entity type are listed in the table following these parameter descriptions.

numeric literal

Identifies a numeric value.

'*character-string-literal*'

Identifies a character string enclosed in quotes.

IS/NOT

Specifies whether the left operand is equal (IS) or is not equal (IS NOT) to the right operand.

NE

Specifies whether the left operand is not equal to the right operand.

EQ/GT/LT/GE/LE

Specifies whether the left operand is equal to, greater than, less than, greater than or equal to, or less than or equal to the right operand. Each operator can be preceded by NOT to specify the opposite of the condition.

mask-comparison

Compares an entity type operand with a mask value.

CONTAINS

Searches the left operand for an occurrence of the right operand. The length of the right operand must be less than or equal to the length of the left operand. If the right operand is not contained entirely in the left operand, the outcome of the condition is false.

MATCHES

Compares the left operand with the right operand one character at a time, beginning with the leftmost character in each operand. When a character in the left operand does not match a character in the right operand, the outcome of the condition is false.

'***mask-value***'

Identifies the right operand as a character string; the specified value must be enclosed in quotation marks. The following characters can be specified in *mask-value*:

- **@** matches any **alphabetic character** in *entity-option*
- **#** matches any **numeric character** in *entity-option*
- ***** matches **any character** in *entity-option*

Valid Entity Options for the WHERE Clause

Entity type	Option
ATTRIBUTE	Entity-type NAME
	PREPARED BY
	REVISED BY
	DATE LAST UPDATED
	DATE CREATED
	CLASS NAME

Entity type	Option
CLASS ENTRY POINT MESSAGE	Entity-type NAME PREPARED BY REVISED BY DATE LAST UPDATED DATE CREATED
DESTINATION ELEMENT FILE LINE LOGICAL-TERMINAL MAP MODULE PANEL PHYSICAL-TERMINAL PROCESS QFILE QUEUE RECORD/REPORT/TRANS- ACTION SYSTEM/SUBSYSTEM TABLE TASK USER	Entity-type NAME VERSION PREPARED BY REVISED BY DATE LAST UPDATED DATE CREATED DESCRIPTION FULL NAME (USERS only) LANGUAGE (MODULEs only) LINE NAME (PHYSICAL-TERMINALs only) LINE TYPE (LINEs only) PANEL NAME (MAPs only) PHYSICAL-TERMINAL NAME (LOGICAL-TERMINALs only)
ELEMENT SYNONYM FILE SYNONYM	SYNONYM NAME SYNONYM VERSION (FILEs only) ELEMENT or FILE NAME VERSION PREPARED BY REVISED BY DATE LAST UPDATED DATE CREATED DESCRIPTION
LOAD MODULE	LOAD MODULE NAME VERSION DATE COMPILED MODULE TYPE

Entity type	Option
PROGRAM	PROGRAM NAME VERSION PREPARED BY REVISED BY DATE LAST UPDATED DATE CREATED DESCRIPTION DATE COMPILED
RECORD SYNONYM REPORT SYNONYM TRANSACTION SYNONYM	SYNONYM NAME SYNONYM VERSION RECORD NAME VERSION PREFIX SUFFIX VIEW ID RECORD NAME VERSION PREPARED BY REVISED BY DATE LAST UPDATED DATE CREATED DESCRIPTION
SCHEMA	Entity-type name PREPARED BY REVISED BY DATE LAST UPDATED DATE CREATED DATE COMPILED DESCRIPTION
SUBSCHEMA	Entity-type name PREPARED BY REVISED BY DATE LAST UPDATED DATE CREATED DESCRIPTION SCHEMA NAME SCHEMA VERSION

Entity type	Option
TASK	NAME
	VERSION
	PREPARED BY
	REVISED BY
	DATE LAST UPDATED
	DATE CREATED
	DESCRIPTION

Date Selection Criteria

In the following WHERE clause options, you can select the date as a value-comparison string in the form 'MM/DD/YY' on the right-hand side of the conditional expression:

- DATE CREATED
- DATE LAST UPDATED
- DATE COMPILED

The extraction interprets the date in CCMMDDYY form to accurately determine the relationship of dates. For example, the following DISPLAY ALL statement specifies the search criteria to identify the RECORD occurrences whose DATE CREATED values (which are also evaluated in CCYYMMDD form) are greater than the specified string:

```
display all records where
    date created ◄ '01/01/96'.
```

The DISPLAY ALL process determines that the date '01/01/96' is greater than the date '12/31/95'.

Alternatively, you can specify the value-comparison string on either side of the conditional expression in the form 'CCYYMMDD' to achieve the same results.

You can substitute day, month, or year for each of the WHERE clause options. For example, the following DISPLAY ALL statement specifies a search condition which is based on month and year:

```
display all records where
    month created = '01' and
    year created ◄ '95'.
```

DISPLAY/PUNCH Examples

Displaying a Single Entity Occurrence

The following four statements illustrate DISPLAY/PUNCH statements that request information about a single entity occurrence.

1. IDD displays as comments the user-defined nests associated with the system PAYROLL.

```
display system payroll
     with user defined nests
     as comments.
*+   add
*+   system name is payroll version is 1
*+       'prerequisite system' is 'employee maintenance'
*+       version 1
*+       .
```

2. SET OPTIONS establishes HISTORY as the default display option and AS COMMENTS as the output format.

```
set options for session
     display with history as comments.
```

3. IDD displays the definition of the CUSTOMER record with ELEMENT and PICTURE OVERRIDES specifications in addition to HISTORY.

```
display record customer
     also with elements picture overrides.
*+   add
*+   record name is customer version is 1
*+       date created is      mm/dd/yy
*+       date last updated is mm/dd/yy
*+       prepared by wmc
*+       revised by wmc
*+       .
*+       record element is cust-name version 1 line is 000100
*+       level number is 02
*+       usage is display
*+       .
```

4. IDD displays the same definition without the PICTURE OVERRIDES option.

```
display record customer
     without picture overrides.
*+   add
*+   record name is customer version is 1
*+       date created is      mm/dd/yy
*+       date last updated is mm/dd/yy
*+       prepared by wmc
*+       revised by wmc
*+       record length is 119
```

```
*+       record name synonym is customer version 1
*+       .
*+       record element is cust-name version 1
*+       level number is 02
*+       .
```

Displaying/Punching Multiple Entity Occurrences

The following example illustrates a DISPLAY/PUNCH statement that requests multiple entity occurrences. IDD displays the first five occurrences of the ELEMENT entity type; to submit the resulting ADD ELEMENT statements to the DDDL compiler, you must press ENTER.

```
display first 5 elements
     verb add
     as syntax.
   add element name is field-array version is 1 .
   add element name is emp-fname-09-ws version is 9 .
   add element name is emp-lname-09-ws version is 9 .
   add element name is emp-name-09-ws version is 9 .
   add element name is emp-info-09-ws version is 9 .
```

The following example illustrates a DISPLAY/PUNCH statement that requests multiple entity occurrences with 'as syntax' appended to each generated line. IDD displays the first five occurrences of the ELEMENT entity type.

```
display first 5 elements
     as syntax recursive.
   display element name is field-array version is 1 as syntax.
   display element name is emp-fname-09-ws version is 9 as syntax.
   display element name is emp-lname-09-ws version is 9  as syntax.
   display element name is emp-name-09-ws version is 9 as syntax.
   display element name is emp-info-09-ws version is 9 as syntax.
```

In the following example, IDD displays as comments all modules that contain the literal MOD- as part of the module name.

```
display all modules
     where name contains 'mod-'
     as comments.
```

In the following example, IDD punches all files to the module DEMO-PUNCH.

```
punch all files
     to module demo-punch
     as syntax.
```

Chapter 5: Entity-Type Syntax

This section contains the following topics:

Overview (see page 128)
Considerations for Syntax Presentation (see page 129)
ATTRIBUTE (see page 131)
CLASS (see page 138)
DESTINATION (see page 144)
ELEMENT (see page 151)
ELEMENT_SYNONYM (see page 167)
ENTRY_POINT (see page 169)
FILE (see page 174)
FILE_SYNONYM (see page 184)
LINE (see page 186)
LOAD_MODULE (see page 192)
LOGICAL_TERMINAL (see page 198)
MAP (see page 205)
MESSAGE (see page 211)
MODULE_(PROCESS/QFILE/TABLE) (see page 219)
PANEL_(SCREEN) (see page 234)
PHYSICAL_TERMINAL (see page 239)
PROCESS (see page 247)
PROGRAM (see page 255)
QFILE (see page 272)
QUEUE (see page 281)
RECORD_(REPORT/TRANSACTION) (see page 287)
RECORD_SYNONYM (see page 333)
SYSTEM_(SUBSYSTEM) (see page 334)
TABLE (see page 341)
TASK (see page 351)
USER (see page 358)
USER-DEFINED_ENTITY (see page 380)

Overview

To populate, update, or access the dictionary, the user submits to the DDDL compiler a source statement that is unique to each entity type.

The verbs described in the following table specify the action the DDDL compiler is to take for each DDDL statement.

Verb	Action
ADD	Creates a new entity occurrence in the dictionary.
MODIFY	Changes an existing entity-occurrence definition.
REPLACE	Initializes to defaults or excludes all options associated with an existing entity occurrence; relationships that have been established through other entity-type statements or other CA IDMS compilers are not affected.
DELETE	Removes an existing entity occurrence. DELETE is not valid for entity occurrences that have been defined with the system generation compiler.
DISPLAY/PUNCH	Lists all or selected portions of one or more existing entity occurrences.

Note: These verbs are valid only if the entity occurrence has been created by the DDDL compiler.

This chapter presents syntax, parameter descriptions, and usage tips for each entity type. Entity types are in alphabetical order.

Considerations for Syntax Presentation

Order of Presentation

This order of presentation is followed for each entity type:

1. Syntax for ADD/MODIFY/REPLACE/DELETE

   Note: Where it is necessary to expand a parameter, the location of the expansion is noted in the diagram.

2. Syntax for DISPLAY/PUNCH

   (for listing a single entity occurrence)

3. Syntax for DISPLAY/PUNCH ALL

   (for listing multiple entity occurrences)

4. Parameter descriptions for all syntax
5. Usage tips
6. Examples

Repetition of Parameter Descriptions

All parameters for the WITH/ALSO WITH/WITHOUT clause of DISPLAY/PUNCH are described in detail under SET OPTIONS Syntax (see page 36); these parameters are not repeated for each entity. An exception to this is when there are special considerations that apply to a specific entity type.

Descriptions of parameters presented in Chapter 4: (see page 67) are not repeated unless special considerations apply.

Verb Synonyms

Where the verbs ADD, MODIFY, and DELETE are used, their respective synonyms (CREATE, ALTER, and DROP) are assumed.

Default Values

Default values shown are for ADD and REPLACE statements.

MODIFY statements assume as defaults the parameters used in the ADD or most recent MODIFY statement.

Relationships between Clauses and Verbs

Note the following points about the relationships between specific clauses and specific verbs:

- Optional clauses can appear in ADD, MODIFY, and REPLACE statements, unless otherwise noted.
- Clauses that apply to only one verb are noted in the parameter description.
- Clauses that are required with ADD statements appear as optional in the syntax, but are noted as mandatory in the parameter description.

Entity Types Used Only with CA IDMS Products

Four of the entity types described in this chapter are used only with specific CA IDMS data management products; the entity types and products are as follows:

Entity Type	Product
MESSAGE	DC/UCF and CA ADS
PROCESS	CA ADS
QFILE	CA OLQ
TABLE	DC/UCF

Security

After securing the product through the SECURITY clause of the SET OPTIONS statement (see SET OPTIONS Statement (see page 34)), the database administrator can control entity-type security by verb, using the AUTHORITY clause in the USER statement.

If the SET OPTIONS statement specifies that security for a certain product is enabled (ON), the user must be assigned the proper authority to issue verbs and statements for related entities.

ATTRIBUTE

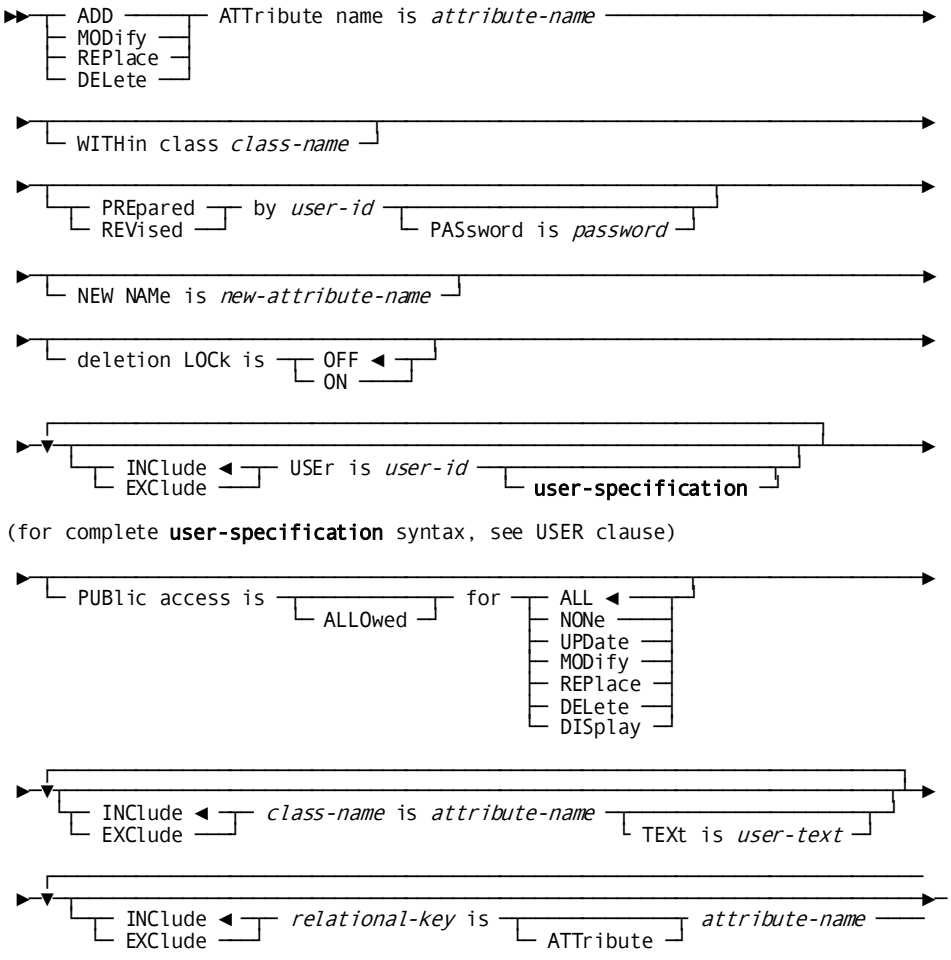
ATTRIBUTE statements are used to establish, maintain, delete, display, and punch attributes. Optional clauses:

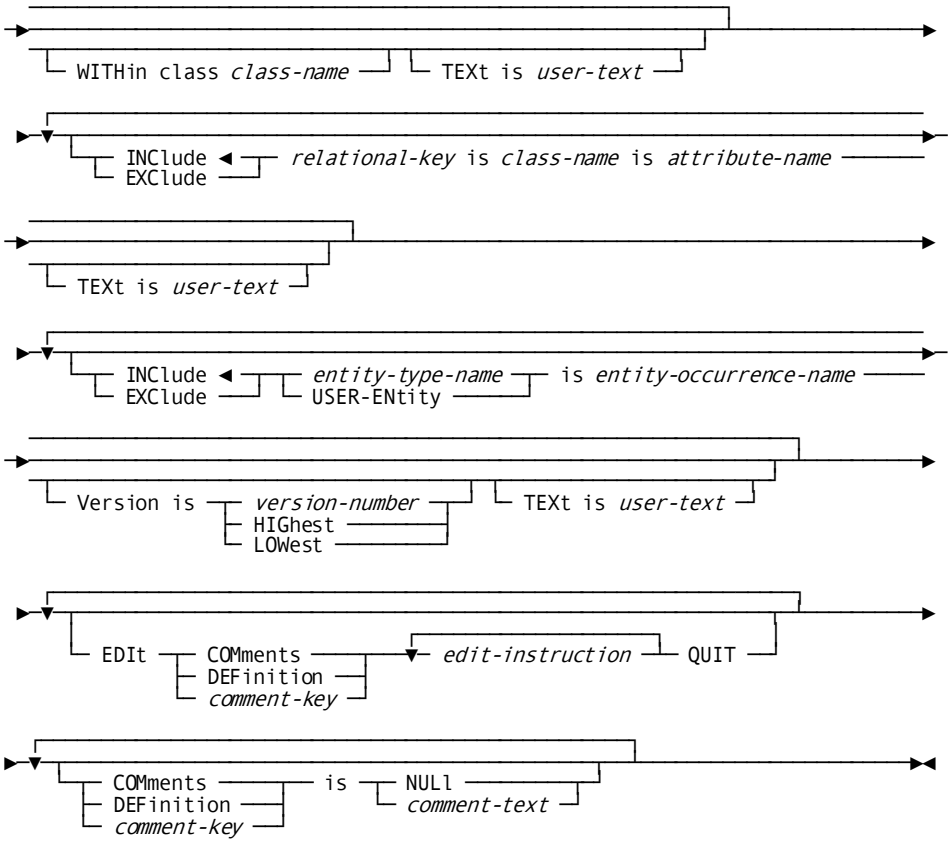
- Identify the user who created or revised the attribute
- Permit or prevent direct deletion of the attribute
- Relate one class and attribute to another class and attribute
- Control the association of an attribute with other attributes by means of relational keys

All entity-occurrence documentation options described in Chapter 3 are supported.

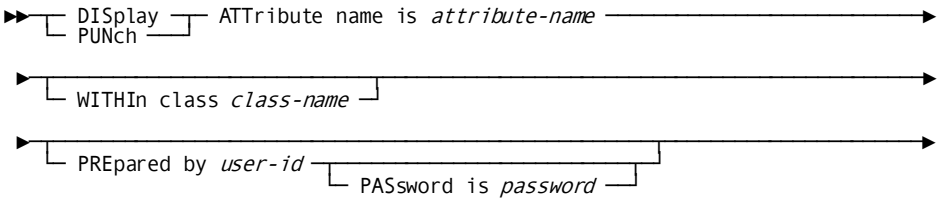
Syntax

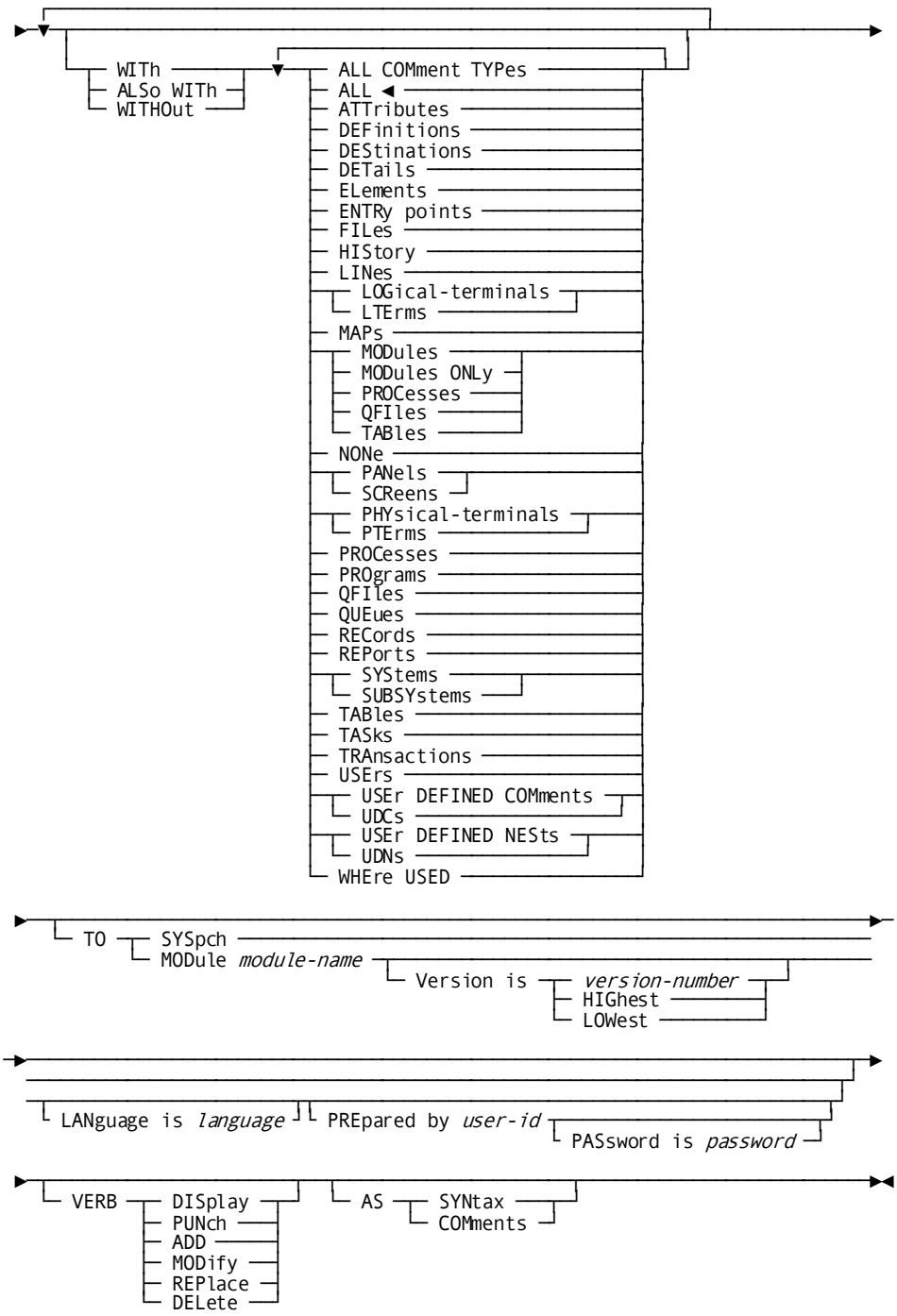
ATTRIBUTE Statement



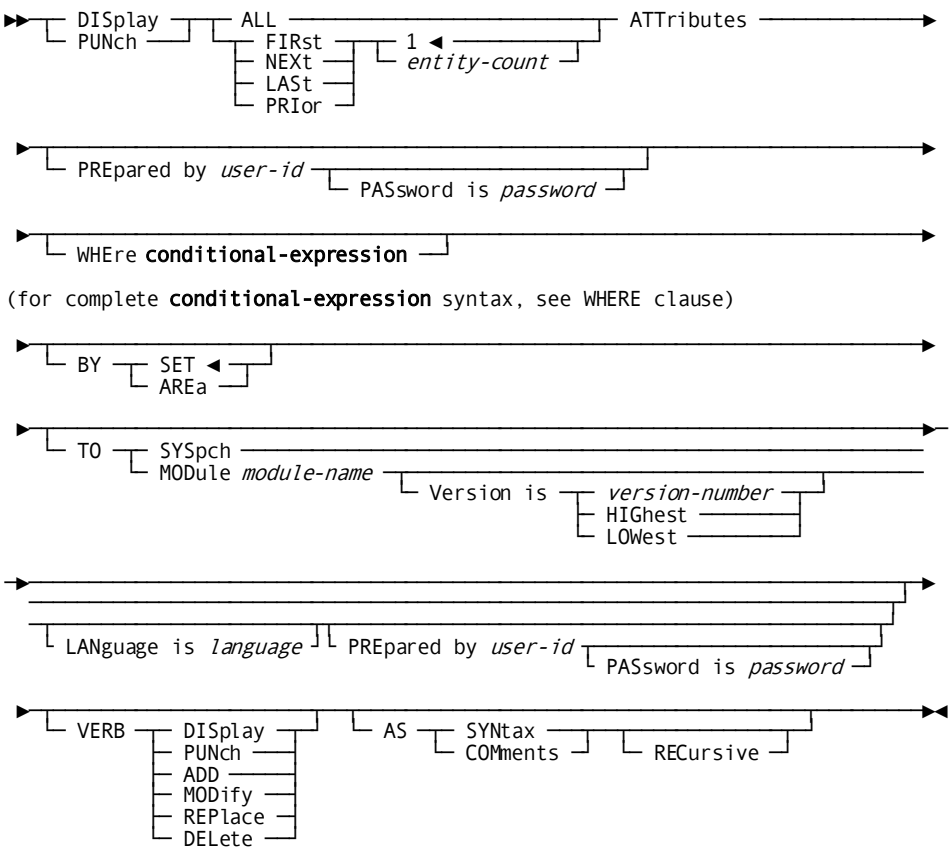


DISPLAY/PUNCH ATTRIBUTE Statement (for a single attribute)




WITh
ALSo WITh
WITHOut
ALL COMment TYPes
ALL
ATTributes
DEFinitions
DESTinations
DETails
ELements
ENTRy points
FILes
HISTory
LINes
LOGical-terminals
LTErms
MAPs
MODules
MODules ONLy
PROCesses
QFIles
TABles
NONe
PANels
SCReens
PHYsical-terminals
PTErms
PROCesses
PROgrams
QFIles
QUEues
RECords
REPorts
SYStems
SUBSYstems
TABles
TASks
TRAnsactions
USErs
USEr DEFINED COMments
UDCs
USEr DEFINED NESts
UDNs
WHEre USED
TO
SYSpch
MODule module-name
Version is
version-number
HIGhest
LOWest
LANguage is language
PREpared by user-id
PASsword is password
VERB
DISplay
PUNch
ADD
MODify
REPlace
DELete
AS
SYNtax
COMments

DISPLAY/PUNCH ATTRIBUTE Statement (for multiple attributes)



Parameters

ATTribute name is *attribute-name*

Identifies a new attribute to be established within an existing class or an existing attribute to be modified, replaced, deleted, displayed, or punched. *Attribute-name* must be a 1- through 40-character value that is unique within the specified class.

WITHin class *class-name*

References an existing class. *Class-name* must be a 1- through 20-character alphanumeric value. WITHIN CLASS is a required parameter within ADD statements and is required within MODIFY, REPLACE, DELETE, DISPLAY, and PUNCH statements if the named attribute is not unique in the dictionary.

NEW NAMe is *new-attribute-name*

Specifies a new name for the requested attribute. *New-attribute-name* must conform to the rules for *attribute-name*, as described above. This clause changes only the name of the attribute; it does not alter or delete any previously defined relationships between the attribute and any class or entity. Subsequent references to the attribute must specify the new name. The attribute cannot be renamed if DELETION LOCK IS ON is specified.

deletion LOCk is

Allows or disallows the deletion or renaming of named attributes.

OFF

Permits the user to delete or rename the named attribute. Attributes within the predefined classes LANGUAGE and MODE cannot be deleted if they are connected to any other entity, even if the deletion lock is off. OFF is the default.

ON

Prohibits the user from deleting or renaming the named attribute. If DELETION LOCK IS ON is specified, a MODIFY ATTRIBUTE statement specifying DELETION LOCK IS OFF must be submitted before the attribute can be deleted or renamed.

relational-key* is ATTribute *attribute-name

Associates the named attribute with another attribute through a previously defined relational key. The keyword ATTRIBUTE is required only if *attribute-name* matches an existing class name.

WITHIn class *class-name*

Specifies a class name when the named attribute participates in more than one class; *class-name* must reference an existing class.

relational-key* is *class-name* is *attribute-name

Associates the named attribute with another attribute through a previously defined relational key.

entity-type-name*/USER-ENtity is *entity-occurrence-name

Relates the named attribute directly to the specified occurrence of the named entity type. USER-ENTITY relates the attribute to the specified user as an attribute of that user.

WITh/ALSo WITh/WITHOut

Includes or excludes the specified options when the named attribute is displayed or punched. Detailed information for each DISPLAY/PUNCH option is under SET OPTIONS Syntax (see page 36). The options that are listed below present special considerations for this entity type.

DETails

Includes the DELETION LOCK specification.

ATTributes

Includes all attributes with which the named attribute is associated. Because attributes can be connected to many entities, a DISPLAY WITH ATTRIBUTES request can generate substantial output.

Usage

Considerations for assigning attributes

The following considerations apply to assigning attributes to a class:

- ADD ATTRIBUTE statements are used to define the attributes for each class that has been assigned the ATTRIBUTES ARE MANUAL qualifier. If a class has been assigned the ATTRIBUTES ARE AUTOMATIC qualifier, its attributes are added to the dictionary automatically the first time that the DDDL compiler encounters an undefined attribute within the class/attribute clause of an entity-type statement. See Attribute/Entity Relationships (see page 109) for further details.
- If the SET OPTIONS statement specifies SECURITY FOR CLASS AND ATTRIBUTE IS ON, the user must be assigned the proper authority to issue ATTRIBUTE statements.
- When a class is deleted, all attributes owned by that class are also deleted, regardless of the delete locks on the attributes or the user authority for the attributes.

If you specify REPLACE

If you specify REPLACE, the DDDL compiler initializes to defaults and/or excludes the following options:

- Related attributes
- USER REGISTERED FOR
- PUBLIC ACCESS
- COMMENTS/DEFINITION/*comment-key*

The following relationships are not affected:

- Attributes to which the named attribute is related
- Entities associated with the named attribute

Examples

Assuming that class ENTITY-STATUS and user DBA exist in the dictionary, the following statement defines attribute DESIGN within ENTITY-STATUS, sets the attribute deletion lock on, and supplies comment text.

```
add attribute design within class entity-status
    prepared by dba password is 'ice 9'
    deletion lock is on
    comments 'designates design occurrences'.
```

The following statement modifies the definition of the attribute DESIGN, disabling the deletion lock in order to rename the attribute.

```
modify attribute design
    revised by dba password is 'ice 9'
    deletion lock is off
    new name is proposed
    deletion lock is on
    comments 'designates proposed occurrences'.
```

The following statement requests the DDDL compiler to display the attribute PRODUCTION along with any programs to which PRODUCTION is related.

```
display attribute production
    with programs.
```

The following statements request the DDDL compiler to disable the deletion lock and to delete the attribute PROPOSED.

```
modify attribute proposed
     revised by dba password is 'ice 9'
     deletion lock is off.

delete attribute proposed.
     prepared by dba
         password is 'ice 9'.
```

The following statement defines the attribute LAUREL PIPPEN within class STUDENT and relates Laurel Pippen to the Hartwell School by means of a class/attribute structure (SCHOOL is the class; HARTWELL is the attribute).

```
add attribute 'laurel pippen' within class student
    school is hartwell.
```

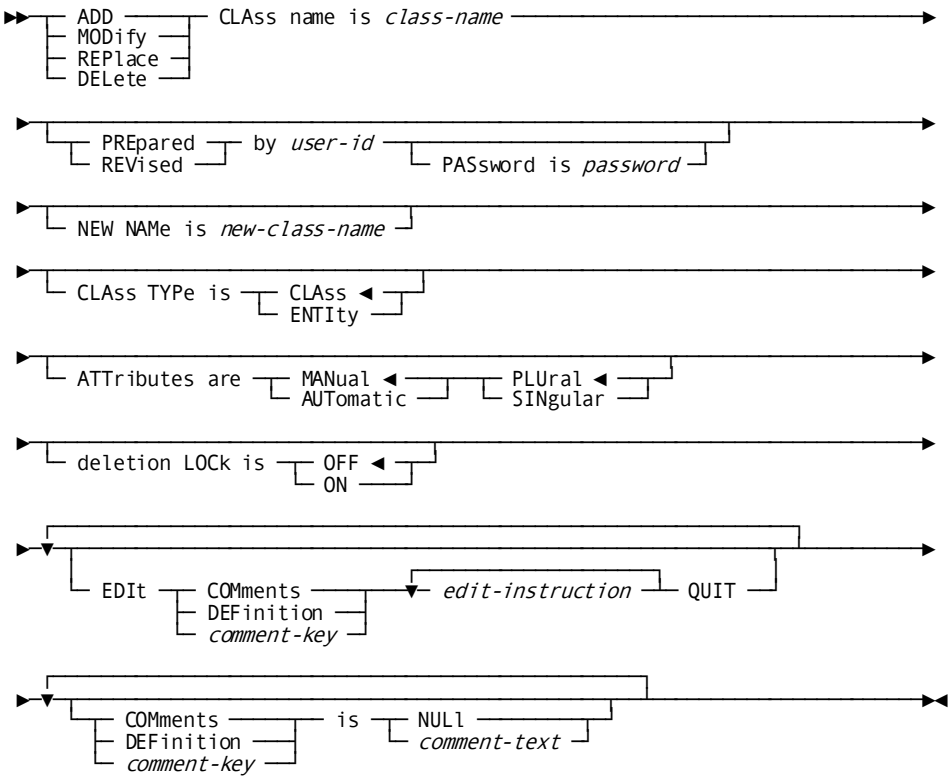
CLASS

CLASS statements are used to establish, maintain, replace, delete, display, and punch classes. Optional clauses are used to:

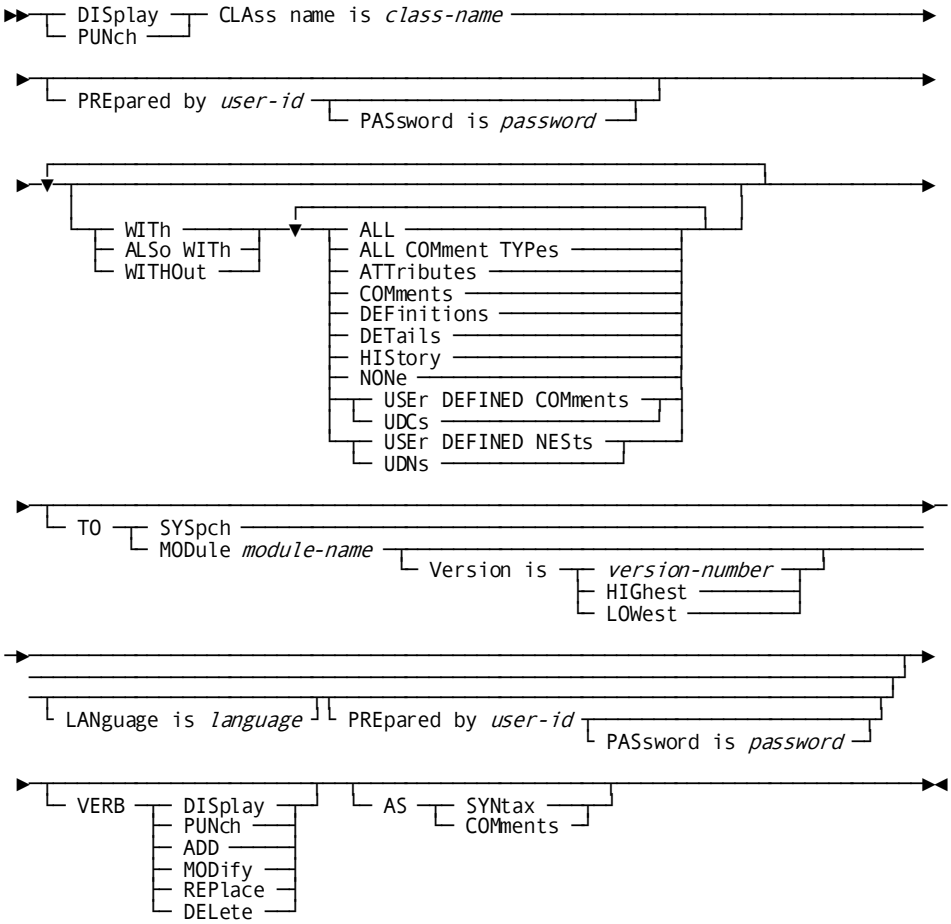
- Identify the user-origin of the class
- Determine if the class is to be directly established as a user-defined entity type
- Specify qualifiers that determine how the class's attributes are to be added to the dictionary and that govern how many attributes can be related to each entity occurrence.
- Permit or prevent the direct deletion or renaming of classes
- Include, delete, or edit comment text

Syntax

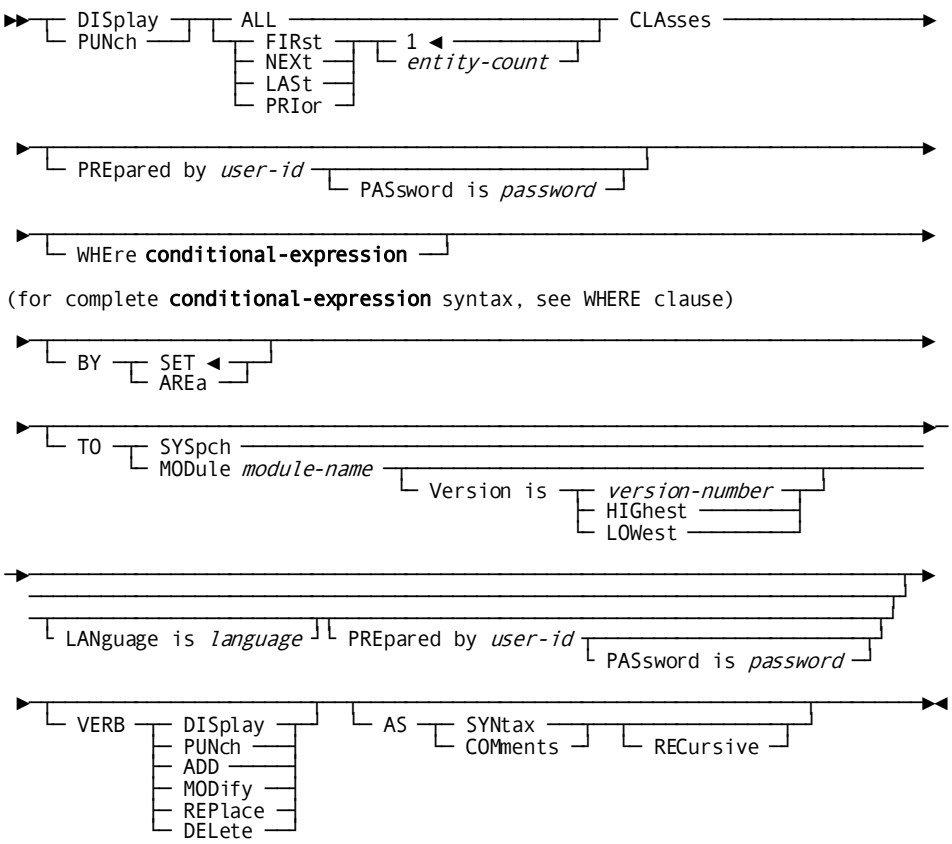
CLASS Statement



DISPLAY/PUNCH CLASS Statement (for a single class)



DISPLAY/PUNCH CLASS Statement (for multiple classes)



Parameters

CLAss name is *class-name*

Identifies a new class to be established in the dictionary or an existing class to be modified, replaced, deleted, displayed, or punched. *Class-name* must be a 1-through 20-character name that does not duplicate an existing class name.

NEW NAMe is *new-class-name*

Specifies a new name for the requested class. *New-class-name* must conform to the rules for *class-name* (described above). This clause changes only the name of the requested class; it does not alter or delete any previously defined attributes or attribute/entity relationships within the class. Subsequent references to the class must specify the new name. If DELETION LOCK IS ON is specified, the DDDL compiler will not process the NEW NAME clause.

CLASs TYPe is

Determines whether class is established as a class or as a user-defined entity type in the dictionary.

CLAss

Class is established as a class in the dictionary. This is the default.

ENTIty

Class is established as a user-defined entity type in the dictionary. This option allows the user to define occurrences of the entity by using the user-defined entity statement as described under USER-DEFINED ENTITY later in this chapter.

ATTributes are

Assigns qualifiers to attributes associated with the named class.

MANual

Specifies that attributes must be added to the dictionary explicitly by using the ADD ATTRIBUTE statement.

AUTomatic

Specifies that attributes are added to the dictionary automatically when they are named in a class/attribute clause within an entity-type statement.

PLUral

Specifies that multiple attributes can be related to an entity occurrence. PLURAL is the default.

SINgular

Specifies that only one attribute can be related to an entity occurrence.

deletion LOCk is

Controls the class deletion lock.

OFF

Permits the user to delete or rename the named class. OFF is the default. Even if DELETION LOCK IS OFF is specified, the predefined classes LANGUAGE and MODE cannot be deleted if any attributes exist within those classes.

ON

Prohibits the user from deleting or renaming the named class. If DELETION LOCK IS ON is specified, a MODIFY CLASS statement specifying DELETION LOCK IS OFF must be processed before the named class can be deleted or renamed.

WITh/ALSo WITh/WITHOut

Includes or excludes the specified options when the named class is displayed or punched. Detailed information for each DISPLAY/PUNCH option is under SET OPTIONS Syntax (see page 36). The option that is listed below presents special considerations for this entity type.

DETails

Includes the DELETION LOCK, ATTRIBUTES ARE, and CLASS TYPE specifications.

Usage

Considerations

The following considerations apply to using CLASS statements:

- If the SET OPTIONS statement specifies SECURITY FOR CLASS AND ATTRIBUTE IS ON, the user must be assigned the proper authority to issue CLASS statements.
- When a class is deleted, all attributes owned by that class are also deleted, regardless of the delete locks on the attributes or the user authority for the attributes.

If you specify REPLACE

If you specify REPLACE, the DDDL compiler initializes to defaults and/or excludes the following option:

- USER REGISTERED FOR
- PUBLIC ACCESS
- COMMENTS/DEFINITION/*comment-key*

Attributes associated with the named class are not affected.

If you specify DELETE

If you specify DELETE, the DDDL compiler deletes the requested class and all attributes owned by that class, regardless of the delete locks on or the user authority for the attributes.

Examples

The following statement adds the class ENTITY-STATUS with the attribute qualifiers of manual and singular and sets the deletion lock on.

```
add class entity-status
    prepared by dba password is 'ice 9'
    deletion lock is on
    attributes are manual singular.
```

The following statements add the class ENTITY-TYPE (by default, the class is assigned the manual and plural qualifiers and the deletion lock is turned off) and modify the class name and default attributes qualifier and deletion lock specifications.

```
add class entity-type
    prepared by dba password is 'ice 9'.

modify class entity-type
     revised by dba password is 'ice 9'
     new name is occurrence-type
     deletion lock is on
     attributes are singular.
```

The following statement adds the class COURSE and assigns it a class type of ENTITY and the automatic and plural qualifiers.

```
add class course
    prepared by dba password is 'ice 9'
    class type is entity
    attributes are automatic plural.
```

DESTINATION

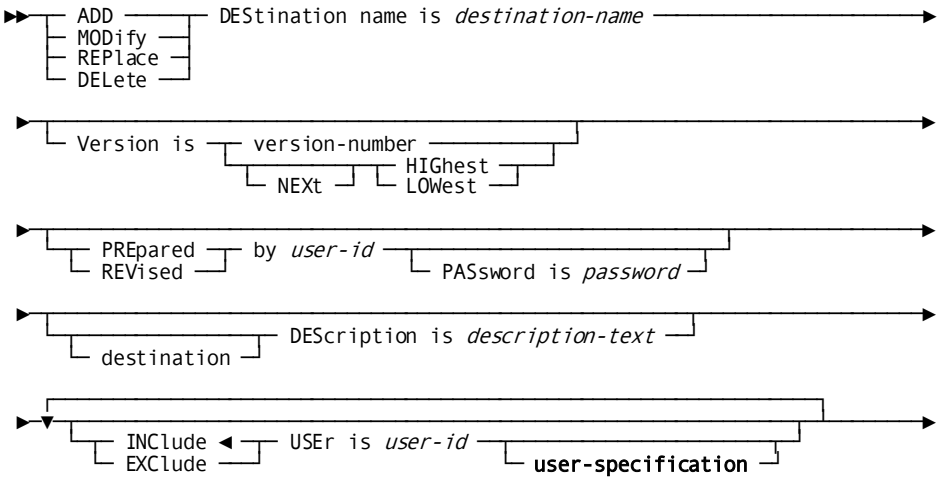
DESTINATION statements are used to document groups of users or logical terminals as a single logical destination within a teleprocessing system. The inclusion of a logical destination in a DC/UCF system permits the routing of a message simultaneously to all users or logical terminals that are included in the destination definition.

Note: It is recommended that you maintain DESTINATION definitions using the system generation compiler, *not* the DDDL compiler. If a system generation component is processed by the DDDL compiler, only dictionary security is checked, *not* system generation security. For more information on using the system generation compiler, refer to *CA IDMS System Generation Guide*.

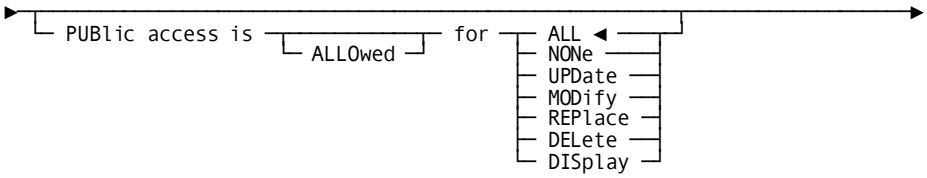
If the SET OPTIONS statement specifies SECURITY FOR IDMS-DC IS ON, the user must be assigned the proper authority to issue DESTINATION statements.

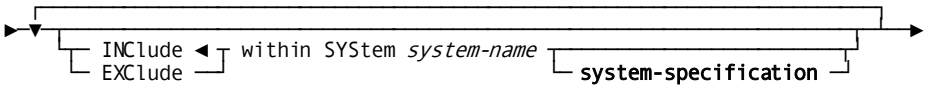
Syntax

DESTINATION Statement

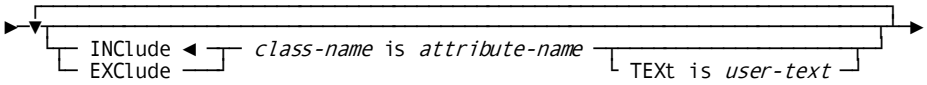


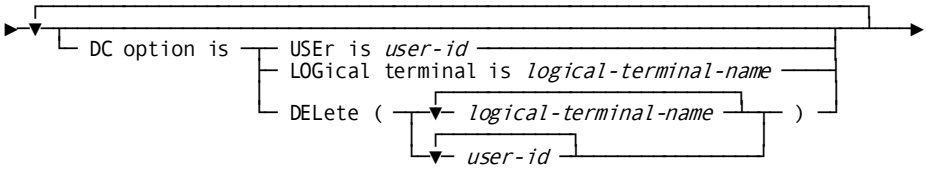
(for complete **user-specification** syntax, see USER clause)

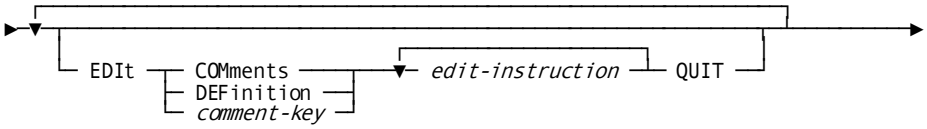


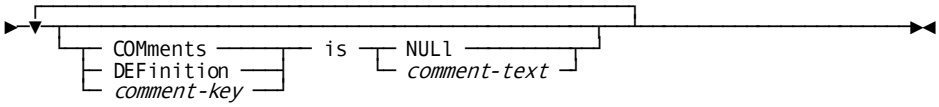


(expanded **system-specification** syntax follows this syntax diagram)

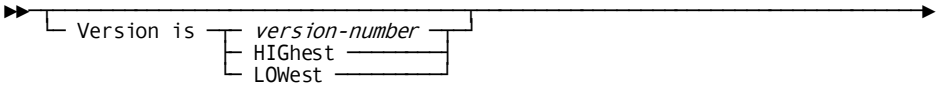


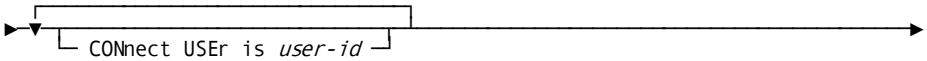


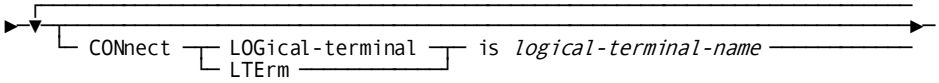


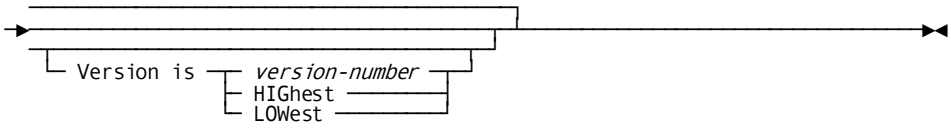


Expansion of *system-specification*

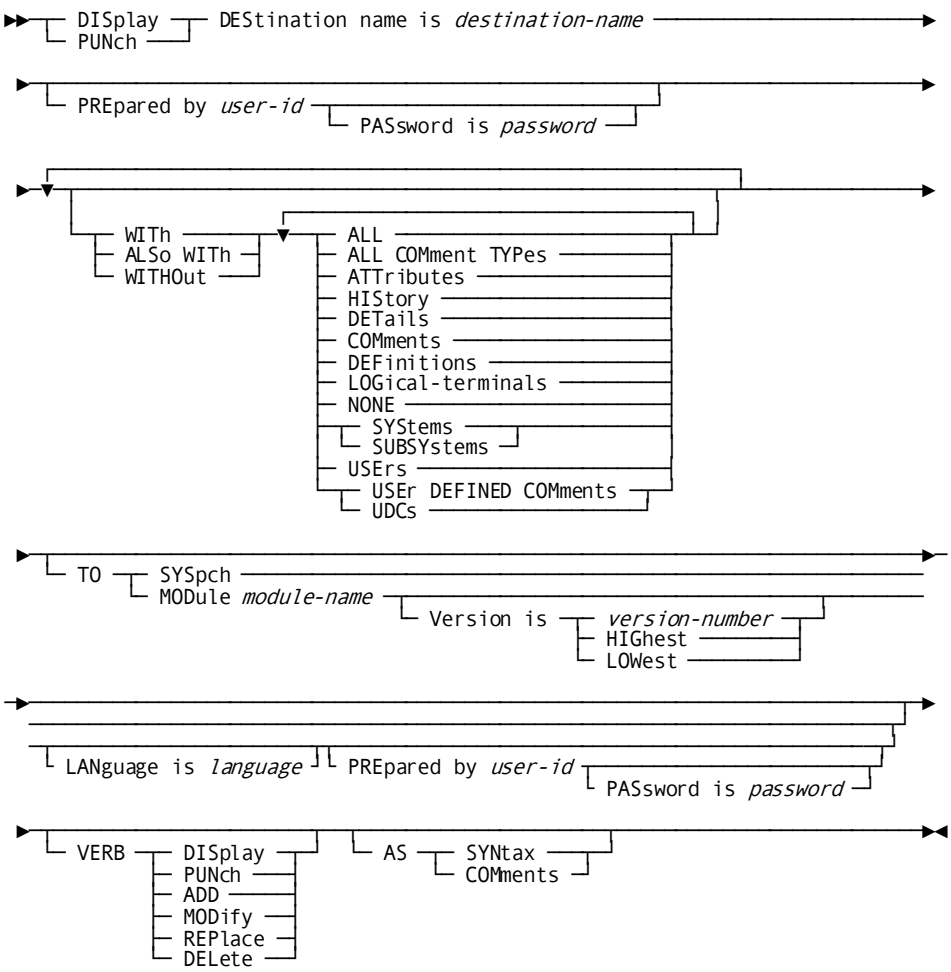




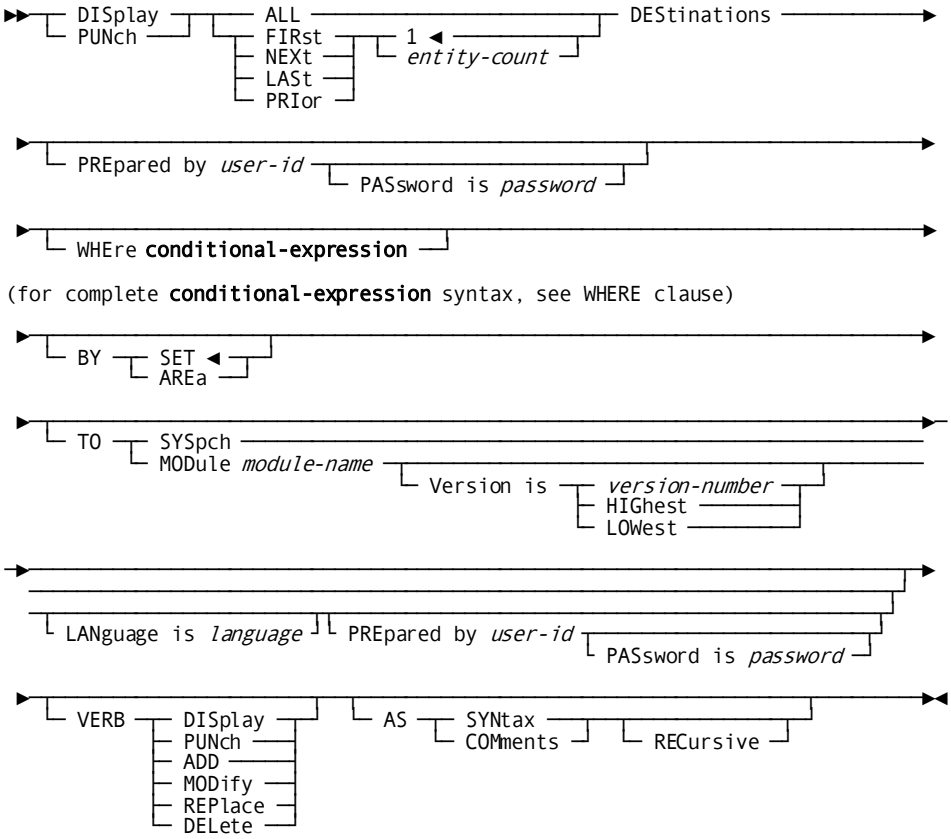




DISPLAY/PUNCH DESTINATION Statement (for a single destination)



DISPLAY/PUNCH DESTINATION Statement (for multiple destinations)



Parameters

DEStination name is *destination-name*

Identifies a new destination to be established in the dictionary or an existing destination to be modified, replaced, deleted, displayed, or punched. *Destination-name* must be a 1- through 8-character alphanumeric value.

within SYStem *system-name*

Associates the named destination with (INCLUDE) or disassociates it from (EXCLUDE) the specified system and defines the users or logical terminals that constitute the destination for that system. *System-name* must be the 1- through 32-character name of an existing system.

If EXCLUDE is specified without a CONNECT specification, the compiler removes the destination/system relationship and any dependent user or logical-terminal associations.

WITHIN SYSTEM is documentation only, unless the system generation compiler COPY facility is to be used to copy destination definitions from an IDD-built system. When the COPY facility is not used, destination/system relationships are established and maintained by the system generation compiler. DESTINATION statements cannot modify or delete destination definitions copied into DC/UCF systems by the system generation compiler.

The WITHIN SYSTEM clause can be repeated to establish additional destination/system relationships.

CONnect USEr is *user-id*

Associates a user with the destination/system relationship. *User-id* must reference an existing user in the dictionary. In DC/UCF environments, CONNECT is documentation only; the functional relationship must be established with the system generation compiler.

CONnect LOGical-terminal is *logical-terminal-name*

Associates a logical terminal with the destination/system relationship. *Logical-terminal-name* must reference an existing logical terminal that is already associated with the named system. In DC/UCF environments, CONNECT is documentation only; the functional relationship must be established with the system generation compiler.

DC option is

Directs the system generation compiler to establish a destination/user or destination/logical terminal relationship when it copies the named destination into a DC/UCF system, and defines or deletes the users or logical terminals.

USEr is *user-name*

Specifies one or more users that constitute the destination. *User-name* must reference an existing user in the dictionary.

LOGical terminal is *logical-terminal-name*

Specifies one or more logical terminals that constitute the destination. *Logical-terminal-name* must reference an existing logical terminal in the dictionary.

DELete *logical-terminal-name/user-name*

Deletes the specified list of logical terminals or users from the destination. Multiple logical-terminal/user names must be separated by a comma *and* one or more blanks.

WITh/ALSo WITh/WITHOut

Includes or excludes the specified options when the named destination is displayed or punched. Detailed information for each DISPLAY/PUNCH option is under SET OPTIONS Syntax (see page 36). The option that is listed below presents special considerations for this entity type.

DETails

Includes the DESCRIPTION IS and DC OPTION clauses.

Usage

If you specify REPLACE

If you specify REPLACE, the DDDL compiler initializes to defaults and/or excludes the following options:

- DESCRIPTION
- USER REGISTERED FOR
- PUBLIC ACCESS
- COMMENTS/DEFINITION/*comment-key*
- WITHIN SYSTEM
- Related attributes

The WITHIN SYSTEM and DC OPTIONS specifications are replaced only if they have been established by the DDDL compiler. The following relationships established by the system generation compiler are not affected:

- Systems and connected users/logical terminals
- Users, logical terminals, and printers constituting the destination

Example

In the following example, the ADD statement associates destination OEBOST with the online system INVENTORY. OEBOST comprises logical terminals LTR22, LTR23, and LTR24. The MODIFY statement disassociates destination OEBOST from the INVENTORY system and defines the logical terminals as components of a DC/UCF system.

```
add destination oebost
    prepared by dba password is 'ice 9'
```

```
    description 'online order entry terminals -- boston'
    within system inventory
    connect logical-terminal is ltr22
    connect logical-terminal is ltr23
    connect logical-terminal is ltr24.

modify destination oebost
    revised by dgs
    description 'online order entry terminals -- boston'
    exclude system inventory
    dc-option is logical-terminal is ltr22
    dc-option is logical-terminal is ltr23
    dc-option is logical-terminal is ltr24.
```

ELEMENT

ELEMENT statements are used to define group or elementary data elements. Also known as *fields* and *data items*, elements can participate in records built by the DDDL compiler, by the CA IDMS schema compiler, or in maps built by the DC/UCF mapping compiler; elements can also exist independently in the dictionary. An element can have a maximum length of 32,767 characters.

Optional ELEMENT statement clauses allow the user to:

- Define element synonyms.
- Describe up to four alternate pictures for an element.
- Relate elements to users and to other elementary and group elements; the syntax supports attribute/entity relationships, all entity-occurrence documentation options, all 1974 ANSI COBOL picture options, and 31 digit zoned and packed decimal numerics.

Defining Group Element Occurrences

The user defines and maintains group element occurrences by means of SUBORDINATE ELEMENT clauses, which provide one method for documenting multiply-occurring or redefined elements/groups in the dictionary.

Modifying an Element Definition

When the user modifies an element definition, the DDDL compiler does not modify the definitions of records in which the named element participates. Record elements must be modified individually by using the RECORD ELEMENT substatement, described under RECORD (REPORT/TRANSACTION) (see page 287) later in this chapter.

Deleting Element Occurrences

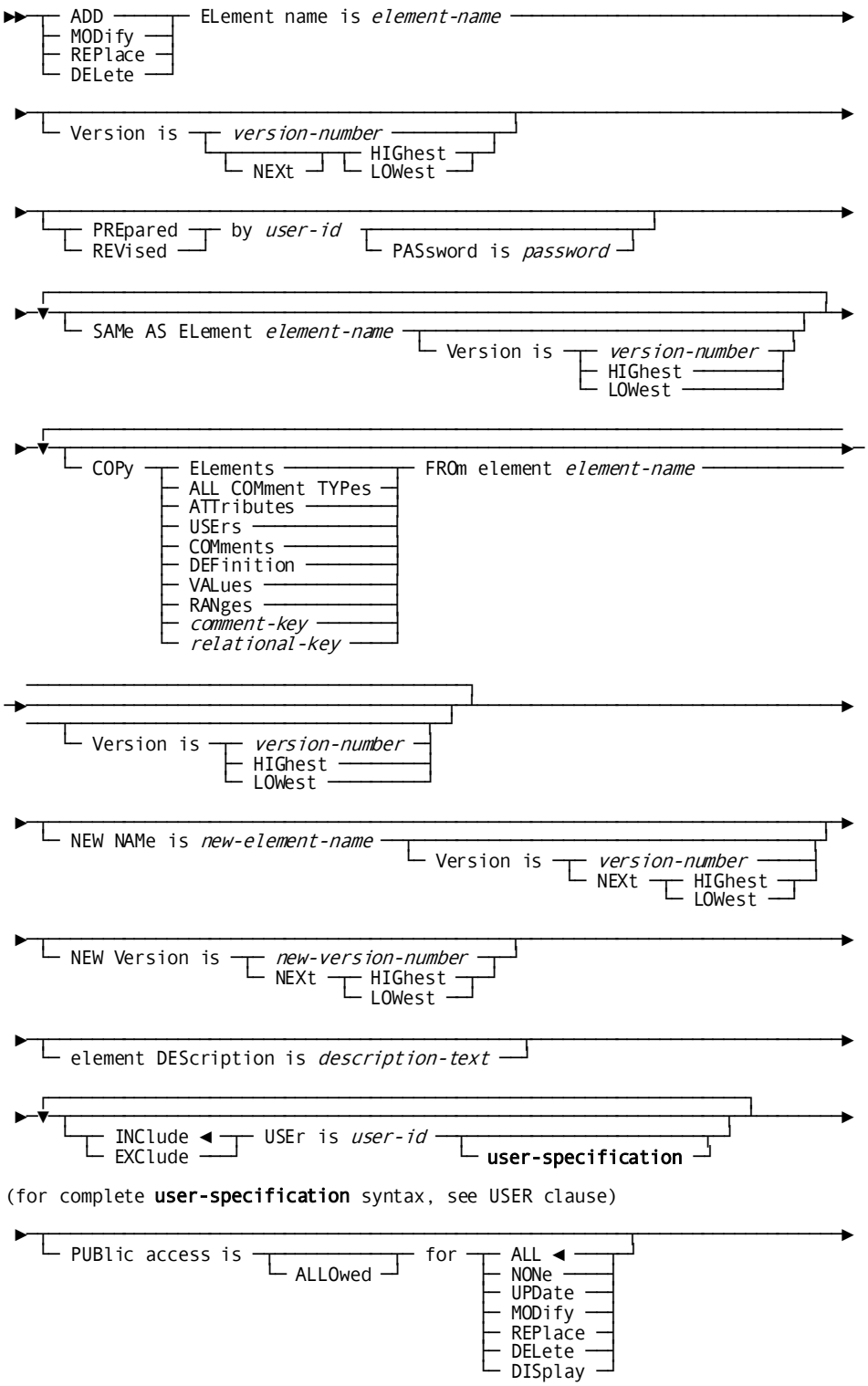
The user cannot delete element occurrences that are members of a group element structure or that participate in IDD- or schema-built records. To delete an elementary element, the user must first disassociate it from the group element; to delete a record element, the user must first delete its associated record or schema. To prevent the deletion of an element when the only record in which it participates is deleted, select the SET OPTIONS statement DELETE IS OFF option for the session; see SET OPTIONS Statement (see page 34) for further details.

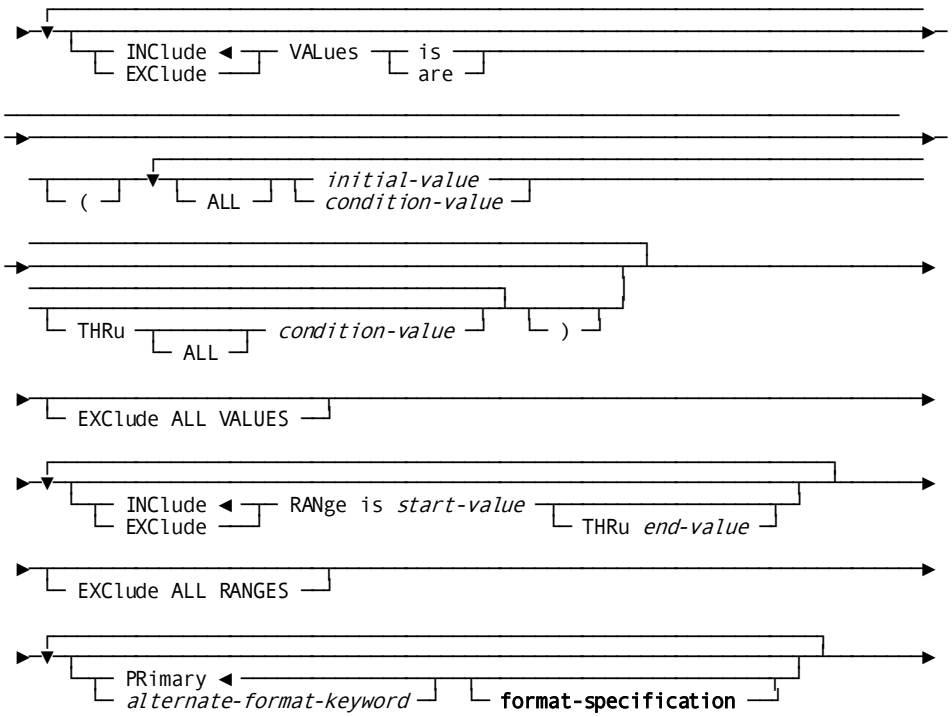
Required Authority

If the SET OPTIONS statement specifies SECURITY FOR IDD IS ON, the user must be assigned the proper authority to issue ELEMENT statements.

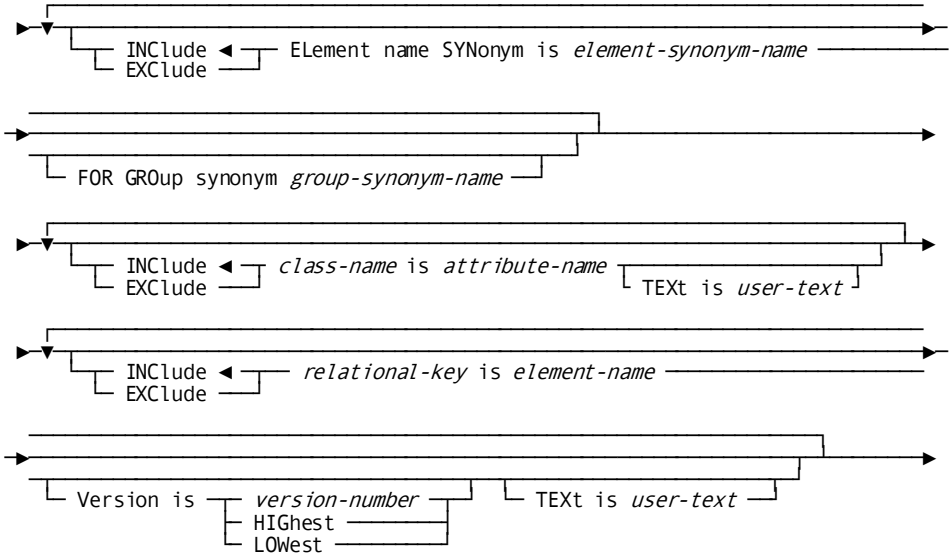
Syntax

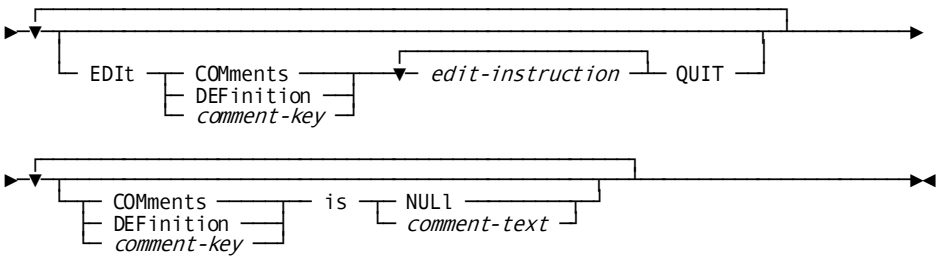
ELEMENT Statement


ADD
MODify
REPlace
DELete
ELement name is element-name
Version is
version-number
NEXt
HIGhest
LOWest
PREpared
REVised
by user-id
PASsword is password
SAMe AS ELement element-name
Version is
version-number
HIGhest
LOWest
COPy
ELements
ALL COMment TYPes
ATTributes
USErs
COMments
DEFinition
VALues
RANges
comment-key
relational-key
FROm element element-name
Version is
version-number
HIGhest
LOWest
NEW NAMe is new-element-name
Version is
version-number
NEXt
HIGhest
LOWest
NEW Version is
new-version-number
NEXt
HIGhest
LOWest
element DEScription is description-text
INClude
EXClude
USEr is user-id
user-specification
(for complete user-specification syntax, see USER clause)
PUBlic access is
ALLOwed
for
ALL
NONe
UPDate
MODify
REPlace
DELete
DISplay

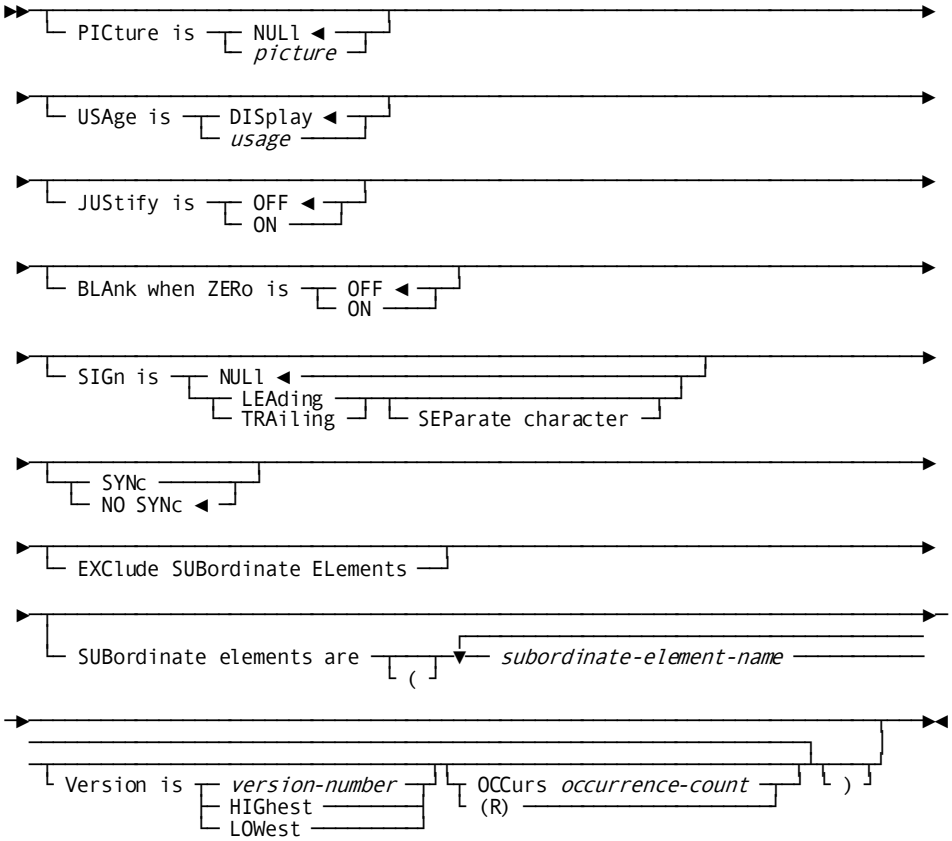


(expanded **format-specification** syntax follows this syntax diagram)

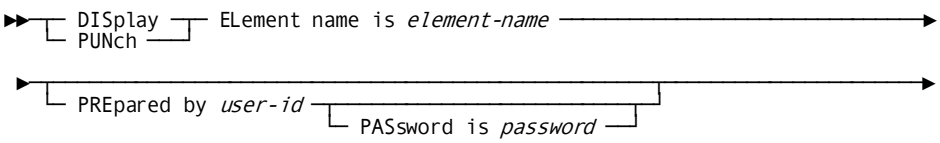


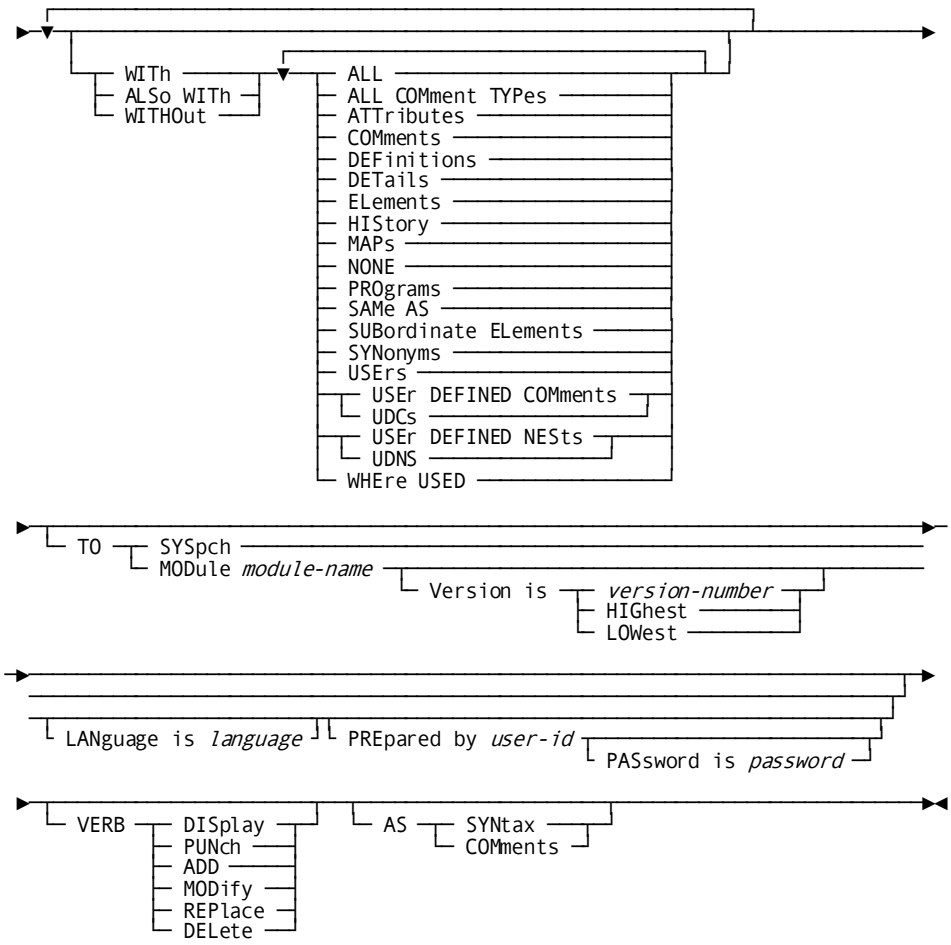


Expansion of *format-specification*

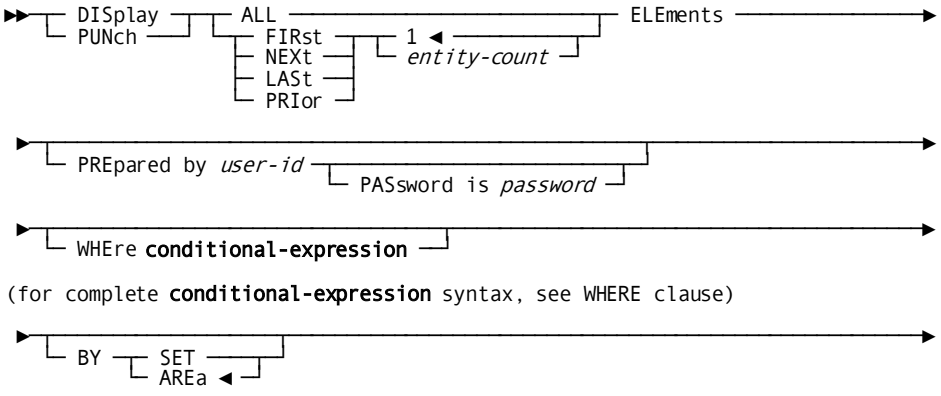


DISPLAY/PUNCH ELEMENT Statement (for a single element)

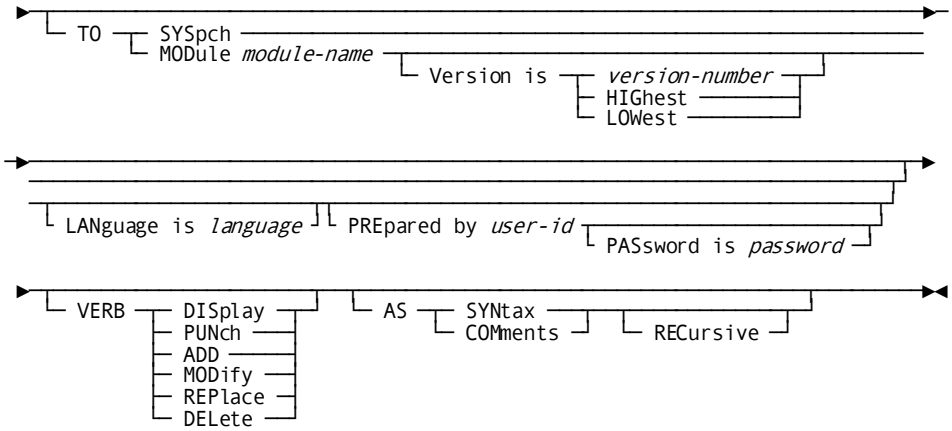




DISPLAY/PUNCH ELEMENT Statement (for multiple elements)



(for complete **conditional-expression** syntax, see WHERE clause)


TO
SYSpch
MODule module-name
Version is
version-number
HIGhest
LOWest
LANguage is language
PREpared by user-id
PASsword is password
VERB
DISplay
PUNch
ADD
MODify
REPlace
DELete
AS
SYNtax
COMments
RECursive

Parameters

ELement name is *element-name*

Identifies a new element to be established in the dictionary or an existing element to be modified, replaced, deleted, displayed, or punched. *Element-name* must be a 1- through 32-character alphanumeric value. (If you use *element-name* in a program, observe the maximum-character limit of the programming language.)

NEW NAMe is *new-element-name*

Specifies a new name for the requested element. If a version number is not specified, the version number defaults to the version associated with the element's original name. This clause changes only the name of the element occurrence; it does not alter or delete any relationships in which the element participates. Subsequent references to the requested element must specify the new name.

Note that if the element's primary synonym (the synonym that is identical to the primary name of the element) participates in a record, the element cannot be renamed.

NEW Version is

Specifies a new version number for the named element.

***new-version-number*/NEXt HIGHEST/NEXT LOWest**

Specifies a version number for the named element. The element name and new version number must not duplicate that of an existing element name and version number.

VALue is ALL *initial-value/condition-value*

Associates (INCLUDE) or disassociates (EXCLUDE) a value, range of values, or a list of values assigned to a COBOL level-88 condition name. A list of values must be enclosed by parentheses, with values separated by a space or comma. *Initial-value* and *condition-value* can be 1- through 32-character numeric literals, quoted literals, or figurative constants.

Note: If the SET OPTIONS statement specifies DECIMAL-POINT ISCOMMA, the DDDL compiler interprets a period in numeric literal as an insertion character, and a comma as a decimal point.

THRu ALL *condition-value*

Specifies multiple values and ranges of values when the element is a COBOL level-88 condition name.

EXClude ALL VALUES

Removes all VALUE clauses from the element definition. The keyword VALUES cannot be abbreviated. Typically, the EXCLUDE ALL VALUES clause is used to remove the values associated with an element in preparation for adding subordinate elements.

RANge is *start-value* THRu *end-value*

Specifies a normal or expected value or range of values for the named element. *Start-value* and *end-value* must be 1- through 32-character numeric literals or figurative constants. Values that contain delimiters or embedded blanks must be enclosed in site-standard quotation marks. The user can enter each acceptable value in a separate RANGE clause or enter a range of values in one clause that includes the THRU option. For example:

```
range is 1 range is 3 range is 5
```

or

```
range is 1 thru 5
```

The RANGE clause is documentation except in the CA ADS Batch environment. CA ADS Batch uses RANGE clause values to validate input data fields (see *CA ADS User Guide*).

EXClude ALL RANGES

Removes all range clauses from the element occurrence.

PRImary

Specifies the default format of the named element. The PRIMARY/*alternate-format-keyword* clause is used in conjunction with the RECORD statement FORMAT clause to determine the format of an element within a record occurrence. For information about the FORMAT clause, see RECORD (REPORT/TRANSACTION) (see page 287), later in this chapter.

alternate-format-keyword

Specifies a keyword that corresponds to an alternate format established previously in the ALTERNATE PICTURE KEYWORD clause of the SET OPTIONS statement (see SET OPTIONS Statement (see page 34)). Up to four alternate formats can be defined for each element. The PRIMARY/*alternate-format-keyword* clause is used in conjunction with the RECORD statement FORMAT clause to determine the format of an element within a record occurrence. For information about the FORMAT clause, see RECORD (REPORT/TRANSACTION) (see page 287), later in this chapter.

PICture is

Identifies the named element as an elementary element and specifies its length and data type.

NULl

Removes the element's PICTURE, USAGE, SIGN, JUSTIFY, and BLANK WHEN ZERO clauses and all associated subordinate elements. NULl is the default.

Note: The maximum length of an element (including its usage) is 32,767 characters.

picture

In the case of a named element which has been previously defined as a group element, PICTURE IS *picture* removes any associated subordinate elements except COBOL level-88 condition names, and *picture* becomes the picture for the resulting elementary element.

Picture must be a 1- through 30-character value that describes alphanumeric, alphabetic, numeric, or numeric-edited data, as shown in the table under the bold heading **Usage**, after this parameter list.

USAge is

Specifies the physical storage characteristics of the named element.

DISplay

Identifies the usage as alphanumeric, zoned decimal, edited, or display floating point. DISPLAY is the default.

Note: For CA ADS users, additional information about defining display floating point numerics is available in the *CA ADS Reference Guide*.

usage

Identifies one of the following usages:

- **BIT**— Bit string
- **POINTER**— Fullword address constant
- **CONDITION-NAME**—:i1.level-88 COBOL level-88 value
- **COMPUTATIONAL (COMP) (COMPUTATIONAL-4) (COMP-4) (BINARY)**— Binary
- **COMPUTATIONAL-1 (COMP-1) (SHORT-POINT)**—Short-precision floating point
- **COMPUTATIONAL-2 (COMP-2) (LONG-POINT)**—Long-precision floating point
- **COMPUTATIONAL-3 (COMP-3) (PACKED)**—Packed decimal

JUStify is

Defines COBOL justification specifications for the named element.

OFF

Specifies that a COBOL JUSTIFIED clause is not to be generated. OFF is the default.

ON

Specifies that a COBOL JUSTIFIED clause is to be generated.

BLAnk when ZERo is

Defines COBOL zero suppression requirements for the named element.

OFF

Specifies that a BLANK WHEN ZERO clause is not to be generated for COBOL. OFF is the default.

ON

Specifies that a BLANK WHEN ZERO clause is to be generated for COBOL.

SIGn is

Defines or deletes a sign specification for the named element.

NULl

Specifies that the sign and LEADING or TRAILING SEPARATE CHARACTER specification is removed from the named element.

LEAding/TRAiling

Specifies that a sign is associated with the named element. Further specifies that the sign appears in either the LEADING or the TRAILING position.

SEParate character

Reserves a separate character in the element definition for the sign designation.

SYNc

Specifies that the DDDL compiler must check the boundary alignment of COMP and COMP-4 record elements when the elements are included in a record. If a record element is not on a fullword or halfword boundary, the record is flagged in error.

NO SYNc

Specifies that the DDDL compiler will *not* check the boundary alignment of COMP and COMP-4 record elements when they are included in a record. NO SYNC is the default.

EXClude SUBordinate ELEments

Removes all subordinate elements, regardless of level number, from a group element. The PICTURE IS NULL specification (described above) ordinarily performs this function; however, if the subordinate elements are all level-88 condition names, the EXCLUDE SUBORDINATE ELEMENTS must be used. To replace existing subordinate elements with new subordinate elements, use the SUBORDINATE ELEMENTS clause (described below).

SUBordinate ELements are *subordinate-element-name*

Specifies that the named element is a group element and identifies one or more subordinate elements. *Subordinate-element-name* is the primary name of a subordinate element that exists in the dictionary. The SUBORDINATE ELEMENTS clause can be repeated to define a group structure of any size. Note that a list of subordinate-element names can be enclosed in parentheses to eliminate errors that can occur if an element name matches a DDDL keyword.

To define a filler as a subordinate element, specify an element name of 'FIL *nnnn*'; *nnnn* must be a 4-digit numeric value (leading zeros are required) and must be separated from the keyword FIL by one space. For example, to generate a filler described as FILLER PIC X(7), specify SUBORDINATE ELEMENT 'FIL 0007'. Note that filler fields need not exist in the dictionary in order to be included in group elements or records. An element can be named only once in a SUBORDINATE ELEMENTS clause; however, fillers can appear as often as required.

Each subordinate element can be qualified with either the OCCURS or R parameter.

OCCurs *occurrence-count*

Specifies that the subordinate element is a multiply-occurring element. *Occurrence-count* must be an integer in the range 1 through 32,767.

(R)

Redefines the previously named subordinate element. NOTE A redefined element cannot be defined with an OCCURS clause, however, it can be subordinate to an element defined with an OCCURS clause.

NOTE If a subordinate element must be defined with both an OCCURS and a REDEFINES clause, include the (R) option within the named element definition and the OCCURS option with the definition of the record in which the subordinate element participates.

ELement name SYNonym is *element-synonym*

Associates (INCLUDE) an alternative name with or disassociates it (EXCLUDE) from the element. *Element-synonym* must be a 1- through 32-character alphanumeric value.

All elements have at least one synonym, known as the primary synonym, with the same name and version as the element itself.

FOR GROup synonym *group-element-name*

Specifies that when the named element appears in a record as part of a group element whose name matches *group-element-name*, the specified element synonym is automatically used in the record. The following considerations apply:

- The group element need not exist in the dictionary.
- *Group-element-name* must identify a group element at the level immediately above *element-synonym*.
- If *element-synonym* is the highest level element in a record, *group-element-name* identifies the record in which the element participates.
- *Group-element-name* can be the primary name of a record or group, or a synonym.

WITh/ALSo WITh/WITHOut

Includes or excludes the specified options when the named element is displayed or punched. Detailed information for each DISPLAY/PUNCH option is under SET OPTIONS Syntax (see page 36). The options that are listed below present special considerations for this entity type.

DETails

Includes the DESCRIPTION, VALUE, and RANGE specifications and PICTURE-related information for primary and alternative formats, including SUBORDINATE ELEMENTS specifications.

ELements

Includes all user-defined nests and subordinate elements.

MAPs

Includes the name of the map associated with the element occurrence or any of its synonyms.

PROgrams

Includes the name of the program and record associated with an element occurrence or any of its synonyms.

SYNonyms

Displays record synonyms associated with element synonyms.

Usage

Specifying a **picture** variable

Picture must be a 1- through 30-character value that describes the types of data shown in the following table.

Category	Character	Description
Alphanumeric data	**X**	Represents one alphanumeric character. If USAGE IS BIT, X represents one bit; the USAGE clause is described in the parameters list.
	(n) An integer in parentheses after an X	Represents *n* repetitions of the alphanumeric character; for example, X(4) is equivalent to XXXX.
Alphabetic data	**A**	Represents one alphabetic character (A-Z).
	(n) An integer in parentheses after an A	Represents *n* repetitions of the alphabetic character
Numeric data	**9**	Represents one numeric character.
	(n) An integer in parentheses after a 9	Represents *n* repetitions of the numeric character.
	V	Represents an assumed decimal point. No more than one V can appear in an element picture. If the V is omitted and the P option (described below) is not used, the assumed decimal point is after the rightmost 9.

Category	Character	Description
	P	Represents an assumed zero. Any number of Ps can appear in the leftmost or the rightmost positions of an element picture. An assumed decimal point is automatically placed before the first P or after the last P. The character P does not occupy a storage position (for example, PP9999 has a data length of 4).
	S	Identifies the number as positive or negative. When used, the S must be the first character in the element picture. When the S is omitted, values for the element description are assumed to be positive.
Numeric-edited data (Includes the numeric data characters described above, along with the editing characters shown at the right)	Z + ' B CR - 0 DB * $.	Represent edit symbols used in reporting data; quotation marks are not required. Refer to the appropriate programming language manual for the individual interpretations of these symbols. If the SET OPTIONS statement specifies DECIMAL-POINT IS COMMA, a period (.) is interpreted as an insertion character and a comma (,) is interpreted as a decimal point.

If you specify REPLACE

If you specify REPLACE in the ELEMENT statement, the DDDL compiler initializes to defaults and/or excludes the following options:

- DESCRIPTION
- USAGE
- PICTURE
- JUSTIFY
- BLANK WHEN ZERO
- USER REGISTERED FOR
- PUBLIC ACCESS
- Related attributes
- VALUE
- RANGE
- ELEMENT NAME SYNONYM
- COMMENTS/DEFINITION/*comment-key*
- SIGN
- SUBORDINATE ELEMENTS
- Alternate picture formats
- Related elements

The following relationships are not affected:

- Group elements in which the named element participates
- Records in which the named element participates

Examples

In the following example, the first three ADD statements define the elements DISCONTINUE-MONTH, DISCONTINUE-DAY, DISCONTINUE-YEAR, documenting the normal range of values for DISCONTINUE-MONTH as 01 through 12 and for DISCONTINUE-DAY as 01 through 31. The SET OPTIONS statement establishes a default PREPARED BY specification for the session. The fourth ADD statement establishes the group element DISCONTINUE-DATE and names its subordinate elements; the last ADD statement establishes an element called DISC-DATE-X.

```
add element discontinue-month
    prepared by dba password is 'ice 9'
    picture 99
    range is 01 thru 12.
```

```
set options for session
    prepared by dba password is 'ice 9'

add element discontinue-day
    picture 99
    range is 01 thru 31.

add element discontinue-year
    picture 99.

add element discontinue-date
    subordinate elements
         discontinue-month
         discontinue-day
         discontinue-year.

add element disc-date-x
    picture x(6).
```

In the following example, the first two ADD statements define the elements LOWER-LIMIT and QUANTITY-ON-HAND. The third statement adds the group element HISTORY and assigns four subordinate elements; (R) indicates that DISC-DATE-X (established in the previous example) redefines DISCONTINUE-DATE. DISCONTINUE-DATE is subordinate to HISTORY and is also a group element.

```
add element lower-limit
    prepared by dba password is 'ice 9'
    picture 999.

add element quantity-on-hand
    prepared by dba password is 'ice 9'
    picture 9(4).

add element history
    subordinate elements
         discontinue-date
         disc-date-x (r)
         lower-limit
         quantity-on-hand.
```

The following example illustrates usage of the PRIMARY/*alternate-format* clause to define alternate formats. SET OPTIONS establishes three alternate picture keywords; the definition of element GROSS-PAY includes three alternate formats. Element GROSS-PAY is associated with the WS-SALARY-HISTORY record; the FORMAT clause determines the format of the elements.

```
set options for dictionary
  first alternate picture keyword is edit-pic
  second alternate picture keyword is report-pic
  third alternate picture keyword is screen-pic.
```

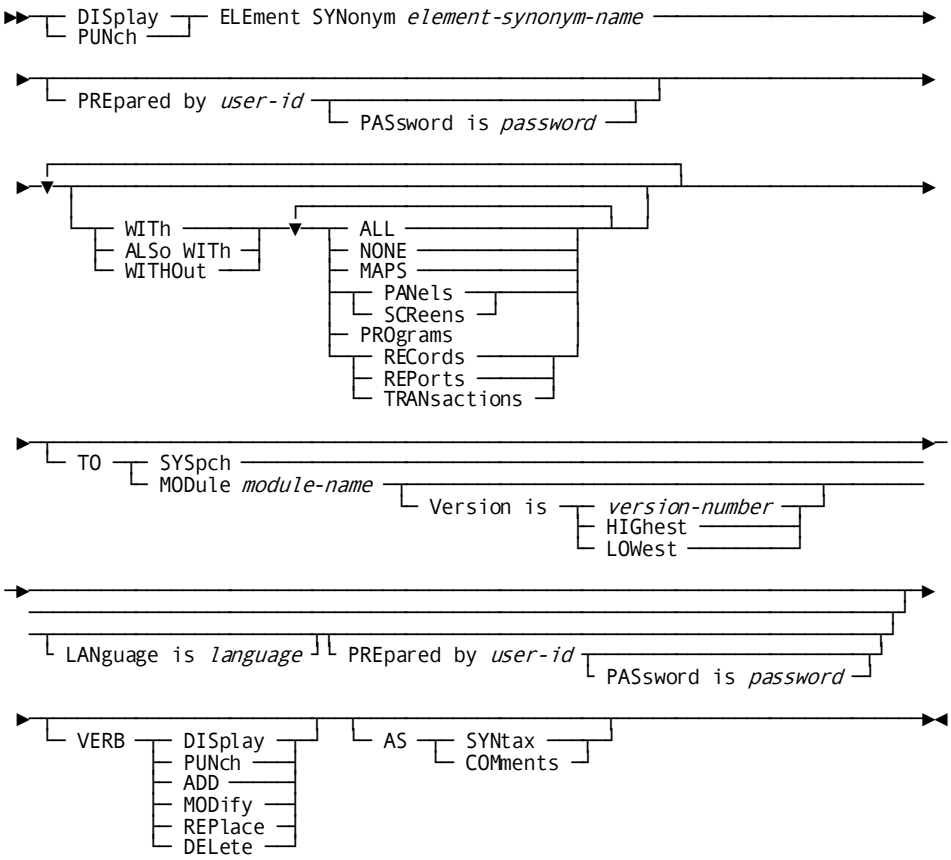
```
add element name is gross-pay
   primary
       picture is s9(7)v99
       usage is comp-3
   edit-pic
       picture is x(9)
       usage is display
   report-pic
       picture is z,zzz,zzz.99
   screen-pic
       picture is 9(7)v99
       blank when zero is on.

add record name is ws-salary-history
   format is edit-pic.
   record element is gross-pay.
   .
   .
   .
```

ELEMENT SYNONYM

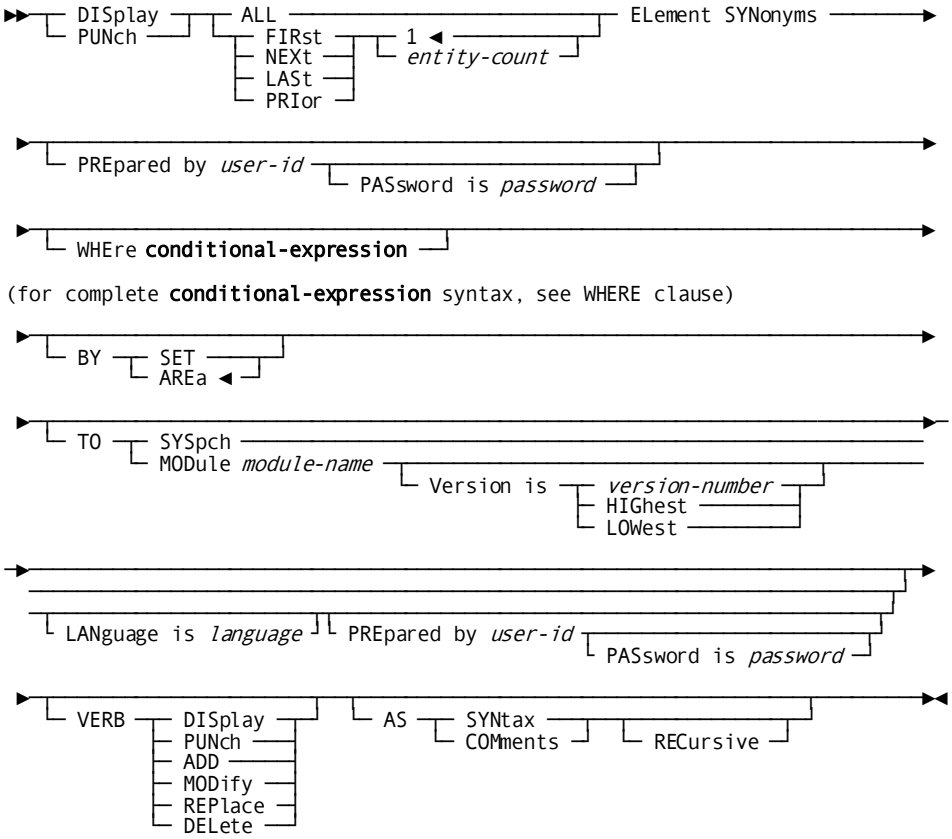
You can display or punch selected element synonyms using the ELEMENT SYNONYM statement.

Syntax

ELEMENT SYNONYM (for a single synonym)



DISPLAY/PUNCH ELEMENT SYNONYM (for multiple synonyms)



ENTRY POINT

ENTRY POINT statements document program entry points. Optional clauses:

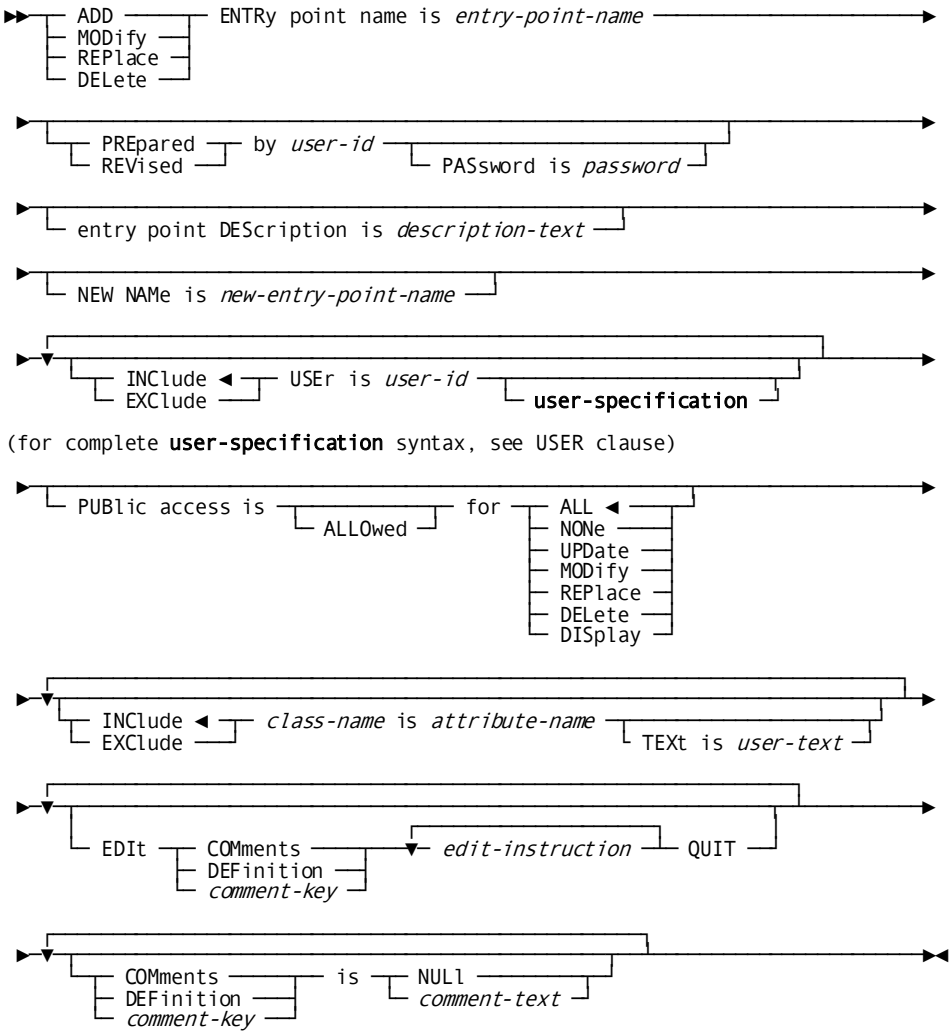
- Associate documentation text with the entry point
- Relate the entry point to users
- Include the entry point in attribute/entity relationships

Entry points are associated with programs through the PROGRAM statement (see PROGRAM (see page 255)).

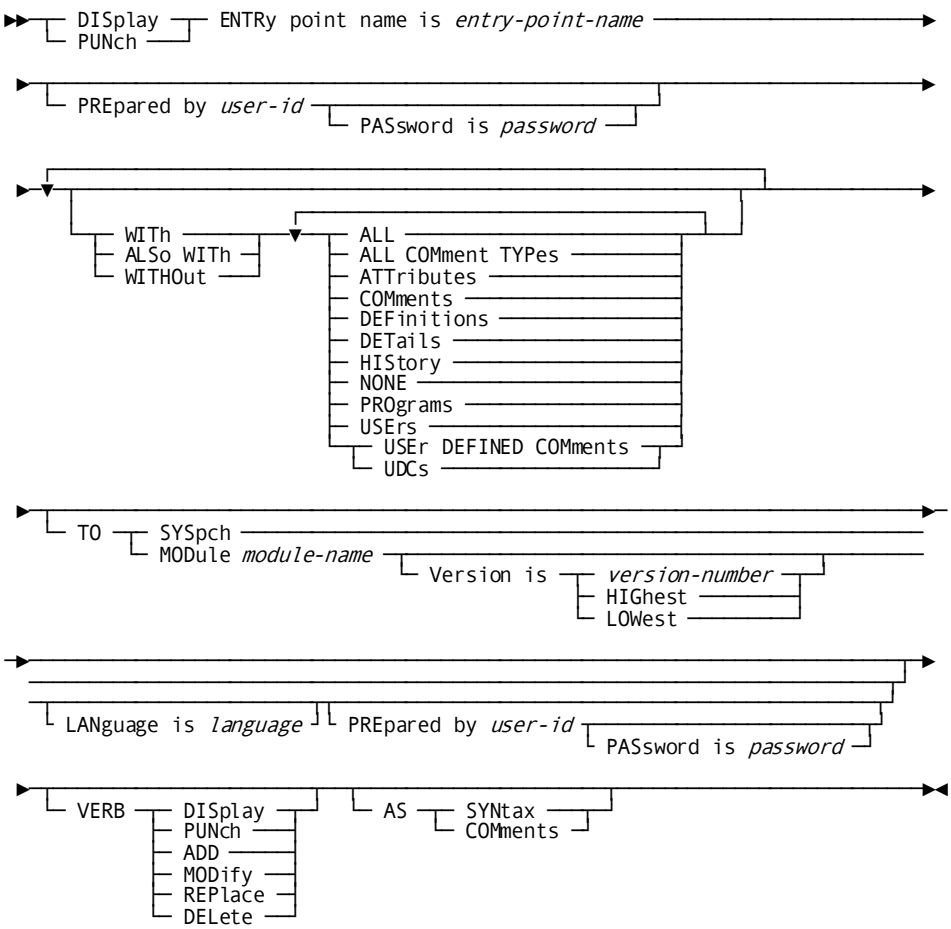
If the SET OPTIONS statement specifies SECURITY FOR IDD IS ON, the user must have the proper authority to issue ENTRY POINT statements.

Syntax

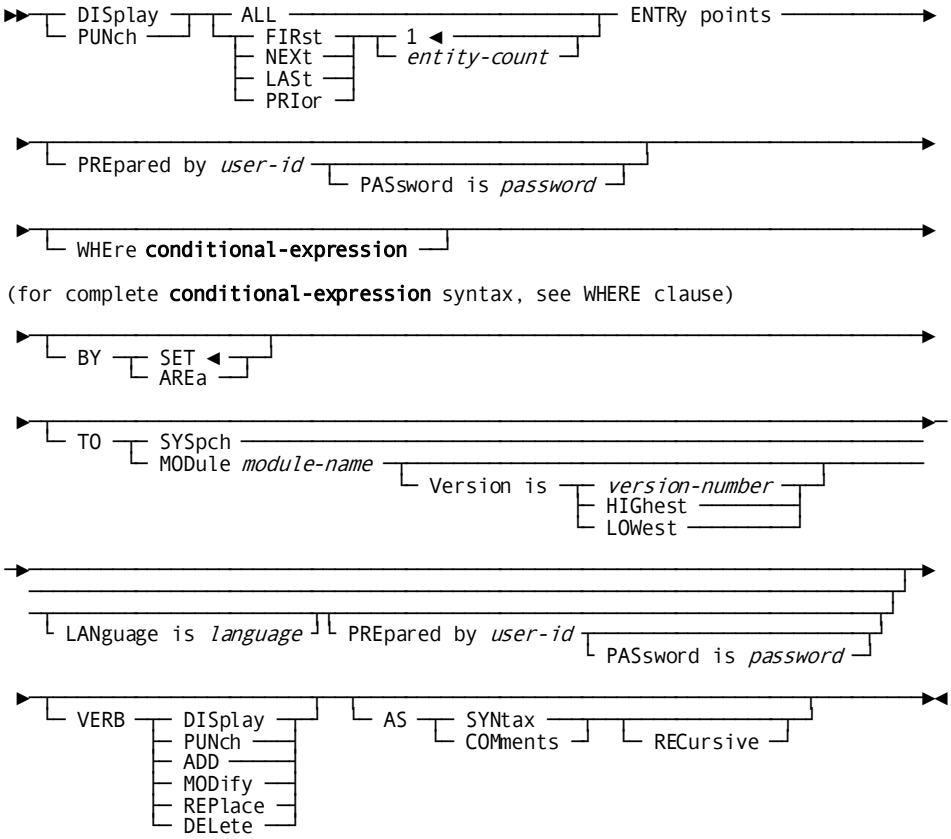
ENTRY POINT Statement


ADD
MODify
REPlace
DELete
ENTRy point name is entry-point-name
PREpared
REVised
by user-id
PASsword is password
entry point DEScription is description-text
NEW NAMe is new-entry-point-name
INClude
EXClude
USEr is user-id
user-specification
(for complete user-specification syntax, see USER clause)
PUBlic access is
ALLOwed
for
ALL
NONe
UPDate
MODify
REPlace
DELete
DISplay
INClude
EXClude
class-name is attribute-name
TEXt is user-text
EDIt
COMments
DEFinition
comment-key
edit-instruction
QUIT
COMments
DEFinition
comment-key
is
NULl
comment-text

DISPLAY/PUNCH ENTRY POINT Statement (for a single entry point)



DISPLAY/PUNCH ENTRY POINT Statement (for multiple entry points)



Parameters

ENTRy point name is *entry-point-name*

Identifies a new entry point to be added to the dictionary, or an existing entry point to be modified, replaced, deleted, displayed, or punched. *Entry-point-name* must be a unique 1- through 8-character name.

NEW NAMe is *new-entry-point-name*

Specifies a new name for the requested entry point. This clause changes only the name of the entry point; it does not alter or delete previously defined relationships in which the entry point participates. Subsequent references to the requested entry point must specify the new name.

WITh/ALSo WITh/WITHOut

Includes or excludes the specified options when the named entry point is displayed or punched. Detailed information for each DISPLAY/PUNCH option is under SET OPTIONS Syntax (see page 36). The option that is listed below presents special considerations for this entity type.

DETAils

Includes the DESCRIPTION clause.

Example

In the following example, the ADD statement defines entry point OREN5A and relates it to the attribute DEVELOPMENT within the class ENTITY-STATUS. The MODIFY statement excludes the DEVELOPMENT attribute and includes the attribute PRODUCTION.

```
add entry point name is oren5a
    description is
         'entry point in order-entry pgm: nxtordor'
    entity-status is development.

modify entry point name is oren5a
    exclude entity-status is development
    include entity-status is production.
```

FILE

FILE statements document card, tape, and other nondatabase files. A file can represent groups of records (and thus elements) if a RECORD statement includes the WITHIN FILE clause; see RECORD (REPORT/TRANSACTION) (see page 287) later in this chapter for further details. Files are related to programs through the PROGRAM statement FILE clause; see PROGRAM (see page 255) later in this chapter for further details. Specifications provided in the FILE statement are used by the CA IDMS DMLO precompiler for COBOL (IDMSDMLC) during the COBOL FD COPY function.

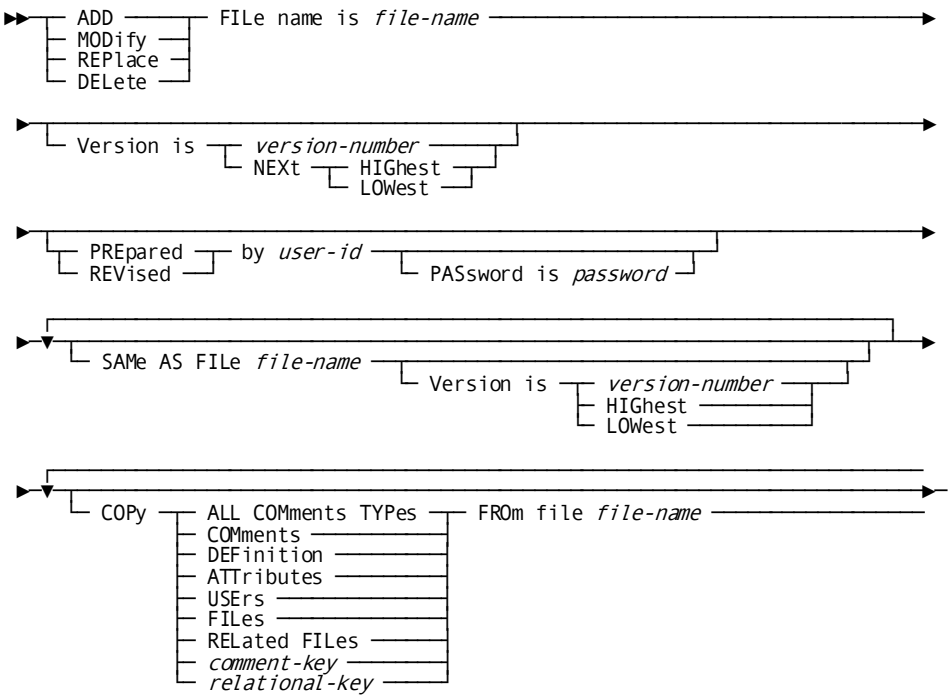
Optional FILE statement clauses:

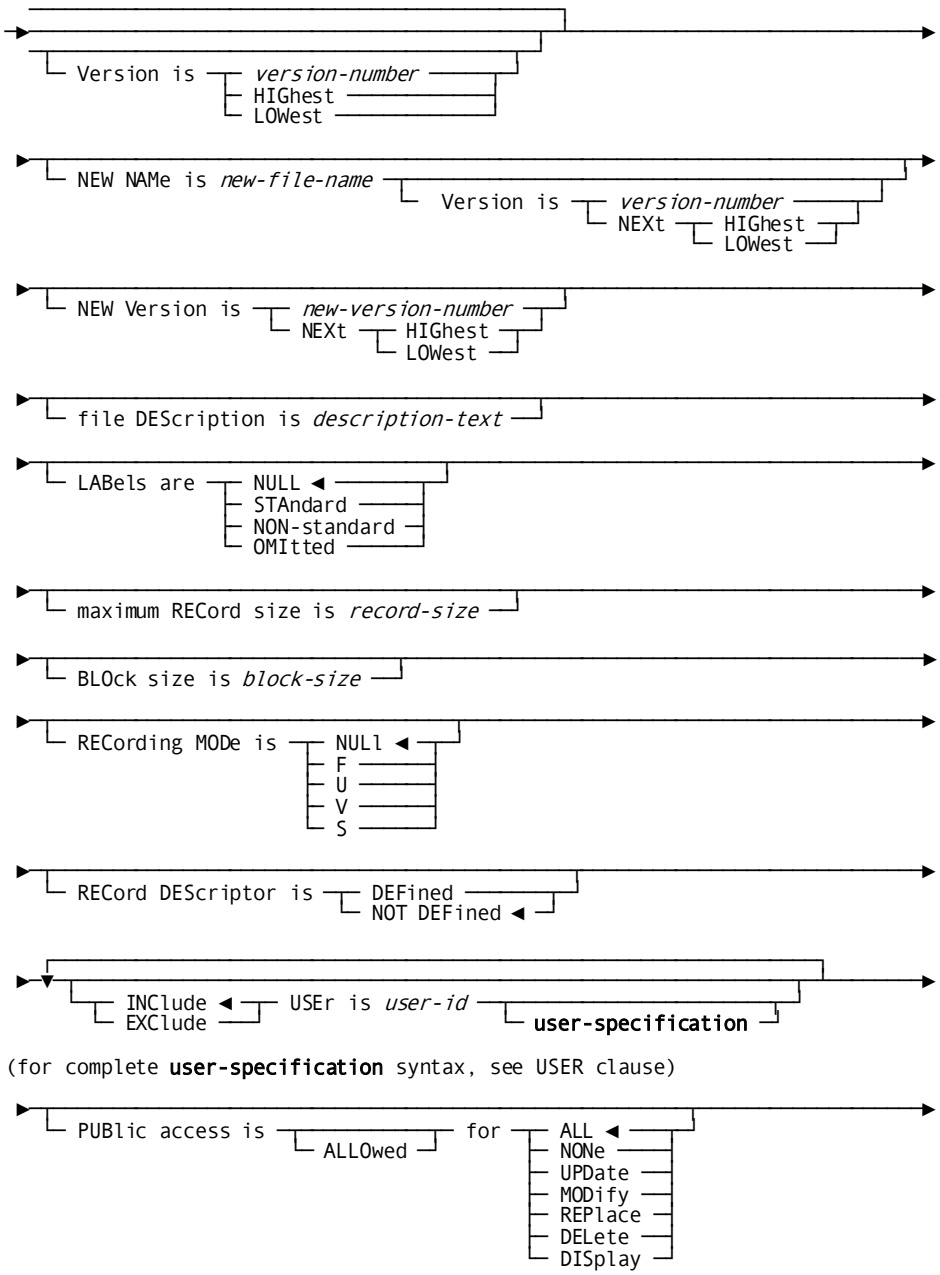
- Relate files to users and to other files
- Control the participation of file occurrences in attribute/entity relationships
- Maintain documentation entries and record information required in COBOL FD statements
- Support the definition of file synonyms

If the SET OPTIONS statement specifies SECURITY FOR IDD IS ON, the user must be assigned the proper authority to issue FILE statements.

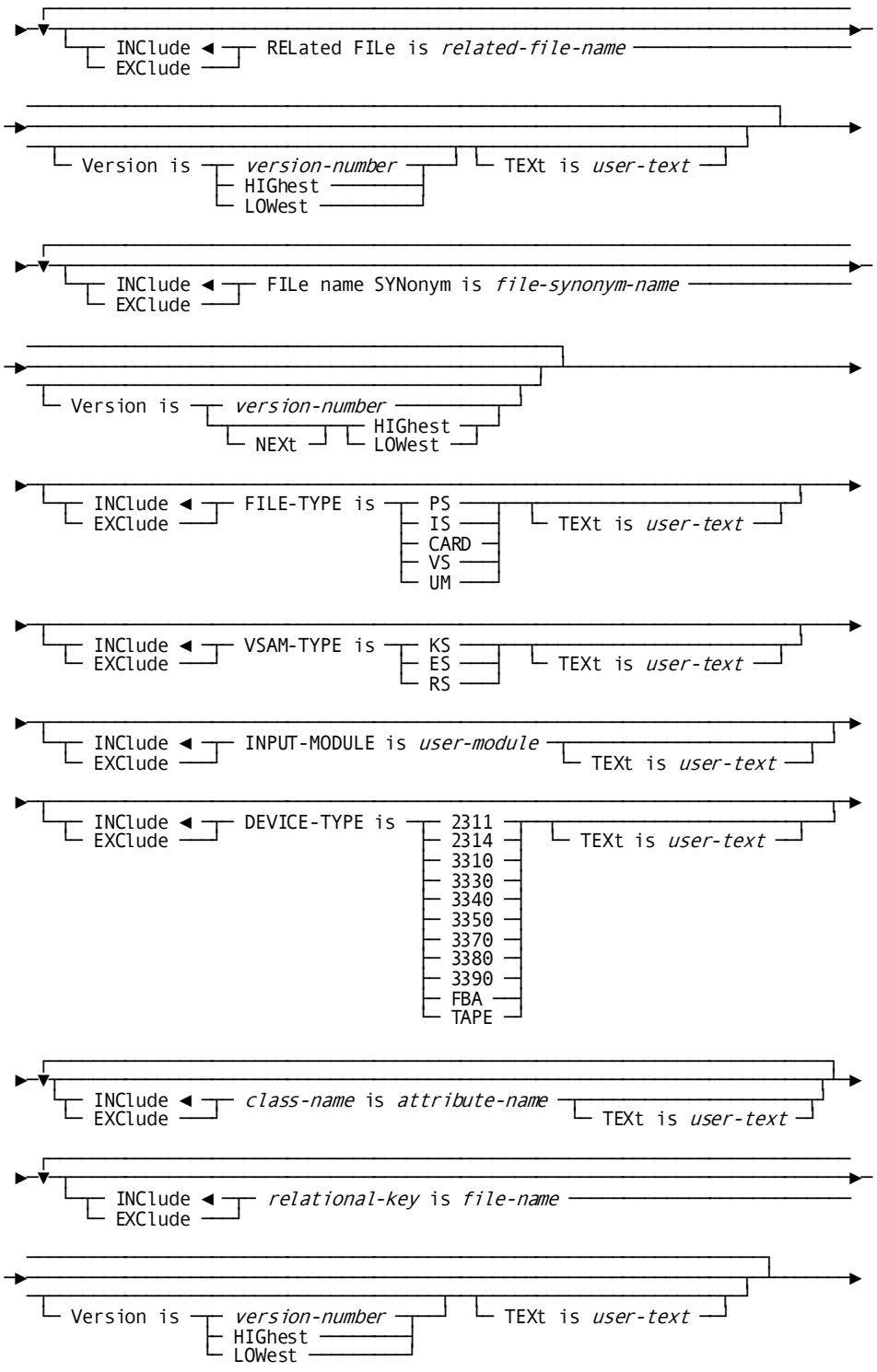
Syntax

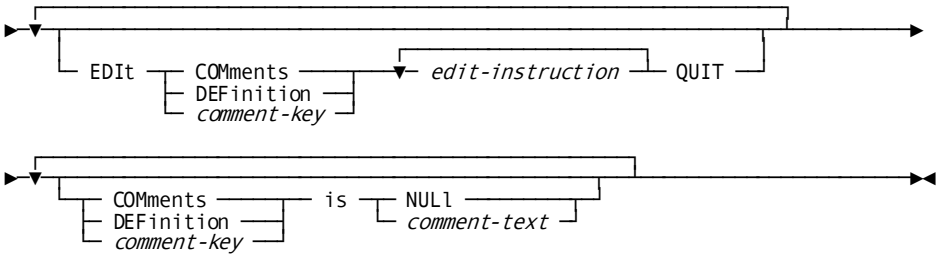
FILE Statement



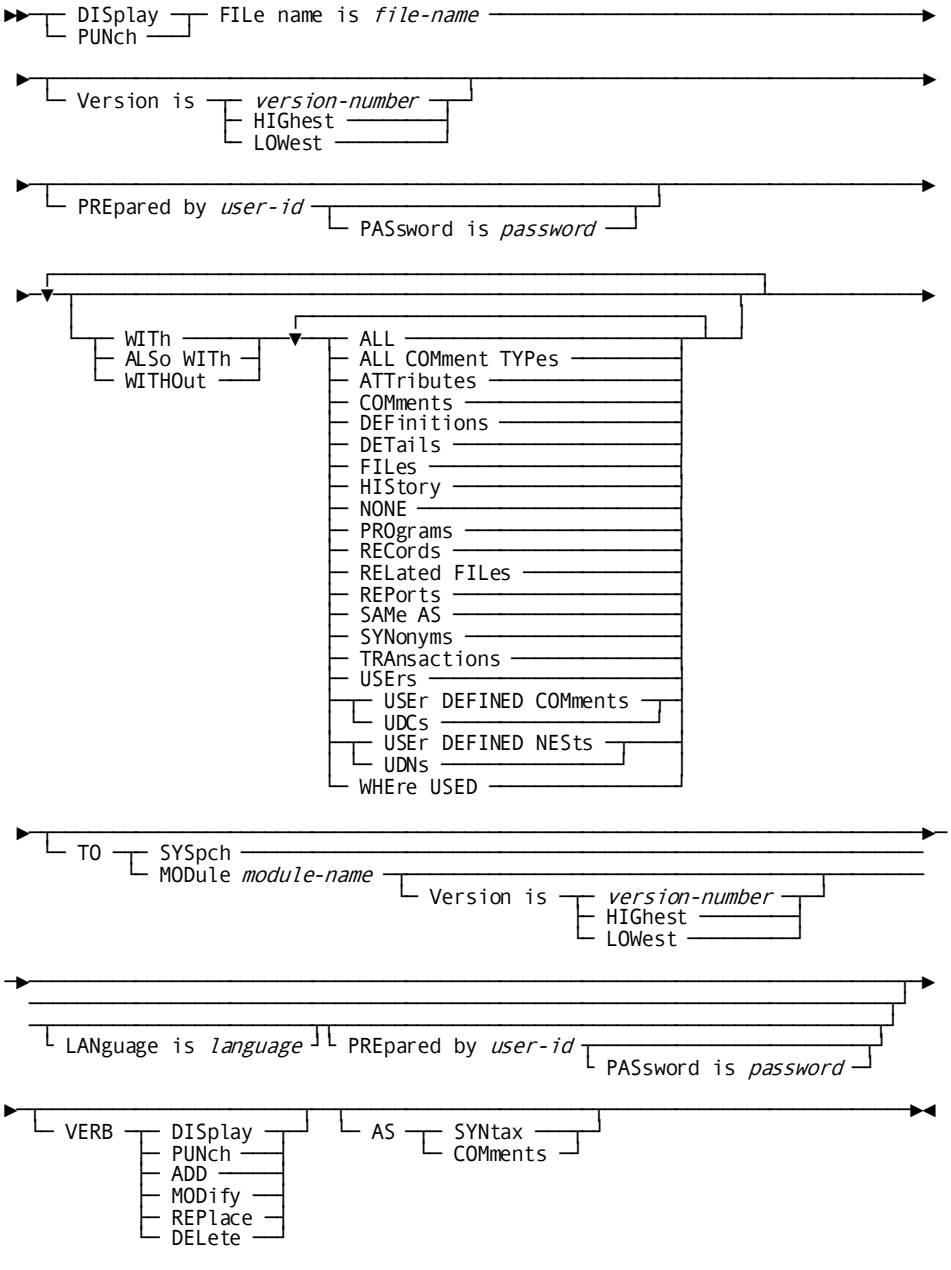


(for complete **user-specification** syntax, see USER clause)

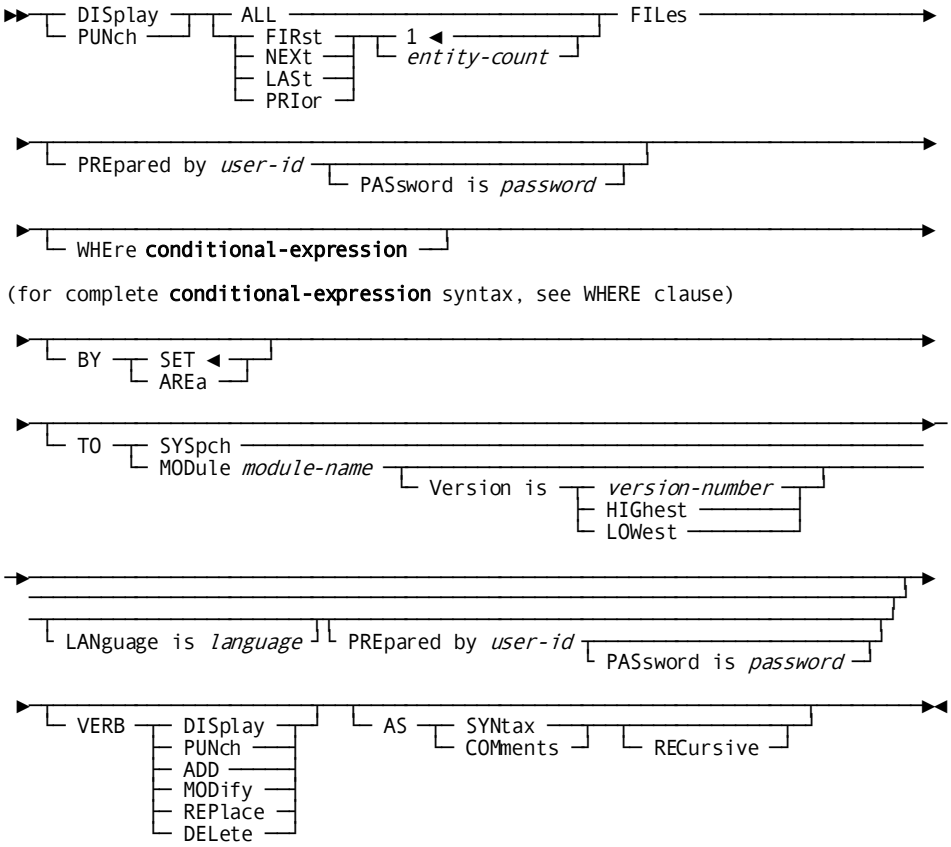

INClude
EXClude
RELated FILe is related-file-name
Version is
version-number
HIGhest
LOWest
TEXt is user-text
INClude
EXClude
FILe name SYNonym is file-synonym-name
Version is
version-number
NEXt
HIGhest
LOWest
INClude
EXClude
FILE-TYPE is
PS
IS
CARD
VS
UM
TEXt is user-text
INClude
EXClude
VSAM-TYPE is
KS
ES
RS
TEXt is user-text
INClude
EXClude
INPUT-MODULE is user-module
TEXt is user-text
INClude
EXClude
DEVICE-TYPE is
2311
2314
3310
3330
3340
3350
3370
3380
3390
FBA
TAPE
TEXt is user-text
INClude
EXClude
class-name is attribute-name
TEXt is user-text
INClude
EXClude
relational-key is file-name
Version is
version-number
HIGhest
LOWest
TEXt is user-text



DISPLAY/PUNCH FILE Statement (for a single file)



DISPLAY/PUNCH FILE Statement (for multiple files)



Parameters

FILe name is *file-name*

Identifies a new file to be established in the dictionary, or an existing file to be modified, replaced, deleted, displayed, or punched. *File-name* must be a 1- through 32-character alphanumeric value. The file name and version must not duplicate that of an existing file or file synonym.

NEW NAMe is *new-file-name*

Specifies a new 1- through 32-character name for the requested file. This clause changes only the name of the file occurrence; it does not alter or delete previously defined relationships in which the file participates. Subsequent references to the file occurrence must specify the new name. If the VERSION clause is not specified, the version number defaults to the version associated with the original file name. The new file name and version number must not duplicate that of an established file or file synonym.

NEW Version is *new-version-number*/NEXt HIGhest/NEXt LOWest

Specifies a new version number for the named file. The file name and new version number must not duplicate that of an existing file or file synonym.

LABels are

Designates a label-processing specification for the named file for use by the IDMSDMLC precompiler during COBOL FD COPY functions. This clause is used only by CA Culprit.

NULl

Specifies no labeling (the default); NULL produces no COBOL FD COPY specification.

STAndard

Specifies standard format file labels; STANDARD produces the LABELS ARE STANDARD COBOL FD COPY specification.

NON-standard

Specifies nonstandard format file labels; NON-STANDARD produces the LABELS ARE NON-STANDARD COBOL FD COPY specification.

OMItted

Specifies unlabeled files; OMITTED produces the LABELS ARE OMITTED COBOL FD COPY specification.

maximum RECord size is *record-size* (used by CULPRIT)

Defines a maximum record size for the named file for use by the IDMSDMLC precompiler during COBOL FD COPY functions. *Record-size* must be an integer in the range 0 through 32,767 and must be equivalent in bytes to the largest record in the file.

BLOck size is *block-size*

Defines a block size for the file for use by the IDMSDMLC precompiler during COBOL FD COPY functions. *Block-size* must be an integer in the range 0 through 32,767 and must represent the number of bytes in each block in the file. This clause is used only by CA Culprit.

RECording MODe is

Defines a recording mode for the file for use by the IDMSDMLC precompiler during COBOL FD COPY functions. This clause is used by CA Culprit only.

NULl

Removes an existing RECORDING MODE clause. NULL is the default.

F

Indicates fixed-length records.

U

Indicates undefined recording mode.

V

Indicates variable-length records.

S

Indicates variable-length spanned records.

RECord DEScriptor is

Specifies whether a 4-byte prefix for variable-length files to be processed by CA Culprit is already defined in the record or whether CA Culprit must generate the prefix. This clause is used by CA Culprit only.

DEFined

Specifies that the record descriptor is to be defined in the record description in the dictionary.

NOT DEFined

Specifies that the record descriptor is not to be defined in the record and must be added by CA Culprit. NOT DEFINED is the default.

RELated FILe is *related-file-name*

Associates (INCLUDE) the named file with or disassociates (EXCLUDE) it from a previously defined file. *Related-file-name* must be a 1- through 32-character alphanumeric value. If the VERSION clause is not specified, the DDDL compiler uses the default version number specified in the SET OPTIONS statement. The file name and version number must reference the primary name of an existing file.

FILe name SYNonym is *file-synonym*

Associates (INCLUDE) an alternative name with the named file or disassociates (EXCLUDE) an existing alternative name from the named file. *File-synonym* is the 1- through 32-character synonym name; the specified synonym can be referenced in an IDMSDMLC COBOL FD COPY function. If no version number is indicated in the FILE NAME SYNONYM clause, yet the FILE NAME clause includes a version number, the version number of the synonym name defaults to the version specified in the FILE NAME clause. The concatenation of the synonym name and version number must not duplicate that of an existing file or file synonym.

All files have at least one synonym, known as the primary synonym, with the same name and version as the file itself.

FILE-TYPE is

Associates (INCLUDE) a file-type with or disassociates it from (EXCLUDE) a file. This clause is used by CA Culprit only.

PS

Specifies a sequential file as the file type.

IS

Specifies ISAM as the file type.

CARD

Specifies a card file as the file type.

VS

Specifies VSAM as the file type; if VS is specified, VSAM-TYPE (described below) must also be specified.

UM

Specifies the CA Culprit user module as the file type; if UM is specified, INPUT-MODULE (described below) must also be specified.

VSAM-TYPE is

Associates (INCLUDE) a VSAM-type with or disassociates (EXCLUDE) it from a file. This clause is used by CA Culprit only.

KS

Specifies a key-sequenced data set (KSDS) as the VSAM file type.

ES

Specifies an entry-sequenced data set (ESDS) as the VSAM file type.

RS

Specifies a relative-record data set (RRDS) as the VSAM file type.

INPUT-MODULE is *user-module*

Associates (INCLUDE) or disassociates (EXCLUDE) the name of the CA Culprit user module as it will appear on a CULPRIT INPUT parameter. This clause is used by CA Culprit only.

DEVICE-TYPE is 2311/2314/3310/3330/3340/3350/3370/3380/3390/FBA/TAPE

Associates (INCLUDE) a device type with or disassociates (EXCLUDE) it from a file. This clause is used by CA Culprit, only in VSE/ESA environments.

WITh/ALSo WITh/WITHOut

Includes or excludes the specified options when the named file is displayed or punched. Detailed information for each DISPLAY/PUNCH option is under SET OPTIONS Syntax (see page 36). The options that are listed below present special considerations for this entity type.

DETails

Includes the DESCRIPTION, LABELS, RECORD SIZE, BLOCK SIZE, RECORDING MODE, and RECORD DESCRIPTOR specifications.

FILes

Includes the SAME AS and RELATED FILES specifications.

Usage

If you specify REPLACE

If the REPLACE verb is specified, the DDDL compiler initializes to defaults and/or excludes the following options:

- DESCRIPTION
- RELATED FILE
- LABELS
- FILE-TYPE
- MAXIMUM RECORD SIZE
- VSAM-TYPE
- BLOCK SIZE
- INPUT-MODULE
- RECORDING MODE
- DEVICE-TYPE
- RECORD DESCRIPTOR
- Related attributes
- PUBLIC ACCESS
- COMMENTS/DEFINITION/*comment-key*

- USER REGISTERED FOR
- FILE SYNONYM (unless the named file synonym is related to a record)

The following relationships are not affected:

- Files to which the named file is a related file
- Programs that access the named file

Example

In the following example, the ADD statement defines the file STOCKFILE and relates it to the attribute STOCK-UPDATE within the class APPLICATION and to the file CRT-TRANFILE. The MODIFY statement adds the synonym STK3FILE to the definition.

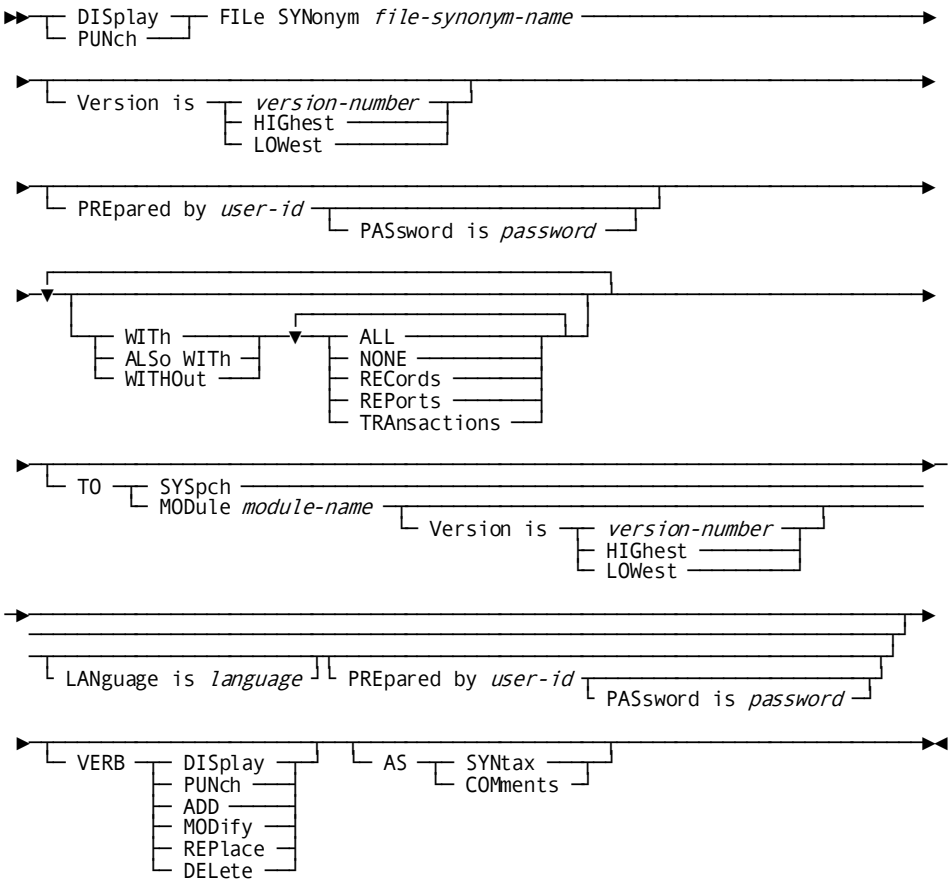
```
add file stockfile
    block size 510
    record size 30
    labels standard
    recording mode f
    application is stock-update
    related file is crt-tranfile
         text 'receives application output'.

modify file stockfile
    file name synonym stk3fil.
```

FILE SYNONYM

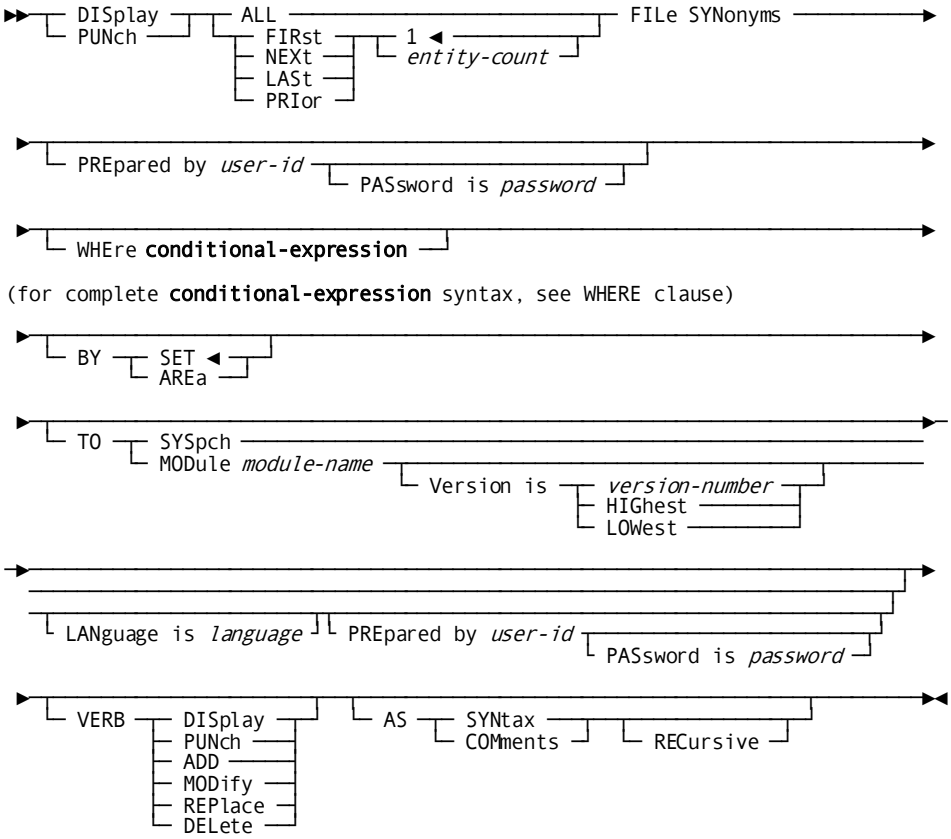
You can display or punch selected file synonyms using the FILE SYNONYM statement.

Syntax

DISPLAY/PUNCH FILE SYNONYM Statement (for a single synonym)



DISPLAY/PUNCH FILE SYNONYM Statement (for multiple synonyms)



LINE

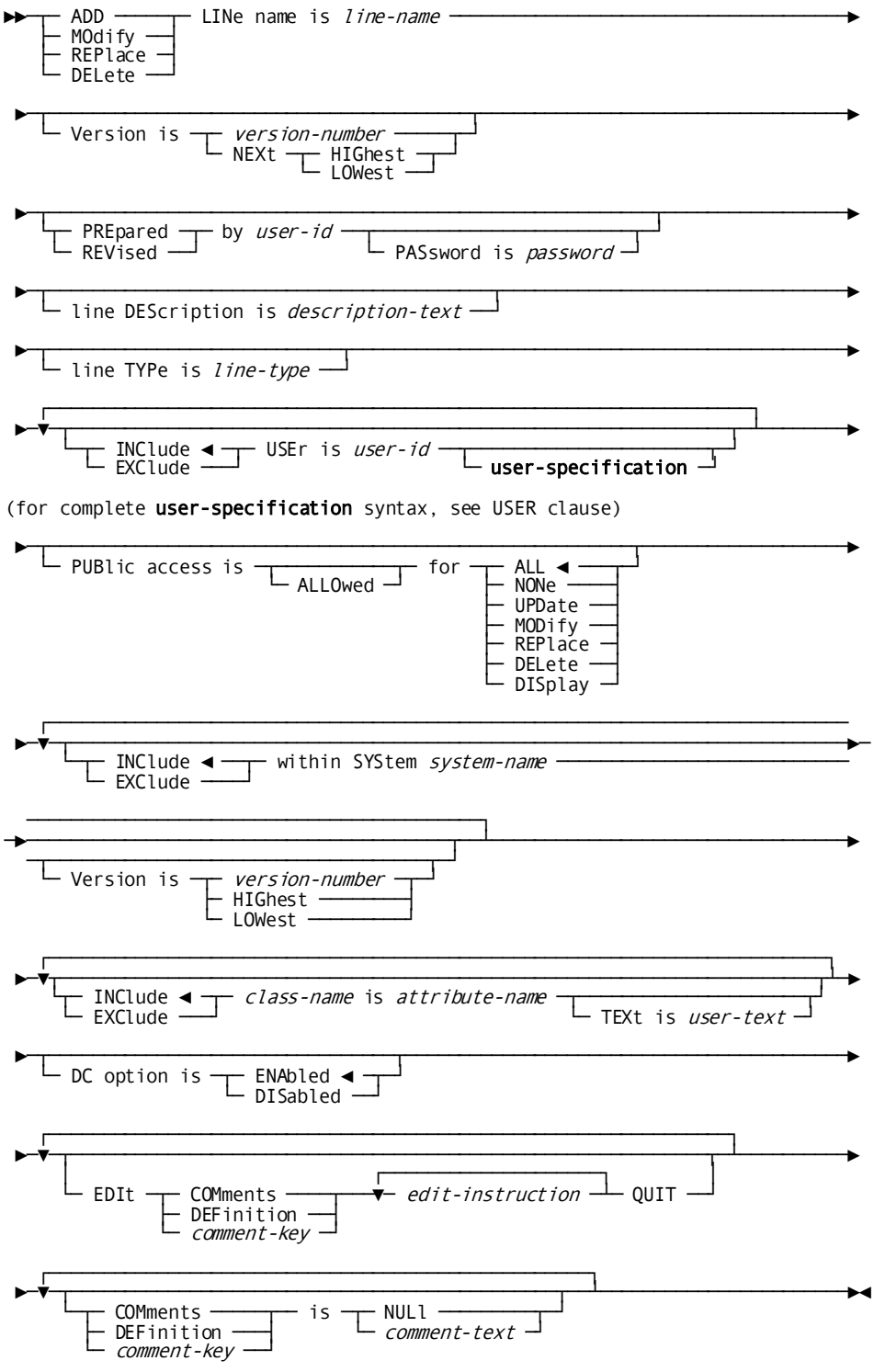
LINE statements are used to document the association between a line and a physical terminal in a teleprocessing system. A physical terminal is associated with a line by means of the PHYSICAL-TERMINAL statement, described under PHYSICAL TERMINAL (see page 239), later in this chapter. Optional LINE statement clauses assign characteristics for use in the DC/UCF system and the Distributed Database System (CA IDMS DDS) environments.

Note: It is recommended that you maintain LINE definitions using the system generation compiler, *not* the DDDL compiler. If a system generation component is processed by the DDDL compiler, only dictionary security is checked, *not* system generation security. For more information on using the system generation compiler, refer to *CA IDMS System Generation Guide*.

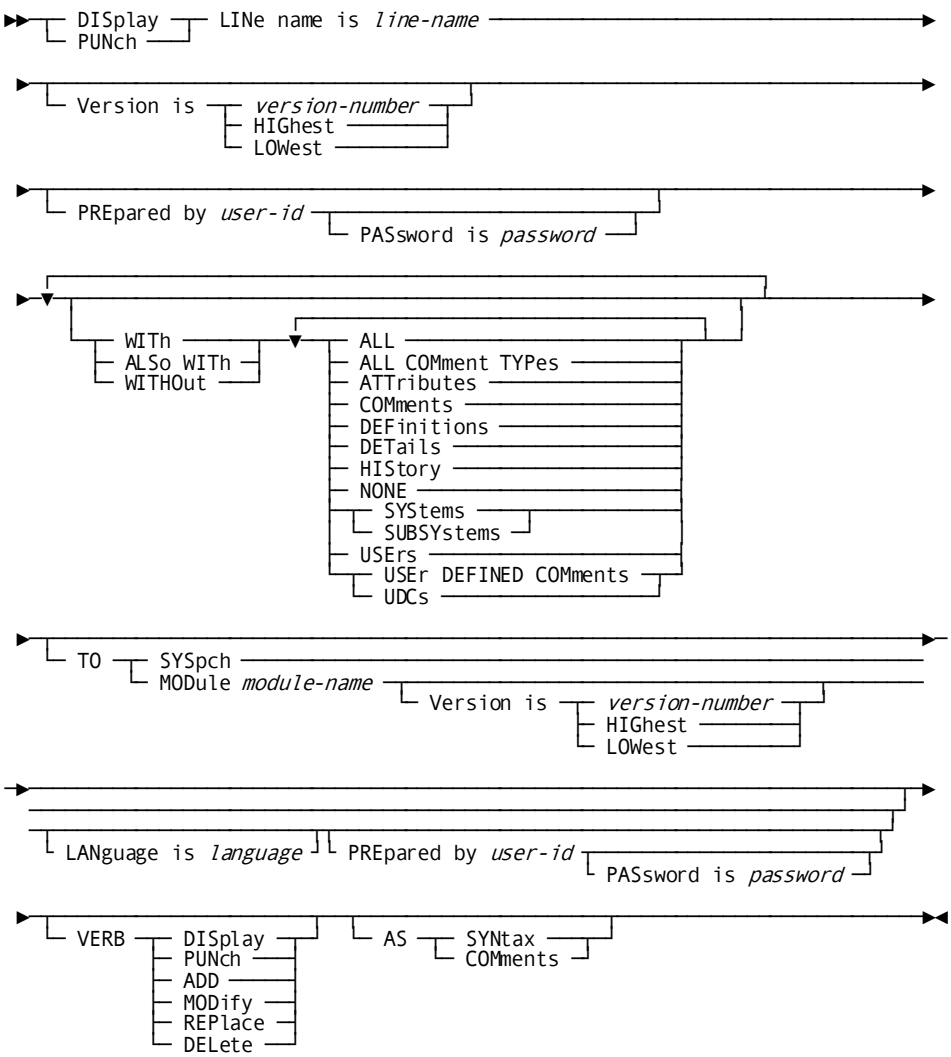
If the SET OPTIONS statement specifies SECURITY FOR IDMS-DC IS ON, the user must be assigned the proper authority to issue LINE statements.

Syntax

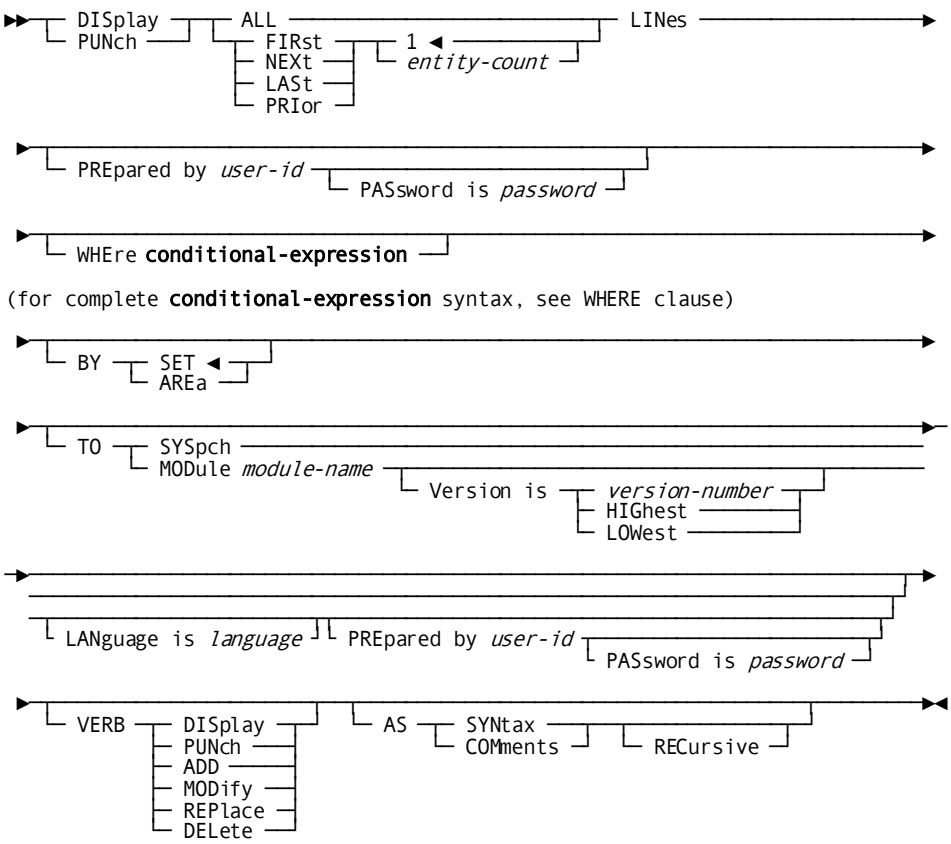
LINE Statement


ADD
MOdify
REPlace
DELete
LINe name is line-name
Version is
version-number
NEXt
HIGhest
LOWest
PREpared
REVised
by user-id
PASsword is password
line DEScription is description-text
line TYPe is line-type
INClude
EXClude
USEr is user-id
user-specification
(for complete user-specification syntax, see USER clause)
PUBlic access is
ALLOwed
for
ALL
NONe
UPDate
MODify
REPlace
DELete
DISplay
INClude
EXClude
within SYStem system-name
Version is
version-number
HIGhest
LOWest
INClude
EXClude
class-name is attribute-name
TEXt is user-text
DC option is
ENAbled
DISabled
EDIt
COMments
DEFinition
comment-key
edit-instruction
QUIT
COMments
DEFinition
comment-key
is
NULl
comment-text

DISPLAY/PUNCH LINE Statement (for a single line)



DISPLAY/PUNCH LINE Statement (for multiple lines)



Parameters

LINe name is *line-name*

Identifies either a new line to be established in the dictionary or an existing line to be modified, replaced, deleted, displayed, or punched. *Line-name* must be a 1- through 8-character alphanumeric value.

LINe TYPe is *line-type*

Specifies a generic line type. *Line-type* must be one of the following values:

ASYNC	BSC2
BSC3	CONSOLE
INOUTL	L3270B
L3280B	S3270Q
SYSOUTL	TCAMLIN
UCFLINE	VTAMLIN

VTAMLU

The LINE TYPE specification is documentation only, unless the line definition is to be copied into a DC/UCF system using the system generation compiler COPY facility.

within SYStem *system-name*

Associates (INCLUDE) the named line with or disassociates (EXCLUDE) it from a system. *System-name* must be the 1- through 32-character name of an existing system. The WITHIN SYSTEM specification is documentation only, unless the system generation compiler COPY facility is to be used to copy the line definition from an IDD-built system. When the COPY facility is not used, all line/system relationships are established and maintained by the system generation compiler.

DC option is

Specifies whether the named line is to be enabled or disabled automatically when the system starts up.

The DC OPTION specification is documentation only, unless the line definition is to be copied into a DC/UCF system using the system generation compiler COPY facility.

ENAbled

Automatically enables the line at system startup. ENABLED is the default.

DISabled

Disables the line until it is explicitly enabled by means of an operator command during system execution.

WITh/ALSo WITh/WITHOut

Includes or excludes the specified options when the named line is displayed or punched. Detailed information for each DISPLAY/PUNCH option is under SET OPTIONS Syntax (see page 36). The option that is listed below presents special considerations for this entity type.

DETails

Includes the DESCRIPTION, LINE TYPE, and DC OPTION specifications.

Usage

If you specify REPLACE

If you specify REPLACE, the DDDL compiler initializes to defaults and/or excludes the following options:

- DESCRIPTION
- LINE TYPE
- USER REGISTERED FOR
- PUBLIC ACCESS
- WITHIN SYSTEM (if built by the DDDL compiler)
- COMMENTS/DEFINITION/*comment-key*
- DC OPTION
- Related attributes

Line-system relationships established by the system generation compiler are not affected.

Example

In the following example, the ADD statement registers the line ROE3 within the system INVENTORY and describes the physical terminals within the line group as 3270s. The MODIFY statement removes the line from the INVENTORY system so that the line can be accessed using the system generation compiler COPY facility.

```
add line roe3
    prepared by dba password is 'ice 9'
    definition is 'line between inventory central and oebost'
    - 'terminals are remote 3270s using btam'
    line type is bsc3
    within system inventory.

modify line roe3
    revised by dba
    exclude system inventory.
```

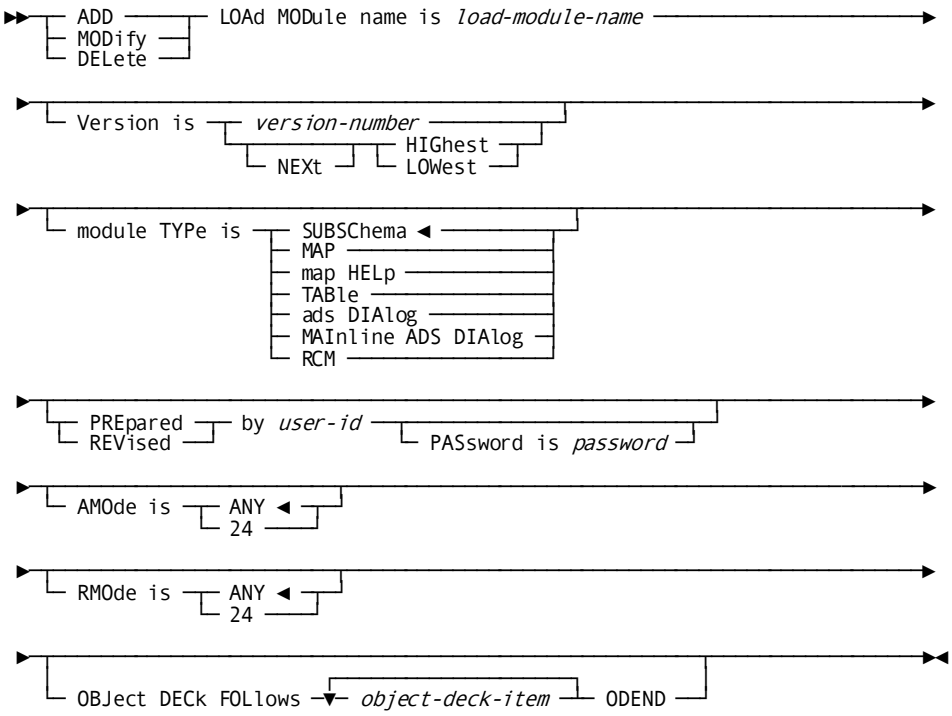
LOAD MODULE

The LOAD MODULE statement is used to submit a relocatable deck to be stored as a load module in the dictionary load area (DDLDCLOD). Load modules are particularly useful in the CA IDMS DDS environment because they can be moved among the dictionaries used by the multiple central versions that participate in the DDS system. CA IDMS subschemas, DC/UCF maps and map editing tables, database name tables, and CA ADS processes can be stored in the dictionary as load modules.

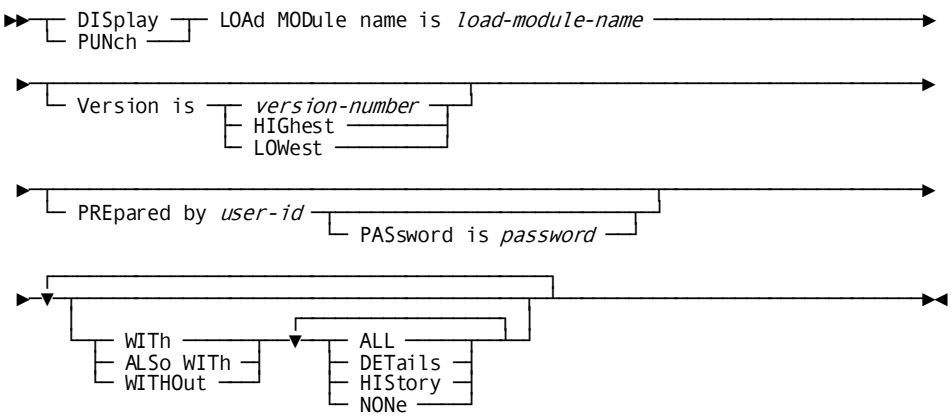
If the SET OPTIONS statement specifies SECURITY FOR LOAD MODULE IS ON, the user must be assigned the proper authority to issue LOAD MODULE statements.

Syntax

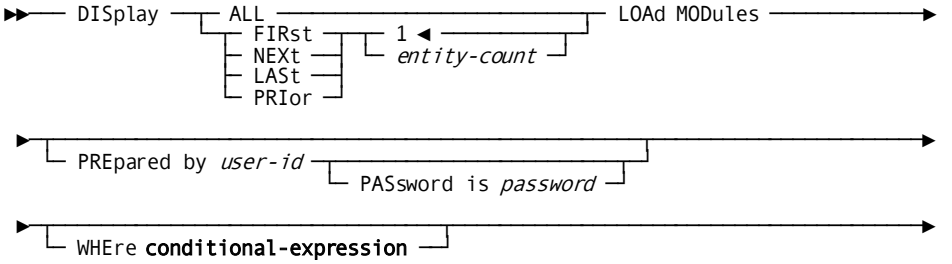
LOAD MODULE Statement



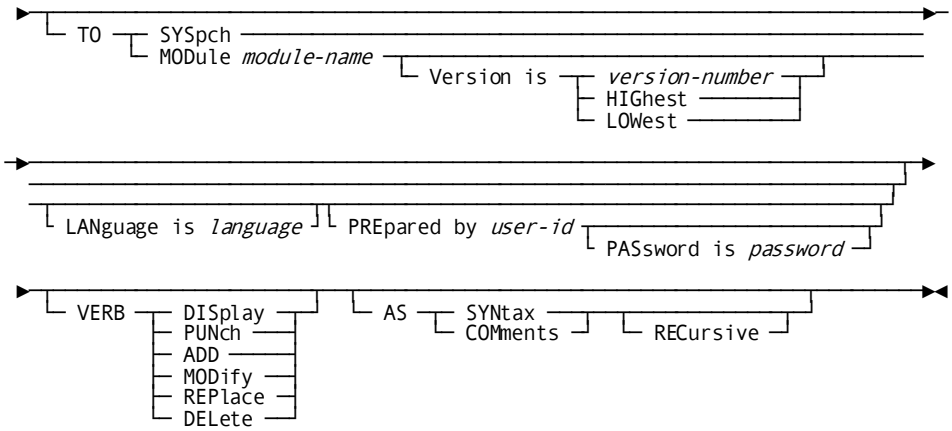
DISPLAY LOAD MODULE Statement (for a single load module)



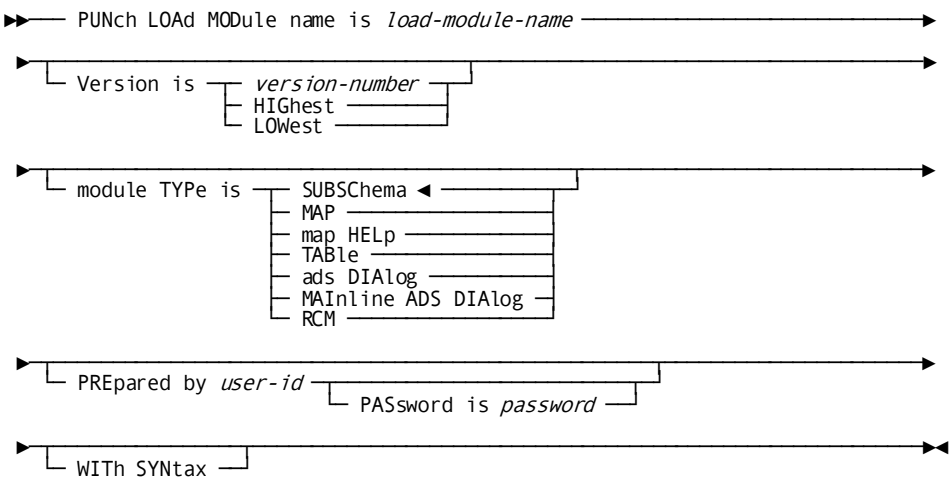
DISPLAY LOAD MODULE Statement (for multiple load modules)



(for complete **conditional-expression** syntax, see WHERE clause)



PUNCH LOAD MODULE Statement



Parameters

LOAd MODule name is *load-module-name*

Identifies a new load module to be established in the dictionary, or an existing load module to be modified, deleted, punched, or displayed. If MODIFY is specified, the only valid parameters are AMODE and RMODE. If PUNCH is specified, the DDDL compiler produces a relocatable deck from the named load module; that deck can subsequently be link edited and placed in a load (core-image) library. *Load-module-name* must be a 1- through 8-character alphanumeric value.

module TYPe is SUBSChema/MAP/map HELp/TABle/ads DIAlog/MAInline ADS DIAlog/RCM

Identifies the load module as a subschema, map, map help, table, CA ADS dialog, mainline dialog, or relational command module (RCM). With all verbs, you can use the MODULE TYPE clause as an additional qualifier immediately after the version clause. If you specify MODULE TYPE as an additional qualifier, the compiler makes sure that the load module named in a MODIFY/ALTER or DELETE/DROP statement is of the same type. The default for this clause is SUBSCHEMA.

AMOde is

Specifies the named module's addressing mode (for ADD and MODIFY only)

ANY

Indicates that the module is invoked in 31-bit addressing mode. ANY is the default.

24

Indicates that the module is invoked in 24-bit addressing mode.

If RMODE IS ANY is specified, AMODE must be ANY.

RMODe is

Specifies the named module's residency mode (for ADD and MODIFY only)

ANY

Indicates that the module can be loaded above or below the 16-megabyte line. For DC/UCF systems running in 24-bit mode, modules are loaded below the 16-megabyte lines, regardless of the RMODE specification. ANY is the default.

24

Indicates that the module must be loaded below the 16-megabyte line.

For DC/UCF systems running in 31-bit mode, modules with an RMODE of ANY are loaded into XA program pools (above the 16-megabyte line); modules with an RMODE of 24 are loaded into non-XA pools (below the 16-megabyte line).

OBJect DECk FOLlows *object-deck-item* ODEND

Specifies the object (relocatable) deck to be stored as the load module in the dictionary (for ADD only). OBJECT DECK FOLLOWS must be coded on the first line by itself; *object-deck-item*s follow on the second and subsequent lines; ODEND terminates the object deck and is coded on the last line.

WITh/ALSo WITh/WITHOut

For DISPLAY only, includes the specified options when the named load module is displayed. For detailed information on each DISPLAY/PUNCH option, see SET OPTIONS Syntax (see page 36). The options that are listed below present special considerations for this entity type.

Note: DISPLAY output always appears as comments, regardless of the default option in effect.

HIStory

Includes the date and time the load module was created.

DETails

Includes module length, entry point address, number of RLD (relocation directory) entries, security class, logical deletion flag, and module type (subschema, map, table, dialog, or mainline dialog).

WITh SYNtax

Punches an object deck accompanied by the ADD LOAD MODULE syntax (*load-module-name*, VERSION, PREPARED BY, OBJECT DECK FOLLOWS, *object-deck-items*, and ODEND). This option is useful for producing an object deck that is to be placed in a load area other than the system load library. (Option is for PUNCH only.)

Usage

Load module considerations

The following considerations apply to load modules:

- Only load modules produced by CA IDMS compilers can be placed in the dictionary load area; COBOL and PL/I programs are not eligible.
- It is recommended that all ADD LOAD MODULE statements be submitted together in a separate run of the DDDL compiler.
- The MODIFY LOAD MODULE statement can be used only with the RMODE and AMODE clauses; other clauses are not valid when using the MODIFY verb.

Deleting load modules

When you *delete* a load module, PROG-051 records associated with the load module are also deleted, providing the PROG-051 records are:

- The same type as the load module (subschema, map, table, dialog, RCM, or map help)

  and

- Not related to any other entity type in the dictionary

Punching a load module

When you punch a load module from the dictionary load area (DDLDCLOD area) into an object module, the DDDL compiler omits the RMODE/AMODE attributes because the RMODE/AMODE clause is not acceptable to the linkage editor. If you are punching the load module to add it to a different dictionary, then you must edit the punched syntax to include the RMODE/AMODE clause.

Example

The following example illustrates the statements required to define the load module STATETBL.

```
      add load module statetbl
      rmode is any
      amode is any
      module type is table
      object deck follows
esd information
txt information ...
rld information
end
      odend.
```

LOGICAL TERMINAL

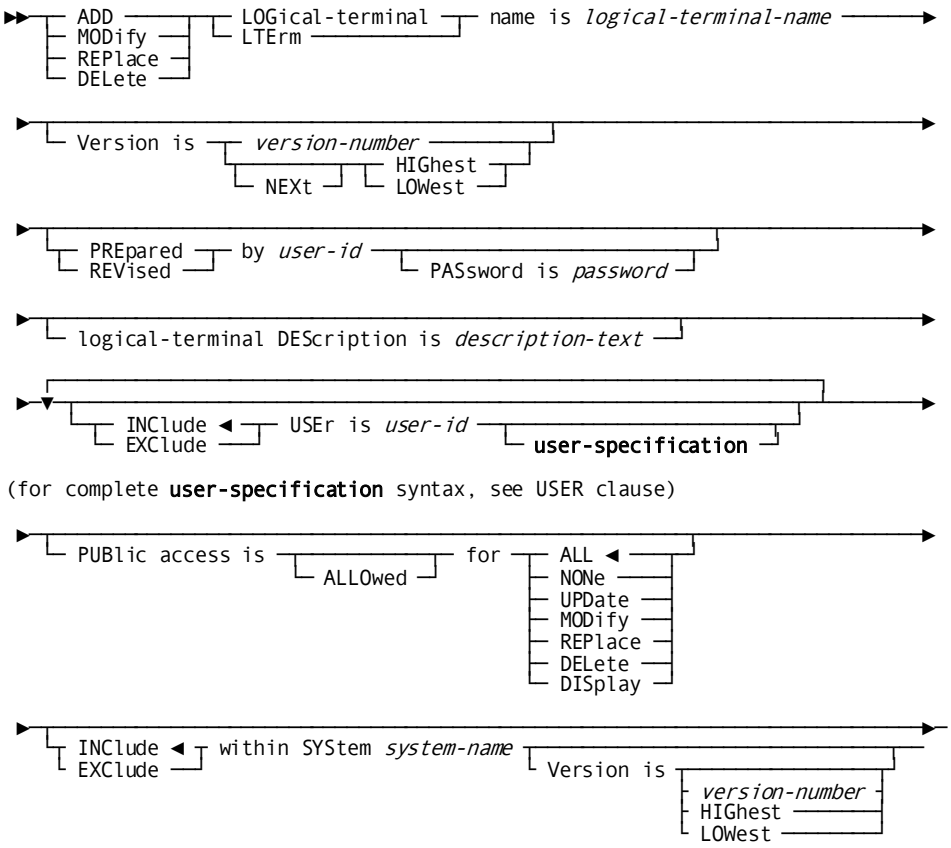
LOGICAL-TERMINAL statements are used to document the logical terminals used in an online environment and to relate those logical terminals to established systems and users, as well as to physical terminals. Logical terminals allow application programs to communicate with DC/UCF systems without specifying physical device identifiers. At runtime, the terminal user's signon information, the executing task, and dynamic storage are associated with the logical terminal.

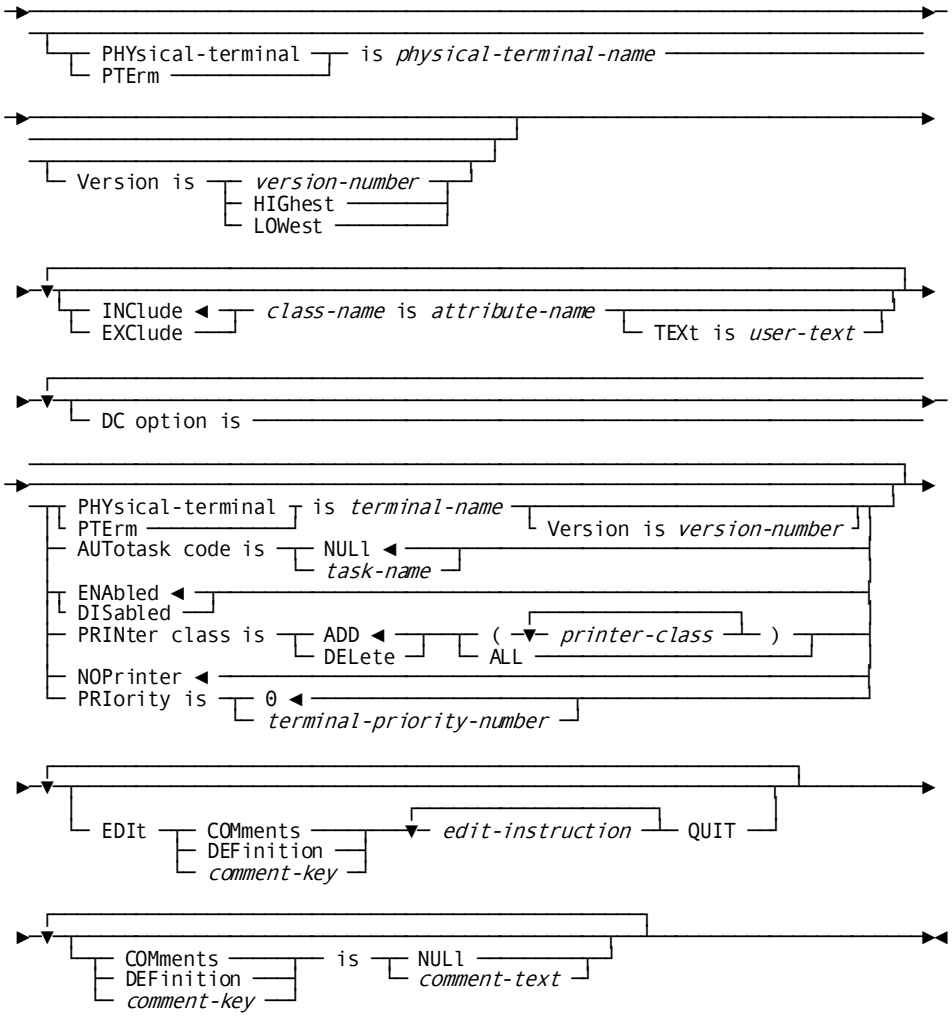
Note: It is recommended that you maintain LOGICAL TERMINAL definitions using the system generation compiler, *not* the DDDL compiler. If a system generation component is processed by the DDDL compiler, only dictionary security is checked, *not* system generation security. For more information on using the system generation compiler, refer to *CA IDMS System Generation Guide*.

If the SET OPTIONS statement specifies SECURITY FOR IDMS-DC IS ON, the user must be assigned the proper authority to issue LOGICAL-TERMINAL statements.

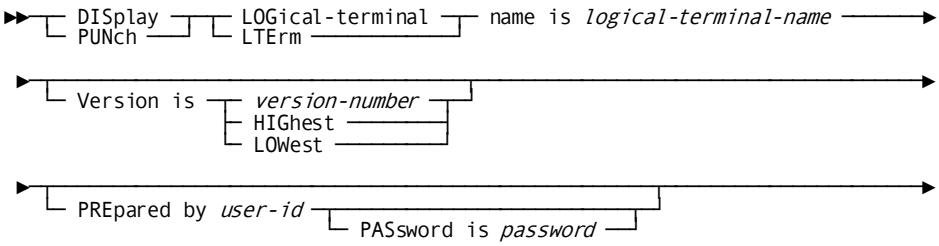
Syntax

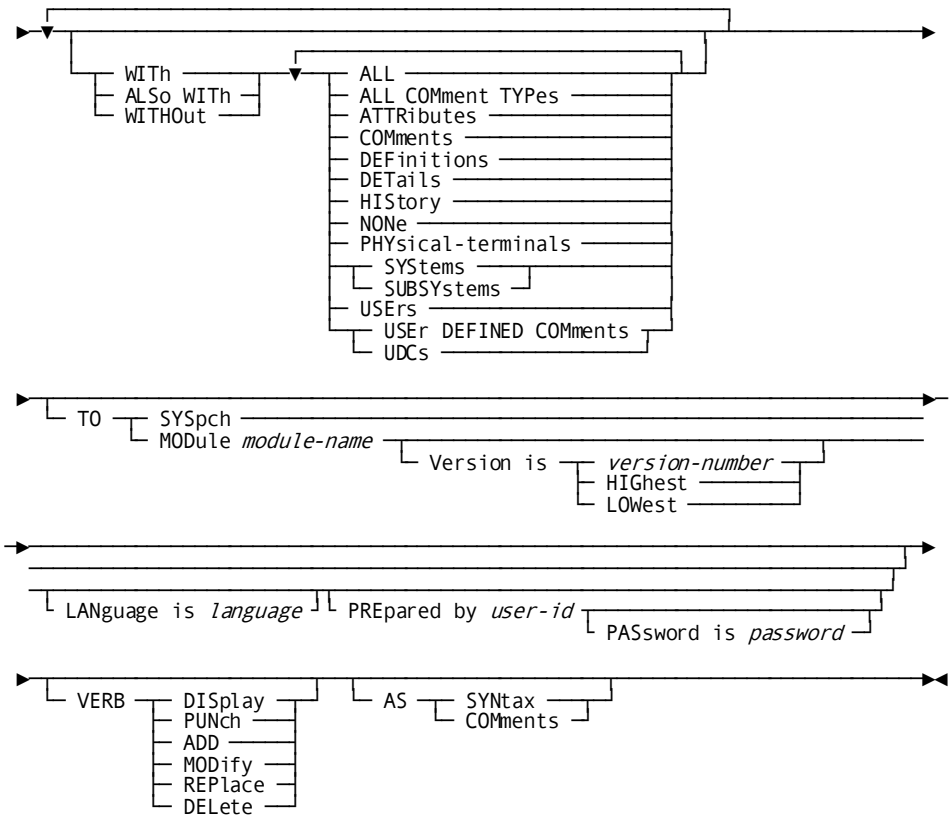
LOGICAL-TERMINAL Statement



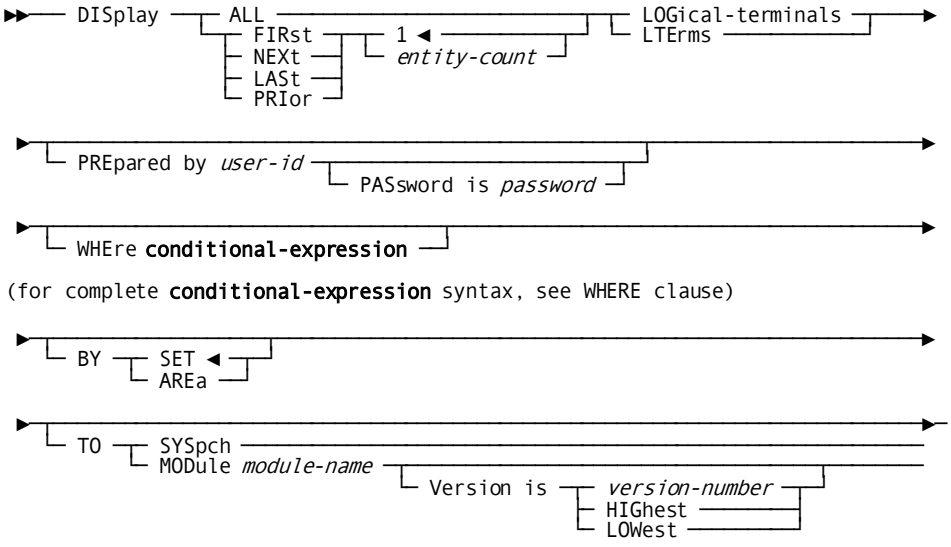


DISPLAY/PUNCH LOGICAL-TERMINAL Statement (for a single terminal)

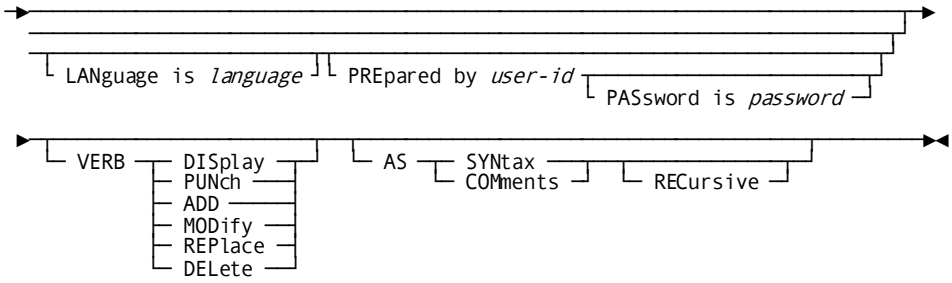




DISPLAY/PUNCH LOGICAL-TERMINAL Statement (for multiple terminals)



(for complete **conditional-expression** syntax, see WHERE clause)


LANguage is language
PREpared by user-id
PASsword is password
VERB
DISplay
PUNch
ADD
MODify
REPlace
DELete
AS
SYNtax
COMments
RECursive

Parameters

LOGical-terminal name is *logical-terminal-name*

Identifies a new logical terminal to be established in the dictionary or an existing logical terminal to be modified, replaced, deleted, displayed, or punched. LTErm is a synonym for LOGical-terminal. *Logical-terminal-name* must be a 1- through 8-character alphanumeric value.

within SYStem *system-name*

Associates the named logical terminal with a system. *System-name* must be the 1- through 32-character name of an existing system. The WITHIN SYSTEM specification is documentation only, unless the system generation compiler COPY facility is to be used to copy logical-terminal definitions from an IDD-built system. When the COPY facility is not used, all logical-terminal/system relationships are established and maintained by the system generation compiler.

PHYsical-terminal is *physical-terminal-name*

Associates a physical terminal with or disassociates it from the logical-terminal/system relationship. The named physical terminal must be defined within the named system. In the DC/UCF environment, this parameter is documentation only; the logical-to-physical terminal association is established by means of the DC OPTION clause (described below) or directly through the system generation compiler.

DC option is

Assigns logical functions to the logical-terminal occurrence and, in DC/UCF environments, associates the logical terminal with a physical terminal.

PHYsical-terminal is *physical-terminal-name*

Specifies the physical terminal with which the named logical terminal is to be associated. Note that the VERSION clause keywords HIGHEST and LOWEST are *not* valid.

Although a logical terminal can be associated with only one physical terminal at a time, the specified association can be changed by means of an operator command during system execution.

AUTotask code is NULl/*task-name*

Specifies whether a task is to be executed automatically when the logical terminal is enabled. NULL (the default) specifies that no task is initiated when the terminal is enabled. *Task-name* specifies that the named task will be initiated automatically when the terminal is enabled. *Task-name* must be a 1- through 8-character alphanumeric value.

If the named task is defined with the INPUT option, task execution is deferred until the terminal operator enters the requested data (see TASK (see page 351)).

Note: Note that AUTOTASK CODE cannot be specified if PRINTER CLASS is specified.

ENAbled/DISabled

Specifies whether the logical terminal is to be enabled or disabled automatically when the DC/UCF system starts up. ENABLED (the default) automatically enables the terminal at system startup. DISABLED disables the terminal until it is enabled explicitly by an operator command during system execution.

PRINter class is ADD/DELete *printer-class-number*/ALL

Specifies one or more print classes. *Printer-class-number* must be an integer in the range 1 through 64. ALL assigns all printer classes (1 through 64) to the logical terminal.

The optional ADD/DELETE parameter adds or deletes the specified printer classes; ADD is the default. Specify the PRINTER CLASS option only if the associated physical terminal is a 3280 or similar device that has print capabilities.

Note: PRINTER CLASS cannot be specified if AUTOTASK CODE is specified.

NOPrinter

Specifies that the logical terminal is not associated with a physical print device.

PRIority is 0/*terminal-priority-number*

Specifies the dispatching priority for the named logical terminal. The DC/UCF system uses the specified value in combination with task and user priorities to determine the dispatching priority of specific requests. *Terminal-priority-number* must be an integer in the range 0 through 255; the default for ADD is 0.

WITh/ALSo WITh/WITHOut (DISPLAY/PUNCH only)

Includes or excludes the specified options when the named logical terminal is displayed or punched. Detailed information for each DISPLAY/PUNCH option is under SET OPTIONS Syntax (see page 36). The option that is listed below presents special considerations for this entity type.

DETails

Includes the DESCRIPTION and DC OPTIONS specifications.

Usage

If you specify REPLACE

If you specify REPLACE, the DDDL compiler initializes to defaults and/or excludes the following options:

- DESCRIPTION
- USER REGISTERED FOR
- PUBLIC ACCESS
- COMMENTS/DEFINITION/*comment-key*
- WITHIN SYSTEM
- DC OPTION

- Related attributes

Logical-terminal/system relationships established by the system generation compiler are not affected.

Example

In the following example, the ADD statement registers logical terminal LTM26 within the system INVENTORY and associates LTM26 with physical terminal TM026. The MODIFY statement removes the logical terminal from the INVENTORY system and defines it as a component of a DC/UCF system.

```
add logical-terminal ltm26
    prepared by dba password is 'ice 9'
    within system inventory
        physical-terminal tm026.

modify logical-terminal ltm26
    revised by dba password is 'ice 9'
    exclude within system inventory
    dc option is physical-terminal tm026
    dc option is autotask code is reser9
    dc option is enabled
    dc option is priority is 15.
```

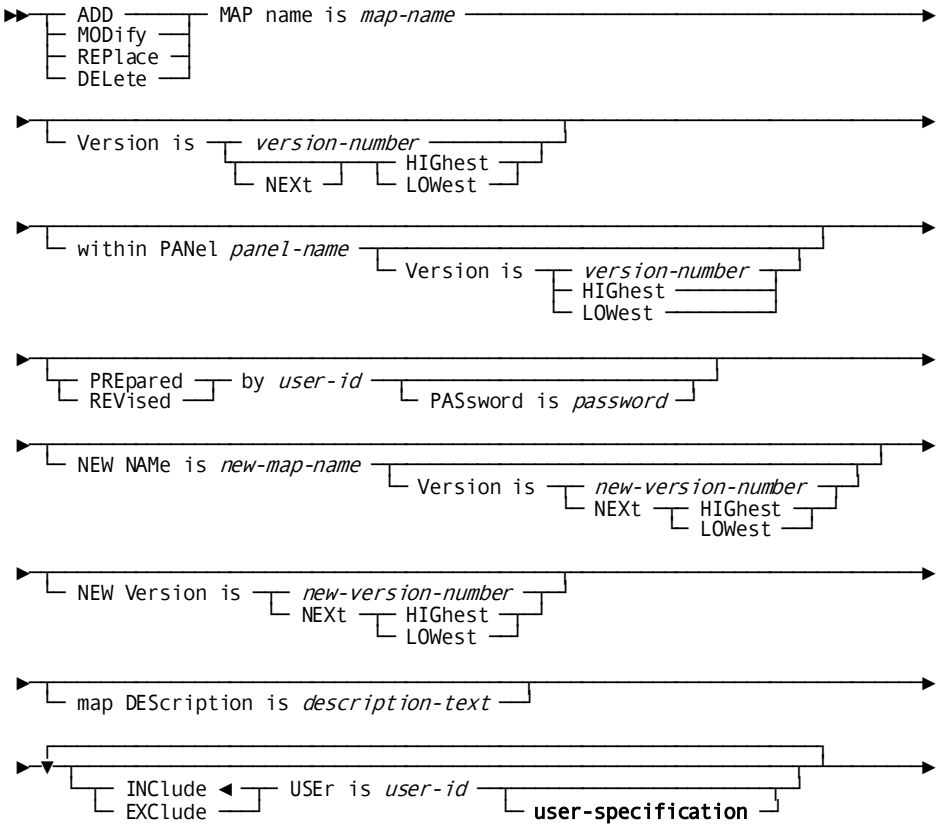
MAP

MAP statements are used to document the maps (or tables) used by teleprocessing monitors to correlate data fields within records with locations on panels (screens) defined for use with 3270-type terminals. Optional MAP statement clauses relate maps to users, systems, and panels and accommodate attribute/entity relationships. MAP statements can document existing map definitions or anticipated map requirements.

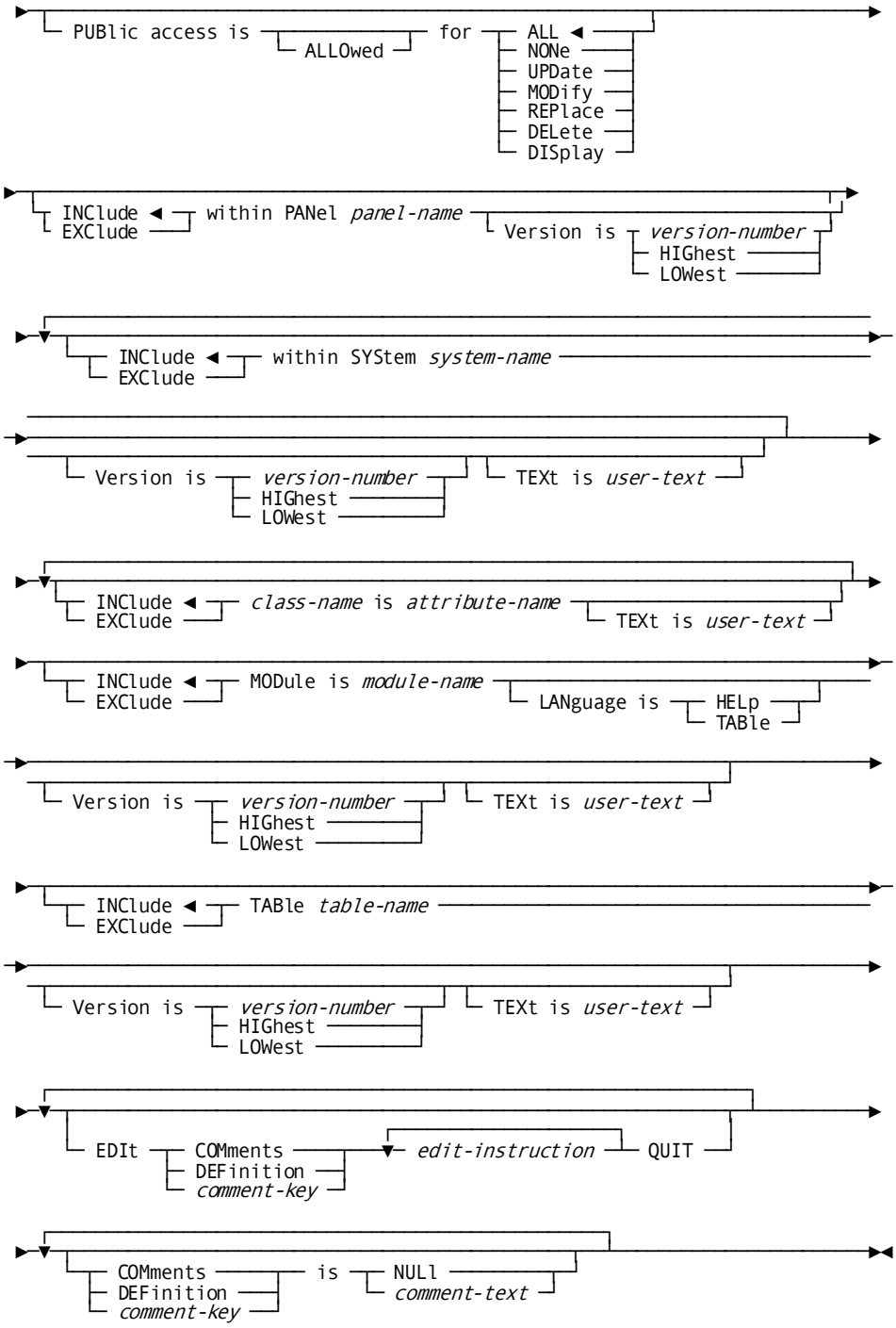
If the SET OPTIONS statement specifies SECURITY FOR IDMS-DC IS ON, the user must be assigned the proper authority to issue MAP statements.

Syntax

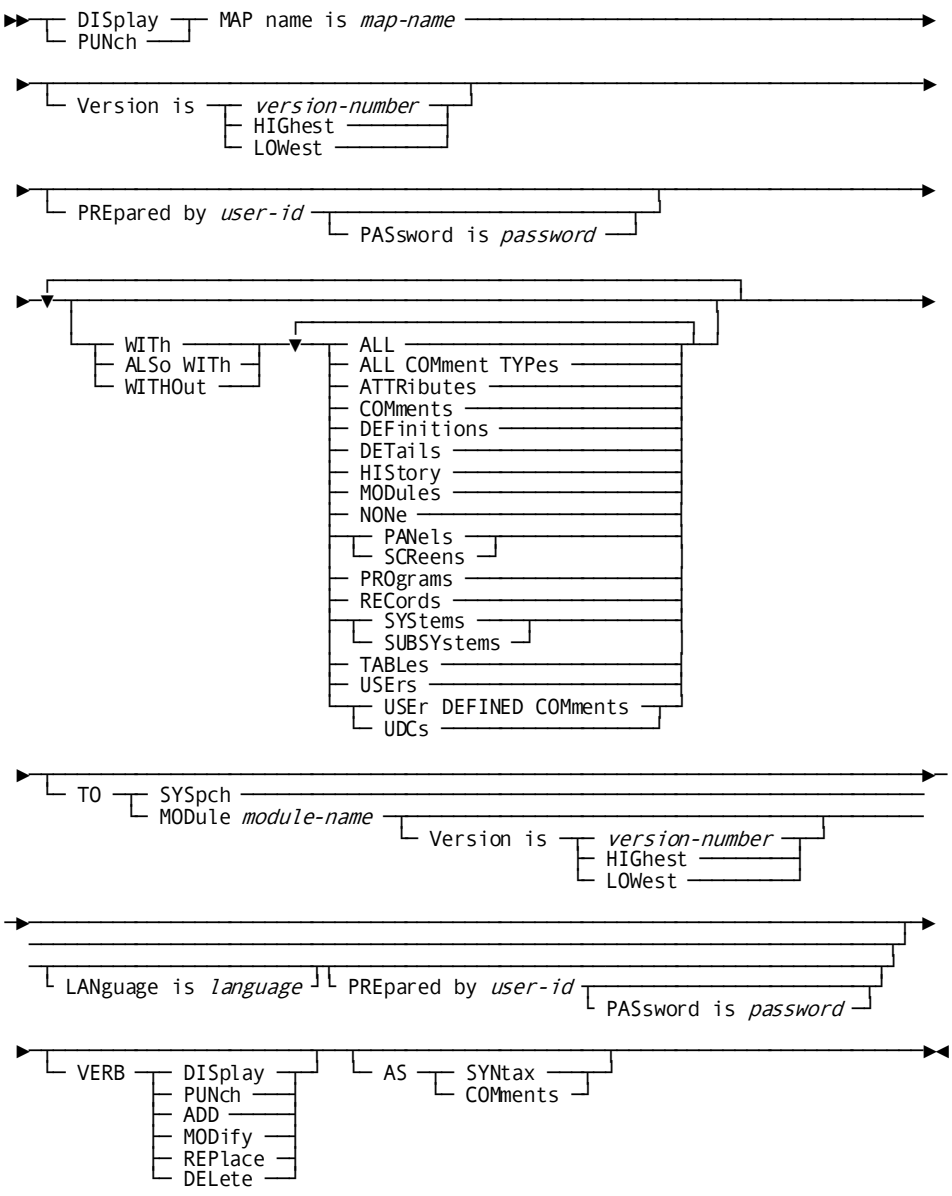
MAP Statement



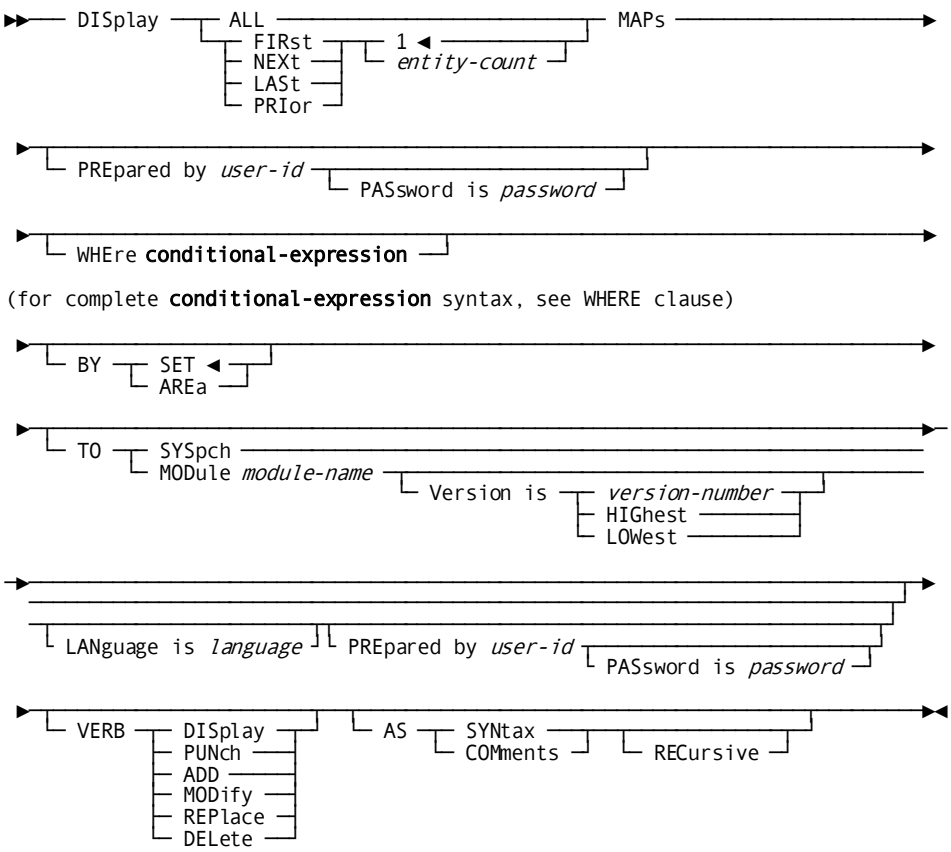
(for complete **user-specification** syntax, see USER clause)


PUBlic access is
ALLOwed
for
ALL
NONe
UPDate
MODify
REPlace
DELete
DISplay
INClude
EXClude
within PANel panel-name
Version is
version-number
HIGhest
LOWest
INClude
EXClude
within SYStem system-name
Version is
version-number
HIGhest
LOWest
TEXt is user-text
INClude
EXClude
class-name is attribute-name
TEXt is user-text
INClude
EXClude
MODule is module-name
LANguage is
HELp
TABle
Version is
version-number
HIGhest
LOWest
TEXt is user-text
INClude
EXClude
TABle table-name
Version is
version-number
HIGhest
LOWest
TEXt is user-text
EDIt
COMments
DEFinition
comment-key
edit-instruction
QUIT
COMments
DEFinition
comment-key
is
NULl
comment-text

DISPLAY/PUNCH MAP Statement (for a single map)



DISPLAY/PUNCH MAP Statement (for multiple maps)



(for complete **conditional-expression** syntax, see WHERE clause)

Parameters

MAP name is *map-name* within PANel *panel-name*

Identifies a new map to be established in the dictionary or an existing map to be modified, replaced, deleted, displayed, or punched. *Map-name* must be a 1- through 8-character alphanumeric value. For ADD operations, the MAP NAME clause must be further qualified by the WITHIN PANEL clause. *Panel-name* must reference an existing panel (see PANEL (SCREEN) (see page 234)).

NEW NAMe is *new-map-name*

Specifies a new name for the requested map. This clause changes only the name of the map occurrence; it does not alter or delete previously defined relationships in which the map participates. Subsequent references to the map occurrence must specify the new name. The concatenation of the new map name and version number must not duplicate that of any other map in the dictionary. If no version is specified, the version associated with the original name is used.

Note: The NEW NAME option cannot be used with maps created using the DC/UCF mapping compiler.

NEW Version is *new-version-number*/NEXt HIGhest/NEXt LOWest

Specifies a new version number for the named map. The map name and new version number must not duplicate that of an established map.

Note: The NEW VERSION option cannot be used with maps created using the DC/UCF mapping compiler.

within PANel *panel-name*

Associates (INCLUDE) the named map with or disassociates (EXCLUDE) it from a panel. *Panel-name* must be the 1- through 32-character name of an existing panel. The named map can be associated with only one panel. In DC/UCF environments, the mapping compiler establishes and maintains map/panel relationships directly and requires that each map be associated with a panel.

within SYStem *system-name*

Associates (INCLUDE) the map with or disassociates (EXCLUDE) it from a system. *System-name* must be the 1- through 32-character name of an existing system. The WITHIN SYSTEM clause is documentation only.

MODule is *module-name* language is HELp/TABle

Associates (INCLUDE) the named map with or disassociates (EXCLUDE) it from a module. The language of the module must be HELP or TABLE. *Module-name* must be the 1- through 32-character name of an existing module.

TABle *table-name*

Associates (INCLUDE) the named map with or disassociates (EXCLUDE) it from a table. *Table-name* must be the 1- through 8-character name of an existing table.

WITh/ALSo WITh/WITHOut

Includes or excludes the specified options when the named map is displayed or punched. Detailed information for each DISPLAY/PUNCH option is under SET OPTIONS Syntax (see page 36). The option that is listed below presents special considerations for this entity type.

DETails

Includes the DESCRIPTION clause.

Usage

If you specify REPLACE

If you specify REPLACE, the DDDL compiler initializes to defaults and/or excludes the following options:

- DESCRIPTION
- COMMENTS/DEFINITION/*comment-key*
- WITHIN SYSTEM (if built by the DDDL compiler)
- USER REGISTERED FOR
- PUBLIC ACCESS
- Related attributes

The following relationships are not affected:

- Panels to which the named map is related
- Map fields to which the named map is related
- Records to which the named map is related
- Programs to which the named map is related

Cross-referencing between maps and tables and modules

Cross-referencing is automatic; however, you can add cross-referencing to document IDD maps (which are not accessed by the mapping facility). Before using the MODULE and TABLE clauses, make sure that modules have a language of HELP or TABLE.

Example

The following is an example of cross-referencing. The ADD statement defines the map SHIPINF within the panel SH5 and within the system INVENTORY. The MODIFY statement removes SHIPINF from the system INVENTORY and associates it with the system SHIPINV.

```
add map name is shipinf within panel sh5
    prepared by dba password is 'ice 9'
    map description 'shipping information query output'
    within system inventory.
```

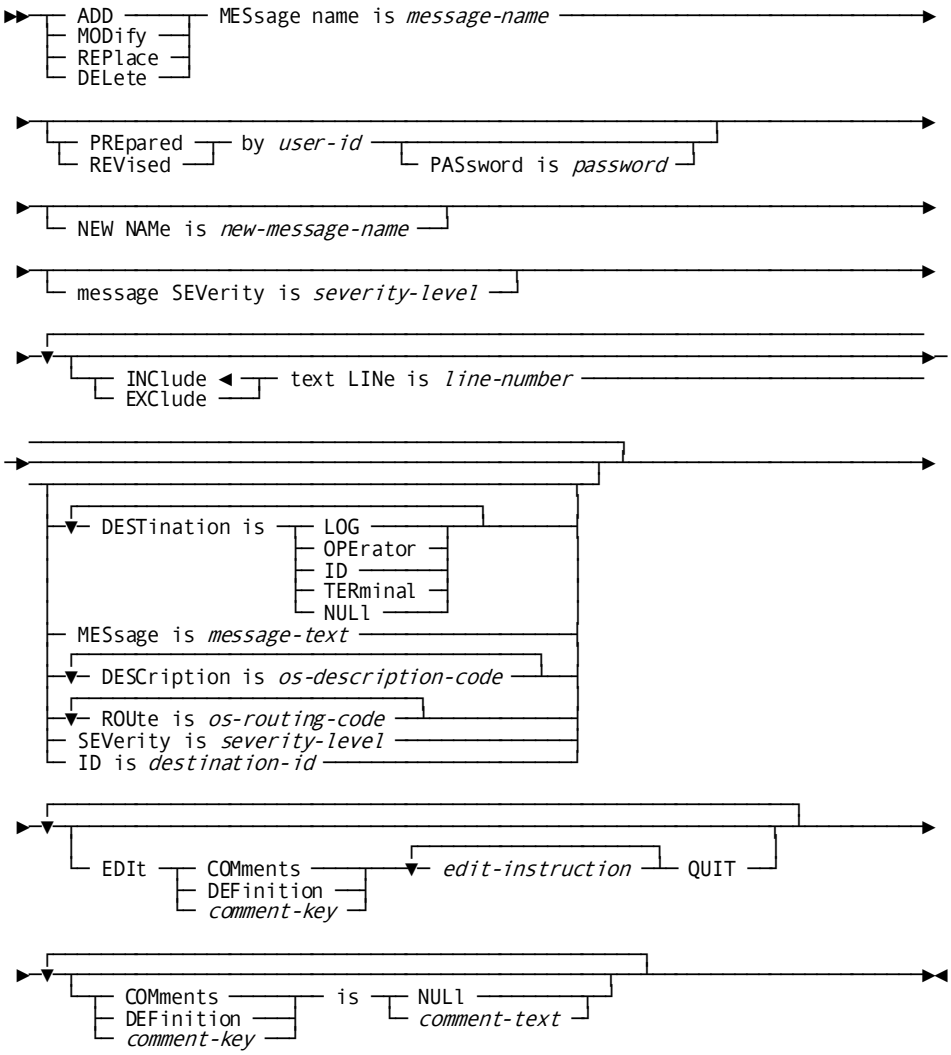
```
modify map shipinf
    revised by dba password is 'ice 9'
    exclude within system inventory
    within system shipinv.
```

MESSAGE

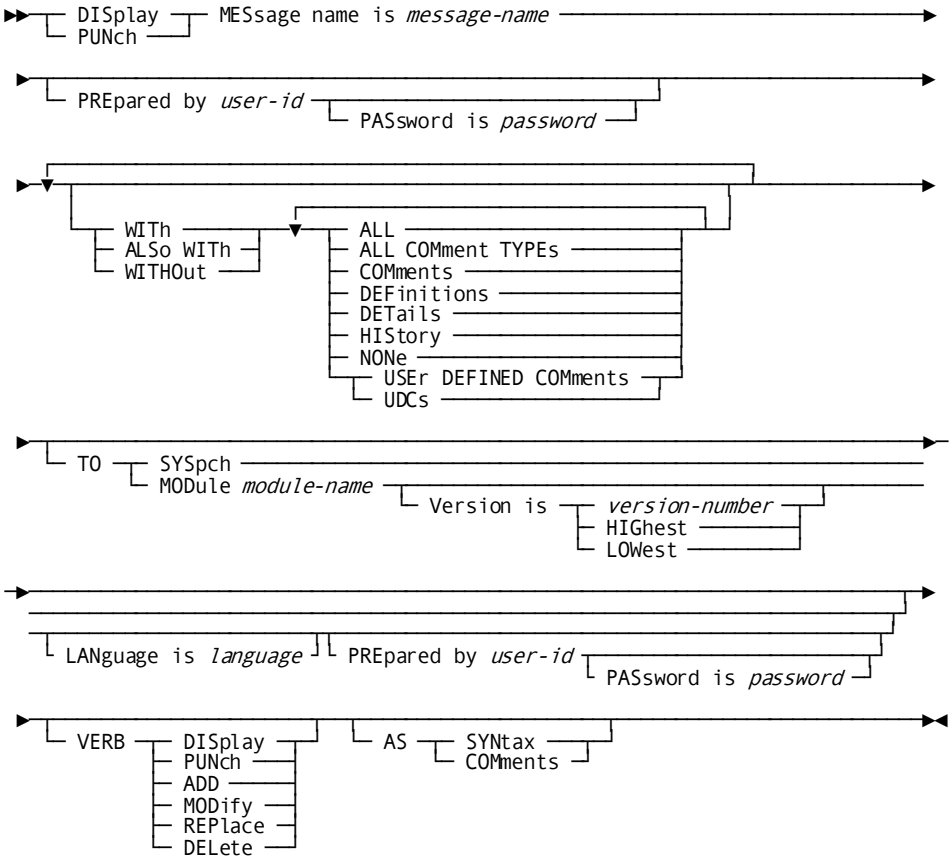
The MESSAGE statement maintains in the dictionary informational messages that are used by CA IDMS software. If the SET OPTIONS statement specifies SECURITY FOR IDMS-DC IS ON, the user must be assigned the proper authority to issue MESSAGE statements.

Syntax

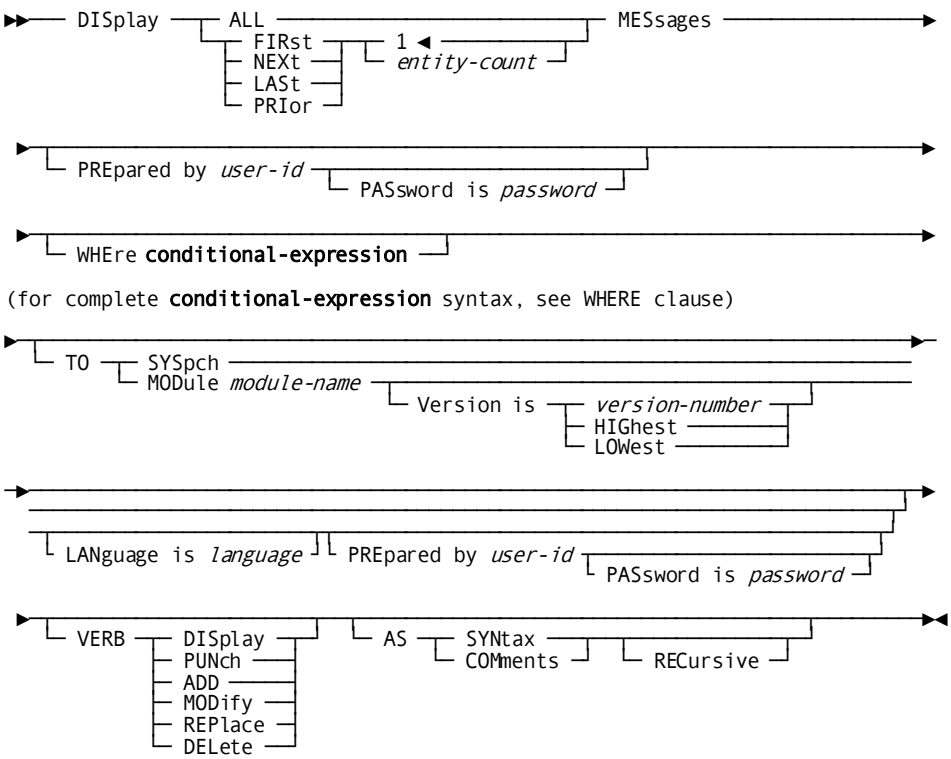
MESSAGE Statement



DISPLAY/PUNCH MESSAGE Statement (for a single message)



DISPLAY/PUNCH MESSAGE Statement (for multiple messages)



Parameters

MESSage name is *message-name*

Identifies a new message to be established in the dictionary or an existing message to be modified, replaced, deleted, displayed, or punched. *Message-name* must be a 1- through 8-character identifier. When used in conjunction with the CA IDMS/DC WRITE LOG statement, the identifier consists of the literal DC followed by six digits.

NEW NAMe is *new-message-name*

Specifies a new name for the requested message. This clause changes only the name of the message occurrence; it does not alter or delete previously defined relationships in which this message participates. Subsequent references to the message must specify the new name.

message SEVerity is *severity-level*

Associates a severity level with all text lines in the named message. The specified severity level directs the DC/UCF system to take a specific action automatically when a program issues the associated message in response to an error condition. *Severity-level* must be a 1-digit unsigned integer in the range 0 through 9; the default is 0. See the table under **Usage** for a list of valid DC/UCF severity levels and the resulting actions.

text LINe is *line-number*

Identifies the relative position of the text line within the named message. *Line-number* must be an integer in the range 1 through 2,147,483,647 and must be unique within the message. Because contiguous line numbers need not be assigned, the user can configure messages in which the same line of text always appears last and into which additional text lines can be inserted.

DEStination is LOG/OPErator/ID/TERminal/NULl

Associates up to four destinations with the named text line or removes a previously specified destination (option for DC/UCF system messages only). Valid destinations are as follows:

- **LOG**— the system log
- **OPERATOR**— the console operator
- **ID**— any terminal known to the DC/UCF system, other than the terminal associated with the user program; the ID IS parameter (described below) assigns the actual terminal. To direct the message to multiple terminals, repeat the LINE IS clause with appropriate DESTINATION, MESSAGE, and ID options for each terminal.
- **TERMINAL**— the terminal associated with the user program
- **NULL**— no destination; this option removes a previously defined destination.

MESsage is *message-text*

Specifies the text for the named line. *Message-text* is restricted to 132 characters and comprises user-supplied literals and operands. If *message-text* must be continued, the continuation character (-) must appear as the first character in the second and subsequent input lines. If *message-text* includes embedded blanks or delimiters, it must be enclosed in site-standard quote characters. Operands that will receive replacement values at runtime can appear anywhere within the message text but must be preceded by an ampersand (&). The relative positions of the replacement values correspond to the values of the symbolic operands in the message text; for example, the first value replaces &01 and the second replaces &02.

DEScription is *os-description-code*

Associates one or more operator-message descriptor codes with the message text line (option is for OS systems only). *Os-description-code* must be an unsigned integer in the range 1 through 16 and must be a valid OS descriptor code in the supervisor services and macro instructions manual for the applicable OS system. A list of codes, separated by commas and/or blanks, can be constructed to any length (for example, 1 2 9 3 5).

ROUte is *os-routing-code*

Associates one or more operator-message routing codes with the message text line (option is for OS systems only). The specified value supplies the ROUTCDE value for WTO macros used by the DC/UCF system. If this option is used, the system administrator should ensure that the values of *os-routing-code* correspond to the values specified during the OS system generation. *Os-routing-code* must be an unsigned integer in the range 1 through 16. A list of codes, separated by commas and/or blanks, can be constructed to any length (for example, 1 2 9 3 5).

SEVerity is *severity-level*

Associates a severity level with the named text line. This specification is documentation only.

ID is *destination-id*

Identifies the terminal to which the message is to be sent.

WITh/ALSo WITh/WITHOut

Includes or excludes the specified options when the named message is displayed or punched. Detailed information for each DISPLAY/PUNCH option is under SET OPTIONS Syntax (see page 36). The option that is listed below presents special considerations for this entity type.

DETails

Includes all TEXT LINE clause specifications.

Usage

DC/UCF system message severity levels

Severity level	Meaning
0	Return to caller
1	Snap task and return to caller
2	Snap system and return to caller
3	Snap task and abend task
4	Snap system and abend task
5	Abend task
6	Not assigned
7	Not assigned
8	Snap system and abend system
9	Abend system

Message occurrence structure

Message occurrences have the following structure:

- **Identifier**— A unique 8-character identifier. DC/UCF messages contain the prefix DC in addition to a 6-digit identifier in the range 000001 through 999999;
- **Message text lines**— Individual lines of literals and operands. Each line is identified by a unique line number. Operands are preceded by an ampersand (&) and receive replacement values when the message is issued at runtime.

  Operands that furnish system-defined replacement values can be placed in messages issued from online tasks. The user can include the following operands in messages issued from online tasks. The run-time system automatically substitutes the indicated data:

Operand	Replacement value
&$0	*Task ID* (from the TCETSKID field of the task control element)
&$1	*Time of day*
&$2	*Date* (ddd.yy)
&$3	*IDMS/DC system version* (from the CSADCVID field of the common system area)
&$4	*Current task code* (from the task control element)
&$5	*Current program* (from the TCECPRGM field of the task control element)
&$6	*User ID* (from the TCESONRC field of the task control element)
&$7.	*CA IDMS/DC system node name* (from the SDSNODE field in the SDS block)
&$8.	*CA IDMS/DC release number*
&$9	*CA IDMS/DC tape volser*

- **User-defined destination**— A code associated with each message text line. Codes are available to direct messages to the console operator, the system log, or to specific terminals.

  Note: Destinations for messages used by CA ADS Batch are documentation only.
- **Operating system and DC/UCF system information**— A description code, route code, and/or a severity level associated with each line of text, according to user-established requirements.

Detailed information about using messages in DC/UCF application programs appears in the *CA IDMS Navigational DML Programming Guide*.

If you specify REPLACE

If you specify REPLACE, the DDDL compiler initializes to defaults or excludes the following options:

- MESSAGE SEVERITY
- TEXT LINE
- COMMENTS/DEFINITION/*comment-key*

Example

In the following example, the ADD statements define two DC/UCF messages; note that message text can be continued between input lines if necessary.

```
add message dc317017
    text line 1
       destination is operator
       severity is 1
       message is
       'end of file encountered before end of idms'
       -'statement.'.

add message dc317020
    text line 1
        destination is operator
        severity is 0
        message is
           'duplicate parameter within this idms statement'.
```

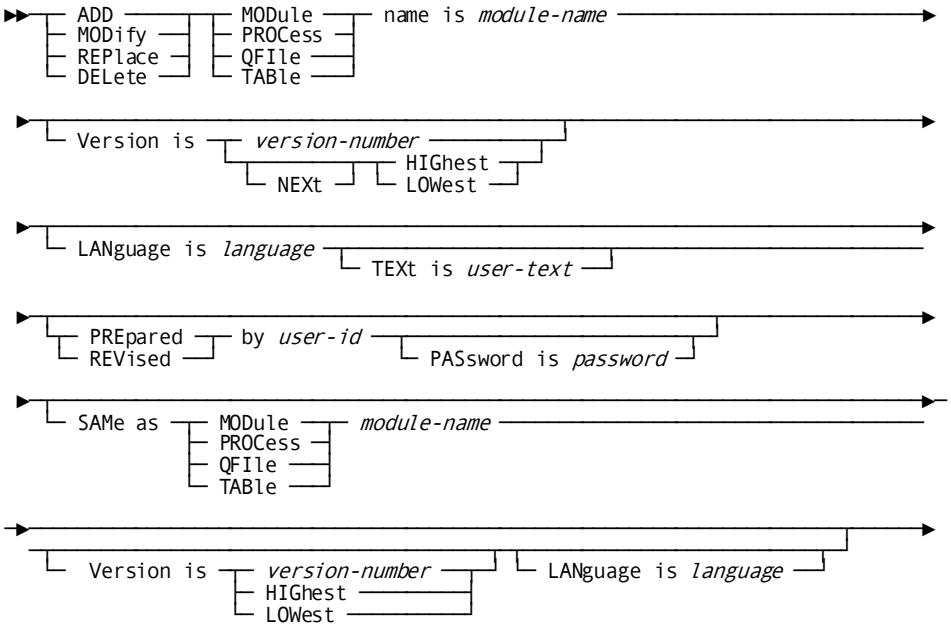
MODULE (PROCESS/QFILE/TABLE)

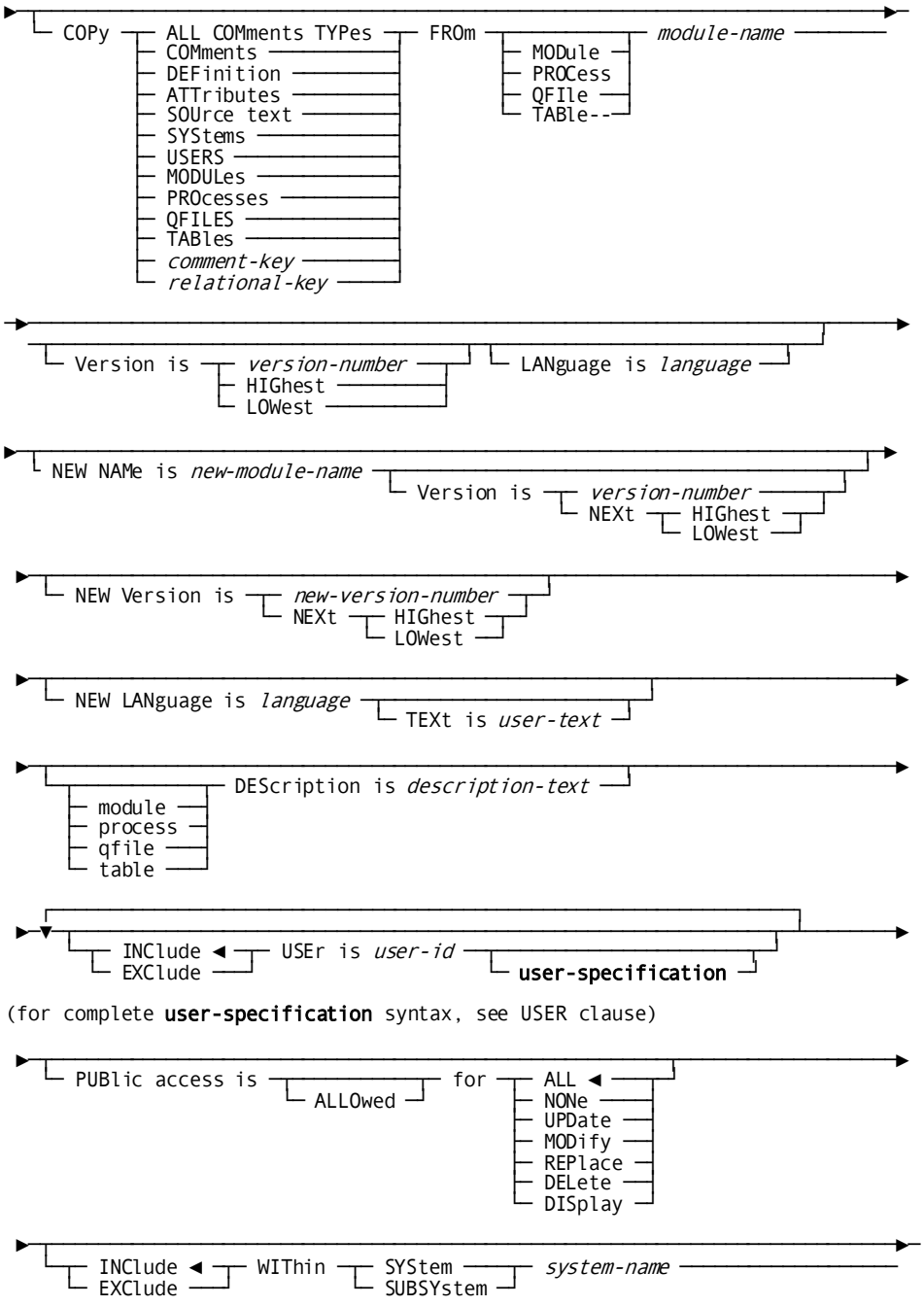
MODULE statements are used to define source code for modules, CA ADS processes, and CA OLQ qfiles and to document edit and code tables. Modules can be standard modules or sequences of DDDL commands, signon profiles, or system command lists. Tables are used by the CA IDMS Mapping Facility for automatic editing and error handling. Optional MODULE statement clauses relate modules, processes, qfiles, and tables to users, systems, and other modules; establish attribute/entity relationships; and maintain documentation entries.

If the SET OPTIONS statement specifies SECURITY FOR IDD IS ON, the user must be assigned the proper authority to issue MODULE statements.

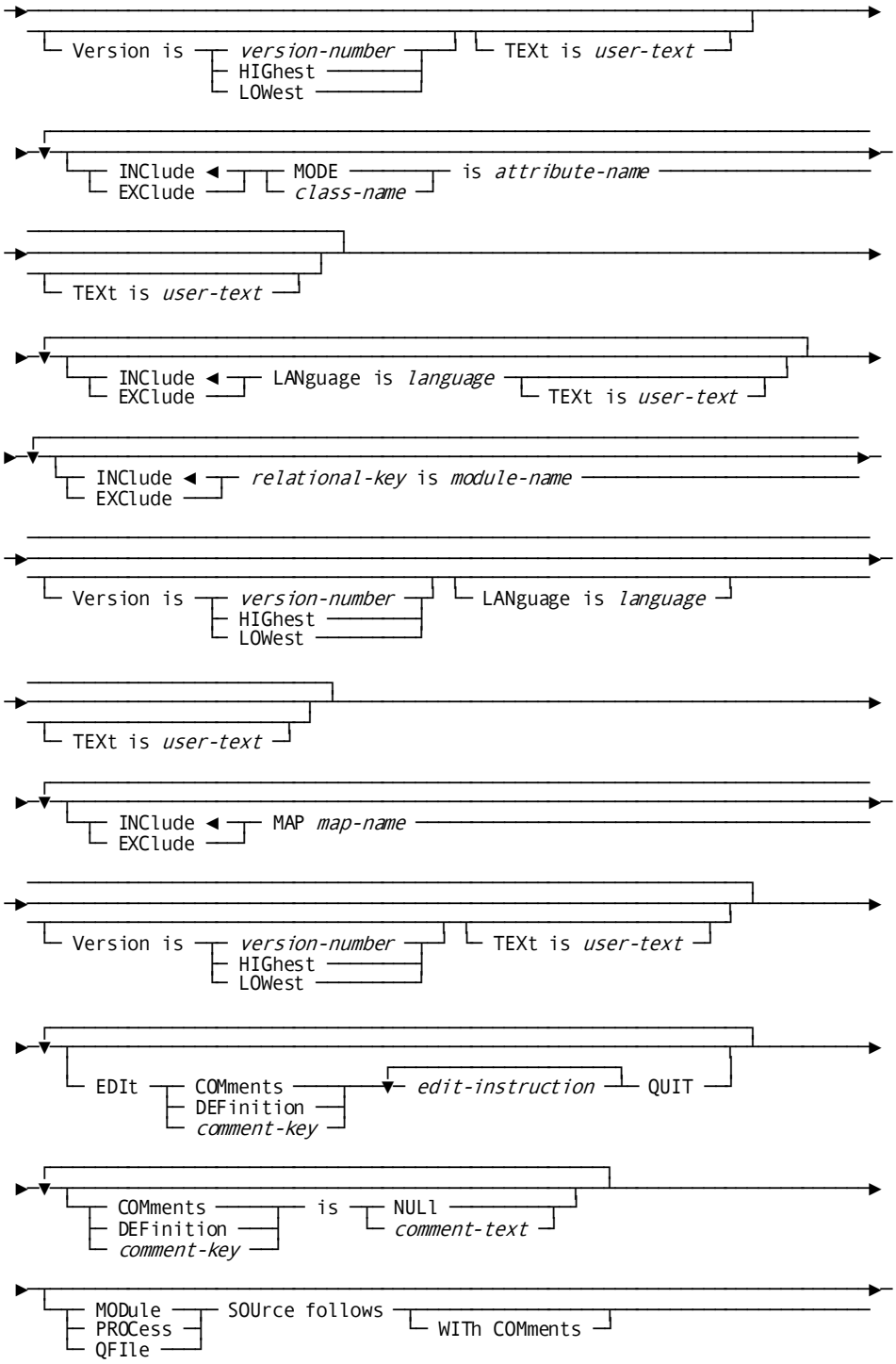
Syntax

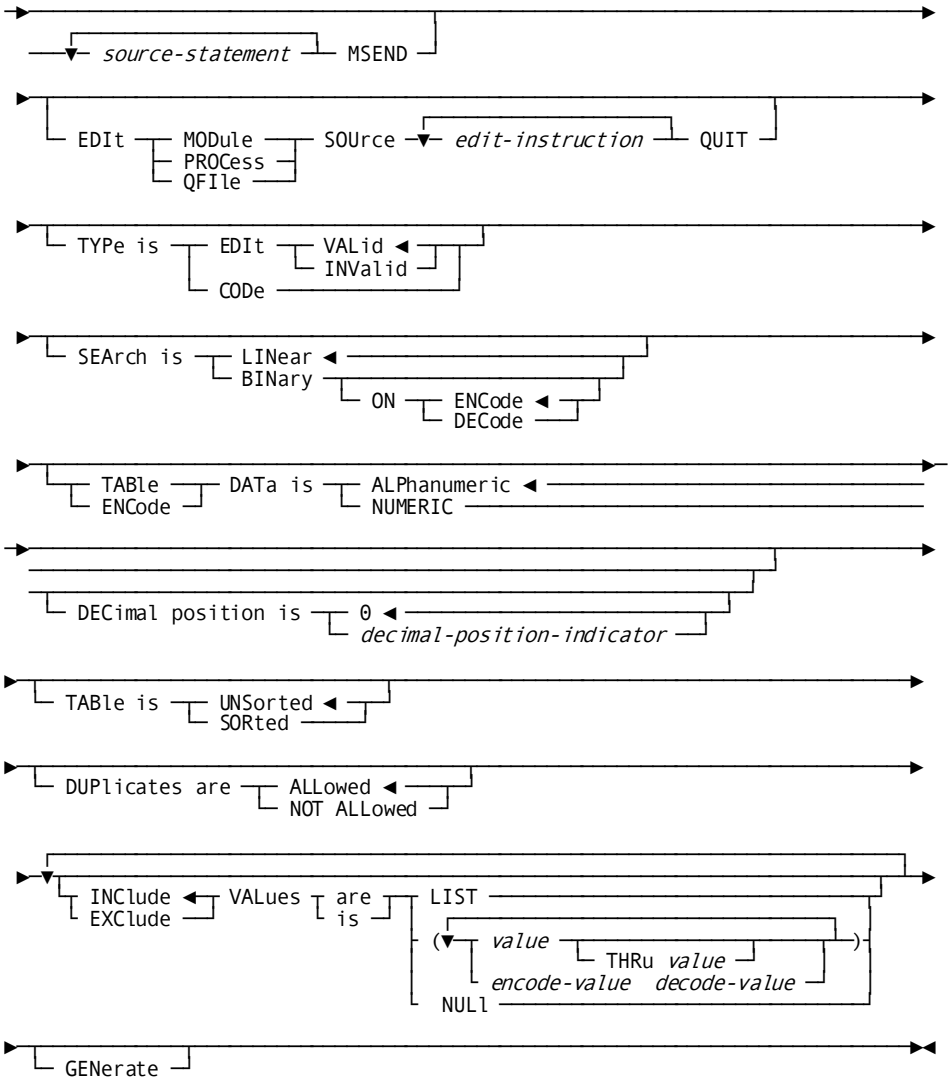
MODULE/PROCESS/QFILE/TABLE statement



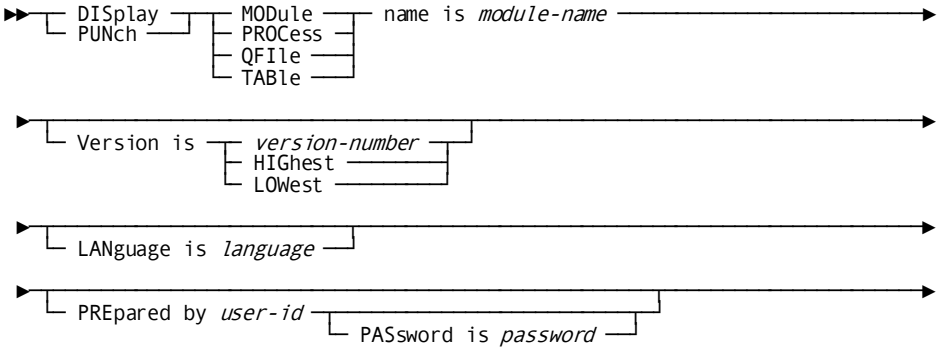


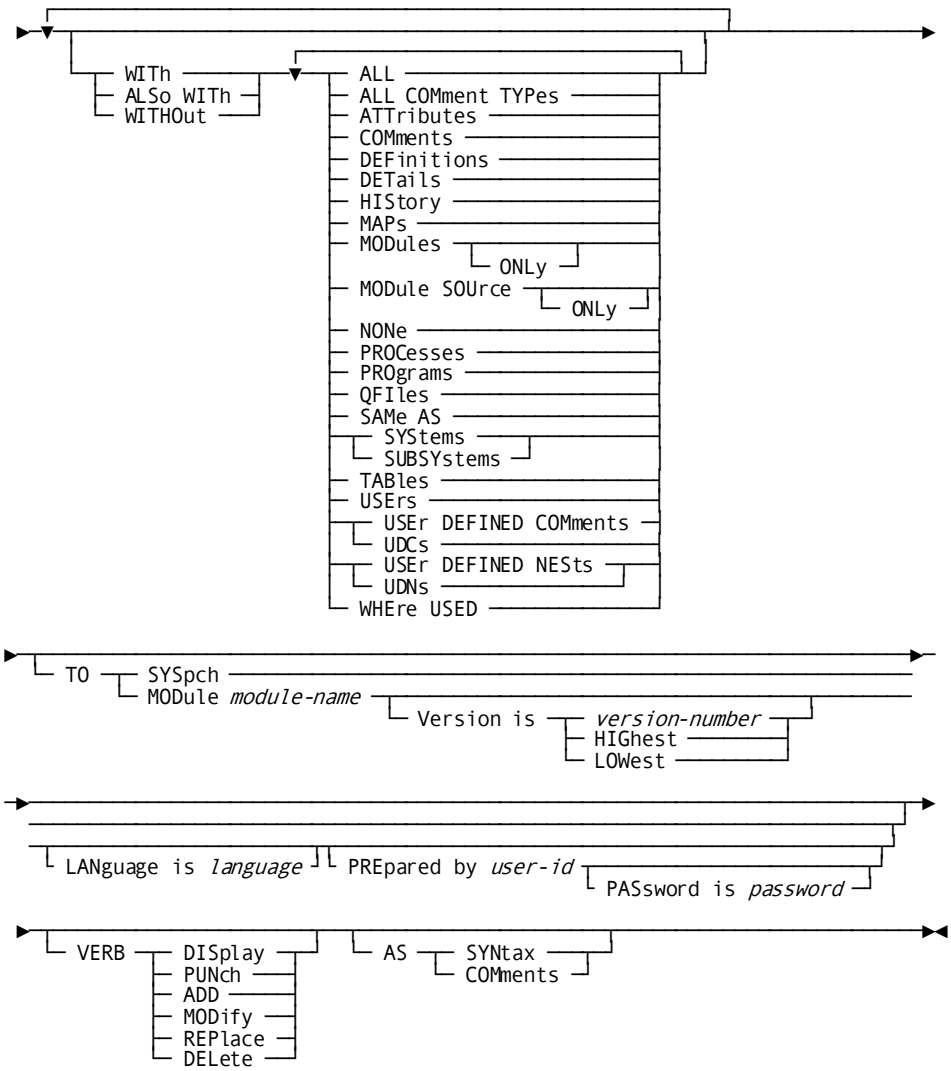
(for complete **user-specification** syntax, see USER clause)


Version is
version-number
HIGhest
LOWest
TEXt is user-text
INClude
EXClude
MODE
class-name
is attribute-name
TEXt is user-text
INClude
EXClude
LANguage is language
TEXt is user-text
INClude
EXClude
relational-key is module-name
Version is
version-number
HIGhest
LOWest
LANguage is language
TEXt is user-text
INClude
EXClude
MAP map-name
Version is
version-number
HIGhest
LOWest
TEXt is user-text
EDIt
COMments
DEFinition
comment-key
edit-instruction
QUIT
COMments
DEFinition
comment-key
is
NULl
comment-text
MODule
PROCess
QFIle
SOUrce follows
WITh COMments

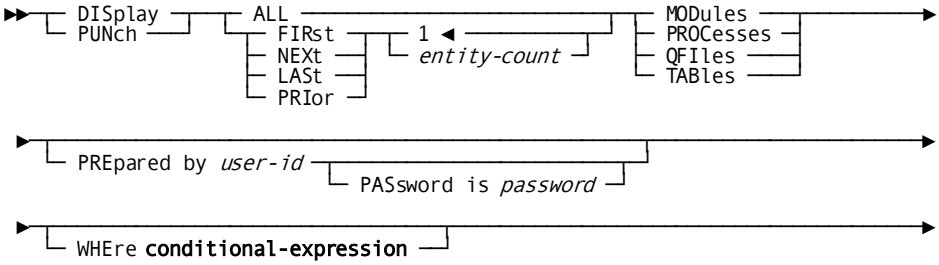


DISPLAY/PUNCH MODULE/PROCESS/QFILE/TABLE Statement (for a single module/process/qfile/table)

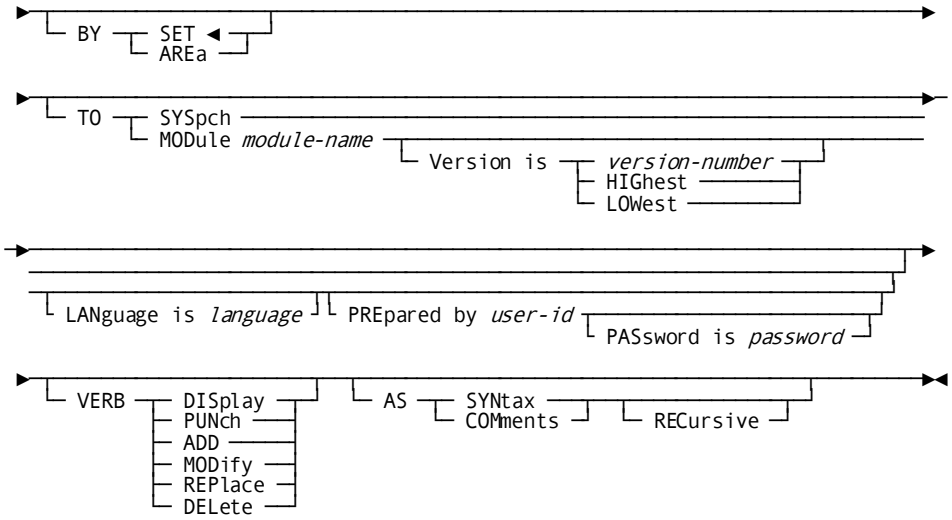




DISPLAY/PUNCH MODULE/PROCESS/QFILE/TABLE Statement (for multiple modules/processes/qfiles/tables)



(for complete **conditional-expression** syntax, see WHERE clause)


BY
SET
AREa
TO
SYSpch
MODule module-name
Version is
version-number
HIGhest
LOWest
LANguage is language
PREpared by user-id
PASsword is password
VERB
DISplay
PUNch
ADD
MODify
REPlace
DELete
AS
SYNtax
COMments
RECursive

Parameters

MODule/PROCess/QFIle/TABle name is *module-name*

Identifies a new module, process, qfile, or table to be established in the dictionary, or an existing occurrence to be modified, replaced, deleted, displayed, or punched. For modules, processes, and qfiles, *module-name* must be a 1- through 32-character alphanumeric value; for tables, *module-name* must be a 1- through 8-character value. The specified name must not duplicate the name of an existing program, map, subschema, or CA ADS dialog.

If a version of HIGHEST or LOWEST is specified (or defaulted to), the module name must be qualified with a language if the module is associated with a language. If the module name and version number do not uniquely identify a module, it must be qualified with a language.

LANguage is *language*

Qualifies the named module with a language; if used, the LANGUAGE clause must be coded directly after the name and version number. *Language* must be a 1- to 40-character language name previously established as an attribute within the LANGUAGE class. The LANGUAGE specification uniquely identifies two modules with the same name and version and is used by the DML precompilers when modules are used in programs. For command lists, acceptable languages are DC and OCF.

SAMe as MODule/PROCess/QFIle/TABle *module-name*

Copies all entries associated with the named module except the name, LANGUAGE, WITHIN MODULE, and WITHIN SYSTEM specifications. The language of the module/process/qfile/table must match the language of the module/process/qfile/table it is to be made the SAME AS.

COPy *entity-option* FROm *entity-type-name entity-occurrence-name*

Copies selected options from an entity-occurrence definition and merges the copied options into this definition. The language of the module that options are being copied from must match the language of this module.

NEW NAMe is *new-module-name*

Specifies a new name for the requested module. This clause changes only the name of the module; it does not alter or delete any previously defined relationships in which the module participates. Subsequent references to the module must specify the new name. *New-module-name* must be a 1- through 32-character value (or 1- through 8-character value in the case of a TABLE). The combination of the new module name, version number, and language must not duplicate that of an established module occurrence.

NEW Version is *new-version-number*/NEXt HIGhest/NEXt LOWest

Specifies a new version number for the named module. The combination of the module name, new version number, and language qualification must not duplicate that of an existing module.

NEW LANguage is *language*

Associates a new language with the module. *Language* is a 1- to 40-character language name previously established as an attribute in the LANGUAGE class. This clause must be used with the verb MODIFY.

The combined module name, version number, and modified language qualification must not duplicate that of an existing module. If the module has been qualified by a language, subsequent references to the module must specify the new language.

within SYStem/SUBSYstem *system-name*

Associates (INCLUDE) the named module with or disassociates (EXCLUDE) it from a system or subsystem. *System-name* must reference an existing system or subsystem.

LANguage is *language*

Associates (INCLUDE) or disassociates (EXCLUDE) a language qualification. The user can change the language qualification of a module by referencing the module using the LANGUAGE clause (described above), then by altering the language qualification with the INCLUDE/EXCLUDE LANGUAGE clause. The combination of the module name, version number, and modified language qualification must not duplicate that of an existing module. If the module has been qualified with a language, subsequent references to the module must specify the new language.

Note: The keyword INCLUDE or EXCLUDE must be present to distinguish this use of the LANGUAGE IS clause from the LANGUAGE clause used for module qualification (described previously in this list of parameters).

relational-key* is *module-name

Associates (INCLUDE) the module with or disassociates (EXCLUDE) it from another module by means of the named relational key. If the modules being related have the same name and version but different languages or if the related module has a version of HIGHEST or LOWEST and is qualified by language, the LANGUAGE parameter must be specified. See Relational Keys (see page 104) for a complete description of defining and using relational keys.

MAP is *map-name*

Associates (INCLUDE) the module with or disassociates (EXCLUDE) it from a map. *Map-name* must refer to an existing map. Only a module with a language of HELP or TABLE may be associated with a MAP.

MODule/PROCess/QFIle SOUrce follows *source-statement* MSEND

Specifies the source code to be associated with the named module, process, or qfile. Each source statement must be specified in 80-character format. DML commands coded as module source will be intercepted by the DML precompilers and translated into CALL statements when the module is copied. COPY/INCLUDE requests will also be executed when the module is copied. The MODULE/PROCESS/QFILE SOURCE FOLLOWS statement must be coded by itself on the first line; the source statements are coded on second and subsequent lines; the keyword MSEND, required to terminate the source statements, must be the first entry on the last line.

Note: The MODULE/PROCESS/QFILE SOURCE FOLLOWS clause is not valid for tables.

If you specify WITH COMMENTS, any source statement identified as a comment line (*+, --, or * in columns 1 and 2) is saved as part of module source. If you previously saved a module with comments and you redisplay the module to replace the source text, you must respecify WITH COMMENTS when you save the module.

TYPe is

Specifies the table type (for the TABLE entity type only). This clause is required for ADD operations.

EDIt

Defines a table that provides a list of values or ranges of values to be checked in a data field.

VALid/INValid

Specifies whether the list contains valid or invalid values; VALID is the default.

CODe

Defines a table that translates internal codes in a record to external report values (decoding) or maps external values back to internal record codes (encoding).

SEArch is

Specifies the method by which the table is to be searched (TABLE entity only).

LINear

Starts the search at the beginning of the table and proceeds line by line until the specified value is found. LINEAR is the default.

BINary

Starts the search in the middle the table and halves the table each time a comparison is made until the specified value is found. Edit tables to be searched by the binary method can include only single values.

ON ENCode/DECode

Specifies whether the binary search is to be performed on encoded or decoded table values (option for code tables only). The default is ENCODE.

TABle/ENCode DATa is ALPhanumeric/NUMERIc

Specifies whether the values in the table are alphanumeric or numeric; ALPHANUMERIC is the default (option is for the TABLE entity only).

DECimal position is *decimal-position-indicator*

Specifies the position of the decimal point (NUMERIC option only). Note that this is an assumed decimal position; no decimal point appears in the values.

TABle is

Specifies whether the table is to be maintained in the dictionary as a sorted table (TABLE entity type only).

UNSorted

Sorts table values at runtime in the order in which they are placed in the dictionary. UNSORTED is the default.

Note: A binary searched table can be stored with the UNSORTED attribute; however, the table will be sorted automatically when it is generated.

SORted

Sorts table values alphabetically or numerically as they are added to the table.

DUPlicates are ALLowed/NOT ALLowed

Specifies whether duplicate values can be included in sorted tables (TABLE entity type only). ALLOWED is the default. Note that DUPLICATES ARE NOT ALLOWED must be specified for binary searched tables.

VALues are

Specifies whether table values are to be listed, inserted, or removed (TABLE entity type only).

LIST

Lists the table values or pairs of values (code tables only) stored in the dictionary.

value* THRu *value

Inserts single values, ranges of values, combinations of single values and ranges, or null values in the edit table. *Value* must be a 1- through 34-character value and must be enclosed in parentheses.

encode-value decode-value

Specifies pairs of values to be inserted in the code table. *Encode-value* must a 1- through 34-character value; *decode-value* must be a 1- through 62-character value. The specified values must be enclosed in parentheses.

NULl

Removes all values from the table.

GENerate

Causes a load module containing all the values in the table to be placed in the dictionary load area (TABLE entity type only). The generated load module has the same name and version number as the named table.

WITh/ALSo WITh/WITHOut

Includes or excludes the specified options when the module, process, qfile, or table is displayed or punched. Detailed information for each DISPLAY/PUNCH option is under SET OPTIONS Syntax (see page 36). The options that are listed below present special considerations for this entity type.

DETails

Includes either the DESCRIPTION clause for modules, processes, and qfiles or table data for tables.

MAPs

Includes cross-referencing information.

WITh MODule SOUrce

Displays/punches the MODULE statement syntax, the source code associated with the module, and any other DISPLAY options.

WITh MODule SOUrce ONLy

Displays/punches only the source code associated with the module; does not display the surrounding ADD MODULE statement (ADD MODULE NAME IS *module-name* ... MSEND).

Usage

MODULE statement considerations

The following considerations apply to the MODULE statement:

- The user can choose to add processes, qfiles, and tables by using the MODULE statement or a DDDL statement that specifically names the entity (that is, a PROCESS statement, QFILE statement, and TABLE statement).

  Note: For more information about statements for qfiles, processes, and tables, see the QFILE (see page 272), PROCESS (see page 247), and TABLE (see page 341) sections in this chapter.

- The reserved words MODULE, PROCESS, QFILE, and TABLE are interchangeable within MODULE statement clauses, unless otherwise noted. In the following discussion, the term *module* applies to processes, qfiles, and tables, unless otherwise noted.
- Qfile occurrences are stored as specially identified module records in the dictionary and are automatically associated with the LANGUAGE class through the OLQ attribute.
- Processes are stored as specially identified module records in the dictionary and are automatically associated with the LANGUAGE class through the PROCESS attribute.
- Tables defined by means of the TABLE statement are referred to as stand-alone tables. The RECORD ELEMENT substatement (described under RECORD (REPORT/TRANSACTION) (see page 287) later in this chapter) is used to define built-in tables. For a description of stand-alone and built-in tables, refer to *CA IDMS Mapping Facility Guide*. Tables are automatically associated with the LANGUAGE class through the TABLE attribute.
- DC/UCF command lists, stored as occurrences of the MODULE entity type, must be assigned a language of DC.

If you specify PROCESS, QFILE, or TABLE

If you specify PROCESS, QFILE, or TABLE, the DDDL compiler supplies the appropriate language automatically.

If you specify REPLACE

If you specify REPLACE, the DDDL compiler initializes to defaults and/or excludes the following options:

- DESCRIPTION
- Related modules
- USER REGISTERED FOR
- Related attributes
- PUBLIC ACCESS
- MODULE SOURCE or table data
- WITHIN SYSTEM
- COMMENTS/DEFINITION/*comment-key*

The following relationships are not affected:

- Modules to which the named module is a related module
- Users accessing the named module
- Programs using the named module
- LANGUAGE specification

Cross-referencing modules with maps

You can add cross-referencing from a module to any map (maps used by the CA IDMS mapping facility or documentation IDD maps). Cross-referencing can only be established for modules having a language of HELP or TABLE.

You must remove all cross-referencing before you can delete a module.

Examples

The following examples illustrate three forms of the MODULE/PROCESS/QFILE/TABLE statement. Note that the LANGUAGE class with MANUAL PLURAL qualifiers and the MODE class with AUTOMATIC PLURAL qualifiers are automatically defined during IDD installation and that the DML precompilers inspect entries within entity occurrences that specify the MODE and LANGUAGE classes when processing IDMS COPY statements.

The following statements add the *module* ACCOUNTING-STATISTICS, assigning it a language of COBOL and relating it to the attribute BATCH by means of the predefined class MODE and to the system STANDARDS, and modify the module source, inserting line number 305 and comment text.

```
add module accounting-statistics
    language is cobol
    mode is batch
```

```
    within system standards
    module source follows
       accounting-statistics.
           accept db-statistics from idms-statistics.
           display ' program name' program-name.
           display '# database requests' calls-to-idms.
           display '# pages read' pages-read.
           display '# cpu time' system-time.
           display '# elapsed time' wait-time.
           display '# pages written' pages-written.
           display '# pages requested' pages-requested.
           display '# records requested' lines-requested.
           display '# record current' recs-current.
     msend.

modify module accounting-statistics
     edit module source
     insert 305
     display 'job acctg info' acct-info.
     cend
     quit
     comments 'module for displaying statistics'
     .
```

The following statements add the *process* GET-A-CUSTOMER to the dictionary and modify the process UPDATE-A-CUSTOMER; note that the language qualification for GET-A-CUSTOMER is automatically supplied.

```
add process get-a-customer
    module source follows
     ready.
     obtain calc customer.
     if db-rec-not-found
       then do
         display message
         text is 'customer does not exist -- will be added'.
         end.
       else do
         display message
         text is 'customer exists -- will be updated'.
         end.
    msend.

modify process update-a-customer
    module source follows
     ready usage-mode is update.
     obtain calc customer.
     if db-rec-not-found
       then do
         store customer.
```

```
          display message
          text 'new customer has been added'.
          end.
        else do
          modify customer.
          display message
          text is 'customer has been updated'.
          end.
    msend.
```

The following statements add the *tables* MONTHTBL and DECODMTH. MONTHTBL is an edit table that contains the valid values 1 through 12 for the months of the year; DECODMTH is a code table that relates the names of the months to the 2-digit month codes used in the database. DECODMTH is defined by means of the keyword MODULE qualified by a LANGUAGE clause.

```
add table name is monthtbl
    table description is 'valid months'
    type is edit
    search is linear
    table data is alphanumeric
    table is unsorted
    values are ( 01 thru 12 )
    .

add module name is decodmth version is 1
    language is table
    table description is 'month code convert'
    type is code
    search is linear
    encode data is alphanumeric
    table is unsorted
    duplicates are allowed
    values are ( 01 jan 02 feb 03 mar 04 apr
         05 may 06 jun 06 june 07 jul 07 july
         08 aug 09 sep 10 oct 11 nov 12 dec
         not found other )
    .
```

The following statements add the *modules* MISPROFILE and JMC-CLIST. MISPROFILE is a signon profile that contains three commands. JMC-CLIST is a command list that can be invoked at runtime. Command lists and signon profiles are identified by the LANGUAGE IS DC clause.

```
add module misprofile version 1
    language is dc
    module source follows
        dcuf set dbnode system82
        dcuf set dbname misdata
        dcuf set print class 3
```

```
    msend.

add module jmc-clist.
    language is dc
    module source follows
        dcmt display time
        dcmt display active tasks
        dcmt display active storage
        dcuf show users all
    msend.
```

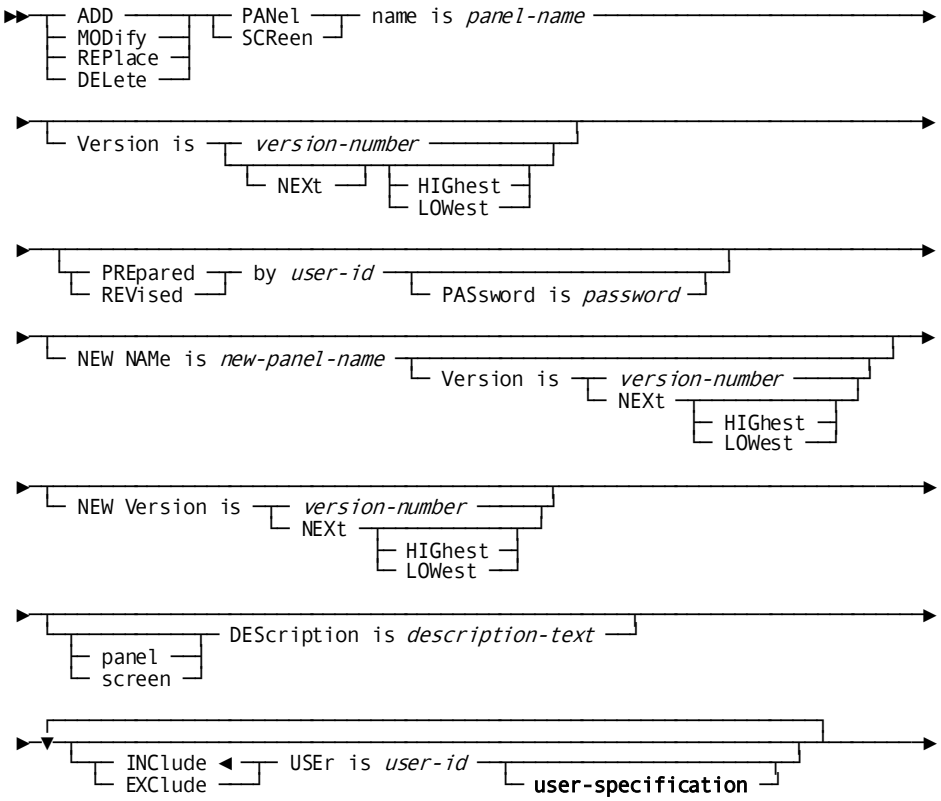
PANEL (SCREEN)

PANEL statements associate documentation entries and users with maps that are used in the 3270-type terminal environment. The keywords PANEL and SCREEN are synonymous; all screens are reported as panels, regardless of the DDDL syntax used to establish and/or maintain the occurrences. Optional clauses relate panels to established users and accommodate attribute/entity relationships. The MAP statement is used to associate established panels with maps; see MAP (see page 205) earlier in this chapter for further details. When a panel is deleted, all maps associated with it are also deleted. Panel occurrences can document either existing panel definitions or anticipated panel requirements.

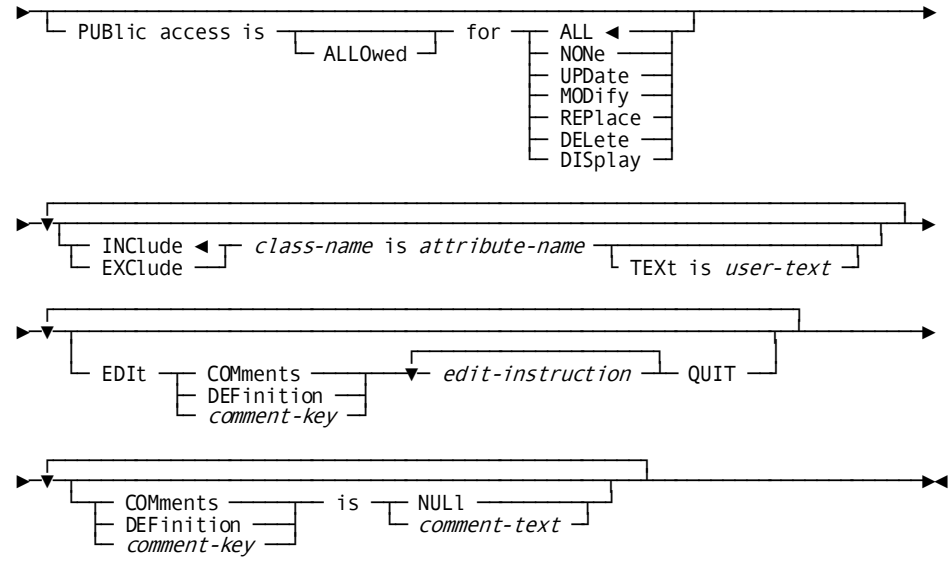
If the SET OPTIONS statement specifies SECURITY FOR IDMS-DC IS ON, the user must be assigned the proper authority to issue PANEL statements.

Syntax

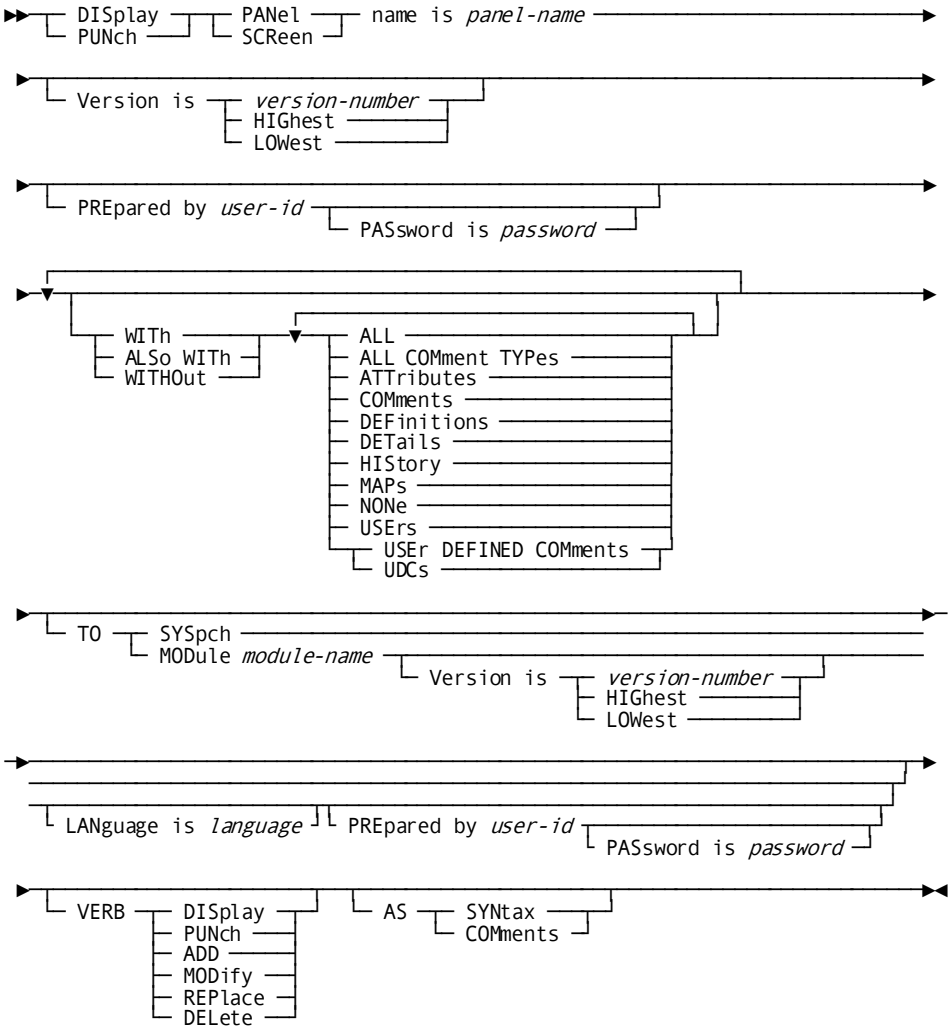
PANEL (SCREEN) statement



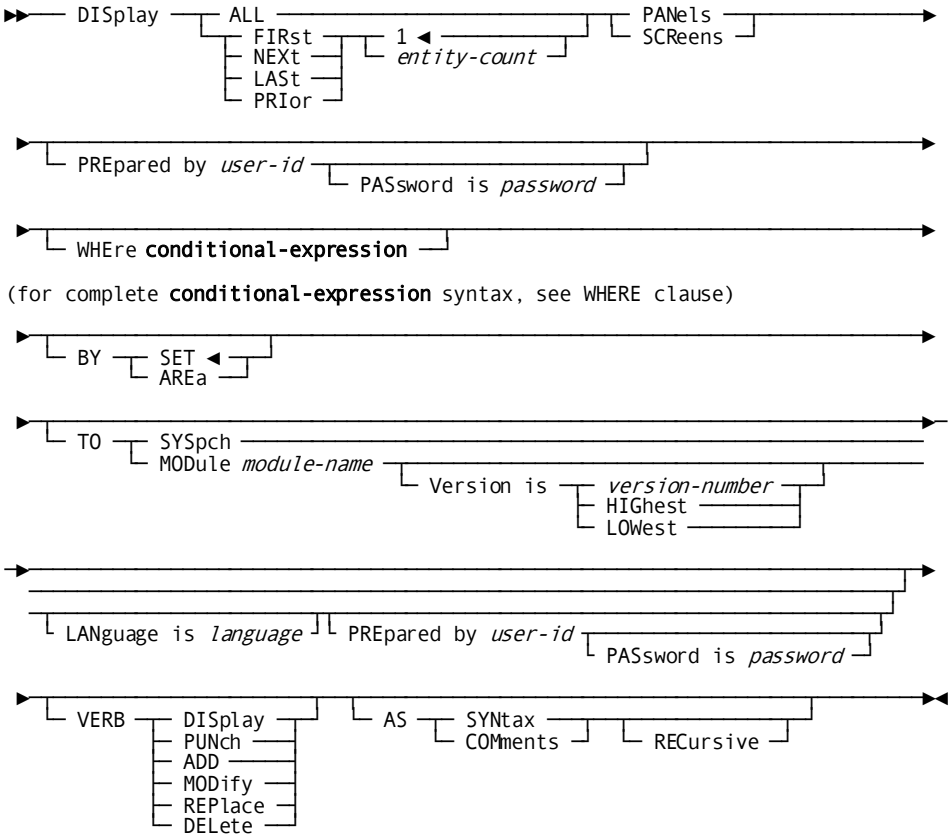
(for complete **user-specification** syntax, see USER clause)



DISPLAY/PUNCH PANEL (SCREEN) (for a single panel)



DISPLAY/PUNCH PANEL (SCREEN) (for multiple panels)



Parameters

PANel (SCReen) name is *panel-name*

Identifies a new panel to be established in the dictionary, or an existing panel to be modified, replaced, deleted, displayed, or punched. *Panel-name* must be a 1- through 32-character alphanumeric value.

NEW NAMe is *new-panel-name*

Specifies a new name for the requested panel. This clause changes only the name of the panel occurrence; it does not alter or delete any previously defined relationships in which the panel participates. Subsequent references to the panel must specify the new name. *New-panel-name* must be a 1- through 32-character value. The concatenation of the new panel name and version number must not duplicate that of an existing panel.

NEW Version is *new-version-number*/NEXt HIGhest/NEXt LOWest

Specifies a new version number for the named panel. The panel name and new version number must not duplicate that of an existing panel.

WITh/ALSo WITh/WITHOut

Includes or excludes the specified options when the panel is displayed or punched. Detailed information for each DISPLAY/PUNCH option is under SET OPTIONS Syntax (see page 36). The option that is listed below presents special considerations for this entity type.

DETails

Includes the DESCRIPTION clause.

Usage

If you specify REPLACE

If you specify REPLACE, the DDDL compiler initializes to defaults and/or excludes the following options:

- DESCRIPTION
- USER REGISTERED FOR
- COMMENTS/DEFINITION/*comment-key*
- Related attributes
- PUBLIC ACCESS

The following relationships are not affected:

- Data fields to which the named panel is related
- Maps to which the named panel is related

Example

In the following example, the ADD statement defines panel SH5.

```
add panel name is sh5
    panel description is 'common shipping queries'.
```

PHYSICAL TERMINAL

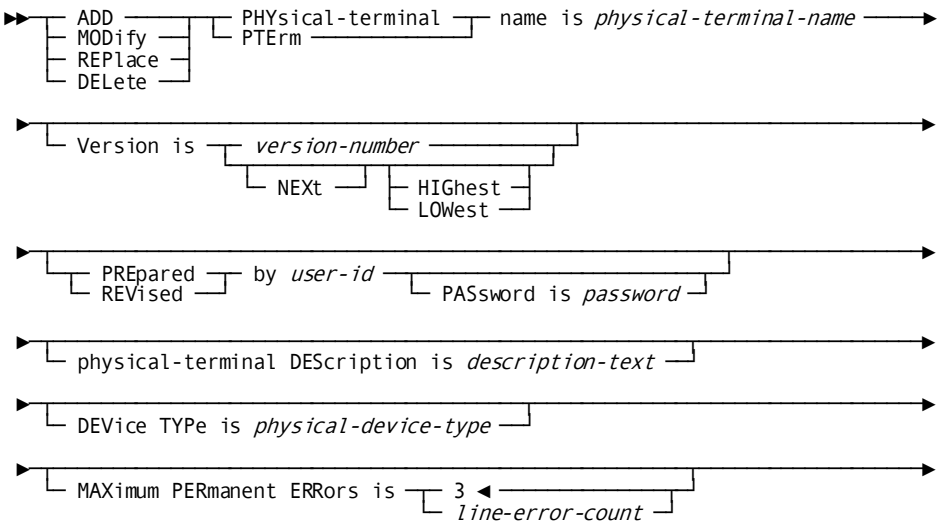
PHYSICAL-TERMINAL statements document the physical CRT, TTY, and printer devices in a teleprocessing system. In the DC/UCF environment, physical terminals are associated with logical terminals. In CA IDMS DDS environments, DDS physical terminals are associated with DDS lines (refer to *CA IDMS DDS Design and Operations Guide*).

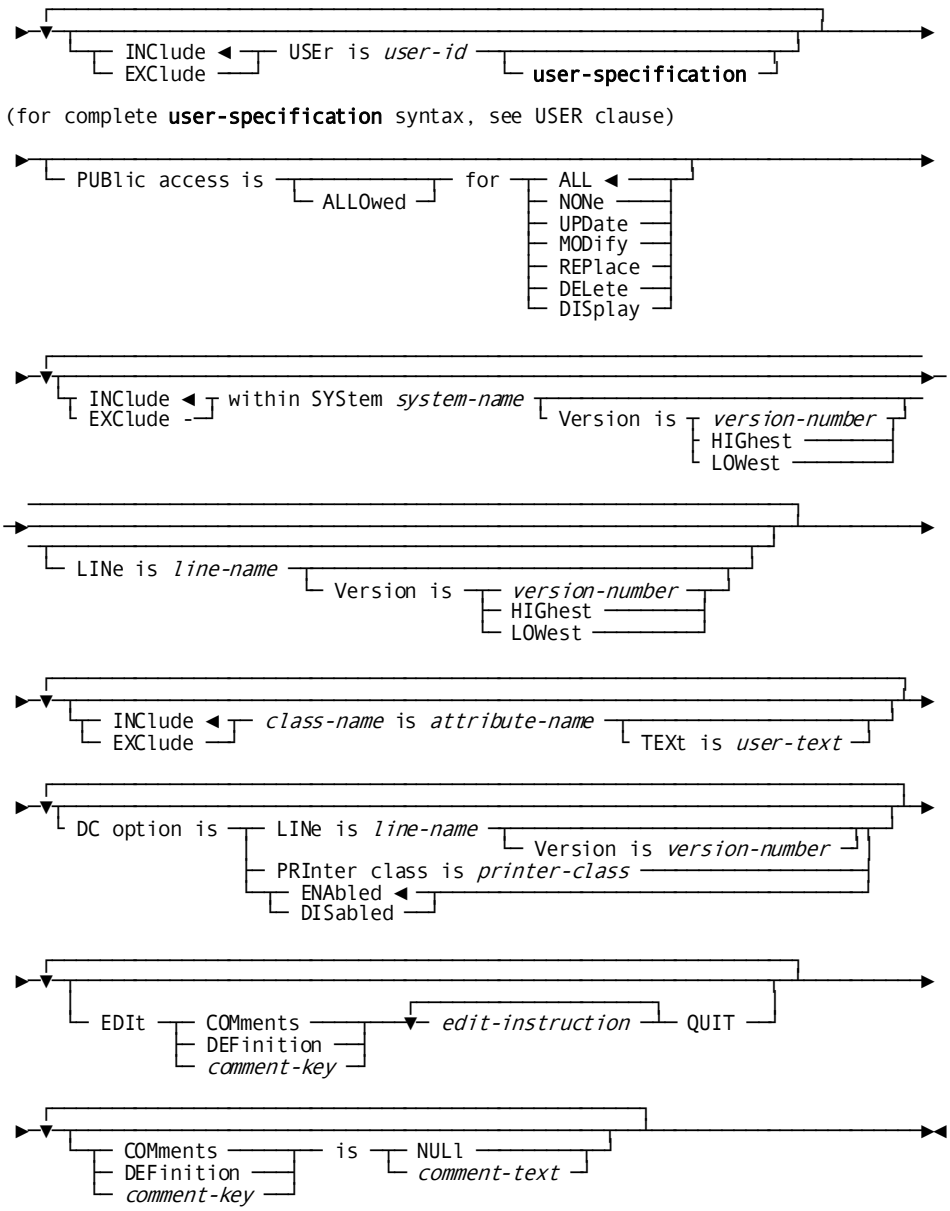
Note: It is recommended that you maintain PHYSICAL TERMINAL definitions using the system generation compiler, *not* the DDDL compiler. If a system generation component is processed by the DDDL compiler, only dictionary security is checked, *not* system generation security. For more information on using the system generation compiler, refer to *CA IDMS System Generation Guide*.

If the SET OPTIONS statement specifies SECURITY FOR IDMS-DC IS ON, the user must be assigned the proper authority to issue PHYSICAL-TERMINAL statements.

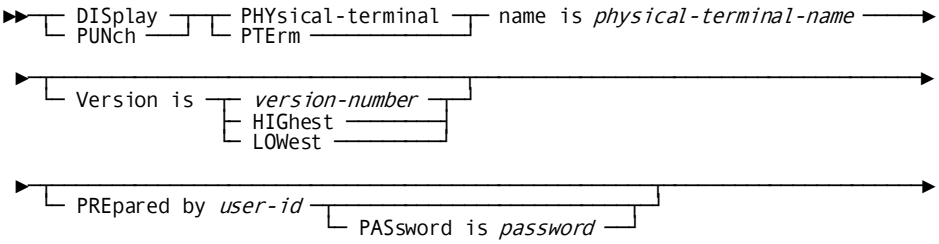
Syntax

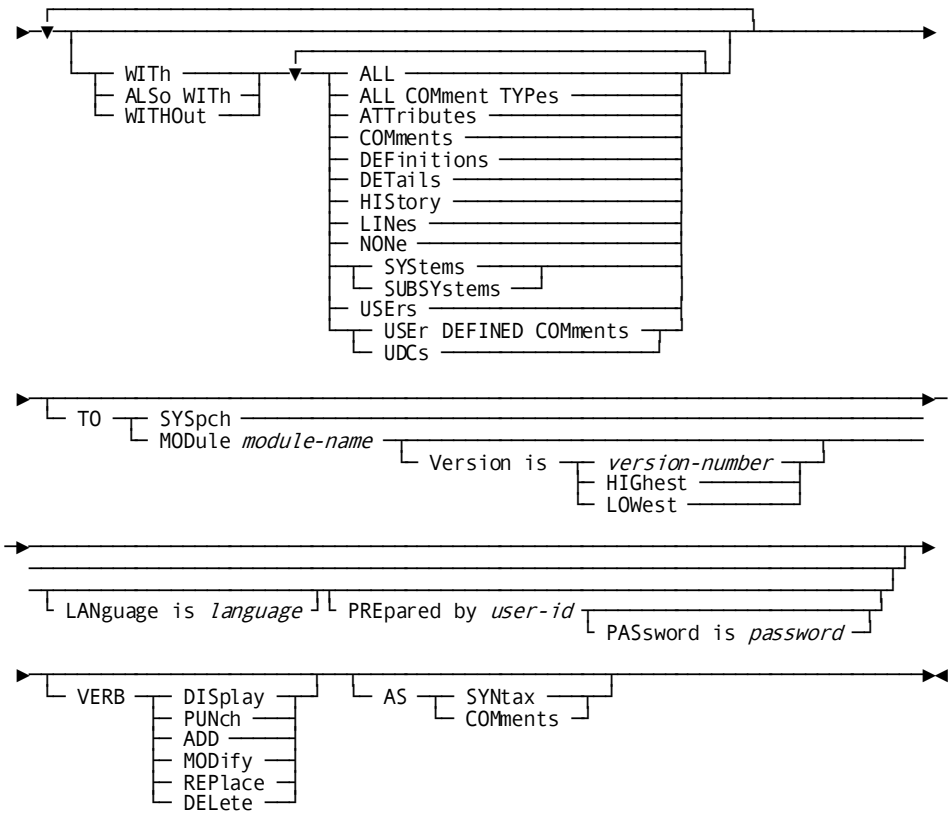
PHYSICAL-TERMINAL Statement



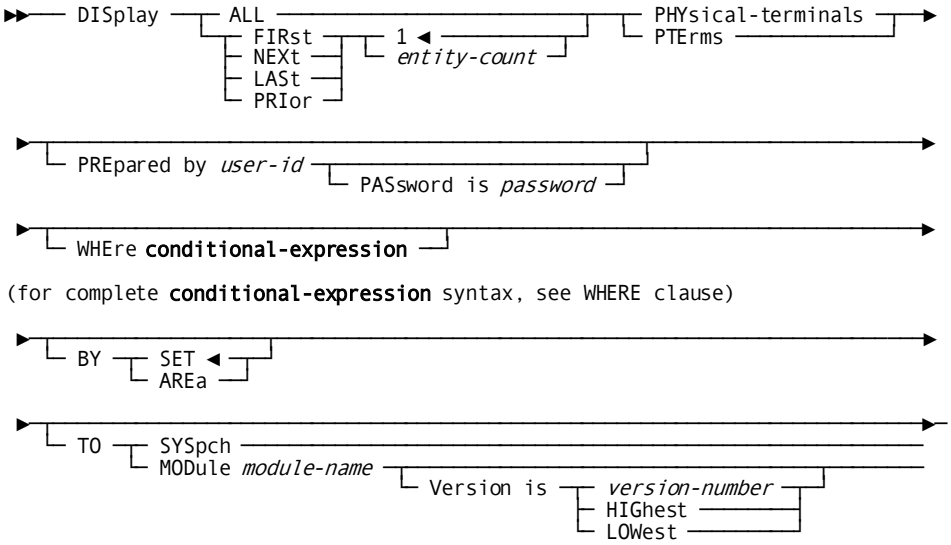


DISPLAY/PUNCH PHYSICAL-TERMINAL (for a single terminal)

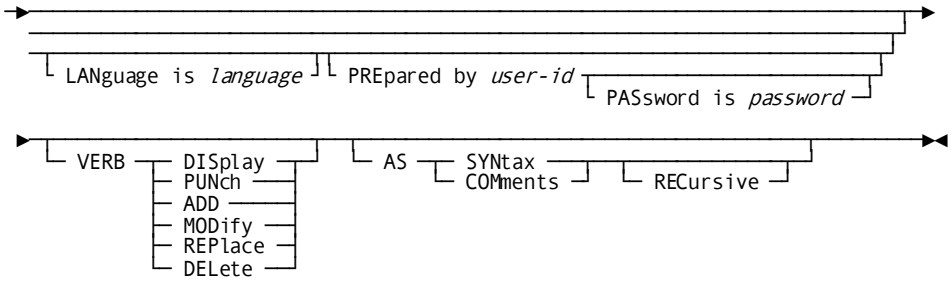




DISPLAY/PUNCH PHYSICAL-TERMINAL (for multiple terminals)



(for complete **conditional-expression** syntax, see WHERE clause)


LANguage is language
PREpared by user-id
PASsword is password
VERB
DISplay
PUNch
ADD
MODify
REPlace
DELete
AS
SYNtax
COMments
RECursive

Parameters

PHYsical-terminal name is *physical-terminal-name*

Identifies a new physical terminal to be established in the dictionary, or an existing physical terminal to be modified, replaced, deleted, displayed, or punched. PTErm is a synonym for physical-terminal. *Physical-terminal-name* must be a 1- through 8-character alphanumeric value.

DEVice TYPe is *physical-device-type*

Specifies the device type of the named physical terminal. The specified device type must be a valid device for the line type defined for the line with which the named physical terminal is associated. Valid values are listed under **Usage**.

MAXimum PERmanent ERRors is 3/*line-error-count*

Specifies the number of retries performed after a terminal I/O error before the teleprocessing monitor will disable the physical terminal. *Line-error-count* must be an integer in the range 0 through 255; the default for ADD is 3.

within SYStem *system-name*

Associates the named physical terminal with the specified system. *System-name* must be the 1- to 32-character name of an existing system. One physical terminal can be associated with multiple systems. The WITHIN SYSTEM specification is documentation only, unless the system generation compiler COPY facility is to be used to copy physical-terminal definitions from an IDD-built system. When the COPY facility is not used, all functional physical-terminal/system relationships are established and maintained by the system generation compiler.

LINe is *line-name*

Associates an existing line with the physical-terminal/system relationship. A physical-terminal/system relationship can be associated with only one line. In the DC/UCF environment, the LINE parameter is documentation. The functional physical-terminal/system relationship is established by means of the DC OPTION clause (described below) or directly through the system generation compiler.

DC option is

Assigns options to the named physical-terminal definition for use with DC/UCF systems.

LINe is *line-name*

Associates a line with the named physical terminal. Note that an explicit version number must be specified; the keywords NEXT HIGHEST and NEXT LOWEST are not valid.

PRInter class is *printer-class*

Assigns a printer class to the physical terminal. *Printer-class* must identify a printer class defined in the LOGICAL-TERMINAL statement and must be an integer in the range 1 through 64. Omit this specification if the physical terminal itself is a printer device such as a 3284.

ENAbled

Automatically enables the terminal at system startup. ENABLED is the default.

DISabled

Disables the terminal until it is enabled explicitly by an operator command during system execution.

WITh/ALSo WITh/WITHOut

Includes or excludes the specified options when the named physical terminal is displayed or punched. Detailed information for each DISPLAY/PUNCH option is under SET OPTIONS Syntax (see page 36). The option that is listed below presents special considerations for this entity type.

DETails

Includes the DESCRIPTION, DC OPTION, DEVICE TYPE, and MAXIMUM PERMANENT ERRORS specifications.

Usage

Valid device and line types

	Device type
ASYNC	CRT/ASR33/2741/RO33
BSC2	R3275S/R3741S/R3780S
BSC3	R3275/R3277/R3278/R3279/R3284 R3286/R3287/R3288/R3289/R3741 R3780
CONSOLE	OPERATOR
DDS	SVC/CTC/BSC/VTAM
INOUTL	INOUTT
L3270B	L3277/L3278/L3279
VTAMLIN	V3277/V3278/V3279/V3284/V3286 V3287/V3288/V3289

	Device type
L3280B	L3284/L3286/L3287/L3288/L3289
SYSOUTL	SYSOUTT
S3270Q	S3277/S3278/S3279
TCAMLIN	TCAMTRM
UCFLINE	UCFTERM
VTAMLU	LU/3600LU/3600PL/3614/LU62

If you specify REPLACE

If you specify REPLACE, the DDDL compiler initializes to defaults and/or excludes the following options:

- DESCRIPTION
- DEVICE TYPE
- MAXIMUM PERMANENT ERRORS
- USER REGISTERED FOR
- COMMENTS/DEFINITION/*comment-key*
- PUBLIC ACCESS
- WITHIN SYSTEM
- DC OPTION
- Related attributes

Physical-terminal/system relationships established by means of the system generation compiler are not affected.

Examples

The following ADD statement defines the physical terminal TM026 within the teleprocessing system INVENTORY; the DEVICE TYPE and LINE clauses further identify the physical terminal as a valid device type within the line A103.

```
add physical-terminal tm026
    physical-terminal description is 'desk 26: assigned dgs'
    device type is l3277
    within system inventory
       line is a103.
```

The following MODIFY statement disassociates the physical terminal from the system INVENTORY in preparation for use by a DC/UCF system; the DC OPTION clause associates the physical terminal with the LINE occurrence.

```
modify physical-terminal tm026
    exclude within system inventory
    dc option is line is a103.
```

PROCESS

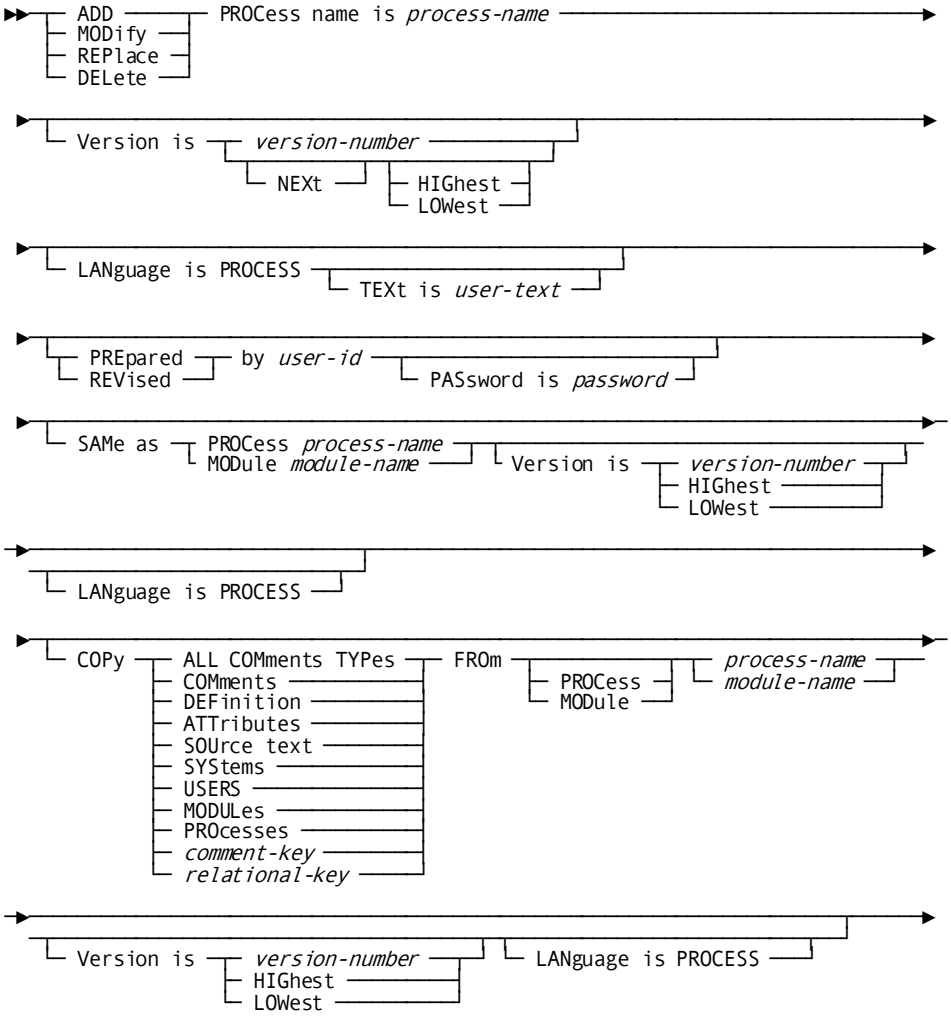
PROCESS statements are used to define source code for CA ADS processes. Optional PROCESS statement clauses:

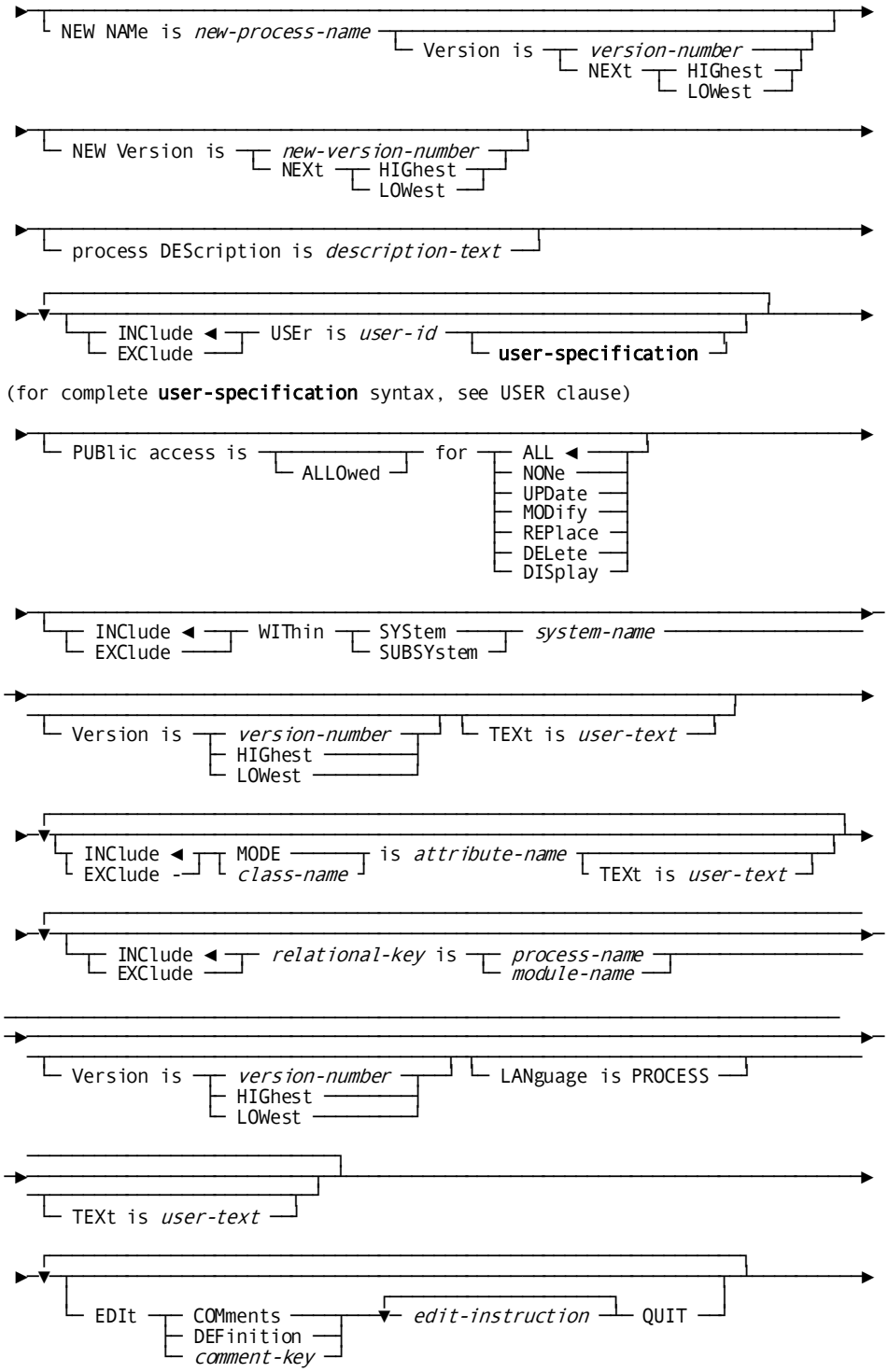
- Relate processes to users, systems, and other processes and modules
- Establish attribute/entity relationships
- Maintain documentation entries

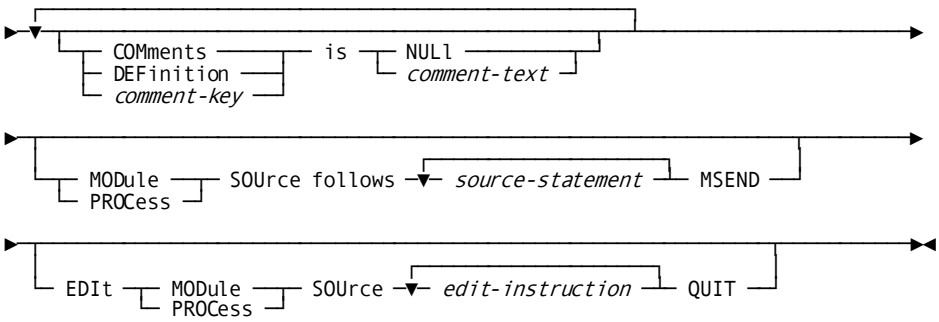
If the SET OPTIONS statement specifies SECURITY FOR IDD IS ON, the user must be assigned the proper authority to issue PROCESS statements.

Syntax

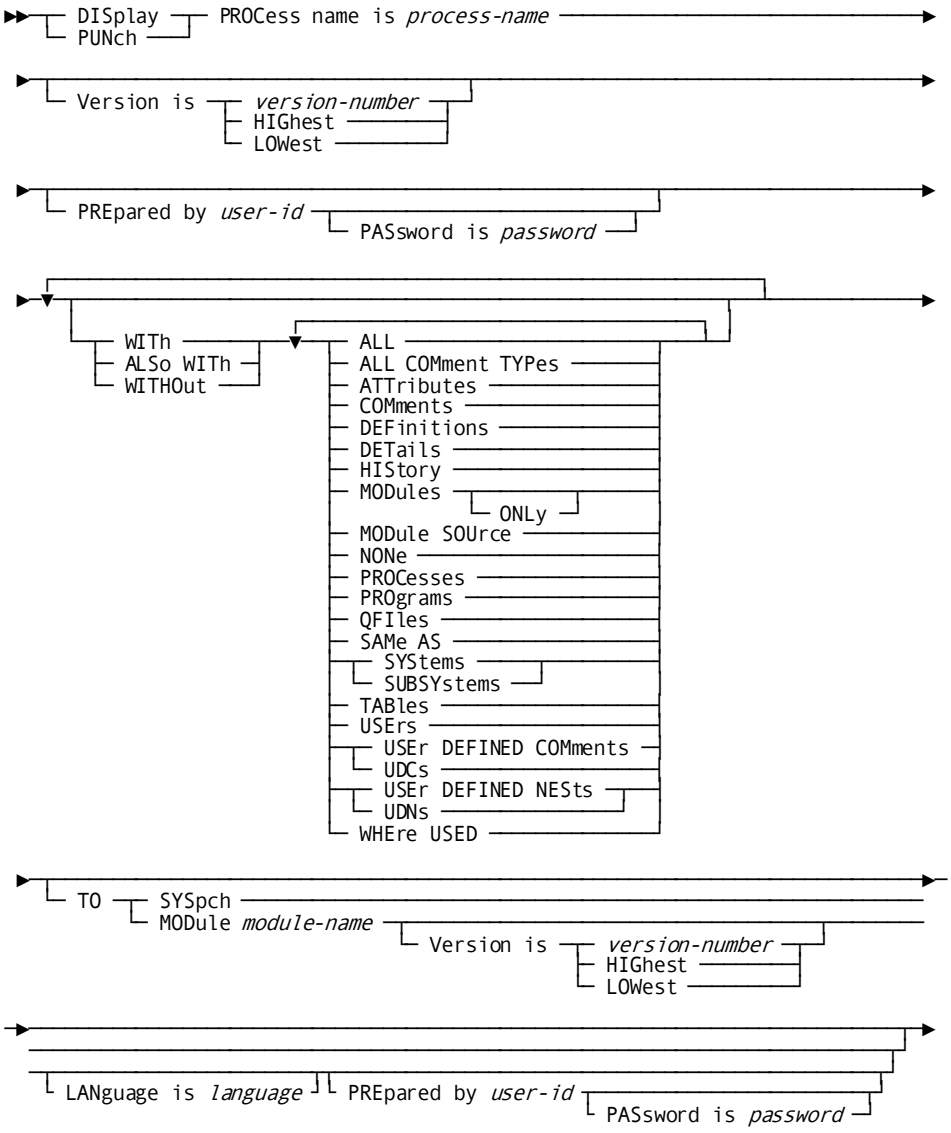
PROCESS Statement

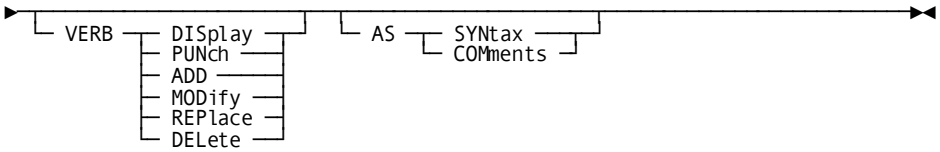



NEW NAMe is new-process-name
Version is
version-number
NEXt
HIGhest
LOWest
NEW Version is
new-version-number
NEXt
HIGhest
LOWest
process DEScription is description-text
INClude
EXClude
USEr is user-id
user-specification
(for complete user-specification syntax, see USER clause)
PUBlic access is
ALLOwed
for
ALL
NONe
UPDate
MODify
REPlace
DELete
DISplay
INClude
EXClude
WIThin
SYStem
SUBSYstem
system-name
Version is
version-number
HIGhest
LOWest
TEXt is user-text
INClude
EXClude
MODE
class-name
is attribute-name
TEXt is user-text
INClude
EXClude
relational-key is
process-name
module-name
Version is
version-number
HIGhest
LOWest
LANguage is PROCESS
TEXt is user-text
EDIt
COMments
DEFinition
comment-key
edit-instruction
QUIT

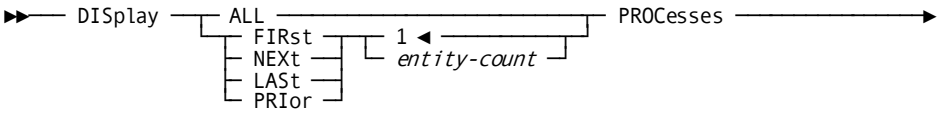


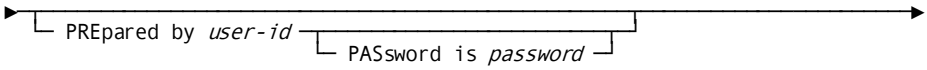
DISPLAY/PUNCH PROCESS Statement (for a single process)

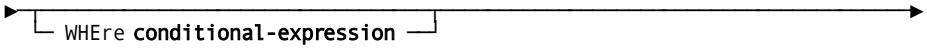




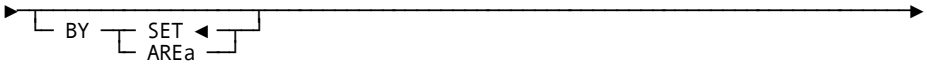
DISPLAY/PUNCH PROCESS statement (for multiple processes)

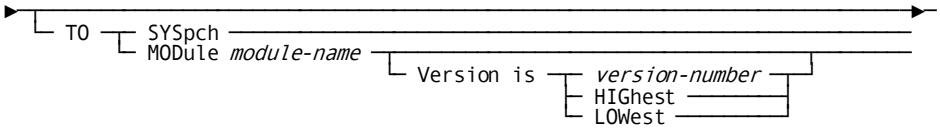


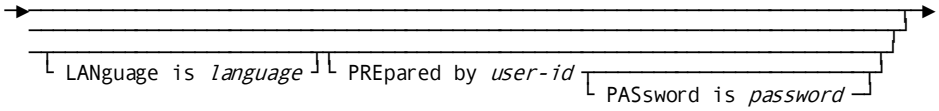


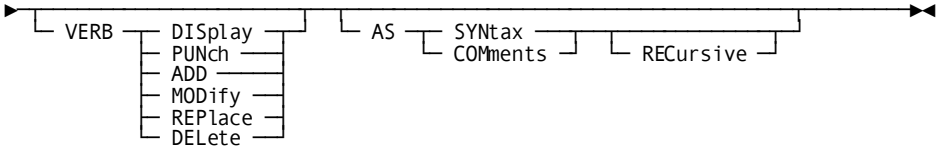


(for complete **conditional-expression** syntax, see WHERE clause)









Parameters

PROCess name is *process-name*

Identifies a new process to be established in the dictionary or an existing occurrence to be modified, replaced, deleted, displayed, or punched. *Process-name* must be a 1- through 32-character alphanumeric value.

LANguage is PROCESS

Documents the named process with a language; if used, the LANGUAGE clause must be coded directly after the name and version number. When the PROCESS statement is specified, the DDDL compiler supplies the appropriate language, PROCESS, automatically.

SAMe as PROCess/MODule *process-name/module-name*

Copies all entries associated with the named process/module except the name and LANGUAGE specifications. The process/module to be copied must have the language PROCESS.

COPy *entity-option* FROm *entity-type-name entity-occurrence-name*

Copies selected options from an entity-occurrence definition and merges the copied options into this definition. PROCESSes can copy only from other modules with a language of PROCESS.

NEW NAMe is *new-process-name*

Specifies a new name for the requested process. This clause changes only the name of the process; it does not alter or delete any previously defined relationships in which the process participates. Subsequent references to the process must specify the new name. *New-process-name* must be a 1- through 32-character value. The combination of new process name, version number, and language must not duplicate that of an established module or process occurrence.

NEW Version is *new-version-number*/NEXt HIGhest/ NEXt LOWest

Specifies a new version number for the named process. The combination of module name, new version number, and language qualification must not duplicate that of an existing module.

within SYStem/SUBSYstem *system-name*

Associates (INCLUDE) the named process with or disassociates (EXCLUDE) it from the specified system or subsystem. *System-name* must reference an existing system or subsystem.

relational-key* is *process-name/module-name

Associates (INCLUDE) the process/module with or disassociates (EXCLUDE) it from another process/module by means of the named relational key. If the modules being related have the same name and version but different languages, or if the related module has a version of HIGHEST or LOWEST and is qualified by language, the LANGUAGE parameter must be specified. See Relational Keys (see page 104) for a complete description of defining and using relational keys.

PROCess/MODule SOUrce follows *source statements* MSEND

Specifies the source code to be associated with the named process or module. Each source statement must be specified in 80-character format. CA ADS process commands coded as process source are compiled by the CA ADS dialog generator when the process is associated with a dialog. INCLUDE requests are executed when the process/module is compiled. PROCESS/MODULE SOURCE FOLLOWS must be coded by itself on the first line; source statements follow on the second and subsequent lines; the keyword MSEND, required to terminate the source statements, must be the first entry on the last line.

WITh/ALSo WITh/WITHOut (DISPLAY/PUNCH only)

Includes or excludes the specified options when the process is displayed or punched. Detailed information for each DISPLAY/PUNCH option is under SET OPTIONS Syntax (see page 36). The option that is listed below presents special considerations for this entity type.

DETails

Includes the DESCRIPTION clause.

Usage

PROCESS statement considerations

The following considerations apply to this statement:

- The reserved words PROCESS and MODULE are interchangeable within PROCESS statement clauses when the MODULE occurrence is qualified with a language of PROCESS, unless otherwise noted.
- Processes are stored as specially-identified module records in the dictionary and are automatically associated with the LANGUAGE class through the PROCESS attribute.

If you specify REPLACE

If you specify REPLACE, the DDDL compiler initializes to defaults and/or excludes the following options:

- DESCRIPTION
- USER REGISTERED FOR
- PUBLIC ACCESS
- WITHIN SYSTEM
- COMMENTS/DEFINITION/*comment-key*
- Related processes and modules
- Related attributes
- PROCESS SOURCE

The following relationships are not affected:

- Processes to which the named process is a related process
- Modules to which the named process is a related process
- Users accessing the named process
- Programs using the named process
- LANGUAGE specification

Example

The following statements add the process GET-A-CUSTOMER to the dictionary and modify the process UPDATE-A-CUSTOMER; the language qualification for GET-A-CUSTOMER is automatically supplied.

```
add process get-a-customer
    module source follows
     ready.
     obtain calc customer.
     if db-rec-not-found
       then do
         display message
         text is 'customer does not exist -- will be added'.
         end.
       else do
         display message
         text is 'customer exists -- will be updated'.
         end.
    msend.

modify process update-a-customer
    module source follows
     ready usage-mode is update.
     obtain calc customer.
     if db-rec-not-found
       then do
         store customer.
         display message
         text 'new customer has been added'.
         end.
       else do
         modify customer.
         display message
         text is 'customer has been updated'.
         end.
    msend.
```

Note: Note that the LANGUAGE class with MANUAL PLURAL qualifiers and the MODE class with AUTOMATIC PLURAL qualifiers are automatically defined during IDD installation. When processing IDMS COPY statements, the DML precompilers inspect entries within entity occurrences that specify the MODE and LANGUAGE classes.

PROGRAM

PROGRAM statements are used to document user application programs and CA ADS Batch transactions.

More information: For more information, refer to the *CA ADS User Guide* for special considerations that apply to defining programs for use in the CA ADS Batch environment.

Optional PROGRAM clauses:

- Relate programs to occurrences of the USER, SYSTEM (SUBSYSTEM), PROGRAM, ENTRY POINT, MODULE, RECORD, and FILE entity types, to subschemas, and to areas, sets, records, and logical records
- Control the participation of programs in attribute/entity relationships
- Maintain documentation entries
- Establish CA IDMS/DC system generation information and CA IDMS/DB database statistics

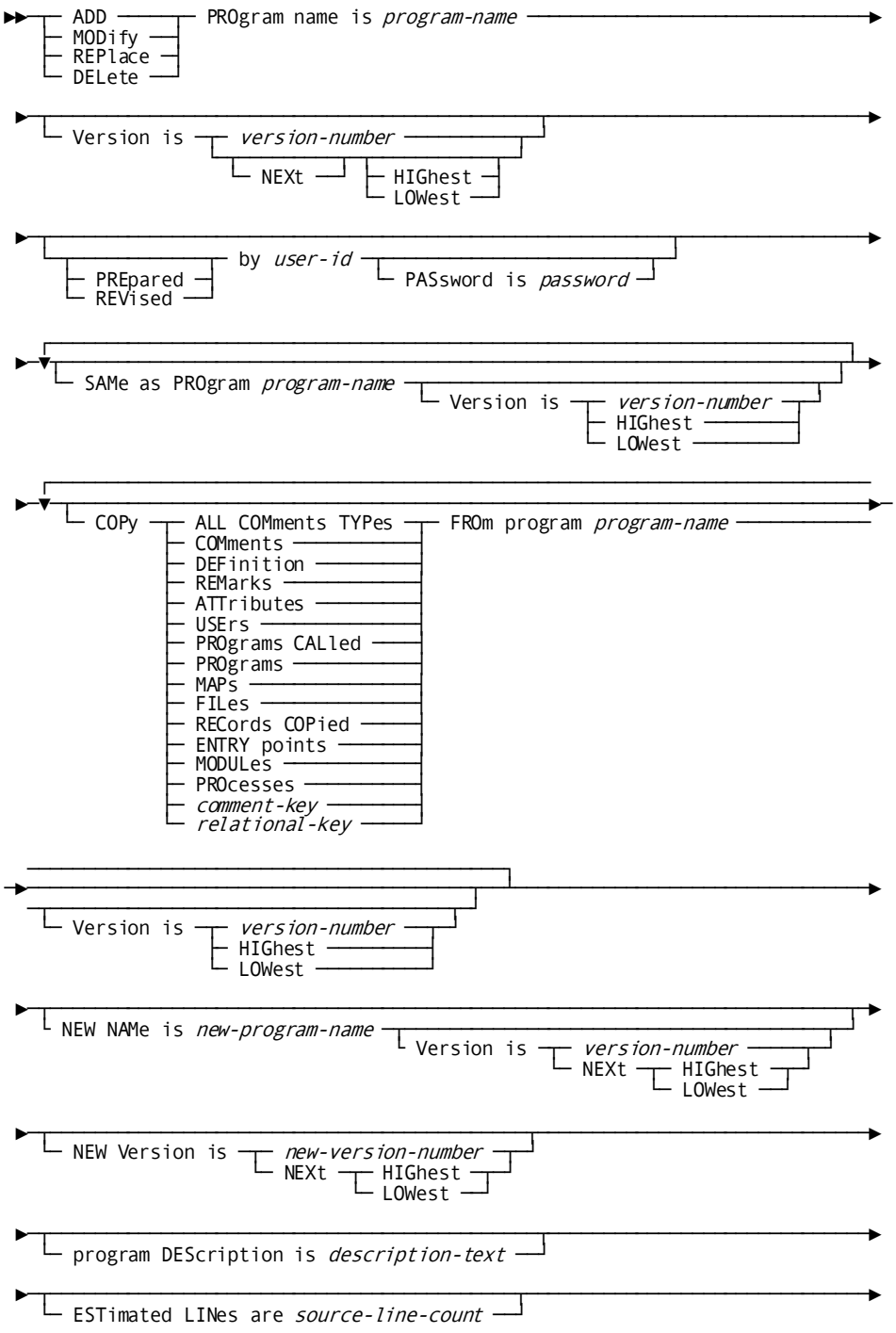
Note: It is recommended that you maintain DC OPTIONS for PROGRAMs using the system generation compiler, *not* the DDDL compiler. If a system generation component is processed by the DDDL compiler, only dictionary security is checked, *not* system generation security. For more information on using the system generation compiler, see *CA IDMS System Generation Guide*.

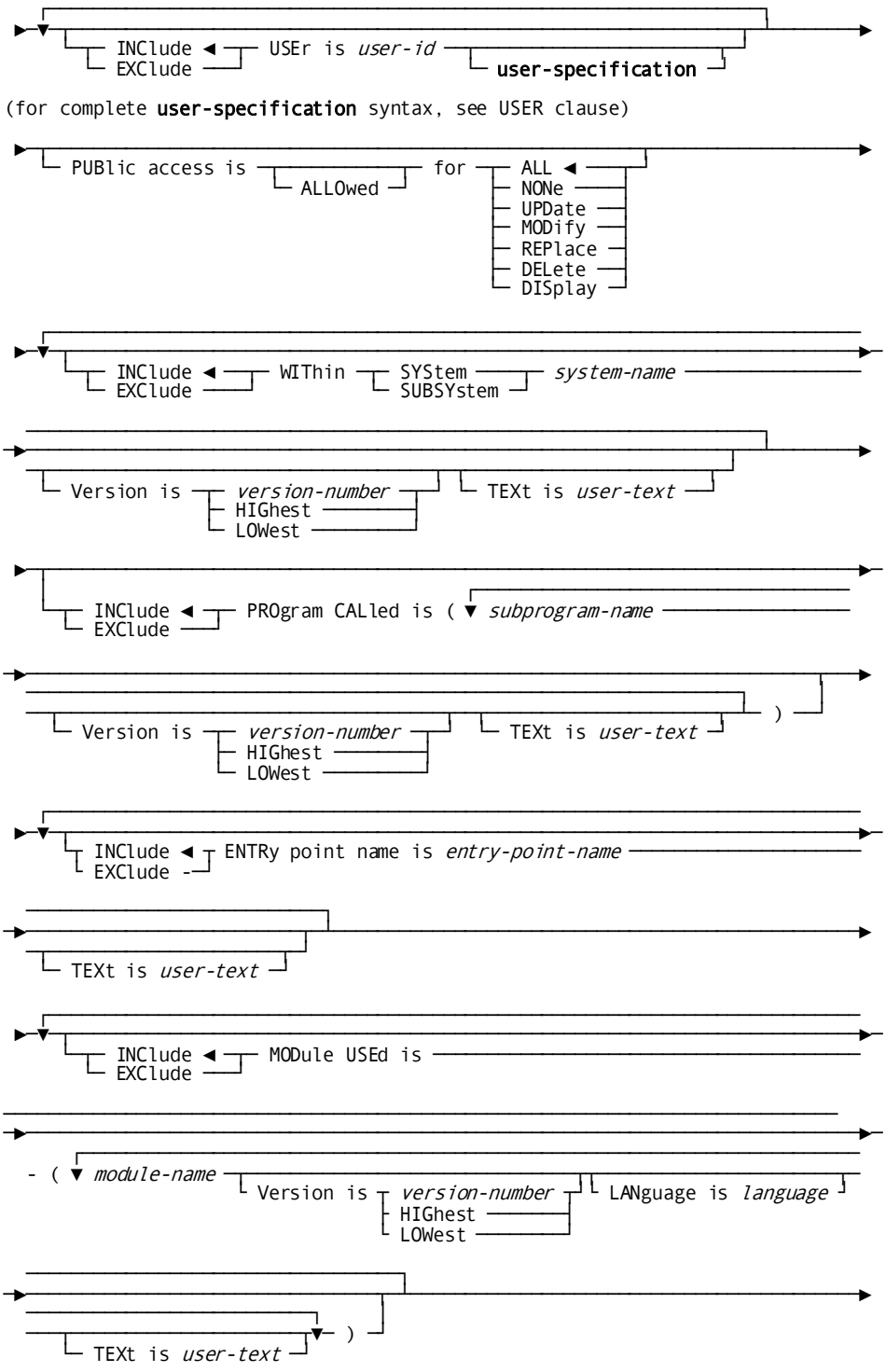
When a DML program requests activity logging, the DML precompiler updates the dictionary. The following program options are established and/or updated:

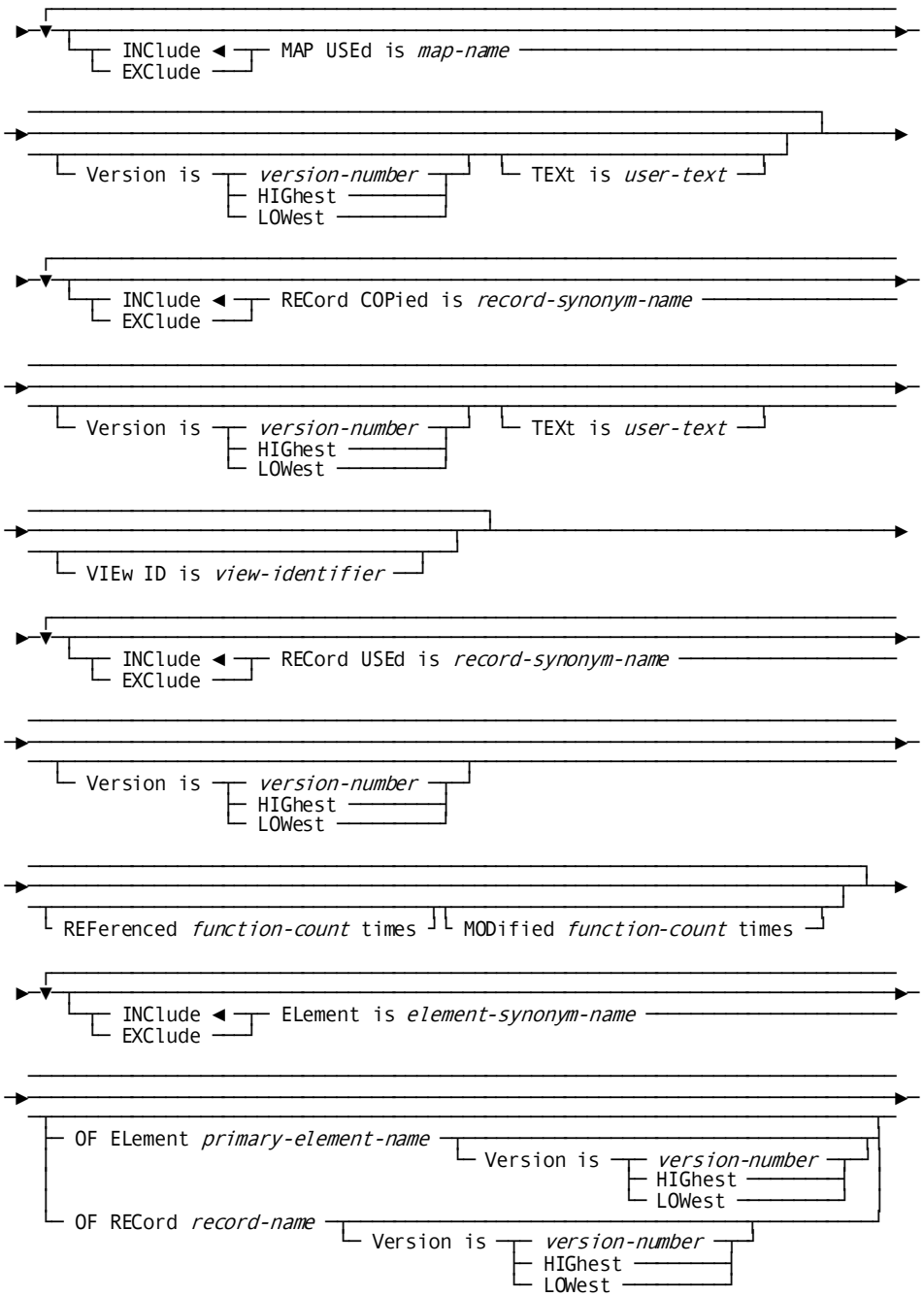
- ESTIMATED LINES
- FILE
- PROGRAM CALLED
- AREA
- ENTRY POINT
- RECORD
- MODULE USED
- SET
- MAP USED
- LOGICAL RECORD
- RECORDS COPIED

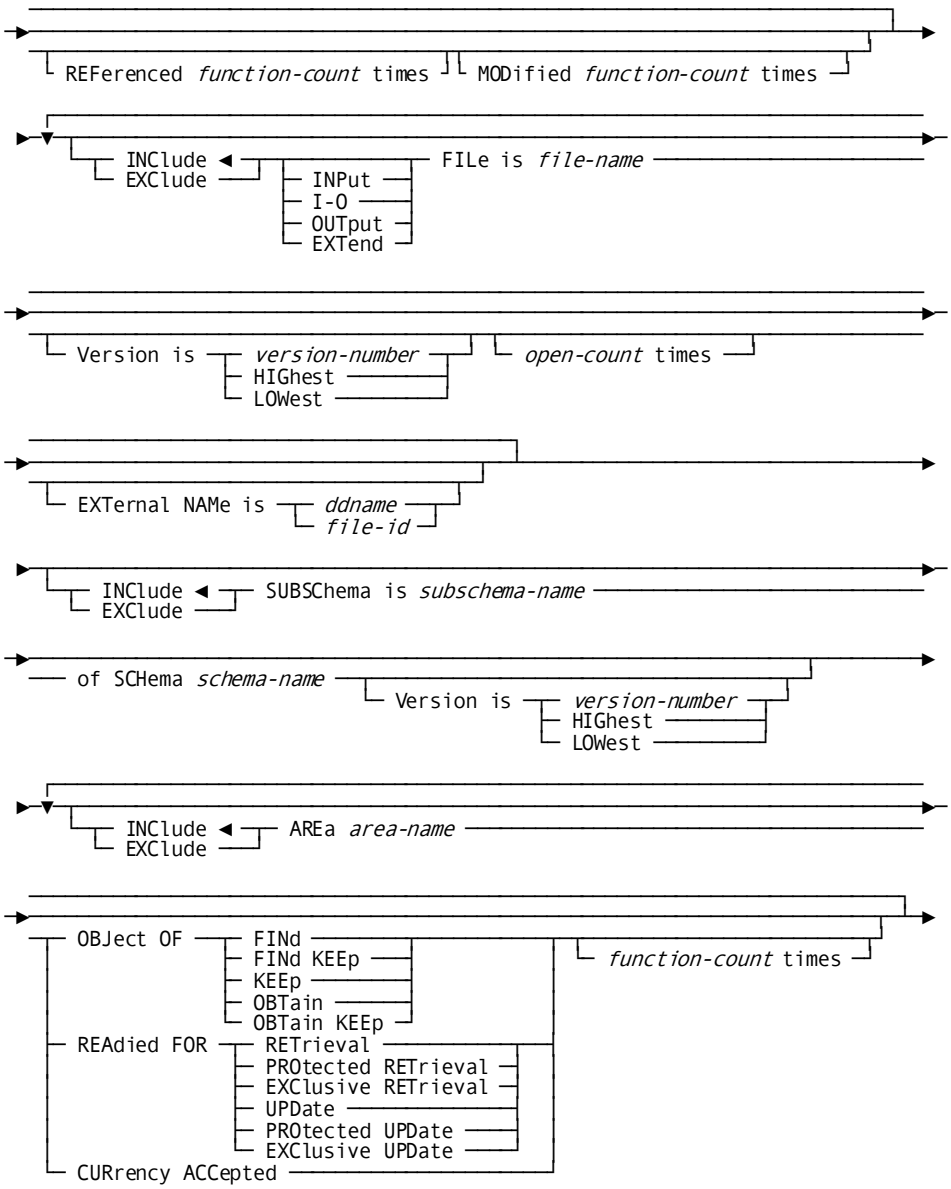
Syntax

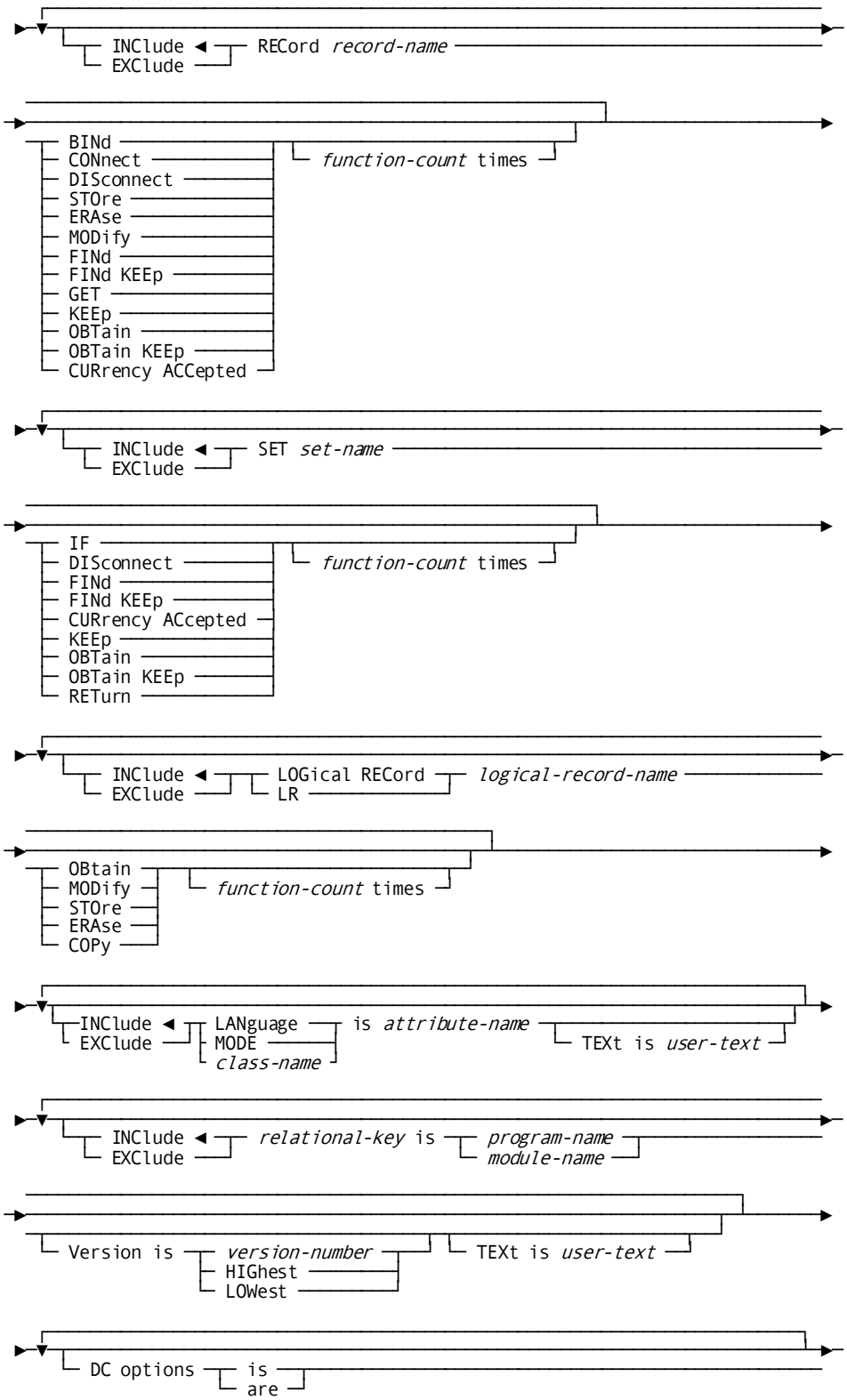
PROGRAM Statement

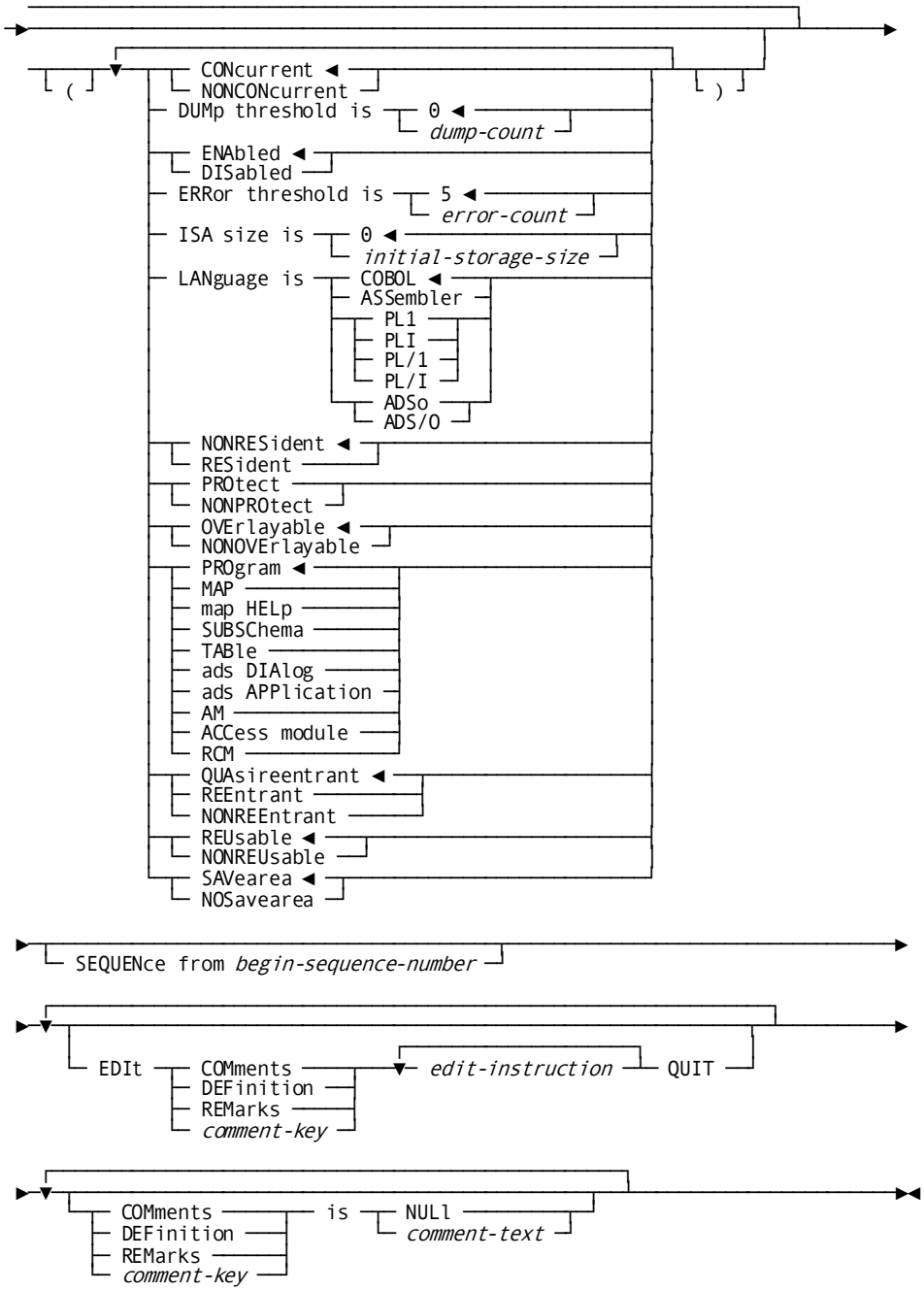

ADD
MODify
REPlace
DELete
PROgram name is program-name
Version is
version-number
NEXt
HIGhest
LOWest
PREpared
REVised
by user-id
PASsword is password
SAMe as PROgram program-name
Version is
version-number
HIGhest
LOWest
COPy
ALL COMments TYPes
FROm program program-name
COMments
DEFinition
REMarks
ATTributes
USErs
PROgrams CALled
PROgrams
MAPs
FILes
RECords COPied
ENTRY points
MODULes
PROcesses
comment-key
relational-key
Version is
version-number
HIGhest
LOWest
NEW NAMe is new-program-name
Version is
version-number
NEXt
HIGhest
LOWest
NEW Version is
new-version-number
NEXt
HIGhest
LOWest
program DEScription is description-text
ESTimated LINes are source-line-count


INClude
EXClude
USEr is user-id
user-specification
(for complete user-specification syntax, see USER clause)
PUBlic access is
ALLOwed
for
ALL
NONe
UPDate
MODify
REPlace
DELete
DISplay
INClude
EXClude
WIThin
SYStem
SUBSYstem
system-name
Version is
version-number
HIGhest
LOWest
TEXt is user-text
INClude
EXClude
PROgram CALled is (▼ subprogram-name
)
Version is
version-number
HIGhest
LOWest
TEXt is user-text
INClude
EXClude
ENTRy point name is entry-point-name
TEXt is user-text
INClude
EXClude
MODule USEd is
- (▼ module-name
Version is
version-number
HIGhest
LOWest
LANguage is language
)
TEXt is user-text

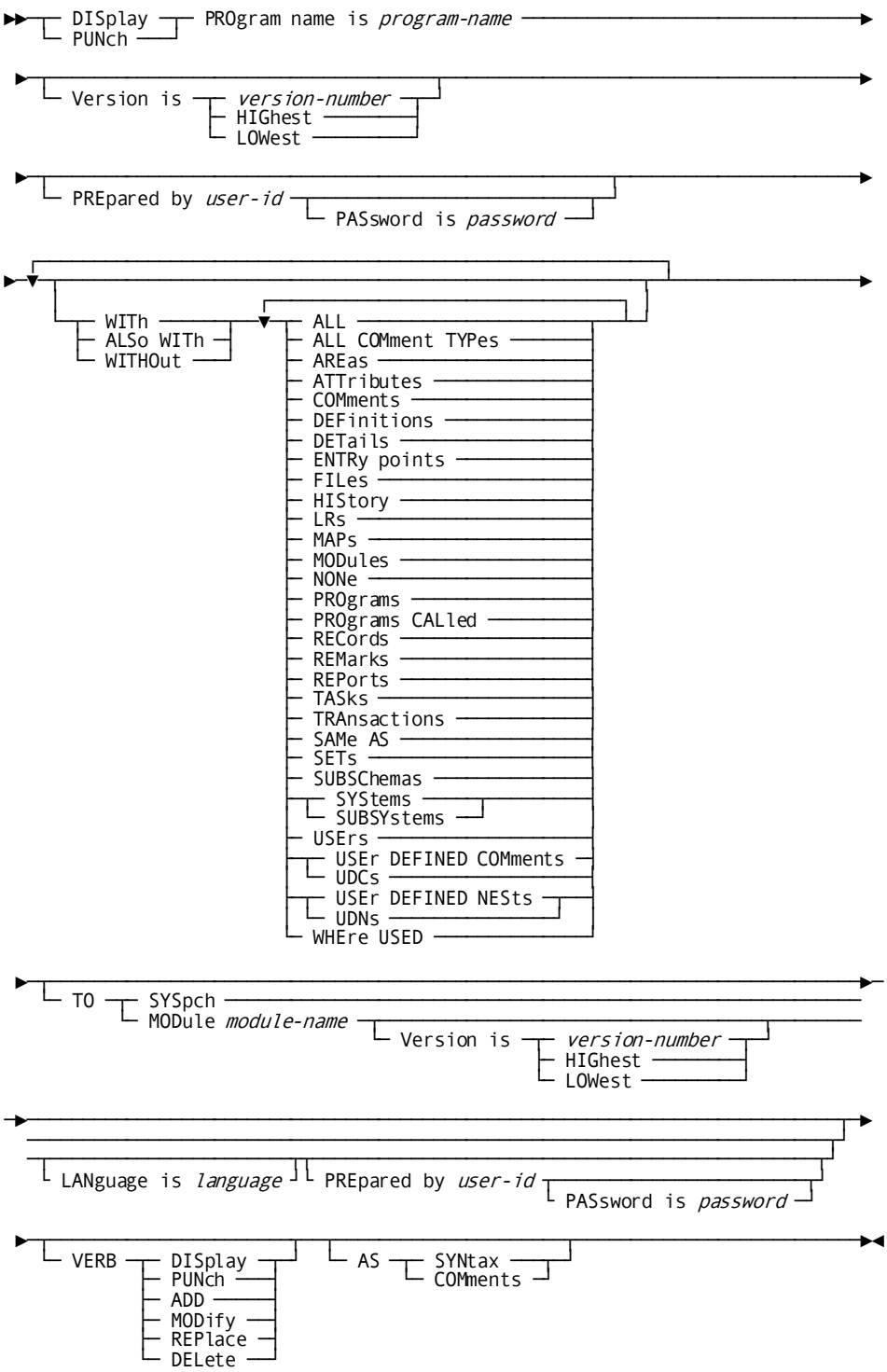

INClude
EXClude
MAP USEd is map-name
Version is
version-number
HIGhest
LOWest
TEXt is user-text
INClude
EXClude
RECord COPied is record-synonym-name
Version is
version-number
HIGhest
LOWest
TEXt is user-text
VIEw ID is view-identifier
INClude
EXClude
RECord USEd is record-synonym-name
Version is
version-number
HIGhest
LOWest
REFerenced function-count times
MODified function-count times
INClude
EXClude
ELement is element-synonym-name
OF ELement primary-element-name
Version is
version-number
HIGhest
LOWest
OF RECord record-name
Version is
version-number
HIGhest
LOWest


REFerenced function-count times
MODified function-count times
INClude
EXClude
INPut
I-O
OUTput
EXTend
FILe is file-name
Version is
version-number
HIGhest
LOWest
open-count times
EXTernal NAMe is
ddname
file-id
INClude
EXClude
SUBSChema is subschema-name
of SCHema schema-name
Version is
version-number
HIGhest
LOWest
INClude
EXClude
AREa area-name
OBJect OF
FINd
FINd KEEp
KEEp
OBTain
OBTain KEEp
function-count times
REAdied FOR
RETrieval
PROtected RETrieval
EXClusive RETrieval
UPDate
PROtected UPDate
EXClusive UPDate
CURrency ACCepted

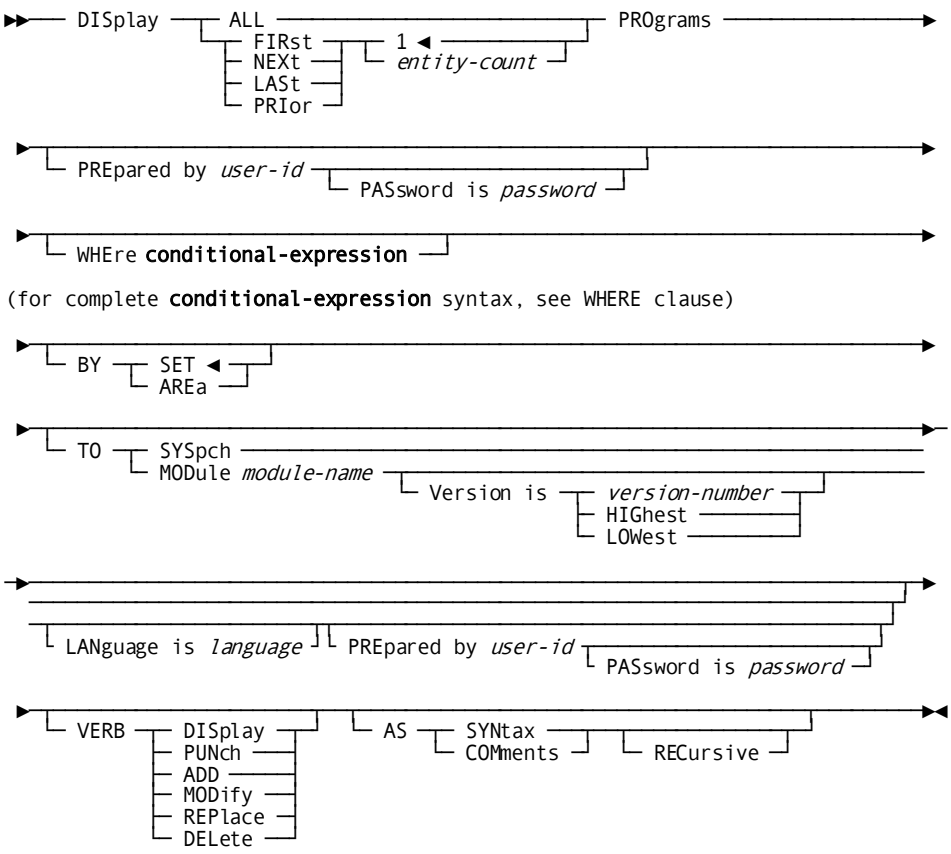

INClude
EXClude
RECord record-name
BINd
CONnect
DISconnect
STOre
ERAse
MODify
FINd
FINd KEEp
GET
KEEp
OBTain
OBTain KEEp
CURrency ACCepted
function-count times
SET set-name
IF
RETurn
LOGical RECord
LR
logical-record-name
OBtain
MODify
STOre
ERAse
COPy
LANguage
MODE
class-name
is attribute-name
TEXt is user-text
relational-key is
program-name
module-name
Version is
version-number
HIGhest
LOWest
DC options
is
are


( )
CONcurrent ◄
NONCONcurrent
DUMp threshold is
0 ◄
dump-count
ENAbled ◄
DISabled
ERROr threshold is
5 ◄
error-count
ISA size is
0 ◄
initial-storage-size
LANguage is
COBOL ◄
ASSembler
PL1
PLI
PL/1
PL/I
ADSo
ADS/O
NONRESident ◄
RESident
PROtect
NONPROtect
OVErlayable ◄
NONOVErlayable
PROgram ◄
MAP
map HELp
SUBSChema
TABle
ads DIAlog
ads APPlication
AM
ACCess module
RCM
QUAsireentrant ◄
REEntrant
NONREEntrant
REUsable ◄
NONREUsable
SAVearea ◄
NOSavearea
SEQUENce from begin-sequence-number
EDIt
COMments
DEFinition
REMarks
comment-key
edit-instruction
QUIT
COMments
DEFinition
REMarks
comment-key
is
NULl
comment-text

DISPLAY/PUNCH PROGRAM Statement (for a single program)



DISPLAY/PUNCH PROGRAM Statement (for multiple programs)



Parameters

PROgram name is *program-name*

Identifies a new program to be established in the dictionary, or an existing program to be modified, replaced, deleted, displayed, or punched. PROGRAM statements cannot modify, replace, or delete programs that participate in DC/UCF systems. *Program-name* must be a 1- through 8-character alphanumeric value.

SAMe as PROgram *program-name*

Copies all entries associated with a previously defined program except the name, WITHIN SYSTEM, SUBSCHEMA, AREA, RECORD, SET, and LOGICAL RECORD specifications.

NEW NAMe is *new-program-name*

Specifies a new name for the named program. This clause changes only the name of the program; it does not alter or delete any relationships in which the program participates. Subsequent references to the program must specify the new name.

New-program-name must be a 1- through 8-character alphanumeric value. The new program name and version number must not duplicate that of an existing program. If a version number is not specified, the version number associated with the original name is used. Note that the NEW NAME clause is not valid if the program participates in a DC/UCF system or if the program was created by the system generation compiler.

NEW Version is *new-version-number*/NEXt HIGhest/NEXt LOWest

Specifies a new version number for the named program. The program name and new version number must not duplicate that of an existing program.

ESTimated LINes are *source-line-count*

Documents the estimated number of source lines in the program. *Source-line-count* must be an integer in the range 1 through 2,147,483,647.

within SYStem/SUBSYstem *system-name*

Associates (INCLUDE) the program with or disassociates (EXCLUDE) it from the specified system or subsystem. *System-name* must be the 1- to 32-character name of an existing system. The WITHIN SYSTEM specification is documentation only, unless the system generation compiler COPY facility is to be used to copy program definitions from an IDD-built system. If the COPY facility is not used, all functional program/system relationships are established and maintained by the system generation compiler.

PROgram CALled is *subprogram-name*

Associates (INCLUDE) a subprogram with or disassociates (EXCLUDE) it from the program. *Subprogram-name* must reference an existing program and can be repeated to specify multiple program/subprogram relationships.

ENTRy point name is *entry-point-name*

Associates (INCLUDE) an entry point with or disassociates (EXCLUDE) it from the program. *Entry-point-name* must be a 1- through 8-character name; a new entry-point occurrence is created whenever *entry-point-name* does not identify an existing entry point in the dictionary.

MODule USEd is *module-name*

Defines (INCLUDE) or disassociates (EXCLUDE) a module to be used or copied by the program. If the named module is not unique in the dictionary, or if it has a version of HIGHEST or LOWEST and is qualified by a language, the optional LANGUAGE IS *language* parameter must be specified; *language* must reference an attribute within the LANGUAGE class.

MAP USEd is *map-name*

Establishes a documentation relationship between the named program and one or more maps. The CA ADS compilers build connections between programs and maps automatically.

RECord COPied is *record-synonym*

Associates (INCLUDE) or disassociates (EXCLUDE) the name of a record or record synonym to be copied by the program. *Record-synonym* must be a 1- to 32-character alphanumeric value that identifies an existing record or record synonym.

VIEw ID is *view-identifier*

Qualifies the supplied name with a view identifier. Record views are established via the VIEW ID substatement, described under RECORD (REPORT/TRANSACTION) (see page 287) later in this chapter.

RECord USEd is *record-synonym*

Documents the program's use of a record or record synonym by specifying the number of times the program references and modifies the record. *Record-synonym* must be a 1- to 32-character alphanumeric value that identifies an existing record or record synonym. If the VERSION IS clause is not specified, the DDDL compiler searches for the correct record-synonym name in the following order:

1. A record name used in a file opened by the program
2. A record copied by the program
3. Any record synonym with the specified name; that is, the most recently added record synonym

REFerenced *function-count* time

Specifies the number of times the named record is referenced by the program.

MODified *function-count* times

Specifies the number of times the named record is modified by the program. *Function-count* must be an integer in the range 0 through 32,767. This clause is produced by the Dictionary Loader when it processes a COBOL program.

ELement is *element-synonym*

Documents the program's use of an element or element synonym by specifying the number of times the element is referenced and modified by the program. *Element-synonym* must be a 1- through 32-character value that references an existing element or element synonym.

OF ELement *primary-element-name*/OF RECord *record-name*

Qualifies the element-synonym name with a primary element name or record name or record synonym name. If neither option is specified, the DDDL compiler searches for the correct element-synonym name in the following order:

1. An element within a record in a file opened by the program
2. An element in any record; that is, the most recently added element synonym
3. Any element with the specified name

REFerenced *function-count* times

Specifies the number of times the named element is referenced by a program. *Function-count* must be an integer in the range 0 through 32,767.

MODIfied *function-count* times

Specifies the number of times the program modifies the element. *Function-count* must be an integer in the range 0 through 32,767.

INPut/I-O/OUTput/EXTend FILe is *file-name*

Documents the program's use of a file or file synonym and optionally specifies whether the program is to open the file for input, input and output, or output.

***open-count* times**

Documents the number of OPEN statements in the program. *Open-count* must be an integer in the range 0 through 32,767.

EXTernal NAMe is *ddname/file-id*

Predefines the 1- through 32-character name by which the file is referenced in JCL statements.

SUBSChema is *subschema-name*

Specifies a subschema to be used by the program. *Subschema-name* must be the 1- to 8-character name of an existing subschema.

of SCHema *schema-name*

Identifies the schema with which the named subschema is associated.

If the subschema definition includes the AUTHORIZATION IS ON option, this clause is required to register the program with the subschema before DML precompilers can precompile the program against the named subschema. However, if the SET OPTIONS statement specifies SECURITY FOR IDMS IS ON, the user must be assigned the proper authority to issue this clause.

AREa *area-name*

Specifies a database area to be accessed by the program and establishes how the program is to use the area. *Area-name* must be the name of an area associated with the schema referenced in the SUBSCHEMA parameter.

OBJect OF

Specifies the number of times that the named area will be the object of an area sweep. One of the following functions must be specified: FIND, FIND KEEP, KEEP, OBTAIN, or OBTAIN KEEP.

READied FOR

Specifies the number of times that the program will ready the named area in the specified usage mode. One of the following usage modes must be specified: RETRIEVAL, PROTECTED RETRIEVAL, EXCLUSIVE RETRIEVAL, UPDATE, PROTECTED UPDATE, or EXCLUSIVE UPDATE.

CURrency ACCepted

Specifies the number of times that the database key of the current record in the named area will be accepted by the DML precompilers

***function-count* times**

Specifies the number of times the named function is performed. *Function-count* must be an integer in the range 0 through 32,767.

RECord *record-name*

Documents the program's use of a database record by specifying the frequency of record use by major DML function. One of the following DML functions must be specified: BIND, CONNECT, DISCONNECT, STORE, ERASE, MODIFY, FIND, FIND KEEP, GET, KEEP, OBTAIN, OBTAIN KEEP, or CURRENCY ACCEPTED. *Record-name* must be a 1- through 16-character value that identifies a record defined in the subschema named in the SUBSCHEMA clause.

***function-count* times**

Specifies the number of times the named function is performed. *Function-count* must be an integer in the range 0 through 32,767.

SET *set-name*

Documents the program's use of a database set by specifying the frequency of set use by major function. One of the following functions must be specified: IF, CONNECT, DISCONNECT, FIND, FIND KEEP, CURRENCY ACCEPTED, KEEP, OBTAIN, OBTAIN KEEP, or RETURN. *Set-name* must be a 1- through 16-character value that identifies a set associated with the subschema named in the SUBSCHEMA clause.

***function-count* times**

Specifies the number of times the function is to be performed. *Function-count* must be an integer in the range 0 through 32,767.

LOGical RECord (LR) *logical-record-name*

Documents a program's use of logical records by specifying the frequency of logical record use by DML function. One of the following DML functions must be specified: OBTAIN, MODIFY, STORE, ERASE, or COPY. *Logical-record-name* must be a 1- through 16-character value that identifies a logical record associated with the subschema named in the SUBSCHEMA clause.

***function-count* times**

Specifies the number of times the function is to be performed; *Function-count* must be an integer in the range 1 through 32,767.

DC options is/are

Assigns one or more DC/UCF options to the named program (DC/UCF programs only).

CONcurrent

Permits more than one task to use the program concurrently. CONCURRENT is the default.

NONCONcurrent

Indicates that only one task at a time can use the program.

DUMp threshold is 0/*dump-count*

Specifies the number of dumps to be taken for program check errors that occur in the program. A memory dump is taken for each program check error, up to and including the specified dump count; additional errors cause the program to terminate abnormally with no memory dump. *Dump-count* must be an integer in the range 0 through 255; the default for ADD is 0.

ENAbled

Automatically enables the program at system startup. ENABLED is the default.

DISabled

Disables the program until it is enabled explicitly by an operator command during system execution.

ERRor threshold is 5/*error-count*

Specifies the number of program check errors that can occur before the program is disabled. The program will continue executing until reaching the specified error threshold; thereafter, the program will not be executed, and tasks attempting to use it will be terminated abnormally. *Error-count* must be an integer in the range 1 through 255; the default for ADD is 5.

ISA size is 0/*initial-storage-size*

Specifies the size in bytes of the initial storage area (ISA) allocated before each execution of the program (ASSEMBLER programs only). *Initial-storage-size* is an integer in the range 0 through 16,777,215; the default for ADD is 0.

LANguage is COBol/ASSembler/PL1/ADSo

Documents the source language of the named program; the default for ADD is COBOL.

Note: This clause does not affect the program's relationship to attributes within the LANGUAGE class.

NONRESident

Specifies that the program is not resident but will be loaded into the program pool as needed. NONRESIDENT is the default.

RESident

Specifies that the program is made resident automatically when the system starts up.

PROtect

Specifies that the storage protection feature is in effect. PROTECT is the default.

NONPROtect

Specifies that the storage protection feature is not in effect.

OVErlayable

Permits the program to be overlaid in the program pool. Specify OVERLAYABLE for executable programs invoked by DC/UCF mechanisms such as LINK and XCTL. OVERLAYABLE is the default.

NONOVErlayable

Prevents the program from being overlaid in the program pool. Specify NONOVERLAYABLE to prevent tables in use from being overwritten in the program pool.

PROgram/MAP/map HELp/SUBSChema/TABle/ads DIAlog/ads APPlication/AM/ACCess module/RCM

Specifies whether the named program is a DC/UCF user program, map, map help, subschema, table, CA ADS dialog, CA ADS application, access module (AM), or relational command module (RCM); the default for ADD is PROGRAM.

REEntrant

Identifies a fully reentrant program that can be executed repeatedly and can be executed before a prior execution has completed. REENTRANT is the default.

QUAsireentrant

Identifies a quasireentrant program that can be executed repeatedly and can be executed before a prior execution has completed. Quasireentrant programs differ from fully reentrant programs in their use of save areas and status information.

NONREEntrant

Identifies a nonreentrant program that can be used by only one DC/UCF task at a time.

REUsable

Identifies the program as reusable. Reusable programs can be executed repeatedly; instructions modified during program execution are returned to their initial state when the program completes execution. Reentrant programs are always reusable, but reusable programs are not necessarily reentrant. REUSABLE is the default.

NONREUsable

Identifies the program as nonreusable. Nonreusable programs modify instructions and do not return them to their initial state after execution. Nonreusable programs must be reloaded each time they are needed.

SAVearea

Acquires a save area automatically before each execution of the program. Specify SAVEAREA if the program uses normal IBM calling conventions and starts by saving registers in a save area. SAVEAREA is the default.

NOSavearea

Does not acquire a save area automatically before each execution of the program.

SEQUENce from *begin-sequence*

Specifies the starting sequence number for an CA ADS Batch transaction if the transaction is to be sequenced. *Begin-sequence* must be an integer in the range 0 through 96,800; zero indicates that no sequence numbers are kept. For further details, refer to the *CA ADS User Guide*.

WITh/ALSo WITh/WITHOut

Includes or excludes the specified options when the named program is displayed or punched. Detailed information for each DISPLAY/PUNCH option is under SET OPTIONS Syntax (see page 36). The options that are listed below present special considerations for this entity type.

DETails

Includes the DESCRIPTION, ESTIMATED LINES, DC OPTION, and SEQUENCE FROM specifications.

PROgrams

Includes the SAME AS and PROGRAMS CALLED specifications and user-defined nests.

Usage

How SET OPTIONS affects PROGRAM statement usage

The SET OPTIONS statement affects PROGRAM statement usage, as follows:

- If the SET OPTIONS statement specifies SECURITY FOR IDD IS ON, the user must be assigned the proper authority to issue PROGRAM statements.
- If the SET OPTIONS statement specifies SECURITY FOR IDMS IS ON, the user must be assigned the proper authority to register a program with a subschema.

- If the SET OPTIONS statement specifies AUTHORIZATION IS ON, DML precompilers will not process a program unless the program has been defined in the dictionary.

If you specify REPLACE

If you specify REPLACE, the DDDL compiler initializes to defaults and/or excludes the following options:

- DESCRIPTION
- FILE IS
- ESTIMATED LINES
- ELEMENT IS
- USER REGISTERED FOR
- SUBSCHEMA IS
- PUBLIC ACCESS
- AREA IS
- WITHIN SYSTEM
- RECORD IS
- PROGRAM CALLED
- SET IS
- ENTRY POINT
- LOGICAL-RECORD IS
- MODULE USED
- DC OPTION
- MAP USED
- SEQUENCE
- RECORD COPIED
- Related programs
- RECORD USED
- Related attributes
- COMMENTS/DEFINITION/*comment-key*

The following relationships are not affected:

- Programs that call the named program
- Systems in which the named program participates
- Tasks that invoke the named program

- Relationships defined by means of the CA IDMS/DC system generation compiler

Example

In the following example, the ADD statement defines the program STCKUPDT, relates the program to the attribute ASSEMBLER within the class LANGUAGE, and supplies comment text using the comment key RECOVERY PROCEDURE. The MODIFY statement adds a DC OPTION clause to the definition of STCKUPDT to associate a language with the program for the purpose of documenting its system generation definition for use by the system generation compiler.

```
add program stckupdt
    program description is 'stock update'
    within system inventory
    language is assembler
    'recovery procedure' is 'restart at step 2'.

modify program stckupdt
    dc option is language is assembler.
```

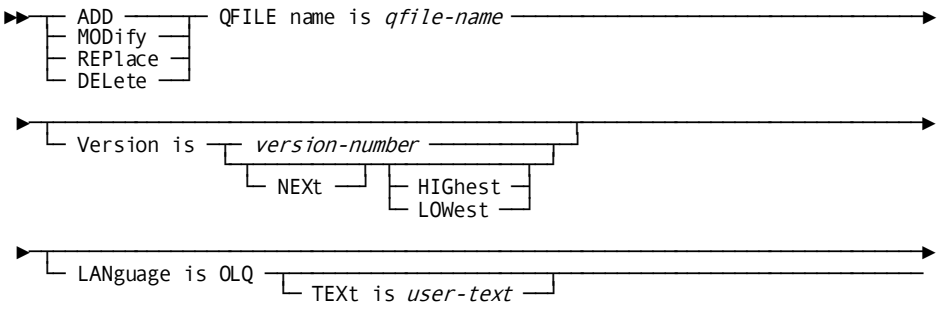
QFILE

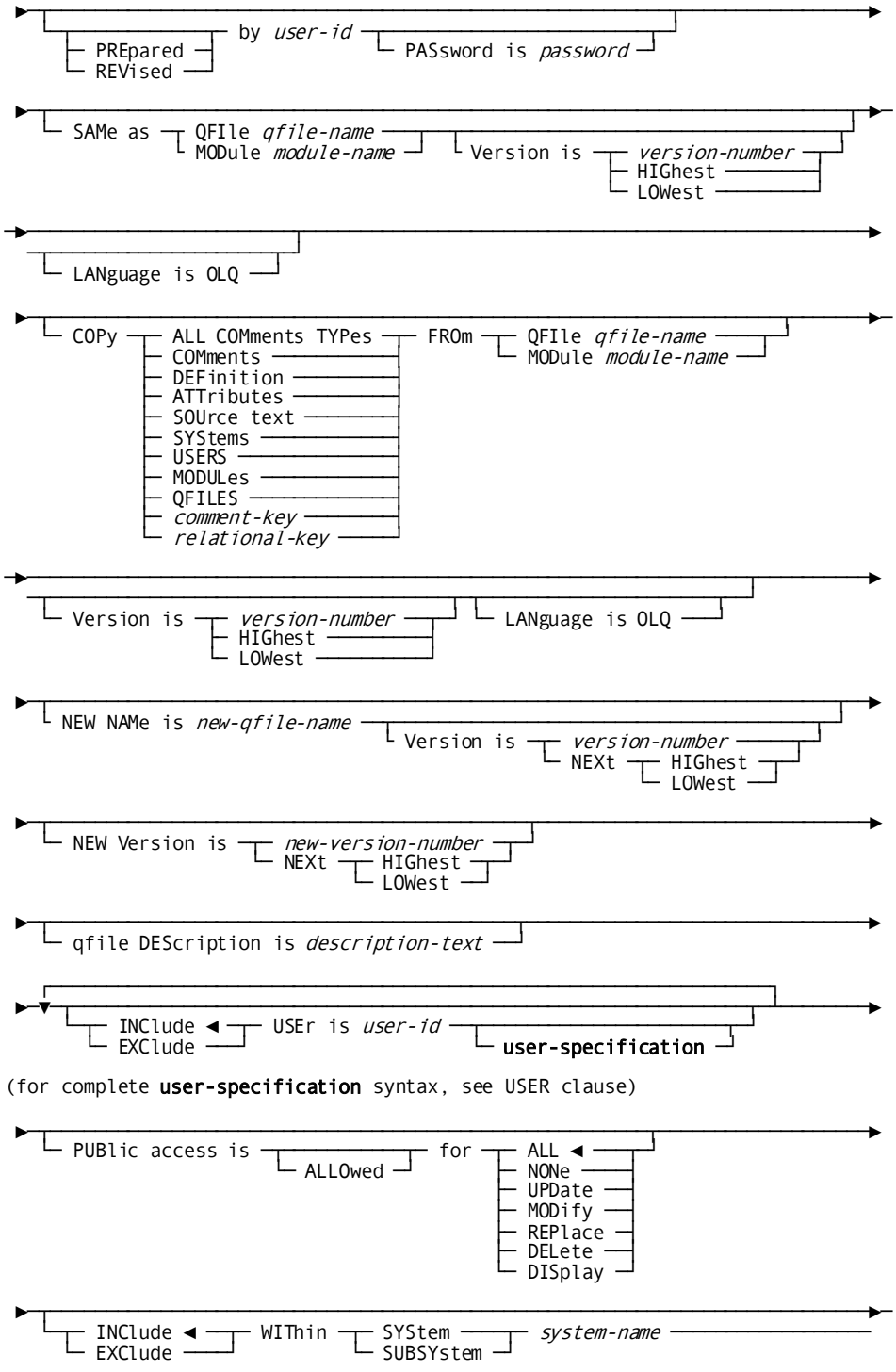
QFILE statements are used to define source code for CA OLQ qfiles. Optional QFILE statement clauses relate qfiles to users, systems, and other qfiles; establish attribute/entity relationships; and maintain documentation entries.

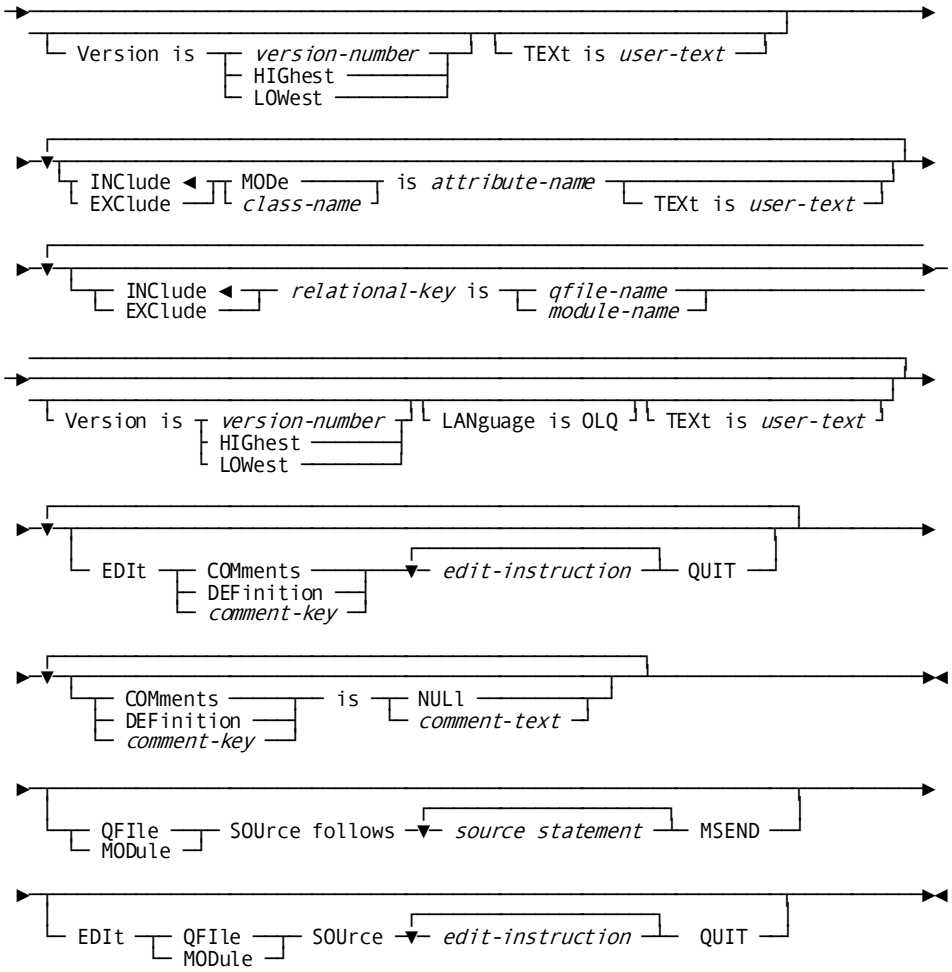
If the SET OPTIONS statement specifies SECURITY FOR IDD IS ON, the user must be assigned the proper authority to issue QFILE statements.

Syntax

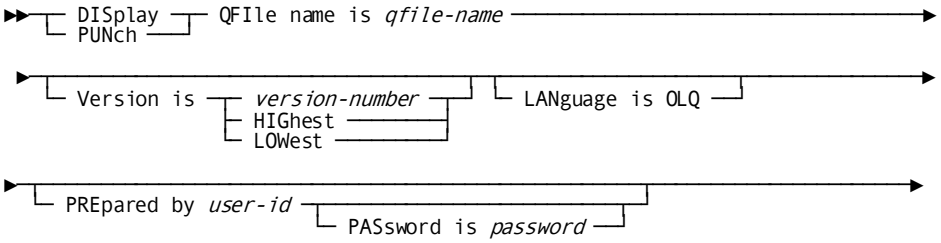
QFILE Statement

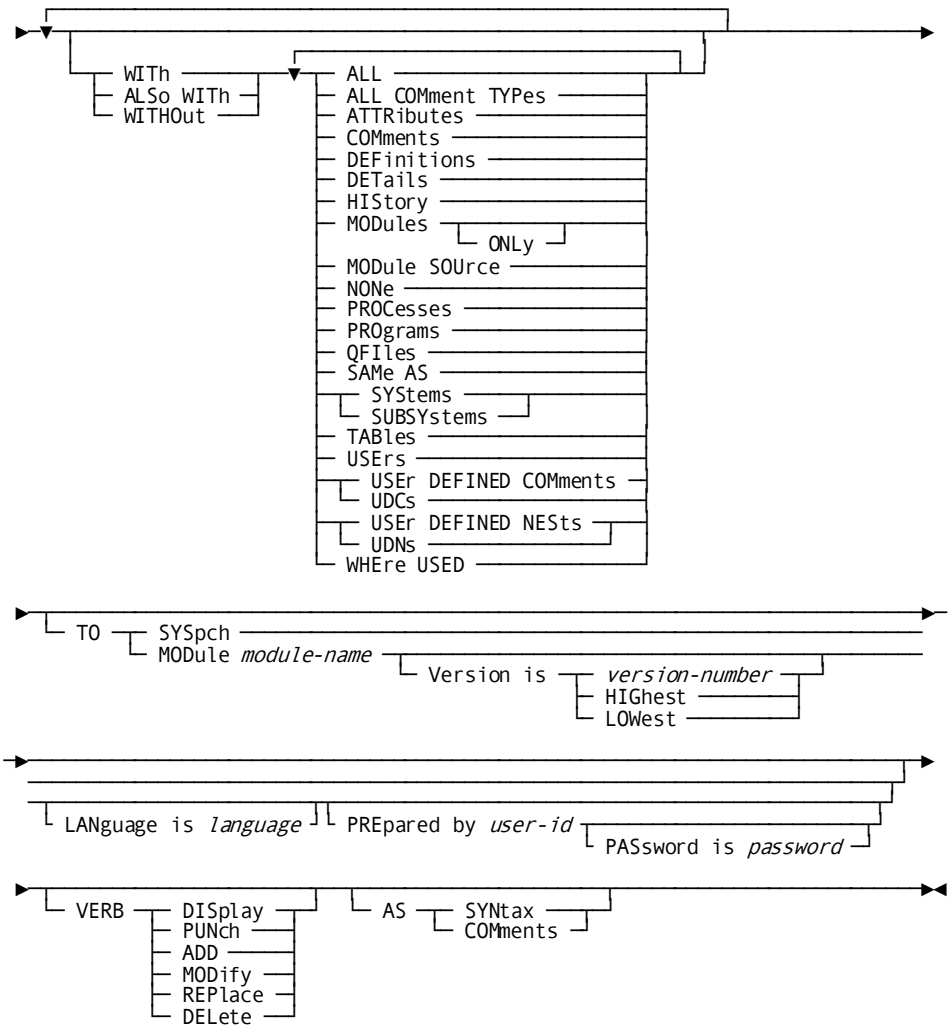



by user-id
PREpared
REVised
PASsword is password
SAMe as
QFIle qfile-name
MODule module-name
Version is
version-number
HIGhest
LOWest
LANguage is OLQ
COPy
ALL COMments TYPes
COMments
DEFinition
ATTributes
SOUrce text
SYStems
USERS
MODULes
QFILES
comment-key
relational-key
FROm
QFIle qfile-name
MODule module-name
Version is
version-number
HIGhest
LOWest
LANguage is OLQ
NEW NAMe is new-qfile-name
Version is
version-number
NEXt
HIGhest
LOWest
NEW Version is
new-version-number
NEXt
HIGhest
LOWest
qfile DEScription is description-text
INClude
EXClude
USEr is user-id
user-specification
(for complete user-specification syntax, see USER clause)
PUBlic access is
ALLOwed
for
ALL
NONe
UPDate
MODify
REPlace
DELete
DISplay
INClude
EXClude
WIThin
SYStem
SUBSYstem
system-name

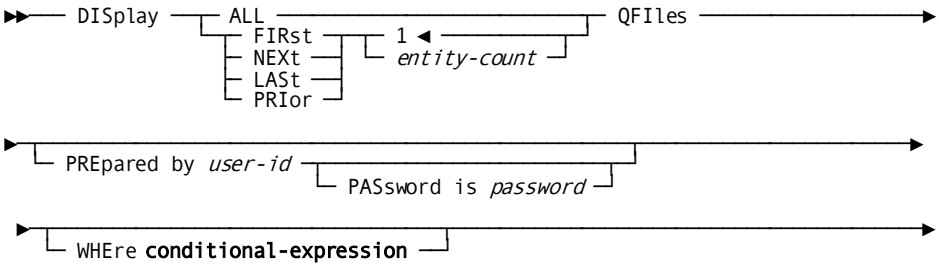


DISPLAY/PUNCH QFILE Statement (for a single qfile)

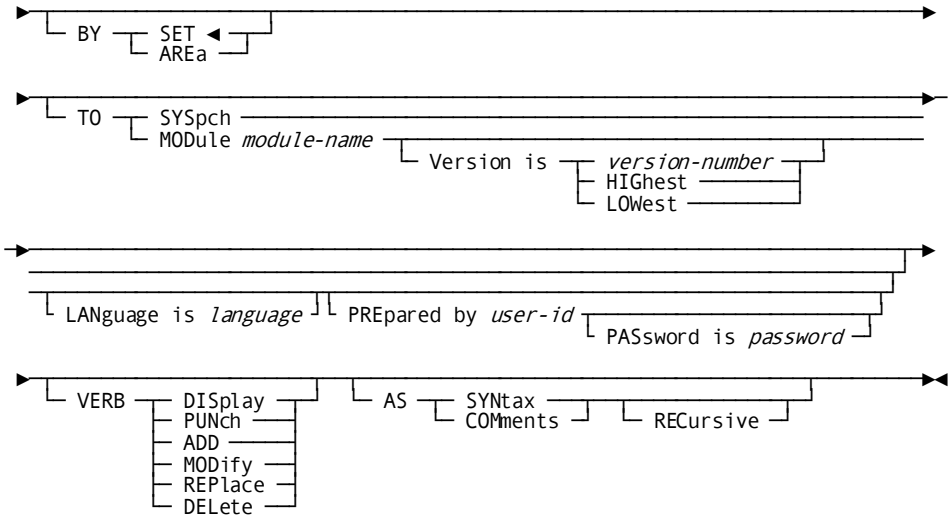




DISPLAY/PUNCH QFILE Statement (for multiple qfiles)



(for complete **conditional-expression** syntax, see WHERE clause)


BY
SET
AREa
TO
SYSpch
MODule module-name
Version is
version-number
HIGhest
LOWest
LANguage is language
PREpared by user-id
PASsword is password
VERB
DISplay
PUNch
ADD
MODify
REPlace
DELete
AS
SYNtax
COMments
RECursive

Parameters

QFILE name is *qfile-name*

Identifies a new qfile to be established in the dictionary, or an existing occurrence to be modified, replaced, deleted, displayed, or punched. *Qfile-name* must be a 1- through 32-character alphanumeric value. The specified name must not duplicate the name of an existing program, map, subschema, or CA ADS dialog.

LANguage is OLQ

Qualifies the requested qfile/module with a language (must be OLQ). When the QFILE statement is specified, the DDDL compiler supplies the appropriate language, OLQ, automatically.

SAMe as QFIle/MODule *qfile-name/module-name*

Copies all entries associated with the named qfile or module except the name and LANGUAGE specifications. The qfile/module to be copied must have the language OLQ.

COPy *entity-option* FROm *entity-type-name entity-occurrence-name*

Copies selected options from an entity-occurrence definition and merges the copied options into this definition. QFILEs can copy only from other modules with a language of OLQ.

TEXt is *user-text*

Associates a 1- to 40-character comment with the new language. If the text includes embedded blanks or delimiters, it must be enclosed in site-specific quote characters.

NEW NAMe is *new-qfile-name*

Specifies a new name for the requested qfile. This clause changes only the name of the qfile; it does not alter or delete any previously defined relationships in which the qfile participates. Subsequent references to the qfile must specify the new name. *New-qfile-name* must be a 1- through 32-character alphanumeric value. The combination of the new qfile name, version number, and language must not duplicate that of an established qfile or module occurrence.

NEW Version is *new-version*/NEXt HIGhest/NEXt LOWest

Specifies a new version number for the named qfile. The combination of the qfile name, new version number, and language qualification must not duplicate that of an existing qfile or module.

WIThin SYStem/SUBSYstem *system-name*

Associates (INCLUDE) the qfile with or disassociates (EXCLUDE) it from the specified system or subsystem. *System-name* must reference an existing system or subsystem.

relational-key* is *qfile-name/module-name

Associates (INCLUDE) the qfile with or disassociates (EXCLUDE) it from another qfile or module by means of the named relational key. If the qfiles and/or modules being related have the same name and version but different languages, or if the related module has a version of HIGHEST or LOWEST and is qualified by language, the LANGUAGE parameter must be specified. For a complete description of the definition and use of relational keys, see Relational Keys (see page 104).

QFIle/MODule SOUrce follows *source-statement* MSEND

Specifies the source code to be associated with the requested qfile. Each source statement must be specified in 80-character format. DML commands coded as module source will be intercepted by the DML precompilers and translated into CALL statements when the module is copied. COPY/INCLUDE requests will also be executed when the module is copied. The QFILE/MODULE/SOURCE FOLLOWS statement must be coded on the first line by itself; source statements follow on the second and subsequent lines; the keyword MSEND, required to terminate the source statements, must be the first entry on the last line.

WITh/ALSo WITh/WITHOut

Includes or excludes the specified options when the qfile is displayed or punched. Detailed information for each DISPLAY/PUNCH option is under SET OPTIONS Syntax (see page 36). The option that is listed below presents special considerations for this entity type.

DETails

Includes the DESCRIPTION clause.

Usage

QFILE statement considerations

The following considerations apply to this statement:

- The reserved words QFILE and MODULE are interchangeable within QFILE statement clauses, unless otherwise noted. In the following discussion, the term *module* applies to processes, qfiles, and tables, unless otherwise noted.
- qfile occurrences are stored as specially identified module records in the dictionary and are automatically associated with the LANGUAGE class through the OLQ attribute.

If you specify REPLACE

If you specify REPLACE, the DDDL compiler initializes to defaults and/or excludes the following options:

- DESCRIPTION
- USER REGISTERED FOR
- PUBLIC ACCESS
- WITHIN SYSTEM
- COMMENTS/DEFINITION/*comment-key*
- Related qfiles
- Related attributes
- QFILE SOURCE

The following relationships are not affected:

- Modules to which the named qfile is a related module
- Users accessing the named qfile
- Programs using the named qfile
- LANGUAGE specification

Example

The following statements add the qfile EMPLOYEES AND OFFICES to the dictionary and modify the qfile CUSTOMER; note that the language qualification for EMPLOYEES AND OFFICES is automatically supplied.

```
add qfile 'employees and offices'
       description is 'olq menu'
       user is tdb
           registered for all
       public access is allowed for all
       qfile source follows
  &path='. '
  signon ss demoss01 schema demoschm(   1)
  options all header echo nofiller full whole interrupt olqheader
  opathstatus nostatistic comments verbose nodbkey picture
  define '&path'. path
  fields for customer are all display nocomma nolead no$
  fields for customer are not cust-zipcode
  get  all   sequential customer
  fields for ordor are all display nocomma nolead no$
  fields for ordor are not ord-date-prom
  fields for ordor are not ord-date-shipped
  find all   ordor in customer-order
  fields for item are all display nocomma nolead no$
  fields for item are not item-quantities
  get  all   item in order-item
  end path
  exec path
  edit cust-number olqheader -
                  'number'
  edit cust-name picture   -
                'x(40)'
       msend.

modify qfile customer
       user is dbc
           registered for update
       public access is allowed for all.
```

QUEUE

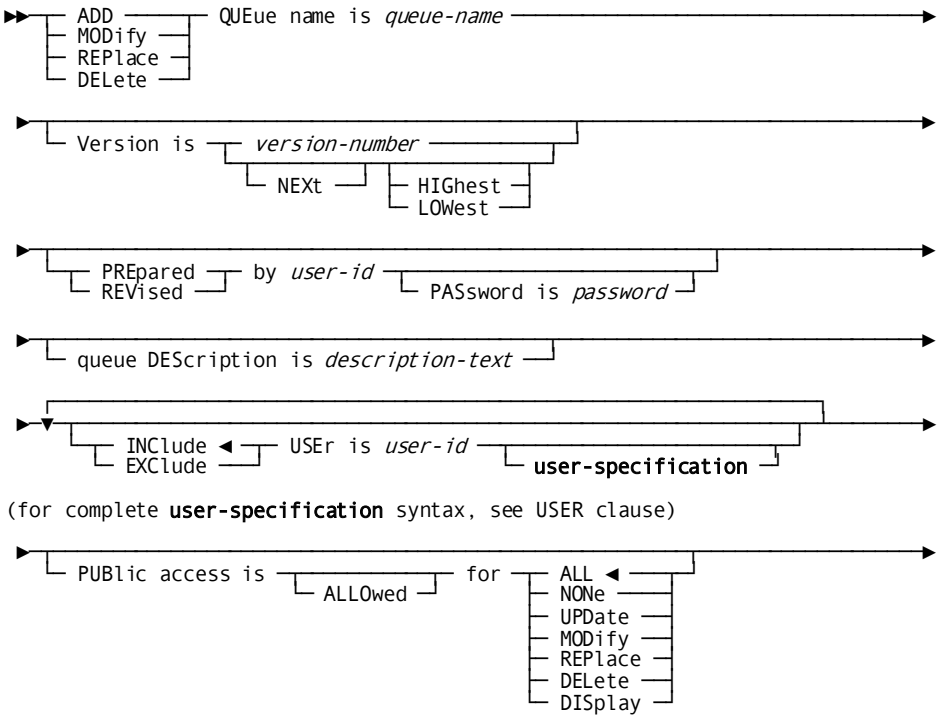
QUEUE statements document the manner in which a teleprocessing system groups similar requests. In the DC/UCF environment, the QUEUE statement can specify the name of a task to be invoked when the queue contains a certain number of entries. When the specified number of entries is reached, the system initiates the required task and processes the queued records.

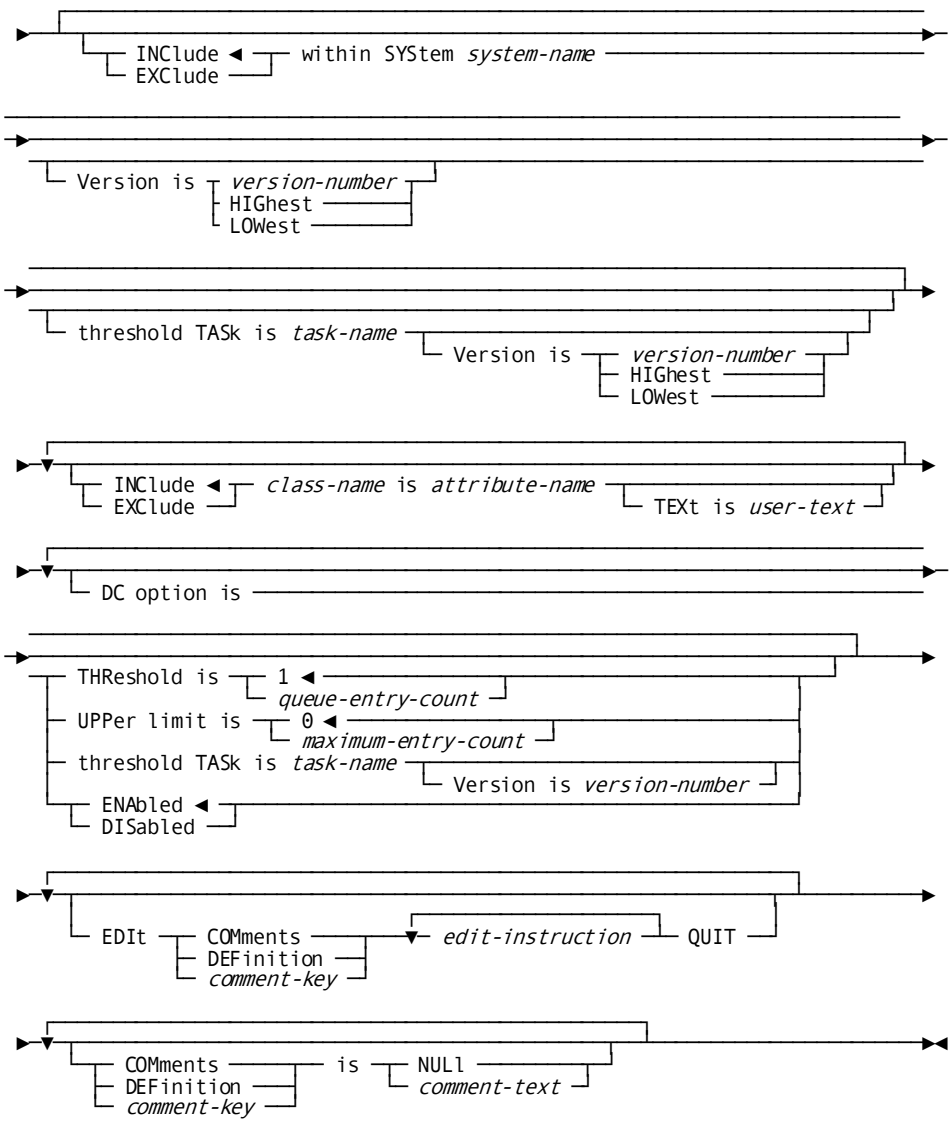
Note: It is recommended that you maintain QUEUE definitions using the system generation compiler, *not* the DDDL compiler. If a system generation component is processed by the DDDL compiler, only dictionary security is checked, *not* system generation security. For more information on using the system generation compiler, refer to *CA IDMS System Generation Guide*.

If the SET OPTIONS statement specifies SECURITY FOR IDMS-DC IS ON, the user must be assigned the proper authority to issue QUEUE statements.

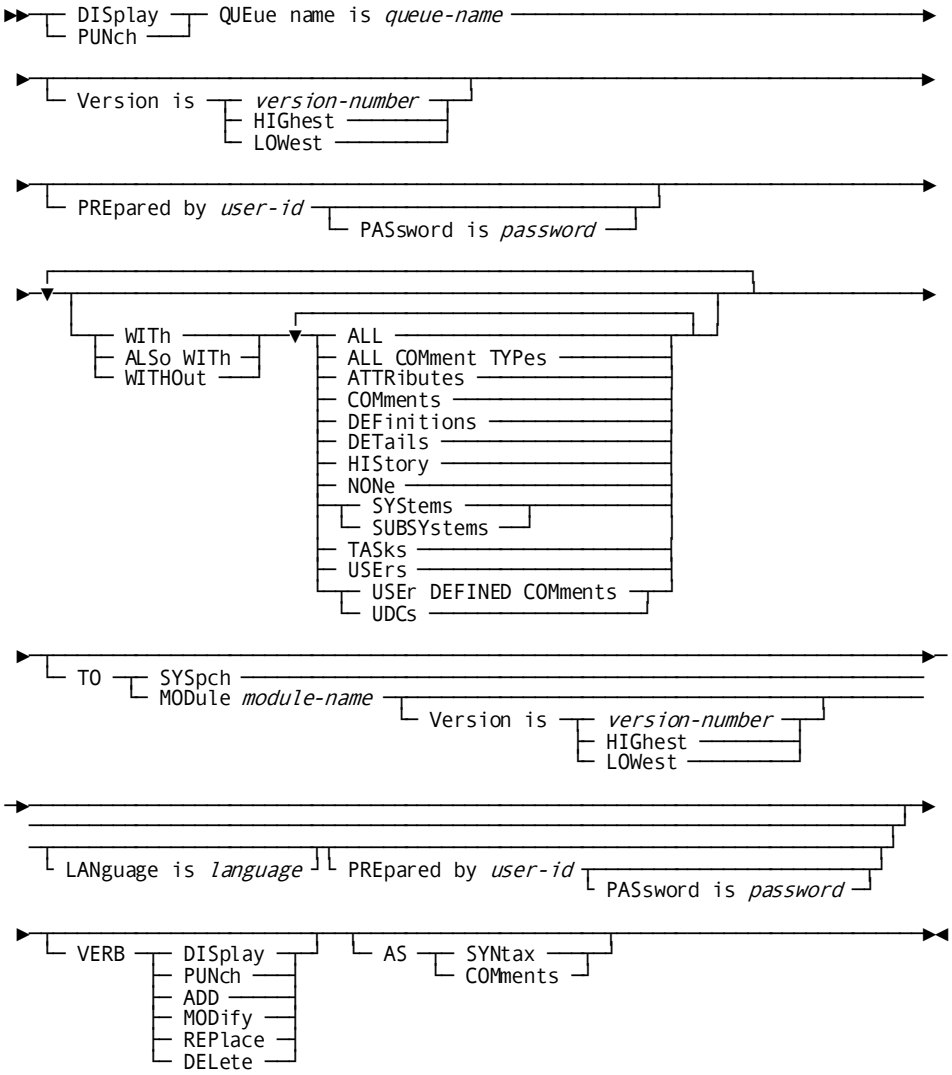
Syntax

QUEUE Statement

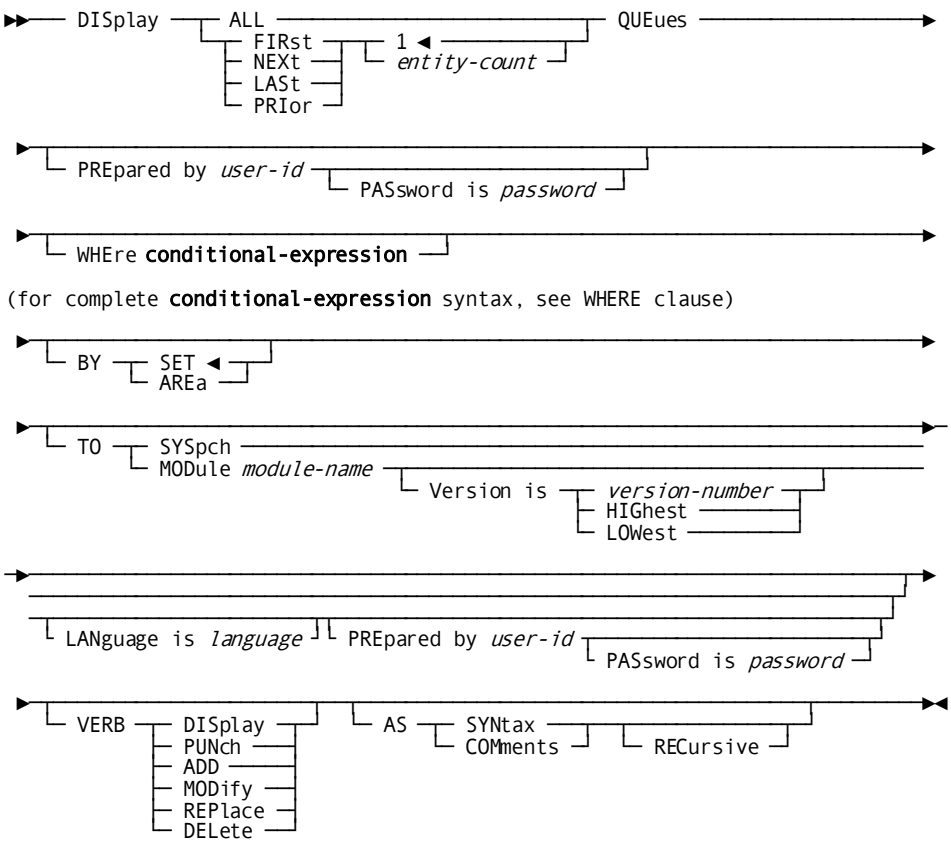



INClude
EXClude
within SYStem system-name
Version is
version-number
HIGhest
LOWest
threshold TASk is task-name
Version is
version-number
HIGhest
LOWest
INClude
EXClude
class-name is attribute-name
TEXt is user-text
DC option is
THReshold is
1
queue-entry-count
UPPer limit is
0
maximum-entry-count
threshold TASk is task-name
Version is version-number
ENAbled
DISabled
EDIt
COMments
DEFinition
comment-key
edit-instruction
QUIT
COMments
DEFinition
comment-key
is
NULl
comment-text

DISPLAY/PUNCH QUEUE Statement (for a single queue)



DISPLAY/PUNCH QUEUE Statement (for multiple queues)



Parameters

QUEue name is *queue-name*

Identifies a new queue to be established in the dictionary, or an existing queue to be modified, replaced, deleted, displayed, or punched. *Queue-name* must be a 1- through 16-character alphanumeric value.

within SYStem *system-name*

Associates (INCLUDE) the queue with or disassociates (EXCLUDE) it from the named system. *System-name* must be a 1- through 32-character value. The WITHIN SYSTEM clause is documentation only, unless the system generation compiler COPY facility is to be used to copy all queues from an IDD-built system. When the COPY facility is not used, functional queue/system relationships are established and maintained by the system generation compiler.

If INCLUDE is specified and the THRESHOLD TASK parameter is omitted, the DDDL compiler establishes a new queue/system relationship. If EXCLUDE is specified and the THRESHOLD TASK parameter is omitted, the DDDL compiler removes the queue/system relationship and any dependent task/queue relationships.

threshold TASk is *task-name*

Associates an established task with the queue/system relationship. *Task-name* must reference a task that is associated with the named system. The teleprocessing monitor invokes the named task when the queue contains a specified number of entries. In DC/UCF environments, this specification is documentation only; use the THRESHOLD TASK parameter in the DC OPTION clause (described below), or define the threshold task directly via the system generation compiler.

DC option is

Documents queue characteristics used by the system generation compiler.

THReshold is 1/*queue-entry-count*

Specifies the number of queue entries required before a task is initiated to process the queue entries. *Queue-entry-count* must be an integer in the range 1 through 32,767; the default for ADD is 1.

UPPer limit is 0/*maximum-entry-count*

Specifies the maximum number of records that can be directed to the queue. This specification is useful in a test environment to prevent looping programs from using all the space in the queue. *Maximum-entry-count* must be an integer in the range 0 through 32,767; the default for ADD is 0. If 0 is specified, no checking is performed.

threshold TASk is *task-name*

Identifies the task to be invoked when the specified queue threshold is reached. *Task-name* must identify an existing task. If specified, the VERSION parameter must identify an explicit number associated with the task; a version of HIGHEST or LOWEST is not acceptable.

ENAbled

Automatically enables the queue at system startup. ENABLED is the default.

DISabled

Automatically disables the queue until it is enabled explicitly by an operator command during system execution.

WITh/ALSo WITh/WITHOut

Includes or excludes the specified options when the named queue is displayed or punched. Detailed information for each DISPLAY/PUNCH option is under SET OPTIONS Syntax (see page 36). The option that is listed below presents special considerations for this entity type.

DETails

Includes the DESCRIPTION and DC OPTION specifications.

Usage

If you specify REPLACE

If you specify REPLACE, the DDDL compiler initializes to defaults and/or excludes the following options:

- DESCRIPTION
- USER REGISTERED FOR
- PUBLIC ACCESS
- COMMENTS/DEFINITION/*comment-key*
- WITHIN SYSTEM
- DC OPTION
- Related attributes

System/queue relationships established by the system generation compiler are not affected.

Example

In the following example, the ADD statement defines the queue REG-IN within the system REGIST and names RUPDT as the threshold task. The MODIFY statement removes REG-IN from the REGIST system in preparation for use by the DC/UCF system.

```
add queue reg-in
    description is 'registration input'
    within system regist
    threshold task is rupdt.

modify queue reg-in
    exclude within system regist
    dc option threshold task is rupdt
    dc option threshold is 3
    dc option upper limit is 9.
```

RECORD (REPORT/TRANSACTION)

Typically, record occurrences consist of groups of elements within a hierarchical structure required by a program or schema; however, records can also exist without elements, usually for documentation or planning purposes. When the user includes an element within a record, the DDDL compiler creates a *record element* and associates it with the named record. A record can have a maximum length of 32,767 characters.

RECORD statements establish and maintain record occurrences but do not directly relate records to elements; the RECORD ELEMENT and COBOL substatements that follow ADD RECORD or MODIFY RECORD statements establish and maintain record-element structures. These substatements are used as follows:

- **RECORD ELEMENT substatement**

  Identifies existing group and elementary elements and defines filler fields for use in the requested record. The DDDL compiler assigns a level number to each element and filler based on the SET OPTIONS statement LEVEL NUMBERS specification. Optional clauses supply record-specific element synonyms, OLQ and CULPRIT column headers, and record-specific editing, value, index, and multiply-occurring element specifications for each record element.

- **COBOL substatement**

  Identifies new or existing elements in a format specific to COBOL language programs. Optional clauses support record synonyms, record-specific element synonyms, comments, and COBOL 74 options; allow the user to define the element's level number, picture, value, and usage; and supply REDEFINES, INDEXED BY, and OCCURS specifications.

COBOL substatements can be followed by RECORD ELEMENT substatements to modify an existing record-element structure. Note, however, that if a COBOL substatement follows a RECORD ELEMENT substatement, the DDDL compiler creates a new record-element structure that replaces the structure associated with the RECORD ELEMENT substatement.

Additional substatements allow the user to rebuild and remove record elements and define restricted record-element structures (called *views*) for use within subschemas and files.

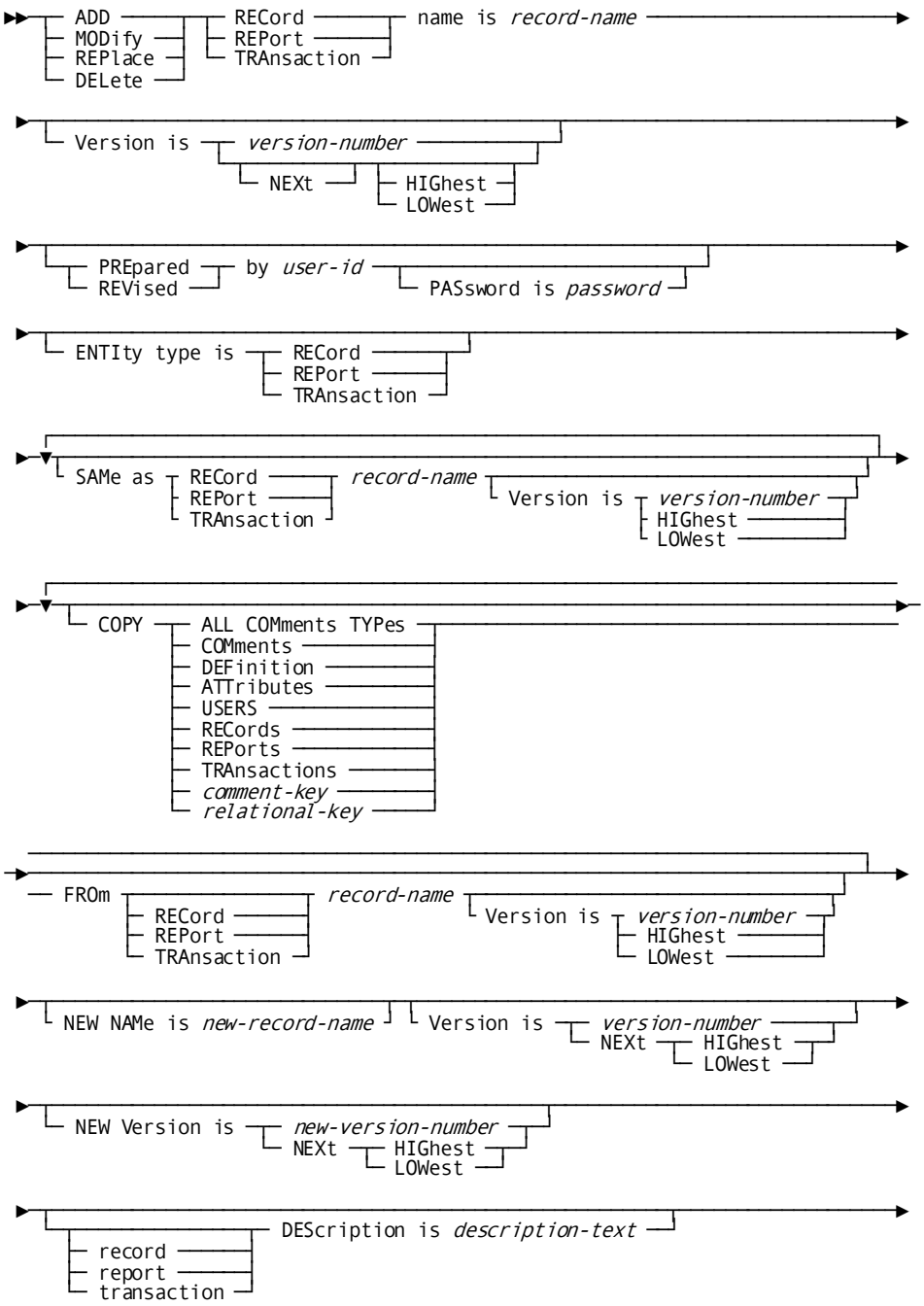
Optional RECORD statement clauses relate records to existing files, users, and other records. (Record occurrences can be related to programs by means of the RECORD COPIED clause of the PROGRAM statement and the DML precompilers.) The RECORD statement also supports comments, attribute/entity relationships, and record synonyms.

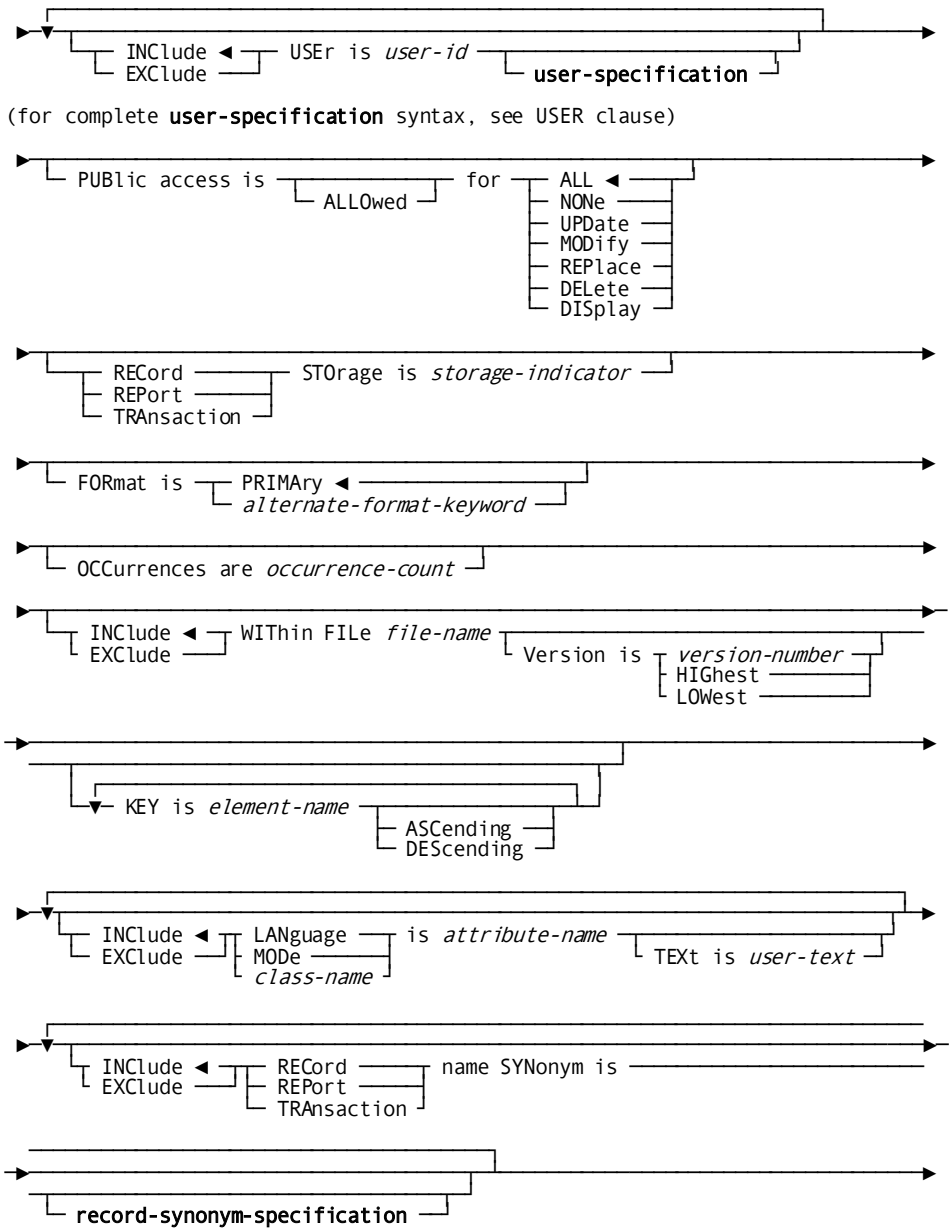
Note: If the keyword REPORT or TRANSACTION is used in place of RECORD, the DDDL compiler creates a special entity occurrence to document the report or transaction in the dictionary. These reports and transactions appear as distinct entity types on dictionary reports.

RECORD Statement

Syntax

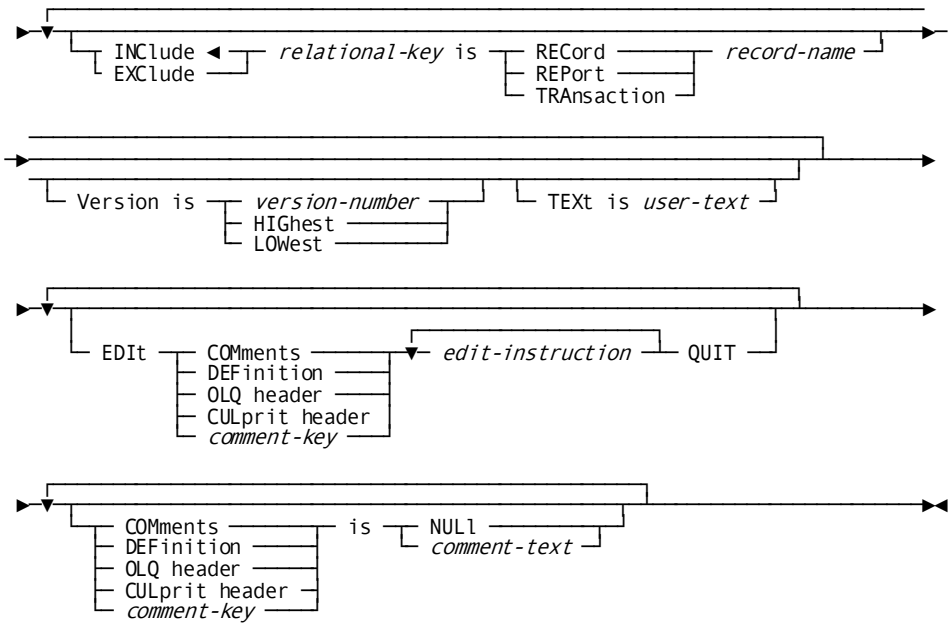
RECORD (REPORT) (TRANSACTION) statement



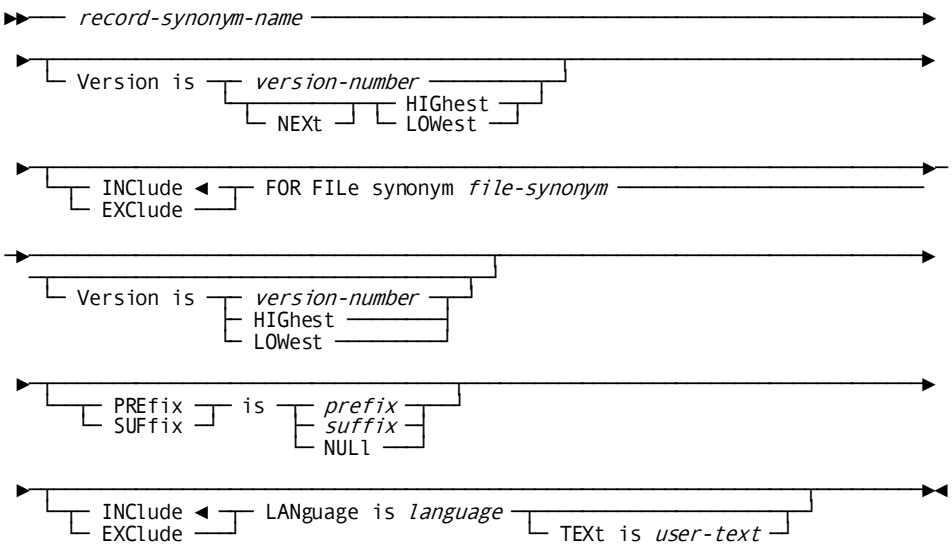


(for complete **user-specification** syntax, see USER clause)

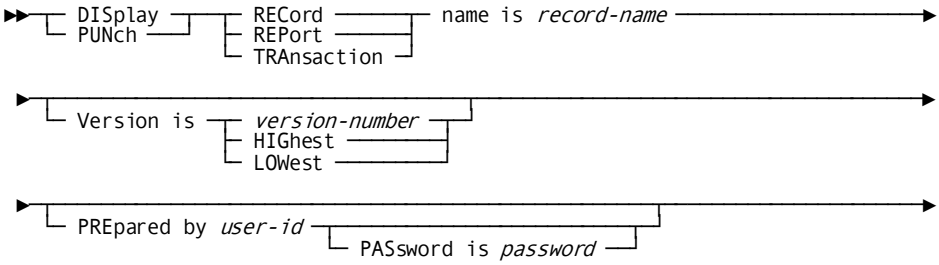
(expanded **record-synonym-specification** syntax follows this syntax diagram)

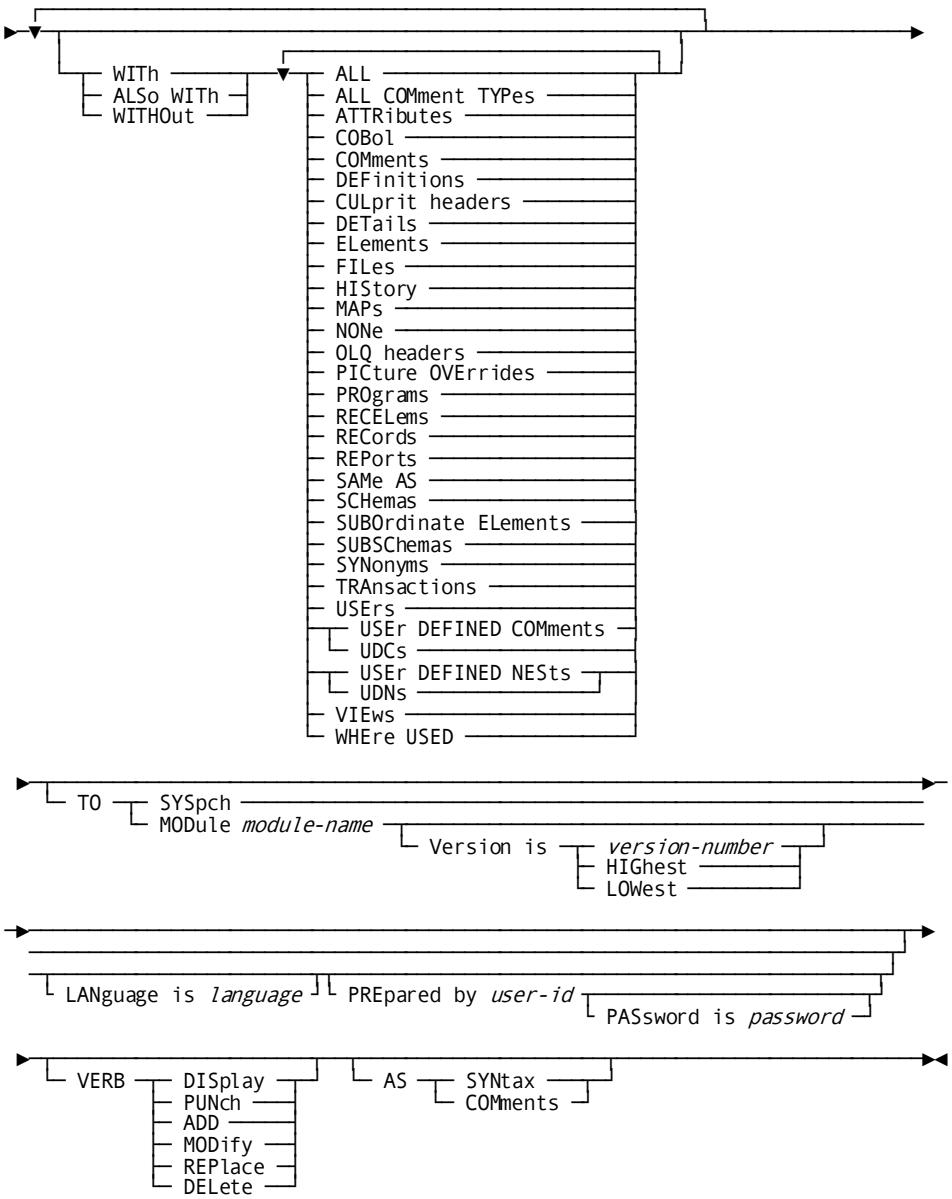


Expansion of *record-synonym-specification*

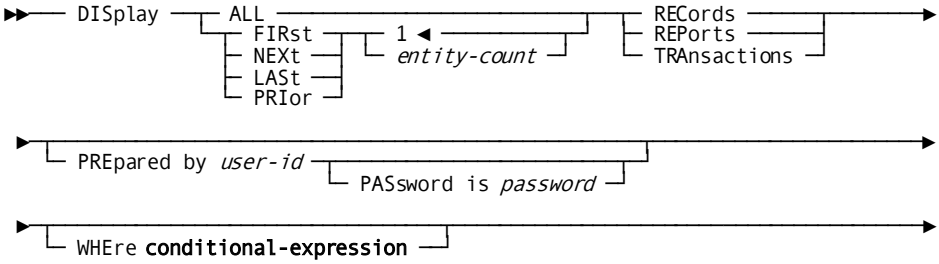


DISPLAY/PUNCH RECORD (REPORT) (TRANSACTION) statement (for a single record)

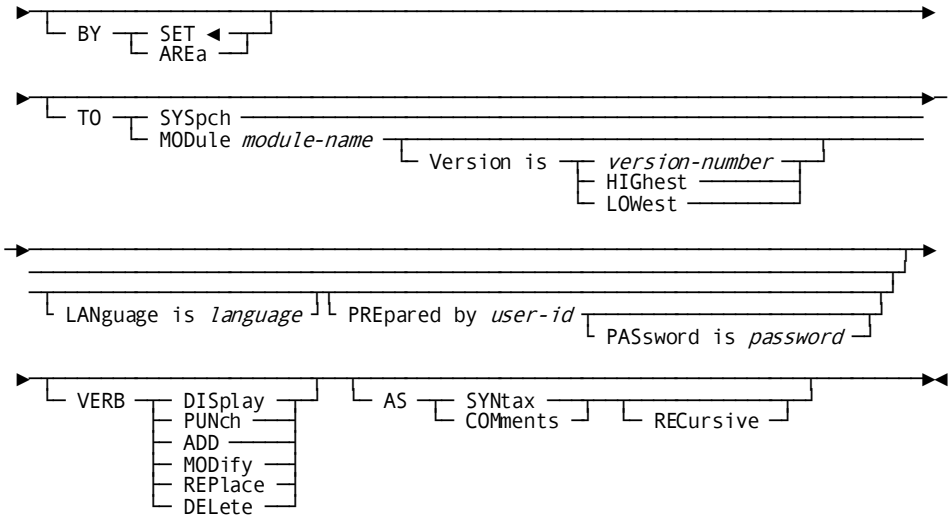




DISPLAY/PUNCH RECORD (REPORT) (TRANSACTION) statement (for multiple records)



(for complete **conditional-expression** syntax, see WHERE clause)


BY
SET
AREa
TO
SYSpch
MODule module-name
Version is
version-number
HIGhest
LOWest
LANguage is language
PREpared by user-id
PASsword is password
VERB
DISplay
PUNch
ADD
MODify
REPlace
DELete
AS
SYNtax
COMments
RECursive

Parameters

RECord/REPort/TRAnsaction name is *record-name*

Identifies a new record (report, transaction) to be established in the dictionary, or an existing record to be modified, replaced, deleted, displayed, or punched. *Record-name* must be a 1- through 32-character alphanumeric value.

The combination of the record name and version number must be unique in the dictionary; that is, it must not duplicate the primary or synonym name of an existing record, report, or transaction.

ENTIty type is RECord/REPort/TRAnsaction

Changes the entity-type name to RECORD, REPORT, or TRANSACTION. This clause is meaningful only with a MODIFY statement.

SAMe AS RECord/REPort/TRAnsaction *record-name*

Copies all entries associated with the specified record occurrence, with the exception of the NAME, WITHIN FILE, RECORD NAME SYNONYM and associated options, ELEMENT NAME SYNONYM, INDEXED BY FOR RECORD SYNONYM, and VIEW ID specifications.

NEW NAMe is *new-record-name*

Specifies a new name for the requested record. This clause changes the name of the record occurrence only; it does not alter or delete any relationships in which the record participates. Subsequent references to the record must specify the new name. *New-record-name* must be a 1- through 32-character alphanumeric value. The combination of the new record name and version number must not duplicate that of an existing record, report, transaction, or synonym in the dictionary. If the requested record participates in a schema, the NEW NAME clause is not valid.

NEW Version is *new-version*/NEXt HIGhest/NEXt LOWest

Specifies a new version number for the named record. The combination of the record name and new version number must not duplicate that of an existing record, report, transaction, or synonym in the dictionary.

RECord/REPort/TRAnsaction STOrage is *storage-indicator*

Documents the named record's storage medium or method; for example, tape or disk. *Storage-indicator* must be a 1- through 16-character alphanumeric value.

FORmat is

Specifies the format to be assigned to every element that participates in the named record-element structure.

Note: This specification applies only to elements that are included in the named record by means of the RECORD ELEMENT substatement.

PRIMary

Specifies that the primary format is to be used.

alternate-format-keyword

Specifies that an alternative format is to be used. *Alternate-format-keyword* must reference a valid alternative picture keyword as defined in the SET OPTIONS statement. If an element within the record does not have a corresponding alternative format, the DDDL compiler assigns the primary format to that element. For further discussion of alternative formats, see ELEMENT (see page 151), earlier in this chapter. Also see SET OPTIONS Statement (see page 34).

OCCurrences are *occurrence-count*

Specifies the actual or estimated number of times the record will occur in files or databases. *Occurrence-count* must be in the range 0 through 2,147,483,647. This clause is documentation only.

WITHIN FILE *file-name*

Associates (INCLUDE) or disassociates (EXCLUDE) a file in which the named record occurs. *File-name* must be the primary name of an existing file. This clause creates a record-synonym/for-file-synonym relationship between the primary record synonym and primary file synonym established by means of the FOR FILE SYNONYM parameter (described below). WITHIN FILE is documentation only. The KEY parameter is not valid with EXCLUDE.

KEY is *element-name*

Specifies that the named record is sequenced on keys within the file. *Element-name* specifies the names of fields to be used for sort control; the specified element need not participate in the named record. Each record definition can include up to five KEY parameters.

ASCending

Specifies that the records in the file are sorted by *element-name* in sequence from lowest to highest value.

DEScending

Specifies that the records in the file are sorted by *element-name* in sequence from highest to lowest value.

LANguage/MODe/*class-name* is *attribute-name*

Relates the named record to the named attribute by means of the specified class. The following considerations apply if the LANGUAGE or MODE class is specified:

- **LANGUAGE** identifies the language of programs in which the named record will be used. If LANGUAGE is specified, *attribute-name* must identify an existing attribute within the LANGUAGE class.
- **MODE** identifies the operating mode of programs in which the named record will be used.

For additional rules pertaining to this clause, see Attribute/Entity Relationships (see page 109).

RECord/REPort/TRAnsaction NAMe synonym is

Establishes (INCLUDE) or removes (EXCLUDE) a synonym (alternative name) for the record or modifies an existing synonym. When a record is added to the dictionary, the DDDL compiler builds a record synonym using the record's primary name and version number. This synonym is known as the primary record synonym. Any number of synonyms can be defined for the primary record synonym. If EXCLUDE is specified, only the RECORD NAME SYNONYM parameter is valid.

record-synonym

Specifies the 1- through 32-character name of a record synonym or the primary record name. If the optional VERSION parameter is not specified, the DDDL compiler uses the default version number established in the SET OPTIONS statement DEFAULT FOR EXISTING VERSION clause; if no record synonym exists with the default existing version, the DDDL compiler uses the default version number established in the SET OPTIONS statement DEFAULT FOR NEW VERSION clause.

FOR FILe synonym *file-synonym*

Associates (INCLUDE) the primary file or file synonym with, or disassociates it from (EXCLUDE), the record synonym. In the CA IDMS COBOL environment, this relationship determines which record synonyms are copied into a program when the DMLC precompiler performs an FD COPY function. In the CA Culprit environment, this relationship determines which record synonyms are associated with the CA Culprit file.

PREfix/SUFfix is *prefix/suffix*/NULl

Specifies a prefix or suffix for use with all elements that participate in the named record-element structure or removes an existing prefix or suffix. The specified prefix or suffix does not become part of the record synonym. However, the DDDL compiler adds the prefix/suffix to the beginning/end of the element or element synonym to form the record-specific element synonym. *Prefix/suffix* must be a 1- to 10-character value. The combined length of the element name and the prefix or suffix must not exceed 32 characters. If the resulting element-synonym name exceeds 32 characters, the DDDL compiler truncates the element name before adding the prefix or suffix.

LANguage is *language*

Associates a language defined as an attribute of the LANGUAGE class with, or disassociates it from, the record synonym. The DML precompilers use the LANGUAGE specification to determine the correct record synonyms to be copied into programs written in various languages.

WITh/ALSo WITh/WITHOut

Includes or excludes the specified options when the named record is displayed or punched. Detailed information for each DISPLAY/PUNCH option is under SET OPTIONS Syntax (see page 36). The options that are listed below present special considerations for this entity type.

DETails

Includes the DESCRIPTION, RECORD STORAGE, FORMAT, OCCURRENCES, VIEW ID, and RECORD LENGTH (displayed as comments) specifications.

ELements

Includes the specifications that describe the record-element format. ELEMENTS displays the names of record elements that are not subordinate to any other elements. To exclude elements from the display, specify WITHOUT ELEMENTS.

RECords

Includes all user-defined nests defined for the named record.

REPorts

Includes all user-defined nests defined for the named report.

TRAnsactions

Includes all user-defined nests defined for the named transaction.

COBol

Includes all COBOL format record elements associated with the named record. Note that ELEMENTS is the overriding option if a display of both ELEMENTS and COBOL is requested.

RECELems

Includes all COBOL format record elements associated with the named record. RECELEMS displays only the record-element name; the names and version numbers of the elements that participate in the record are not displayed. Note that ELEMENTS is the overriding option if a display of both ELEMENTS and RECELEMS is requested.

SUBOrdinate ELements

Includes subordinate elements. SUBORDINATE ELEMENTS is valid only with the RECORD ELEMENT format; to display or punch a COBOL format, specify the DISPLAY WITH COBOL option. To exclude subordinate elements from the display, specify WITH ELEMENTS PICTURE OVERRIDES WITHOUT SUBORDINATE ELEMENTS.

VIEws

Includes subschema or IDD views.

SYNonyms

Includes all synonyms associated with the record. Specify SYNONYMS to display programs, schemas, subschemas, and maps that are connected to the record synonym. For example, to display the programs with which the CUSTOMER record is associated, specify DISPLAY RECORD CUSTOMER WITH PROGRAMS SYNONYMS.

PICture OVErrides

Includes element picture definitions for the record, including the start position of the element within the record and the length of the element, in bytes.

Usage

Restrictions on the RECORD statement

The following restrictions apply to the RECORD statement:

- If the SET OPTIONS statement specifies SECURITY FOR IDD IS ON, the user must be assigned the proper authority to issue RECORD statements.
- *Records that participate in schemas* require special consideration when they are deleted or replaced. *Records that participate in schemas* cannot be *deleted* by the DDDL compiler; documentation entries can, however, be submitted.

  Within records that participate in schemas, record elements can be *replaced* by one or more record elements; optionally, one or more record elements that follow the replaced record elements in the record structure can be *removed*. The following considerations apply:

  - To modify record elements, use the RECORD ELEMENT substatement, described later in this chapter. It is recommended that the LINE option be used to accurately position the record element.
  - The primary replacement record element must have the same RECORD ELEMENT NAME as the original record element; however, a different version number is valid.
  - After issuing the REPLACE command for a particular record element, the user can *insert* or *remove* record elements immediately following the replaced record element, subject to the length restrictions described below.
  - Record elements to be inserted into the record structure following the replacement record elements must be previously defined in the dictionary.
  - The record elements to be replaced cannot be defined as the *schema control field* (CALC-key, sorted set key, or index set key), nor can they contain a subordinate element defined as the schema control field.
  - The *total length* of the replacement record elements must equal the length of the element being replaced. The logical position of the elements following the replaced element cannot be altered; the overall record length cannot be changed. When the DDDL compiler detects a change in the replacement record length, it rejects the request; the compiler restores the original elements in the record, removes replacement elements, and displays an error message.
  - A *filler field* (RECORD ELEMENT IS 'FIL *nnnn*') can be replaced by any element previously defined in the dictionary.
  - Elements defined as *COBOL level-88 items* (USAGE IS CONDITION-NAME) can be inserted, replaced, or removed from the record structure without restriction.
- *Records that participate in maps* can be modified; the following considerations apply:
  - *To modify record elements*, use the RECORD ELEMENT substatement, described later in this chapter. It is recommended that the LINE option be used to accurately position the record element.

- When a REPLACE RECORD ELEMENT command specifies that the length of the replacement elements is equal to the length of the original element, the DDDL compiler removes the original element from the record and inserts the new elements in its place. Recompilation of maps in which the record participates and programs that use the maps is not necessary.
- When a record element is removed from or inserted into the record structure, or is replaced with a *record element of unequal length*, the DDDL compiler updates the record and flags the maps and programs associated with the record for recompilation.
- Record elements and group record elements with subordinate record elements that are identified as *map fields* cannot be removed from or replaced in the record. The same restriction applies to record elements that are the object of the following statements that implicitly remove or replace record elements: REMOVE/REBUILD/REPLACE RECORD ELEMENTS, REMOVE ALL, and COBOL.
- An *occurrence count* for a multiply-occurring record element cannot be decreased if it makes a field in a map obsolete. For example, a MODIFY RECORD statement followed by a RECORD ELEMENT substatement that specifies OCCURS 11 TIMES produces an error if the 12th occurrence of the field was mapped.
- A *record synonym* that participates in a map cannot be excluded.
- *Modifications* to the RECORD statement clauses listed in the following table may necessitate regeneration of all maps in which the record participates and, in some cases, recompilation of the programs that use those maps. To obtain a list of such programs, issue a DISPLAY MAP statement for each map in which the named record participates; the output lists the programs compiled against that map.
- Elements defined as *COBOL level-88 items* (USAGE IS CONDITION-NAME) can be inserted, replaced, or removed from the record structure without restriction.

Regenerate and recompile requirements

Modified RECORD clause	Map regeneration required?	Program recompilation required?
BLANK WHEN ZERO	NO	NO
CODE TABLE	YES	NO
COMMENTS/DEFINITION	NO	NO
EDIT	NO	NO
EDIT TABLE	YES	NO
ELEMENT NAME SYNONYM	NO	NO

Modified RECORD clause	Map regeneration required?	Program recompilation required?
EXTERNAL PICTURE	YES	NO
INDEX KEY	NO	NO
INDEXED BY	NO	NO
JUSTIFY	NO	NO
NEW RECORD NAME/VERSION	NO	NO
OCCURS	YES	YES
OCCURS DEPENDING ON name	NO	NO
PICTURE	YES	YES
REMOVE RECORD ELEMENT	YES[1]	YES[1]
REPLACE RECORD ELEMENT	YES[1]	YES[1]
ADD RECORD ELEMENT	YES[1]	YES[1]
RECORD NAME SYNONYM PREFIX/SUFFIX	NO	NO
REDEFINES	YES	YES
SIGN	YES	YES
SYNC/NOSYNC	NO	NO
USAGE	YES	YES
VALUE	NO	NO

Note: [1] Only necessary if displacements are affected

If you specify REPLACE

If you specify REPLACE, the DDDL compiler initializes to defaults and/or excludes the following clauses:

- DESCRIPTION
- USER REGISTERED FOR
- PUBLIC ACCESS
- STORAGE
- FORMAT
- OCCURRENCES
- WITHIN FILE
- COMMENTS/DEFINITIONS/OLQ HEADER/CULPRIT HEADER/ *comment-key*
- ATTRIBUTES
- RECORD SYNONYMS (except as noted below)
- RECORD ELEMENT
- VIEW
- PREFIX/SUFFIX
- Related records, reports, transactions
- LANGUAGE

The following relationships are not affected:

- Primary record synonyms
- Records to which the named record is related
- Programs that have copied or access the named record
- Map records associated with the named record
- Record synonyms that are copied or accessed by programs or maps or that have not been built by the DDDL compiler

Record synonyms (sites using CA IDMS SQL)

If the language specified is SQL, no other record synonym associated with the record may have a language of SQL associated. The language of SQL is associated with record synonyms to be used (at sites with CA IDMS SQL) in SQL to access non-SQL databases.

Displaying records

If you display a record:

- WITH ELEMENTS— the names of record elements that are *not* subordinate to any other elements are displayed
- WITH ELEMENTS ALSO WITH SUBORDINATE ELEMENTS—the names of subordinate record elements are displayed
- WITH ELEMENTS ALSO WITH PICTURE OVERRIDES—the element definitions for the specified record are displayed, including: the start position of the element within the record and the length of the element, in bytes

Note: The message RECORD CONTAINS CRITICAL ERRORS applies only to CA products and appears when you display any record, report, or transaction that contains a critical error (for example, a record or record element with a length of 0).

If WITHOUT PICTURE OVERRIDES is specified

If WITHOUT PICTURE OVERRIDES is requested, the displayed output includes the record elements, subordinate elements, element synonyms, and all information that can be specified only at the record-element or subordinate-record-element level rather than in the element definition (for example, SYNC, OCCURS, INDEXED BY, and INDEX KEY). The LINE IS, SUBORDINATE ELEMENT REDEFINES, and SUBORDINATE ELEMENT OCCURS specifications and picture-related information are excluded from the display.

This option is useful in an online environment for rebuilding a record, modifying portions of a record, or building a new record, as follows:

- *To rebuild a record*, issue a DISPLAY request, specifying the WITHOUT PICTURE OVERRIDES, WITHOUT SYNONYMS, VERB IS MODIFY, and AS SYNTAX options. Insert a REMOVE ALL substatement immediately following the MODIFY RECORD statement. Resubmit the displayed definition to the DDDL compiler, which rebuilds the record as if it were performing an ADD operation (using the current element definitions to build the record-element structure). The displayed record definition can be replaced by specifying the VERB IS REPLACE parameter on the DISPLAY/PUNCH request.
- *To rebuild portions of a record*, issue a DISPLAY request, specifying the WITHOUT PICTURE OVERRIDES, VERB IS MODIFY, and AS SYNTAX options. Specify the REPLACE option for each displayed record element to be changed. Resubmit the displayed definition to the DDDL compiler. The DDDL compiler uses the current definition of each element named in the record (for which REPLACE has been specified) to rebuild the record.
- *To build a new record using an existing element structure*, issue a DISPLAY request, specifying the WITHOUT PICTURE OVERRIDES and AS SYNTAX options. Supply the new record name and/or version number, and resubmit the record definition to the DDDL compiler. The DDDL compiler uses the picture-related information and group-to-subordinate-element structure from the current definition of each element named in the record to build the new record.

Note: If a subordinate record element is defined with both a REDEFINES and an OCCURS clause, the REDEFINES specification is supplied from the element and the OCCURS specification is supplied from the subordinate record element when the record is built or rebuilt.

Example 1

This example shows:

1. Definition of two elements
2. Definition of a group element
3. Association of the group element with a record
4. Display of the entire record

1) Defining elements

The following two ADD ELEMENT statements establish the elements CUSTOMER-NUMBER and CUSTOMER-NAME in the dictionary. Element names are used when the elements appear with the record having the primary record name. Element synonyms provide language-specific names.

```
add element name is customer-number version is 1
    element synonym is customer_number for group synonym customer_group
    element synonym is custnum for group synonym custgrup
    element synonym is custno for group synonym custgp
        picture is 9(6)
        .

add element name is customer-name version is 1
    element synonym is customer_name for group synonym customer_group
    element synonym is custname for group synonym custgrup
    element synonym is custnm for group synonym custgp
        picture is x(30)
        .
```

2) Defining a group element

The following ADD ELEMENT statement establishes the *group* element CUSTOMER-GROUP. The SUBORDINATE ELEMENT clause incorporates the elements CUSTOMER-NAME and CUSTOMER-NUMBER. ELEMENT SYNONYM clauses are used to establish a connection between the ELEMENT definition and a RECORD definition.

```
add element name is customer-group version is 1
    element synonym is customer_group for group synonym customer_record
    element synonym is custgrup for group synonym custrecd
    element synonym is custgp for group synonym custrc
    subordinate elements are
        customer-number version is 1
        customer-name version is 1
```

.

3) Associating the group element with a record

The following ADD RECORD statement adds the record CUSTOMER-RECORD to the dictionary and includes the group element CUSTOMER-GROUP.

The DDDL compiler compares the record synonyms with the group synonyms in the element definition. When a match is found, the element synonym associated with that group synonym is automatically copied into the record for that record synonym.

```
add record name is customer-record version is 1
    record name synonym is customer_record version 1
    record name synonym is custrecd version 1
    record name synonym is custrc version 1.
    record element is customer-group.
```

4) Displaying the record

The following DISPLAY RECORD statement displays the CUSTOMER-RECORD structure defined in steps 1 through 3.

```
display record customer-record.
*+ add
*+ record name is customer-record version is 1
*+     date created is      mm/dd/yy
*+     prepared by mjj
*+     record length is 36
*+     record name synonym is customer-record version 1
*+     record name synonym is customer_record version 1
*+     record name synonym is custrecd version 1
*+     record name synonym is custrc version 1
*+     .
*+     record element is customer-group version 1 line is 000100
*+     level number is 02
*+     usage is display
*+     element name synonym for record synonym customer_record version 1 is
*+           customer_group
*+     element name synonym for record synonym custrecd version 1 is custgrup
*+     element name synonym for record synonym custrc version 1 is custgp
*+     .
*+     subordinate element is customer-number version 1 line is 000200
*+     level number is 03
*+     picture is 9(6) usage is display
*+     element name synonym for record synonym customer_record version 1 is
*+           customer_number
*+     element name synonym for record synonym custrecd version 1 is custnum
*+     element name synonym for record synonym custrc version 1 is custno
*+     .
*+     subordinate element is customer-name version 1 line is 000300
```

```
*+     level number is 03
*+     picture is x(30) usage is display
*+     element name synonym for record synonym customer_record version 1 is
*+           customer_name
*+     element name synonym for record synonym custrecd version 1 is custname
*+     element name synonym for record synonym custrc version 1 is custnm
*+     .
```

Example 2

The following example illustrates the modification of a record that participates in a schema. It shows:

1. The original record layout
2. The DDDL statements that modify the record
3. The final record layout

1) The original record layout

The CUSTOMER record has been previously defined in the dictionary. The length of CUST-ADDRESS is 40 bytes.

Note: The COBOL layout and version numbers of the elements are for illustrative purposes only. Each of these elements must be defined in the dictionary using RECORD ELEMENT syntax.

```
Line
Num       Record Element

 100   05  cust-number             ver 1   pic x(10).
 200   05  cust-name               ver 1   pic x(20).
 300   05  cust-ssn                ver 1   pic x(09).
 400   05  cust-address            ver 1.
 500       10  cust-addr1          ver 1   pic x(20).
 600       10  cust-addr2          ver 1.
 700           15  cust-city       ver 1   pic x(15).
 800           15  cust-zip-code   ver 1   pic x(05).
 900           15  cust-zipcode    ver 1   redefines cust-zip-code
                                            pic 9(05).
1000   05  filler                          pic x(05).
1100   05  cust-credit             ver 1   pic x(03).
1200       88  cust-credit-exec    ver 1             value 'aaa'.
1300       88  cust-credit-good    ver 1             value '   '.
1400       88  cust-credit-poor    ver 1             value 'xxx'.
1500   05  cust-sales-info         ver 1.
1600       10  cust-sales-qtr      ver 1   occurs 4  indexed by cuix.
1700       10  cust-num-sales      ver 1   pic 9(05) comp-3.
1800       10  cust-amt-sales      ver 1   pic 9(07) comp-3.
1900   05  filler                          pic x(03).
```

The user defines four new elements in the dictionary, using the DDDL compiler. The length of CUST-ADDRESS VERSION 2 is 44 bytes:

Note: The COBOL layout and version numbers of the elements are for illustrative purposes only. Each of these elements must be defined in the dictionary using RECORD ELEMENT syntax.

```
          Record Element

      05  cust-nr-numeric          ver 1   pic 9(10).

      05  cust-ssn                 ver 2.
          10  cust-ssn-3           ver 1   pic x(03).
          10  cust-ssn-2           ver 1   pic x(02).
          10  cust-ssn-4           ver 1   pic x(04).

          88  cust-credit-unkn     ver 1          value'unk'.

      05  cust-address             ver 2.
          10  cust-street          ver 1   pic x(20).
          10  cust-addr2           ver 2.
              15  cust-city        ver 2   pic x(13).
              15  cust-state       ver 1   pic x(02).
              15  cust-zip-code    ver 2.
                  20  filler               pic x(04).
                  20  cust-zip-5   ver 1   pic x(05).
              15  cust-zipcode     ver 2   redefines
                  cust-zip-code     ver 2   pic 9(09).
```

2) The DDDL statements that modify the record

The user issues a MODIFY RECORD command and RECORD ELEMENT substatements to the DDDL compiler to place the newly defined elements into the CUSTOMER record.

CUST-NUMBER is replaced by the new element definition CUST-NR-NUMERIC.

```
replace record customer.

    replace record element cust-number   version 1    line 100 .

        record element cust-nr-numeric   version 1    line 110
            redefines cust-number .
```

CUST-SSN and CUST-ADDRESS are replaced using a new element with the same name but a different version number. Because CUST-ADDRESS VERSION 2 is four bytes longer than VERSION 1, a new one-byte filler field is inserted following CUST-ADDRESS, and the original five-byte filler field is removed. A new COBOL level-88 item is inserted.

```
        replace record element cust-ssn     version 2 line 300 .
```

```
replace record element cust-address version 2 line 400 .

        record element 'fil 0001'          line 910 .

remove record element 'fil 0005'           line 1000 .

        record element cust-credit-unkn       line 1110 .
```

3) The final record layout

The record elements associated with the newly modified CUSTOMER record are shown below. The schema definition is automatically adjusted to reflect the changes.

```
Line
Num       Record Element

 100   05  cust-number             ver 1   pic x(10).
 200   05  cust-nr-numeric         ver 1   redefines cust-number
                                           pic 9(10).
 300   05  cust-name               ver 1   pic x(20).
 400   05  cust-ssn                ver 2.
 500       10  cust-ssn-3          ver 1   pic x(03).
 600       10  cust-ssn-2          ver 1   pic x(02).
 700       10  cust-ssn-4          ver 1   pic x(04).
 800   05  cust-address            ver 2.
 900       10  cust-street         ver 1   pic x(20).
1000       10  cust-addr2          ver 2.
1100           15  cust-city       ver 2   pic x(13).
1200           15  cust-state      ver 1   pic x(02).
1300           15  cust-zip-code   ver 2.
1400               20  filler              pic x(04).
1500               20  cust-zip-5  ver 1   pic x(05).
1600           15  cust-zipcode    ver 2   redefines
                   cust-zip-code    ver 2   pic 9(09).
1700   05  filler                          pic x(01).
1800   05  cust-credit             ver 1   pic x(03).
1900       88  cust-credit-unkn    ver 1             value 'unk'.
2000       88  cust-credit-exec    ver 1             value 'aaa'.
2100       88  cust-credit-good    ver 1             value '   '.
2200       88  cust-credit-poor    ver 1             value 'xxx'.
2300   05  cust-sales-info         ver 1.
2400       10  cust-sales-qtr      ver 1   occurs 4  indexed by cuix.
2500       10  cust-num-sales      ver 1   pic 9(05) comp-3.
2600       10  cust-amt-sales      ver 1   pic 9(07) comp-3.
2700   05  filler                          pic x(03).
```

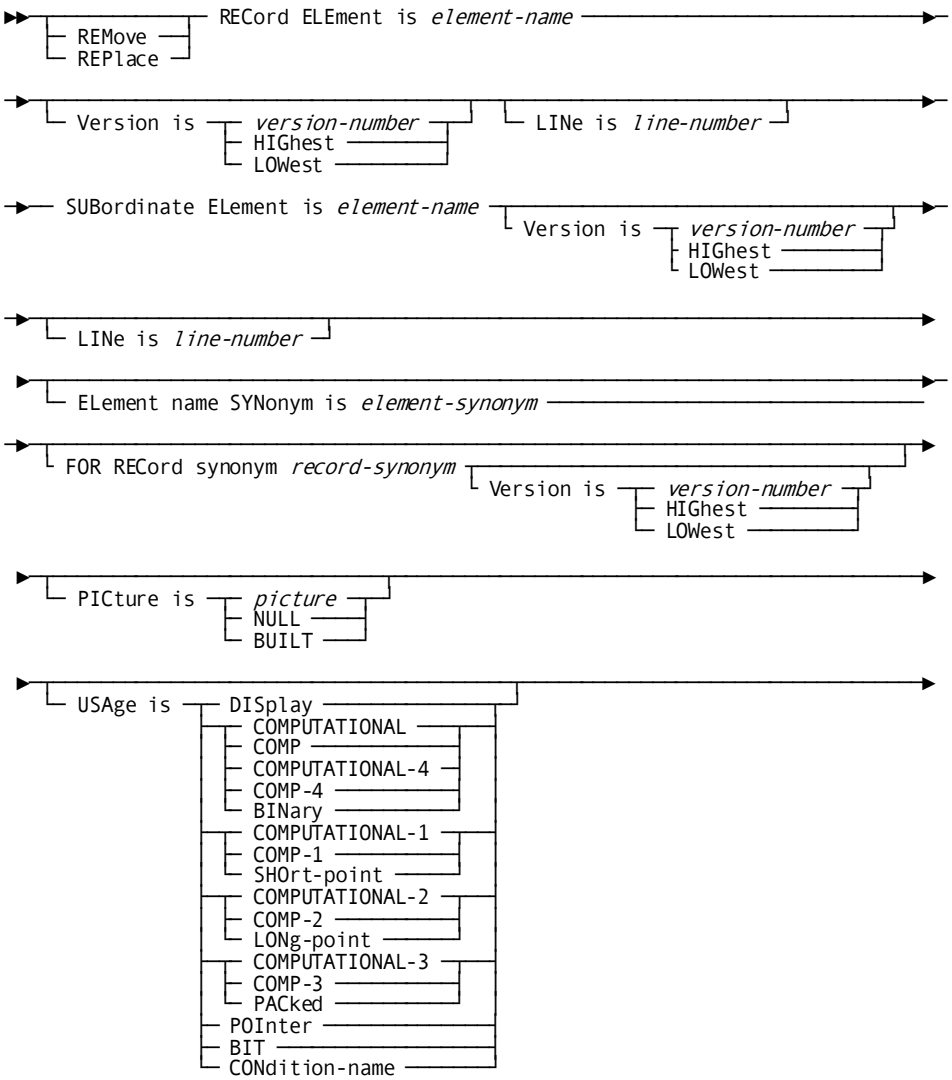
Syntax and parameter descriptions for the RECORD ELEMENT, COBOL, REMOVE ALL, and VIEW substatements, and the DISPLAY/PUNCH RECORD SYNONYM statement follow.

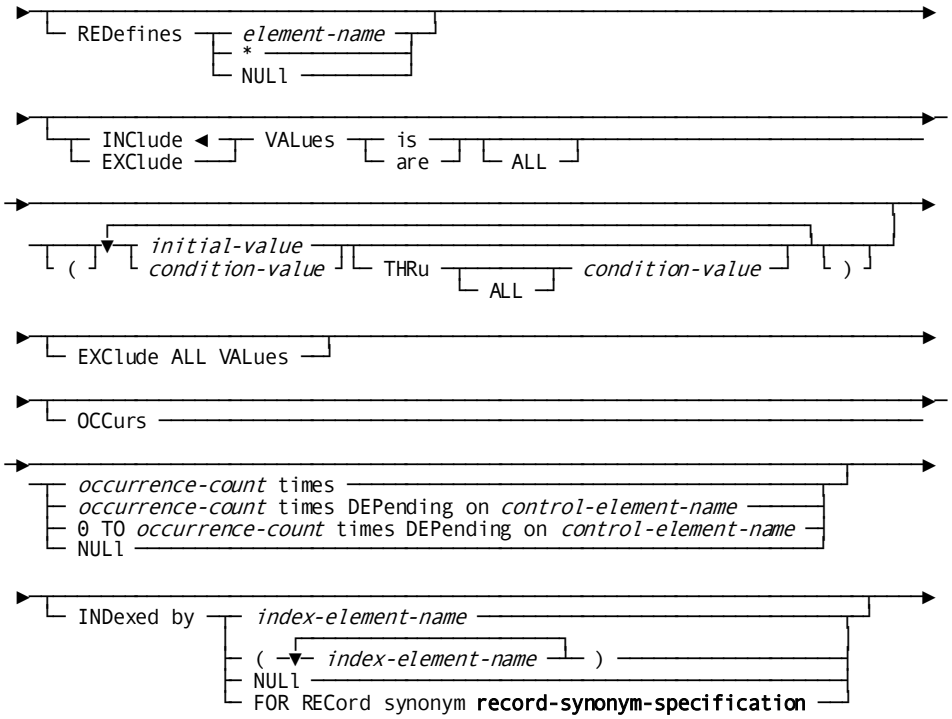
RECORD ELEMENT Substatement

RECORD ELEMENT substatements associate existing elements with records and update existing record-element structures. To include an element within a record-element structure, specify a RECORD statement followed by the keywords RECORD ELEMENT. After the RECORD ELEMENT identification, enter optional clauses that define record-specific characteristics for the element.

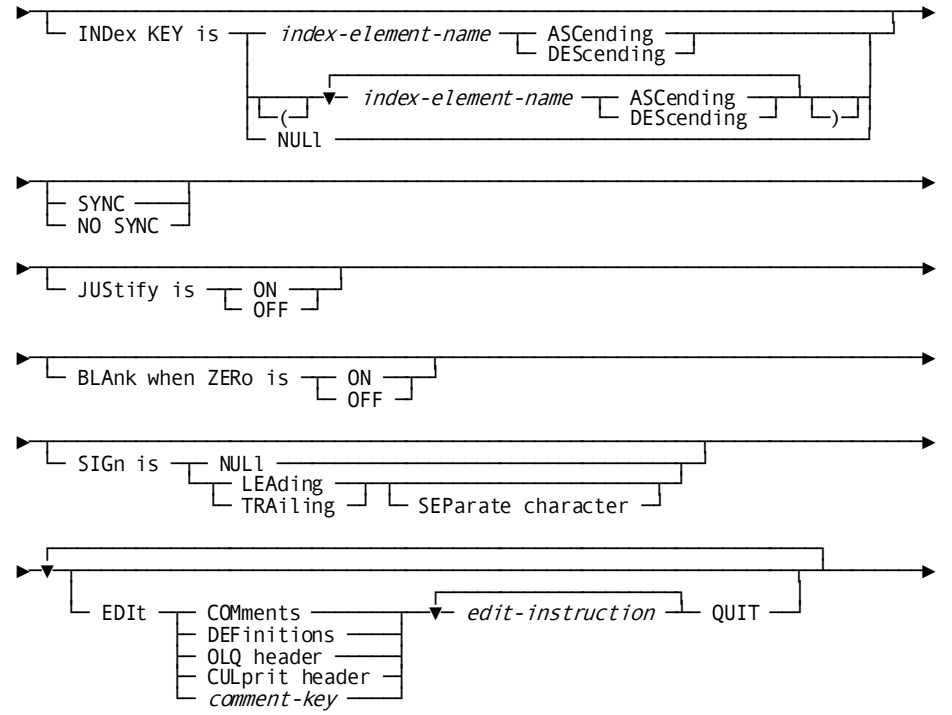
Syntax

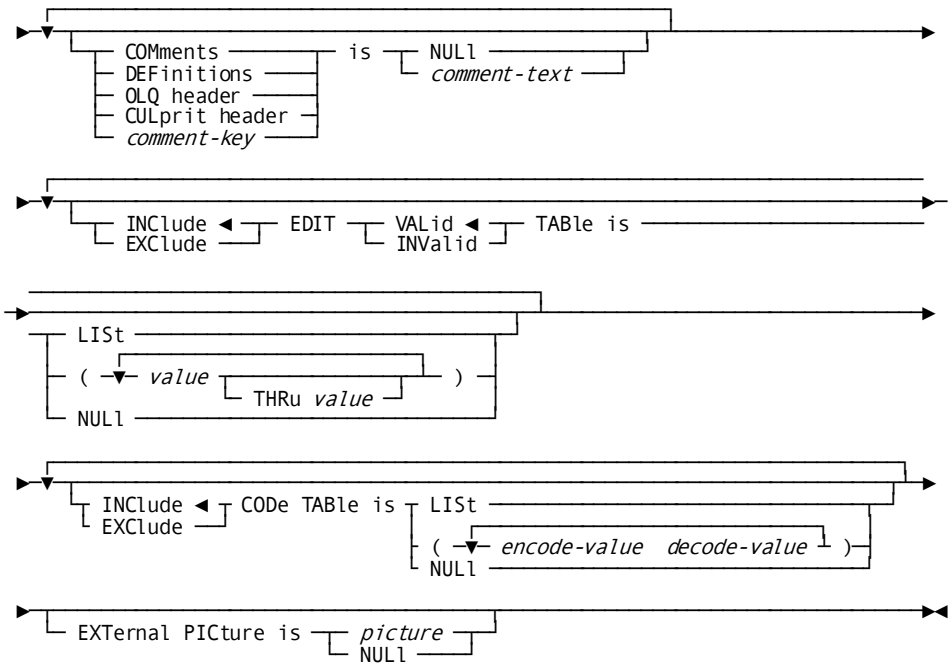
RECORD ELEMENT substatement



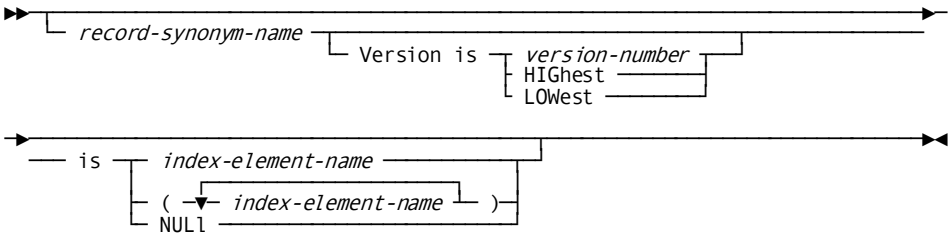


(expanded **record-synonym-specification** syntax follows this syntax diagram)





Expansion of *record-synonym-specification*



Parameters

RECord ELEment is *element-name*

Specifies the element that is the object of the RECORD ELEMENT substatement. *Element-name* must be the primary name of an existing element; the named element must be the highest level element within a record (usually an 02 level), or a level-88 item. If the named element is not in the record-element structure, the DDDL compiler adds the existing element definition and any record-specific characteristics to the end of the record-element structure. If the optional LINE parameter (described below) is not specified, the element definition is placed at the end of the record-element structure. If the named element already participates in the record-element structure, the DDDL compiler modifies the record-element definition based on the optional clauses specified.

LINe is *line-n*

Qualifies nonunique record-element names or specifies where the DDDL compiler is to insert a new element definition in the record-element structure. *Line-n* must be an integer in the range 1 through 999,999.

This parameter must be specified unless the requested record element is the first nonunique element within the structure. Following compilation of the RECORD statement, the DDDL compiler assigns sequence numbers to all record elements; the default sequence number specified in the SET OPTIONS statement SEQUENCE clause is the starting and increment value. The assigned numbers appear on record reports and in DISPLAY/PUNCH output. If the LINE parameter in a MODIFY RECORD RECORD ELEMENT statement references the line number of an existing record element, but that element has a different name than the requested record element, the DDDL compiler issues an error message, unless the REPLACE parameter has been specified.

Note: If you do *not* specify a LINE clause in a RECORD ELEMENT substatement (one that doesn't use REPLACE or REMOVE) the compiler adds the record element to the end of the record definition.

REMove/REPlace

Deletes or replaces the specified record element and its subordinate elements. If REMOVE or REPLACE is specified, *element-name* must match the name of an element in the named record-element structure.

SUBordinate ELement is *element-name*

Identifies an existing subordinate element that is to be modified for use within the named record-element structure. All clauses between this substatement and another SUBORDINATE ELEMENT or RECORD ELEMENT substatement apply to the named subordinate element or level-88 item.

Note: The SUBORDINATE ELEMENT specification is used to change record-element characteristics such as picture and usage; it is *not* used to create group-element/subordinate-element structures. These structures must be defined by means of the ELEMENT entity statement (described under ELEMENT (see page 151), earlier in this chapter).

ELement name SYNonym is *element-synonym*

Establishes a synonym (alternative name) for the element when it participates in the named record-element structure. *Element-synonym* is the 1- to 32-character synonym name. If this clause appears following a SUBORDINATE ELEMENT substatement, the synonym is associated with the subordinate element.

This clause can be coded once for each synonym to be associated with the named record. Any prefix or suffix defined for the record synonym with which the element synonym is associated will be appended to the element-synonym name.

FOR RECord synonym *record-synonym-specification*

Associates the element (or subordinate element) synonym with the designated record synonym.

Note: If the FOR RECORD SYNONYM clause is not specified, the named record element (or subordinate element) is associated with the primary record synonym only.

PICture is

Specifies a record-specific PICTURE clause for the record element or subordinate element.

picture

Creates a record-specific PICTURE clause for the named record element. If the named record element is an elementary element, *picture* becomes the record-specific picture for the element. If the named record element is a group element, *picture* becomes the record-specific picture for the group element; the DDDL compiler removes from the record any subordinate elements defined for the group.

Note: The maximum length of a record element (including its usage) is 32,767 characters.

Picture must be a 1- through 30-character value that describes alphanumeric, alphabetic, numeric, or numeric-edited data, as shown in the table under **Usage**.

NULL

Removes a record-specific PICTURE clause from the named record element.

BUILT

Creates an alphanumeric display PICTURE clause for the requested group record element. The DDDL compiler deletes the subordinate elements from the group description and uses the combined lengths of all subordinate elements in the group to form the group picture.

USAge is

Specifies a record-specific USAGE clause for the named record element.

DISplay

Alphabetic, alphanumeric, zoned decimal, edited, or display floating point

COMPUTATIONAL

Binary; COMP, COMPUTATIONAL-4, COMP-4, and BINary are synonyms for COMPUTATIONAL.

COMPUTATIONAL-1

Short-precision floating point; COMP-1 and SHOrt-point are synonyms for COMPUTATIONAL-1.

COMPUTATIONAL-2

Long-precision floating point; COMP-2 and LONG-point are synonyms for COMPUTATIONAL-2.

COMPUTATIONAL-3

Packed decimal; COMP-3 and PACked are synonyms for COMPUTATIONAL-3.

POInter

Fullword address constant

BIT

Bit string definition

CONdition-name

COBOL level-88 item; the level number is generated by the DDDL compiler

REDefines

Specifies a record-specific REDEFINES clause for the named record element. A redefined element (*element-name* or the element referenced by *) cannot include an OCCURS clause; the element can, however, be subordinate to an element with an OCCURS clause.

element-name

Identifies the element being redefined. The specified element must be at the same level as the element that is the object of the RECORD ELEMENT substatement and must immediately precede that element in the record-element structure.

*

Instructs the DDDL compiler to automatically redefine the previous element at the same level in the record-element structure. The user need not specify the element name.

NULl

Removes a previously established REDEFINES clause.

VALue is/are ALL *initial-value/condition-value* THRu ALL *condition-value*

Specifies a record-specific VALUE clause for the named record element. *Initial-value/condition-value* specifies a value, range of values, or a list of values assigned to a COBOL level-88 condition name. A list of values must be enclosed in parentheses. Each value in the list must be separated from the next by a space or a comma. The value must be a figurative constant, a numeric literal, or an alphanumeric literal enclosed in quotation marks; alphanumeric literals cannot exceed 32 characters.

The optional THRU parameter is valid only with COBOL condition names (level-88 items). To specify a new value for a new or existing record element, first issue an EXCLUDE ALL VALUES clause. Note that if the SET OPTIONS statement specifies DECIMAL-POINT IS COMMA and the VALUE clause specifies a numeric literal, periods (.) are interpreted as insertion characters and commas (,) are interpreted as decimal points.

EXClude ALL VALues

Removes all VALUE clauses associated with the named record element. This clause is required to remove existing values.

OCCurs

Specifies a record-specific OCCURS clause for the named record element.

***occurrence-count* times**

Specifies the number of times the element can occur within the record. *Occurrence-count* must be an integer in the range 0 through 32,767.

occurrence-count* times/0 TO *occurrence-count* times DEPending on *control-element-name

Defines a control element within the record, that determines the actual number of times the element will occur. *Occurrence-count* must be an integer in the range 1 through 32,767. *Control-element-name* specifies a previously defined field in the record; this field must be a halfword or fullword binary item if the record is to be used in a schema or by CA ADS.

NULL

Removes an existing OCCURS clause.

INDexed by

Specifies one or more INDEXED BY clauses for the named multiply-occurring record element or record-element synonym, or removes an existing INDEXED BY clause.

This clause applies only to records used in COBOL programs and can be specified once for each record element, subordinate element, and record synonym associated with the element. Each specified index is prefixed or suffixed for each record synonym associated with the record element.

Note: Within one INDEXED BY clause, the user can specify either a multiply-occurring record element or a record-element synonym (the clause cannot contain both elements and synonyms).

index-element-name

Specifies a 1- through 32-character index name that cannot duplicate an element or element-synonym name in the record.

NULl

Removes an existing INDEXED BY clause.

FOR RECord synonym *record-synonym* is *index-element-name*/NULl

Specifies an INDEXED BY name for a record synonym associated with the record element.

Index-element-name is a 1- through 32-character index name that must not duplicate the name of an existing element or element synonym in the record. Multiple index names must be enclosed in parentheses and separated by blanks. The COBOL precompiler copies the specified index names into the program's DATA DIVISION as part of the COPY IDMS function.

NULl removes an existing INDEXED BY clause.

INDex KEY is

Specifies one or more record-specified index keys for a multiply-occurring group record element or a subordinate record element. The INDEX KEY clause applies only to records used in COBOL programs. Only one INDEX KEY clause can be specified for each record element or subordinate element.

index-element-name

Identifies an elementary element that is subordinate to the associated element. *Index-element-name* must be the primary name of the subordinate element; it cannot be a synonym. The specified element name is appended with a prefix or suffix assigned to the record synonym associated with the element.

ASCending/DEScending

Specifies the order of the subordinate elements within the multiply-occurring element.

NULl

Removes an existing INDEX KEY clause.

SYNC/NO SYNC

Determines whether boundary alignment is to be defined for the named record element. The correct alignment is determined by the USAGE specification. If the element's usage is COMP or COMP-4, the DDDL compiler issues a warning message when the element is not on the proper boundary alignment. This clause is documentation only, unless the COBOL precompiler is used to copy the record; in this case, the specified boundary alignment will be applied by the COBOL precompiler. If the record element is copied into a schema, it causes a critical schema error.

JUStify is

Supplies a justification specification for the named record element. The JUSTIFY clause applies only to records used in COBOL programs.

ON

Specifies that a COBOL JUSTIFIED clause is to be generated.

OFF

Specifies that a COBOL JUSTIFIED clause is *not* to be generated.

BLAnk when ZERo is

Supplies a BLANK WHEN ZERO specification for the named record element.

ON

Specifies that blanks are automatically placed in the element when it contains all zeroes.

OFF

Specifies that the element's value will not be changed when it contains all zeroes.

SIGn is

Specifies whether the sign is to be removed from a numeric field or whether it is to appear in the leading or trailing position.

NULl

Removes existing sign specifications (for signed DISPLAY numeric fields only).

LEAding/TRAiling

Places the sign in the leading or trailing position. If SEPARATE CHARACTER is specified, the sign will appear as a separate byte.

EDIt VALid/INValid TABle is

Specifies whether edit table values are to be listed, inserted, or removed (for DC/UCF tables only).

LISt

Edit table values in the dictionary are listed on the Integrated Data Dictionary Activity List or in the online IDD work file.

value* THRu *value

Inserts (INCLUDE) or removes (EXCLUDE) single values or ranges of values in the edit table. Each value can have a maximum size of 34 characters. The specified values must be enclosed in parentheses; for example:

```
('A' 'C' 'F' 'H' 'R' THRU 'Z')
```

NULl

All values are removed from the table.

VALid/INValid

Specifies whether the edit table contains a list of valid or invalid values; the default is VALID.

CODe TABle is

Specifies whether code table values are to be listed, inserted, or removed (for DC/UCF tables only). For the rules for defining the values for edit and code tables, refer to the *CA IDMS Mapping Facility Guide*.

LISt

Specifies that code table values that are in the dictionary are to be listed in pairs. The first value is the encoded value; the second value is the decoded value.

encode-value decode-value

Specifies that pairs of values are inserted in the table. The first or *encoded* value can have a maximum size of 34 characters; the second or *decoded* value can have a maximum of 64 characters. Null values ('') and the keywords NOT FOUND are also valid. The specified values must be enclosed in parentheses.

NULl

Specifies that all values are removed from the table.

EXTernal PICture is *picture*/NULl

Defines the display format for record-element data (*picture*) or removes an existing external picture specification (NULL). The picture is available to all map fields that use the record element. For the rules for defining external pictures, refer to the *CA IDMS Mapping Facility Guide*.

Usage

RECORD ELEMENT considerations

The following considerations apply to the RECORD ELEMENT substatement:

- A record element can have a maximum length of 32,767 characters.
- Any number of record elements can be associated with one record.
- Clauses in a RECORD ELEMENT substatement request the DDDL compiler to change the *record-element* structure; the element definition in the dictionary remains unchanged.
- When MODIFY RECORD is specified and the named element exists, the DDDL compiler modifies only those portions of the record element definition that are referenced by RECORD ELEMENT substatement clauses.
- When the requested record element is not in the record, the DDDL compiler adds the definition to the end of the record.
- When a RECORD ELEMENT substatement names a group element, the DDDL compiler automatically copies all of the group's subordinate elements into the definition.
- The RECORD ELEMENT substatement does *not* build group-to-subordinate-element relationships. These relationships must be established using ELEMENT entity statements (see ELEMENT (see page 151), earlier in this chapter.

- RECORD ELEMENT statements are used to define record elements as tables that are used by the DC/UCF mapping facility for automatic editing and error handling. Tables defined by means of the RECORD ELEMENT substatement are called *built-in tables*.

  The TABLE statement (see TABLE (see page 341), later in this chapter) is used to define stand-alone tables. For a complete description of built-in and stand-alone tables, refer to the *CA IDMS Mapping Facility Guide*.

Adding a filler field to a record-element structure

To add a filler field to a record-element structure, specify RECORD ELEMENT IS 'FIL *nnnn*'. *Nnnn* is a 4-digit value that represents the number of characters of filler; the specified value must be separated from the keyword FIL by one blank and must contain leading zeros, if appropriate. For example, to generate the filler described as FILLER PIC X(7), specify RECORD ELEMENT 'FIL 0007'.

If you specify REPLACE

The following considerations apply to the REPLACE option:

- If REPLACE RECORD ELEMENT is specified with no optional clauses, the DDDL compiler removes and rebuilds the definition of the named record element from the current ELEMENT definition. Any record-specific modifications that have been made to the named element must be respecified; record-specific modifications for each subordinate element in a group must also be respecified.
- A record element that is replaced will be removed from any views in which it participates. The replacement record element will not automatically be included in any views.
- If REPLACE is specified with a line number, the DDDL compiler replaces the contents of the specified line number, whether or not the record element at that line has the same name as the element named in the REPLACE statement.

  Note: For information about using the REPLACE command to modify map-owned or schema-owned *records*, see the previous discussion under RECORD Statement (see page 288).

SUBORDINATE ELEMENT considerations

The following considerations apply to the SUBORDINATE ELEMENT clause:

- If no RECORD ELEMENT substatement has been specified, the search for the SUBORDINATE ELEMENT starts at the beginning of the record.
- A SUBORDINATE ELEMENT substatement cannot reference an element at the highest level in the record. Use the RECORD ELEMENT substatement to reference the highest level.
- Only one subordinate element can be referenced in each SUBORDINATE ELEMENT substatement.

- Multiple SUBORDINATE ELEMENT substatements must be specified in the order that the record elements appear within the group or within the record if no RECORD ELEMENT substatement has been specified.
- Each RECORD ELEMENT substatement that references a group element can be followed by one SUBORDINATE ELEMENT substatement for each subordinate element within the named group, and one or more of the optional clauses described below.

Specifying a **picture** variable

Picture must be a 1- through 30-character value that describes the types of data shown in the following table.

Category	Character	Description
Alphanumeric data	**X**	Represents one alphanumeric character. If USAGE IS BIT, X represents one bit; the USAGE clause is described in the parameters list.
	(n) An integer in parentheses after an X	Represents *n* repetitions of the alphanumeric character; for example, X(4) is equivalent to XXXX.
Alphabetic data	**A**	Represents one alphabetic character (A-Z).
	(n) An integer in parentheses after an A	Represents *n* repetitions of the alphabetic character
Numeric data	**9**	Represents one numeric character.
	(n) An integer in parentheses after a 9	Represents *n* repetitions of the numeric character.
	V	Represents an assumed decimal point. No more than one V can appear in an element picture. If the V is omitted and the P option (described below) is not used, the assumed decimal point is after the rightmost 9.

Category	Character	Description
	P	Represents an assumed zero. Any number of Ps can appear in the leftmost or the rightmost positions of an element picture. An assumed decimal point is automatically placed before the first P or after the last P. The character P does not occupy a storage position (for example, PP9999 has a data length of 4).
	S	Identifies the number as positive or negative. When used, the S must be the first character in the element picture. When the S is omitted, values for the element description are assumed to be positive.
Numeric- edited data (Includes the numeric data characters described above, along with the editing characters shown at the right)	Z + , B CR - 0 DB * $.	Represent edit symbols used in reporting data; quotation marks are not required. Refer to the appropriate programming language manual for the individual interpretations of these symbols. If the SET OPTIONS statement specifies DECIMAL-POINT IS COMMA, a period (.) is interpreted as an insertion character and a comma (,) is interpreted as a decimal point.

Examples

The following example shows the creation of the record PARTS-RECORD using the RECORD ELEMENT substatement. The statement:

- Defines the record PARTS-RECORD with a language of COBOL, an alternative format of DISPLAY, and an occurrence count of 20,000
- Uses RECORD NAME SYNONYM to define the record synonym ST3PARTS for use with Assembler, and relate the synonym to the file synonym STK3FIL
- Uses RECORD ELEMENT substatements to add:
  - Two elementary elements — PARTNUMBER and DESIGN-DATE
  - Two group elements — INVENTORY-DATA and INVENTORY-DATE

```
add record parts-record
    record storage is database
    language is cobol
    format is display
    occurrences are 20000
    record name synonym is st3parts for file synonym
            stk3fil language is assembler.
    record element partnumber.
    record element design-date.
    record element inventory-data.
    record element inventory-date.
```

The following MODIFY statement:

- Changes the record storage and occurrence specifications
- Inserts a new element, HISTORY, that redefines INVENTORY-DATA
- Identifies synonyms for the various record elements and their subordinate elements

```
modify record parts-record
    record storage is file
    occurrences are 80000.
    record element history line 810 redefines inventory-data
    element name synonym hstry for record st3parts.
    subordinate element discontinue-date
        element name synonym dscdt for record st3parts.
    subordinate element discontinue-month
        element name synonym dscmo for record st3parts.
    subordinate element discontinue-day
        element name synonym dscdy for record st3parts.
    subordinate element discontinue-year
        element name synonym dscyr for record st3parts.
    subordinate element lower-limit
        element name synonym lowlt for record st3parts.
    subordinate element quantity-on-hand
```

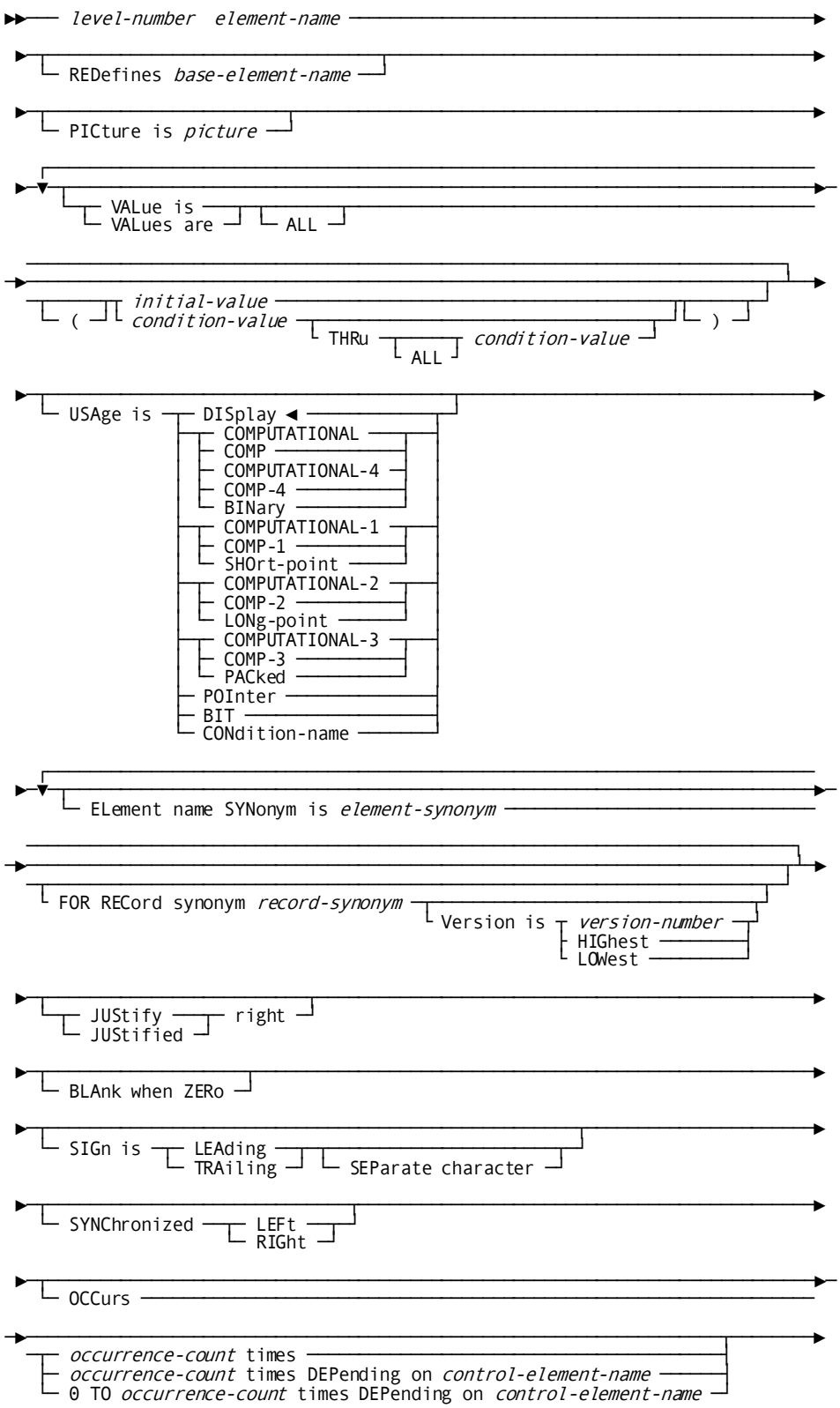
```
    element name synonym qtyhld for record st3parts.
record element partnumber
    element name synonym partno for record st3parts.
record element design-date
    element name synonym dsndt for record st3parts.
record element inventory-data
    element name synonym invdata for record st3parts.
subordinate element in-process
    element name synonym nrproc for record st3parts.
subordinate element quantity1
    element name synonym quone for record st3parts.
subordinate element quantity2
    element name synonym qutwo for record st3parts.
subordinate element quantity3
    element name synonym quthree for record st3parts.
```

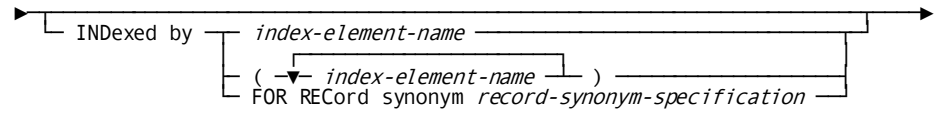
COBOL Substatement

The COBOL substatement creates a record-element structure using an approximation of standard COBOL syntax. Elements named in COBOL substatements need not exist in the dictionary.

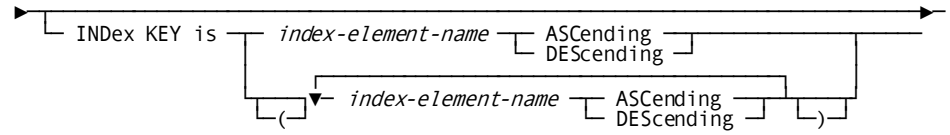
Syntax

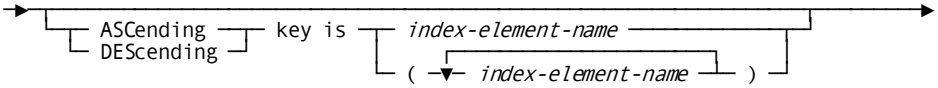
COBOL element substatement

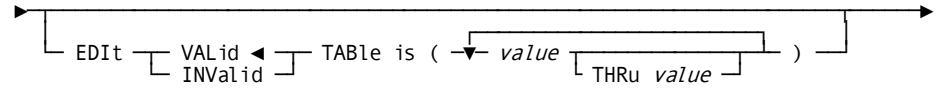

level-number element-name
REDefines base-element-name
PICture is picture
VALue is
VALues are
ALL
initial-value
(
condition-value
THRu
ALL
)
USAge is
DISplay
COMPUTATIONAL
COMP
COMPUTATIONAL-4
COMP-4
BINary
COMPUTATIONAL-1
COMP-1
SHOrt-point
COMPUTATIONAL-2
COMP-2
LONg-point
COMPUTATIONAL-3
COMP-3
PACked
POInter
BIT
CONdition-name
ELement name SYNonym is element-synonym
FOR RECord synonym record-synonym
Version is
version-number
HIGhest
LOWest
JUStify
JUStified
right
BLAnk when ZERo
SIGn is
LEAding
TRAiling
SEParate character
SYNChronized
LEFt
RIGht
OCCurs
occurrence-count times
occurrence-count times DEPending on control-element-name
0 TO occurrence-count times DEPending on control-element-name

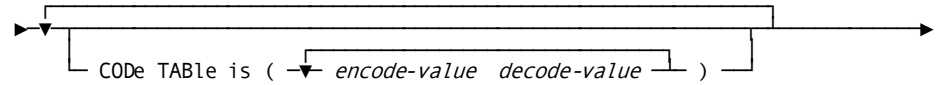


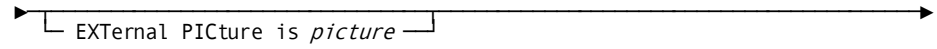
(expanded **record-synonym-specification** syntax follows this syntax diagram)

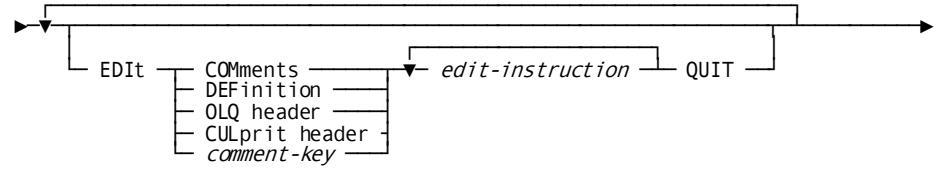


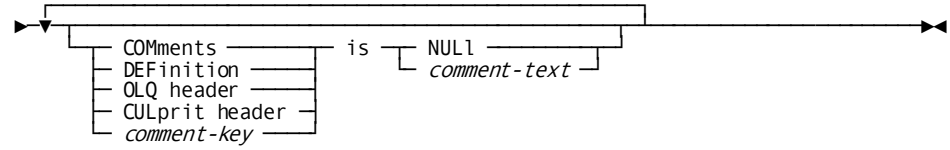




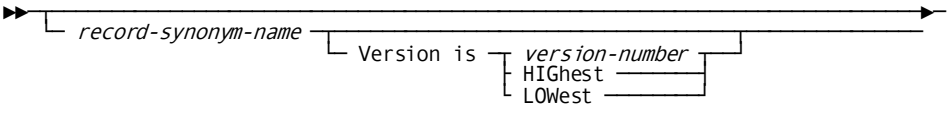


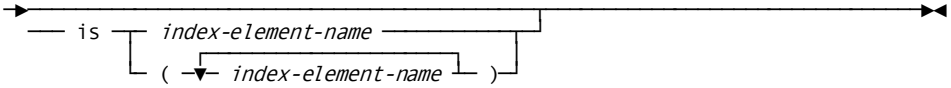






Expansion of *record-synonym-specification*





Parameters

level-number element-name

Specifies the level number and name of the COBOL element. *Level-n* must be an unsigned integer in the range 02 through 49, or 88. Note that the 01-level name is the record name itself or a synonym. *Element-name* must be the 1- through 32-character name of the element. The specified name will be appended with a prefix or suffix if any record synonyms associated with the record have been assigned prefixes or suffixes.

REDefines *base-element-name*

Specifies an alternative description for a previously defined element. The requested element is assigned the same storage space as *base-element-name*. Note that a redefined element cannot be defined with an OCCURS clause; it can, however, be subordinate to an element defined with an OCCURS clause.

PICture is *picture*

Describes the format of the COBOL element. The maximum length of a COBOL element (including its usage) is 32,767 characters. *Picture* must be a 1- through 30-character value specified as shown in the table under **Usage**.

VALue is/VALues are *initial-value/condition-value*

Specifies a value, range of values, or a list of values assigned to a COBOL level-88 condition-name. A list of values must be enclosed in parentheses. Each value in the list must be separated from the next value by a space or comma. A value can be a 1- through 32-character value specified as shown in the list under the bold heading **Usage**.

USAge is

Specifies the method of storing elementary item values at program runtime.

DISplay

Specifies that values are stored one character to a byte according to EBCDIC conventions. DISPLAY is the default.

COMPUTATIONAL

Numeric values are stored in binary format; COMP, COMPUTATIONAL-4, COMP-4, and BINary are synonyms for COMPUTATIONAL.

COMPUTATIONAL-1

Numeric values are stored in internal floating point (short precision) format; COMP-1 and SHOrt-point are synonyms for COMPUTATIONAL-1.

COMPUTATIONAL-2

Numeric values are stored in internal floating point (long precision) format. COMP-2 and LONg-point are synonyms for COMPUTATIONAL-2.

COMPUTATIONAL-3

Numeric values are stored in packed decimal format; COMP-3 and PACked are synonyms for COMPUTATIONAL-3.

BIT

Values are stored one bit at a time as 0s or 1s. BIT cannot be used in COBOL programs.

POInter

Fullword address constant.

CONdition-name

COBOL level-88 values. CONDITION-NAME is assumed if the level number specified for the record element is 88.

ELement name SYNonym is *element-synonym*

Establishes a synonym (alternative name) for the COBOL element. *Element-synonym* is the 1- to 32-character synonym name. The specified name will be appended with a prefix or suffix if a prefix or suffix has been defined for the associated record-synonym name. This clause can be specified once for each record synonym associated with the record.

FOR RECord synonym *record-synonym*

Associates the element synonym with the designated record synonym.

Note: If the FOR RECORD SYNONYM parameter is not specified, the ELEMENT NAME SYNONYM clause applies only to the primary record name.

JUStify right

Specifies that the COBOL element's value is to be right justified at runtime.

BLAnk when ZERo

Specifies that when the COBOL element's values contains all zeroes it is to be changed to spaces at runtime.

SIGn is LEAding/TRAiling

Specifies whether the sign for a numeric field is to appear in the leading or trailing position.

SEParate character

Specifies that the sign is to appear as a separate byte.

SYNChronized LEFt/RIGht

Determines whether boundary alignment is to be defined for the named COBOL element. The correct alignment is determined by the USAGE specification. If the element's usage is COMP or COMP-4, the DDDL compiler issues a warning message if the element is not on the proper boundary alignment. This clause is documentation only, unless the DMLC precompiler is used to copy the record; in this case, the specified boundary alignment will be applied by the COBOL compiler. If the COBOL element is copied into a schema, it causes a critical schema error.

OCCurs

Specifies a record-specific OCCURS clause for the named COBOL element.

***occurrence-count* times**

Specifies the number of times the element can occur within the record. *Occurrence-count* must be an integer in the range 0 through 32,767.

occurrence-count* times/0 TO *occurrence-count* times DEPending on *control-element-name

Defines a control element within the record that determines the actual number of times the COBOL element will occur. *Occurrence-count* must be an integer in the range 1 through 32,767. *Control-element-name* specifies a previously defined field in the record; this field must be a halfword or fullword binary item if the record is to be used in a schema or by CA ADS.

INDexed by

Specifies one or more indexes for a multiply-occurring element or for a record synonym associated with the named COBOL element.

The INDEXED BY clause can be specified once for each record element or subordinate element and record synonym; the index element name is appended with a prefix or suffix as appropriate. The INDEXED BY clause applies only to records used in COBOL programs and should be specified only when the named element definition contain on OCCURS or OCCURS DEPENDING ON clause. The specified index name is copied into the program's DATA DIVISION by the DMLC precompiler as part of the COPY IDMS function.

index-element-name

Specifies an INDEXED BY name for the named COBOL element. The specified value must be a 1- through 32-character name that does not duplicate an existing element or element-synonym name.

FOR REcord synonym *record-synonym* is *index-element-name*

Specifies an INDEXED BY name for a record synonym associated with the record element. *Index-element-name* is a 1- through 32-character name that cannot duplicate an element or element-synonym name in the record.

INDex KEY is

Specifies one or more index keys through one of the following options; note that each option is functionally the same.

INDex KEY is *index-element-name* ASCending/DEScending

Specifies a record-specific index key for the record element or subordinate record element. *Index-element-name* identifies an elementary element that is subordinate to the associated element and must be the primary name of the record element. ASCENDING or DESCENDING specifies the manner in which the subordinate element values will be ordered within the multiply-occurring group.

ASCending/DEScending key is *index-element-name*

Specifies one or more record-specific index keys for the multiply-occurring group element or subordinate element and defines the manner in which subordinate element values will be ordered within the multiply-occurring group. *Index-element-name* must be the primary name of an element that is subordinate to the named group element. The named element and the ASCENDING/DESCENDING specification govern the ordering of values of the subordinate element within the multiply-occurring group. Each index element name is prefixed or suffixed for each record synonym associated with the element.

EDIt VALid/INValid TABle is *value* THRu *value*

Specifies a single value or range of values to be inserted in the edit table (for DC/UCF tables only). Each value can have a maximum size of 34 characters and must be enclosed in parentheses; for example:

```
('A' 'B' 'D' 'F' 'R' THRU 'T' 'V' 'X' THRU 'Z')
```

VALid/INValid

Identifies the supplied list as a list of valid values or a list of invalid values. The default is VALID.

CODe TABle is *encode-value decode-value*

Specifies values to be inserted in the table in pairs (for DC/UCF tables only). The first or *encoded* value can have a maximum size of 34 characters; the second or *decoded* value can have a maximum of 64 characters. Null values ('') and the keywords NOT FOUND are also valid. The specified values must be enclosed in parentheses. For example:

```
('CA' 'CALIFORNIA' 'NY' 'NEW YORK')
```

For detailed information about defining code tables, refer to the *CA IDMS Mapping Facility Guide* manual.

EXTernal PICture is *picture*

Defines the display format for record-element data. The picture is available to all map fields that use the record element. For more information about external pictures, refer to the *CA IDMS Mapping Facility Guide* manual.

Usage

COBOL substatement considerations

The following considerations apply to the COBOL substatement:

- The named record element is validated against elements in the dictionary, as follows:

- If identical primary or synonym names are found, the DDDL compiler examines each element for identical PICTURE, USAGE, BLANK WHEN ZERO, JUSTIFY, and SIGN specifications, and for identical group structures (if any); level-88 elements are also examined. If the definition of the named COBOL element matches the definition of an existing element, the DDDL compiler copies the definition of the existing element into the named record. If the two elements have matching names only, a new element is added to the dictionary and is automatically assigned the highest existing version number plus 1.

  Note: Differences in the entry formats for COBOL PIC are resolved (for example, pic x(2) is recognized as equivalent to pic xx).

- If identical primary or synonym names are not found, a new element is added to the dictionary and automatically assigned a version number of 1.

- Elementary fillers are treated the same way as elementary elements. If the PICTURE, USAGE, BLANK WHEN ZERO, JUSTIFY, and SIGN specifications match those associated with an existing filler, that filler is used; otherwise, a new filler is added to the dictionary. Note that when a filler with a VALUE clause is copied into the record-element structure, the value itself is not copied; rather, a value of NO VALUES is assigned to the filler.
- Fillers can be group elements; in this case, the rules for forming group elements apply.
- COBOL substatements are used to define record elements as tables that are used by the DC/UCF mapping facility for automatic editing and error handling. Tables defined by means of the COBOL substatement are called *built-in tables*.

  The TABLE statement (described under TABLE (see page 341), later in this chapter) is used to define stand-alone tables. For a complete description of built-in and stand-alone tables, refer to the *CA IDMS Mapping Facility Guide*.

Specifying a **picture** variable

Picture must be a 1- through 30-character value that describes the types of data shown in the following table.

Category	Character	Description
Alphanumeric data	**X**	Represents one alphanumeric character. If USAGE IS BIT, X represents one bit; the USAGE clause is described in the parameters list.
	(n) An integer in parentheses after an X	Represents *n* repetitions of the alphanumeric character; for example, X(4) is equivalent to XXXX.
Alphabetic data	**A**	Represents one alphabetic character (A-Z).

Category	Character	Description
	(n) An integer in parentheses after an A	Represents *n* repetitions of the alphabetic character
Numeric data	**9**	Represents one numeric character.
	(n) An integer in parentheses after a 9	Represents *n* repetitions of the numeric character.
	V	Represents an assumed decimal point. No more than one V can appear in an element picture. If the V is omitted and the P option (described below) is not used, the assumed decimal point is after the rightmost 9.
	P	Represents an assumed zero. Any number of Ps can appear in the leftmost or the rightmost positions of an element picture. An assumed decimal point is automatically placed before the first P or after the last P. The character P does not occupy a storage position (for example, PP9999 has a data length of 4).
	S	Identifies the number as positive or negative. When used, the S must be the first character in the element picture. When the S is omitted, values for the element description are assumed to be positive.
Numeric- edited data (Includes the numeric data characters described above, along with the editing characters shown at the right)	Z + , B CR - 0 DB * $.	Represent edit symbols used in reporting data; quotation marks are not required. Refer to the appropriate programming language manual for the individual interpretations of these symbols. If the SET OPTIONS statement specifies DECIMAL-POINT IS COMMA, a period (.) is interpreted as an insertion character and a comma (,) is interpreted as a decimal point.

Valid values for the VALUE clause

Valid types of values for the VALUE clause are as follows:

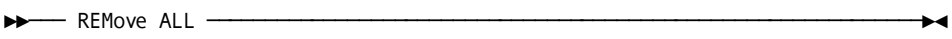
- **Figurative constant**— For alphanumeric and numeric data items, ZERO, ZEROS, ZEROES. For alphanumeric data items only: SPACE, SPACES, HIGH-VALUE, HIGH-VALUES, LOW-VALUE, LOW-VALUES.
- **Nonnumeric literal**— For alphanumeric data items only, a string of characters enclosed in single quotation marks. The character string must not exceed the size specified in the element's PICTURE clause.
- **Numeric literal**— For numeric items only, a string of numeric characters, optionally preceded by a plus (default value) or minus sign and optionally containing a decimal point. The numeric string must not exceed the size of the data item as defined in the PICTURE clause.

REMOVE ALL Substatement

The REMOVE ALL substatement is used in conjunction with an ADD RECORD or MODIFY RECORD statement to delete the record-element structure associated with the named record. You can create a new record-element structure by coding RECORD ELEMENT substatements immediately following the REMOVE ALL substatement; the rules for ADD RECORD apply. REMOVE ALL also removes any IDD-built subschema views (see VIEW ID Substatement (see page 332) below).

Syntax

REMOVE ALL substatement



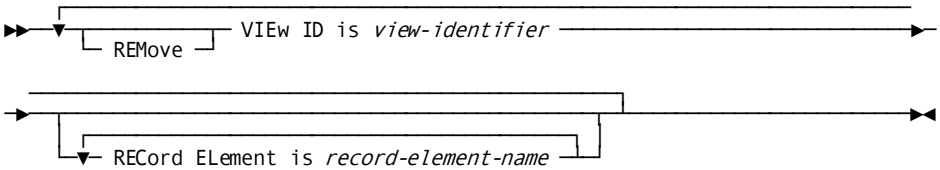
VIEW ID Substatement

The VIEW ID substatement establishes or removes a view of record elements. Once established, this view can be copied into one or more CA IDMS subschemas. Before issuing a VIEW ID substatement, you should ensure that all record elements identified in the view are present in the record. The following rules apply to the VIEW ID substatement:

- VIEW ID must be the last substatement coded in the RECORD statement.
- Record elements named in the view must be identified by their primary names.
- All record elements named in the view must be at the same level.
- A record element that is subordinate to an OCCURS clause cannot be included in the view.
- A REDEFINES element cannot be included in the view. If a redefined element is included, all redefining elements are automatically included in the view.
- An OCCURS DEPENDING ON record element must be the last record element in the view.
- Bit fields cannot be named.
- A record element can be named only once.
- If a group element is included, all subordinate elements are automatically included in the view. The order of group/subordinate record elements is retained.

Syntax

VIEW ID substatement



Parameters

VIEw ID is *view-identifier*

Identifies a list of record elements that is to comprise a view. *View-identifier* must be a 1- through 32-character alphanumeric value. The VIEW ID substatement can appear any number of times in one RECORD statement. If the optional REMOVE parameter is specified, the named view is deleted.

RECord ELement is *record-element-name*

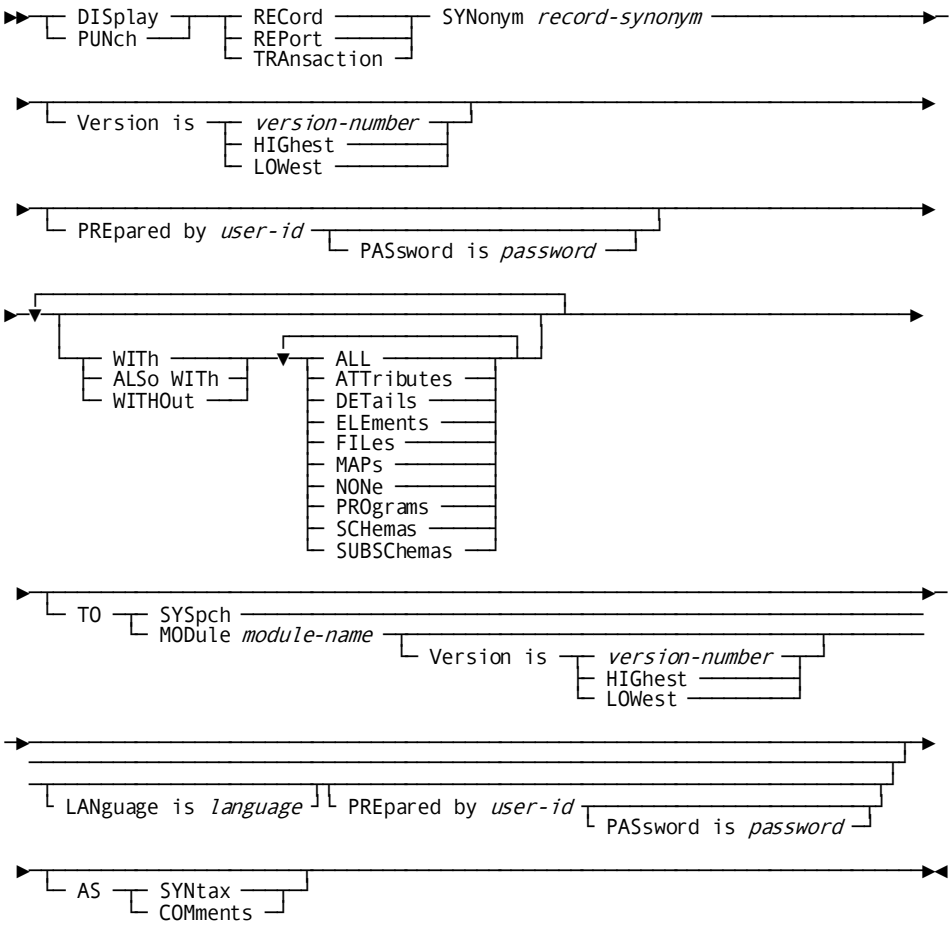
Identifies a record element to be added to the view. *Record-element-name* must be an element that exists in the named record-element structure. This clause can appear any number of times in one VIEW ID substatement.

RECORD SYNONYM

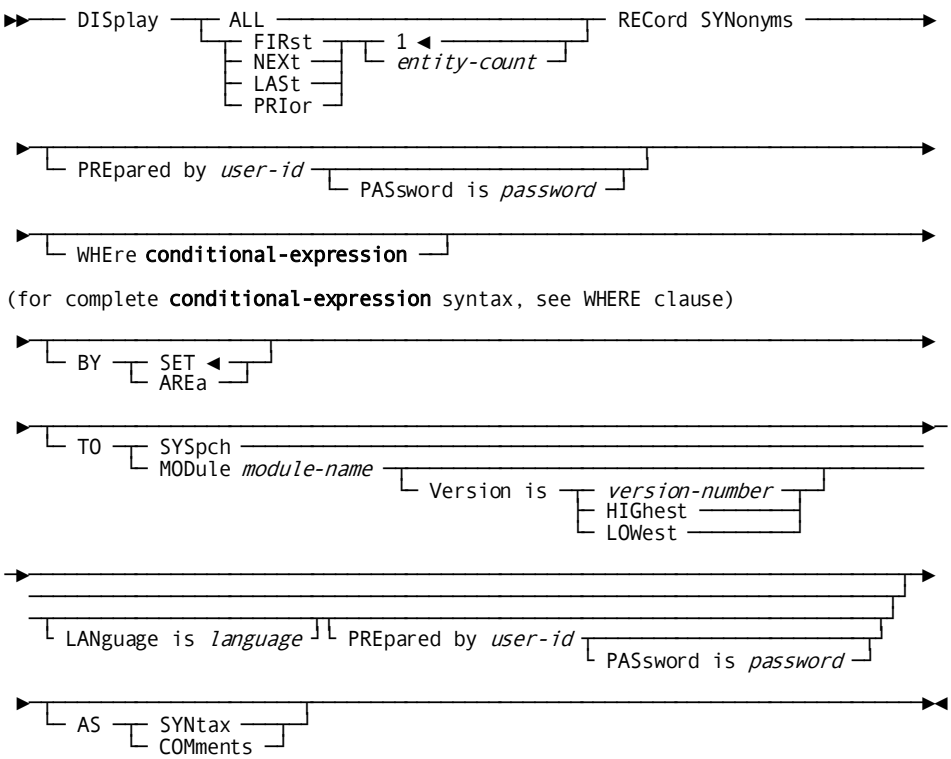
You can display or punch selected record synonyms by using the DISPLAY/PUNCH RECORD SYNONYM statement.

Syntax

DISPLAY/PUNCH RECORD SYNONYM statement (for a single synonym)



DISPLAY/PUNCH RECORD SYNONYM statement (for multiple synonyms)



SYSTEM (SUBSYSTEM)

SYSTEM statements are used to document automated or manual data processing systems. Optional clauses relate systems to users and to other systems and support attribute/entity relationships and documentation entries.

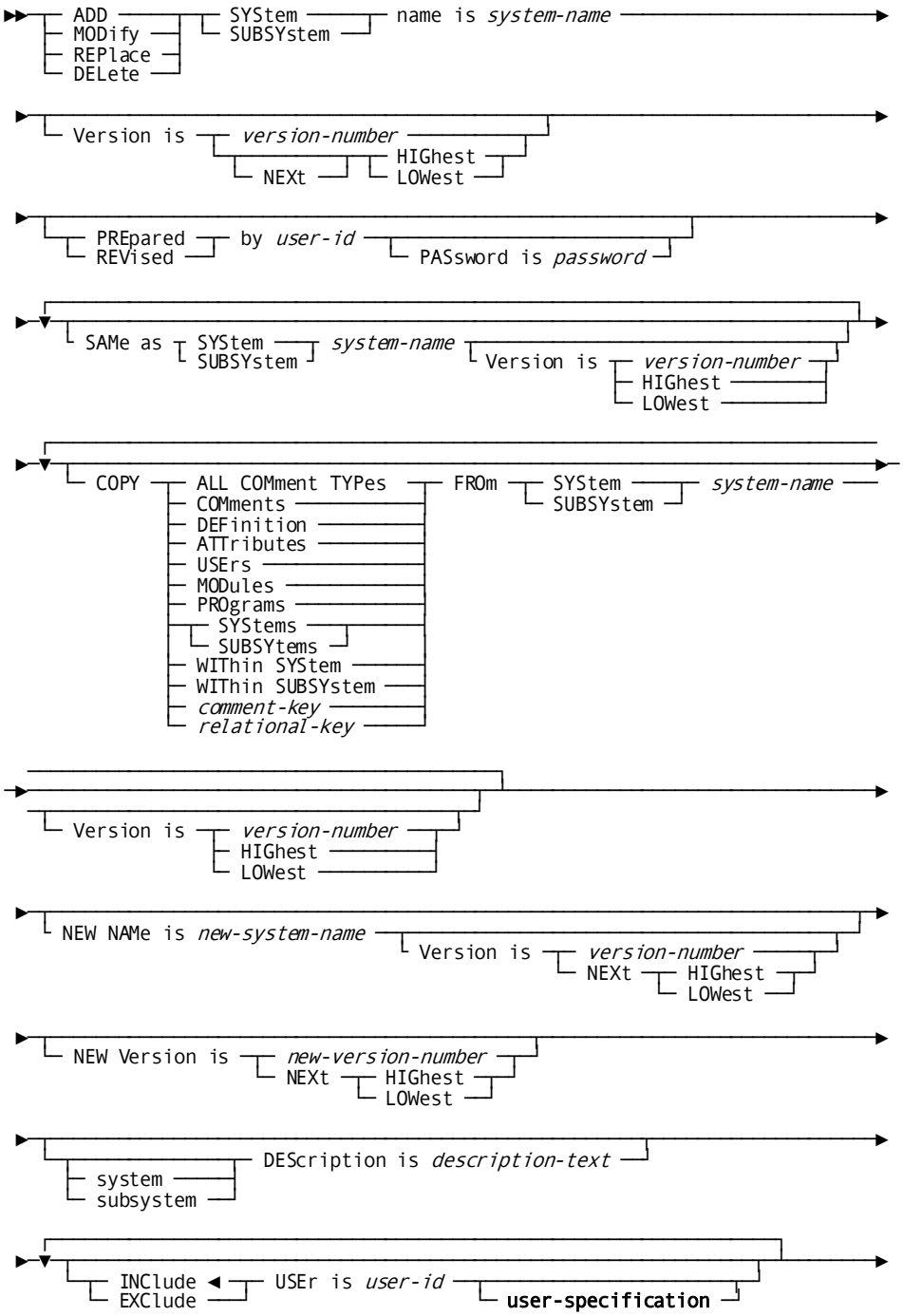
Note: It is recommended that you maintain SYSTEM definitions using the system generation compiler, *not* the DDDL compiler. If a system generation component is processed by the DDDL compiler, only dictionary security is checked, *not* system generation security. For more information on using the system generation compiler, refer to *CA IDMS System Generation Guide*.

Note: The keyword SUBSYSTEM can be used interchangeably with the keyword SYSTEM.

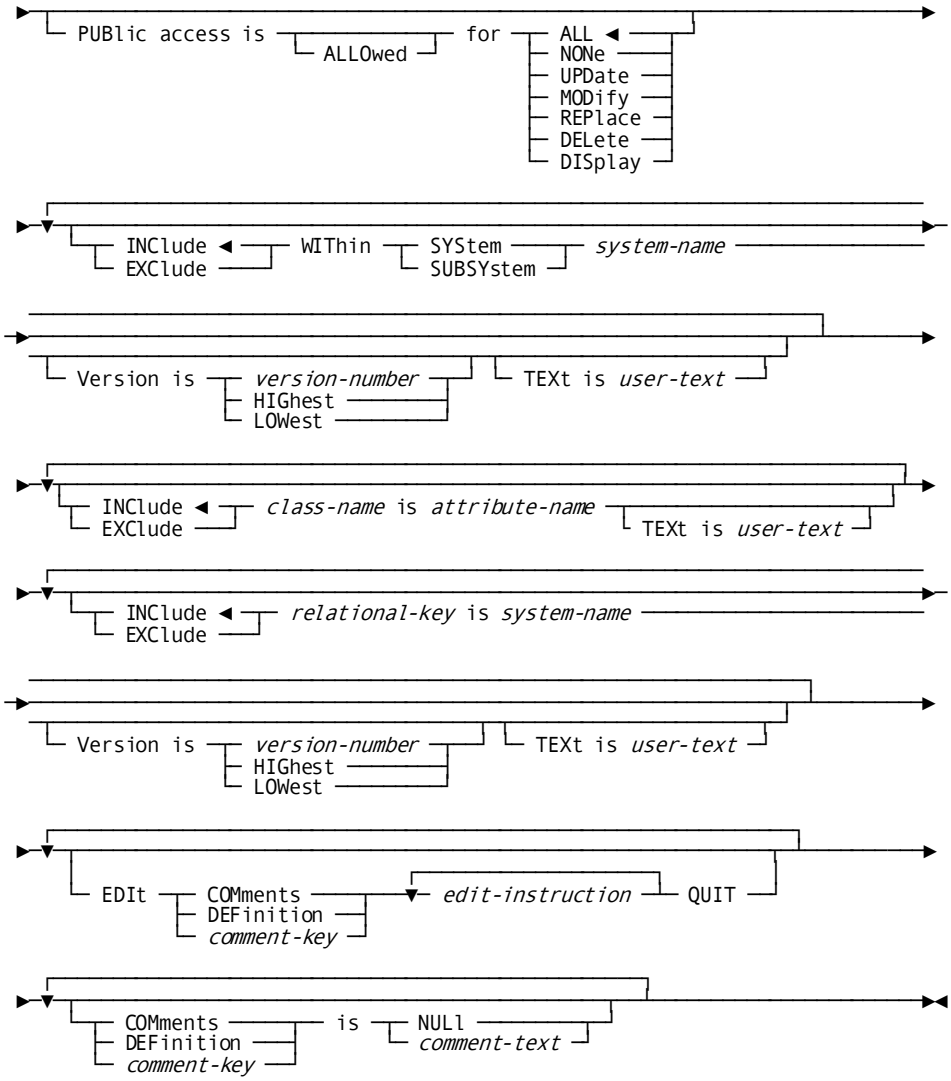
If the SET OPTIONS statement specifies SECURITY FOR IDD IS ON, the user must be assigned the proper authority to issue SYSTEM statements. Note that DDDL statements cannot be used to delete systems built by the system generation compiler.

Syntax

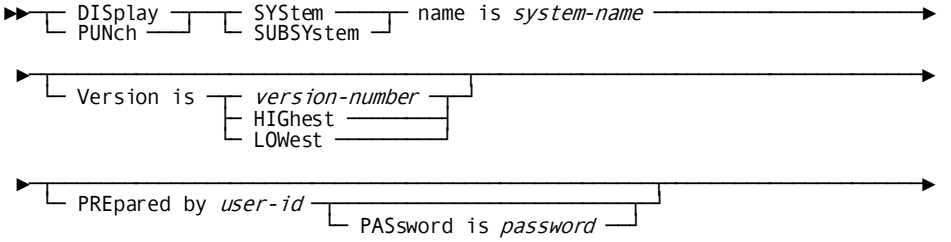
SYSTEM (SUBSYSTEM) statement

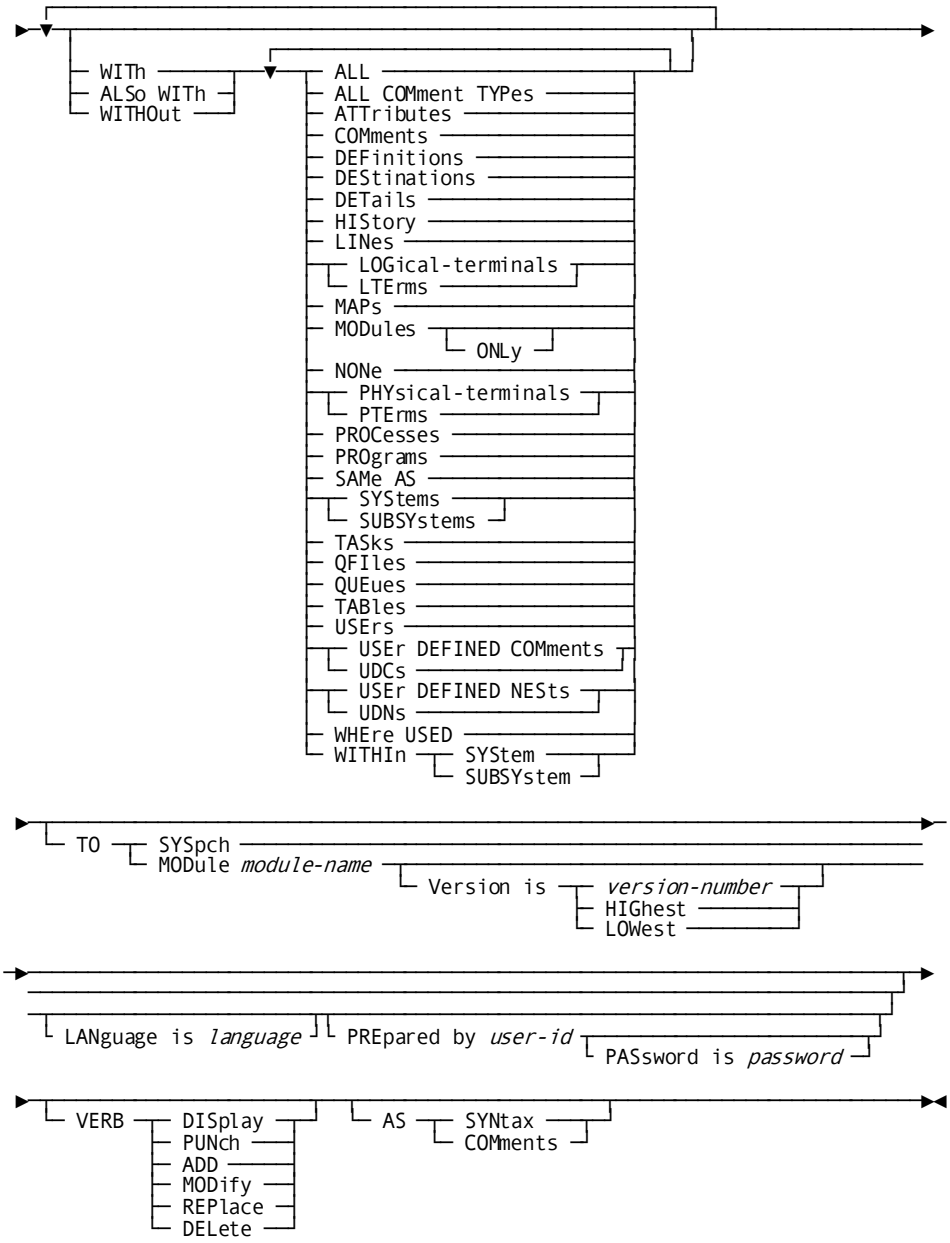


(for complete **user-specification** syntax, see USER clause)

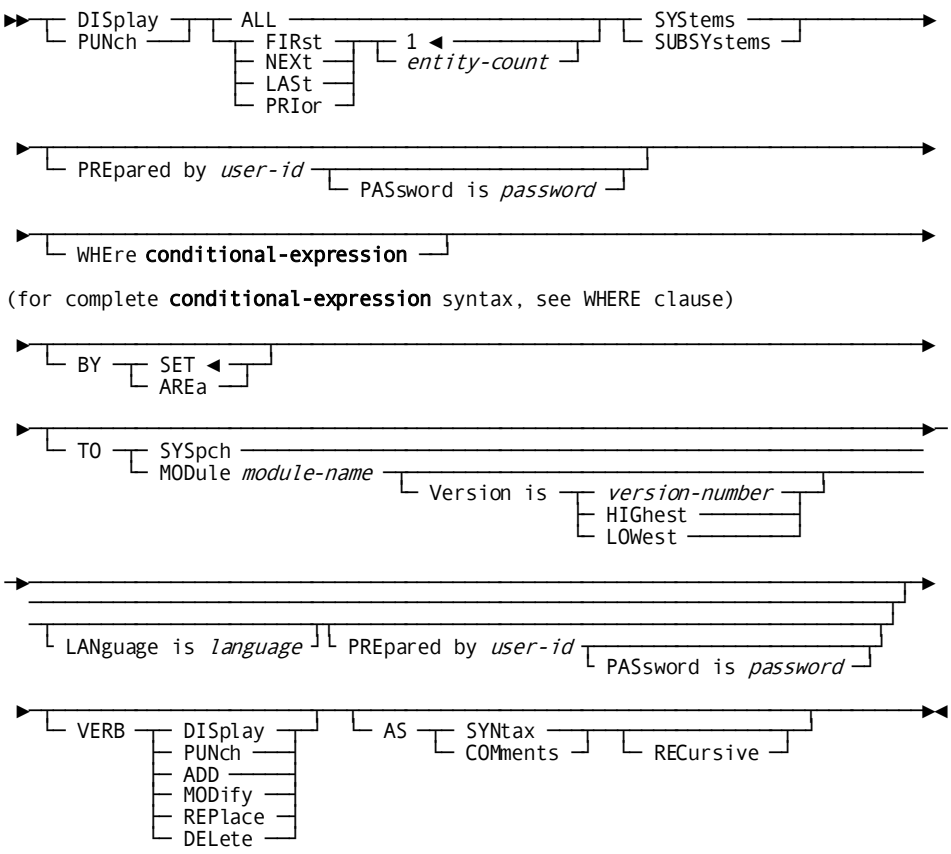


DISPLAY/PUNCH SYSTEM (SUBSYSTEM) statement (for a single system)




WITh
ALSo WITh
WITHOut
ALL
ALL COMment TYPes
ATTributes
COMments
DEFinitions
DEStinations
DETails
HIStory
LINes
LOGical-terminals
LTErms
MAPs
MODules
ONLy
NONe
PHYsical-terminals
PTErms
PROCesses
PROgrams
SAMe AS
SYStems
SUBSYstems
TASks
QFIles
QUEues
TABles
USErs
USEr DEFINED COMments
UDCs
USEr DEFINED NESts
UDNs
WHEre USED
WITHIn
SYStem
SUBSYstem
TO
SYSpch
MODule module-name
Version is
version-number
HIGhest
LOWest
LANguage is language
PREpared by user-id
PASsword is password
VERB
DISplay
PUNch
ADD
MODify
REPlace
DELete
AS
SYNtax
COMments

DISPLAY/PUNCH SYSTEM (SUBSYSTEM) statement (for multiple systems)



Parameters

SYStem/SUBSYstem name is *system-name*

Identifies a new system to be established in the dictionary, or an existing system to be modified, replaced, deleted, displayed, or punched. *System-name* must be a 1- through 32-character alphanumeric value.

NEW NAMe is *new-system-name*

Specifies a new name for the requested system. This clause changes the name of the requested system only; it does not alter or delete any relationships in which the system participates. Subsequent references to the system must specify the new name. The concatenation of the new system name and version number must not duplicate that of an existing system occurrence. The NEW NAME clause is not valid for systems created by the CA IDMS/DC system generation compiler.

NEW Version is *new-version-number*/NEXt HIGhest/NEXt LOWest

Specifies a new version number for the named system. The combination of system name and new version number must not duplicate that of an existing system occurrence. The NEW VERSION clause is not valid for systems created by the CA IDMS/DC system generation compiler.

WIThin SYStem/SUBSYstem *system-name*

Associates (INCLUDE) the named system with or disassociates (EXCLUDE) it from the system/subsystem identified by the 1- through 32-character *system-name*.

WITh/ALSo WITh/WITHOut

Includes or excludes the specified options when the named system is displayed or punched. Detailed information for each DISPLAY/PUNCH option is under SET OPTIONS Syntax (see page 36). The options that are listed below present special considerations for this entity type.

DETails

Includes the DESCRIPTION specification.

SYStems (SUBSYstems)

Includes WITHIN SYSTEM specifications and user-defined nests.

Usage

If you specify REPLACE

If the REPLACE verb is specified, the DDDL compiler initializes to defaults and/or excludes the following:

- DESCRIPTION
- USER REGISTERED FOR
- PUBLIC ACCESS
- COMMENTS/DEFINITIONS/*comment-key*
- WITHIN SYSTEM/SUBSYSTEM
- ATTRIBUTES

The following relationships are not affected:

- Users assigned access to the named system
- CA IDMS/DC definitions, destinations, lines, logical terminals, maps, programs, physical terminals, queues, modules, tasks, and systems in which the named system participates as a component

Example

The following ADD statement defines the system INVENTORY, relates that system to two existing users and an existing system, and establishes two documentation relationships by means of a class/attribute structure and a relational key.

```
add system inventory
    prepared by dba password is 'ice 9'
    system description is 'present inventory system'
    user is accounting
    user is receiving
    within system order-control
    status is production
    'similar system' is back-order.
```

This second ADD statement defines version 2 of the same system by copying the definition of version 1 and removing copied options that are not applicable to the proposed system. Note that the DDDL compiler generates a PREPARED BY entry for the second system only if a SET OPTIONS statement has provided a default PREPARED BY specification; use of the SAME AS option does not generate a PREPARED BY or REVISED BY entry.

```
add system inventory version is 2
    same as system inventory
    exclude status is production
    status is design
    exclude 'similar system' is back-order
    exclude within system order-control
    system description is 'proposed inventory system'.
```

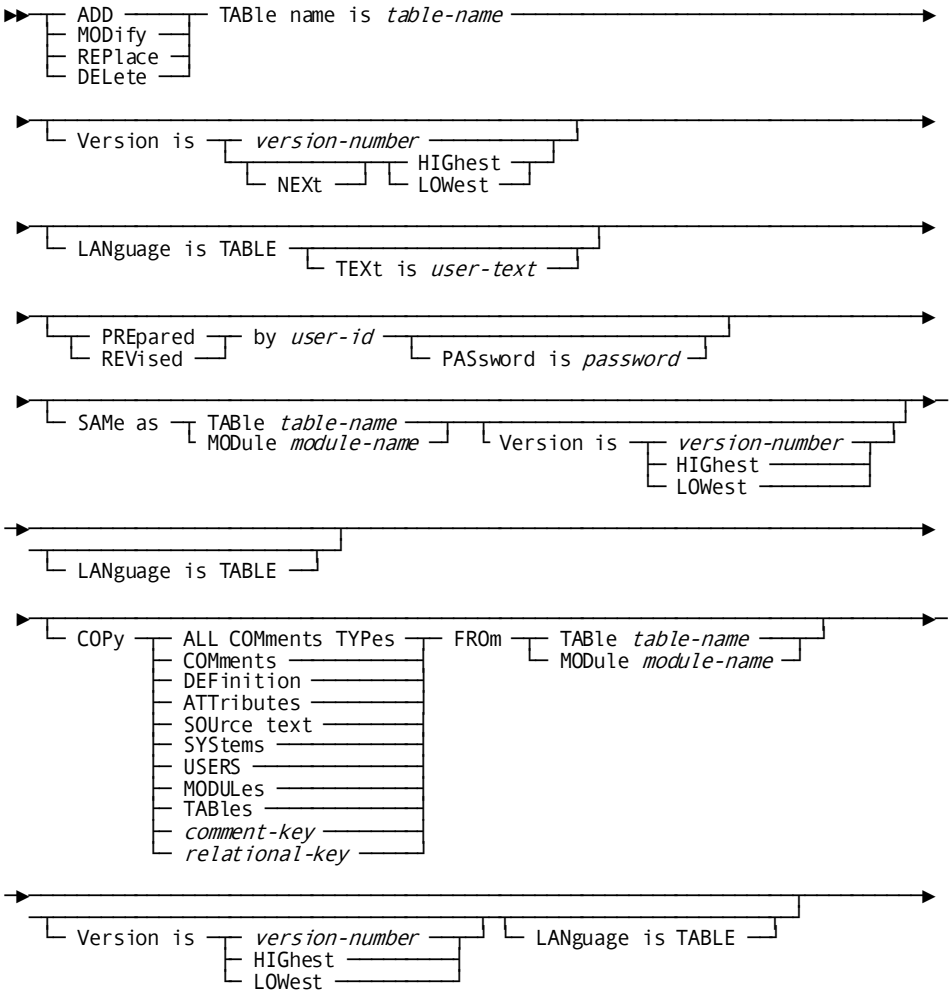
TABLE

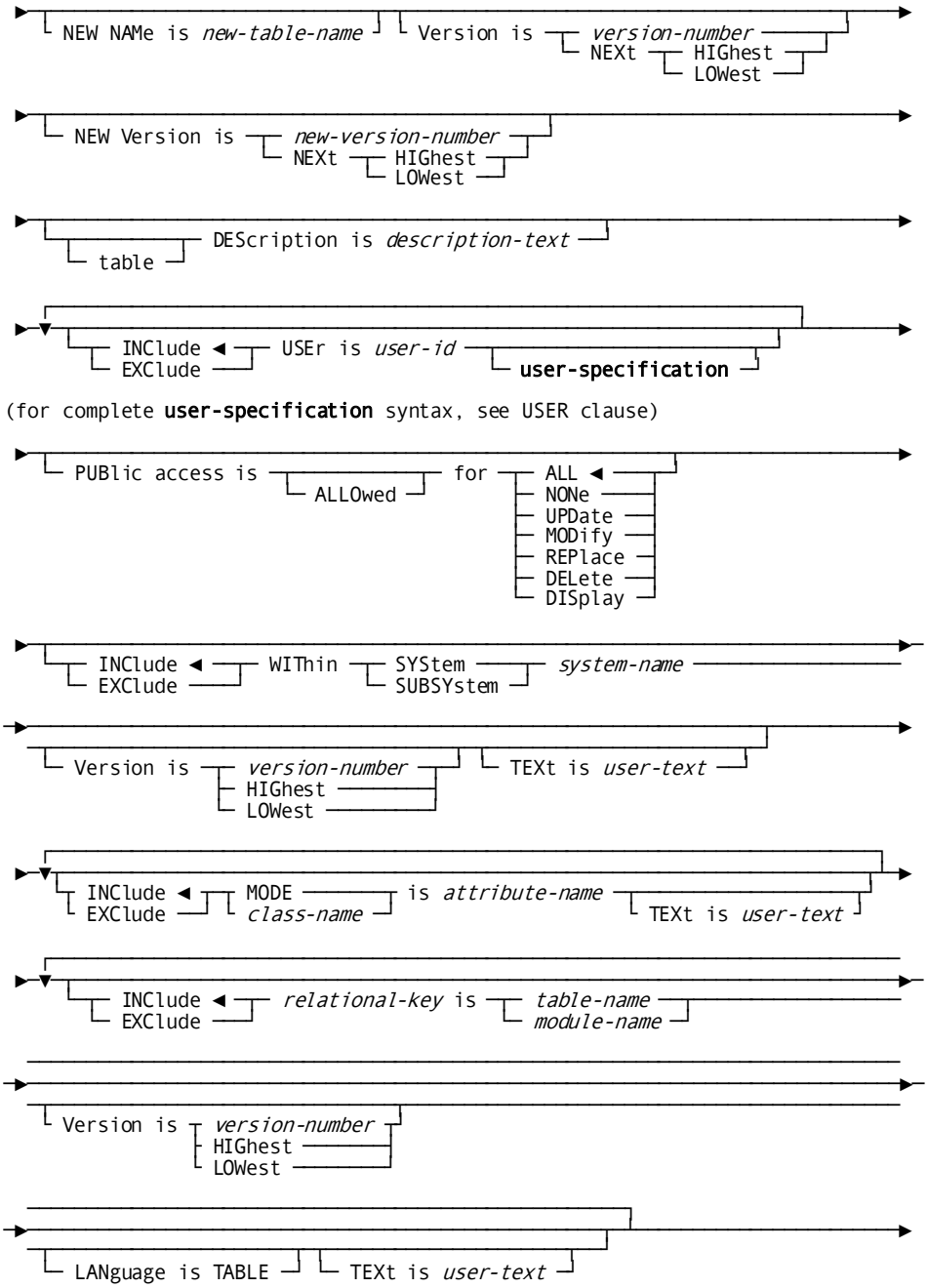
TABLE statements are used to document edit and code tables. Tables are used by the CA IDMS mapping facility for automatic editing and error handling. Optional TABLE statement clauses relate tables to maps, users, systems, modules, and other tables; establish attribute/entity relationships; and maintain documentation entries.

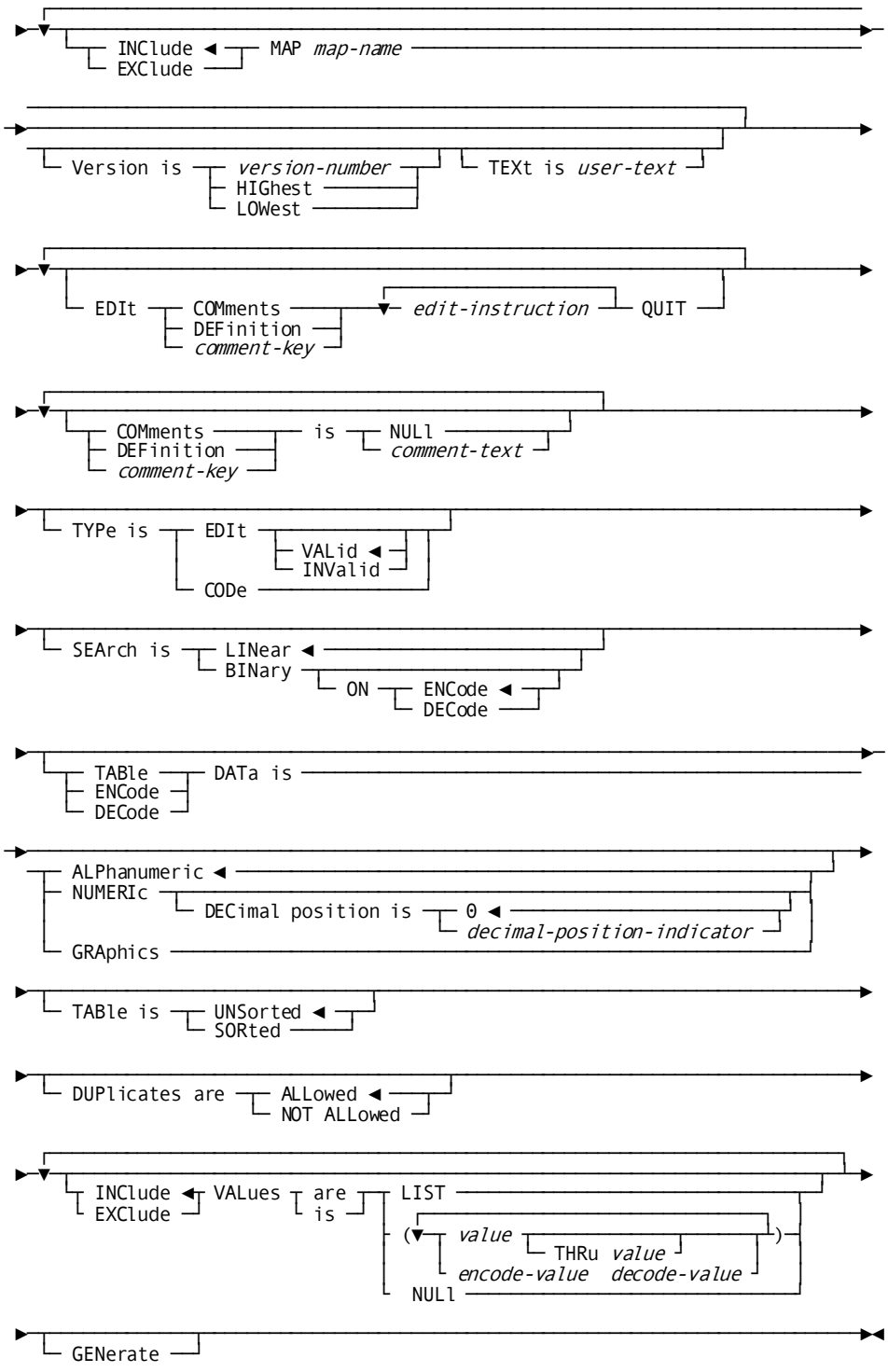
If the SET OPTIONS statement specifies SECURITY FOR IDD IS ON, the user must be assigned the proper authority to issue TABLE statements.

Syntax

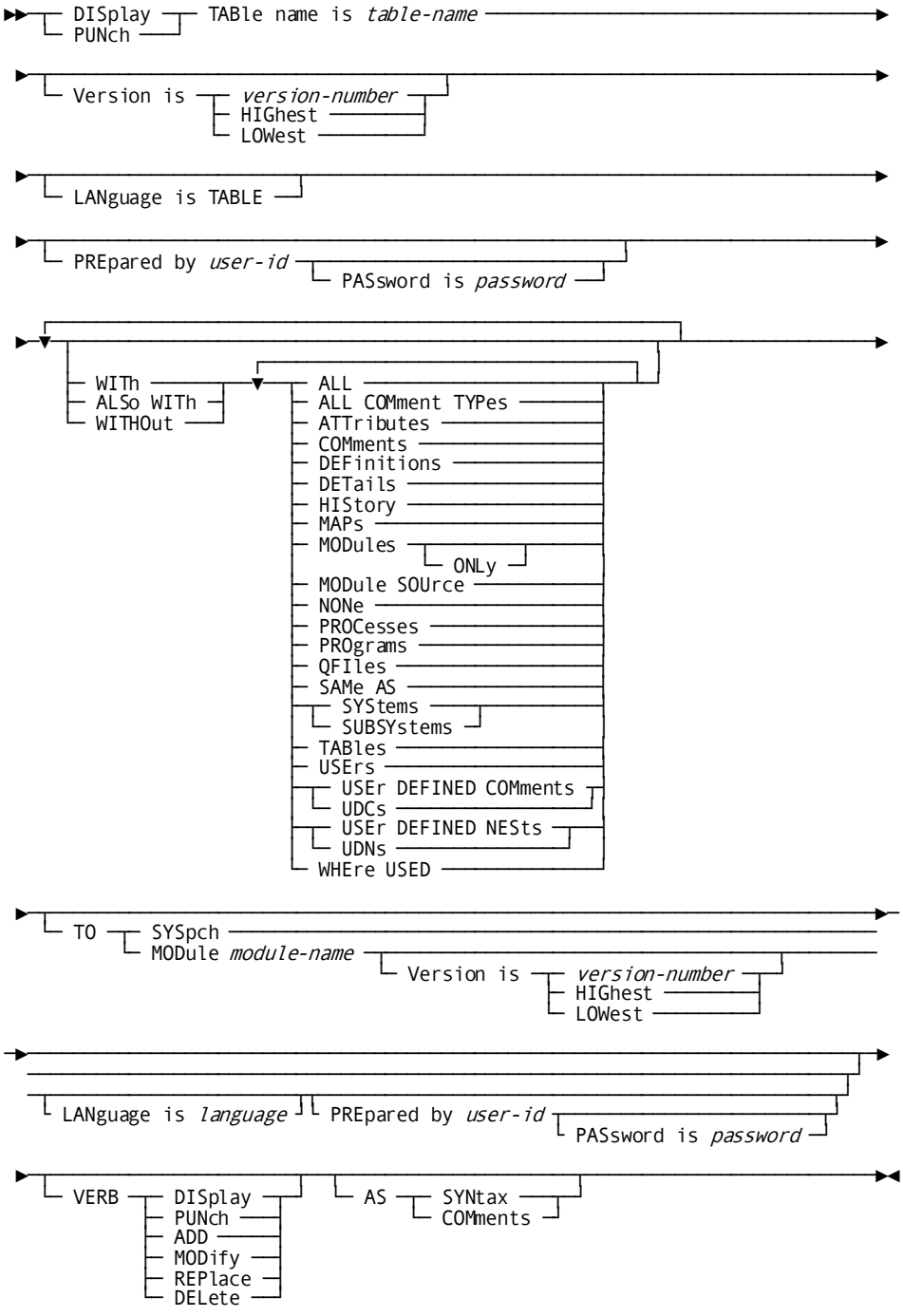
TABLE statement



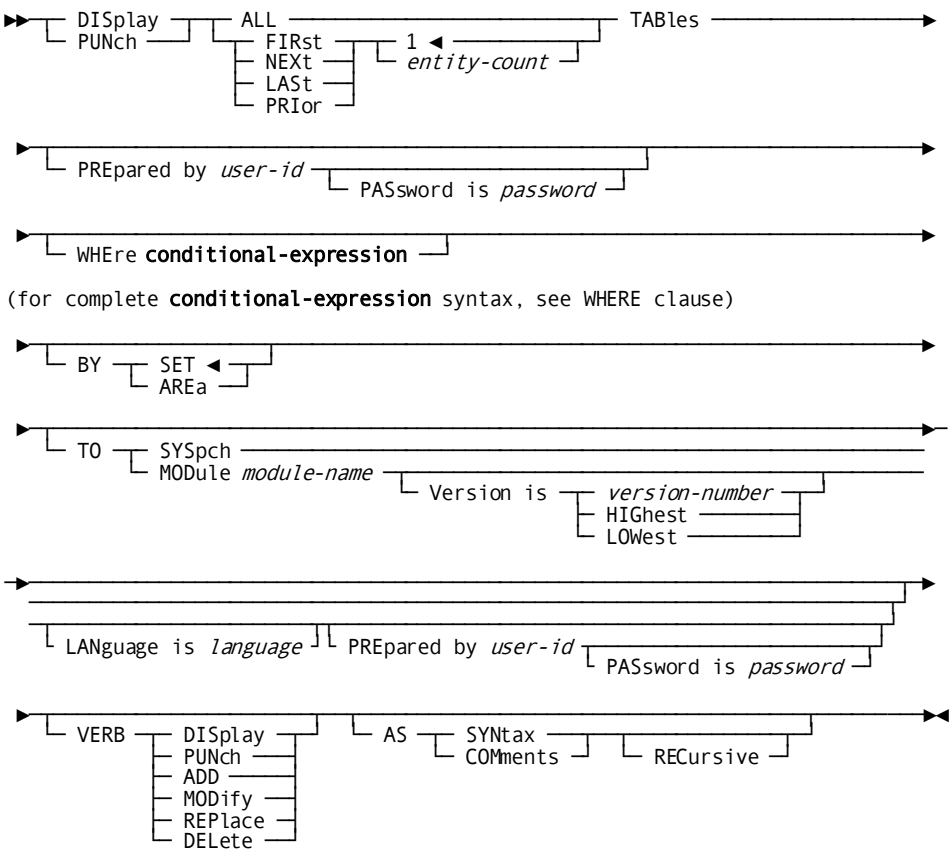

NEW NAMe is new-table-name
Version is version-number
NEXt
HIGhest
LOWest
NEW Version is new-version-number
NEXt
HIGhest
LOWest
table
DEScription is description-text
INClude
EXClude
USEr is user-id
user-specification
(for complete user-specification syntax, see USER clause)
PUBlic access is
ALLOwed
for
ALL
NONe
UPDate
MODify
REPlace
DELete
DISplay
INClude
EXClude
WIThin
SYStem
SUBSYstem
system-name
Version is
version-number
HIGhest
LOWest
TEXt is user-text
INClude
EXClude
MODE
class-name
is attribute-name
TEXt is user-text
INClude
EXClude
relational-key is
table-name
module-name
Version is
version-number
HIGhest
LOWest
LANguage is TABLE
TEXt is user-text


INClude
EXClude
MAP map-name
Version is
version-number
HIGhest
LOWest
TEXt is user-text
EDIt
COMments
DEFinition
comment-key
edit-instruction
QUIT
COMments
DEFinition
comment-key
is
NULl
comment-text
TYPe is
EDIt
VALid
INValid
CODe
SEArch is
LINear
BINary
ON
ENCode
DECode
TABle
ENCode
DECode
DATa is
ALPhanumeric
NUMERIc
DECimal position is
0
decimal-position-indicator
GRAphics
TABle is
UNSorted
SORted
DUPlicates are
ALLowed
NOT ALLowed
INClude
EXClude
VALues
are
is
LIST
value
THRu value
encode-value decode-value
NULl
GENerate

DISPLAY/PUNCH TABLE (for a single table)



DISPLAY/PUNCH TABLE (for multiple tables)



Parameters

TABle name is *table-name*

Identifies a new table to be established in the dictionary, or an existing occurrence to be modified, replaced, deleted, displayed, or punched. *Table-name* must be a 1- through 8-character alphanumeric value. The specified name must not duplicate the name of an existing program, map, subschema, or CA ADS dialog.

LANguage is

Qualifies the requested table/module with a language. The LANGUAGE specification uniquely identifies two modules with the same name and version and is used by the DML precompilers when modules are used in programs.

TABLE

When used with the LANGUAGE IS clause, supplies the appropriate language, TABLE, automatically.

SAMe as TABle/MODule *table-name/module-name*

Copies all entries associated with the named table or module, except the name and LANGUAGE specifications. The table/module to be copied must have the language TABLE.

COPy *entity-option* FROm *entity-type-name entity-occurrence-name*

Copies selected options from an entity-occurrence definition and merges the copied options into this definition. TABLEs can copy only from other modules with a language of TABLE.

NEW NAMe is *new-table-name*

Specifies a new name for the requested table. This clause changes only the name of the table; it does not alter or delete any previously defined relationships in which the table participates. Subsequent references to the table must specify the new name. *New-table-name* must be a 1- through 8-character alphanumeric value. The concatenation of the new table name, version number, and language must not duplicate that of an established table or module occurrence.

NEW Version is *new-version*/NEXt HIGhest/NEXt LOWest

Specifies a new version number for the named table. The combination of the table name, new version number, and language qualification must not duplicate that of an existing table or module.

WIThin SYStem/SUBSYstem *system-name*

Associates the requested table with (INCLUDE) or disassociates it from (EXCLUDE) the specified system or subsystem. *System-name* must reference an existing system or subsystem.

relational-key* is *table-name/module-name

Associates (INCLUDE) the table with or disassociates it from (EXCLUDE) another table or module by means of the named relational key. If the tables and/or modules being related have the same name and version but different languages, or if the related module has a version of HIGHEST or LOWEST and is qualified by language, the LANGUAGE parameter must be specified. For a complete description of defining and using relational keys, see Relational Keys (see page 104).

MAP *map-name*

Associates (INCLUDE) the table with or disassociates (EXCLUDE) it from a map. *Map-name* must reference an existing map.

TYPe is

Specifies the table type. This clause is required for ADD operations.

EDIt

Defines a table that contains a list of values or ranges of values; a data field will be checked against the table.

VALid/INValid

Specifies whether the list contains valid or invalid values; VALID is the default.

CODe

Defines a table that translates internal codes in a record to external report values (decoding) or maps external values back to internal record codes (encoding).

SEArch is

Specifies the method by which the table is to be searched.

LINear

Starts the search at the beginning of the table and proceeds line by line until the specified value is found. LINEAR is the default.

BINary

Starts the search in the middle of the table and halves the table each time a comparison is made until the specified value is found. Edit tables to be searched by the binary method can include only single values.

ON ENCode/DECode

Specifies whether the binary search is to be performed on encoded or decoded table values. (The option is for code tables only.) The default is ENCODE.

TABle/ENCode/DECode DATa is

Specifies the type of table. DECODE allows different types of encode and decode values.

ALPhanumeric

Specifies that the corresponding table values in the value list are one of the following types of literals:

- A literal that contains only EBCDIC characters
- A literal that contains only DBCS characters enclosed in the shift codes
- A literal that contains a combination of characters with the DBCS characters enclosed in shift codes

The character strings must be enclosed in the site-specific quote character. ALPHANUMERIC is the default.

NUMeric

Specifies numeric data.

GRAphics

Specifies that the corresponding table values in the value list are graphic (G-) literals. You use G-literals when an element must be interpreted without the shift codes. The external picture of the data element must be X, unless the table is to be used with mapping. In this case, the external picture of the data element must be G.

More information: For more information about using graphics literals, see Double-Byte Character Set (DBCS) Strings.

DECimal position is

Specifies the position of the decimal point (NUMERIC option only). Note that this is an assumed decimal position; no decimal point appears in the values.

TABle is

Specifies whether the table is to be maintained in the dictionary as a sorted table.

UNSorted

Sorts table values at runtime in the order in which they are placed in the dictionary. UNSORTED is the default.

SORted

Sorts table values alphabetically or numerically as they are added to the table.

Note: A binary searched table can be stored with the UNSORTED attribute; however, the table is automatically sorted when it is generated.

DUPlicates are ALLowed/NOT ALLowed

Specifies whether duplicate values can be included in sorted tables; ALLOWED is the default. Note that DUPLICATES ARE NOT ALLOWED must be specified for binary searched tables.

VALues are

Specifies whether table values are to be listed, inserted, or removed.

LIST

Lists the table values or pairs of values (code tables only) stored in the dictionary.

value* THRu *value

Inserts single values, ranges of values, combinations of single values and ranges, or null values in the edit table. *Value* must be a 1- through 34-character value and must be enclosed in parentheses.

encode-value decode-value

Specifies pairs of values to be inserted in the code table. *Encode-value* must be a 1- through 34-character value; *decode-value* must be a 1- through 62-character value. The specified values must be enclosed in parentheses.

NOT FOUND is a condition to be acted upon and may be used as an *encode-value* or as an *decode-value* or as both (refer to the *CA IDMS Mapping Facility Guide* document for more information).

NULl

Removes all values from the table.

GENerate

Causes a load module containing all the values in the table to be placed in the dictionary load area. The generated load module has the same name and version number as the named table.

WITh/ALSo WITh/WITHOut

Includes or excludes the specified options when the specified table is displayed or punched. Detailed information for each DISPLAY/PUNCH option is under SET OPTIONS Syntax (see page 36). The option that is listed below presents special considerations for this entity type.

DETails

Includes table data.

Usage

TABLE statement considerations

The following considerations apply to TABLE:

- The reserved words TABLE and MODULE are interchangeable within TABLE statement clauses, unless otherwise noted.
- Tables are automatically associated with the LANGUAGE class through the TABLE attribute.
- Tables defined by means of the TABLE statement are referred to as *stand-alone tables*. The RECORD ELEMENT and COBOL substatements (described under RECORD (REPORT/TRANSACTION) (see page 287), previously in this chapter) are used to define built-in tables. For a description of stand-alone and built-in tables, refer to the *CA IDMS Mapping Facility Guide* manual.

If you specify REPLACE

If you specify REPLACE, the DDDL compiler initializes to defaults and/or excludes the following options:

- DESCRIPTION
- Related tables
- USER REGISTERED FOR
- Related attributes
- PUBLIC ACCESS
- Table data
- WITHIN SYSTEM
- COMMENTS/DEFINITIONS/*comment-key*

The following relationships are not affected:

- Modules to which the named table is related
- Users accessing the named table
- Programs using the named table
- LANGUAGE specification

Cross-referencing maps and tables

You can add cross-referencing from a table to any MAP (maps used by the CA IDMS mapping facility or documentation IDD maps). You must remove all cross-referencing before you can delete a table.

Example

The following statements add tables MONTHTBL and DECODMTH. MONTHTBL is an edit table that contains the valid values 1 through 12 for the months of the year; DECODMTH is a code table that relates the names of the months to the 2-digit month codes used in the database:

```
add table name is monthtbl
    table description is 'valid months'
    type is edit
    search is linear
    table data is alphanumeric
    table is unsorted
    values are ( 01 thru 12 )
    .

add table name is decodmth
    table description is 'month code convert'
    type is code
    search is linear
```

```
encode data is alphanumeric
table is unsorted
duplicates are allowed
values are ( 01 jan 02 feb 03 mar 04 apr
     05 may 06 jun 06 june 07 jul 07 july
     08 aug 09 sep 10 oct 11 nov 12 dec
     not found other )
.
```

TASK

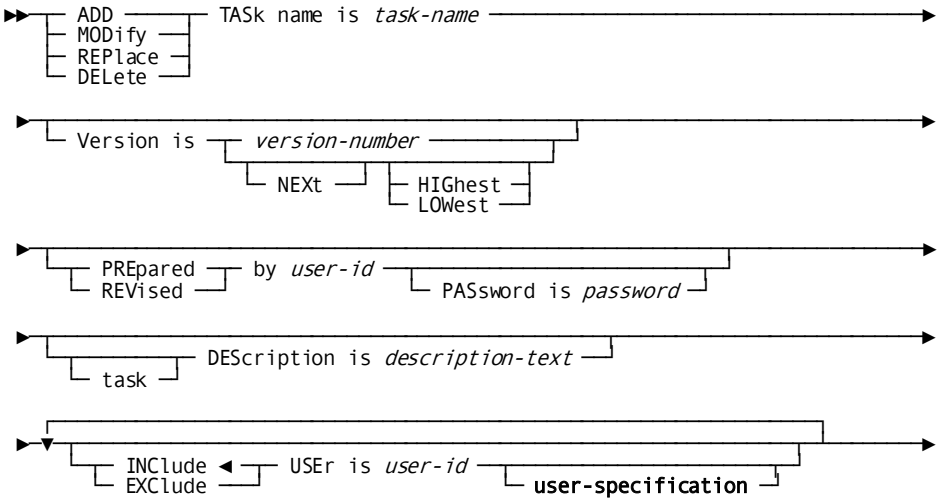
TASK statements are used to document teleprocessing system tasks. Optional clauses define the program invoked by the task and the task's priority and maximum-wait interval.

Note: It is recommended that you maintain TASK definitions using the system generation compiler, *not* the DDDL compiler. If a system generation component is processed by the DDDL compiler, only dictionary security is checked, *not* system generation security. For more information on using the system generation compiler, refer to *CA IDMS System Generation Guide*.

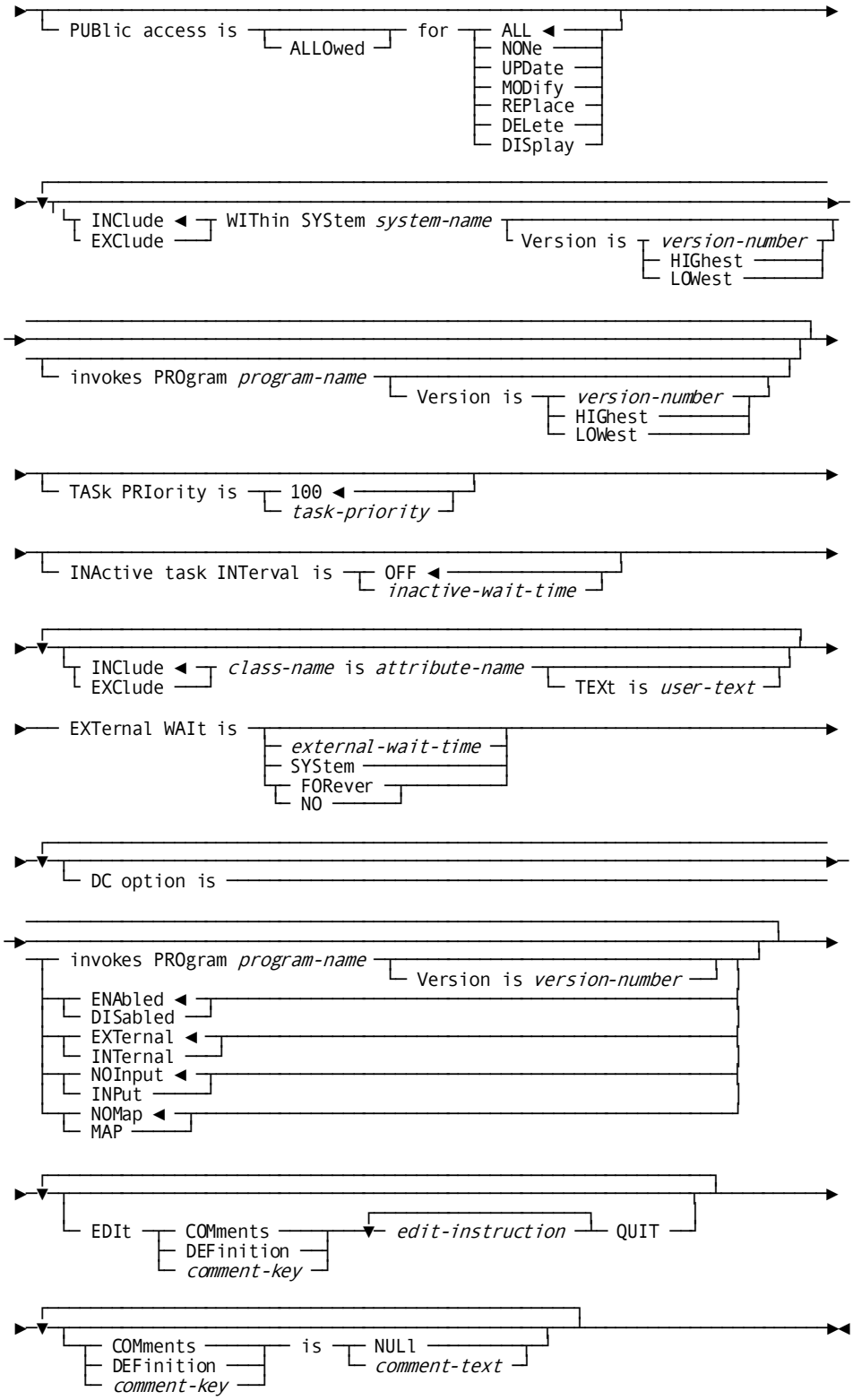
If the SET OPTIONS statement specifies SECURITY FOR IDMS-DC IS ON, the user must be assigned the proper authority to issue TASK statements.

Syntax

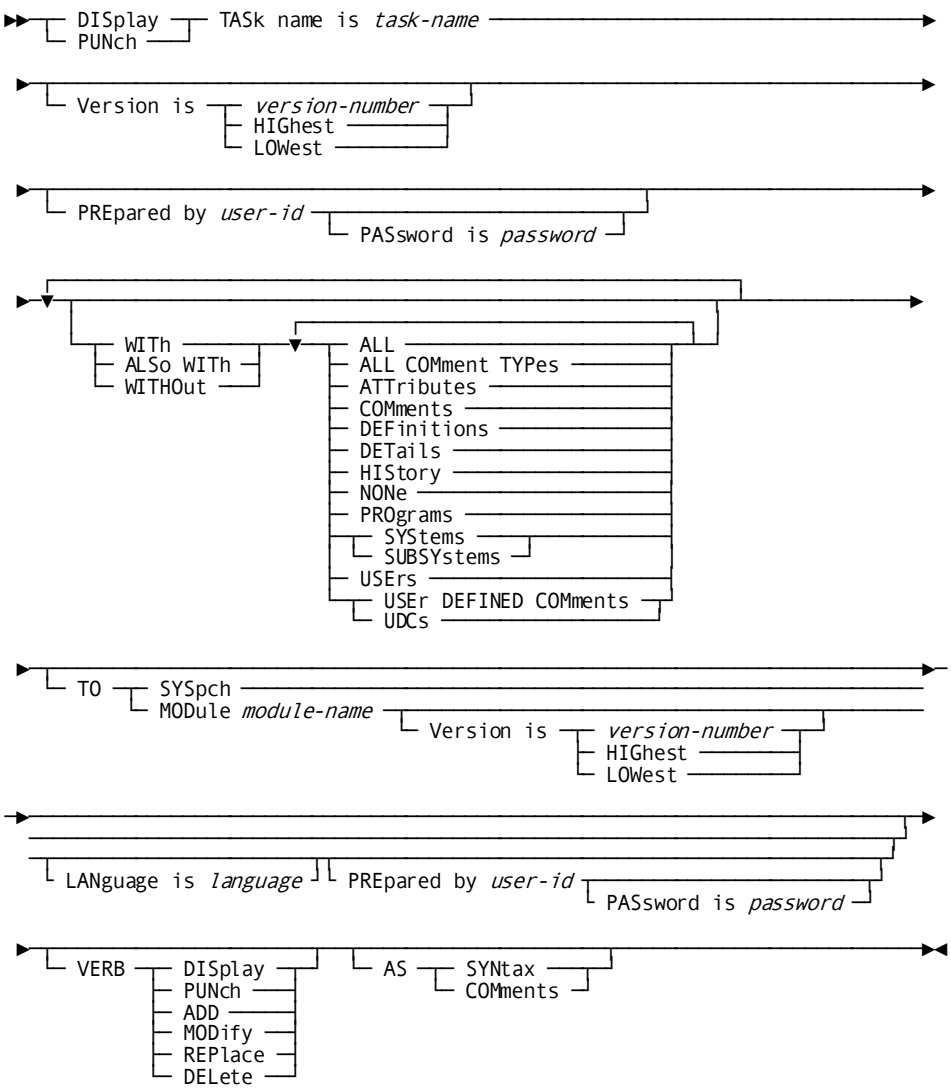
TASK statement



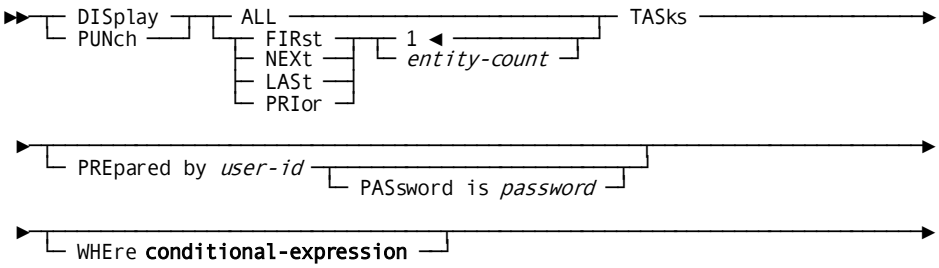
(for complete **user-specification** syntax, see USER clause)


PUBlic access is
ALLOwed
for
ALL
NONe
UPDate
MODify
REPlace
DELete
DISplay
INClude
EXClude
WITHin SYStem system-name
Version is
version-number
HIGhest
LOWest
invokes PROgram program-name
Version is
version-number
HIGhest
LOWest
TASk PRIority is
100
task-priority
INActive task INTerval is
OFF
inactive-wait-time
INClude
EXClude
class-name is attribute-name
TEXt is user-text
EXTernal WAIt is
external-wait-time
SYStem
FORever
NO
DC option is
invokes PROgram program-name
Version is version-number
ENAbled
DISabled
EXTernal
INTernal
NOInput
INPut
NOMap
MAP
EDIt
COMments
DEFinition
comment-key
edit-instruction
QUIT
COMments
DEFinition
comment-key
is
NULl
comment-text

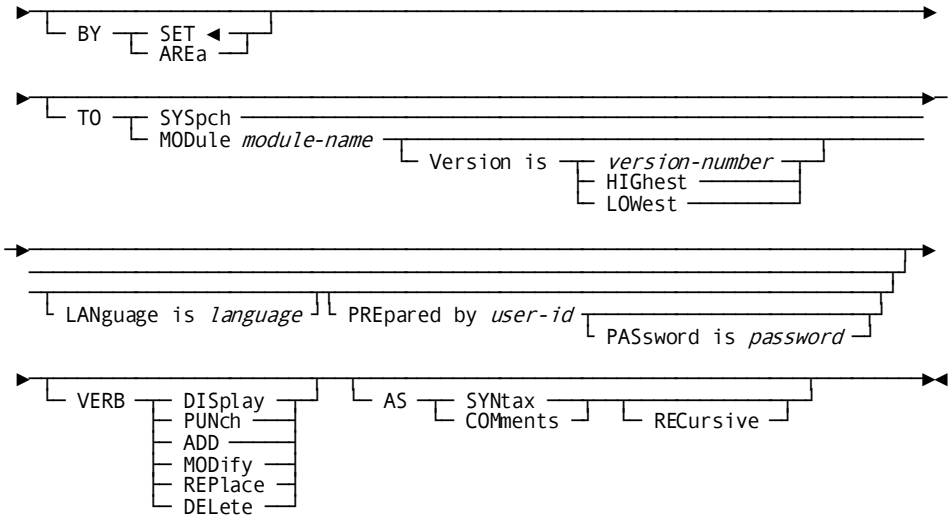
DISPLAY/PUNCH TASK statement (for a single task)



DISPLAY/PUNCH TASK statement (for multiple tasks)



(for complete **conditional-expression** syntax, see WHERE clause)


BY
SET
AREa
TO
SYSpch
MODule module-name
Version is
version-number
HIGhest
LOWest
LANguage is language
PREpared by user-id
PASsword is password
VERB
DISplay
PUNch
ADD
MODify
REPlace
DELete
AS
SYNtax
COMments
RECursive

Parameters

TASk name is *task-name*

Identifies a new task to be established in the dictionary, or an existing task to be modified, replaced, deleted, displayed, or punched. *Task-name* must be a 1- through 8-character alphanumeric value.

within SYStem *system-name*

Associates the named task with the system identified by the 1- through 32-character *system-name*. The WITHIN SYSTEM clause is documentation only, unless the system generation compiler COPY facility is to be used to copy task occurrences from an IDD-built system. When the COPY facility is not used, functional task/system relationships are established and maintained by the system generation compiler.

invokes PROgram *program-name*

Identifies the initial program to be invoked by the teleprocessing monitor for the named task. *Program-name* must be the 1- through 8-character name of an existing program. This parameter associates an existing program with the task/system relationship. The INVOKES PROGRAM parameter is documentation only.

If INVOKES PROGRAM is specified, the named program must have been previously related to the system by means of the WITHIN SYSTEM clause of the PROGRAM statement.

If the INVOKES PROGRAM option is omitted, INCLUDE establishes a new task/system relationship and EXCLUDE removes the task/system relationship and any dependent task/program relationships.

TASk PRIority is 100/*task-priority-number*

Specifies a dispatching priority for the named task. *Task-priority-number* must be an integer in the range 1 through 255; the default for ADD is 100. In an environment, a high number indicates a high priority. Task priorities are used in combination with user and logical-terminal priorities to establish the run-time dispatching priority of the task.

INACtive task INTerval is

Specifies the time the named task can be permitted to wait for a resource before being terminated.

OFF

Specifies that the task will never terminate due to elapsed time. OFF is the default.

inactive-wait-time

Specifies that the task will terminate if the specified wait time is exceeded. *Inactive-wait-time* is specified in seconds and must be an integer in the range 1 through 32,767.

EXTernal WAIt is

Overrides the system generation statement EXTERNAL WAIT parameter specification for the named program.

External-wait-time

Specifies the amount of time, in wall-clock seconds, the system is to wait for the program to issue a database request before abnormally terminating the program. *External-wait-time* must be an integer in the range 0 through 32,767.

SYStem

Directs the system to use the external wait time specified in the SYSTEM statement. A value of 0 is synonymous with SYSTEM.

FORever/NO

Directs the system not to terminate the program based on an external wait time.

DC option is

Documents the information used to define the named task during system generation.

INVOKES PROGRAM *program-name*

Identifies the initial program to be invoked by DC/UCF for the task. *Program-name* must reference an existing program. This parameter is required for DC/UCF tasks.

ENAbled

Automatically enables the task at system startup. ENABLED is the default.

DISabled

Disables the task until it is enabled explicitly by an operator command during system execution.

EXTernal

Specifies that the task can be invoked externally from a terminal. EXTERNAL is the default.

INTernal

Specifies that the task can be invoked only by means of a DC RETURN from an executing program.

NOInput

Specifies that the task's terminal input buffer is to contain only the task code. NOINPUT is the default.

INPut

Specifies that the task's terminal input buffer can contain data in addition to the task code. INPUT must be specified if the task's initial program reads the input line.

NOMap

Specifies that a map is not invoked. NOMAP is the default.

MAP

Specifies that tasks defined to write maps to user terminals can perform that function exclusively. DC/UCF displays a map automatically at the user terminal and eliminates the need for a program to perform this I/O function.

WITh/ALSo WITh/WITHOut

Includes or excludes the specified options when the named task is displayed or punched. Detailed information for each DISPLAY/PUNCH option is under SET OPTIONS Syntax (see page 36). The option that is listed below presents special considerations for this entity type.

DETails

Includes the DESCRIPTION, TASK PRIORITY, INACTIVE TASK INTERVAL, and DC OPTION specifications.

Usage

If you specify REPLACE

If you specify REPLACE, the DDDL compiler initializes to defaults and/or excludes the following:

- DESCRIPTION
- USER REGISTERED FOR
- PUBLIC ACCESS
- WITHIN SYSTEM
- COMMENTS/DEFINITIONS/*comment-key*
- TASK PRIORITY
- INACTIVE TASK INTERVAL
- ATTRIBUTES
- DC OPTION

Task/system relationships established by the system generation compiler are not affected.

Example

The following ADD statement defines the task RESER9 within the system RES and associates the program RES1054 with the task/system relationship. Additional clauses supply priority and wait-time-before-termination specifications.

```
add task name is reser9
    task description is 'reserve interaction'
    within system res
    invokes program res1054
```

```
    task priority is 215
    inactive interval is 5.
```

The following MODIFY statement disassociates the task from the system RES; the DC OPTION clauses establish the task's initial program and identify the task as part of a continuing transaction for use by a DC/UCF system.

```
modify task name is reser9
    exclude within system res
    dc option is invokes program res1054
    dc option is transaction nostart.
```

USER

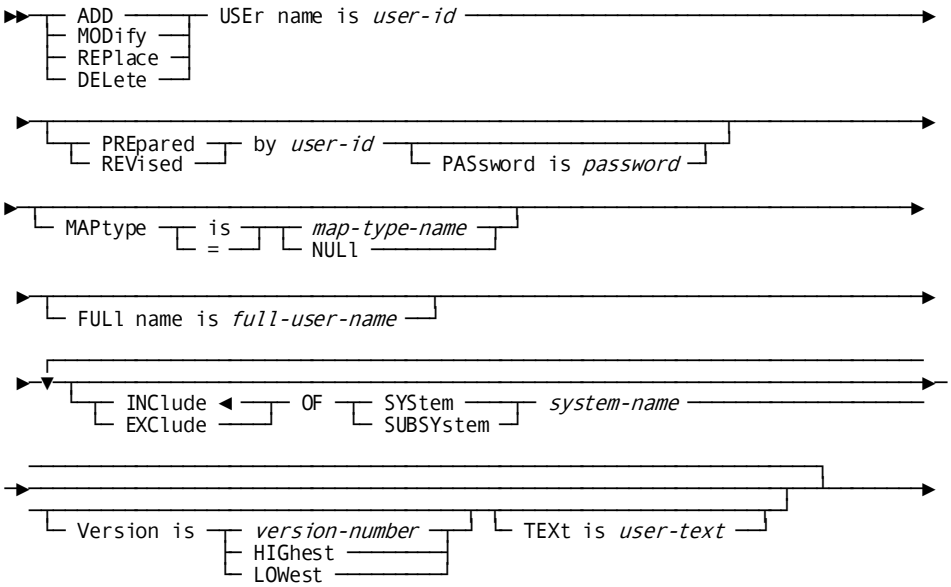
USER statements document users in the dictionary by relating users to systems and to other users, assigning users the authority to access secured products and entity types and to perform secured operations, and supporting attribute/entity relationships and documentation entries.

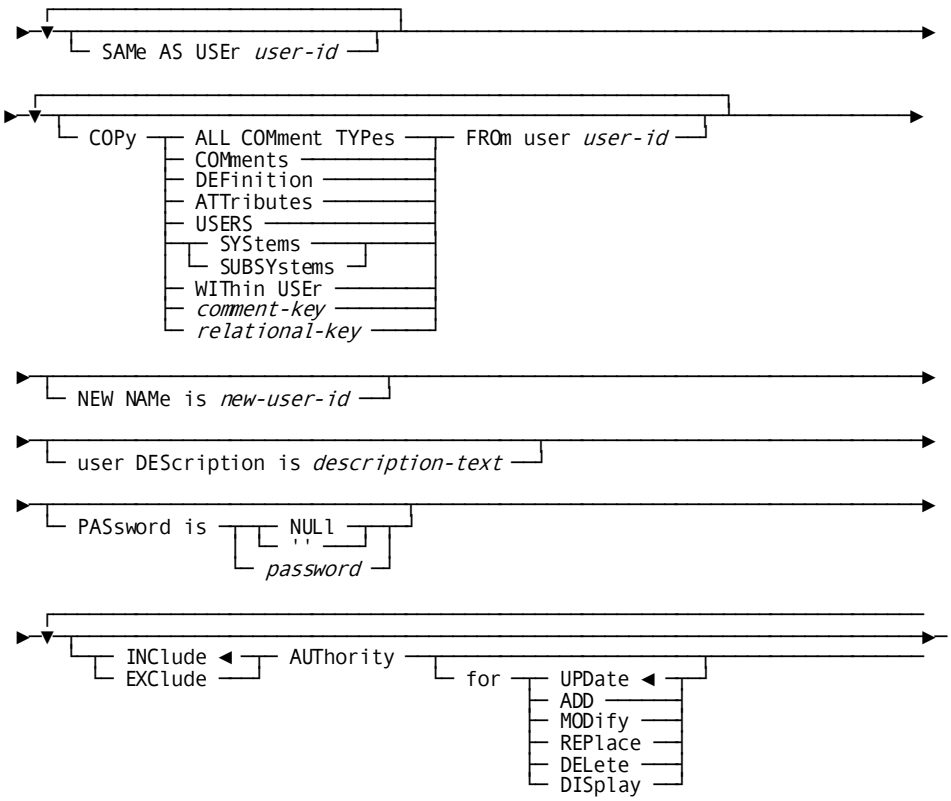
The system generation compiler can be used in conjunction with the DDDL compiler to complete user definitions. For additional information, refer to *CA IDMS System Generation Guide*.

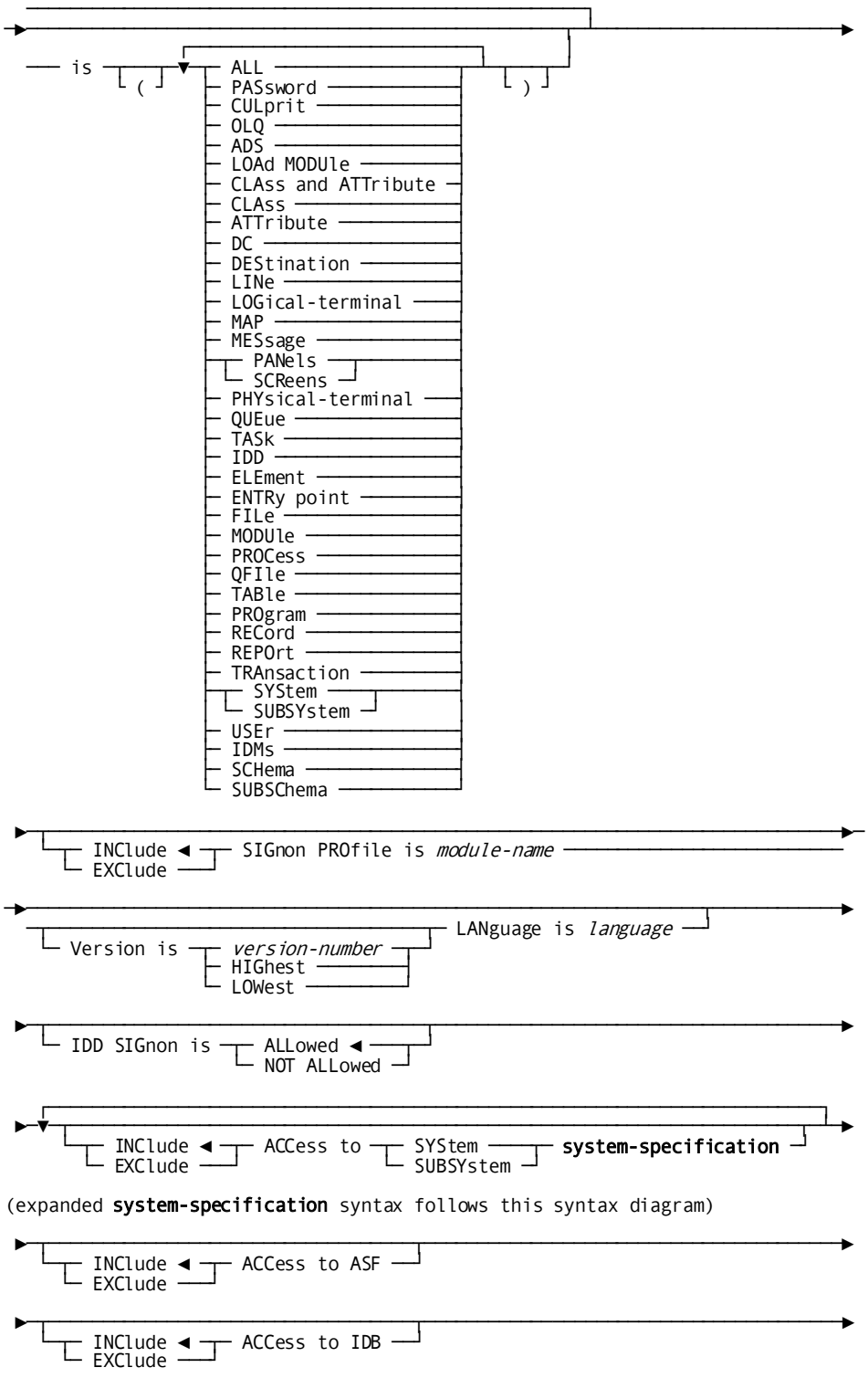
If the SET OPTIONS statement specifies SECURITY FOR IDD IS ON, the user must be assigned the proper authority to issue USER statements.

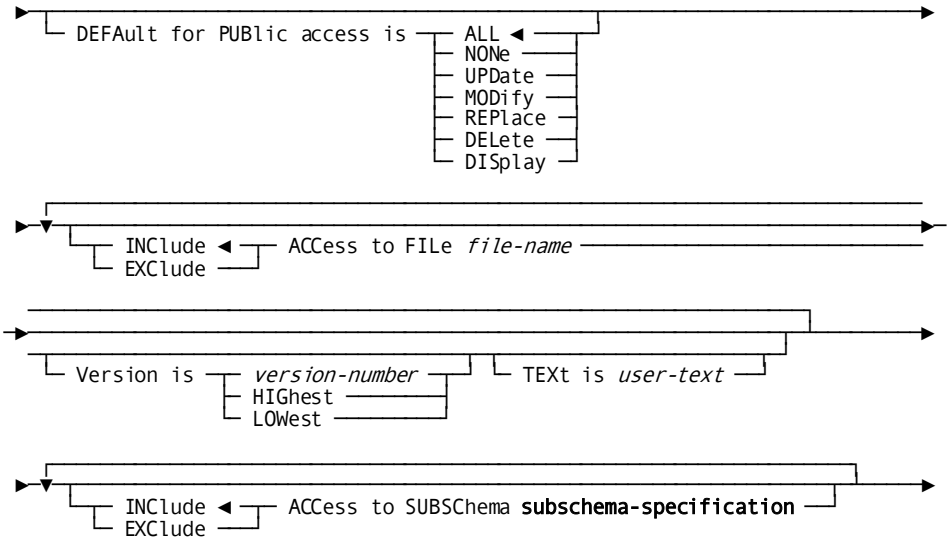
Syntax

USER statement

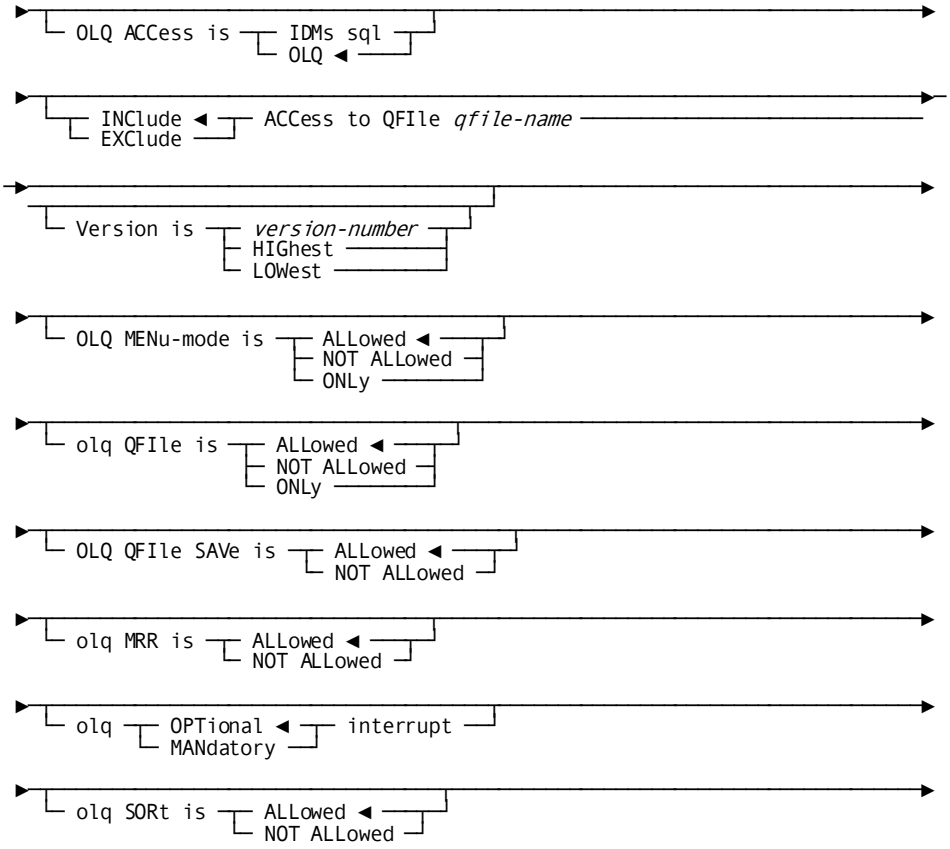


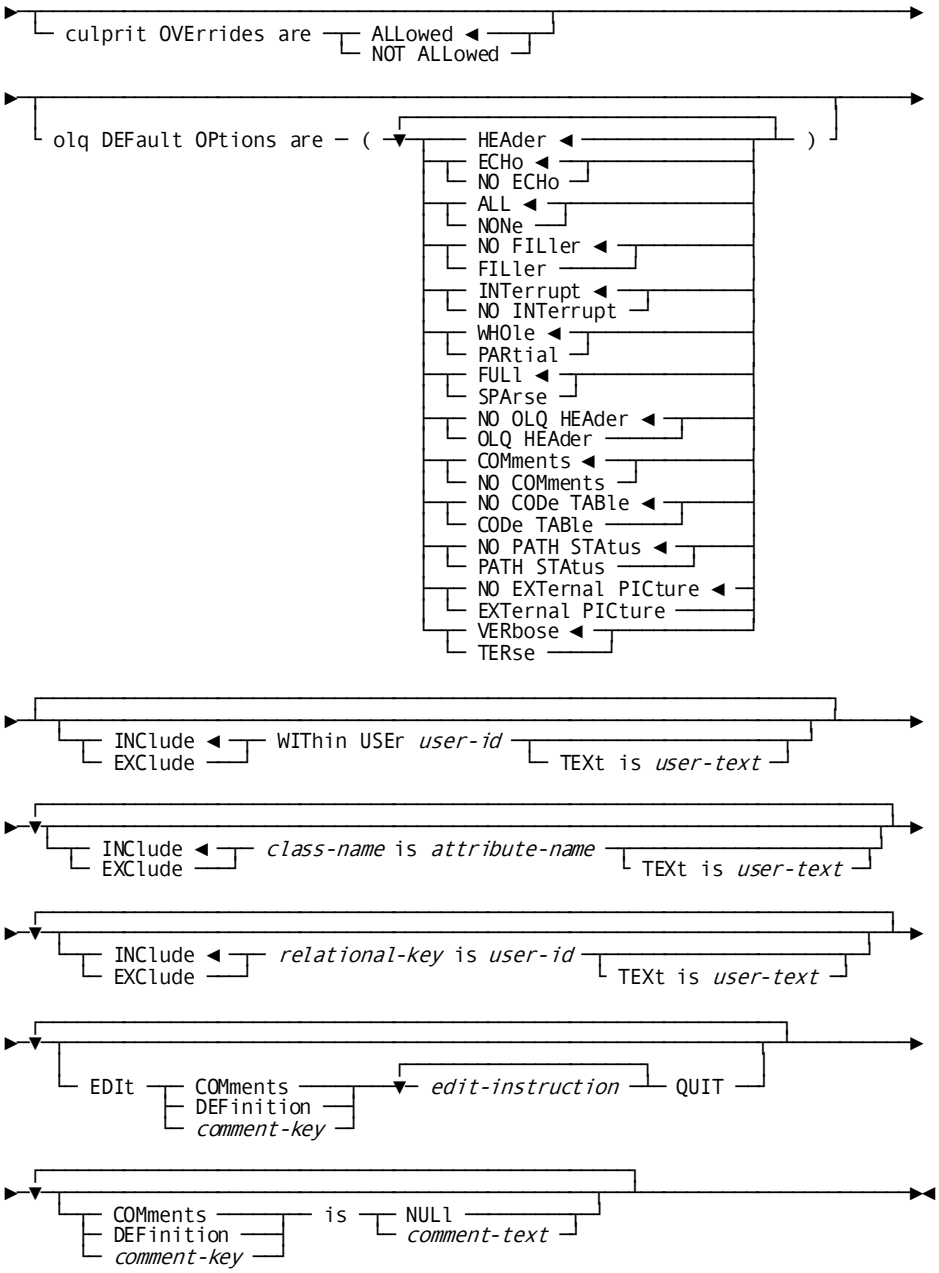

SAMe AS USEr user-id
COPy
ALL COMment TYPes
COMments
DEFinition
ATTributes
USERS
SYStems
SUBSYstems
WITHin USEr
comment-key
relational-key
FROm user user-id
NEW NAMe is new-user-id
user DEScription is description-text
PASsword is
NULl
''
password
INClude
EXClude
AUThority
for
UPDate
ADD
MODify
REPlace
DELete
DISplay


is
(
)
ALL
PASsword
CULprit
OLQ
ADS
LOAd MODUle
CLAss and ATTribute
CLAss
ATTribute
DC
DESTination
LINe
LOGical-terminal
MAP
MESsage
PANels
SCReens
PHYsical-terminal
QUEue
TASk
IDD
ELEment
ENTRy point
FILe
MODUle
PROCess
QFIle
TABle
PROgram
RECord
REPOrt
TRAnsaction
SYStem
SUBSYstem
USEr
IDMs
SCHema
SUBSChema
INClude
EXClude
SIGnon PROfile is module-name
LANguage is language
Version is
version-number
HIGhest
LOWest
IDD SIGnon is
ALLowed
NOT ALLowed
INClude
EXClude
ACCess to
SYStem
SUBSYstem
system-specification
(expanded system-specification syntax follows this syntax diagram)
INClude
EXClude
ACCess to ASF
INClude
EXClude
ACCess to IDB

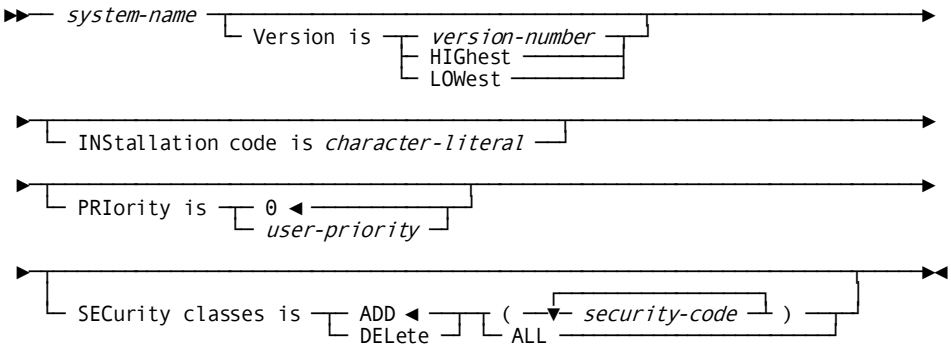


(expanded **subschema-specification** syntax follows this syntax diagram)

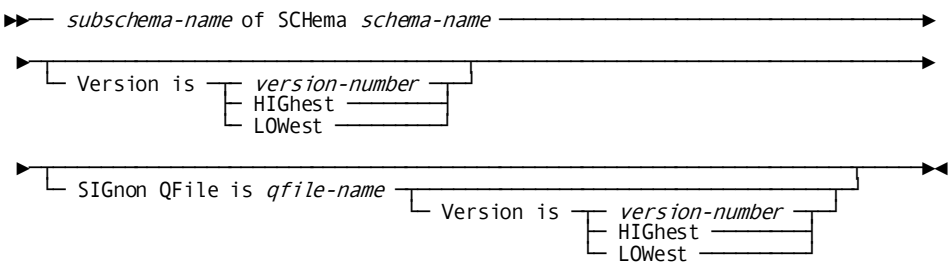



culprit OVErrides are
ALLowed
NOT ALLowed
olq DEFault OPtions are
(
)
HEAder
ECHo
NO ECHo
ALL
NONe
NO FILler
FILler
INTerrupt
NO INTerrupt
WHOle
PARtial
FULl
SPArse
NO OLQ HEAder
OLQ HEAder
COMments
NO COMments
NO CODe TABle
CODe TABle
NO PATH STAtus
PATH STAtus
NO EXTernal PICture
EXTernal PICture
VERbose
TERse
INClude
EXClude
WITHin USEr user-id
TEXt is user-text
class-name is attribute-name
relational-key is user-id
EDIt
COMments
DEFinition
comment-key
edit-instruction
QUIT
is
NULl

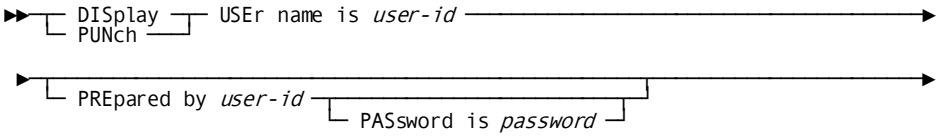
Expansion of *system-specification*

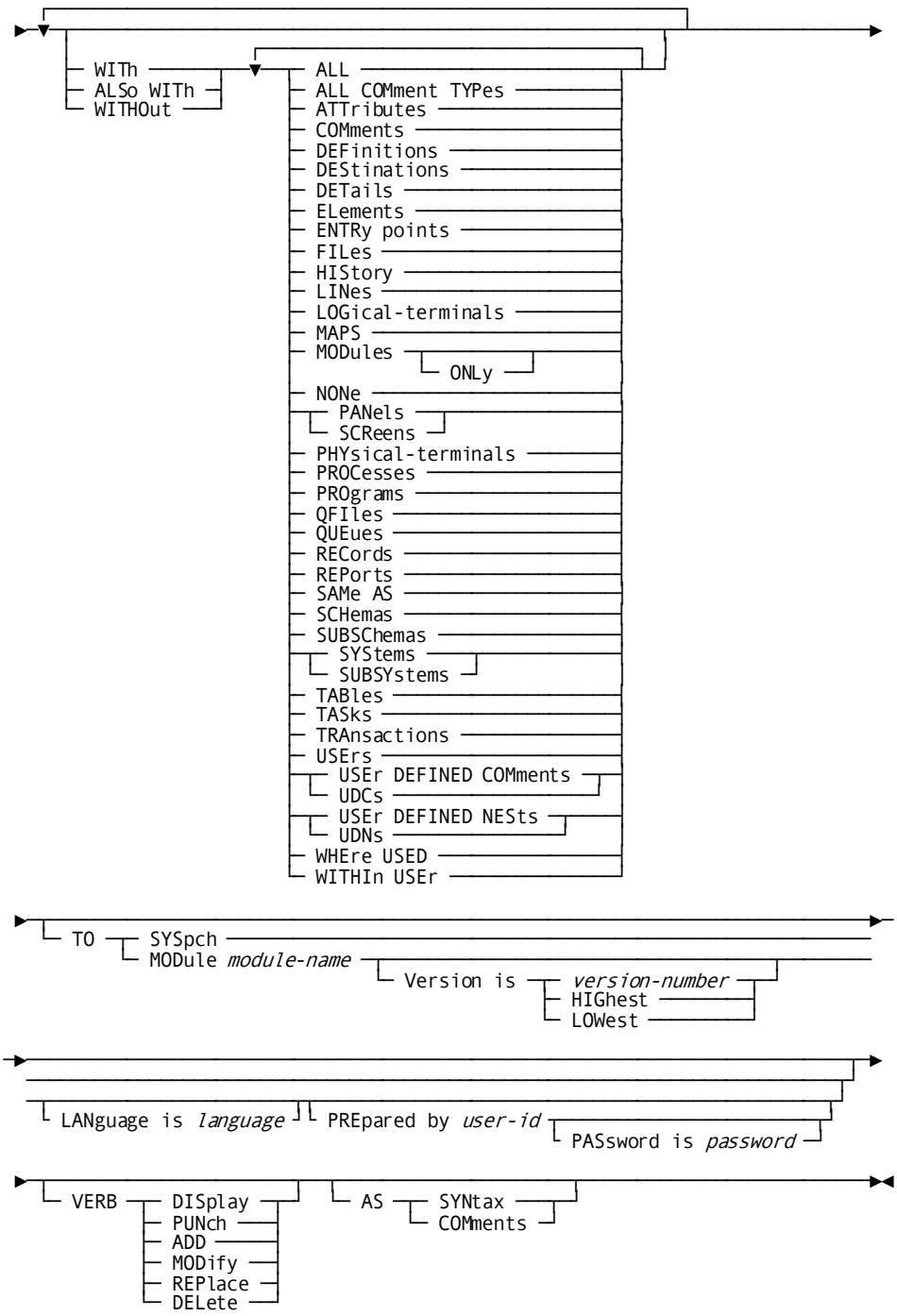


Expansion of *subschema-specification*

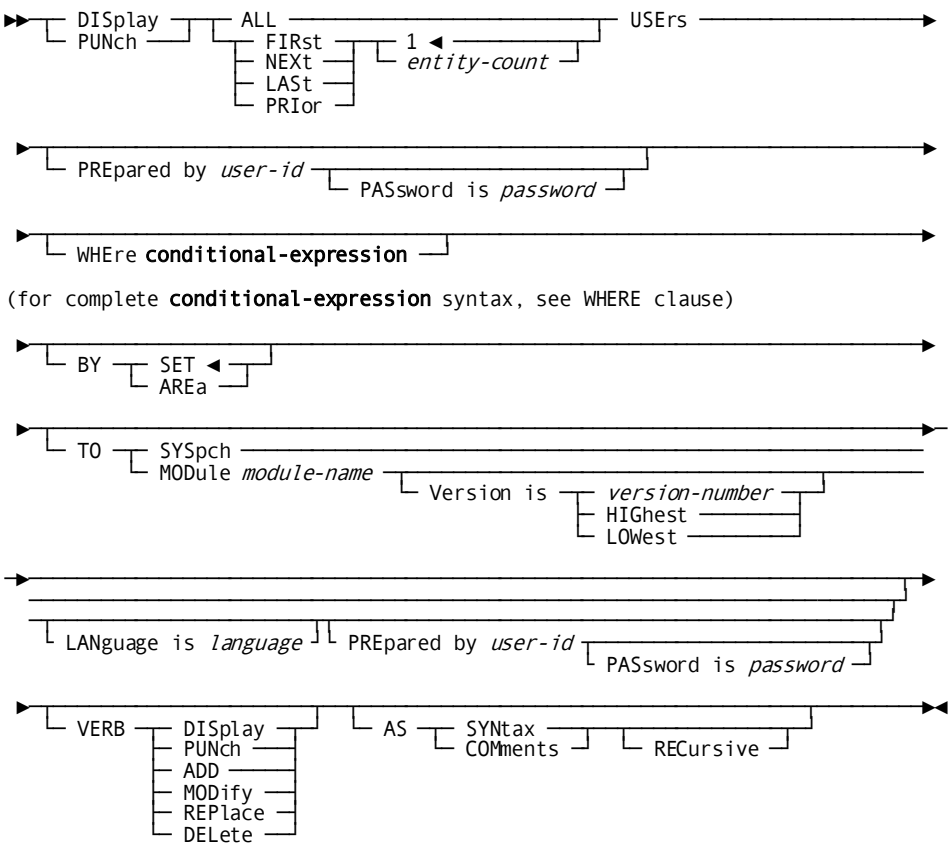


DISPLAY/PUNCH USER statement (for a single user)




WITh
ALSo WITh
WITHOut
ALL
ALL COMment TYPes
ATTributes
COMments
DEFinitions
DEStinations
DETails
ELements
ENTRy points
FILes
HIStory
LINes
LOGical-terminals
MAPS
MODules
ONLy
NONe
PANels
SCReens
PHYsical-terminals
PROCesses
PROgrams
QFILes
QUEues
RECords
REPorts
SAMe AS
SCHemas
SUBSChemas
SYStems
SUBSYstems
TABles
TASks
TRAnsactions
USErs
USEr DEFINED COMments
UDCs
USEr DEFINED NESts
UDNs
WHERe USED
WITHIn USEr
TO
SYSpch
MODule module-name
Version is
version-number
HIGhest
LOWest
LANguage is language
PREpared by user-id
PASsword is password
VERB
DISplay
PUNch
ADD
MODify
REPlace
DELete
AS
SYNtax
COMments

DISPLAY/PUNCH USER statement (for multiple users)



Parameters

USEr name is *user-id*

Identifies a new user to be established in the dictionary, or an existing user to be modified, replaced, deleted, displayed, or punched. *User-id* must specify a 1- through 32-character alphanumeric value and must be unique in the dictionary.

MAPtype is/= *map-type-name*/NULl

MAPTYPE has no meaning for CA IDMS since Release 12.0. It does not give an error so that migration can run without changes. MAPTYPE is now processed with DCUF SET MAPTYPE or with a PROFILE as specified in the MAPPING FACILITIES manual. For further information, see *Advantage™ CA-IDMS™ Mapping Facility.*

FULl name is *full-user-name*

Specifies a 1- through 32-character name that clarifies or supplements *user-name* or supplies the full name for an abbreviated user name.

OF SYStem/SUBSYstem *system-name*

Establishes (INCLUDE) or removes (EXCLUDE) a documentation relationship between the named user and the requested system or subsystem.

SAMe AS USEr *user-id*

Copies the following options from the definition of the named user: user nests, attributes, systems associated with the user by means of the OF SYSTEM/SUBSYSTEM clause, and comments.

NEW NAMe is *new-user-id*

Specifies a new name for the requested user. This clause changes only the name specification; it does not alter or delete any previously defined relationships in which the named user participates. Subsequent references to the user must specify the new name. *New-user-id* must be a 1- through 32-character value that does not duplicate the name of an existing user.

PASsword is NULl/*password*

Establishes, replaces, or deletes a password for the named user. *password* must be a 1- through 8-character alphanumeric value. Specify PASSWORD IS NULL or PASSWORD IS '' to delete a password. This password must appear whenever the user name appears in an IDD SIGNON statement or in a PREPARED BY or REVISED BY clause.

If the SET OPTIONS statement specifies INDIVIDUAL PASSWORD SECURITY OVERRIDE IS ON and the USER statement is modifying the issuing user's password, neither AUTHORITY FOR UPDATE IS PASSWORD nor AUTHORITY FOR MODIFY IS USER need be specified; the AUTHORITY clause is described below. However, if the SET OPTIONS statement specifies INDIVIDUAL PASSWORD SECURITY OVERRIDE IS OFF, the issuing user must be assigned PASSWORD authority as well as the appropriate USER entity-type authority.

AUThority for UPDate/ADD/MODify/REPlace/DELete/DISplay

Assigns to (INCLUDE) or removes from (EXCLUDES) the named user the authority to access a secured product or entity type or to perform a secured operation. (Security must have been previously enabled by means of a SET OPTIONS statement SECURITY clause.)

This clause also specifies the verbs that the named user can issue for entities within secured products:

- UPDATE specifies that the user can issue all verbs (ADD, MODIFY, REPLACE, DELETE, and DISPLAY/PUNCH). UPDATE is the default.
- ADD specifies that the user can issue only ADD and DISPLAY/PUNCH verbs.
- MODIFY specifies that the user can issue only MODIFY and DISPLAY/PUNCH verbs.
- REPLACE specifies that the user can issue only REPLACE and DISPLAY/PUNCH verbs.
- DELETE specifies that the user can issue only DELETE and DISPLAY/PUNCH verbs.
- DISPLAY specifies that the user can issue only DISPLAY/PUNCH verbs.

To specify the INCLUDE/EXCLUDE parameter, the PREPARED/REVISED BY clause must identify a user with the AUTHORITY FOR UPDATE IS PASSWORD option. For more information about IDD security, see Securing the Dictionary (see page 72).

ALL

Assigns the user the authority to access all products and entity types and in order to perform all secured operations. AUTHORITY FOR UPDATE IS ALL is required to establish default processing options for a specified dictionary by issuing the SET OPTIONS FOR DICTIONARY statement. This authority is also required to use the FIRST/SECOND/THIRD/FOURTH ALTERNATE PICTURE KEYWORD clause of the SET OPTIONS statement. Finally, AUTHORITY FOR UPDATE IS ALL is required to turn off entity-occurrence security with the REGISTRATION OVERRIDE clause.

The IDD installation procedure establishes one user with the AUTHORITY FOR UPDATE IS ALL attribute. This user is named 'CULL DBA' and assigned the password DBAPASS. After the installation, rename 'CULL DBA' and modify the password. Create a backup by adding another user with AUTHORITY IS ALL; if the new name of the DBA is inadvertently forgotten or lost, the backup user can be used.

PASsword

Allows the user to assign or change passwords for other users and to issue the AUTHORITY FOR PASSWORD clause for other users. A user with password authority can update the AUTHORITY clause of *any* user ID, including his own, to any level. Note that if PASSWORD is selected, the keyword UPDATE must be specified in the FOR clause (described above).

CULprit

Allows the user to access files and subschemas to run CA Culprit reports, change record layouts and file definitions (if the named user is assigned the CULPRIT OVERRIDES ARE ALLOWED option), and to generate DDR reports (if the named user is assigned the CULPRIT OVERRIDES ARE ALLOWED option and is authorized to access subschema IDMSNWKA of schema IDMSNTWK, version 1). This parameter allows the user to perform CA Culprit-related activities when the default processing options for the session include SECURITY FOR CULPRIT IS ON. Note that if CULPRIT is selected, the keyword UPDATE must be specified in the FOR clause (described above).

OLQ

Allows the user to code USER statement clauses that control access to CA OLQ files and subschema views and assign OLQ command authorities and processing/reporting options when the default processing options for the session include SECURITY FOR OLQ IS ON. If OLQ is specified, the keyword UPDATE must be specified in the FOR clause (described above).

ADS

Allows the user to generate CA ADS dialogs when the default processing options for the session include SECURITY FOR ADS IS ON. If the keyword UPDATE is specified in the FOR clause (described above), either MODIFY or REPLACE allows the user to modify CA ADS dialogs.

LOAd MODUle

Allows the user to access load modules when the default processing options for the session include SECURITY FOR LOAD MODULE IS ON.

CLAss and ATTribute

Allows the user to access classes, attributes, and user-defined entities when the default processing options for the session include SECURITY FOR CLASS AND ATTRIBUTE IS ON. Note that the keywords CLASS and ATTRIBUTE can be issued separately to assign individual authority for classes or attributes (user-defined entities).

DC

Allows the user to access teleprocessing entities (DESTINATION, LINE, LOGICAL-TERMINAL, MAP, MESSAGE, PANEL, PHYSICAL-TERMINAL, QUEUE, and TASK) when the default processing options for the session include SECURITY FOR IDMS-DC IS ON. Note that the keywords DESTINATION, LINE, LOGICAL-TERMINAL, MAP, MESSAGE, PANEL, PHYSICAL-TERMINAL, QUEUE, and TASK can be issued to assign authority for the specified entity type only.

IDD

Allows the user to access IDD entities (ELEMENT, ENTRY POINT, FILE, MODULE, PROCESS, PROGRAM, QFILE, RECORD, REPORT, TRANSACTION, SYSTEM, TABLE, and USER) when the default processing options for the session include SECURITY FOR IDD SIGNON and/or IDD IS ON. Note that the keywords ELEMENT, ENTRY POINT, FILE, MODULE, PROCESS, PROGRAM, QFILE, RECORD, REPORT, TRANSACTION, SYSTEM, TABLE, and USER can be issued to assign authority only for the specified entity type.

IDMs

Allows the user to access CA IDMS entities (SCHEMA, SUBSCHEMA, and DMCL) when the default processing options for the session include SECURITY FOR IDMS IS ON. Note that the keywords SCHEMA, SUBSCHEMA, and DMCL can be issued to assign authority only for the specified entity type.

SIGnon PROfile is *module-name*

Associates (INCLUDE) or disassociates (EXCLUDE) a module that has been defined for use as a signon profile. *Module-name* must reference an existing module. The LANGUAGE parameter is required; *language* specifies the language of the signon profile; for example, OLQ or DC. All languages, including user-defined languages, can be specified.

When the named user signs onto an application, the commands within the signon profile module are executed automatically. These profiles are not executed when signing onto a DC SYSTEM.

IDD SIGnon is

Specifies whether the named user is authorized to sign on to and execute the online or batch DDDL compiler when the SET OPTIONS statement specifies SECURITY FOR IDD IS ON.

Note that the issuing user must be assigned IDD SIGNON authority.

ALLowed

Authorizes the user to sign on to the DDDL compiler. ALLOWED is the default.

NOT ALLowed

Prohibits the user from signing on to the DDDL compiler.

ACCess to SYStem/SUBSYstem *system-name*

Establishes (INCLUDE) or removes (EXCLUDE) a system access privilege. If this clause is specified in a non CA IDMS environment, the user/system relationship is documentation.

Note: You must have IDMS-DC authority to use this clause.

INStallation code is *character-literal*

Specifies an installation code for the named user. This code can be accessed at runtime by user exits or programs to provide additional security. *Character-literal* must be a 1- through 32-character alphanumeric symbol specified as an absolute expression.

PRIority is 0/*user-priority*

Specifies the dispatching priority for the named user. DC/UCF uses the dispatching priority in combination with task and logical terminal priorities to establish a run-time dispatching priority for tasks initiated by the named user. *User-priority* must be an integer in the range 0 through 255; the default for ADD operations is 0. A high number indicates a high dispatching priority.

SECurity classes is

Adds or deletes securityclass codes for the named user; the user can execute only programs and tasks with matching security classes.

ADD/DELete

Specifies that the named security classes are added to or deleted from the user definition; ADD is the default for ADD operations.

***security-code*/ALL**

Specifies that the named security classes or all security classes are the object of the ADD or DELETE request. *Security-code* must be an integer in the range 1 through 255; multiple values must be enclosed in parentheses and separated by blanks.

ACCess to ASF

Specifies that the named user has (INCLUDE) or does not have (EXCLUDE) access to the CA IDMS ASF

ACCess to IDB

Specifies that the named CA IDMS or Information Center Management System (ICMS) user has (INCLUDE) or does not have (EXCLUDE) access to the Information Database (IDB).

DEFAult for PUBlic access is

Assigns a default public access specification to the named user. This feature, for ASF users only, is used to identify the public access level to be established by the user when storing entity-occurrence definitions in the dictionary through ASF. If an option other than ALL is specified, ASF automatically generates the appropriate registration option within the entity definition.

ACCess to FILe *file-name*

Specifies that the named CA Culprit user has access to the named file. Note that if CA Culprit security is enabled, the requested user must be assigned CULPRIT authority in order to access the named file.

ACCess to SUBSChema *subschema-name* of SCHema *schema-name*

Specifies that the named CA OLQ or CA Culprit user has access to (INCLUDE) or does not have access to (EXCLUDE) the named subschema. *Subschema-name* must identify a subschema view associated with *schema-name*. If CA OLQ or CA Culprit product security has been enabled in the SET OPTIONS statement SECURITY clause, the issuing user must be assigned OLQ or CULPRIT authority.

SIGnon QFIle is *qfile-name*

Associates an existing qfile with the named subschema and establishes access privilege to that qfile for the named CA OLQ user. The named qfile is invoked automatically when the user signs on to OLQ and names the associated subschema.

Note: The qfile access privilege does not permit the named user to execute qfiles; the qfile execution privilege is established separately by means of the OLQ QFILE clause described below.

OLQ ACCess is

Indicates an CA OLQ user's type of qfile access.

IDMs sql

Specifies qfile access using the functionality available with the CA IDMS SQL, providing the CA IDMS SQL is installed. IDMs sql, IDMssql, and IDMS-SQL are synonyms and can be used interchangeably.

More information: For more information on CA IDMS SQL, see the *CA IDMS SQL Reference Guide*.

OLQ

Specifies qfile access using the functionality available with CA OLQ. OLQ is the default for OLQ ACCESS.

ACCess to QFIle *qfile-name*

Specifies that the named CA OLQ user has access to (INCLUDE) or does not have access to (EXCLUDE) the named qfile. Note that the qfile access privilege does not permit the named user to execute qfiles; qfile execution privilege is established separately by means of the OLQ QFILE clause described below.

OLQ MENu-mode is

Specifies whether the named user is authorized to access CA OLQ in menu mode. If the SET OPTIONS statement specifies SECURITY FOR OLQ IS ON, the issuing user must be assigned OLQ authority.

ALLowed

Authorizes the CA OLQ user to access CA OLQ in menu mode. ALLOWED is the default.

NOT ALLowed

Prohibits the CA OLQ user from accessing CA OLQ in menu mode.

ONLy

Specifies that the CA OLQ user is allowed to access CA OLQ in menu mode only.

OLQ QFIle is

Specifies whether the named user is authorized to execute CA OLQ qfiles. If the SET OPTIONS statement specifies SECURITY FOR OLQ IS ON, the issuing user must be assigned OLQ authority.

ALLowed

Authorizes the CA OLQ user to execute qfiles. ALLOWED is the default.

NOT ALLowed

Prohibits the CA OLQ user from executing qfiles.

ONLy

Specifies that the CA OLQ user is authorized to access CA OLQ only through qfiles.

OLQ QFIle SAVe is

Specifies whether the named CA OLQ user is authorized to save paths and CA OLQ command groups as qfiles. If the SET OPTIONS statement specifies SECURITY FOR OLQ IS ON, the issuing user must be assigned OLQ authority.

ALLowed

Authorizes the CA OLQ user to save paths and groups of commands as qfiles. ALLOWED is the default.

NOT ALLowed

Prohibits the CA OLQ user from saving paths and groups of commands as qfiles.

olq MRR is

Specifies whether the named CA OLQ user is authorized to retrieve multiple record occurrences with a single CA OLQ command. If the SET OPTIONS statement specifies SECURITY FOR OLQ IS ON, the issuing user must be assigned OLQ authority.

ALLowed

Authorizes the CA OLQ user to retrieve multiple record occurrences with a single OLQ command. ALLOWED is the default.

NOT ALLowed

Prohibits the CA OLQ user from retrieving multiple record occurrences with a single CA OLQcommand.

olq OPTional/MANdatory interrupt

Specifies whether the named CA OLQ user is authorized to select the OLQ NOINTERRUPT option (described below). If the SET OPTIONS statement specifies SECURITY FOR OLQ IS ON, the issuing user must be assigned OLQ authority.

OPTional

Authorizes the CA OLQ user to select the OLQ NOINTERRUPT option.

MANdatory

Requires that the OLQ INTERRUPT be enabled at all times for the user.

olq SORt is

Specifies whether the named CA OLQ user can issue the CA OLQ SORT command. If the SET OPTIONS statement specifies SECURITY FOR OLQ IS ON, the issuing user must be assigned OLQ authority.

ALLowed

Authorizes the CA OLQ user to issue the CA OLQ SORT command. ALLOWED is the default.

NOT ALLowed

Prohibits the CA OLQ user from issuing the CA OLQ SORT command.

culprit OVErrides are

Specifies whether the named CA Culprit user is authorized to define file attributes and records. If the SET OPTIONS statement specifies SECURITY FOR CULPRIT IS ON, the issuing user must be assigned CULPRIT authority.

ALLowed

Authorizes the CA Culprit user to code file attributes and REC parameters. ALLOWED is the default.

NOT ALLowed

Prohibits the CA Culprit user from coding file attributes and REC parameters.

olq DEFAult OPTions are

Specifies the CA OLQ processing control and display options that will be in effect when the named user signs on to CA OLQ. If the SET OPTIONS statement specifies SECURITY FOR OLQ IS ON, the issuing user must be assigned OLQ authority.

HEAder/NO HEAder

Specifies whether CA OLQ report files will contain a header line. This option has no effect on single-record-occurrence retrieval displays. The default for ADD is HEADER.

ECHo/NO ECHo

Specifies whether a user-entered command will be repeated by CA OLQ on the output device. The default for ADD is ECHO.

ALL/NONe

Specifies whether the default internal field list for all records retrieved during the named user's CA OLQ session will contain all or none of the fields. The default for ADD is ALL.

NO FILler/FILler

Specifies whether filler field values will be displayed. The default for ADD is NO FILLER.

INTerrupt/NO INTerrupt

Specifies whether the processing interrupt feature for multiple record retrievals will be enabled or disabled. The default for ADD is INTERRUPT.

Note: The OLQ MANDATORY INTERRUPT specification takes precedence over NO INTERRUPT.

WHOle/PARtial

Specifies the content of displayed path retrieval report lines. WHOLE displays only those lines containing a retrieved occurrence for every record type in a path definition. PARTIAL displays all lines, whether or not they contain data for every path record type. The default for ADD is WHOLE.

FULl/SPArse

Specifies the format of displayed path retrieval report lines. FULL displays data associated with a record type once for each retrieved occurrence. SPARSE displays data associated with a record type only once, regardless of how many associated record occurrences are retrieved. The default value for ADD is FULL.

NO OLQ HEAder/OLQ HEAder

Specifies whether the CA OLQ report file contains a header line. This option has no effect on single-record-occurrence retrieval displays. The default for ADD is NO OLQ HEADER.

COMments/NO COMments

Specifies whether comments will accompany the output from HELP RECORDS, HELP SUBSCHEMAS, and HELP QFILE requests. The default for ADD is COMMENTS.

NO CODe TABle/CODe TABle

Specifies whether CA OLQ will access a code table to encode and decode data. The default for ADD is NO CODE TABLE.

NO PATH STAtus/PATH STAtus

Specifies the conditions under which CA OLQ will retrieve a logical record. NO PATH STATUS requests CA OLQ to retrieve a logical record only when the path status of LR-FOUND is returned. PATH STATUS requests CA OLQ to retrieve a logical record when any DBA-defined path status is returned. The default for ADD is NO PATH STATUS.

NO EXTernal PICture/EXTernal PICture

Specifies whether CA OLQ will use external pictures for displaying data. The default for ADD is NO EXTERNAL PICTURE.

VERbose/TERse

Controls the amount of information displayed following record and field-level breaks. The default for ADD is VERBOSE.

WIThin USEr *user-id*

Associates (INCLUDE) the user with or disassociates (EXCLUDE) the user from the user identified by *user-id*.

WITh/ALSo WITh/WITHOut

Includes or excludes the specified options when the named user is displayed or punched. Detailed information for each DISPLAY/PUNCH option is under SET OPTIONS Syntax (see page 36). The options that are listed below present special considerations for this entity type.

DETails

Includes the following specifications:

- DESCRIPTION
- PASSWORD IS ASSIGNED
- FULL NAME
- AUTHORITY
- OLQ MENU-MODE
- OLQ QFILE
- OLQ MRR
- OLQ INTERRUPT
- OLQ SORT
- OLQ DEFAULT OPTIONS
- CULPRIT OVERRIDES
- IDD SIGNON

USErs

Includes all the users related by the WITHIN USER clause or relational-key clause.

Usage

If you specify REPLACE

If you specify REPLACE, the DDDL compiler initializes to defaults and/or excludes the following:

- FULL NAME
- DESCRIPTION
- OLQ DEFAULT OPTIONS
- OLQ options
- CULPRIT OVERRIDES
- USER REGISTERED FOR
- PUBLIC ACCESS
- COMMENTS/DEFINITIONS/*comment-key*
- AUTHORITY
- ACCESS TO SUBSCHEMA
- ACCESS TO SYSTEM/SUBSYSTEM
- ACCESS TO QFILE
- WITHIN USER
- ATTRIBUTES

The following relationships that include the named user or that the user is related to or registered for are not affected:

- Attributes
- Destinations
- Elements
- Files
- Lines
- Logical terminals
- Modules
- Panels
- Physical terminals
- Processes
- Programs

- qfiles
- Queues
- Records
- Systems (subsystems)
- Tables
- Tasks
- Users to which the named user is related

Additionally, the following definitions are not affected:

- User definitions built by other CA IDMS components
- Users that are related to other users

If you specify DELETE

If you specify DELETE, the DDDL compiler disassociates the named user from all entity occurrences, unless the user is the last user assigned the REGISTERED FOR ALL option; see PUBLIC ACCESS Clause (see page 79) for further details.

Default public access (ASF)

The default public access for entity occurrences stored by the named user through ASF is assigned as follows:

- ALL specifies that unregistered users are allowed to issue all verbs and perform all secured operations. ALL is the default.
- NONE specifies that unregistered users are not allowed to access the entity occurrence.
- UPDATE specifies that unregistered users are allowed to issue all verbs.
- MODIFY specifies that unregistered users are allowed to issue only MODIFY and DISPLAY/PUNCH verbs.
- REPLACE specifies that unregistered users are allowed to issue only REPLACE and DISPLAY/PUNCH verbs.
- DELETE specifies that unregistered users are allowed to issue only DELETE and DISPLAY/PUNCH verbs.
- DISPLAY specifies that unregistered users are allowed to issue only DISPLAY/PUNCH verbs.

USER AUTHORITY considerations

Consider the following points regarding user authority:

- Authority for IDD (or for a specific entity) is required to access a *basic* entity.
- Authority for CA IDMS (or for a specific entity) is required to access a *database* entity.
- Authority for IDD or MODULE is required before INCLUDE clauses can be processed.
- Authority for DC only applies to IDD usage. If a DC component was built or is owned by the system generation compiler and the DDDL compiler processes the component, only dictionary security is checked, not the central security used by system generation.
- Authority for MODULE includes authority for QFILE, TABLE, and PROCESS.
- ELEMENT authority is *not* required to:
  - Associate an existing element with a record.
  - Delete an existing element by using DELETE RECORD if the element doesn't exist in another record.
  - RECORD authority is *not* required to associate an existing record with a schema if you use the SHARE STRUCTURE parameter of the schema RECORD statement.
  - LOAD MODULE authority is *not* required to generate tables, subschemas, or DC/UCF systems. It *is* required to use LOAD MODULE with the subschema and DDDL compilers.
  - CLASS and ATTRIBUTE authority are *not* required to associate an attribute with an automatic class (a class defined as AUTOMATIC PLURAL).
  - ATTRIBUTE authority is *not* required to associate an existing user-defined comment or nest with an entity.

Example

In the following example, the ADD statement defines user DGS as a user of the systems INVENTORY and STOCK-UPDATE, supplying a full name, a password, and a description. The ACCESS TO SUBSCHEMA clauses assign access to two versions of a subschema and two signon qfiles.

The ACCESS TO SYSTEM clauses allow the user to access the systems INVENTORY and STOCK UPDATE through DC/UCF.

Additional clauses authorize DGS to change the OLQ INTERRUPT option and grant DGS IDMS authority. The OLQ DEFAULT OPTIONS clause specifies display of FILLER fields and PARTIAL lines. The class/attribute clause associates the LIBRARY class with the attribute PRIVATE. The relational-key clause associates user MRS with user DGS.

```
add user name is dgs
    prepared by dba password is 'ice 9'
    password is sgd
    full name is 'dianna g. smith'
    user description is programmer
    within user development
    of system inventory
    of system stock-update
    access to subschema invbasea of schema invbase version 2
           signon qfile is invon version 2
    access to subschema invbasea of schema invbase
           signon qfile is invon
    access to system inventory
    access to system stock-update
    optional interrupt
    olq default options filler partial
    authority for display is idms
    authority for update is password
    library is private
    'other developer' is mrs.
```

The MODIFY statement changes the password for the user DGS:

```
modify user dgs
    prepared by dgs password is sgd
    password is gsd.
```

USER-DEFINED ENTITY

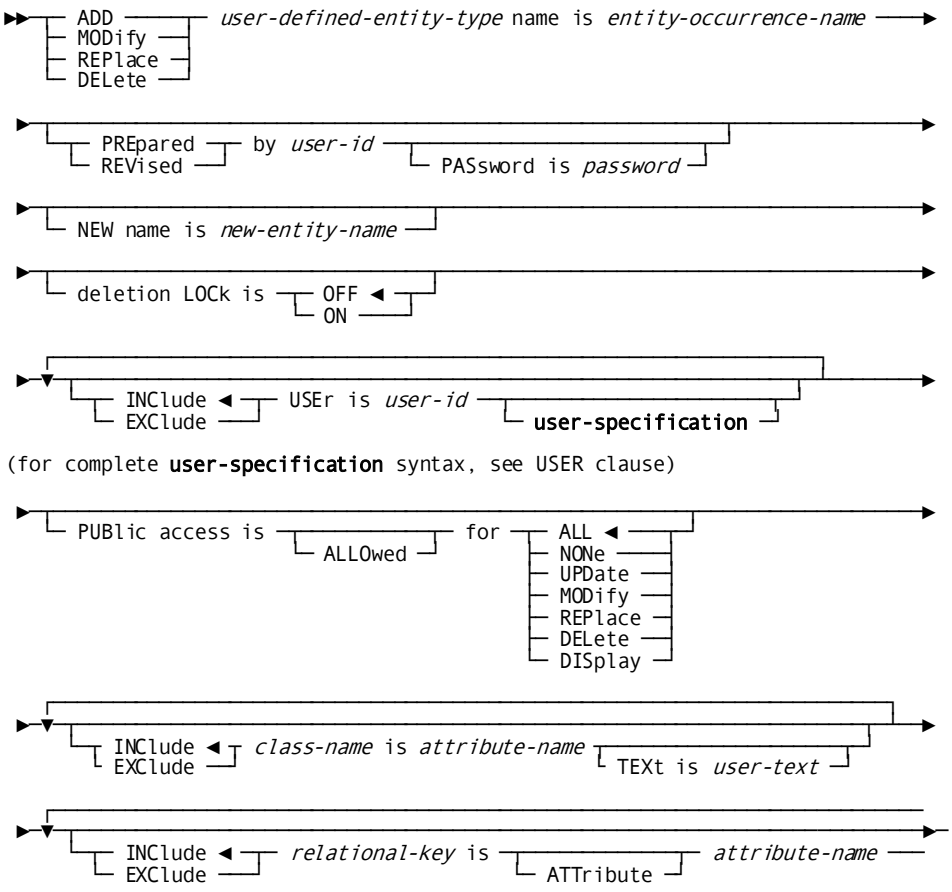
User-defined statements are used to directly establish user-defined entities in the dictionary. Optional clauses relate user-defined entities to other user-defined entities and to classes and attributes.

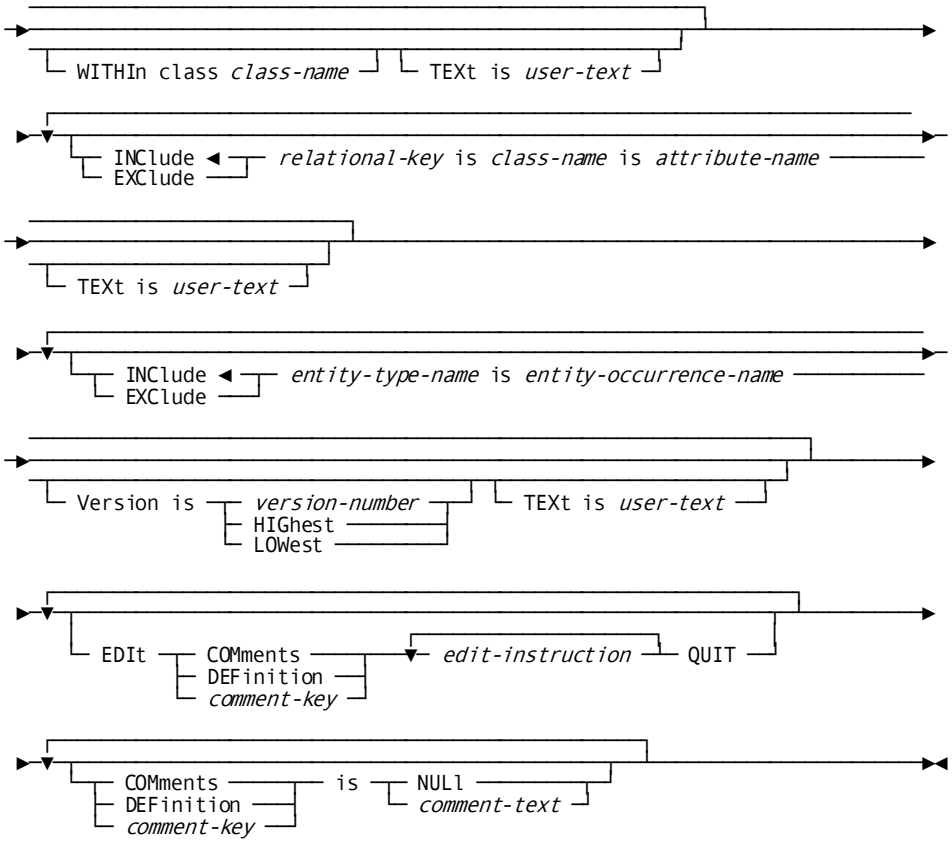
User-defined entities are established as classes by using the CLASS TYPE IS ENTITY clause of the CLASS statement. Statements for establishing and maintaining occurrences of user-defined entity types are similar to the ADD and MODIFY ATTRIBUTE statements. Once established, user-defined entities can be referenced by using any syntax that applies to classes and attributes.

If the SET OPTIONS statement specifies SECURITY FOR CLASS AND ATTRIBUTE IS ON, the user must be assigned the proper authority to issue user-defined entity statements.USER-DEFINED ENTITY.

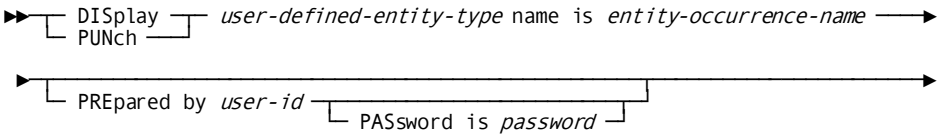
Syntax

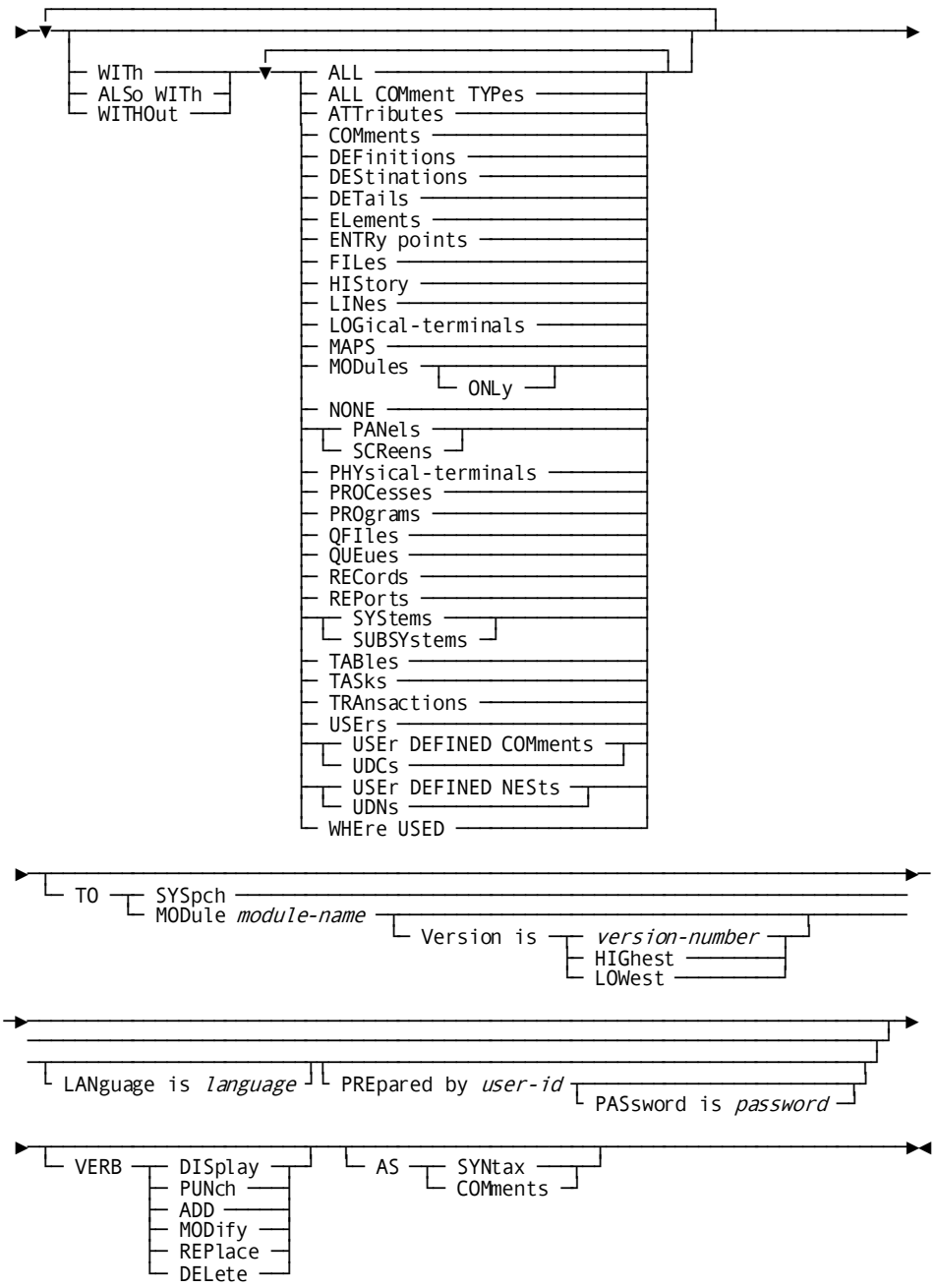
USER-DEFINED ENTITY statement



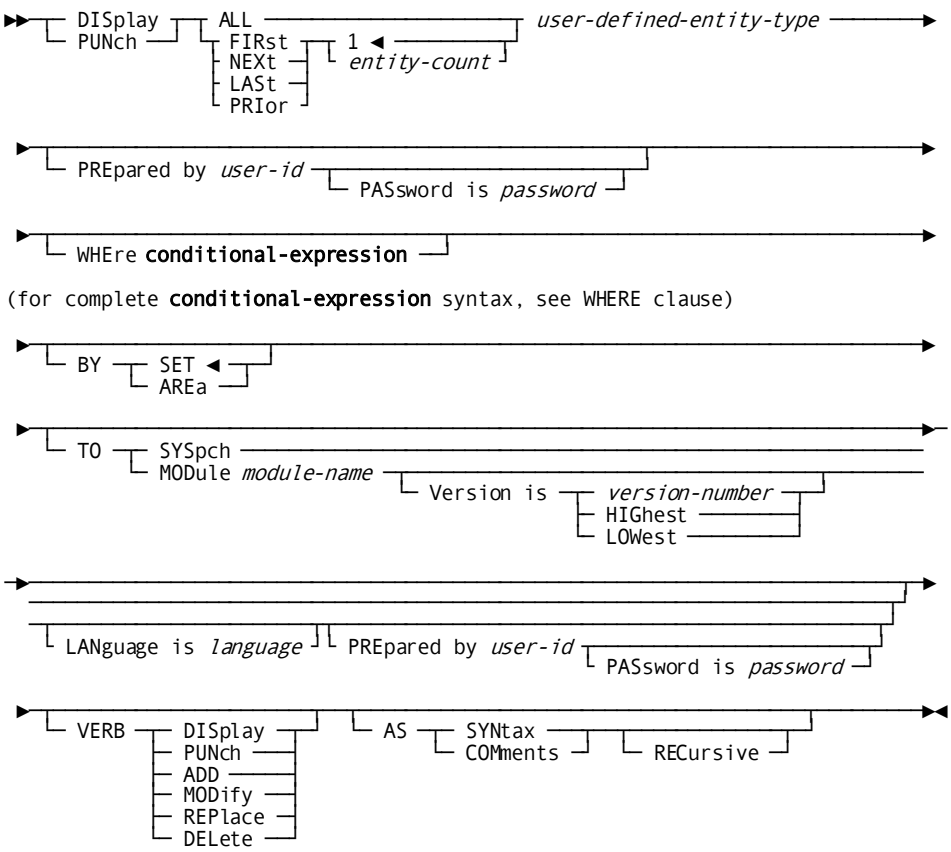


DISPLAY/PUNCH user-defined entity (for a single entity)




WITh
ALSo WITh
WITHOut
ALL
ALL COMment TYPes
ATTributes
COMments
DEFinitions
DEStinations
DETails
ELements
ENTRy points
FILes
HIStory
LINes
LOGical-terminals
MAPS
MODules
ONLy
NONE
PANels
SCReens
PHYsical-terminals
PROCesses
PROgrams
QFILes
QUEues
RECords
REPorts
SYStems
SUBSYstems
TABles
TASks
TRAnsactions
USErs
USEr DEFINED COMments
UDCs
USEr DEFINED NESts
UDNs
WHEre USED
TO
SYSpch
MODule module-name
Version is
version-number
HIGhest
LOWest
LANguage is language
PREpared by user-id
PASsword is password
VERB
DISplay
PUNch
ADD
MODify
REPlace
DELete
AS
SYNtax
COMments

DISPLAY/PUNCH user-defined entity (for multiple entities)



Parameters

user-defined-entity-type name is entity-occurrence-name

Identifies a new user-defined entity to be established in the dictionary, or an existing user-defined entity to be modified, replaced, deleted, displayed, or punched. User-defined-entity-type must be the 1- through 20-character name of a class defined with the CLASS TYPE IS ENTITY option.

Entity-occurrence-name must be a unique 1- through 40-character name within user-defined entity-type-name.

NEW NAMe is new-entity-name

Specifies a new name for the requested user-defined entity. New-entity-name must conform to the rules for entity-occurrence-name presented above. This clause changes only the name of the named entity;

it does not alter or delete any previously defined relationships in which the entity participates. Subsequent references to the entity must specify the new name. Note that the user-defined entity occurrence cannot be

renamed if DELETION LOCK IS ON (described below) is specified.

deletion LOCk is

Enables or disables the entity deletion lock.

OFF

Disables the deletion lock; the user can delete or rename the entity occurrence directly. OFF is the default.

ON

Enables the deletion lock; the user cannot delete or rename the entity. If DELETION LOCK IS ON is specified, MODIFY user-defined-entity DELETION LOCK IS OFF must be specified to delete or rename the

requested entity occurrence.

relational-key is ATTribute attribute-name

Associates the named entity with another entity through a previously defined relational key. The optional keyword ATTRIBUTE must be specified if the named class is defined with the same name as the attribute.

WITHIn class class-name

Uniquely identifies an established attribute. This parameter must be specified if the named attribute does not uniquely identify an established attribute. Class-name must match the name of a previously defined class.

relational-key is class-name is attribute-name

Associates an occurrence of a class with an occurrence of an attribute or user-defined entity through a previously defined relational key.

entity-type-name is entity-occurrence-name

Associates (INCLUDE) the named entity occurrence with or disassociates (EXCLUDE) it from an occurrence of the specified entity type.

WITh/ALSo WITh/WITHOut

Includes or excludes the specified options when the user-defined entity is displayed or punched. Detailed information for each DISPLAY/PUNCH option is under 3.4.2, “SET OPTIONS Syntax” on page 39. The options that are listed below present special considerations for this entity type.

DETails

Includes the DELETION LOCK specification.

ATTributes

Includes all user-defined entities to which the named user-defined entity is related.

Example:

In the following example, the user-defined entity types DEPARTMENT and EMPLOYEE are established in the dictionary by means of the CLASS TYPE IS ENTITY clause of the CLASS statement.

DOCUMENTATION and JMP are added as occurrences of DEPARTMENT and EMPLOYEE, respectively. The employee's birth date and date of hire can be added by relating two occurrences of DATE to an occurrence of EMPLOYEE.

```
add class department
class type is entity.

add class employee
class type is entity.

add department documentation.

add employee jmp
department documentation.

add class date
attributes are automatic
class type is entity.

modify entity attribute
user defined nest is hire
user defined nest is birth.

add employee tlm
department is personnel
birth date is 7/5/52
hire date is 2/2/82.
```

The MODIFY PROGRAM statement relates the predefined program PAYROLL to the user-defined entity occurrence EMPLOYEE TLM:

```
modify program payroll
employee is tlm.
```

Chapter 6: Online DDDL Compiler

This section contains the following topics:

Overview (see page 387)
Screen Format (see page 388)
Online Sessions (see page 389)
Online Commands (see page 393)
Program Function Keys Assigned to Operations (see page 396)

Overview

The DDDL compiler can be executed online to process requests to add, modify, replace, delete, display, and punch entity-occurrence definitions. The online DDDL compiler uses the same syntax as the batch DDDL compiler and provides a uniform screen for manipulating entity-occurrence definitions; separate maps are not required.

Full-screen and Line Modes

You can enter online requests in 3270 full-screen mode or in TTY line mode through:

- CA IDMS/DC
- CA IDMS UCF
- CICS
- TSO
- z/VM

The DDDL compiler supports large- and wide-screen 3270-type terminals.

Full-screen Mode

In full-screen mode, the online DDDL compiler employs a *text editor* that operates independently of the compiler. The text editor writes input to and output from the DDDL compiler to a *work file* associated with each session. The work file can contain multiple pages of compiler input or output; a page is equivalent to the number of lines on the terminal's screen. The user manipulates the contents of the work file by using *online text editing commands*. The ability to display and modify the contents of the work file allows the user to edit compiler output and resubmit it as input.

This chapter describes the format of the online IDD screen, how to conduct an online IDD session, the online commands that can be used during the session, and the PF keys assigned to various operations.

Screen Format

In full-screen mode, the online DDDL compiler uses a standard screen that has:

- **A preformatted top line**
- **An unformatted input/output area**

```
   END&sub1.      IDD nn.n ONLINE&sub2.    NO ERRORS&sub3.    DICT=EDUCDICT&sub4.
1/69&sub5.

  DISPLAY REC REC-LAYOUT.
  *+   ADD
  *+   RECORD NAME IS REC-LAYOUT VERSION IS 1
  *+       DATE CREATED IS      mm/dd/yy
  *+       TIME LAST UPDATED IS 07491666
  *+       PREPARED BY GDJ
  *+       RECORD LENGTH IS 240
  *+       PUBLIC ACCESS IS ALLOWED FOR ALL
  *+       RECORD NAME SYNONYM IS REC-LAYOUT VERSION 1
  *+           COPIED INTO MAP TOON-REC VERSION 1 WITHIN PANEL TOON-REC-OLMPANEL
  *+                VERSION 1
  *+           COPIED INTO MAP GER-OCCU VERSION 1 WITHIN PANEL GER-OCCU-OLMPANEL
  *+                VERSION 1
  *+       .
  *+       RECORD ELEMENT IS LINE-LAYOUT VERSION 1
  *+       LINE IS 000100
  *+       LEVEL NUMBER IS 02
  *+       USAGE IS DISPLAY
  *+       ELEMENT LENGTH IS 240
  *+       POSITION IS 1
  *+       .
  *+       SUBORDINATE ELEMENT IS CHAR-LAYOUT VERSION 1
  *+       LINE IS 000200
```

Top Line

The top line of the screen contains the following areas (areas are numbered in the sample screen above):

1. **Command area**—:ih1.command area Provides twenty spaces in columns 2 through 21 for entering commands to manipulate the work file and to communicate with the DDDL compiler; these commands are listed under Top-line Commands (see page 394) later in this chapter; they are described in detail in *CA IDMS Common Facilities Guide*.
2. **Name area**—Displays the name of the compiler and the release number.
3. **Message area**—Displays one of the following, as appropriate: the work-file page and line number; the literal NO ERRORS; the number of error messages issued for the compile; or a message describing the status of a top-line command.
4. **Dictionary area**—Displays either the name of the current dictionary (if other than the default) or the literal BLK, which signifies use of a line command.

5. **Line number area**—Displays one of the following, as appropriate: the literal EMPTY or the top (current) line of the screen I/O area, followed by the total number of lines (last line) in the work file.

Input/output Area

Below the screen's top line, the input/output area covers the remainder of the screen. The online DDDL compiler uses a line length of 79 regardless of the terminal width; however, the number of lines displayed varies based on the type of terminal in use.

Online Sessions

An online DDDL session begins when the user signs on to the compiler and ends when the user signs off from the compiler or otherwise terminates the session. The user can also suspend a session and transfer to another online CA IDMS software component.

The following considerations about online sessions are discussed in the following subsections:

- Beginning a session
- Conducting a session
- Terminating a session
- Session recovery

Beginning a Session

To begin an online session:

1. Sign on to the host teleprocessing (TP) monitor, according to site-standard procedures.
2. Enter one of the following:
   - Site-standard task code that invokes the online DDDL compiler; the installation default is IDD.
   - Site-standard task code that invokes the online DDDL compiler under the transfer control facility (TCF); the installation default is IDDT. For a complete description about using online IDD under TCF, refer to the *CA IDMS Common Facilities Guide* manual.
3. Optionally, enter the SIGNON command on the first line of the screen I/O area. If the SET OPTIONS statement specifies SECURITY FOR IDD SIGNON IS ON, SIGNON must be the first command issued in the session; see SIGNON Statement (see page 30) for additional information.
4. Optionally, enter a SET OPTIONS statement to establish session- or dictionary-specific processing options; see SET OPTIONS Statement (see page 34) for additional information.

You can also initiate an online DDDL compiler session from another online component by using the transfer control facility; for additional information, refer to the *CA IDMS Common Facilities Guide* manual.

Conducting an Online Session

The following table provides guidelines for conducting an online IDD session.

Guideline for:	Description
Types of statements	Enter ADD, MODIFY, REPLACE, DELETE, DISPLAY/PUNCH, and INCLUDE statements in the screen I/O area.
Cursor movement	Use the TAB, BACK TAB, and CURSOR keys to move the cursor around the screen I/O area and to position the cursor in the command area.
Text-editing commands	Use online text-editing commands, discussed in Line Commands (see page 395) later in this chapter, to manipulate the contents of the work file.

Guideline for:	Description
End-of-file indicator	:i1.end-of-file indicator Specify a logical end-of-file indicator to establish the point at which input from the work file to the DDDL compiler is to end. A default end-of-file indicator of **/*** is established during IDD installation. The user can change this indicator on a dictionary or session basis by using the SET OPTIONS statement EOF clause.
Suspending IDD	Transfer from online IDD to another online CA IDMS component, then resume the original IDD session, using the transfer control facility. The user moves from one online session to another by means of the top-line SWITCH command, described later in this chapter, and the transfer control facility Selection screen. (refer to the *CA IDMS Common Facilities Guide* manual.)

Input and Output Are Displayed

The online compiler displays each input statement, followed by the requested output. For example:

```
display first 2 records.
*+      display record name is cust-rec version is 1.
*+      display record name is cust-rec version is 2.
```

Error Handling

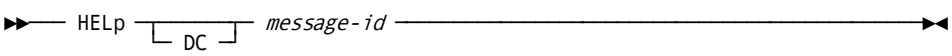
The online DDDL compiler responds to errors encountered in source input statements by:

- Indicating the total number of E-level errors from the most recent compiler execution in the *message area* of the screen.
- Listing error messages on the line immediately following the line in error. Each message is preceded by ***+**, which indicates that the text is commentary only; if the screen is resubmitted, the message text is ignored by the DDDL compiler.

Using HELP DC to Debug

To aid the debugging process, you can issue a HELP DC command to obtain a detailed online description of any error or warning message produced in a DDDL compiler run. The user must type the HELP DC command in the screen I/O area.

Syntax for the HELP DC command is as shown below, where *message-id* must be the 6-digit identifier associated with the error or warning message.



For example, to display information about message 601034, enter:

```
help 601034
```

or enter:

```
help dc601034
```

The online DDDL compiler responds with the ID, severity, text, and explanation associated with the message, as follows:

```
*+  E   DC601034  INVALID VERSION
*+
*+  An invalid version specification has been
*+  encountered. The version number is too
*+  long or contains nonnumeric characters.
*+  Supply a valid version number according
*+  to the syntax rules.
```

More information: For descriptions of all DDDL compiler messages, refer to *CA IDMS Messages and Codes Guide*.

Terminating a Session

To end an online session, choose one of the following options:

- **Enter SIGNOFF, LOGOFF, or BYE on the first line of the screen I/O area and press ENTER.**

  This terminates the full-screen text editor, deletes the contents of the work file, clears the default processing options established for the session, and displays the session transaction summary. After reviewing the transaction summary, press CLEAR.

- **Enter END in the command area and press ENTER.**

  This terminates the session and clears the contents of the work file; the transaction summary is not displayed. Control is returned directly to the host TP monitor.

Recovering a Session

Consider the termination situations shown in the following table when you recover a DDDL compiler session.

Termination situation	Effect
The DDDL compiler, central version, *or* DC/UCF system terminates abnormally during an online DDDL compiler session.	All updates made to the dictionary during the session remain intact. The contents of the work file and the default session options are lost.
The DDDL compiler terminates abnormally.	You can resume after recovering the session using the compiler task code. The session resumes at the point before which the last command was entered; text changes made to the last screen are applied to the work file.

Online Commands

The commands that direct an online session of the DDDL compiler and manipulate the contents of the work file fall into two categories:

- Top-line commands
- Line (or text-editing) commands

These commands are listed and described in the following two subsections.

Note: For a list of program function (PF) keys you can use as alternatives to top-line commands, see Program Function Keys Assigned to Operations (see page 396), later in this chapter.

For detailed information on using top-line commands, line commands, and PF keys, refer to the *CA IDMS Common Facilities Guide* manual.

Top-line Commands

Top-line commands are used to direct an online DDDL compiler session. The user enters top-line commands in the *command area* of the screen.

The top-line commands described in the following table are available for use with the online DDDL compiler. Note that all commands, with the exception of RESHOW, update both the screen and the work file.

Top-line command	Description
APPLY	Updates the screen and work file but does not execute the DDDL compiler.
CLEAR	Deletes all data contained in the work file.
DISPLAY LINE	Displays a page of the work file, starting with the specified line.
DISPLAY PAGE	Displays the requested page from the work file.
END	Immediately terminates the current session.
ENTER	Sets the ENTER key to execute the APPLY or the UPDATE command (described below).
ESCAPE	Establishes the escape character that must be used with line commands.
FIND	Locates a character string by searching forward or backward in the work file.
HELP	Lists each top-line command and the PF key currently assigned to execute that command.
INSERT	Inserts lines into the work file after the line at which the cursor is positioned.
PRINT (DC/UCF only)	Prints the contents of the work file.
REPEAT	Repeats the line at which the cursor is positioned.
RESHOW	Cancels all changes made to the current screen and redisplays the previous screen.
SUSPEND	Suspends the current session and returns control to the host TP monitor.
SWAP	Restores the screen and the work file to their condition prior to the last execution of the compiler.

Top-line command	Description
SWITCH (only if the DDDL compiler is executing under the control of the Transfer Control Facility)	Suspends the session and transfers control to the specified online CA IDMS component or to the transfer control facility Selection screen.
UPDATE	Updates the work file and executes the DDDL compiler.

Abbreviating Top-line Commands

You can abbreviate top-line commands to a minimum of three characters, except for:

- FIND which can be abbreviated to F
- PRIOR which can be abbreviated to PRIO (four characters distinguish it from the keyword PRINT)

To enter a top-line command, either type the command on the top line of the screen and press ENTER or use the program function (PF) key assigned to the desired function (see Program Function Keys Assigned to Operations (see page 396)).

Line Commands

Line commands, also called text-editing commands, are used to copy, delete, move, and repeat lines or blocks of lines within a work file.

How to Enter a Line Command

A line command consists of a one- to three-character value followed by a space.

These commands must begin with the *escape character* (%) which signals to the text editor that the line contains a command. Enter a command in column 1 of the line to which it applies, and end the command with a space. For detailed information on using line commands, refer to the *CA IDMS Common Facilities Guide* manual. Line commands are listed in the following table. Note that:

- The percent sign (%) is the default escape character.
- *n* represents the number of lines (including the current and subsequent lines) to which the operation applies.
- The **(space)** represents the mandatory space that must follow each line command.

Operation	Command format
After	%A (space)
Before	%B (space)

Operation	Command format
Copy	Copy a single line: ■ %C*n* (space) Copy a block of lines: ■ %CB (space) (on the first line of the block) ■ %CE (space) (on the last line of the block)
Delete	Delete a single line: ■ %D*n* (space) Delete a block of lines: ■ %DB (space) (on the first line of the block) ■ %DE (space) (on the last line of the block)
Move	Move a single line: ■ %M*n* (space) Move a block of lines: ■ %MB (space) (on the first line of the block) ■ %ME (space) (on the last line of the block)
Repeat	Repeat a single line: ■ %R*n* (space) Repeat a block of lines: ■ %RB (space) (on the first line of the block) ■ %RE (space) (on the last line of the block)

Program Function Keys Assigned to Operations

Program function (PF) keys can be used as an alternative to typing top-line commands. To display the current PF-key assignments, use the top-line HELP command.

The following table lists the PF keys established as installation defaults for the DDDL compiler.

PF key	Corresponding online command and description
PF1, PF13 PF8, PF20	DISPLAY PAGE NEXT Scrolls forward one page
PF2, PF14 PF7, PF19	DISPLAY PAGE PRIOR Scrolls backward one page

PF key	Corresponding online command and description
PF3, PF15	DISPLAY LINE NEXT Scrolls forward one line
PF4, PF16	INSERT Inserts up to a full screen of lines
PF5, PF17	APPLY Updates screen contents and work file but does not invoke the compiler
PF6, PF18	UPDATE Updates work file and executes the compiler
PF9, PF21	SWAP Restores work-file contents
PF12, PF24	PRINT Prints work-file contents (DC/UCF only)
PA1	Cancel FIND Cancels the FIND command
PA2	RESHOW Cancels changes to the current screen and redisplays the screen
CLEAR	CLEAR Clears the work file
ENTER=APPLY	Updates the screen and work file
ENTER=UPDATE	Updates the work file and executes the compiler

Chapter 7: IDD Menu Facility

This section contains the following topics:

Overview (see page 399)
Screen Formats (see page 400)
Using Menu Facility Screens (see page 405)
Online Commands (see page 409)
Conducting a Menu Facility Session (see page 411)
Descriptions of IDD Menu Facility Screens (see page 419)
Sample Session (see page 436)

Overview

An alternative to freeform online IDD input, the IDD menu facility guides you through a series of standard, fixed-format screens. The menu facility supports all basic non-teleprocessing DDDL compiler options and entity-type syntax except REPORTS, TRANSACTIONS, and ENTRY POINTS.

Because the IDD menu facility calls the DDDL compiler, the entity types and parameters that apply to the batch and online IDD environments apply to the menu facility as well. Only the method of input is different; menu facility screens present the available options and provide fields in which to input the definition.

This chapter presents how to use the IDD menu facility to define an entity occurrence in the dictionary.

The following topics are discussed:

- Screen formats
- Using menu facility screens
- Online commands
- Conducting a menu facility session
- Descriptions of IDD menu facility screens

A sample session follows at the end of the chapter.

More information: For complete definitions of parameters listed on screens, see Chapters 1 through 4 of this manual.

Screen Formats

IDD menu facility features two types of screen design: *fixed* (nonpageable) and *pageable*. The two screen types are discussed separately in this section.

Fixed Screens

Fixed screens provide session, entity-occurrence, and control-key information.

The following is an example of a fixed screen.

Example of a Fixed Screen

```
     IDD REL nn.n           *** RECORD ENTITY ***                        RECD
➔
                    RECORD 'DEPARTMENT' VERSION 1 DISPLAYED

X DISPLAY      RECORD NAME.....: DEPARTMENT
_ MODIFY
_ ADD          VERSION NUMBER..: 1       _ HIGHEST    _ NEXT HIGHEST
_ DELETE                                 _ LOWEST     _ NEXT LOWEST

               DESCRIPTION.....:

               RECORD LENGTH...: 56




_ RELM  =  RECORD ELEMENTS   <PF9>          _ COBL  =  COBOL ELEMENTS    <PF11>
_ RELL  =  REC ELEMENT LIST  <PF10>         _ RECX  =  RECORD EXTENSION
_ REGN  =  USER REGISTRATION <PF2>          _ PUBL  =  PUBLIC ACCESS     <PF3>
_ CLAT  =  CLASS/ATTRIBUTES  <PF4>          _ RKEY  =  RELATIONAL KEYS   <PF5>
_ COMM  =  COMMENTS          <PF6>          _ COML  =  COMMENT KEY LIST  <PF7>
_ HIST  =  HISTORY           <PF8>          _ COPY  =  COPY FROM/SAME AS
_ XREF  =  CROSS REFERENCE                  _ HELP  =  HELP              <PF1>
```

Three Areas of a Fixed Screen

The IDD menu facility fixed screens are divided into three areas:

- Heading and message area
- Specification area
- Screen selection area

Heading and Message Area

The heading and message area contains a preformatted first line, the command area, and the message line:

- The first line contains the simulated PF-key field, the product name, the release number, the screen title, and the screen name. The screen title identifies the screen; the screen name is a 4-character symbol used to reference the screen.

  Note: The simulated PF-key field is an untitled 2-character field that can be used if the terminal is not equipped with program function (PF) keys. For additional information about this field, see Predefined Control Keys (see page 405) later in this chapter.

- The second line contains an arrow pointing to the command area. The command area can be used to:
  - Move from one screen to another by typing in a screen name
  - Leave the session by entering DDDL top-line commands such as SUSPEND

  Use of the command area is described in detail under Conducting a Menu Facility Session (see page 411) later in this chapter. For a list of top-line commands, see Top-line Commands (see page 394).
- The message line prompts you for additional information or actions, indicates that your response has been processed, or explains why information has not been processed. The message line is described in detail in Message Display and Field Highlighting (see page 407) later in this chapter.

Specification Area

The specification area contains fields that identify and define an entity occurrence or signon information.

Screen Selection Area

The screen selection area lists subordinate screens that are available in order to select the next action. The selection area is formatted with entity-specific options on the first lines and general options on the lower lines. For additional information, see Navigating Screens (see page 413), later in this chapter.

Pageable Screens

Pageable screens allow menu facility users to submit many source statements or options to the DDDL compiler. Pageable screens are identified by the upper right corner, which displays page and line numbers in the following format:

PAGE *page-number* LINE *line-number*

The following is an example of a pageable screen.

Example of a Pageable Screen

```
     IDD REL nn.n            *** COBOL ELEMENTS ***                      COBL
 ➔                                        PAGE 1 LINE 1                  1/36
                          RECORD 'DEPARTMENT' VERSION 1

 ---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----
     02 DEPT-ID
         PICTURE IS  9(4)
         USAGE IS DISPLAY
*+       ELEMENT LENGTH IS 4
*+       POSITION IS 1
         ELEMENT NAME SYNONYM IS DEPTID
             FOR RECORD SYNONYM DEPARTMT VERSION 1
         ELEMENT NAME SYNONYM IS DPID
             FOR RECORD SYNONYM DEPT VERSION 1
         .
     02 DEPT-NAME
         PICTURE IS  X(45)
         USAGE IS DISPLAY
*+       ELEMENT LENGTH IS 45
*+       POSITION IS 5
         ELEMENT NAME SYNONYM IS DEPTNAME
             FOR RECORD SYNONYM DEPARTMT VERSION 1
         ELEMENT NAME SYNONYM IS DPNAME
             FOR RECORD SYNONYM DEPT VERSION 1
```

Two areas of a Pageable Screen

IDD menu facility pageable screens are divided into two areas:

- Heading and message area
- Specification area

Heading and Message Area

The heading and message area contains a preformatted first line, the command area, and the message line:

- The first line contains the simulated PF-key field, the product name, the release number, the screen title, and the screen name. The screen title identifies the screen; the screen name is a 4-character symbol used to reference the screen.

Note: The simulated PF-key field is an untitled 2-character field that can be used if the terminal is not equipped with program function (PF) keys. For additional information about this field, see Predefined Control Keys (see page 405) later in this chapter.

- The second line contains an arrow pointing to the command area. The command area can be used to:
  - Move from one screen to another by typing in a screen name.
  - Manipulate the definition in the dictionary or leave the session by entering DDDL top-line commands such as SUSPEND or REPEAT.

  Use of the command area is described in detail under Conducting a Menu Facility Session (see page 411) later in this chapter. For a list of top-line commands, see Top-line Commands (see page 394).
- The message line prompts you for additional information or actions, indicates that your response has been processed, or explains why information has not been processed. The message line is described in detail in Message Display and Field Highlighting (see page 407) later in this chapter.

Specification Area

The specification area contains screen-specific information according to the type of pageable screen. For example, in a pageable screen that contains text, the specification area is unformatted. In a pageable screen that identifies relationships between two entities, the specification area is formatted with blocks of lines that represent a relationship.

Two Types of Pageable Screens

There are two types of pageable screens:

- Screens that contain text
- Screens that identify relationships between two entities

The following text describes how paging works for each of the two screen types.

Screens that contain text

On these screens, each line of text is considered one line of data. As with online IDD screens, text is typed in freeform style. A column scale at the top of the screen makes formatting easier.

Examples of screens that contain text:

- Comments screen
- Module Source screen

Screens that identify relationships

On these screens, each group of lines that represents an entity is considered one line of data.

Examples of screens that identify relationships:

- Within Systems screen
- Record Elements screen

Using Menu Facility Screens

The IDD menu facility provides the following features that allow efficient access and use of screens:

- Predefined control keys
- Cursor positioning
- Message display and field highlighting
- Default value assignment
- Help screens

These features and the functions they perform are explained in this section.

Predefined Control Keys

Control keys and their associated functions are predefined under the IDD menu facility. Control key assignments fall into two categories: global and local. These two categories are discussed, followed by a description of using PF-key simulation (for terminals with no PF keys).

Global Control Keys

Global control keys always perform the same functions during a menu facility session. Menu facility global-control key assignments correspond to online IDD control-key assignments. Online IDD control-key assignments are established during IDD installation or with the system generation compiler.

Installation defaults for global control keys are shown in the following table.

Control key	Description
ENTER	Submits information through the IDD menu facility for processing.

Control key	Description
CLEAR	Displays the previous screen. You can press CLEAR from the Master Selection screen to leave the IDD menu facility and return to the DC/UCF system.
PA1	Clears fixed screens; on pageable screens, PA1 cancels a FIND command.
PA2	Refreshes the screen. Pressing PA2 cancels changes just typed on the screen. The screen is rewritten as it appeared the last time you pressed a control key.
PF1	Invokes the HELP function of the IDD menu facility; the appropriate help screen is displayed for the current IDD option or entity type. For further information about help screens, see Help Screens (see page 409) later in this chapter.
PF7 (pageable screens only)	Scrolls backward to the previous page of the pageable screen.
PF8 (pageable screens only)	Scrolls forward to the next page of the pageable screen.

Local Control Keys

The screen that a local control key invokes depends on the screen from which it is pressed. Each screen selection area lists the local control-key assignments for that screen. For example, PF7 invokes the Record Entity screen from the Master Selection screen.

PF Key Simulation

On terminals that have no PF keys, you can perform PF-key related functions by using PF-key simulation. To activate PF-key simulation, type any character in the PF-KEY SIMULATION ON field on the Master Selection screen. To request the next action from any menu facility screen, type a 1- or 2-digit PF key number in the simulated PF key field and press ENTER. For example, to perform the function associated with PF2, type 2 in the simulated PF-key field and press ENTER.

Cursor Positioning

On all IDD menu facility screens, you can enter or change information only in response fields. To move the cursor quickly between menu facility response fields, the cursor control keys can be used in conjunction with the tab, back tab, or return keys.

When the cursor is positioned at a response field, you can type in new information or replace existing information. The space bar or the erase end-of-field key can be used to delete characters across the response field. The cursor control keys can be used to move across the response field without deleting characters.

Message Display and Field Highlighting

The IDD menu facility displays messages and highlights fields in response to the use of a screen.

Message Display

Messages are displayed on the message line. They:

- Request that additional information be supplied
- Inform you about the next required action
- Confirm the results of current processing
- Indicate why information has not been processed

Field Highlighting

Highlighting is used in conjunction with a message to reference a related field on the screen. The IDD menu facility highlights fields that have been modified or those that are in error. Screen names and the cursor position are also highlighted.

Error Display Screen

When the menu facility returns a message that appears to be truncated, you can request the Error Display screen by typing ERRS in the command area. The Error Display screen displays the IDD DDDL syntax that corresponds to the requested entity options; error messages are listed immediately following each line in error. Error messages consist of the message identifier, the line (CARD) number and word that caused the processing error, and the message text.

After identifying the error, you can return to the screen on which the error occurred by pressing the CLEAR key. You can then correct the error and resubmit the entity options to the compiler.

Default Value Assignment

The IDD menu facility automatically supplies default values in many response fields. These default values allow you quick access to information most likely to be used in response fields.

You can accept a default value or override it by typing the appropriate value over the default. In most cases, even if the new value is placed in a separate response field from the default field, IDD menu facility automatically recognizes the new information and ignores the default value.

Example of Overriding a Default

For example, on the Display All screen, the default value for the COMPARISON ACTION field is EQ (equal). To override the default, you type an x in the CONTAINS field.

```
     IDD REL nn.n         *** DISPLAY ALL ***                        DISP
➔


DISPLAY PROCESSING ORDER..:X FIRST  _ NEXT   _ LAST   _ PRIOR  _ ALL

NUMBER OF OCCURRENCES.....:20

OF ENTITY TYPE............:_ ATTRIBUTES       _ PROCESSES        _ TABLES
                           _ CLASSES          _ PROGRAMS         _ USERS
                           x ELEMENTS         _ QFILES           _ MESSAGES
                           _ FILES            _ RECORDS
                           _ MODULES          _ SYSTEMS
                           _ MODULES ONLY (WITHOUT PROCESSES, QFILES, TABLES)
                           _ USER DEFINED ENTITY...:

   WHERE 'VALUE'/FIELD....:name

       COMPARISON ACTION..:X EQ   _ NE   _ GT   _ GE   _ LT   _ LE
                           x CONTAINS    _ MATCHES

          'VALUE'/FIELD....:'w-emp'
       OR 'VALUE'/FIELD....:
       OR 'VALUE'/FIELD....:
```

Help Screens

The IDD menu facility features an online help screen for each menu facility screen. Each help screen contains the following:

- Table of contents
- List of global control keys
- Description of screen usage
- Special rules, if any
- List of screen titles and names
- Overview of the IDD menu facility

Requesting a Help Screen

To request a help screen, choose one of the following options:

- Type HELP in the command area and press ENTER.
- Press PF1.
- If applicable, type a nonblank character in the appropriate field in the screen selection area and press ENTER.

Exiting a Help Screen

To exit a help screen and return to the previous screen, press CLEAR.

Online Commands

The commands that direct an IDD menu facility session and manipulate the contents of pageable screens fall into two categories: *top-line commands* and *line* commands. These types of commands are described in this section.

Note: For complete information regarding the menu facility online commands, see *CA IDMS Common Facilities Guide*.

Top-line Commands

Top-line commands are used to direct a menu facility session. You enter top-line commands in the command area on any screen.

To enter a top-line command, either type the command in the command area and press ENTER or use the assigned global control key. For a list of control keys, see Predefined Control Keys (see page 405) earlier in this chapter. For an in-depth discussion of menu facility top-line commands, refer to *CA IDMS Common Facilities Guide*.

The following table lists and describes the top-line commands available to the IDD menu facility. An asterisk (*****) identifies those commands which apply to *pageable screens only*.

Top-line command	Description
END	Immediately terminates the current session.
HELP	Invokes the help tutorial associated with the current screen (refer to Help Screens (see page 409) for additional information).
SUSPEND	Suspends the current session and returns control to the host TP monitor.
SWITCH (only if the IDD menu facility is executing under the control of the transfer control facility)	Suspends the session and transfers control to the specified online CA IDMS component or to the transfer control facility selection screen.
APPLY *	Updates the screen but does not submit the screen to the DDDL compiler.
DELETE ALL *	Deletes all information contained in all pages of the pageable screen.
DISPLAY LINE *	Displays the requested line at the top of the screen.
DISPLAY PAGE *	Displays the requested page of the pageable screen.
ENTER *	Sets the ENTER key to execute the APPLY or the UPDATE command.
ESCAPE * (pageable screens that contain text only)	Establishes the escape character that must be used with line commands.
FIND *	Locates a character string by searching forward or backward within the pageable screen.
INSERT *	Inserts new text or definitions into the pageable screen.

Top-line command	Description
REPEAT * (pageable screens that contain text only)	Repeats the line at which the cursor is positioned.
UPDATE *	Updates the screen and invokes the DDDL compiler.

Line Commands

Line commands (also called text-editing commands) move, copy, delete, or repeat a line or group of lines in a pageable screen that contains text or source code statements.

Note: For a list of line commands that apply to the IDD menu facility, see Line Commands (see page 395).

Line command syntax and rules are fully documented in the *CA IDMS Common Facilities Guide* document.

Conducting a Menu Facility Session

An IDD menu facility session begins when you invoke the IDD menu facility at the ENTER NEXT TASK CODE system prompt. The session ends when you exit the menu facility and returns control to the system.

To conduct a menu facility session, you should understand the following activities:

- Beginning a session
- Navigating screens
- Displaying entity occurrences
- Adding entity occurrences
- Modifying entity occurrences
- Deleting entity occurrences
- Terminating a session

Each of these activities is discussed separately in this section.

Beginning a Session

To begin an IDD menu facility session:

1. Sign on to the system.
2. Enter the task code that invokes the menu facility. The installation default is IDDM. The menu facility displays the *Master Selection* screen.
3. Sign on to the menu facility by using one of the following methods:
   - Press the ENTER key. The menu facility automatically accepts the signon information.
   - Explicitly provide the appropriate information and press ENTER.

     This method must be used in the following situations:
     - IDD has been used to establish security for the dictionary to be accessed by the menu facility. You must identify the user name and, optionally, a password. The identified user must have been assigned IDMS-DC authority through the AUTHORITY clause of the USER statement (see USER (see page 358)).
     - A DDS user needs to specify the node name of the central version that controls the dictionary to be accessed by the menu facility.
     - You want to access a dictionary other than the default dictionary.
     - You want to override the default usage mode (shared update) in which the DDDL compiler is to access the dictionary.

After the menu facility confirms a successful signon, you can establish session-specific processing options by selecting the *Session Options* screen. To select this screen, either type the screen name OPTI in the command area of the Master Selection screen or type any nonblank character in the appropriate field in the screen selection area.

More information: For more information about each processing option offered on the Session Options screen, see SET OPTIONS Statement (see page 34). For additional information about requesting screens, see Navigating Screens (see page 413).

Navigating Screens

The IDD menu facility consists of the *Master Selection* screen and *subordinate screens* arranged in a hierarchical structure. To implement a definition in the dictionary, you must navigate through the menu facility, choosing the next screen.

Screen Descriptions

The following table shows the names of screens and what can be done from each screen.

Screen	Description
Master Selection screen	This screen is at the top of the IDD menu facility structure. From the Master Selection screen, you can: ■ Transfer to an Entity screen to process a definition ■ Transfer to the Session Options screen or the Display All screen ■ Press CLEAR to terminate the session
Entity screens	These screens identify the entity type and occurrence to be defined. From an Entity screen, you can: ■ Transfer to a subordinate screen to further define the entity occurrence ■ Transfer to another Entity screen to begin another definition ■ Go back to the Master Selection screen
Subordinate screens	These screens complete the entity-occurrence definition. From a subordinate screen, you can: ■ Transfer to a lower level subordinate screen to further define the entity occurrence ■ Transfer to any screen on the same level within the entity occurrence ■ Return to a higher level screen within the same entity occurrence ■ Return back to the current Entity screen or to any other Entity screen to begin another definition

More information: For more information on screens, see Descriptions of IDD Menu Facility Screens (see page 419) later in this chapter.

Selecting Screens

You can select screens using any of the following methods:

- Enter any character at the underscore that immediately precedes a screen name listed in the *screen selection area* and press ENTER.
- Enter a screen name in the *command area* and press ENTER.
- Press the appropriate *global or local control key*.
- Type the PF-key number in the *simulated PF-key field* and press ENTER.
- Press CLEAR to return to the prior screen.

  Note: Returning to the prior screen by pressing CLEAR does not process changes on the current screen. To apply any changes to the definition, be sure to press ENTER before pressing CLEAR.

Considerations for entering a screen name

The following considerations apply when you enter a screen name to select the next screen:

- You can request an entity screen by typing the DDDL entity type, rather than the screen name. For example, to access the RECD screen, you can type RECORD in the command area. The alternatives are listed as follows:

Screen name	DDDL Entity-type name
ATTR	ATTRIBUTE
CLAS	CLASS
ELEM	ELEMENT
FILE	FILE
MSGS	MESSAGE
MODU	MODULE
PROC	PROCESS
PROG	PROGRAM
QFIL	QFILE
RECD	RECORD
SYST	SYSTEM
TABL	TABLE

Screen name	DDDL Entity-type name
USER	USER
ENTY	Not applicable

- To eliminate keystrokes when selecting an entity occurrence, you can do either of the following:
  - Type only the first three letters of the screen name or the DDDL entity type. Certain screens, such as PROC or PROG, must be invoked using the first *four* letters to ensure accuracy. For example, to request the Message Entity screen, type MSG in the command area.
  - Type the screen name or DDDL entity type followed by the name of the entity occurrence. For example, to display the DENTAL-CLAIM record, you type REC DENTAL-CLAIM in the command area.

    The entity-occurrence name consists of one parameter; version number or other qualifiers are not acceptable. When more than one entity occurrence exists in the dictionary, the menu facility displays the default version number of the requested occurrence.

Displaying Entity Occurrences

To display an entity occurrence previously defined in the dictionary:

1. Request the appropriate Entity screen.
2. Enter the entity-occurrence name (and version number, if appropriate).
3. Press ENTER.

The default verb is DISPLAY. The menu facility automatically recognizes the request, displays the definition on the Entity screen, and returns a confirmation message on the message line. You can request subordinate screens to display the complete entity-occurrence definition.

Adding Entity Occurrences

To add a new entity-occurrence definition to the dictionary:

1. Request the appropriate Entity screen.
2. Enter the entity-occurrence name (and version number, if appropriate).
3. Type a nonblank character at the underscore that immediately precedes the ADD literal.
4. Press ENTER.

The menu facility automatically recognizes that the ADD request overrides the default DISPLAY request, submits the syntax to the DDDL compiler, and returns a message confirming that the definition has been added to the dictionary.

You can continue to define the entity occurrence by requesting subordinate screens. The menu facility automatically updates the new entity-occurrence definition in the dictionary when you request a subordinate screen and/or press ENTER.

Modifying Entity Occurrences

To modify an entity occurrence previously defined in the dictionary:

1. Request the appropriate Entity screen.
2. Enter the entity-occurrence name (and version number, if appropriate).
3. Press ENTER to display the occurrence to be modified.

   For example, to modify a record called CUST-REC, the user first displays the occurrence by requesting the Record Entity screen and typing in the record name CUST-REC.
4. Choose one of the following actions:
   - If the information to be modified is located *on the Entity screen*: change the information, type a nonblank character at the underscore that immediately precedes the MODIFY literal, and press ENTER.
   - If the information to be modified is located *on a subordinate screen*, transfer to the screen that contains the information to be modified and submit the changes.

Updating relationships between entity occurrences

On some subordinate screens (for example, the User Registration screen), you can update relationships between the named entity occurrence and other entity occurrences:

- To *replace* one relationship with another, you can type the new definition over the obsolete definition.
- To *delete* or *disassociate* a definition from the named occurrence, you can display the obsolete definition, then perform one of the following procedures:
  - Erase, space over, or blank out the related occurrence name and press ENTER.
  - Type a nonblank character at the underscore that immediately precedes the EXCLUDE option of the appropriate definition and press ENTER.

Deleting Entity Occurrences

To delete an entity occurrence previously defined in the dictionary:

1. Request the appropriate Entity screen.
2. Enter the name (and version number, if appropriate) of the entity occurrence to be deleted.
3. Type a nonblank character at the underscore that immediately precedes the DELETE verb.
4. Press ENTER.

The menu facility automatically recognizes the request, deletes the definition from the dictionary, and returns a confirmation message that the occurrence has been deleted.

Terminating a Session

To end an IDD menu facility session and return control to system, you can choose one of the following methods:

- Type a DDDL signoff command (SIGNOFF, BYE, END, or LOGOFF) in the command area and press ENTER.
- Type TOP in the command area and press ENTER to return to the Master Selection screen. Press the CLEAR key.
- Press CLEAR as many times as needed to move through the menu facility structure, through the Master Selection screen, and back to the system.

Descriptions of IDD Menu Facility Screens

These general topics are presented below followed by descriptions of each entity type and associated screens:

- Entry and processing screens
- Screens common to all entity types

In this discussion, the structure of the IDD menu facility is illustrated by entity type. For each screen within an entity-type structure, the following information is also listed:

- Screen title
- Screen name
- The aspects of the entity-occurrence definition that can be implemented on the named screen

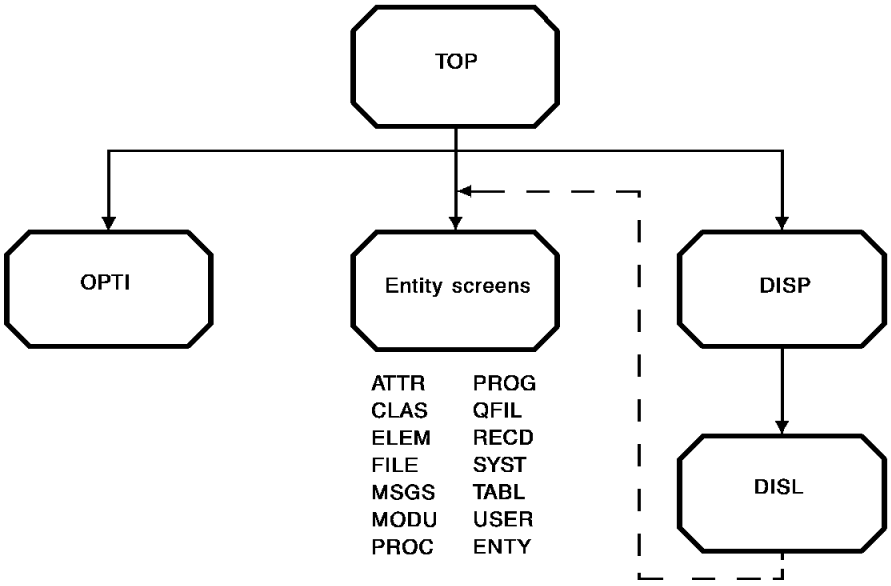
Note: Refer to previous chapters in this manual for a complete description of each entity type and its associated clauses.

Entry and Processing Screens

The Master Selection screen is the entry screen of the IDD menu facility. From this screen, you can:

- Select the Session Options screen and/or the Display All screen to define processing options
- Select an Entity screen to define an entity occurrence

The following figure illustrates the top of the menu facility's hierarchical structure.



The names of the Entity screens that can be selected from the Master Selection screen are listed. Each Entity screen and its associated subordinate screens are described separately in this section.

Note: Using the Display All screen, you can select an entity occurrence to be displayed. The IDD menu facility transfers control to the appropriate Entity screen to display the occurrence. If you subsequently press CLEAR from the Entity screen, control returns to the Master Selection screen, not the Display All List screen.

Entry and Processing Screens

The following table describes the screens associated with the menu facility entry and processing options.

Screen name	Screen title	Description
DISP	Display All	Defines the criteria to select entity occurrences for display

Screen name	Screen title	Description
DISL	Display All List	Lists entity occurrences specified on the Display All screen; allows selection of an entity occurrence for further display under the appropriate entity-type screens
OPTI	Session Options	Defines session processing options
TOP	Master Selection	Identifies the primary screen; furnishes signon information; provides access to subordinate screens

Screens Common to All Entity Types

Certain screens apply generally to all entity types within the IDD menu facility. These subordinate screens function at various levels within the menu facility structure. The figures accompanying the following entity screen description tables show where the common screens appear in the menu facility structure.

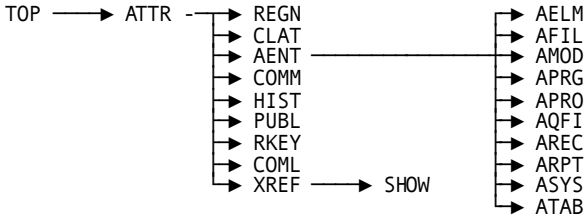
Common Screens

The following table describes screens that are common to all entity types. Each screen can function at various levels within the IDD menu facility.

Screen name	Screen title	Description
CLAT	Class/Attribute	Associates attributes and classes with an entity
COML	Comment Key List	Lists all comment keys defined for an entity; selects one for review
COMM	Comments	Associates text with a comment key for an entity
ERRS	Error Display	Lists DDDL syntax and messages related to current processing errors
HELP	Help	Displays a help tutorial
HIST	History	Shows the chronological entity history
PUBL	Public Access	Defines entity security for unregistered users
REGN	User Registration	Assigns user registration by entity occurrence
SHOW	Cross Reference	Displays information requested on the Cross Reference selection screen (XREF)

ATTRIBUTE Entity Screens

The following figure shows the entity screen and subordinate screens associated with ATTRIBUTE entity definitions. The arrows show the path through these screens.



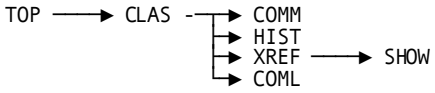
The following table describes the entity screen and subordinate screens associated with ATTRIBUTE entity definitions.

Screen name	Screen title	Description
AELM	Attribute Elements	Associates elements with the named attribute
AENT	Attribute/Entity	Requests display of all occurrences of an entity type that have the named attribute
AFIL	Attribute Files	Associates files with the named attribute
AMOD	Attribute Modules	Associates modules with the named attribute
APRG	Attribute Programs	Associates programs with the named attribute
APRO	Attribute Processes	Associates CA ADS processes with the named attribute
AQFI	Attribute Qfiles	Associates CA OLQ qfiles with the named attribute
AREC	Attribute Records	Associates records with the named attribute
ARPT	Attribute Reports	Associates reports with the named attribute
ASYS	Attribute Systems	Associates systems with the named attribute
ATAB	Attribute Tables	Associates tables with the named attribute
ATRN	Attribute Transactions	Associates transactions with the named attribute
ATTR	Attribute Entity	Identifies an attribute occurrence

Screen name	Screen title	Description
AUSR	Attribute Users	Associates users with the named attribute
RKEY	Relational Keys	Associates attributes with the named attribute through predefined relational keys
XREF	Attribute Cross Reference	Requests display of attributes, schemas, and/or subschemas that reference the named attribute in their definitions

CLASS Entity Screens

The following figure shows the entity screen and subordinate screens associated with CLASS entity definitions. The arrows show the path through these screens.



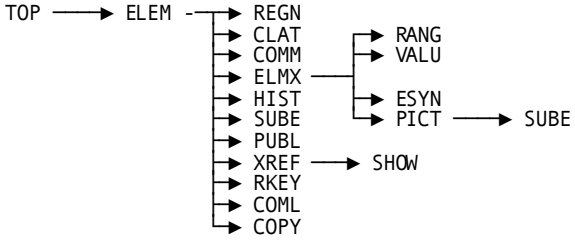
The following table describes the entity screen and subordinate screens associated with CLASS entity definitions.

Screen name	Screen title	Description
CLAS	Class Entity	Identifies a class occurrence
XREF	Class Cross Reference	Requests display of all attributes associated with the named class

ELEMENT Entity Screens

The following figure shows the entity screen and subordinate screens associated with ELEMENT entity definitions. The arrows show the path through these screens.

Note: The Subordinate Elements (SUBE) screen can be invoked from the Element Entity (ELEM) screen and the Element Picture (PICT) screen.

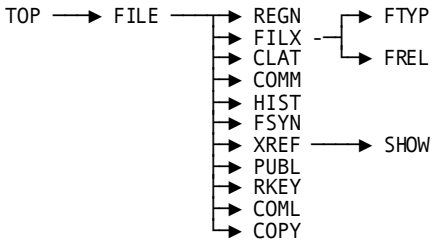


The following table describes the entity screen and subordinate screens associated with ELEMENT entity definitions.

Screen name	Screen title	Description
COPY	Element Copy	Copies all or specified options of an element definition into the definition of the named element
ELEM	Element Entity	Identifies an element occurrence
ELMX	Element Extension	Selects a picture format for definition and/or renames the requested element; accesses the Element Picture screen, the Element Values screen, the Element Synonyms screen, and the Element Ranges screen.
ESYN	Element Synonym	Defines element synonyms (alternative names for an element)
PICT	Element Picture	Associates a picture definition with the picture format keyword selected on the Element Extension screen; accesses subordinate elements associated with a group element (SUBE)
RANG	Ranges	Assigns valid ranges of values for an element
RKEY	Relational Keys	Associates elements with the named element through predefined relational keys
SUBE	Subordinate Elements	Associates subordinate elements with a group element
VALU	Values	Assigns an initial value to an element when it is copied into a program; if the element is a level-88 item, assigns multiple values
XREF	Element Cross Reference	Requests display of elements, records, reports, and/or transactions that reference the named element in their definitions

FILE Entity Screens

The following figure shows the entity screen and subordinate screens associated with FILE entity definitions. The arrows show the path through these screens.

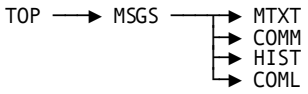


The following table describes the entity screen and subordinate screens associated with FILE entity definitions.

Screen name	Screen title	Description
COPY	File Copy	Copies all or specified options of a file definition into the definition of the named file
FILE	File Entity	Identifies a file occurrence
FILX	File Extension	Selects CA Culprit-related file options; renames the requested file; accesses the File Type screen and the Related Files screen
FREL	Related Files	Associates files with the named file
FSYN	File Synonyms	Identifies file synonyms (alternative names for a file)
FTYP	File Type	Assigns a file type, a VSAM file type, and/or a file device type
RKEY	Relational Keys	Associates files with the named file through predefined relational keys
XREF	File Cross Reference	Requests display of files, records, reports, transactions, and/or users that reference the named file in their definitions

MESSAGE Entity Screens

The following figure shows the entity screen and subordinate screens associated with MESSAGE entity definitions. The arrows show the path through these screens.

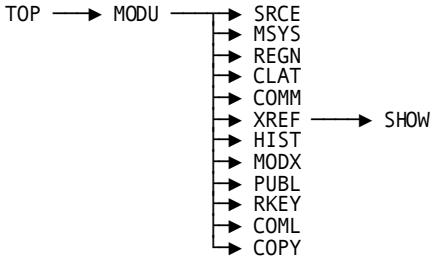


The following table describes the entity screen and subordinate screens associated with MESSAGE entity definitions.

Screen name	Screen title	Description
MSGS	Message Entity	Identifies a message occurrence
MTXT	Message Text Line	Associates text lines with the named message

MODULE Entity Screens

The following figure shows the entity screen and subordinate screens associated with MODULE entity definitions. The arrows show the path through these screens.



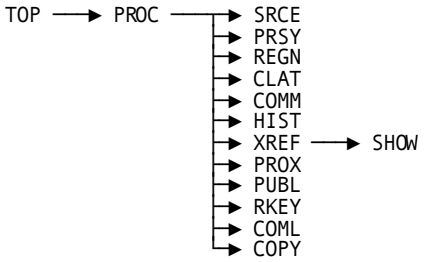
The following table describes the entity screen and subordinate screens associated with MODULE entity definitions.

Screen name	Screen title	Description
COPY	Module Copy	Copies all or specified options of a module definition into the definition of the named module
MODU	Module Entity	Identifies a module occurrence
MODX	Module Extension	Renames the requested module; establishes a new language
MSYS	Within Systems	Associates systems with the named module

Screen name	Screen title	Description
RKEY	Relational Keys	Associates modules with the named module through predefined relational keys
SRCE	Module Source	Associates source text lines with the named module
XREF	Module Cross Reference	Requests display of modules, users, and/or programs that reference the named module in their definitions

PROCESS Entity Screens

The following figure shows the entity screen and subordinate screens associated with PROCESS entity definitions. The arrows show the path through these screens.



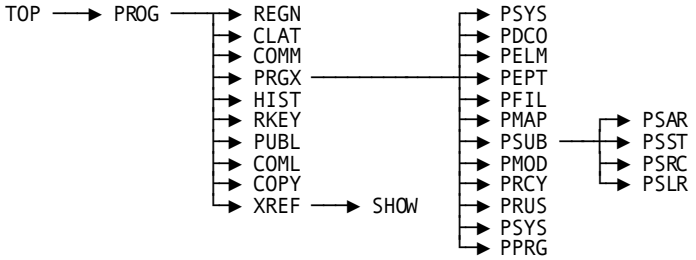
The following table describes the entity screen and subordinate screens associated with PROCESS entity definitions.

Screen name	Screen title	Description
COPY	Process Copy	Copies all or specified options of a process definition into the definition of the named process
PROC	Process Entity	Identifies an CA ADS process occurrence
PROX	Process Extension	Renames the requested process
PRSY	Within Systems	Associates systems with the named process
RKEY	Relational Keys	Associates processes with the named process through predefined relational keys for modules established as processes
SRCE	Process Source	Associates source text lines with the named process

Screen name	Screen title	Description
XREF	Module Cross Reference	Requests display of modules, users, and/or programs that reference the named process in their definitions

PROGRAM Entity Screens

The following figure shows the entity screen and subordinate screens associated with PROGRAM entity definitions. The arrows show the path through these screens.



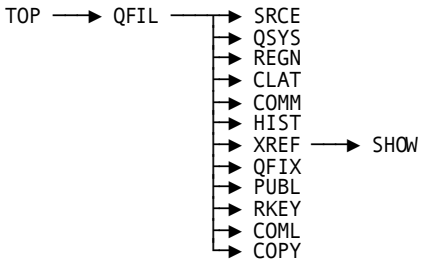
The following table describes the entity screen and subordinate screens associated with PROGRAM entity definitions.

Screen name	Screen title	Description
COPY	Program Copy	Copies all or specified options of a program definition into the definition of the named program
HIST	History	Shows a chronological account of a program's existence
PDCO	Program DC Options	Assigns CA IDMS/DC options to the named program
PELM	Program Elements	Describes elements used by the named program
PEPT	Program Entry Points	Associates entry points with the named program
PFIL	Program Files	Associates files with the named program
PMAP	Program Maps Used	Describes maps used by the named program
PMOD	Program Modules Used	Describes modules used by the named program

Screen name	Screen title	Description
PPRG	Programs Called	Describes programs called by the named program
PRCY	Program Records Copied	Describes records copied by the named program
PRGX	Program Extension	Defines the estimated number of lines; for an CA ADS Batch transaction, defines the starting sequence number in the named program; renames the requested program; accesses screens that relate programs to occurrences of subschemas and other entity types or that further define programs
PROG	Program Entity	Identifies a program occurrence
PRUS	Program Records Used	Describes records used by the named program
PSAR	Program Subschema Areas	Describes subschema areas accessed by the named program
PSLR	Program Logical Records	Describes subschema logical records used by the named program
PSRC	Program Subschema Records	Describes subschema records used by the named program
PSST	Program Subschema Sets	Describes subschema sets used by the named program
PSUB	Program Subschema	Describes the subschema used by the named program; accesses the Program Subschema Areas screen, the Program Subschema Records screen, the Program Subschema Sets screen, and the Program Logical Records screen
PSYS	Within Systems	Associates systems with the named program
RKEY	Relational Keys	Associates programs with the named program through predefined relational keys
XREF	Program Cross Reference	Requests display of programs and tasks that reference the named program in their definitions

QFILE Entity Screens

The following figure shows the entity screen and subordinate screens associated with QFILE entity definitions. The arrows show the path through these screens.



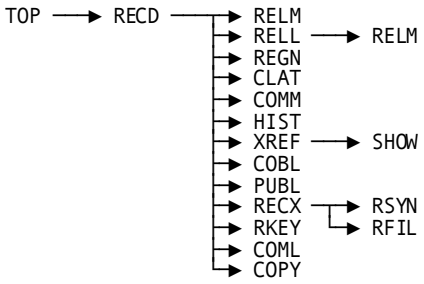
The following table describes the entity screen and subordinate screens associated with QFILE entity definitions.

Screen name	Screen title	Description
COPY	Qfile Copy	Copies all or specified options of a qfile definition into the definition of the named qfile
QFIL	Qfile Entity	Identifies an CA OLQ qfile occurrence
QFIX	Qfile Extension	Renames the requested qfile
QSYS	Within Systems	Associates systems with the named qfile
RKEY	Relational Keys	Associates qfiles with the named qfile through predefined relational keys for modules established as qfiles
SRCE	Qfile Source	Associates source text lines with the named qfile
XREF	Module Cross Reference	Requests display of modules, users, and/or programs that reference the named qfile in their definitions

RECORD Entity Screens

The following figure shows the entity screen and subordinate screens associated with RECORD entity definitions. The arrows show the path through these screens.

Note: The Record Element List(RELL) screen can be invoked from the Record Entity (RECD) screen and the Record Element (RELM) screen.



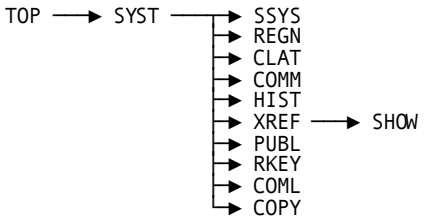
The following table describes the entity screen and subordinate screens associated with RECORD entity definitions.

Screen name	Screen title	Description
COBL	COBOL Elements	Displays the COBOL format of the record elements associated with the named record
COPY	Record Copy	Copies all or specified options of a record definition into the definition of the named record
RECD	Record Entity	Identifies a record occurrence
RECX	Record Extension	Defines storage medium and estimated number of occurrences for the named record; renames the requested record; accesses the Record Synonym screen and the Within File screen
RELL	Record Element List	Lists record elements associated with the named record; selects record element occurrences for further display by means of the Record Element screen
RELM	Record Element	Displays or associates record elements with the named record
RFIL	Within File	Associates files with the named record

Screen name	Screen title	Description
RKEY	Relational Keys	Associates records with the named record through predefined relational keys
RSYN	Record Synonym	Identifies record synonyms (alternative names for a record)
XREF	Record Cross Reference	Requests display of records, programs, maps, schemas, and/or subschemas that reference the named record in their definition

SYSTEM Entity Screens

The following figure shows the entity screen and subordinate screens associated with SYSTEM entity definitions. The arrows show the path through these screens.

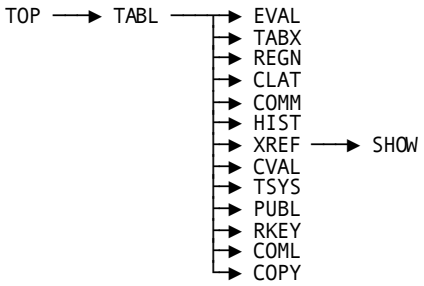


The following table describes the entity screen and subordinate screens associated with SYSTEM entity definitions.

Screen name	Screen title	Description
COPY	System Copy	Copies all or specified options of a system definition into the definition of the named system
RKEY	Relational Keys	Associates systems with the named system through predefined relational keys
SSYS	Within Systems	Associates systems with the named system
SYST	System Entity	Identifies a system occurrence
XREF	System Cross Reference	Requests display of entities that reference the named system in their definitions

TABLE Entity Screens

The following figure shows the entity screen and subordinate screens associated with TABLE entity definitions. The arrows show the path through these screens.

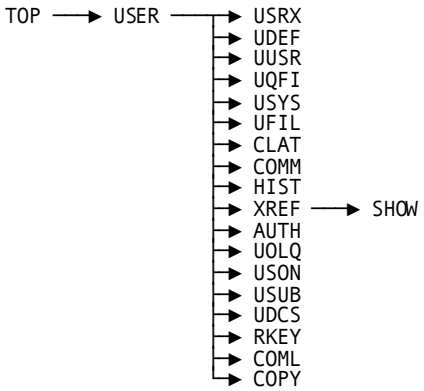


The following table describes the entity screen and subordinate screens associated with TABLE entity definitions.

Screen name	Screen title	Description
COPY	Table Copy	Copies all or specified options of a table definition into the definition of the named table
CVAL	Code Table Values	Associates encode and decode values with the named code table
EVAL	Table Edit Values	Associates values and value ranges with the named edit table
TABL	Table Entity	Identifies a table occurrence
TABX	Table Extension	Renames the requested table
TSYS	Within Systems	Associates systems with the named table
RKEY	Relational Keys	Associates tables with the named table through predefined relational keys for modules established as tables
XREF	Module Cross Reference	Requests display of modules, users, and/or programs that reference the named table in their definition

USER Entity Screens

The following figure shows the entity screen and subordinate screens associated with USER entity definitions. The arrows show the path through these screens.

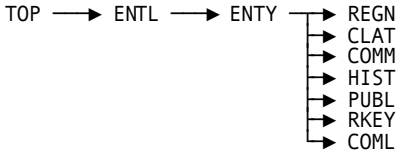


The following table describes the entity screen and subordinate screens associated with USER entity definitions.

Screen name	Screen title	Description
AUTH	User Authority	Assigns product and entity-type authority to the named user
COPY	User Copy	Copies all or specified options of a user definition into the definition of the named user
RKEY	Relational Keys	Associates users with the named user through predefined relational keys
UDCS	Access to DC/UCF Systems	Defines the user's access to particular systems
UDEF	OLQ/CULPRIT Definition	Assigns CA OLQ and CA Culprit access options to the named user
UFIL	Access to Files	Defines the user's access to particular CA Culprit files
UOLQ	OLQ Default Options	Assigns default processing options for CA OLQ to the named user
UQFI	Access to Qfiles	Defines the user's access to particular qfiles
USER	User Entity	Identifies a user occurrence

Screen name	Screen title	Description
USON	Signon Profiles	Associates signon profiles with the named user (modules that can be executed when the user signs on to a system or an application)
USRX	User Extension	Assigns access to ASF, IDB, and/or IDD to the named user; indicates a default public access specification for entities added by the named user under ASF; renames the requested user
USUB	Access to Subschemas	Defines the user's access to particular subschemas
USYS	Of Systems	Associates systems with the named user
UUSR	Within Users	Associates users with the named user
XREF	User Cross Reference	Requests display of entities that reference the named user in their definitions

The following figure shows the entity screen and subordinate screens associated with USER-DEFINED entity definitions. The arrows show the path through the screens.



The following table describes the entity screen and subordinate screens associated with USER-DEFINED entity definitions.

Screen name	Screen title	Description
ENTL	User-Defined Entity List	Lists all user-defined entity types in the dictionary; allows selection of a user-defined entity occurrence for further display
ENTY	User Defined Entity	Identifies a user-defined entity occurrence; renames a user-defined entity occurrence

Sample Session

A sample session that illustrates the use of the menu facility is presented below. During this session, you request the menu facility to display a list of ELEMENT entity occurrences, then chooses one entity occurrence for display.

Each step of the session is described and illustrated by the current screen. Default values for response fields (indicated by underscores) are listed when applicable.

Beginning a Menu Facility Session

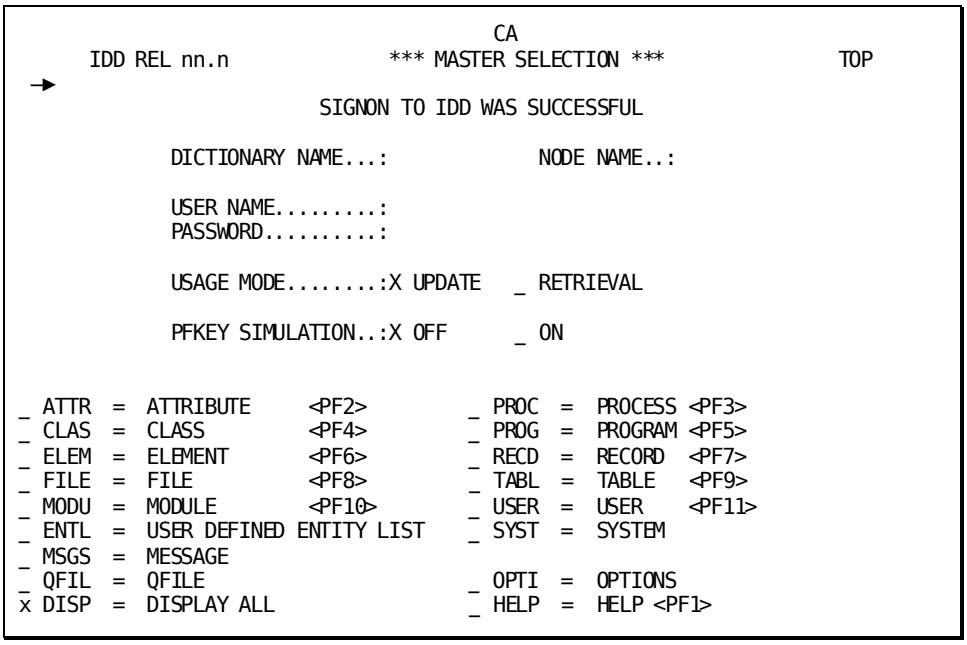
To begin an IDD menu facility session:

1. Sign on to the system.
2. When signon is accepted, type the task code **IDDM** to invoke the IDD menu facility and display the Master Selection screen.
3. Press ENTER to sign on to the IDD menu facility, using the system signon information. The menu facility returns a message that indicates a successful signon.

```
                                        CA
    IDD REL nn.n              *** MASTER SELECTION ***              TOP
➔
                         SIGNON TO IDD WAS SUCCESSFUL

             DICTIONARY NAME...:              NODE NAME..:

             USER NAME.........:
             PASSWORD..........:

             USAGE MODE........:X UPDATE   _ RETRIEVAL

             PFKEY SIMULATION..:X OFF      _ ON


_ ATTR  =  ATTRIBUTE     <PF2>       _ PROC  =  PROCESS    <PF3>
_ CLAS  =  CLASS         <PF4>       _ PROG  =  PROGRAM    <PF5>
_ ELEM  =  ELEMENT       <PF6>       _ RECD  =  RECORD     <PF7>
_ FILE  =  FILE          <PF8>       _ TABL  =  TABLE      <PF9>
_ MODU  =  MODULE        <PF10>      _ USER  =  USER       <PF11>
_ ENTL  =  USER DEFINED ENTITY LIST  _ SYST  =  SYSTEM
_ MSGS  =  MESSAGE
_ QFIL  =  QFILE                     _ OPTI  =  OPTIONS
_ DISP  =  DISPLAY ALL               _ HELP  =  HELP       <PF1>
```

Displaying a List of Entity Occurrences

To display a list of entity occurrences, you type a character next to DISPLAY ALL in the screen selection area and press ENTER:



The menu facility displays the *Display All* screen.

Specifying Selection Criteria

The WHERE response area specifies criteria to be used by the menu facility in selecting the occurrences to be displayed. To display the first 20 ELEMENT entity occurrences that contain the string W-EMP in the element name, you:

1. Request a comparison, using the syntax option NAME
2. Override the default comparison action EQ by typing a character at the underscore that immediately precedes the CONTAINS field
3. Request that the ELEMENT NAME be searched for the string W-EMP by typing the string in the 'VALUE'/FIELD field

More information: For more information about the WHERE clause, see WHERE Clause (Conditional Expressions) (see page 118).

```
    IDD REL nn.n        *** DISPLAY ALL ***                        DISP
➔


 DISPLAY PROCESSING ORDER..:X FIRST  _ NEXT   _ LAST   _ PRIOR  _ ALL

 NUMBER OF OCCURRENCES.....:20

 OF ENTITY TYPE............:_ ATTRIBUTES       _ PROCESSES        _ TABLES
                             _ CLASSES          _ PROGRAMS         _ USERS
                             X ELEMENTS         _ QFILES           _ MESSAGES
                             _ FILES            _ RECORDS
                             _ MODULES          _ SYSTEMS
                             _ MODULES ONLY (WITHOUT PROCESSES, QFILES, TABLES)
                             _ USER DEFINED ENTITY...:

   WHERE 'VALUE'/FIELD....:name

       COMPARISON ACTION..:X EQ   _ NE   _ GT   _ GE   _ LT   _ LE
                           X CONTAINS    _ MATCHES

          'VALUE'/FIELD....:'w-emp'
       OR 'VALUE'/FIELD....:
       OR 'VALUE'/FIELD....:
```

The menu facility displays the pageable *Display All List* screen.

Display All List Screen

The SELECTED ON field and underlying headers display the selection criteria; that is, element names that contain W-EMP. The entity occurrences that fulfill the requirements are listed in columnar format by name and version number.

You can scan the list of element entity occurrences on the Display All List screen and choose to view additional information about element W-EMP-ADDRESS by typing a character next to the list item and pressing ENTER.

```
    IDD REL nn.n          *** DISPLAY ALL LIST ***                 DISL
→                                          PAGE 1 OF 2

SELECTED ON:  NAME CONTAINS 'W-EMP'

ELEMENT NAME                       VER

_ W-EMPOSITION-VERB                  1
_ W-EMPLOYEE-VERB                    1
_ W-EMP-BIRTH-DAY                    1
_ W-EMP-TERM-DAY                     1
_ W-EMP-START-DAY                    1
_ W-EMP-SS-NUMBER                    1
_ W-EMP-STATUS                       1
_ W-EMP-HOME-PHONE                   1
_ W-EMP-ZIP-LAST-4                   1
_ W-EMP-STATE                        1
_ W-EMP-CITY                         1
_ W-EMP-STREET                       1
X W-EMP-ADDRESS                      1
_ W-EMP-ZIP-FIRST-5                  1
_ W-EMP-ZIP                          1
_ W-EMP-SEX                          1
```

The IDD menu facility displays the element occurrence W-EMP-ADDRESS on the *Element Entity* screen.

Element Entity Screen

The message line contains an informative message confirming your request. The specification area identifies the entity occurrence by NAME, VERSION NUMBER, and USAGE.

```
    IDD REL nn.n            *** ELEMENT ENTITY ***                  ELEM
→
                  ELEMENT 'W-EMP-ADDRESS' VERSION 1 DISPLAYED

X DISPLAY      ELEMENT NAME....:W-EMP-ADDRESS
_ MODIFY
_ ADD          VERSION NUMBER..:1      _ HIGHEST   _ NEXT HIGHEST
_ DELETE                               _ LOWEST    _ NEXT LOWEST

 DESCRIPTION:

 PICTURE....:

 USAGE......:X DISPLAY                 _ CONDITION NAME (LEVEL 88)
             _ COMP/COMP-4 (BINARY)    _ COMP-3 (PACKED DECIMAL)
             _ COMP-1 (SHORT FLOATING) _ COMP-2 (LONG FLOATING)
             _ BIT                     _ POINTER

_ ELMX  =  ELEMENT EXTENSION <PF9>     _ SUBE  =  SUBORD ELEMENTS  <PF11>
_ REGN  =  USER REGISTRATION <PF2>     _ PUBL  =  PUBLIC ACCESS    <PF3>
_ CLAT  =  CLASS/ATTRIBUTES  <PF4>     _ RKEY  =  RELATIONAL KEYS  <PF5>
_ COMM  =  COMMENTS          <PF6>     _ COML  =  COMMENT KEY LIST <PF7>
_ HIST  =  HISTORY           <PF8>     _ COPY  =  SAME AS/COPY FROM
_ XREF  =  CROSS REFERENCE   <PF10>    _ HELP  =  HELP             <PF1>
```

Selecting Fields to View Additional Information

In order to view other information about W-EMP-ADDRESS, refer to the screen selection area at the bottom of the Element Entity screen. The screen selection area lists other screens that contain information about the ELEMENT entity occurrence.

To view information about subordinate elements, you type a character at the underscore immediately preceding the SUBE field and press ENTER.

```
    IDD REL nn.n              *** ELEMENT ENTITY ***                  ELEM
 →
                    ELEMENT 'W-EMP-ADDRESS' VERSION 1 DISPLAYED

X DISPLAY      ELEMENT NAME....:W-EMP-ADDRESS
_ MODIFY
_ ADD          VERSION NUMBER..:1       _ HIGHEST   _ NEXT HIGHEST
_ DELETE                                _ LOWEST    _ NEXT LOWEST

 DESCRIPTION:

 PICTURE....:

 USAGE......:X DISPLAY                  _ CONDITION NAME (LEVEL 88)
             _ COMP/COMP-4 (BINARY)     _ COMP-3 (PACKED DECIMAL)
             _ COMP-1 (SHORT FLOATING)  _ COMP-2 (LONG FLOATING)
             _ BIT                      _ POINTER

_ ELMX  =  ELEMENT EXTENSION <PF9>     X SUBE  =  SUBORD ELEMENTS  <PF11>
_ REGN  =  USER REGISTRATION <PF2>     _ PUBL  =  PUBLIC ACCESS    <PF3>
_ CLAT  =  CLASS/ATTRIBUTES  <PF4>     _ RKEY  =  RELATIONAL KEYS  <PF5>
_ COMM  =  COMMENTS          <PF6>     _ COML  =  COMMENT KEY LIST <PF7>
_ HIST  =  HISTORY           <PF8>     _ COPY  =  SAME AS/COPY FROM
_ XREF  =  CROSS REFERENCE   <PF10>    _ HELP  =  HELP             <PF1>
```

The IDD menu facility displays the *Subordinate Elements* screen, which contains a list of elements that are subordinate to group element W-EMP-ADDRESS.

Subordinate Elements Screen

A message displayed on the message line identifies the group element entity occurrence.

```
    IDD REL nn.n          *** SUBORDINATE ELEMENTS ***              SUBE
→                                          PAGE 1 LINE 1          1/4
                      ELEMENT 'W-EMP-ADDRESS' VERSION 1

 ELEMENT NAME....:W-EMP-STREET
 VERSION NUMBER..:1     _ HIGHEST  _ LOWEST
        OCCURS..:       TIMES      _ REDEFINES PREVIOUS SUBORDINATE ELEMENT

 ELEMENT NAME....:W-EMP-CITY
 VERSION NUMBER..:1     _ HIGHEST  _ LOWEST
        OCCURS..:       TIMES      _ REDEFINES PREVIOUS SUBORDINATE ELEMENT

 ELEMENT NAME....:W-EMP-STATE
 VERSION NUMBER..:1     _ HIGHEST  _ LOWEST
        OCCURS..:       TIMES      _ REDEFINES PREVIOUS SUBORDINATE ELEMENT

 ELEMENT NAME....:W-EMP-ZIP
 VERSION NUMBER..:1     _ HIGHEST  _ LOWEST
        OCCURS..:       TIMES      _ REDEFINES PREVIOUS SUBORDINATE ELEMENT

 ELEMENT NAME....:
 VERSION NUMBER..:      _ HIGHEST  _ LOWEST
        OCCURS..:       TIMES      _ REDEFINES PREVIOUS SUBORDINATE ELEMENT
```

After choosing to review cross-reference information about element W-EMP-ADDRESS you request the Element Cross Reference screen by typing the screen name XREF in the command area of the Subordinate Elements screen and pressing ENTER.

```
    IDD REL nn.n          *** SUBORDINATE ELEMENTS ***              SUBE
→XREF                                      PAGE 1 LINE 1          1/4
                      ELEMENT 'W-EMP-ADDRESS' VERSION 1

ELEMENT NAME....:W-EMP-STREET
VERSION NUMBER..:1     _ HIGHEST  _ LOWEST
        OCCURS..:       TIMES      _ REDEFINES PREVIOUS SUBORDINATE ELEMENT

ELEMENT NAME....:W-EMP-CITY
VERSION NUMBER..:1     _ HIGHEST  _ LOWEST
        OCCURS..:       TIMES      _ REDEFINES PREVIOUS SUBORDINATE ELEMENT

ELEMENT NAME....:W-EMP-STATE
VERSION NUMBER..:1     _ HIGHEST  _ LOWEST
        OCCURS..:       TIMES      _ REDEFINES PREVIOUS SUBORDINATE ELEMENT

ELEMENT NAME....:W-EMP-ZIP
VERSION NUMBER..:1     _ HIGHEST  _ LOWEST
        OCCURS..:       TIMES      _ REDEFINES PREVIOUS SUBORDINATE ELEMENT

ELEMENT NAME....:
VERSION NUMBER..:      _ HIGHEST  _ LOWEST
        OCCURS..:       TIMES      _ REDEFINES PREVIOUS SUBORDINATE ELEMENT
```

The menu facility displays the *Element Cross Reference* selection screen.

Element Cross Reference Screen

To review all related information, you type a character in the CROSS REFERENCE TO ALL CATEGORIES LISTED BELOW field and press ENTER.

```
    IDD REL nn.n      *** 'ELEMENT' CROSS REFERENCE ***              XREF
→
                      ELEMENT 'W-EMP-ADDRESS' VERSION 1


           X CROSS REFERENCE TO ALL CATEGORIES LISTED BELOW

           _ SAME AS OTHER ELEMENTS

           _ RELATED TO OTHER ELEMENTS

           _ WITHIN RECORDS    _ WITHIN REPORTS   _ WITHIN TRANSACTIONS




      NOTE - SELECT ONE OR MORE OF THE ABOVE CATEGORIES TO DISPLAY THE
             CROSS REFERENCE INFORMATION ASSOCIATED WITH THIS ELEMENT.
```

The entity occurrence is highlighted on the message line of the *Element Cross Reference* screen. Related cross-reference information is displayed in the specification area. W-EMP-ADDRESS is an element in the record JMH-WORD-REC-01, version number 1.

```
    IDD REL nn.n      *** 'ELEMENT' CROSS REFERENCE ***              SHOW
→                                          PAGE 1 OF 1
                      ELEMENT 'W-EMP-ADDRESS' VERSION 1

        WITHIN RECORD JMH-WORK-REC-01 VERSION 1
            WITHIN GROUP W-EMPLOYEE VERSION IS 1
```

Ending the Menu Facility Session

To end the IDD menu facility session and return control to the system, you type the signoff command BYE in the command area of the current screen and press ENTER.

```
    IDD REL nn.n      *** 'ELEMENT' CROSS REFERENCE ***              SHOW
→BYE                                       PAGE 1 OF 1
                      ELEMENT 'W-EMP-ADDRESS' VERSION 1

        WITHIN RECORD JMH-WORK-REC-01 VERSION 1
            WITHIN GROUP W-EMPLOYEE VERSION IS 1
```

Appendix A: DDDL Compiler Batch Execution JCL

This appendix shows the JCL/commands you use to execute the batch DDDL compiler (IDMSDDDL) under z/OS, z/VSE, and z/VM.

This section contains the following topics:

IDMSDDDL Under z/OS (see page 443)
IDMSDDDL Under z/VSE (see page 445)
IDMSDDDL Under z/VM (see page 453)

IDMSDDDL Under z/OS

Executing Under the Central Version

z/OS JCL for running IDMSDDDL under the CA IDMS/DB central version follows:

IDMSDDDL (z/OS)

```
//stepname EXEC PGM=IDMSDDDL,REGION=1024K
//STEPLIB  DD   DSN=idms.dba.loadlib,DISP=SHR
//         DD   DSN=idms.custom.loadlib,DISP=SHR
//         DD   DSN=idms.cagjload,DISP=SHR
//sysctl   DD   DSN=idms.sysctl,DISP=SHR
//dcmsg    DD   DSN=sysmsg.ddldcmsg,DISP=SHR
//SYSLST   DD   SYSOUT=A
//SYSIDMS  DD   *
DMCL=dmcl-name
DICTNAME=dictionary-name
Other optional SYSIDMS parameters
/*
//SYSIPT   DD   *
DDDL source statements
/*
```

More information: For more information on optional SYSIDMS parameters, refer to the *CA IDMS Common Facilities Guide* document.

idms.dba.loadlib	Dataset name of the load library containing the DMCL and database name table load modules
idms.custom.loadlib	Dataset name of the load library containing the customized CA IDMS executable module

idms.cagjload	Dataset name of the load library containing the CA IDMS executable modules that do not require customization
sysctl	DDname for the SYSCTL file; *sysctl* is SYSCTL unless specified otherwise in IDMSOPTI
idms.sysctl	Dataset name of the SYSCTL file
dcmsg	DDname name of the system message (DDLDCMSG) area
sysmsg.ddldcmsg	Dataset name of the system message (DDLDCMSG) area
dmcl-name	Name of the DMCL to be accessed
dictionary-name	Name of the dictionary to be accessed

SYSPCH Statement

If you are going to be using any PUNCH statements, include the SYSPCH statement in JCL. For example:

```
//SYSPCH    DD    DSN=dataset-name,DISP=(NEW,KEEP,DELETE),
                  DCB=(RECFM=FB,BLKSIZE=9040,LRECL=80),
                       SPACE=space-specification,
                       UNIT=unit,VOL=SER=nnnnnn
```

Executing in Local Mode

To execute the DDDL compiler in local mode, remove the SYSCTL DD statement and replace it with the following statements:

```
//dictdb  DD  DSN=idms.appldict.ddldml,DISP=SHR
//dloddb  DD  DSN=idms.appldict.ddldclod,DISP=SHR
//sysjrnl DD  DSN=idms.tapejrnl,DISP=SHR
```

Note: These statements are needed only if the DDDL compiler run accesses the LOAD MODULE entity type.

dictdb	DDname of the application dictionary definition (DDLDML) area
idms.appldict.ddldml	Dataset name of the application dictionary definition (DDLDML) area
dloddb	DDname of the application dictionary load (DDLDCLOD) area
idms.appldict.ddldclod	Dataset name of the application dictionary load (DDLDCLOD) area
sysjrnl	DDname of tape journal file; the name must be appropriate to the DMCL module being used

idms.tapejrnl	Dataset name of tape journal file

IDMSDDDL Under z/VSE

Executing Under the Central Version

The z/VSE JCL used to run IDMSDDDL under the central version follows:

IDMSDDDL (z/VSE)

```
// EXEC PROC=IDMSLBLS
// UPSI b                 If specified in IDMSOPTI module
// DLBL     IJSYSPH,'temp.dddl',0
// EXTENT   SYSPCH,nnnnnn,,,ssss,llll
   ASSGN    SYSPCH,X'ddd'
// EXEC     IDMSDDDL
Optional SYSIDMS parameters
/*
DDDL source statements
/*
```

IDMSLBLS	Name of the procedure provided at installation that contains the file definitions for CA IDMS dictionaries and databases. For a complete listing of IDMSLBLS, see "IDMSLBLS procedure", later in this section. IDMSLBLS references the SYSIDMS input file, a file you can use to specify runtime parameters, such as DMCL or dictionary name. For information on SYSIDMS parameters, refer to *CA IDMS Common Facilities Guide*.
b	Appropriate 1- to 8-character UPSI bit string, as specified in the IDMSOPTI module
nnnnnn	Serial number of the disk volume
temp.dddl	File ID of the output file
ssss	Starting track (CKD) or block (FBA) of disk extent
llll	Number of tracks (CKD) or blocks (FBA) of disk extent
ddd	Disk device assignment for punched output

Executing in Local Mode

To execute IDMSDDDL in local mode, remove the UPSI specification and replace it with the following statements:

```
// EXTENT   sys017,nnnnnn
// ASSGN    sys017,DISK,VOL=nnnnnn,SHR
// TLBL     sysjrnl,'idms.tapejrnl',,nnnnnn,,f
// ASSGN    sys009,TAPE,VOL=nnnnnn
```

sys017	Logical unit assignment for dictionary load area
sys009	Filename of the tape journal file; the name must be appropriate to the DMCL module being used
idms.tapejrnl	File ID of the tape journal file
f	File number of the tape journal file
nnnnnn	Serial number of the tape volume

IDMSLBLS Procedure

IDMSLBLS is a procedure that contains file definitions for the dictionaries, sample databases, disk journal files, and SYSIDMS file provided during installation.

You can tailor the following IDMSLBLS procedure (provided at installation) to reflect the filenames and definitions in use at your site. Reference IDMSLBLS as shown in the previous z/VSE JCL job stream.

```
* ———————— LIBDEFS ————————
// LIBDEF  *,SEARCH=idmslib.sublib
// LIBDEF  *,CATALOG=user.sublib
/*  ———————————————————— LABELS ————————————————————
// DLBL    idmslib,'idms.library',1999/365                              00061000
// EXTENT  ,nnnnnn,,,ssss,1500                                          00062000
// DLBL    dccat,'idms.system.dccat',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,31
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dccatl,'idms.system.dccatlod',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,6
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dccatx,'idms.system.dccatx',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,11
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dcdml,'idms.system.ddldml',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,101
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dclod,'idms.system.ddldclod',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,21
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dclog,'idms.system.ddldclog',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,401
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dcrun,'idms.system.ddldcrun',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,68
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dcscr,'idms.system.ddldcscr',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,135
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dcmsg,'idms.sysmsg.ddldcmsg',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,201
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dclscr,'idms.sysloc.ddlocscr',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,6
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dirldb,'idms.sysdirl.ddldml',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,201
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dirllod,'idms.sysdirl.ddldclod',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,2
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    empdemo,'idms.empdemo1',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,11
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    insdemo,'idms.insdemo1',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,6
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    orgdemo,'idms.orgdemo1',1999/365,DA
```

```
// EXTENT  SYSnnn,nnnnnn,,,ssss,6
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    empldem,'idms.sqldemo.empldemo',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,11
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    infodem,'idms.sqldemo.infodemo',1999/365,DA

// EXTENT  SYSnnn,nnnnnn,,,ssss,6
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    projdem,'idms.projseg.projdemo',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,6
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    indxdem,'idms.sqldemo.indxdemo',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,6
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    sysctl,'idms.sysctl',1999/365,SD
// EXTENT  SYSnnn,nnnnnn,,,ssss,2
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    secdd,'idms.sysuser.ddlsec',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,26
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dictdb,'idms.appldict.ddldml',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,51
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dloddb,'idms.appldict.ddldclod',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,51
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    sqldd,'idms.syssql.ddlcat',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,101
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    sqllod,'idms.syssql.ddlcatl',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,51
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    sqlxdd,'idms.syssql.ddlcatx',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,26
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    asfdml,'idms.asfdict.ddldml',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,201
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    asflod,'idms.asfdict.asflod',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,401
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    asfdata,'idms.asfdict.asfdata',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,201
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    ASFDEFN,'idms.asfdict.asfdefn',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,101
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    j1jrnl,'idms.j1jrnl',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,54
```

```
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    j2jrnl,'idms.j2jrnl',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,54
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    j3jrnl,'idms.j3jrnl',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,54
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    SYSIDMS,'#SYSIPT',0,SD
/+
/*
```

idmslib.sublib	Name of the sublibrary within the library containing CA IDMS modules
user.sublib	Name of the sublibrary within the library containing user modules
idmslib	Name of the file containing CA IDMS modules
idms.library	ID associated with the file containing CA IDMS modules
SYSnnn	Logical unit of the volume for which the extent is effective
nnnnnn	Volume serial identifier of appropriate disk volume
ssss	Starting track (CKD) or block (FBA) of disk extent
dccat	Filename of the system dictionary catalog (DDLCAT) area
idms.system.dccat	ID of the system dictionary catalog (DDLCAT) area
dccatl	Filename of the system dictionary catalog load (DDLCATLOD) area
idms.system.dccatlod	ID of the system dictionary catalog load (DDLCATLOD) area
dccatx	Name of the system dictionary catalog index (DDLCATX) area
idms.system.dccatx	ID of the system dictionary catalog index (DDLCATX) area
dcdml	Name of the system dictionary definition (DDLDML) area
idms.system.ddldml	ID of the system dictionary definition (DDLDML) area
dclod	Name of the system dictionary definition load (DDLDCLOD) area
idms.system.ddldclod	ID of the system dictionary definition load (DDLDCLOD) area

dclog	Name of the system log area (DDLDCLOG) area
idms.system.ddldclog	ID of the system log (DDLDCLOG) area
dcrun	Name of the system queue (DDLDCRUN) area
idms.system.ddldcrun	ID of the system queue (DDLDCRUN) area
dcscr	Name of the system scratch (DDLDCSCR) area
idms.system.ddldcscr	ID of the system scratch (DDLDCSCR) area
dcmsg	Name of the system message (DDLDCMSG) area
idms.sysmsg.ddldcmsg	ID of the system message (DDLDCMSG) area
dclscr	Name of the local mode system scratch (DDLOCSCR) area
idms.sysloc.ddlocscr	ID of the local mode system scratch (DDLOCSCR) area
dirldb	Name of the IDMSDIRL definition (DDLDML) area
idms.sysdirl.ddldml	ID of the IDMSDIRL definition (DDLDML) area
dirllod	Name of the IDMSDIRL definition load (DDLDCLOD) area
idms.sysdirl.dirllod	ID of the IDMSDIRL definition load (DDLDCLOD) area
empdemo	Name of the EMPDEMO area
idms.empdemo1	ID of the EMPDEMO area
insdemo	Name of the INSDEMO area
idms.insdemo1	ID of the INSDEMO area
orgdemo	Name of the ORGDEMO area
idms.orgdemo1	ID of the ORDDEMO area
empldem	Name of the EMPLDEMO area
idms.sqldemo.empldemo	ID of the EMPLDEMO area
infodem	Name of the INFODEMO area
idms.sqldemo.infodemo	ID of the INFODEMO area
projdem	Name of the PROJDEMO area
idms.projseg.projdemo	ID of the PROJDEMO area
indxdem	Name of the INDXDEMO area
idms.sqldemo.indxdemo	ID of the INDXDEMO area
sysctl	Name of the SYSCTL file
idms.sysctl	ID of the SYSCTL file

secdd	Name of the system user catalog (DDLSEC) area
idms.sysuser.ddlsec	ID of the system user catalog (DDLSEC) area
dictdb	Name of the application dictionary definition area
idms.appldict.ddldml	ID of the application dictionary definition (DDLDML) area
dloddb	Name of the application dictionary definition load area
idms.appldict.ddldclod	ID of the application dictionary definition load (DDLDCLOD) area
sqldd	Name of the SQL catalog (DDLCAT) area
idms.syssql.ddlcat	ID of the SQL catalog (DDLCAT) area
sqllod	Name of the SQL catalog load (DDLCATL) area
idms.syssql.ddlcatl	ID of SQL catalog load (DDLCATL) area
sqlxdd	Name of the SQL catalog index (DDLCATX) area
idms.syssql.ddlcatx	ID of the SQL catalog index (DDLCATX) area
asfdml	Name of the asf dictionary definition (DDLDML) area
idms.asfdict.ddldml	ID of the asf dictionary definition (DDLDML) area
asflod	Name of the asf dictionary definition load (ASFLOD) area
idms.asfdict.asflod	ID of the asf dictionary definition load (ASFLOD) area
asfdata	Name of the asf data (ASFDATA) area
idms.asfdict.asfdata	ID of the asf data area (ASFDATA) area
ASFDEFN	Name of the asf data definition (ASFDEFN) area
idms.asfdict.asfdefn	ID of the asf data definition area (ASFDEFN) area
j1jrnl	Name of the first disk journal file
idms.j1jrnl	ID of the first disk journal file
j2jrnl	Name of the second disk journal file
idms.j2jrnl	ID of the second disk journal file
j3jrnl	Name of the third disk journal file
idms.j3jrnl	ID of the third disk journal file
SYSIDMS	Name of the SYSIDMS parameter file

IDMSDDDL Under z/VM

Executing Under the Central Version

The z/VM commands you use to run IDMSDDDL under the central version follow:

IDMSDDDL (z/VM)

```
FILEDEF SYSIPT DISK sysipt data a (RECFM F LRECL ppp BLKSIZE nnn
FILEDEF SYSIDMS DISK sysidms parms a (RECFM F LRECL ppp BLKSIZE nnn
EXEC IDMSFD
OSRUN IDMSDDDL
```

sysipt data a	Filename, type, and mode of the file containing DDDL statements
sysidms parms a	Filename, type, and mode of the file containing SYSIDMS parameters (parameters you use to specify your runtime environment)
ppp	Record length of the file
nnn	Block size of the file
IDMSFD	Exec which defines all FILEDEFs, TXTLIBs, and LOADLIBs required by the system
IDMSDDDL	Program to be executed from the z/VM LOADLIB

Executing in Local Mode

To specify that IDMSDDDL is executing in local mode, do one of the following:

- Link IDMSDDDL with an IDMSOPTI program that specifies local execution mode
- Specify *LOCAL* as the first input parameter in *sysipt data a*, the file referenced in the FILEDEF SYSIPT statement.
- Modify the OSRUN statement, as follows:

  ```
  OSRUN IDMSDDDL PARM='*LOCAL*'
  ```

  Note: This option is valid only if you issue the OSRUN command from a System Product interpreter or an EXEC2 file.

Creating the SYSIPT File

To create the SYSIPT file, enter these z/VM commands:

```
XEDIT sysipt data a (NOPROF
INPUT
 .
 .
 .
DDDL source statements
 .
 .
 .
FILE
```

Editing the SYSIDMS File

To edit the SYSIDMS file, enter these z/VM commands:

```
XEDIT sysidms parms a (NOPROF
INPUT
 .
 .
 .
SYSIDMS parameters
 .
 .
 .
FILE
```

More information: For more information on SYSIDMS parameters you can specify, refer to *CA IDMS Common Facilities Guide*.

Appendix B: Syntax Converters for COBOL and PL/I

The IDD syntax converters capture COBOL and PL/I record and element definitions. The output file containing these definitions can be used as input to IDMSDDDL.

The JCL used to execute the IDD syntax converters for COBOL (IDMSIDDC) and for PL/I (IDMSIDDP) is presented in this appendix.

This section contains the following topics:

IDMSIDDC (COBOL Converter) (see page 455)
IDMSIDDP (PL/I Converter) (see page 455)

IDMSIDDC (COBOL Converter)

IDMSIDDC reads a COBOL source program and/or one or more COBOL copy books and converts FILE SECTION 01 and subsequent level statements (including level 88 statements) to DDDL ADD RECORD statements for processing by the DDDL compiler. The following rules apply to executing IDMSIDDC:

- Because the input stream is flushed when IDMSIDDC encounters a WORKING-STORAGE SECTION or PROCEDURE DIVISION header, only one COBOL source program can be processed in a single IDMSIDDC run.
- Although any number of copy books can be concatenated and processed in a single IDMSIDDC run, if the input stream comprises one or more copy books and one COBOL program, the COBOL program must be processed last; IDMSIDDC ignores all copy books encountered following the COBOL program.
- IDMSIDDC ignores COBOL program DATA DIVISION COPY statements. Accordingly, individual books copied must be inserted into the input stream to be converted.
- IDMSIDDC does not support the COBOL SYNC clause at the 01 level.
- The IDMSIDDC input record format is an 80-character card image.

IDMSIDDP (PL/I Converter)

IDMSIDDP reads one or more PL/I copy books and converts the data structures in the DECLARE statements into DDDL ADD ELEMENT and ADD RECORD statements for processing by the DDDL compiler. Any number of copy books can be concatenated for input in a single IDMSIDDP run; the IDMSIDDP input record format is an 80-character card image.

Appendix C: Data Transfer Between Dictionaries

This appendix outlines the procedures for transferring some or all of the data in one dictionary to another dictionary. An overview of the procedure is presented, followed by an example. The DDDL PUNCH and INCLUDE statements facilitate this transfer process, as follows:

- **PUNCH** provides the user with the ability to collect, by means of a single command, all occurrences of all entity types being transferred and to direct the output to an IDD module where it is stored as DDDL syntax.
- **INCLUDE** allows the user to execute the DDDL syntax punched to an IDD module during a PUNCH operation, thereby generating the syntax used to complete the transfer.

This section contains the following topics:

Overview (see page 458)
Steps for Data Transfer (see page 459)
Example of Transferring Data Between Dictionaries (see page 460)
Completing the Data Transfer (see page 461)
Transferring in Batch Mode (see page 461)

Overview

To maintain existing entity relationships while transferring data, the user must transfer entity definitions in the proper order. The following list specifies the order in which entities, entity options, and nests should be added to or replaced in the target dictionary:

- Alternative pictures
- Comment keys
- Relational keys
- Users
- Classes
- Attributes for classes defined with manual attributes
- User-defined entities that IDD will reference
- Systems
- Files
- Elementary elements
- Subordinate elements
- Group elements
- Records
- Modules
- Programs
- User-defined entities that reference IDD entities

The Effect of ATTRIBUTES ARE AUTOMATIC

Attributes within classes assigned the ATTRIBUTES ARE AUTOMATIC qualifier are transferred automatically. Therefore, to facilitate the transfer, it is recommended that the user change all class definitions that include the ATTRIBUTES ARE MANUAL specification to ATTRIBUTES ARE AUTOMATIC and return the specification to ATTRIBUTES ARE MANUAL when the transfer is complete.

Handling Unresolved Relationships

Even if entities are transferred to a dictionary in the order specified above, unresolved relationships may exist if any of the following clauses are present in USER, SYSTEM, PROGRAM, and FILE statements:

Statement	Optional clause

Statement	Optional clause
USER	OF SYSTEM/SUBSYSTEM ACCESS TO SUBSCHEMA/SIGNON QFILE ACCESS TO QFILE ACCESS TO SYSTEM/SUBSYSTEM ACCESS TO FILE SIGNON PROFILE WITHIN USER
SYSTEM	WITHIN SYSTEM/SUBSYSTEM
FILE	RELATED FILE
PROGRAM	PROGRAM CALLED

If any entities being transferred contain one or more of these clauses, modify the definition of those entities to include the unresolved relationships once the transfer is complete.

Steps for Data Transfer

Follow these steps to transfer data from a test dictionary to a production dictionary:

1. Issue PUNCH ALL statements naming each entity type to be transferred and specifying selection criteria, as appropriate. This step creates a file containing PUNCH ELEMENT and PUNCH RECORD statements.
2. Execute the module source in this file by issuing an INCLUDE statement. This step creates a module containing DDDL syntax.

   Note: You must ensure that all entities, entity options, and nests upon which the entities being transferred are dependent exist in the production dictionary.

Each step is described in greater detail in the following example.

Example of Transferring Data Between Dictionaries

All records and elements that were prepared by or revised by user DBA are first transferred from dictionary TEST to dictionary PROD. Then, according to the statements below, the user:

1. Signs on to the dictionary TEST and specifies that module PUNCH-ALL is to receive punched output; the REPLACE verb is used to delete existing source statements associated with PUNCH-ALL.
2. Punches the desired elements and records, ensuring that subordinate elements precede group elements. Note that the PUNCH verb output contains the element/record name and version number only (for example, PUNCH RECORD A); no other entity-type options appear.
3. Issues a REPLACE verb to delete any existing source statements associated with the module DECOMPILE (which is to be the default PUNCH destination).
4. Establishes default DISPLAY/PUNCH processing options. By naming REPLACE as the default verb, the user accommodates record and element definitions that exist in the production dictionary.
5. Issues an INCLUDE statement that executes the source statements in the module PUNCH-ALL. This step punches to the module DECOMPILE the detailed syntax for each element and record being transferred.

At the end of this series of steps, the module DECOMPILE is ready to be transferred to the dictionary PROD.

```
signon dictionary name is test.

replace module name is punch-all version is 1.

punch all elements
        where prepared by is 'dba' or revised by is 'dba'
        to module punch-all version is 1
        verb punch
        as syntax.

punch all records
        where prepared by is 'dba' or revised by is 'dba'
        to module punch-all version is 1
        verb punch
        as syntax.

replace module name is decompile version is 1.

set options for session
        punch to module decompile version is 1
        display as syntax verb replace.
```

```
include module punch-all.
```

Completing the Data Transfer

Assuming that all dependent entities are already defined in dictionary PROD, the user completes the data transfer by using the online DDDL compiler, as follows:

1. Issue a DISPLAY MODULE DECOMPILE VERSION IS 1 AS SYNTAX statement to display the detailed syntax from module DECOMPILE.
2. Retain the displayed syntax, but change the first line of the screen I/O area to SIGNON DICTIONARY NAME IS PROD.
3. Press ENTER to execute the SIGNON statement.
4. Issue an INCLUDE statement naming the module DECOMPILE.

An Alternative Method

An alternative method of completing the transfer is to delete everything in the screen I/O area, up to and including the keywords MODULE SOURCE FOLLOWS, as well as the MSEND statement; only the detailed syntax for the records and elements being transferred should remain. Then, change the first line to SIGNON DICTIONARY NAME IS PROD and press ENTER to sign on to and add each record and element directly to the dictionary PROD.

This method may be preferable in that it requires less space in the dictionary.

Transferring in Batch Mode

To accomplish a transfer in batch mode, you can establish the SYSPCH file as the default PUNCH destination. Following the first batch run, the data set defined to SYSPCH contains the syntax for adding or replacing the desired entities. After editing this data set, you complete the transfer by executing another batch job against dictionary PROD specifying the edited data set as the SYSIPT file.

Appendix D: Default Version Number Conventions

The table in this appendix lists the default version number conventions used by CA IDMS data management components when a reference to an entity occurrence does not include a version number.

CA IDMS component	Action	Version selected
DML precompilers		Highest existing version number
DDL compilers	Adding new records	Highest existing version number plus 1; for a newly defined record, version number is the dictionary default for new version established by SET OPTIONS statement or 1
	Adding new schema records, using SHARE STRUCTURE parameter	Dictionary default for existing version established by SET OPTIONS statement
	Adding new schema elements, using COPY ELEMENTS FROM RECORD syntax	Dictionary default for existing version established by SET OPTIONS statement
	Establishing element occurrences for schema-defined elements	If the named element does exist, version number is 1 If the schema definition matches the DDDL definition of an existing element, current version number is used; if it does not match, current version number plus 1 is used
IDD DDDL compiler	Adding new records	Highest existing version number plus 1; for a newly defined record, version number is the dictionary default for new version established by SET OPTIONS statement or 1
System generation compiler	Creating new entity occurrences	Dictionary default for new version established by SET OPTIONS statement or 1
	Modifying an existing entity occurrence	Dictionary default for existing version established by SET OPTIONS statement

CA IDMS component	Action	Version selected
Mapping compiler	For maps and panels	Highest existing version number; for new maps and panels, version number is 1
CA OLQ	For q-files	Highest existing version number
	Accessing schemas	Highest existing version number; if no schema name is specified, CA OLQ selects first schema under which named subschema was compiled and uses the highest existing version number for that schema
CA Culprit	Accessing files, modules, schemas	Highest existing version number
CA ADS generators (online)		1
CA ADS/Batch transaction processor	For ADL routines	Highest existing version number
	For transactions	1
CA ADS/Batch language translator	For transactions and ADL routines	1

Appendix E: IDD User-Exit Program

This appendix presents the procedures for coding an IDD user-exit program, which is called by the DDDL compiler to:

- Perform security checks
- Enforce entity-occurrence naming conventions

Maintain an audit trail of dictionary activity

This section contains the following topics:

When a User Exit Is Called (see page 466)
Rules for Writing the User-exit Program (see page 467)
Control Blocks and Sample User-exit Programs (see page 469)
Sample IDD User-exit Program (see page 470)

When a User Exit Is Called

The IDD user-exit module is called by the DDDL compiler when it encounters any of these four points:

- **SIGNON/SIGNOFF/COMMIT**

  After the signon procedure is complete and the compiler's security checks have been passed, or immediately after signoff or COMMIT processing.

- **Major command**

  After an ADD, MODIFY, REPLACE, DELETE, DISPLAY/PUNCH, INCLUDE/EXCLUDE, or REMOVE request has been issued. The program is invoked just after the DDDL compiler has identified the entity that is the object of the request and has successfully checked authorization requirements. Object entities can be any standard or user-defined entity type; any element, file, or record synonym, or any record element, COBOL element, or view.

- **Card image**

  After each input statement (card image) is passed to the user-exit control block after the statement has been:

  - Scanned and printed on the Integrated Data Dictionary Activity List
  - Displayed at the terminal
  - Written to the print file (online DDDL compiler interface only)

  The data administrator can build an audit trail of accesses and updates to the dictionary.

- **End of converse**

  When one of the following occurs, the user can perform a termination activity, such as a write-to-log:

  - The user presses ENTER during an online DDDL session
  - A batch run of the DDDL compiler processes a SIGNOFF statement
  - A batch run of the DDDL compiler detects an end-of-file condition.

Rules for Writing the User-exit Program

This section describes the rules that apply to writing the user-exit program.

Language

You can write the user-exit module in any language that supports OS calling conventions. However, it is recommended that you write user-exit modules in Assembler to allow the online DDDL compiler to remain reentrant.

Note: User-exit modules cannot be CA ADS dialogs.

Versions

You can code and maintain separate versions of user-exit modules for the batch and online DDDL compilers, or you can code modules that can be executed both in batch mode and online.

Macros

The user-exit facility supports all CA IDMS/DC macros for exits to be used with the online DDDL compiler. For exits to be used with the batch DDDL compiler, the only CA IDMS/DC macros supported are: #WTL, #ABEND, #GETSTG, #FREESTG, #LOAD, and #DELETE; under VSE/ESA, the only valid form of #DELETE is EPADDR=.

Run units

You can start a run unit within an exit, however you should ensure that the run unit does not deadlock with the DDDL compiler run unit. If a user-exit run unit will access a dictionary area, the run unit should ready the object area in a retrieval usage mode.

Entry point

The entry point of the user exit invoked by the batch and online compilers differ.

Compiler Name	Description	User Exit Entry Point
IDMSDDDL	Batch DDDL compiler	IDDEXITB
IDMSDDDC	Online DDDL compiler	IDDEXITO

Although each exit has a unique entry point name, you can use the same exit code for more than one compiler by assigning multiple entry point names to the same set of code.

Interface

User exits written in COBOL to run under the online DDDL compiler require a user-exit interface, written in Assembler with an entry point of IDDUXITO, to be link edited with IDMSUXIT. This interface should issue a #LINK to the COBOL program (with an entry point other than IDMSDDDC) to isolate it from IDMSDDDC, which is storage protected.

Register Conventions

User-exit modules are called using the following OS register conventions:

```
R15        Entry point of module IDDUXIT

R14        Return address

R13        18 fullword SAVEAREA

R1         Fullword parameter list
```

Parameters 3 and 4

For all four types of user exits, parameter 1 points to a user-exit control block and parameter 2 points to a SIGNON element block. The information addressed in parameters 3 and 4 varies based on the type of user exit, as follows:

- For the **SIGNON/SIGNOFF/COMMIT** and **end-of-conversation** exits, parameter 3 points to a SIGNON block.
- For the **major command user exit**, parameter 3 points to an entity control block.
- For the **card-image** user exit, parameter 3 points to a card-image control block.
- For **all user exits except the card-image user exit**, parameter 4 is reserved for use by IDD and should be defined as a PIC X(80) field in the user-exit module.
- For the **card-image user exit**, parameter 4 points to the input card image, which is defined as a PIC X(80) field.

The user-exit control blocks are described separately later in this appendix.

Information Modification

With the exception of the fields identified within the user-exit control block described below, a user-exit module should not modify any of the information passed.

Return Codes

On return from a user-exit module, the user must set a return code and, optionally, specify a message ID and message text to be issued by the DDDL compiler, as follows:

Code	IDD action
0	No message is issued; IDD continues with normal processing.
1	An informational message is issued; IDD continues with normal processing.
4	A warning message is issued; IDD continues with normal processing.
8	An error message is issued; IDD initiates error processing.

Control Blocks and Sample User-exit Programs

This section presents the formats of these five control blocks:

- User-exit control block
- SIGNON element block
- SIGNON block
- Entity control block
- Card-image control block

Sample IDD User-exit Program

The following sample IDD user-exit program can be used to enforce naming conventions for elements. The source code for this program can be found in the installation source library under member name IDDSUXIT.

```
*********************************************************************
IDDUXIT  TITLE '         NAMING CONVENTION CHECKER'
*********************************************************************
*
*
*  PROGRAM NAME : IDDUXIT
*
*  DATE         : mm/dd/yy
*
*
*  DESCRIPTION  : THIS IS AN EXAMPLE OF AN IDD USER EXIT.  THIS
*
*                 PROGRAM SHOWS HOW A SHOP COULD CHECK THE ENTITY
*
*                 NAMES FOR A SHOP STANDARD.  ANY VIOLATIONS OF
*
*                 THE NAMING CONVENTION ARE TREATED AS AN IDD ERROR
*
*                 AND THE ACTION (ADD, MOD, DEL) IS NOT ALLOWED.
*
*********************************************************************
*
IDDUXIT  CSECT
         #REGEQU
         ENTRY IDDEXITO
IDDEXITO DS    0H                    Online DDDL compiler entry
         ENTRY IDDEXITB
IDDEXITB DS    0H                    Batch DDDL compiler entry
*********************************************************************
*
*        SET UP ADDRESSABILITY
*
*********************************************************************
*
         STM   R14,R12,12(R13)      SAVE CALLERS REGISTERS
         LR    R12,R15
         USING IDDUXIT,R12
         L     R4,12(R1)             GET THE
         L     R3,8(R1)                 CORRECT
         L     R2,4(R1)                    PARAMETER
         L     R1,0(R1)                       ADDRESSES
*
IDDUXITR DS    0H                    BASE THE CONTROL BLOCKS
```

```
*
         USING UXITCB,R1             USER EXIT CONTROL BLOCK
         MVC   UXITRCDE,F0           ZERO OUT THE RETURN CODE
         MVC   UXITMID(8),BLANKS     BLANK OUT THE MESSAGE ID
         MVC   UXITMTXT(80),BLANKS   BLANK OUT THE MESSAGE
*
*********************************************************************
*
*        INTERROGATE THE IDD COMMAND
*
*********************************************************************
*
         SPACE
UXIENTY  EQU   *
         USING UXITECB,R3            ENTITY CONTROL BLOCK
*
         CLC   UXITEVRB,UXICSON        IS IT AN SIGNON?
         BE    USIGNON                   YES, CHECK THE USER NAME
*
         CLC   UXITEVRB,UXICARD        IS IT AN CARD IMAGE EXIT?
         BE    UCARD                     YES, CHECK THE CARD
*
         CLC   UXITEVRB,UXICADD        IS IT AN ADD?
         BE    UXIECHK                   YES, CHECK THE ENTITY-NAME
*
         CLC   UXITEVRB,UXICMOD        IS IT A MODIFY?
         BE    UXIECHK                   YES, CHECK THE ENTITY-NAME
*
         CLC   UXITEVRB,UXICREP        IS IT A DELETE?
         BE    UXIECHK                   YES, CHECK THE ENTITY-NAME
*                                        NO
         MVC   UXITMID(8),ELSEID      MOVE IN 'ELSE' MESSAGE ID
         MVC   UXITMTXT(80),ELSEMSG   MOVE IN 'ELSE' MESSAGE
         BNE   UXIEBYE
*                                        YES, CHECK THE ENTITY-NAME
*********************************************************************
*
*        CHECK THE CARD IMAGE
*
*********************************************************************
*
         SPACE
UCARD    EQU   *
*
         MVC   UXITMID(8),CARDID     FILL IN THE MESSAGE ID
         MVC   UXITMTXT(80),CARDMSG  FILL IN THE MESSAGE TEXT
         B     UXIEBYE               BACK TO THE LAND OF IDD
*
*********************************************************************
```

```
*
*        CHECK THE USER NAME FOR ME
*
*********************************************************************
*
         SPACE
USIGNON  EQU   *
*
         USING UXITSEB,R2            SIGNON ELEMENT BLOCK
         USING UXITSB,R3             SIGNON BLOCK
*
         CLC   UXITUSER(3),WHOME     IS IT ME
         BE   UXIEDC                  YES GO CHECK FOR DC NAME
*                                    NO, GO TO JAIL, GO DIRECTLY TO
*                                    JAIL, DO NOT PASS GO DO NOT
USNAME   EQU   *                     COLLECT $200.
         MVC   UXITRCDE,F8           FILL IN THE RETURN CODE
         MVC   UXITMID(8),NOSNID     FILL IN THE MESSAGE ID
         MVC   UXITMTXT(80),NOSNMSG  FILL IN THE MESSAGE TEXT
         B     UXIEBYE               BACK TO THE LAND OF IDD
*

UXIEDC   EQU   *
         TM    UXITFLG1,UXIT1DC      ARE WE RUNNING DC
         BZ    UXIEBYE                 NO, SKIP DC ID CHECK
*
         CLC   UXITUSER,UXITIUSR     IS THE IDD USER THE SAME AS DC
         BE    UXIEBYE                 YES, OK LET IT PASS
*                                    NO, DON'T LET THEM SIGNON
         MVC   UXITRCDE,F8           FILL IN THE RETURN CODE
         MVC   UXITMID(8),NODCID     FILL IN THE MESSAGE ID
         MVC   UXITMTXT(80),NODCMSG  FILL IN THE MESSAGE TEXT
         B     UXIEBYE               BACK TO THE LAND OF IDD
*
*********************************************************************
*
*        CHECK THE ENTITY-NAME FOR VALID NAMING CONVENTION
*
*********************************************************************
*
         SPACE
UXIECHK  EQU   *
         USING UXITECB,R3            ENTITY CONTROL BLOCK
*
         CLC   UXITENME(3),NAMECHK   DOES THE NAME FOLLOW THE RULES?
         BE    UXIEBYE                 YES, LET THIS ONE PASS.
*                                      NO, RETURN AN ERROR
*
         MVC   UXITRCDE,F8           FILL IN THE RETURN CODE
         MVC   UXITMID(8),NONOID     FILL IN THE MESSAGE ID
```

```
         MVC   UXITMTXT(80),NONOMSG  FILL IN THE MESSAGE TEXT
*
*****************************************************************
*
*        RETURN BACK TO IDD
*
*****************************************************************
*
         SPACE
UXIEBYE  EQU   *
         LM    R14,R12,12(R13)       RELOAD CALLER'S REGISTERS
         BR    R14                   RETURN TO CALLER
         EJECT
*****************************************************************
*        CONSTANTS AND LITERALS                                 *
*****************************************************************
UXICADD  DC    CL16'ADD             '
UXICMOD  DC    CL16'MODIFY          '
UXICREP  DC    CL16'REPLACE         '
UXICSON  DC    CL16'SIGNON          '
UXICARD  DC    CL16'CARD IMAGE      '
NAMECHK  DC    CL3'XYZ'
WHOME    DC    CL3'XYZ'
WKLEN    DC    F'100'
NONOID   DC    CL8'DC999001'
NONOMSG  DC    CL80'NAMING CONVENTION VIOLATED - ACTION NOT ALLOWED'
NOSNID   DC    CL8'DC999002'

NOSNMSG  DC    CL80'SIGNON ERROR - USER NOT ALLOWED ACCESS TO IDD'
NODCID   DC    CL8'DC999003'
NODCMSG  DC    CL80'SIGNON ERROR - IDD USER NAME NOT DC USER NAME'
CARDID   DC    CL8'DC999004'
CARDMSG  DC    CL80'MESSAGE PRODUCED BY CARD IMAGE EXIT          '
ELSEID   DC    CL8'DC999005'
ELSEMSG  DC    CL80'MESSAGE PRODUCED BY CARD IMAGE EXIT          '
BLANKS   DC    CL80' '
F0       DC    F'0'              NORMAL RETURN CODE - NO ERRORS
F2       DC    F'1'              INFORMATION MESSAGE
F4       DC    F'4'              WARNING MESSAGE
F8       DC    F'8'              ERROR MESSAGE
*****************************************************************
*        USER EXIT CONTROL BLOCK                                *
*****************************************************************
UXITCB   DSECT
UXITCPLR DS    CL8           COMPILER NAME 'IDMSDDDL'
UXITDATE DS    CL8           COMPILER START DATE MM/DD/YY
UXITTIME DS    CL8           COMPILER START TIME HHMMSSMM
UXITWORK DS    F             USER FULLWORD INITIALIZED TO 0
UXITRCDE DS    0F            RETURN CODE RETURNED BY USER
         DS    XL3           UNUSED
```

```
UXITRC   DS    X
UXITRC00 EQU   X'00'              NORMAL RETURN CODE - NO ERRORS
UXITRC01 EQU   X'01'              INFORMATION MESSAGE
UXITRC04 EQU   X'04'              WARNING MESSAGE
UXITRC08 EQU   X'08'              ERROR MESSAGE
UXITMID  DS    CL8           USER MESSAGE ID RETURNED BY USER
UXITMTXT DS    CL80          USER MESSAGE TEXT RETURNED BY USER
UXITCBLN EQU   *-UXITCB      USER EXIT CONTROL BLOCK LENGTH
*********************************************************************
*        USER EXIT SIGNON ELEMENT BLOCK                             *
*********************************************************************
UXITSEB  DSECT
UXITIDLN DS    X             LENGTH OF USERID FOR #WTL'S
UXITUSER DS    CL32          USER ID
         DS    0A            ROUND UP TO FULLWORD
UXITSNLN EQU   *-UXITSEB     LENGTH OF IDD SIGNON ELEMENT
*********************************************************************
*        USER EXIT SIGNON BLOCK                                     *
*********************************************************************
UXITSB   DSECT
UXITTYPE DS    CL16          IDD VERB
UXITDICT DS    CL8           DICTIONARY NAME
UXITNODE DS    CL8           NODE NAME
UXITIUSR DS    CL32          IDD USER ID

UXITIPSW DS    CL8           IDD USER'S PASSWORD
UXITFLG0 DS    CL1           ENVIRONMENT FLAG
UXIT0DOS EQU   X'80'            COMPILER RUNNING UNDER VSE/ESA
UXIT0MEN EQU   X'40'            RUNNING UNDER 'IDD MENU' MODE
UXITFLG1 DS    CL1           ENVIRONMENT FLAG
UXIT1LCL EQU   X'80'            COMPILER RUNNING IN INTERNAL SUBROUTINE MO
UXIT1DC  EQU   X'40'            COMPILER RUNNING UNDER DC
         DS    CL2           RESERVED FOR FUTURE FLAGS
         DS    CL20          RESERVED
UXITDMLM DS    H             DDLDML USAGE MODE
*                                36=UPDATE
*                                37=PROTECTED UPDATE
*                                38=RETRIEVAL
UXITLODM DS    H             DDLDCLOD USAGE MODE
UXITMSGM DS    H             DDLDCMSG USAGE MODE
         DS    CL10          RESERVED
UXITSLEN EQU   *-UXITSB      LENGTH OF USER EXIT SIGNON BLOCK
*********************************************************************
*        USER EXIT ENTITY CONTROL BLOCK                             *
*********************************************************************
UXITECB  DSECT
UXITEVRB DS    CL16          IDD VERB
UXITENTY DS    CL32          IDD ENTITY-TYPE
UXITENME DS    CL40          ENTITY NAME
UXITEVER DS    H             VERSION
```

```
UXITEADQ DS    CL64          ADDITIONAL QUALIFIER
*              CL40               LANGUAGE (ENTITY TYPE = MODULE)
*              CL20               CLASS    (ENTITY TYPE = ATTRIBUTE)
UXITPREP DS    CL32          PREPARED BY USER NAME
UXITREV  DS    CL32          REVISED BY USER NAME
UXITELEN EQU   *-UXITECB     LENGTH OF USER EXIT ENTITY CONTROL BLK
*****************************************************************
*        END OF EXIT                                            *
*****************************************************************
         END
```

Appendix F: Using the DDDL Compiler as a Subprogram

This section contains the following topics:

IDMSDB--Overview (see page 477)
Compiler Interface Parameter List (see page 479)
Work-area File (see page 479)
Sample Program that Calls IDD (see page 480)

IDMSDB--Overview

Any program can call the DDDL compiler (IDMSDDDC) as a subroutine to extract information from or update information in the dictionary. The program or dialog passes to IDMSDDDC an input file that contains the DDDL statements to be used to obtain the desired information. The DDDL compiler places the extracted data in an output file, which can be examined and processed by the program or dialog.

The DDDL compiler uses these files:

- An input file (SYSIPT)
- A print file (SYSLST)
- A punch file (SYSPCH)

Each of these files consists of 80-byte records. Normally, the compiler controls these files, directing the input and print files to the terminal and discarding the punch file. However, when a program or dialog calls IDMSDDDC as a subroutine, the calling program specifies that these three files can be directed to a work-area file, a scratch area, a queue, another program, or a null file. Advantages and disadvantages associated with each of these storage mechanisms are as shown in the following table.

Storage type	Advantages/disadvantages
Work-area file	Offers the fastest access but is limited in size; this mechanism is the best choice for small files.
Scratch area	Can accommodate a large volume of data; however, scratch areas are volatile and may require the calling program to perform I/Os.

Queue	Can accommodate a large volume of data; however, the calling program must perform I/Os and initiate run units to access queues.
User program exit	Offers the most advantages. The user has maximum control over the file's records, selecting certain records for special processing.
Null file	Suppresses the output from IDMSDDDC. If the program tries to read the null file, an end-of-file condition is returned immediately.

Combining Storage Types

It may be advantageous to combine two mechanisms. For example, direct the file to a user program exit that directs a work area's overflow to a scratch area. The work area is described under Work-area File.

Input File Statements

The input file can contain any valid DDDL statements. All standard compiler security applies to issuing these statements.

First Two Statements— SIGNON and SET OPTIONS

To ensure proper access to the compiler, it is recommended that the first two commands in the input file be SIGNON and SET OPTIONS.

The dictionary named or defaulted to in the SIGNON command must match the dictionary being used by the calling program.

You may also wish to specify SET OPTIONS FORMAT IS FIXED because the resulting columnar format is easier for the calling program to parse. Note, however, that the columnar format associated with each entity-occurrence definition is subject to change from release to release. Therefore, the user program should be coded in such a way as to easily accommodate such changes.

You should ensure that null values are not passed to the DDDL compiler as part of an entity name; if nulls are present, the compiler will not be able to locate the object entity.

Last Statement— SIGNOFF

When a user program calls the compiler interface, that program automatically starts a compiler session. Therefore, the last command passed to the compiler by the program should be a SIGNOFF command. If the SIGNOFF command is not present in the input file, the compiler interface suspends the compiler session. If the calling program terminates, the compiler session remains suspended. If the user then signs on to the compiler from the same logical terminal, the suspended session is reactivated and any session options established by the calling program remain in effect; however, the work file has been cleared.

How the Compiler Is Called

To call the DDDL compiler, the user program issues a LINK request, naming the module IDMSDDDC and passing seven parameters: a compiler input/output (CIO) block, one compiler input/output file (CIOF) block for each of the three IDD files (input, print, and punch), followed by a user parameter for each of the three files (input, print, and punch).

Compiler Interface Parameter List

The CIO block, CIOF block, and user parameters are described separately in this section.

Work-area File

The work-area file is a block of program variable storage that contains a series of 80-byte records. The following rules apply to the work-area file:

- The maximum number of records in the work-area file must be placed in the CIOFSZMX field of the applicable CIOF block by the user program before the program invokes the compiler interface.
- The size of the work-area file is determined by the user program; it must be a multiple of 80.
- If an compiler output file is exhausted when the work-area file is written to by the compiler, a return code of 28 (X'1C') is placed in the CIOIORC field of the CIO block and the excess records are lost.
- If the compiler file is exhausted when the work-area file is read from by the compiler, an end-of-file condition is returned to the compiler.

Upon return to the user program, the CIOFSZUS field contains the number of records actually read from or written to the file.

Sample Program that Calls IDD

The following sample COBOL program calls IDD and requests IDD to display an element.

```
*************************************************************
 IDENTIFICATION DIVISION.
*************************************************************
 PROGRAM-ID.             CALLIDD.
 DATE WRITTEN.           MONTH DD, YEAR
 DATE COMPILED.
*************************************************************
* REMARKS.
*************************************************************
*
*  THIS IS A SAMPLE DC COBOL PROGRAM THAT DEMONSTRATES HOW
*  AN APPLICATION PROGRAM CAN CALL IDD AS A SUBPROGRAM AND
*  PASS TO IDD A REQUEST TO DISPLAY AN ELEMENT.  THE OUTPUT
*  OF THE REQUEST IS DISPLAYED BY THE COBOL PROGRAM.
*
*************************************************************
 ENVIRONMENT DIVISION.
*************************************************************
 CONFIGURATION SECTION.
 SOURCE-COMPUTER.        IBM-370.
 OBJECT-COMPUTER.        IBM-370.

 IDMS-CONTROL SECTION.
 PROTOCOL.               MODE IS IDMS-DC DEBUG
                         IDMS-RECORDS MANUAL.
*************************************************************
 DATA DIVISION.
*************************************************************
 MAP SECTION.
     MAP CDSIMAP1.
     EJECT
*************************************************************
 WORKING-STORAGE SECTION.
*************************************************************
 01  BEGIN-WS.
     03 FILLER                  PIC X(40) VALUE
        '***** WORKING STORAGE BEGINS HERE ******'.
*************************************************************
*    SWITCHES-AREA - PROGRAM CONTROL SWITCHES               *
*************************************************************
 01  SWITCHES-AREA.
     03 FILLER                  PIC X(08) VALUE 'SWITCHES'.
     03 IDD-EOF-SW              PIC 9 VALUE 0.
        88 IDD-EOF              VALUE 1.
     03 FIRST-TIME-SW           PIC 9 VALUE 0.
```

```
           88 FIRST-TIME            VALUE 1.
        03 ERROR-SW                 PIC 9 VALUE 0.
           88 NO-ERRORS             VALUE 0.
      ************************************************************
      *    WORK-FIELDS   - PROGRAM WORK FIELDS                   *
      ************************************************************
       01  WORK-FIELDS.
           03 FILLER                 PIC X(08) VALUE 'WORKAREA'.
           03 SUB                    PIC 99 VALUE 0.

     03 AID-BYTE                 PIC X.
        88 CLEAR-HIT             VALUE '_'.
        88 PA1-HIT               VALUE '%'.
     03 TASK-CODE                PIC X(8).
     03 GOOD-RC                  PIC S9(8) COMP VALUE +0.
     03 Q-EL                     PIC X(32) VALUE ALL '?'.
*****************************************************************
*    MESSAGES-AREA - OPERATOR MESSAGES                          *
*****************************************************************
 01  MESSAGES-AREA.
     03 FILLER                   PIC X(08) VALUE 'MESSAGES'.
     03 OK-MSG                   PIC X(40) VALUE
        'PROCESSING COMPLETE - PROCEED          '.
     03 NO-ELEMENT-MSG           PIC X(40) VALUE
        'ELEMENT NAME MISSING, PLEASE FILL IT IN'.
     03 CIO-ERROR-MSG.
        05 FILLER                PIC X(36) VALUE
        'CIO PROCESSING ERROR - RETURN CODE ='.
        05 CIO-ERROR-CODE        PIC X(4) VALUE '0000'.
*****************************************************************
*    SCR-RCD       - SCRATCH RECORD AREA                        *
*****************************************************************
 01  SCR-RCD.
     03 SCR-DBK                  PIC S9(8) COMP.
     03 SCR-RCDID                PIC S9(8) COMP.
     03 SCR-STATUS               PIC X.
     03 SCR-RCD-END              PIC X.
     EJECT
*****************************************************************
*    PARAMETER 1 - THE COMPILER INOUT/OUTPUT BLOCK              *
*****************************************************************
 01  CIO-PARM1.
     03 CIO-ID                   PIC X(4)  VALUE 'CIO '.
     03 CIO-USER                 PIC S9(8) COMP VALUE +0.
     03 CIO-IO-RC                PIC S9(8) COMP VALUE +0.
     03 CIO-DDDL-RC              PIC S9(8) COMP VALUE +0.
     03 CIO-RESERVED             PIC X(8)  VALUE SPACES.
     03 CIO-ERROR-FILE           PIC X(8)  VALUE SPACES.
        88 SYSIPT-ERROR          VALUE 'SYSIPT'.
        88 SYSLST-ERROR          VALUE 'SYSLST'.
```

```
        88 SYSPCH-ERROR          VALUE 'SYSPCH'.
     03 CIO-NULL                 PIC X(4) VALUE 'NULL'.
*****************************************************************
*    PARAMETER 2 - CIOF INPUT BLOCK                             *
*****************************************************************
 01  CIO-PARM2.
     03 CIOF-I-TYPE              PIC X(8) VALUE 'WORKAREA'.
     03 CIOF-I-NAME              PIC X(16) VALUE SPACES.
     03 CIOF-I-F-RC              PIC S9(8) COMP VALUE +0.
     03 CIOF-I-SIZE-US           PIC S9(8) COMP VALUE +0.
     03 CIOF-I-SIZE-MAX          PIC S9(8) COMP VALUE +5.

*****************************************************************
*    PARAMETER 3 - INPUT DATA AREA                              *
*****************************************************************
 01  CIO-PARM3.
     03 FILLER                   PIC X(80) VALUE
        '        SIGNON.            '.
     03 FILLER                   PIC X(80) VALUE
        '        DISPLAY ELEMENT NAME IS'.
     03 CIO-I-LINE2.
        05 FILLER                PIC X(8) VALUE SPACES.
        05 CIO-I-NAME            PIC X(32) VALUE SPACES.
        05 FILLER                PIC X(40) VALUE SPACES.
     03 FILLER                   PIC X(80) VALUE
        '        .'.
     03 FILLER                   PIC X(80) VALUE
        '        SIGNOFF.           '.
*****************************************************************
*    PARAMETER 4 - CIOF OUTPUT BLOCK                            *
*****************************************************************
 01  CIO-PARM4.
     03 CIOF-O-TYPE              PIC X(8) VALUE 'WORKAREA'.
     03 CIOF-O-NAME              PIC X(16) VALUE SPACES.
     03 CIOF-O-F-RC              PIC S9(8) COMP VALUE +0.
     03 CIOF-O-SIZE-US           PIC S9(8) COMP VALUE +0.
     03 CIOF-O-SIZE-MAX          PIC S9(8) COMP VALUE +100.
*****************************************************************
*    PARAMETER 5 - OUTPUT DATA AREA                             *
*****************************************************************
 01  CIO-PARM5.
     03 CIOF-OUTPUT-LINE         PIC X(80)
         OCCURS 100 TIMES.
     EJECT
*****************************************************************
*    PARAMETER 6 - CIOF PUNCH BLOCK                             *
*****************************************************************
 01  CIO-PARM6.
     03 CIOF-P-TYPE              PIC X(8) VALUE 'WORKAREA'.
     03 CIOF-P-NAME              PIC X(16) VALUE SPACES.
```

```
     03 CIOF-P-F-RC             PIC S9(8) COMP VALUE +0.
     03 CIOF-P-SIZE-US          PIC S9(8) COMP VALUE +0.
     03 CIOF-P-SIZE-MAX         PIC S9(8) COMP VALUE +0.
*****************************************************************
*    PARAMETER 7 - PUNCH DATA AREA                              *
*****************************************************************
 01  CIO-PARM7                 PIC X(80) VALUE 'NULL'.
*****************************************************************
*    IDMS AREA                                                  *
*****************************************************************
 COPY IDMS SUBSCHEMA-CTRL.
 COPY IDMS MAP-CONTROLS.
 COPY IDMS MAP-RECORDS.
     EJECT
 PROCEDURE DIVISION.

      ***************************************************************
      *                                                             *
      * ROUTINE - 0000-MAIN-LINE                                    *
      *                                                             *
      * THIS ROUTINE IS THE MAIN CONTROL OF THE PROGRAM, CALLING    *
      * THE OTHER ROUTINES TO DO THE ACTUAL WORK.                   *
      *                                                             *
      ***************************************************************
       0000-MAIN-LINE.
           PERFORM 1000-GET-SCRATCH-REC THRU 1999-EXIT.
           IF FIRST-TIME
               PERFORM 2000-DISPLAY-MAP THRU 2999-EXIT
               GO TO 0800-RETURN-SCREEN.
           PERFORM 3000-GET-MAP THRU 3999-EXIT.
           IF CLEAR-HIT
               GO TO 0900-DC-RETURN.
           PERFORM 4000-EDIT-DATA THRU 4999-EXIT.
           IF NO-ERRORS
               PERFORM 5000-CALL-IDD THRU 5999-EXIT.
           MAP OUT USING CDSIMAP1 WAIT IO OUTPUT DATA YES.

      ***************************************************************
      *                                                             *
      * ROUTINE - 0800-RETURN-SCREEN                                *
      *                                                             *
      * THIS ROUTINE SETS UP THE RETURN SO THAT THIS TRANSACTION    *
      * WILL BE THE NEXT TRANSACTION EXECUTED FROM THE TERMINAL.    *
      *                                                             *
      ***************************************************************
       0800-RETURN-SCREEN.
           ACCEPT TASK CODE INTO TASK-CODE.
           DC RETURN NEXT TASK CODE TASK-CODE.
      ***************************************************************
      *                                                             *
```

```
* ROUTINE - 0900-DC-RETURN                                      *
*                                                               *
* THIS ROUTINE DELETES THE SCRATCH RECORD AND THEN RETURNS      *
* CONTROL TO THE DC SYSTEM.                                     *
*                                                               *
*****************************************************************
 0900-DC-RETURN.
     DELETE SCRATCH RECORD ID SCR-RCDID.
     DC RETURN.
     EJECT
*****************************************************************
*                                                               *
* ROUTINE - 1000-GET-SCRATCH-REC.                               *
*                                                               *
* THIS ROUTINE ATTEMPTS TO GET THE SCRATCH RECORD, WHICH        *
* IS USED TO DETERMINE IF THIS IS THE FIRST TIME THE            *
* TRANSACTION HAS BEEN EXECUTED.                                *
*                                                               *
*****************************************************************

 1000-GET-SCRATCH-REC.
     MOVE 1 TO SCR-RCDID.
     GET SCRATCH RECORD ID SCR-RCDID KEEP
         INTO SCR-RCD TO SCR-RCD-END
         ON ANY-ERROR-STATUS
            IF ERROR-STATUS NOT = '0000'
                MOVE 1 TO FIRST-TIME-SW
            ELSE
                MOVE 0 TO FIRST-TIME-SW.
 1999-EXIT.
     EXIT.
*****************************************************************
*                                                               *
* ROUTINE - 2000-DISPLAY-MAP                                    *
*                                                               *
* THIS ROUTINE CREATES A SCRATCH RECORD AND DOES THE INITIAL    *
* MAP OUT.                                                      *
*                                                               *
*****************************************************************
 2000-DISPLAY-MAP.
     MOVE 0 TO SCR-DBK.
     MOVE '1' TO SCR-STATUS.
     PUT SCRATCH FROM SCR-RCD TO SCR-RCD-END
         RECORD ID SCR-RCDID.
     PERFORM 8000-INITILIZE-MAP THRU 8099-EXIT.
     MAP OUT USING CDSIMAP1 OUTPUT NEWPAGE.
 2999-EXIT.
     EXIT.
*****************************************************************
*                                                               *
```

```
* ROUTINE - 3000-GET-MAP                                        *
*                                                               *
* THIS ROUTINE GETS THE MAP.                                    *
*                                                               *
*****************************************************************
 3000-GET-MAP.
     PERFORM 8000-INITILIZE-MAP THRU 8099-EXIT.
     MAP IN USING CDSIMAP1.
     INQUIRE MAP CDSIMAP1 MOVE AID TO AID-BYTE.
 3999-EXIT.
     EXIT.

     EJECT
*****************************************************************
*                                                               *
* ROUTINE - 4000-EDIT-DATA                                      *
*                                                               *
* THIS ROUTINE CHECKS THE ELEMENT NAME TO SEE IF IT HAS BEEN    *
* FILLED IN.  IF IT IS BLANK OR NULLS, AN ERROR MESSAGE IS      *
* DISPLAYED, AND THE MAP IS RETURNED TO THE OPERATOR FOR        *
* CORRECTION.                                                   *
*                                                               *
*****************************************************************

 4000-EDIT-DATA.
     MOVE 0 TO ERROR-SW.
     IF (CDSIELNM = SPACES)
     OR (CDSIELNM = LOW-VALUES)
         MOVE 1 TO ERROR-SW
         MOVE NO-ELEMENT-MSG TO CDSIMSG
         MOVE Q-EL TO CDSIELNM
         MODIFY MAP CDSIMAP1 TEMPORARY
             FOR CDSIELNM ATTRIBUTES BRIGHT
         GO TO 4999-EXIT.
     MOVE CDSIELNM TO CIO-I-NAME.
 4999-EXIT.
     EXIT.
*****************************************************************
*                                                               *
* ROUTINE - 5000-CALL-IDD                                       *
*                                                               *
* THIS ROUTINE CALLS IDD, PASSING THE SEVEN PARAMETERS THAT     *
* ARE REQUIRED.  IF THE RETURN CODE FROM IDD IS GOOD (ALL       *
* BINARY ZEROS) THE FIRST TEN LINES FROM THE CIOF OUTPUT        *
* WORKAREA (THE IDD SYSLST FILE) ARE MOVED TO THE MAP.          *
* IF THE RETURN CODE FROM IDD IS BAD (NOT BINARY ZEROS) AN      *
* ERROR MESSAGE IS DISPLAYED WITH THE ERROR CODE.               *
*                                                               *
*****************************************************************
 5000-CALL-IDD.
```

```
            TRANSFER CONTROL TO 'IDMSDDDC' RETURN
                USING CIO-PARM1
                      CIO-PARM2
                      CIO-PARM3
                      CIO-PARM4
                      CIO-PARM5
                      CIO-PARM6
                      CIO-PARM7.
            IF CIO-IO-RC NOT = GOOD-RC
                MOVE CIO-IO-RC TO CIO-ERROR-CODE
                MOVE CIO-ERROR-MSG TO CDSIMSG
                GO TO 5999-EXIT.
            PERFORM 5100-MOVE-IDD-OUTPUT THRU 5109-EXIT
                VARYING SUB FROM 1 BY 1
                    UNTIL IDD-EOF.
            MOVE OK-MSG         TO CDSIMSG.
            GO TO 5999-EXIT.
        5100-MOVE-IDD-OUTPUT.
            MOVE CIOF-OUTPUT-LINE(SUB) TO CDSILINE(SUB).
            IF (SUB = 10) OR (SUB = CIOF-O-SIZE-US)
                MOVE 1 TO IDD-EOF-SW.
        5109-EXIT.
            EXIT.
        5999-EXIT.
            EXIT.

            EJECT

*****************************************************************
*                                                               *
* ROUTINE - 8000-INITILIZE-MAP                                  *
*                                                               *
* THIS ROUTINE DOES THE IDMS MAP BINDS.                         *
*                                                               *
*****************************************************************
 8000-INITILIZE-MAP.
     COPY IDMS MAP-BINDS.
 8099-EXIT.
     EXIT.
 EJECT
     COPY IDMS IDMS-STATUS.
 IDMS-ABORT.
 IDMS-ABORT-EXIT.
     EXIT.
```

Appendix G: Double-Byte Character Set (DBCS) Strings

This section contains the following topics:

Overview (see page 488)
Coding DBCS Strings (see page 489)

Overview

You can define data to handle double-byte character set (DBCS) strings in the dictionary, provided that your terminal has DBCS hardware installed. A double-byte character set uses two bytes to express a single character. This means that you can work with nonroman (non-EBCDIC) alphabets, such as the Kanji alphabet used in Japan or Chinese characters, used in Taiwan.

With DBCS support, you can:

- Use DBCS characters in user-supplied variables
- Assign a special type of DBCS string, called a graphic (G-) literal, to values that initialize an element or are used in conditional expressions
- Assign graphic external pictures to RECORD ELEMENT and COBOL substatements
- Define DBCS data-editing criteria in code and edit tables

Variables for Which You Can Use DBCS: You can code a DBCS character string for most variables in the DDDL syntax that require the use of quotation marks. The variables listed below can accommodate DBCS strings:

- comment-key
- comment-text
- condition-value
- decode-value
- description-text
- encode-value
- end-value
- initial-value
- inverse-relational-key
- numeric-literal in WHERE clause expressions that do not specify the CONTAINS or MATCHES options
- message-text
- password
- start-value
- user-text
- value

You cannot use a DBCS character string for the user ID, dictionary name, node name, or database name.

496 CA

Coding DBCS Strings

To code a DBCS string

1. Type the site-specific quote character.
2. Switch the terminal to DBCS.

   The computer hardware automatically inserts a **shiftout character ([SO])**.
3. Type the double-byte character string.
4. Switch the terminal to EBCDIC.

   The computer hardware automatically inserts a **shiftin character ([SI])**.
5. Type the site-specific quote character.

Depending on your terminal hardware, the shift code occupies from one to three bytes. The shift codes are invisible. However, on an IBM-5550 machine, a shift code occupies a screen position; on Fujitsu hardware, it does not.

Examples

This first example uses a double-byte character string in the COMMENTS clause of the DESTINATION statement.

```
add destination name is dest01
    comments is '[so]dbcs-character string[si]' .
```

This second example uses a double-byte character string in the VALUE clause of a COBOL Record Element. Note that the [si] and [so] sequences, which are located at the very beginning and end of the quoted string, must be coded as 1 byte hexadecimal characters for the z/OS, z/VM and z/VSE platforms. The hex representation is provided for clarity.

```
_____________________________________________________________
          02 FIELD-5            PIC G(5) DISPLAY-1
 44444444444FF4CCCDC6F4444444444444DCC4C4F54CCEDDCE6F444444
 00000000000020695340500000000000079307D5D0492731801000000
_____________________________________________________________
                                   VALUE G' DBCS-STR ' .
 44444444444444444444444444444444444ECDEC4C70CCCE6EED074444
 00000000000000000000000000000000005134507DE42320239FD0B00
_____________________________________________________________
```