CA IDMS™

Mapping Facility Guide

Version 18.5.00

CA Technologies Product References

This document references the following CA products:

- CA IDMS™/DB
- CA ADS™
- CA IDMS™/DC
- DC/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.

Contents

Chapter 1: Introduction

This manual describes the capabilities of the CA IDMS Mapping Facility and serves as a reference tool for the CA IDMS applications developer in designing maps.

It also provides an overview of the interactions between the mapping facility, the Integrated Data Dictionary (IDD), user-written programs, and CA ADS for CA IDMS dialogs. This manual also describes the online and batch methods of map-definition.

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.

─────────────────────►

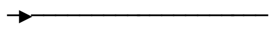
Indicates that the syntax continues on the next line.

►─────────────────────

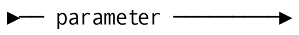
Indicates that the syntax continues on this line.

────────────────────►─

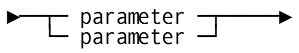
Indicates that the parameter continues on the next line.



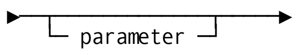
Indicates that a parameter continues on this line.



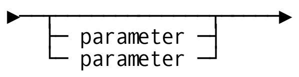
Indicates a required parameter.



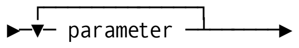
Indicates a choice of required parameters. You must select one.



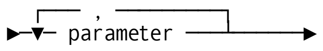
Indicates an optional parameter.



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



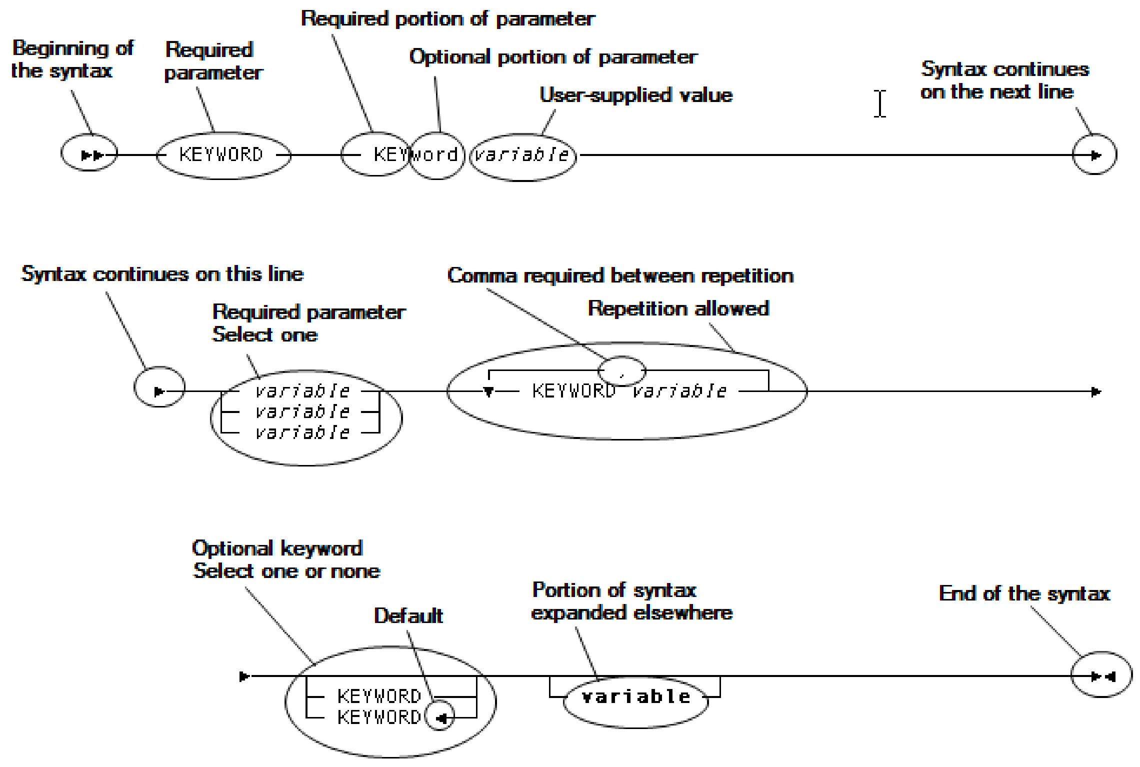
Indicates that you can repeat the parameter or specify more than one parameter.



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:


Beginning of the syntax
Required parameter
Required portion of parameter
Optional portion of parameter
User-supplied value
Syntax continues on the next line
KEYWORD
KEYword
variable
Syntax continues on this line
Required parameter Select one
Comma required between repetition
Repetition allowed
variable
variable
variable
KEYWORD variable
Optional keyword Select one or none
Default
Portion of syntax expanded elsewhere
End of the syntax
KEYWORD
KEYWORD
variable

Chapter 2: Introduction to the Mapping Facility

This section contains the following topics:

Overview

What is the Mapping Facility?

A *map* is a formatted terminal screen used to communicate between an application and a terminal operator. The CA IDMS mapping facility is used to define the layout of maps. The mapping facility simplifies the development, storage, and use of input/output (I/O) displays and fields.

- Development—The mapping facility enables the map developer to:
  - Use commands or screen prompts to simplify the definition of maps
  - Take advantage of consistent procedures to establish fields on a map
  - Predefine the characteristics of the data transmitted between the I/O device and program variable storage
  - Use the mapping facility to specify links between a map and records
  - Generate an automatic screen layout from a record definition
- Storage—The mapping facility handles the storage and retrieval of map-related entity occurrences and map load modules
- Use—The mapping facility creates maps that can be integrated easily with dialogs and programs.
  - Dialogs generated by the application development system, CA ADS, use CA IDMS maps exclusively for screen I/O operations.
  - Programs written in COBOL, PL/I, and Assembler can use maps for I/O transactions when appropriate Data Manipulation Language (DML) statements are included in the code.

Note: References in this manual to user-written programs apply equally to CA ADS dialogs and host language programs unless otherwise noted.

Online and Batch Capabilities

The map developer can use the mapping facility to create and maintain maps in either an online or batch environment as described below:

- The *online mapping compiler* is a convenient online tool for generating and maintaining map-related entity occurrences and map load modules. A map developer using the online compiler does not need to write code or execute JCL routines. The mapping facility is integrated with other CA online systems.
- The *batch compiler and utility* are the batch equivalent to the online compiler. The developer generates and maintains map-related entity occurrences by submitting statements to the batch compiler. Map load modules are generated and maintained by statements that are submitted to the batch utility.

  The online compiler can be used to modify most map-related entity occurrences and load modules created by using the batch compiler and utility. The batch compiler and utility can be used to modify any map entity occurrence or load module generated by the online compiler.

Input/Output Operations

At runtime, stored records that are associated with the map are bound to areas of program variable storage that are defined by the CA ADS dialog or application program that uses the map. Values are transmitted on output and input operations:

- *On output (mapout) operations*, values in record elements used by the map can be transmitted to the map along with any fixed character strings (that is, literal fields) that are defined for the map. The amount of data that is transmitted is determined by the runtime application and by specifications made for the map during map-definition time.
- *On input (mapin) operations*, values that a user supplies and that the map is prepared to receive are transmitted to program variable storage.

Note: For more information about CA IDMS I/O modes and for programming considerations that affect modes of I/O, see the *CA IDMS DML Reference Guide for COBOL*, *CA IDMS DML Reference Guide for Assembler*, or the *CA IDMS DML Reference Guide for PL/I*.

What's in this Chapter?

This introductory section briefly discusses the components used to create and maintain maps:

- The online compiler
- The batch compiler and utility

Additionally, the following runtime features that augment the use of maps at runtime are discussed:

- Automatic editing and error-handling
- Alternative maps

Finally, a discussion of terminals supported by the mapping facility completes this section.

The Online Compiler

How Does it Work?

The online mapping compiler facilitates map creation by supplying a set of screens that prompt a map developer for map and map field specifications. The developer can paint the screen automatically using the Autopaint feature or position the fields manually. All the work is done using a terminal. The developer can modify and redraw the map in one or more sessions until satisfied with the map.

The online compiler uses specifications made during a session to populate the data dictionary with all relevant definitions and information. Many record-keeping functions are performed internally by the online compiler for the map developer.

What Functions can it Perform?

The online compiler can be used to:

- Create map-related entity occurrences in the data dictionary
- Modify or delete map-related entity occurrences that were created either by online or batch mapping
- Copy a map load module
- Generate map load modules
- Delete map load modules and dictionary entity occurrences
- Associate help text with the map

A Sample Session

The following figures illustrate an abbreviated online mapping compiler session. The primary screens used during an online session are shown in this sample session. The names and major functions of each screen are listed:

- The *Main Menu*, which is used to specify basic information about the map, is the first screen in a session. The developer supplies the name and version number of the map and the name and node of the dictionary to be used.

  From this screen, the developer can:

  - *Either* move to other screens to define more specific information about the map such as general options, help text, the layout of the map, field descriptions, or associated records
  - *Or*, select an action to perform on the specified map from the action bar at the top of the screen

  For the purposes of this sample session, we will move through the screens to create and define a map.

- The *General Options screens* are used to define general characteristics of the map such as its title, the screen size, display and print options, attributes for redisplayed fields, as well as to indicate whether automatic editing takes place.
- The *Map-Level Help Text Definition screen* is used to associate help text with a map. The help text itself resides in an IDD module.
- The *Associated Records screen* is used to specify the records that contain the elements that will populate the map.

  At this point, the developer can choose to create the map automatically using the *Autopaint* option or go to the Layout screen.

- The *Layout screen* is used to layout the map. If the Autopaint option was chosen, the screen initially displays the layout created by the automatic map painter. Otherwise, the developer can begin to layout the map as desired.
- The *Field Definition screens* are used to define the specific information about the field.

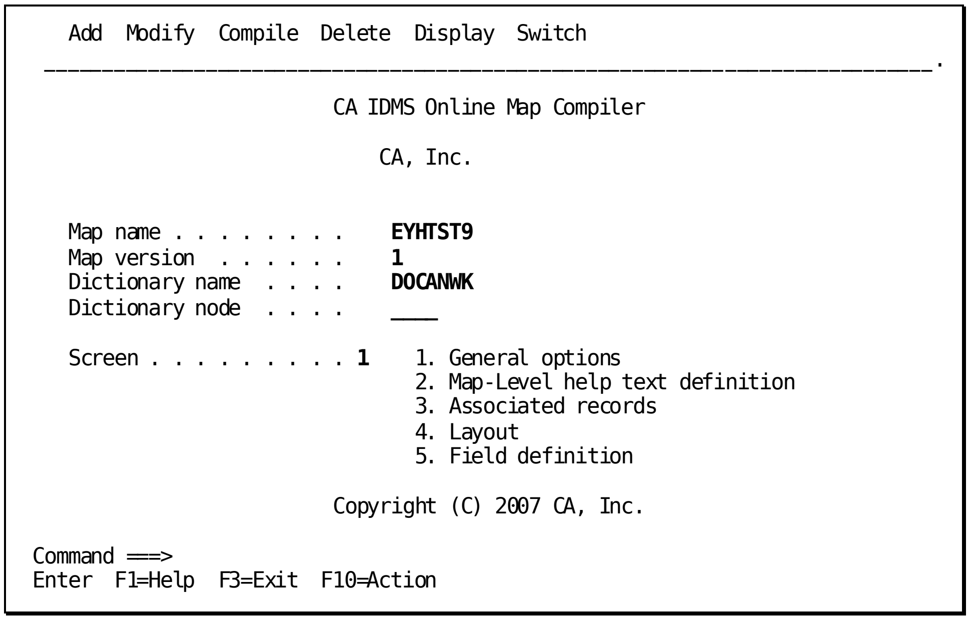
  The seven screens used to expand on the field definition are:

  - Field Definition
  - Map Read/Write Options
  - Additional Edit Criteria
  - Field-Level Help Text definition
  - Device-Dependent Options
  - User-Defined Edit Modules
  - Pageable Options

For the Sample Session

- Default values for specifications are used in the sample session unless otherwise indicated. For more information on the screens, see "Online Mapping Compiler Reference".
- To move from option to option, <F5> is pushed; to move from screen to screen within a given option, <F8> is pushed.

Specifying Basic Information

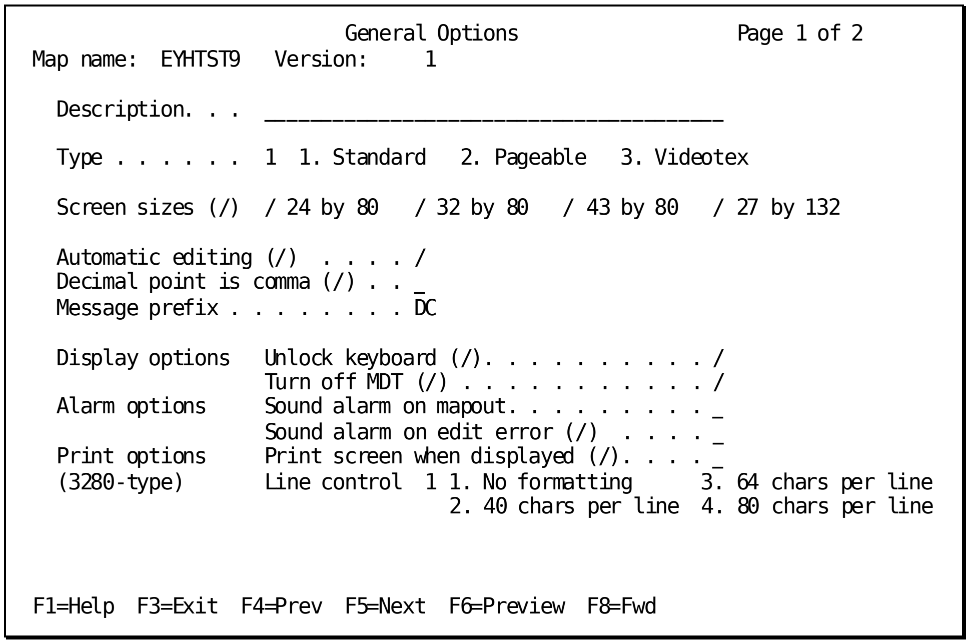


```
   Add  Modify  Compile  Delete  Display  Switch
 ____________________________________________________________________________.

                          CA IDMS Online Map Compiler

                              CA, Inc.


   Map name . . . . . . . .    EYHTST9
   Map version  . . . . . .    1
   Dictionary name  . . . .    DOCANWK
   Dictionary node  . . . .    ____

   Screen . . . . . . . . . 1    1. General options
                                 2. Map-Level help text definition
                                 3. Associated records
                                 4. Layout
                                 5. Field definition

                          Copyright (C) 2007 CA, Inc.

Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

What is Entered?

The developer enters the name and version number of the map as well as the dictionary name and node. The developer also enters a 1 to indicate that options are specified.

Specifying General Options



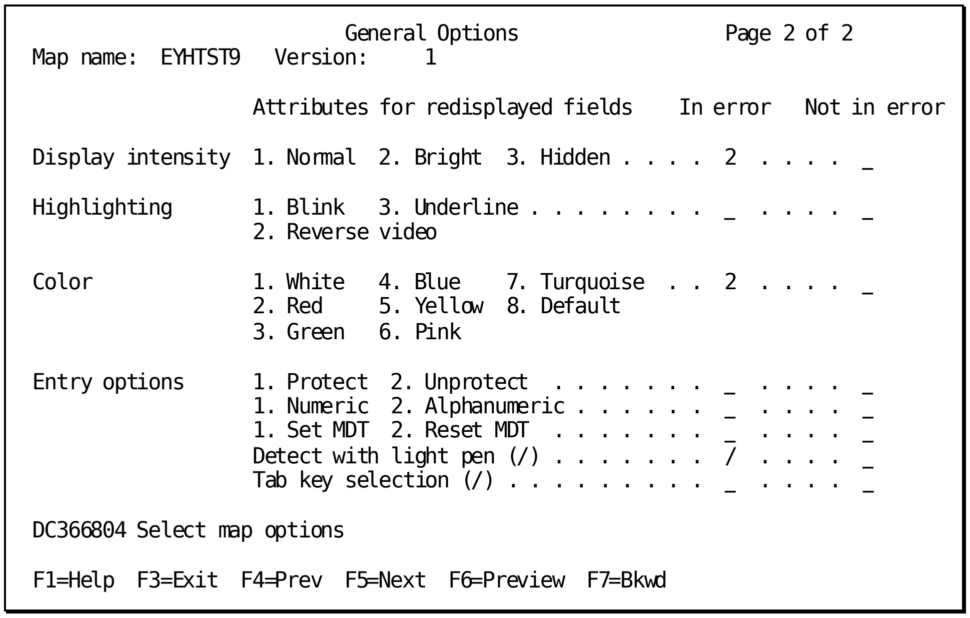
```
                           General Options                   Page 1 of 2
Map name:  EYHTST9   Version:     1

  Description. . .  ______________________________________

  Type . . . . . .  1  1. Standard   2. Pageable   3. Videotex

  Screen sizes (/)  / 24 by 80   / 32 by 80   / 43 by 80   / 27 by 132

  Automatic editing (/)  . . . . /
  Decimal point is comma (/) . . _
  Message prefix . . . . . . . . DC

  Display options   Unlock keyboard (/). . . . . . . . . . /
                    Turn off MDT (/) . . . . . . . . . . . /
  Alarm options     Sound alarm on mapout. . . . . . . . . _
                    Sound alarm on edit error (/)  . . . . _
  Print options     Print screen when displayed (/). . . . _
  (3280-type)       Line control  1 1. No formatting      3. 64 chars per line
                                    2. 40 chars per line  4. 80 chars per line



F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F8=Fwd
```

What is Entered?

The general options associated with the map are entered using the General Options screen. The General Options screen has two pages.

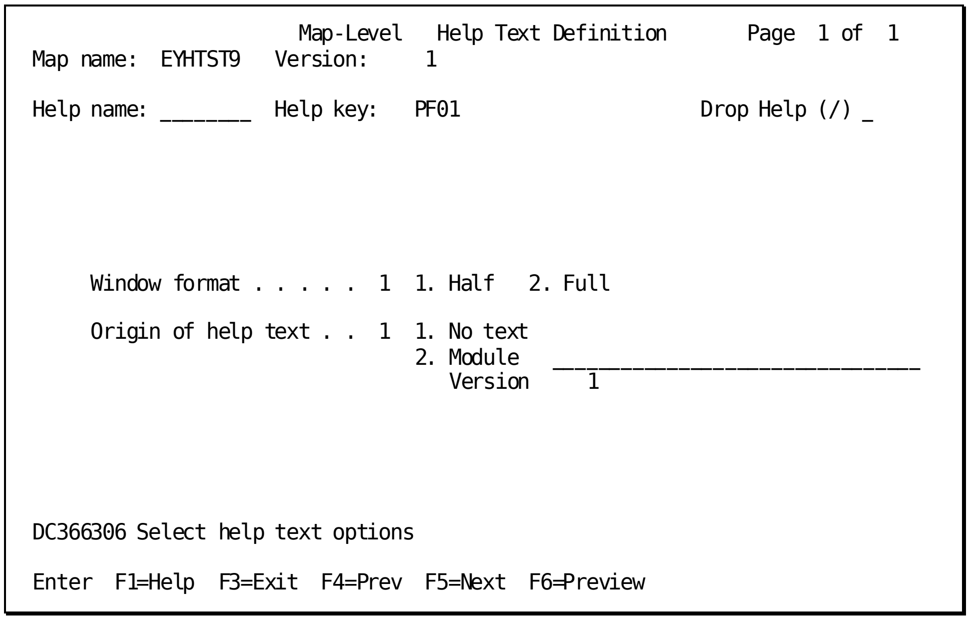
The first page is shown previously and the second page is shown. To move between the two, use <F7> and <F8>.

General Options Screen - Page 2



```
                         General Options                  Page 2 of 2
Map name:  EYHTST9   Version:     1

                 Attributes for redisplayed fields    In error   Not in error

Display intensity  1. Normal  2. Bright  3. Hidden . . . .  2  . . . .  _

Highlighting       1. Blink   3. Underline . . . . . . . .  _  . . . .  _
                   2. Reverse video

Color              1. White   4. Blue    7. Turquoise  . .  2  . . . .  _
                   2. Red     5. Yellow  8. Default
                   3. Green   6. Pink

Entry options      1. Protect  2. Unprotect  . . . . . . .  _  . . . .  _
                   1. Numeric  2. Alphanumeric . . . . . .  _  . . . .  _
                   1. Set MDT  2. Reset MDT  . . . . . . .  _  . . . .  _
                   Detect with light pen (/) . . . . . . .  /  . . . .  _
                   Tab key selection (/) . . . . . . . . .  _  . . . .  _

DC366804 Select map options

F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F7=Bkwd
```

Defining Map-level Help

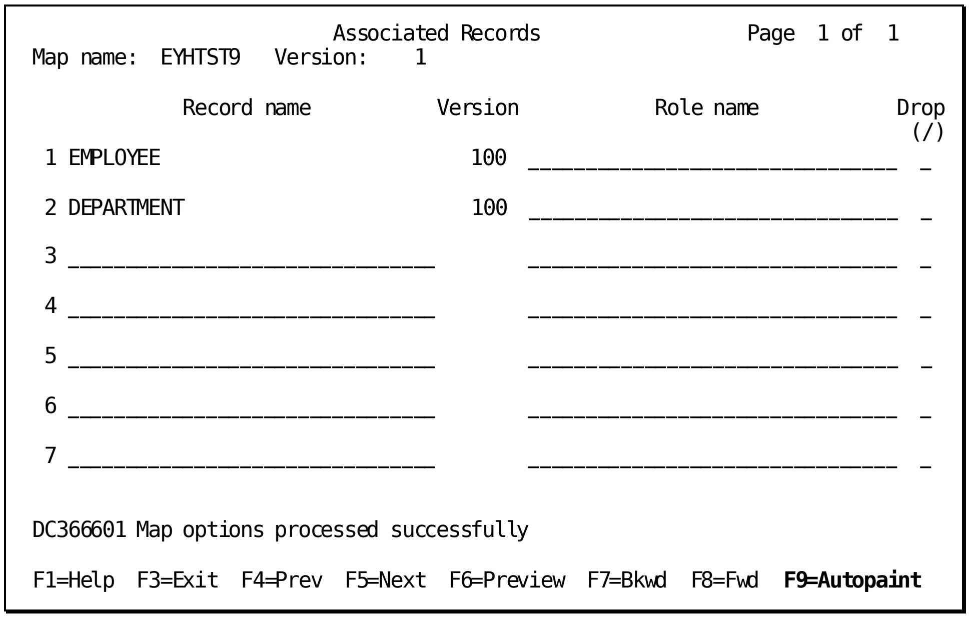

```
                        Map-Level   Help Text Definition       Page  1 of  1
Map name:  EYHTST9   Version:     1

Help name: ________  Help key:   PF01                   Drop Help (/) _





     Window format . . . . .  1  1. Half   2. Full

     Origin of help text . .  1  1. No text
                                 2. Module   ________________________________
                                    Version     1




DC366306 Select help text options

Enter  F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview
```

What is Entered?

To associate help with a map, the developer specifies:

- The name of the load module that contains the help information in the Help name field
- The function key that invokes map-level help in the Help key field
- Whether the help is displayed in a half or full window in the window format field
- The name of the IDD module that contains the help text for this map in the Origin of help text field

Specifying the Associated Records

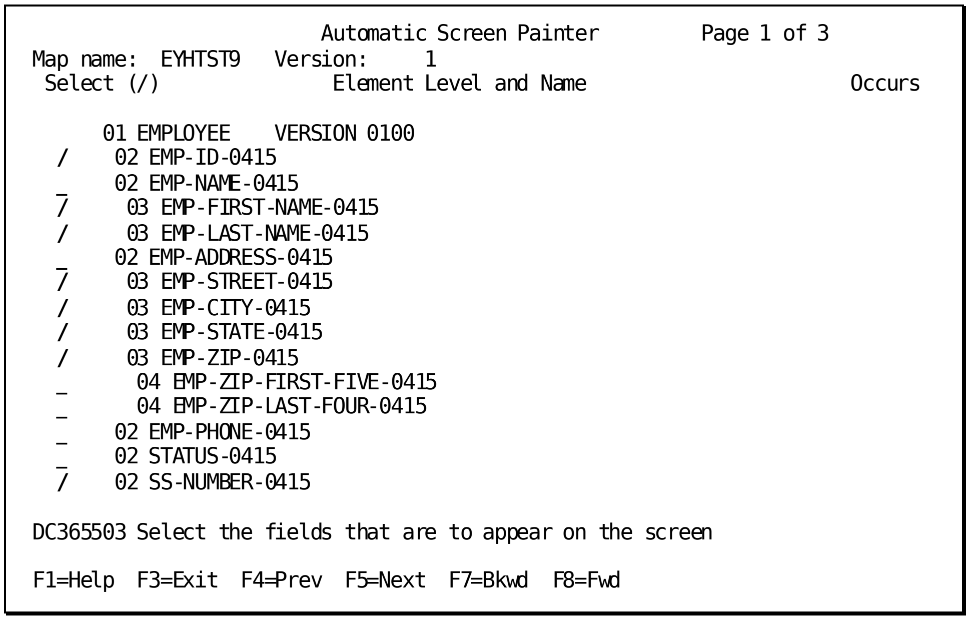


```
                          Associated Records                Page  1 of  1
 Map name:  EYHTST9   Version:    1

            Record name           Version          Role name           Drop
                                                                        (/)
  1 EMPLOYEE                        100  ________________________________  _

  2 DEPARTMENT                      100  ________________________________  _

  3 ________________________________     ________________________________  _

  4 ________________________________     ________________________________  _

  5 ________________________________     ________________________________  _

  6 ________________________________     ________________________________  _

  7 ________________________________     ________________________________  _


 DC366601 Map options processed successfully

 F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F7=Bkwd  F8=Fwd  F9=Autopaint
```

What is Entered?

The developer enters the names and version numbers of the records associated with the map.

Creating a Map Automatically

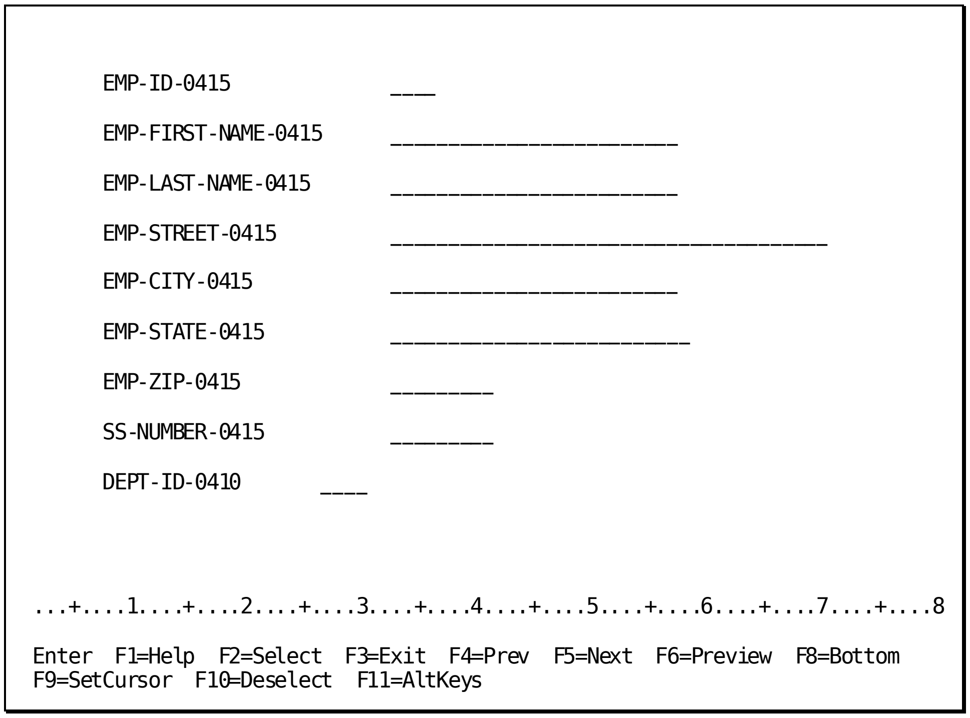


```
                          Automatic Screen Painter        Page 1 of 3
Map name:  EYHTST9   Version:     1
 Select (/)                Element Level and Name                      Occurs

      01 EMPLOYEE    VERSION 0100
  /    02 EMP-ID-0415
  _    02 EMP-NAME-0415
  /     03 EMP-FIRST-NAME-0415
  /     03 EMP-LAST-NAME-0415
  _    02 EMP-ADDRESS-0415
  /     03 EMP-STREET-0415
  /     03 EMP-CITY-0415
  /     03 EMP-STATE-0415
  /     03 EMP-ZIP-0415
  _      04 EMP-ZIP-FIRST-FIVE-0415
  _      04 EMP-ZIP-LAST-FOUR-0415
  _    02 EMP-PHONE-0415
  _    02 STATUS-0415
  /    02 SS-NUMBER-0415

DC365503 Select the fields that are to appear on the screen

F1=Help  F3=Exit  F4=Prev  F5=Next  F7=Bkwd  F8=Fwd
```

What is Entered?

The developer selects the fields (/) that will be displayed on the map. Often, the list of elements is displayed on more than one screen as illustrated here.

Previewing the Map



```
     EMP-ID-0415                 ____
     EMP-FIRST-NAME-0415         _________________________
     EMP-LAST-NAME-0415          _________________________
     EMP-STREET-0415             ______________________________________
     EMP-CITY-0415               _________________________
     EMP-STATE-0415              __________________________
     EMP-ZIP-0415                _________
     SS-NUMBER-0415              _________
     DEPT-ID-0410          ____


...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8

Enter  F1=Help  F2=Select  F3=Exit  F4=Prev  F5=Next  F6=Preview  F8=Bottom
F9=SetCursor  F10=Deselect  F11=AltKeys
```

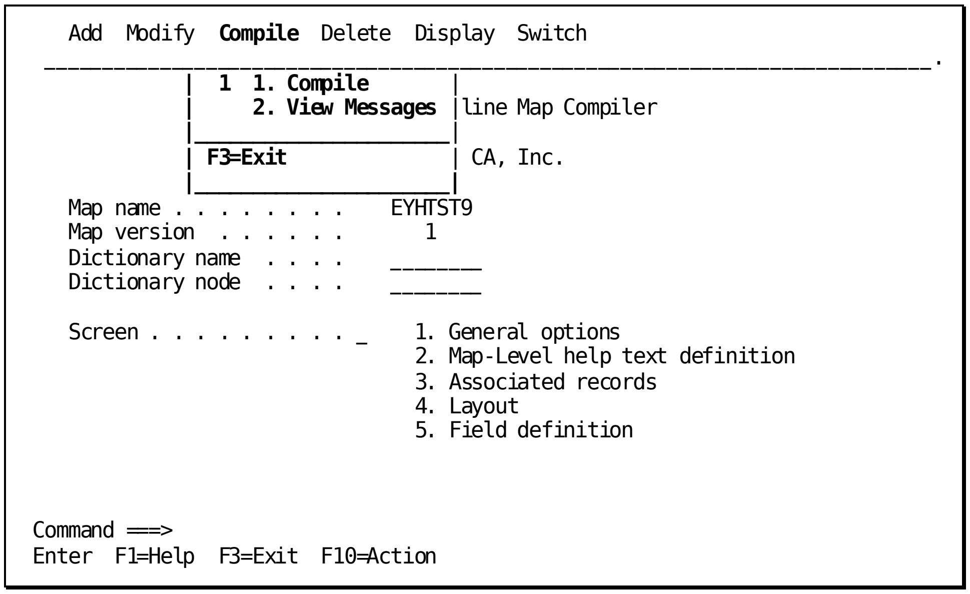
What is Entered?

The online compiler uses the fields the developer selected to create a map which can be previewed.

Modifying the Map

The developer can then add, modify, or delete the fields as appropriate.

Compiling the Map


```
   Add  Modify  Compile  Delete  Display  Switch
 ____________________________________________________________________________.
             |  1  1. Compile        |
             |     2. View Messages  |line Map Compiler
             |_______________________|
             | F3=Exit               | CA, Inc.
             |_______________________|
   Map name . . . . . . . .    EYHTST9
   Map version  . . . . . .       1
   Dictionary name  . . . .    ________
   Dictionary node  . . . .    ________

   Screen . . . . . . . . . _    1. General options
                                 2. Map-Level help text definition
                                 3. Associated records
                                 4. Layout
                                 5. Field definition



Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

What is Entered?

To compile the map, the developer returns to the main menu and selects the Compile option from the action bar.

The Batch Compiler and Utility

The batch compiler and utility can be used to perform all of the operations that are available through the online compiler:

Batch Compiler Functions

- *Creates, modifies, or deletes entity occurrences for panels* in the data dictionary; panel occurrences predefine screen layouts for maps
- *Creates, modifies, or deletes entity occurrences for maps* in the data dictionary based on existing panel occurrences and on specifications made by using the CA IDMS mapping language

Batch Utility Functions

- *Generates or deletes map load modules* in the load area of the data dictionary
- *Produces map and panel reports for any map* generated either by the mapping facility
- *Produces a facsimile of a map or panel format* on hard copy from the map or panel definition
- *Decompiles maps* generated by either the batch or online compiler

Performing Online Compiler Operations

A map developer can use the batch compiler and utility to perform all of the operations available through the online compiler. Map developers often prefer to use the batch compiler and utility to perform the following operations:

- *Modify or copy maps*—Developers who are more familiar with or prefer syntax can quickly decompile a map into source, alter that source, and then submit the altered source for compilation.
- *Create several similar maps in one session*—The developer can copy the source for one map several times, alter the various copies of that source, and then submit all of the altered map-definitions for compilation in one JCL operation.

Performing Non-online Compiler Operations

Additionally, the batch compiler and utility can be used to perform the following operations that the online compiler does not provide:

- *Migrate a map from one dictionary to another*—The developer uses the batch utility to decompile a map and then uses the batch compiler to recompile the syntax to another dictionary.
- *Decompile entity occurrences that store the current map-definition*—The developer uses the batch utility to produce the source definition for a map from the related entity occurrences for the map.
- *Produce reports*—The developer receives reports on batch compiler and utility operations.
- *Create maps with specific support for multiple devices* —The developer uses the FOR *device-code-a* option of the MAP1 MFLD and PFLD clauses to create the maps.
- *Create maps for devices larger than the one on which a developer is working.*

Automatic Editing and Error Handling

Purpose

The CA IDMS mapping facility provides automatic editing and error-handling capabilities for use by CA IDMS maps. Automatic editing and error-handling simplify data validation, I/O data conversions, and redisplay of input errors when a map is displayed by a dialog or program.

Automatic editing and error-handling for maps and fields can be enabled using either the batch or online compiler as described:

- *Automatic editing* simplifies data validation and I/O data conversions:
  - *Data validation is performed on input*. For example, automatic editing could be used to validate that a terminal operator has supplied a valid state name (such as Alabama) in the STATE field on a map that displays employee address data.
  - *Data conversions are performed on input and output*. For example, automatic editing could be used to display the value Alabama when 01 is stored for the STATE field.
- *Error handling* simplifies the redisplay of input errors. Error handling, when enabled for a map, prepares a map to be redisplayed when an input error is detected. Input errors can be highlighted for the operator's attention, and error messages can be displayed.

Example

The following figure illustrates operations performed by automatic editing and error-handling at map runtime. The following occurs when the operator keys data in fields on the sample HILT TECHNOLOGIES, INC. screen and presses a control key:

- *Automatic editing evaluates the data* supplied by the operator, using automatic editing criteria specified for each field.

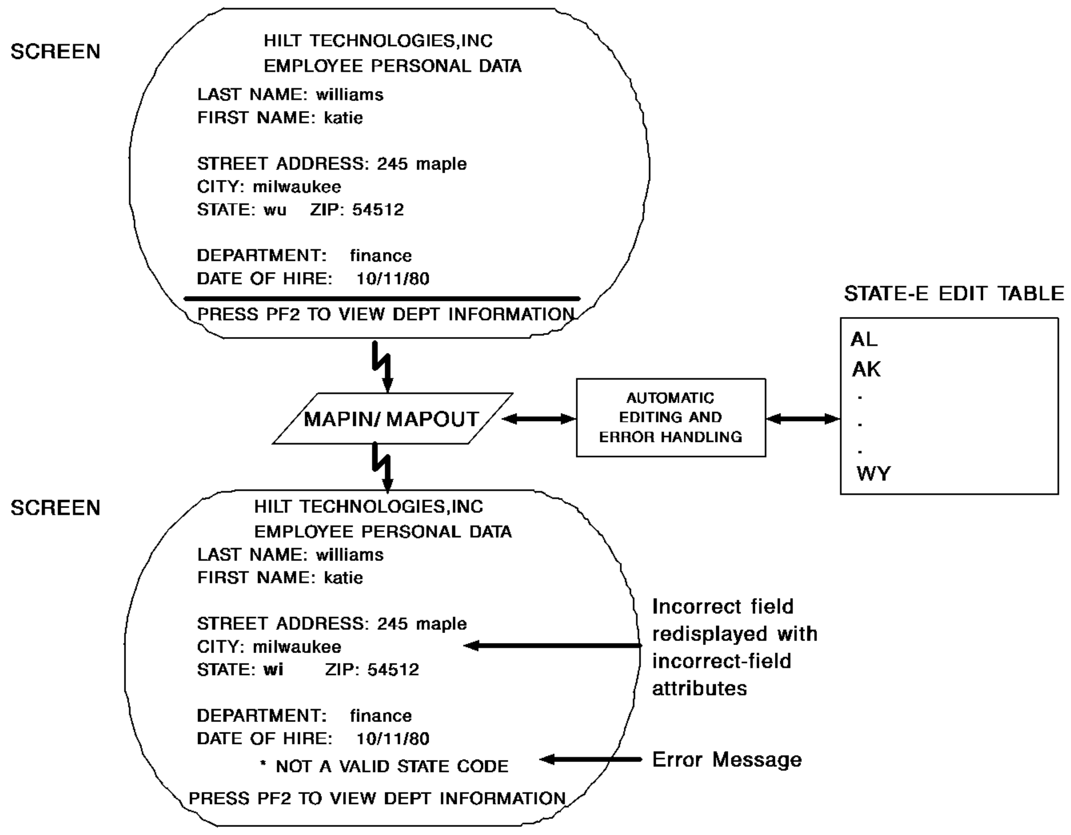
  The STATE-E edit table that is defined for the sample STATE field in the following figure lists valid two-character state abbreviations for the STATE field. The values in the STATE-E table are used to determine that WU is not a valid value for the STATE field.

- *The CA ADS dialog or application program redisplays the map* using error-handling attributes. Error handling redisplays the sample map with the following changes:

  – The incorrect data in the STATE data field is highlighted in BRIGHT.

  – The error message NOT A VALID STATE CODE is displayed in the message field for the map.

Note: For more information about automatic editing and error-handling, see the chapter "Automatic Editing and Error Handling".



Alternative Maps

Purpose

The mapping facility allows the use of alternative maps. This feature is useful in any application in which a dialog or program should show different copies of the same map to different users. When the alternative map feature is used, each user sees the appropriate copy of a given map.

Generating Alternative Maps

The developer can use the online mapping facility to generate an alternative copy of a map.

Note: For more information on defining and using alternative maps, see the appendix "Alternative Maps."

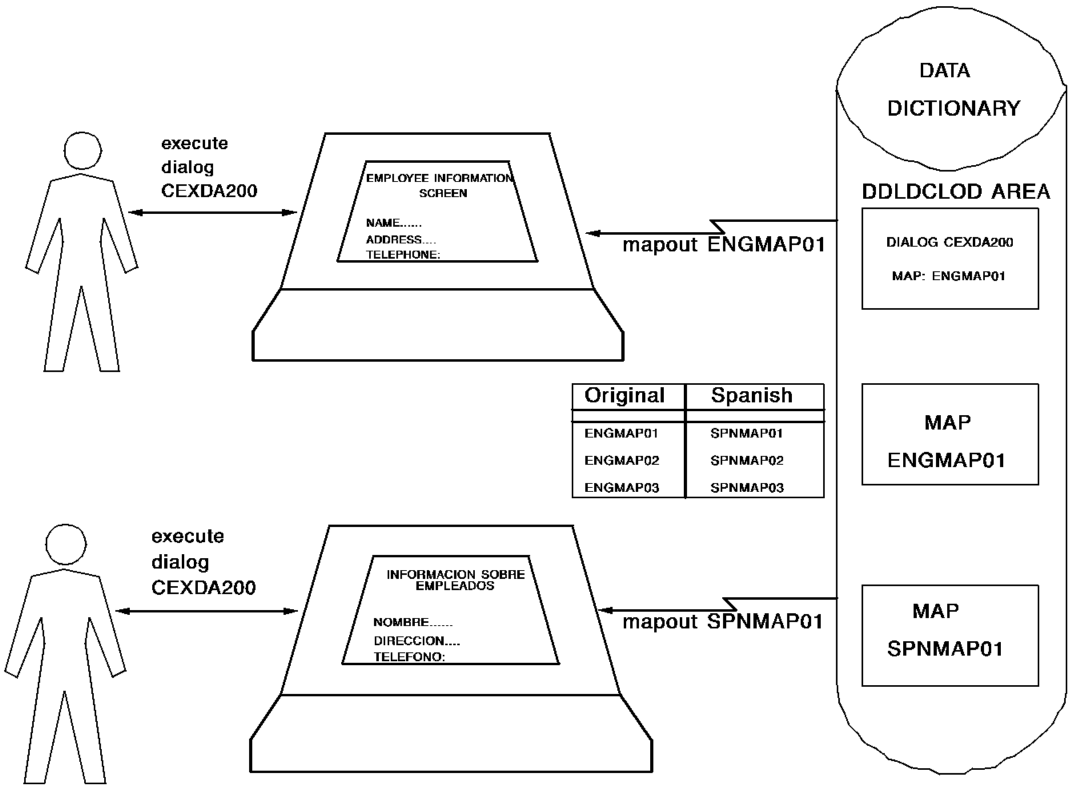
Possible Applications

- Users who speak different languages can be shown versions of maps in their own languages.

  For example, the following figure shows an English-language map and its Spanish-language alternative. The name of the English-language map, ENGMAP01, is changed to SPNMAP01; the title of the map is changed from EMPLOYEE INFORMATION SCREEN to INFORMACION SOBRE EMPLEADOS.

- Users with different levels of expertise or authority can be shown versions of maps that display only the information those users are qualified to view.

  For example, users who are not authorized to see salary information would not be shown the SALARY field on a map; a version that shows the SALARY field would be displayed to authorized users only.



Original	Spanish
ENGMAP01	SPNMAP01
ENGMAP02	SPNMAP02
ENGMAP03	SPNMAP03

Terminals Supported by the Mapping Facility

The following terminals are supported by the CA IDMS mapping facility:

Terminal	Mapping Consideration
3270-type	Runtime mapping uses attribute bytes to establish individual fields on the screen. Fields are displayed on the screen with the attributes available on the terminal. Attributes such as color and underscoring are not available on a 3270-type terminal.
3279-type	Runtime mapping uses attribute bytes to establish fields on the screen. Fields are displayed on the screen with the attributes available on the terminal. Display attributes such as color and underscoring are available on 3279-type terminals.
Glass TTY-type	The screen is processed as a single wraparound data line; however, runtime mapping displays data so that it appears to the operator that fields are posted to the map as on 3270-type terminals. Fields are displayed according to the screen's default display options; most 3270- and 3279-type display attributes are unavailable.

Note: For more information about attributes and attribute bytes, see "Attributes for Fields".

Chapter 3: Map Design Considerations

This chapter discusses about the map design considerations.

This section contains the following topics:

Overview

Most applications use more than one map to supply information to and request information from the user. A well-planned application uses maps that make sense to terminal operators and that take advantage of features provided by the mapping facility.

Map developers should consider the following topics, which are described, when creating maps:

- Preliminary information gathering
- Designing maps
- Designing map fields

Preliminary Information Gathering

Before beginning work on maps, the map developer should be familiar with the application for which maps are needed and with the site at which the maps are used. Examples of application- and site-specific information are presented.

Application-specific Information

To generate efficient maps, the developer should be aware of the following information about the application:

- The *amount and type of information* the entire application will request through mapped displays. Applications generally require more than one map to request and/or present information efficiently.
- The *records needed* by each map and any attendant information:
  - The *name of each record element* within the record that the map displays or updates as data fields.
  - The *external picture* (if any) defined for each record element. The length of a data field is determined by the external picture of the associated element.
  - The *name of the edit table* (if any) stored in the data dictionary for each record element and the values/ranges that the table defines as either valid or invalid.
  - The *name of the code table* (if any) stored in the data dictionary for each record element and the conversions that the table performs.

  Note: External pictures, edit tables, and code tables are discussed in "Automatic Editing and Error Handling".

The developer can determine how many maps need to be designed and can begin to place fields on each map based on the information summarized previously.

Site-specific Information

The map developer should also be familiar with the resources that are available at the site.

Existing Site Information

- *Names and version numbers of any existing maps* that could be adapted easily for the application being developed
- *Name of the dictionary* used for map storage
- *Name of the dictionary node*

Existing Conventions and Standards

Many sites define standards that establish guidelines for map developers. These standards can address the following topics:

- *Naming conventions* for map-related entities, as presented in Naming Conventions (see page 41)
- *Map layout standards,* as discussed in chapter "Designing Map Fields".

Different Types of Terminals

For sites at which operators use more than one type of terminal, the developer should be familiar with the types of terminals that are used when the maps are displayed to operators. The map developer should take into account the following considerations:

- Terminal display size differs from terminal to terminal; a map designed for display on more than one type of terminal should be no wider than the width of the narrowest terminal.
- Certain attributes available on 3279-type terminals, such as colors and underscoring, are not available on 3270-type or glass TTY terminals. Certain 3270-type attributes, such as the ability to display in bright mode or to suppress numeric input, are not available on glass TTY terminals.
- Names of control keys (for example, PA1 or PF1) can differ from terminal to terminal. Screens that name control keys should use names that the operator will encounter on all terminals that can display the map.

Online Compiler Options

Finally, a map developer planning to use the online mapping compiler to generate maps should contact the DBA for the online mapping compiler sysgen options in effect at the site.

Note: For more information about available options, see the *CA IDMS System Generation Guide*.

Designing Maps

Successful Layouts

The layout of a map should make the map easy to use. A successful map layout exhibits the following characteristics:

- **Consistency**—Entities such as fields, headings, labels, responses, messages, and control keys should have the same meaning or effect throughout the application. The meaning or effect need not be identical for every map but should be consistent within the broader confines of the system.

  Message and response fields should appear in the same location on each map in an application. These fields should remain standard for all applications at a site.
- **Supportiveness**—The reactions of the system should make it easy for the operator to handle normal situations. For example, displayed informational and/or error messages should be meaningful.

What's Included in this Section

These standards, which are discussed separately, can be adapted as necessary to conform to the needs of a particular site or application:

- General considerations
- Control key standards
- Naming conventions
- Layout and display standards
- Pageable map considerations

General Considerations

The following general considerations promote the design of efficient maps.

- **Consistency**—Spelling and abbreviation of terms should be consistent.
- **Clarity**—Messages and prompts should be clear and understandable to users.
- **Control keys**—Terminal operators should be able to initiate processing by providing the requested data and then pressing a control key.
- **Program logic**—Operators should not be required to make decisions that could be incorporated into program logic.
- **Automatic editing**—Features provided through automatic editing and error-handling should be used whenever possible. Correct and incorrect fields should be redisplayed consistently. For example, a particular color or display intensity can be used systematically to draw the operator's attention to fields that contain input errors.
- **Standards for pageable maps**—At sites that use pageable maps, map layout standards should be developed that apply equally well to pageable and non-pageable maps.

Note: For more information about pageable maps, see the chapter "Pageable Map Considerations”.

Control Key Standards

Identify Control Keys

The CA ADS dialog or application program that uses a map defines control keys for use with the map. If possible, the control keys available to a map operator should be documented by literal fields on a map. The operations performed by the control keys should also be documented for the operator.

When different terminals are available to terminal operators and control key names differ from keyboard to keyboard, each control key name should be documented so that operators do not have to memorize lists of key equivalencies.

Use Control Keys Consistently

Control keys should also be used consistently. For example, the same key should be used to take the operator to the next map in each application. A sample standard for control key usage is presented in the following table.

Sample PF Key Standards

PF Key	Function
PF1	Help
PF3	Exit
PF7	Display previous page
PF8	Display next page
PF12	Print

Naming Conventions

Why use Conventions?

It is advisable for a site to develop naming conventions for map-related entity occurrences. While mnemonic names work well at sites that have a few basic applications, mnemonic names become difficult to use as the number and complexity of applications increase.

Adhering to naming conventions makes it easier to construct names, easier to reconstruct names if one is forgotten, and easier for different users to determine the purpose of a map or map component. Naming conventions facilitate the construction of alternative map tables when alternative maps are required.

Sample Naming Conventions

The following naming conventions illustrate a sample system for giving names to map-related entities. PANEL and TABLE occurrence names can be from 1 through 32 characters in length. Names of MAP occurrences can be from 1 through 8 characters in length. The sample naming convention specifies 8-character names since it is intended for MAP occurrences as well as for PANEL and TABLE occurrences.

Notes:

- For more information on PANEL occurrences, see the chapter "Panels and Maps".
- For more information on TABLE occurrences, see appendix, “Generating Edit and Code Tables."

Position	Meaning	Value	Description
1	Source of entity	C	CA entity
2-3	Application	EX	Example application
		FS	Financial system application
		SY	System application
4	Type of entity	L	Panel occurrence
		M	Map occurrence
		C	Table occurrence (code table)
		E	Table occurrence (edit table)
5	Special information	E	English-language application
		F	French-language application
		G	German-language application
		S	Spanish-language application
6-8	Identification of entity	xxx	

Sample Names

The following sample names demonstrate the naming conventions described in the previous table:

- **CEXME104** names a CA map that is used in a sample application. Literals are in English. The number 104 identifies this map.
- **CEXMG104** names a copy of map CEXME104 that contains German literals. Map CEXMG104 is an alternative for map CEXME104 and is used instead of CEXME104 when alternative maps are supported at the site and an operator with the GERMAN user type uses the map.

- **CEXLE221** names a CA panel that is used in a sample application. Literals are in English. The number 221 identifies the panel.
- **CEXCFWRB** names a CA code table that is used in a sample application. The decoded values in the code table are in French. The letters WRB identify the code table.

It is often useful to integrate map-naming conventions for map-related entities with conventions used for dialogs and programs.

Layout and Display Standards

Layout and display standards promote the creation of consistent maps. The following considerations should be kept in mind when designing map layout and design standards.

Associate Areas with Particular Uses

Each specific area of a map should be devoted to a particular use. For example, the top five lines of the screen could be reserved for product and title information and the bottom five for map and control key information.

Display like Fields in the Same Location

Similar fields should be displayed in the same location and with consistent display attributes on all maps. A given site usually uses the same or similar fields on several maps. For example, message fields and titles are included on most maps, and should be presented consistently.

Limit the Number of Fields on a Screen

The amount of data transmitted down a line is affected by the number of fields on a map. This can affect line contention and should be considered when designing a map. Do not do borders or one-byte literal fields.

Handle Operations and Prompts Consistently

Frequently used operations or prompts should be handled consistently. For example, yes and no responses should be requested in the same way on every map.

Handle Default Values Consistently

Consistent methods should be developed for indicating default values. For example, an asterisk (*) or a special display color might be used to identify the default value for a list of options. A default entry for a data field might be specified by a literal field adjacent to the data field.

Standardize Map Templates

A standard map template should be used for each type of map. For example, many applications prompt the operator to select an item from a menu. A template for this type of screen is different than a template for a screen that requests words and numbers from a user. The following considerations apply to the design of a template:

- A map becomes cluttered when it contains too many map fields
- Fields are easier to read when double spaced
- Data fields are more visible when all data fields for the map begin in the same column
- Use of bright or attention-drawing fields increases the map's effectiveness
- The cursor should be in the position most likely to be used for data entry when the map is first displayed at a terminal
- When using the TAB key to move to the next field, the sequence of fields should match the most common or logical pattern used for data entry

Pageable Map Considerations

The CA IDMS mapping facility provides a format for pageable maps. The format for pageable maps should be consistent with the formats of other maps designed at a site, making pageable maps easier for operators to use.

Sample Pageable Map

```


EMPLOYEE PERSONAL DATA


EMPLOYEE NAME: JANE       FERNDALE
EMPLOYEE NUMBER: 0032         SOCIAL SECURITY NUMBER: 034-56-7890
EMPLOYEE NAME: TOM        FITZHUGH
EMPLOYEE NUMBER: 0081         SOCIAL SECURITY NUMBER: 112-34-5678
EMPLOYEE NAME: GEORGE      FONRAD
EMPLOYEE NUMBER: 0045         SOCIAL SECURITY NUMBER: 092-34-7890
EMPLOYEE NAME: ROBIN       GARDNER
EMPLOYEE NUMBER: 0053         SOCIAL SECURITY NUMBER: 022-34-4444
EMPLOYEE NAME: JENNIFER     GARFIELD
EMPLOYEE NUMBER: 0003         SOCIAL SECURITY NUMBER: 021-99-4516




PAGE: 0004
```

Components of Pageable Maps

- The *header area* (optional) is located across the top of the screen.
- The *footer area* (optional) is located across the bottom of the screen.
- The *detail area* (required) is located across the middle of the screen. The fields that are defined in the detail area make up the *detail occurrence* for the map.

The header and footer areas provide a frame for the detail area.

Note: For more information about the areas and detail occurrences for pageable maps, see Pageable Maps (see page 91).

Considerations

- The *size of header and footer areas* for pageable maps should be specified.
- The *header area* for a pageable map should contain the same type of information as header areas for other pageable maps. For example, fields that identify the purpose of a map (such as the title) are typically placed in the header area of a pageable map.
- The *footer area* for a pageable map should be consistent with footer areas for other pageable maps. For example, fields that document control keys are often placed in the footer area.

  Note: At runtime, the footer area floats up to just below the detail area. Therefore, it may not appear in the exact location where it was defined originally.

- The *layout of fields* in each area of a pageable map is affected by the following considerations:
  - A *message field* should be located as follows:
    - *In the header or footer area*, if the most important messages for the map are generated by the ADS DISPLAY MESSAGE command and/or by automatic editing in an error cycle
    - *In the detail area*, if messages are not generated by the ADS DISPLAY MESSAGE command or by automatic editing in an error cycle

    At runtime, the single message field that is defined as part of the detail occurrence is mapped out once in each occurrence in the detail area.
  - A *page field* can be included in either the header or footer area for a pageable map.
  - *Fields in the detail area* (that is, the detail occurrence) repeat at runtime as many times as is necessary to display the data retrieved by the map. A detail area that takes up five lines cannot be displayed as many times on the screen as a detail area that takes up two lines.

    Additionally, the amount of storage specified for runtime pageable map sessions is influenced by the number of lines specified for the detail occurrence of a pageable map.

    Note: For information about pageable map storage, see Estimating Pageable Map Storage (see page 353).

Designing Map Fields

What is a Map Field?

A *map field* is an area on a map that is used to communicate with the terminal operator. For example, a field might be used to collect information from an operator or to provide instructions about the map.

Most maps contain several map fields. When designing map fields, the following topics, which are discussed, should be considered:

- Types of fields
- Attributes for fields

Types of Fields

Each map field must be designated as either a literal, data, message, page, or response field. The field designation determines the functions that a field can perform:

Type of Field	Function
Literal	Displays a predefined literal string that provides a title, prompt, or other information to the terminal operator
Data field	Displays the value (if any) of the record element associated with the field and optionally allows the operator to input data.
Message field	Displays messages generated by an application program or by the automatic error-handling facility. Error messages and automatic error-handling are discussed in "Automatic Editing and Error Handling". A map message field is associated with the system $MESSAGE field.
Page field	Pageable maps only—displays the current page number and permits the operator to request the next page to be displayed. A page field on a pageable map is associated with the system $PAGE field.
Response field	CA ADS only—allows the operator to select a CA ADS response process. A map response field is associated with the system $RESPONSE field.

Required Fields

When defining a data field, the developer specifies whether the field is a *required field*. Operators must supply input in data fields designated as required. Failing to enter data in a required field constitutes an input error.

Attributes for Fields

What is an Attribute?

An *attribute* is a characteristic of a map field provided by the terminal. Different characteristics can be assigned to fields on a map. For example, the BRIGHT attribute is assigned to fields that should be displayed at an intensity that is brighter than normal. The BLINK attribute is assigned to fields that should blink at runtime.

When are Attributes Specified?

The General Options screen is used to assign attributes to map fields that are applied during error cycles in the runtime system.

All other attributes are assigned to the fields using the field definition screens. If the developer does not specify attributes for a field, default attributes are used for the field.

The field attributes defined using the online compilers can be overridden by the program and dialog processes that use the map.

Note: For more information about commands that modify map attributes, see the *CA ADS Reference Guide* or the *CA IDMS Navigational DML Programming Guide*.

When do Attributes take Effect?

The attributes for a field take effect when the field is mapped out to a screen at runtime. An *attribute byte* is a single-character, nondisplayable byte positioned at runtime at the coordinate immediately preceding a map field. Runtime mapping uses information contained within the attribute byte to determine the appearance and characteristics of the field.

Attributes on 3270s and 3279s

Attributes provided by 3270- and 3279-type terminals are listed in the following table:

Attribute	Description
ALPHANUMERIC/ NUMERIC	■ An ALPHANUMERIC field can contain any characters. ■ A NUMERIC field can contain periods and minus signs, as well as numbers in the range 0-9.
PROTECTED/ UNPROTECTED	■ A PROTECTED field does not accept data from a terminal operator. ■ An UNPROTECTED field accepts data from the operator.

Attribute	Description
SKIP/NOSKIP	■ SKIP specifies that the operator cannot tab to the given field. ■ NOSKIP specifies that the operator can tab to the start of the field; the field must be UNPROTECTED.
DETECTABLE/ NONDETECTABLE	■ DETECTABLE specifies that the field is selector-pen (light-pen) detectable. ■ NONDETECTABLE specifies that the field is not detectable with a selector-pen.
DISPLAY/BRIGHT/ DARK	■ DISPLAY specifies that the contents of a field appear on the screen at normal intensity. ■ BRIGHT specifies that the contents of a field appear at high intensity. ■ DARK specifies that the contents of a field are not visible on the screen at runtime.
MDT/NOMDT	■ MDT specifies that a field is marked as modified (the modified data tag is set *on*), whether or not a terminal operator enters data in it. ■ NOMDT specifies that a field is marked as modified only if an operator enters data in it. On a mapin operation, only those fields with MDT set on are automatically moved into program variable storage.
DELIMIT/ NODELIMIT	■ DELIMIT inhibits entry of data that contains more characters than specified by the external picture for the field. ■ NODELIMIT does not inhibit entry of excess characters. The operator can enter characters in a field up to the space before the next map field.
SKIP DELIMIT	At runtime, when the operator enters data in the last character position of the field assigned the SKIP DELIMIT attribute, SKIP DELIMIT causes the cursor to tab automatically to the next UNPROTECTED field.

Attributes on a 3279 ONLY

The following map field attributes are available on *3279-type terminals only*.

Note: Three of these attributes, BLINK, REVERSE VIDEO, and UNDERSCORE, are mutually exclusive. For example, neither REVERSE VIDEO nor UNDERSCORE can be assigned to a field for which the BLINK attribute is defined.

Attribute	**Description**
BLINK/NOBLINK	■ BLINK specifies that the field blinks. ■ NOBLINK specifies that the field does not blink.
NORMAL VIDEO/ REVERSE VIDEO	■ NORMAL VIDEO specifies that the color of the field and of thebackground are not reversed. ■ REVERSE VIDEO specifies that the colors are reversed.
UNDERSCORE/ NOUNDERSCORE	■ UNDERSCORE specifies that the field is underscored. ■ NOUNDERSCORE specifies that the field is not underscored.
BLUE/RED/PINK/ GREEN/TURQUOISE/ YELLOW/WHITE/ NOCOLOR	Any one of these color attributes can be assigned to a field. NOCOLOR specifies that the default display color for the terminal is used.

Default Values for Attributes

Literal Fields	**Variable Fields**
ALPHANUMERIC	ALPHANUMERIC
PROTECTED	UNPROTECTED
SKIP	NOSKIP
NONDETECTABLE	NONDETECTABLE
DISPLAY	DISPLAY
NOMDT	NOMDT
NOBLINK	NOBLINK
NORMAL VIDEO	NORMAL VIDEO
NOUNDERSCORE	NOUNDERSCORE
NOCOLOR	NOCOLOR

Chapter 4: Automatic Editing and Error Handling

This chapter discusses about automatic editing and handling methods.

This section contains the following topics:

Automatic Editing

What is Automatic Editing?

The *automatic editing* capability of the CA IDMS mapping facility is used to edit and validate data entered in map data fields. With automatic editing, a map can tolerate greater variation in operator input, making it easier for the terminal operator to use.

For example, if month values are stored as 2-digit numbers (01 through 12) but the terminal operator prefers to see the month values spelled out in words (for example, January, February), automatic editing is used to translate spelled-out months into their numeric equivalents. The map developer can thus create a map that requests spelled-out months but stores their equivalent numeric values.

Editing Input

Automatic editing can perform any of the following operations *on input*:

- *Verify that the terminal operator has entered data* in all fields for which input is required
- *Validate terminal operator input* based on an external picture and a verification (edit) table
- *Convert input to storage format* based on a code table associated with a field
- *Convert input to internal format* using both the internal and external picture
- *Validate that an input buffer contains data* (content required)

Editing Output

Automatic editing can perform the following optional operations *on output*:

- *Decode data* based on a code table associated with a field
- *Convert data to external format* for display based on both the external picture and internal pictures
- *Validate that a buffer contains valid data* and aborts if not

When should Automatic Editing be Enabled?

Automatic editing should be enabled for a field (and the map that contains it) when the related record element has a usage other than DISPLAY:

- When *zeros are to be displayed when an operator nulls the value in a field* by pressing the ERASE EOF key, as specified by either of the following options:
  - The *Zero when null* option on the Map Read/Write Options screen
  - The ZEROED WHEN NULL option of the batch compiler MFLD statement
- When *blanks are to be displayed in the field when the value for a field is zero*, as specified by either of the following options:
  - The *Blank when zero* option on the Map Read/Write Options screen
  - The *Blank when zero* option of the batch compiler MFLD statement
- When *underscores are to be displayed* as specified by either of the following options:
  - The *Underscore blank fields* option on the Map Read/Write Options screen
  - The UNDERSCORE IF BLANK option
- When *uppercase translation by field* is selected

Error Handling

What is Error-handling?

The **error-handling** capability can be used to define display characteristics in the event that a map is redisplayed due to input error. Redisplay of a map on input error is controlled by the CA ADS dialog or application program that uses the map at runtime.

Error Handling Functions

Error handling can perform one or more of the following operations when the map is redisplayed:

- *Redisplay incorrect input* with predefined attributes that attract the operator's attention
- *Redisplay correct input* with predefined attributes
- *Provide messages* to the operator
- *Sound an alarm*

Before You use Editing and Error-handling

The following steps, which are described, must be performed before automatic editing and error-handling can be used:

1. Enable automatic editing and error-handling for the map and for each field to be edited
2. Optionally define error-handling criteria for the map
3. Define editing criteria for each field

Enabling Automatic Editing and Error Handling

Overview

Enabling automatic editing for a map also enables error-handling for that map. Automatic editing can be enabled for an entire map and also for individual fields.

Default Values

The mapping facility supplies the following default settings for automatic editing:

- **Entire map—**Enabled
- **Each field—**Disabled, unless any of the following editing criteria is specified

Important: Autopainted fields default to enabled if editing is appropriate.

A map that uses only these default values is not edited at runtime. The map developer must use the online or batch compiler to enable automatic editing for map fields. If automatic editing is disabled at the map level, automatic editing is disabled for all of that map's fields.

Map-level Editing

Automatic editing is enabled/disabled for an *entire map* when:

- The *Automatic editing* prompt on the first General Options screen is used to either enable (/) or disable automatic editing.
- The *EDIT/NOEDIT option of the batch compiler MAP statement* is used to enable (EDIT) or disable (NOEDIT) automatic editing.

Field-level Editing

- *Specifying any of the following* on the Field Definition screen enables automatic editing for a field:
  - **/** for the *Automatically edited* prompt (Online compiler only)
  - An external picture
  - An edit table
  - A code table
- *Naming a user-written edit module* enables or disables automatic editing:
  - Editing is enabled if the edit module is to be performed either before or after automatic editing.
  - Editing is disabled if the edit module is to be performed instead of automatic editing.

Automatic editing is enabled or disabled according to the most recent automatic editing or user-written error module specification. The General Options, Additional Edit Criteria, and Map Read/Write Options screens in the online compiler and the MFLD statement of the batch compiler are used to make these specifications and to enable automatic editing for a field.

Automatic Editing Criteria

Overview

- An *internal picture* describes the format in which data for a field is stored in the user buffer.
- An *external picture* describes the format in which data for a field is displayed on the operator's screen.
- An *edit table* optionally defines a set of valid or invalid values or ranges of values for a field.
- A *code table* optionally defines values for encoding and decoding data.

External pictures can be defined by using the online or batch compiler as well as IDD; internal pictures, edit tables, and code tables are defined externally to the mapping facility.

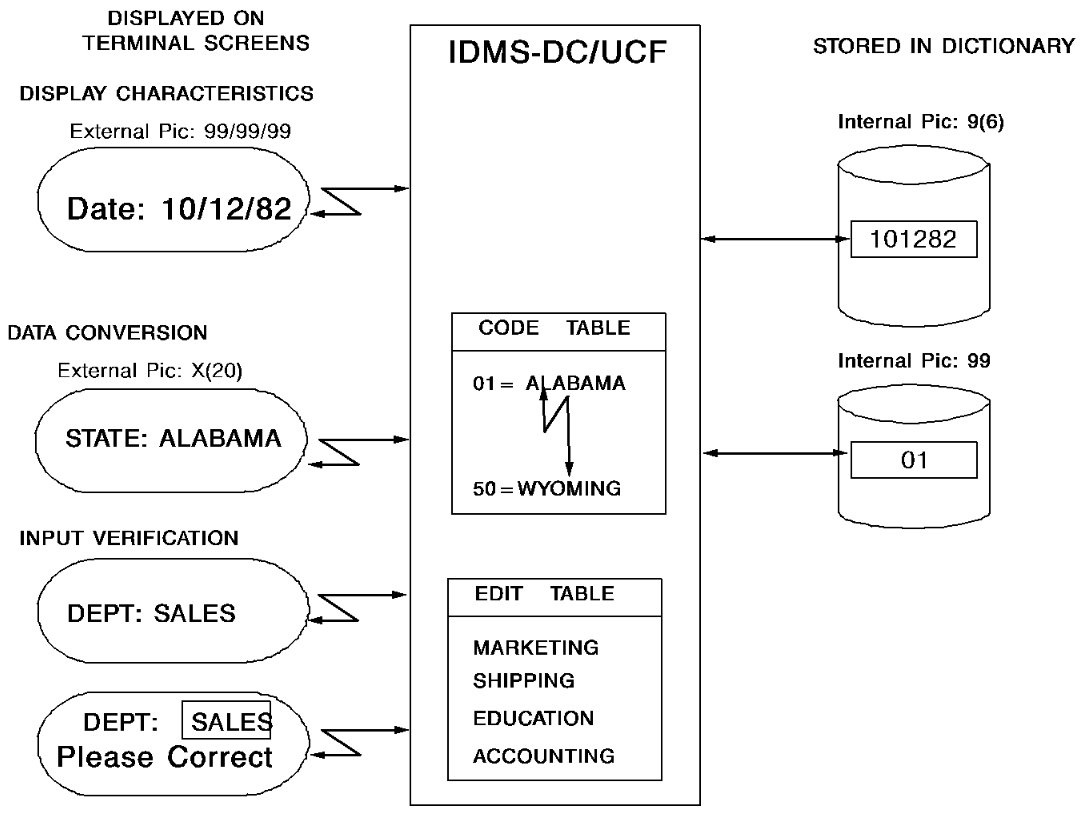
Types of Editing

The three automatic editing operations depicted in the following illustration and discussed on the following pages, can be performed:

- Display characteristics
- Data conversion
- Input verification

Automatic Editing Operations

The following diagram illustrates how automatic editing works.



Display Characteristics

Display characteristics are determined by the external picture.

For example, a date field can have an internal picture of 9(6) and an external picture of 99/99/99. If the internal picture for a DATE data field is 9(6) as many as six numeric digits can be stored for the field. The external picture of 99/99/99 specifies how a stored value is displayed on the terminal screen. Using this example, a date stored internally 101297 displays externally as 10/12/97.

Data Conversion

Data that is entered on a map is converted and validated against a code table. For example, each value for the STATE data field is stored as a 2-digit value, as specified by its internal picture 99. A code table then translates each 2-digit value to a complete state name and vice versa.

For example, the operator views ALABAMA when the stored value is 01. The value 50 is stored in program variable storage when the operator enters the word WYOMING.

Input Verification

Data entered on a map is compared to values in an edit table. Correct values for the sample DEPT data field are listed in an edit table. Operator input is validated against values in the edit table. (An edit table can contain either correct or incorrect values for a field.) For example, the term SALES is determined to be incorrect based on the edit table for the field; the dialog or program redisplays the error and asks the operator to correct the value.

Internal Pictures

Definition

Internal pictures define the data storage format for elements and the map fields associated with the elements. Internal pictures cannot be created or altered by the mapping facility.

Specifying Internal Pictures

An internal picture can be specified for a record element when the record element is defined by using either the IDD Data Dictionary Definition Language (DDDL) or the IDMS schema compiler. Internal pictures cannot be defined or altered by using the online or batch compiler.

How the Input is Converted

Before edited input is moved into program variable storage, if editing is on it is converted into its internal format based on the internal picture defined for the record element associated with the input field. The internal picture for a map field is the internal picture of the related record element.

Maximum Length

An internal picture can contain a maximum of 32 characters. The characters used to construct alphanumeric, alphabetic, and numeric internal pictures are listed in the following table:

Data Type	Character	Description
Alphanumeric	X	A single alphanumeric character.
	(*n*)	Follows an X to represent n consecutive repetitions of alphanumeric characters. *N* must be an integer in the range 1 through 9999.
Alphabetic	A	A single alphabetic character (A through Z).
	(*n*)	Follows an A to represent *n* consecutive repetitions of alphabetic characters. *N* must be an integer in the range 1 through 9999.
Numeric	9	A single numeric character (0 through 9).
	(*n*)	Follows a 9 to represent *n* consecutive repetitions of numeric characters. Preceding an implied decimal point position, *n* must be an integer in the range 1 through 9999. Following an implied decimal point position, *n* must be an integer in the range 1 through 255.
	V	Represents a decimal point position in fixed decimal numeric data. An internal picture can contain only one decimal. If a fixed decimal picture does not contain a V, the decimal position for the picture is after the rightmost 9.
	S	Indicates that signed data is maintained as either positive or negative. When used, S must be the first character in an internal picture.
	. (decimal point)	Represents the decimal point in floating point data with DISPLAY usage only. An internal picture can contain only one decimal point.
	E	Indicates the start of the floating point exponent. When used, an E must be preceded by at least one 9 and followed by at least one 9.

How are the Elements Stored at Runtime?

A USAGE clause in the record element definition determines the method of storing values for an element at runtime. The USAGE clause for an element associated with a map field can specify one of the following storage methods:

Storage Method	Will Store Data This Way
DISPLAY	Values are stored one character to a byte according to EBCDIC conventions. DISPLAY must be specified for alphanumeric and alphabetic internal pictures. DISPLAY can also be specified for numeric internal pictures.
COMP	Numeric values are stored in binary format.
COMP-1	Numeric values are stored in internal floating point (short precision) format.
COMP-2	Numeric values are stored in internal floating point (long precision) format.
COMP-3	Numeric values are stored in packed decimal format.

Note: COMP, COMP-1, -2, and -3 usages apply only to numeric data. Internal pictures cannot be specified for elements with COMP-1 or COMP-2 usage.

Note: For more information about record element definitions and the USAGE clause, see the *CA IDMS IDD DDDL Reference Guide*.

External Pictures

Automatic editing uses the external picture for a field on mapout and mapin as follows:

Mapout

On mapout, the external picture describes how data for the field is displayed on a terminal screen. The following example illustrates the interaction of output data and an external picture:

Program variable storage value	External Picture	Field Display
123456789	#XXX-XX-XXXX	#123-45-6789
99365	99/999	99/365

Mapin

On mapin, automatic editing uses the external picture as follows:

- The external picture is checked to determine if the characters in the field are valid. An external picture can be NUMERIC, ALPHABETIC, or ALPHANUMERIC; data in the field must conform to the external picture specifications to be valid.
- The external picture is used to eliminate insertion characters from data.

The following example illustrates the interaction of input data and an external picture:

User Input	External Picture	Value Stored
#123-45-6789	#XXX-XX-XXXX	123456789
99/365	99/999	99365

The external picture and input data are processed from left to right. If automatic editing is not enabled for both the map and field, the external picture is not used on mapin.

Implicit External Pictures

What is an Implicit External Picture?

If the developer does not use the online or batch compiler to explicitly specify an external picture for the field, an implicit external picture is constructed for a field. The *implicit external picture* is derived from the internal picture and/or from the usage mode defined for the related record element.

If Automatic Editing is not Enabled

The status of automatic editing for the field determines the external picture that is constructed for the field:

If automatic editing is not enabled for the field, the online compiler constructs the external picture for a field:

- *The data type* is alphanumeric
- *The length* is determined by the length (in characters/bytes) that is specified by the internal picture and/or by the usage mode of the associated record element

For example, the following table illustrates how an external picture is constructed for a field for which automatic editing is not enabled:

Internal Picture	Usage Mode	External Picture
XXX	DISPLAY	X(3)

Internal Picture	Usage Mode	External Picture
A(8)	DISPLAY	X(8)
S99V99	DISPLAY	X(4)
S9(4)	COMP	X(2)
	COMP-1	X(4)
	COMP-2	X(8)
9(7)V99	COMP-3	X(5)

Note: Internal pictures cannot be specified for COMP-1 or COMP-2 elements. The previous external pictures for COMP-1 and COMP-2 elements are the default external pictures for these elements when automatic editing is disabled for a field. To avoid an error (such as PROG-470, PROG-402, or PROG-403), automatic editing should be enabled for all fields with usage other than DISPLAY.

If Automatic Editing is Enabled

If automatic editing is enabled for the field, the online compiler assigns an external picture to the field:

- *If an external picture is defined for the associated element*, that external picture is assigned to the field
- *If an external picture is not defined for the element*, the mapping facility constructs an external picture for the field:
  - *The data type* is the same as the data type of the internal picture, when applicable. Fields associated with COMP-1 and COMP-2 elements are assigned predetermined numeric external pictures.
  - *The length* is determined by the length (in characters/bytes) that is specified by the internal picture, when applicable.
  - *The composition* is derived from the internal picture according to the translation equivalents listed in the following table.

The following table illustrates how an external picture is constructed for a field when automatic editing is enabled:

Internal Picture	Usage Mode	External Picture
XXX	DISPLAY	X(3)
A(8)	DISPLAY	A(8)
S99V99	DISPLAY	+99.99

Internal Picture	Usage Mode	External Picture
S9(4)	COMP (Half word)	+9(4)
S9(8)	COMP (Full word)	+9(8)
S9(16)	COMP (Double word)	+9(4)
	COMP-1	+9.9(7)+99
	COMP-2	+9.9(16)+99
9(7)V99	COMP-3	9(7).99

Note: Internal pictures cannot be specified for COMP-1 or COMP-2 elements. The previous external pictures for COMP, COMP-1, -2 and -3 elements are the default external pictures for these elements when automatic editing is enabled for a field.

Internal to External Translation

Internal Picture Character	External Picture Character
X	X
A	A
9	9
S	S
V	. (Decimal point)
E	E
. (Decimal point)	. (Decimal point)

Explicit External Pictures

To explicitly specify an external picture for a field, the developer uses the online or batch compiler. If an explicit external picture is specified for the field, an implicit external picture is not built.

When to Specify an External Picture

An external picture can be explicitly specified for a map field either during a map-definition session or in the IDD record element definition.

During Map Definition

An external picture is explicitly specified at map definition in any of the following:

- The *Edit picture* prompt on the Field Definition screen
- The EXTERNAL PICTURE clause of the batch compiler MFLD statement.

An external picture specification made for a map field overrides any other external picture specification that has been made for the field.

Using IDD

An external picture can be explicitly defined by using IDD. Specifying INTERNAL for the map field's external picture causes the online compiler to use the external picture associated with the record element definition. External pictures for record elements are defined in the IDD DDDL RECORD ELEMENT or COBOL substatement.

Note: For more information about these substatements, see the CA IDMS *IDD DDDL Reference Guide*.

How External Pictures are Constructed

If INTERNAL is specified for the map field's external picture but the record element is not associated with an external picture, the online compiler will construct an external picture as described in the following table:

Data Type	Character	Description
Alphanumeric	X	A single alphanumeric character.
	B	A single blank character; B can appear anywhere in the picture.
	Other	Characters other than A, B, or parentheses can be used as insertion characters.
Numeric	9	A single numeric character (0 through 9).

Data Type	Character	Description
	Z	An insertion character when it is preceded by a 9, a decimal point, or a zero-suppression character. Otherwise, a Z is a zero-suppression character.
	$	Multiple dollar signs at the beginning of an external picture represent a floating dollar sign. The dollar sign is an insertion character when preceded by a 9, a decimal point, or a zero-suppression character.
	*	Multiple asterisks at the beginning of an external picture provide check protection. The asterisk is an insertion character when preceded by a 9, a decimal point, or a zero-suppression character.
	+	In the first position of an external picture, indicates signed data, and appears as either a minus sign or a plus sign depending on the sign of the data. Multiple plus signs at the beginning of an external picture represent a floating sign. The plus sign is an insertion character when preceded by a 9, a decimal point, or a zero-suppression character.
	- (Minus sign)	In the first position of an external picture, indicates signed data. The sign position appears as a blank if the data is positive and as a minus sign if the data is negative. Multiple minus signs at the beginning of an external picture represent a floating sign. The minus sign is an insertion character when it is preceded by a 9, a decimal point, or a zero-suppression character.

Data Type	Character	Description
	. (Decimal point)	Used as a decimal point. Data is aligned with the decimal point in an external picture and is truncated or padded when necessary. The decimal point terminates zero suppression when zero-suppression characters precede the decimal point. Zero-suppression characters become insertion characters if placed after a decimal point. The first period in a series of period characters is the decimal point in a picture. If no decimal point exists in the data, a decimal point is assumed after the rightmost numeric character. The comma (,) is used as a decimal point if DECIMAL POINT IS COMMA is specified.
	B	A single blank character; B can appear anywhere in the picture.
	(*n*)	Follows a 9, A, Z, $, *, +, -, or B to represent *n* consecutive repetitions of the character. *N* must be an integer in the range 1 through 9999. When used following an implied decimal point position, as represented by a V, *n* must be in the range 1 through 255.
	V	Indicates the decimal point position in fixed decimal data.
	E+99	Indicates a floating point data field. The mantissa can be positive or negative. The exponent must be two numeric digits preceded by a plus sign. If more than two digits are entered, the online compiler truncates the picture; if less than two digits are entered, the picture is extended. The online compiler supplies a default external picture if automatic editing is enabled for the field. The default external picture for internal short float type data fields is +9.9(7)E+99. The default external picture for internal long float type data fields is +9.9(16)E+99.
	Other	Characters other than 9, Z, $, *, +, -, B, V, or parentheses can be used as insertion characters.

Special Considerations

An alphanumeric external picture must be specified for a field associated with a group element, regardless of the data type of the subordinate elements in the group.

Alphanumeric, alphabetic, and numeric external pictures must begin with specific characters:

- **Alphanumeric pictures** must begin with either X or B
- **Alphabetic pictures** must begin with either A or B
- **Numeric pictures** must begin with one of the following characters:
  - B
  - Z
  - 9
  - $
  - *
  - +
  - -

Allowing Insertion Characters

Including insertion characters in external pictures provides the terminal operator with flexibility in supplying data. The terminal operator need only type necessary data characters; the operator can include insertion characters in data, but omitting these characters does not constitute an error.

For example, given a telephone number with an external picture of XXXBXXX-XXXXB#BXXX, the terminal operator could enter any of the following values:

- 617 555-1212 # 341
- 617555-1212#341
- 6175551212341
- 6175551212

Considerations

The following considerations apply to the use of insertion characters in external pictures:

- Insertion characters can be included in external pictures to assist terminal operators in reading or interpreting output data. Insertion characters do not become part of the stored data.
- Insertion characters that are embedded in zero-suppression characters are not displayed until a significant digit appears in the data.
- The dollar sign ($) is the only insertion character that can occupy the first position in an external picture.

The following considerations apply to data supplied in fields with numeric external pictures:

- Leading and/or trailing blanks are automatically deleted from data entered in numeric fields. Thus, the terminal operator can start entering numeric data in any field position.
- A decimal point is assumed to exist after the rightmost digit of a terminal operator's input if the following conditions are in effect:
  - A decimal point is defined in the external picture for the field
  - No decimal point is supplied by the operator.

    The sample external picture 99.99 affects the display of numeric data in the examples in the following table:

Value input by operator	Value displayed
1	01.00
.1	00.10
10	10.00

Truncation

If data typed by the terminal operator exceeds the length permitted by the external picture, excess characters are deleted from the data as described in the following table:

Type of Field	Truncation Process
Alphanumeric	The rightmost characters are deleted from the data.

Type of Field	Truncation Process
Numeric	■ *If excess characters are supplied to the right of thedecimal*, the rightmost (low-order) characters are deleted from the data. ■ *If excess characters are supplied to the left of the decimal*, the leftmost (high-order) characters are deleted from the data.

Data from which low-order characters are deleted is moved to program variable storage as usual. Numeric data from which high-order characters are deleted is not moved to program variable storage; an input error occurs for the field

Sample Truncations

The following sample external pictures demonstrate how excess characters are deleted from data:

External picture	Operator inputs	Data stored as
XXX	A234	A23
AAA	WXYZ	WXY
999	1234	no data
99.99	23.4	23.40
99.99	123.4	no data
99.99	23.456	23.45

A CA ADS dialog or application program can determine whether excess data has been deleted from data in a field by inquiring if the field has been truncated.

Edit and Code Tables

Overview

Existing edit and code tables can be optionally associated with map fields. Automatic editing uses the edit and code tables for a field on mapout and mapin as follows:

- *On mapout*, the code table (if any) is used to convert stored data to its decoded form for display at the terminal.
- *On mapin*, the edit and code tables for a field are used as described:
  - The *edit table* (if any) is used to validate operator input. An edit table can contain either valid or invalid values:
    - *If the edit table contains valid values*, data input in the map field is valid only if it is listed in the table
    - *If the edit table contains invalid values*, data input in the map field is valid only if it is not listed in the table
  - The *code table* (if any) is used to convert data typed in a field to its encoded form for storage. In this way, the code table also validates data unless NOT FOUND is used to pass incorrect data through.

A given map field can have a maximum of one edit table and one code table enabled for it. Edit and code tables cannot be specified for a field that is associated with a group element unless all its elements are defined with usage DISPLAY.

The arrangement of values in a table, the searching algorithm for a table, and the relationship between a table and a map load module that uses the table are determined by the IDD DDDL statement that defines the table.

The definition of tables is discussed in the appendix "Generating Edit and Code Tables".

Values in Edit and Code Tables

Edit and code tables contain the values used by automatic editing. Values in a table can be either numeric or alphanumeric depending on the data type specified when the table was created.

Edit Table Values

Edit table values must be compatible with the data type and length of data to be evaluated at runtime. The configuration of input data at runtime depends on the specific automatic editing processes that are performed on the data.

Automatic editing performs the following processes prior to edit table operations:

- *Numeric input data is reversed* if REVERSE NUMERIC is enabled for the field.
- *Data is altered according to external picture specifications* for example:
  - Leading and trailing blanks are stripped from numeric data
  - The decimal point in numeric data is aligned with the decimal point in the external picture
  - Insertion characters are stripped from data
  - Characters that exceed the length specified by the external picture are deleted from the data

Insertion characters are not included in edit table values. For example, the following sample edit table lists values for a ROOM NUMBER field with an external picture of X-X(3) and an internal picture of X(4):

External Picture	Edit Table Values
E-101	E101
E-203	E203
G-221	G221

Code Table Values

Code table values specify encoded and decoded values:

- *Encoded values* specify data that can be stored in program variable storage. The data type and length of encoded values must be compatible with the internal format for the field with which the code table is associated. The internal format for a field is determined by the internal picture defined for the associated record element.

  For example, the following sample encoded values are valid in terms of their corresponding internal pictures:

Encoded value	External picture	Definition
OS	XX	department value
01	9(2)	state value

- *Decoded values* specify data that can be displayed on a terminal screen. The data type and length of decoded values must be valid in terms of the external picture for the field with which the code table is associated. Insertion characters are not included as decoded values in code tables.

  For example, the following sample decoded values are valid in terms of their corresponding external pictures:

Encoded value	Internal picture	Definition
OFFICE SERVICES	X(20)	department value
ALABAMA	X(20)	state value

On mapin, the edit table (if any) is invoked before the code table (if any) for a field.

DDDL Compiler Options

DDDL compiler conventions apply when specifying values for edit and code tables:

- **Delimit words or clauses** by using one of the following delimiter characters:
  - Blank
  - Comma
  - Period
  - Semicolon
  - Colon
  - Apostrophe
  - Parenthesis
  - Quotation mark

- **Embed one or more delimiter characters** by enclosing the value that contains delimiter characters in a pair of site-standard quote characters. The single quote (') is the default. For example, the value OFFICE SERVICES in the following sample code table contains an embedded blank character:

```
01         SHIPPING
02        PERSONNEL
03       ACCOUNTING
04        MARKETING
05  'OFFICE SERVICES'
```

  Single quotes around the value OFFICE SERVICES indicate that the space character between the two words in the value is part of the value rather than a delimiter between two values.

- **Embed the site-standard quote character** in a value by coding the quote character twice. For example, the value USER'S SITE in the following sample code table includes the default quote character ('):

```
50       DENVER
60       'BOSTON RGNL OFFICE'
70       'CHICAGO RGNL OFFICE'
85       'USER''S SITE
```

Note: For more information about DDDL compiler conventions and specifying values for tables, see the CA IDMS *IDD DDDL Reference Guide*.

Special Values

The following special values can be included in code tables to facilitate use of the tables at runtime:

- The **NOT FOUND** keyword can be included in a code table to define a catchall for a decoded or encoded value:
  - As an **encoded value**, NOT FOUND ensures that an unanticipated stored value does not cause an abend on mapout. The decoded value that corresponds to NOT FOUND is displayed for unanticipated input.
  - As a **decoded value**, NOT FOUND ensures that unanticipated input does not cause an input error when evaluated by the code table. The encoded value that corresponds to NOT FOUND is stored for unanticipated data.

  NOT FOUND must not be enclosed in quotation marks when used as a keyword. For example, the following sample code table for a DEPARTMENT field includes NOT FOUND as an encoded and a decoded value:

```
01          SHIPPING
02          PERSONNEL
03          ACCOUNTING
04          MARKETING
05          'OFFICE SERVICES'
00          NOT FOUND
NOT FOUND   MISSING
```

  The value 00 is stored for the field when an operator supplies a value that is not included as a decoded value in this sample table. The word MISSING is displayed when a value other than 01, 02, 03, 04, 05, or 00 is stored for the field.

  Note: For more detailed information about the NOT FOUND keyword, see the appendix “Generating Edit and Code Tables".

- The **null value**, (''), two consecutive quotation marks, should be included as a decoded value in a code table in the following cases:
  - The **numeric data field** is to be filled with zeros when automatic editing is enabled and the operator presses the ERASE EOF key, as specified by either of the following options:
    - The *Zero when null* option on the Map Read/Write Options screen
    - The ZEROED WHEN NULL option of the batch compiler MFLD statement
  - The **alphanumeric data field** is defined without a pad character.

  The following sample code table for a DEPARTMENT field included the null value as a decoded value:

```
00            ''
01            SHIPPING
02            PERSONNEL
03            ACCOUNTING
04            MARKETING
05            'OFFICE SERVICES'
```

  The decoded value 00 is stored for this example when the operator presses ERASE EOF at the beginning of the associated alphanumeric map field and submits a null value for the field.

Enabling tables

Built-in Edit or Code Tables

If a record element includes an edit and/or code table in its definition, the table is referred to as a *built-in* edit or code table. The built-in edit or code table is available to any map data field that uses the record. The built-in table for a field is used for the field if both of the following conditions are met:

- Automatic editing is enabled for the field (and map).
- The map developer does not use the mapping facility to specify an edit/code table.

Suppressing Built-in Tables

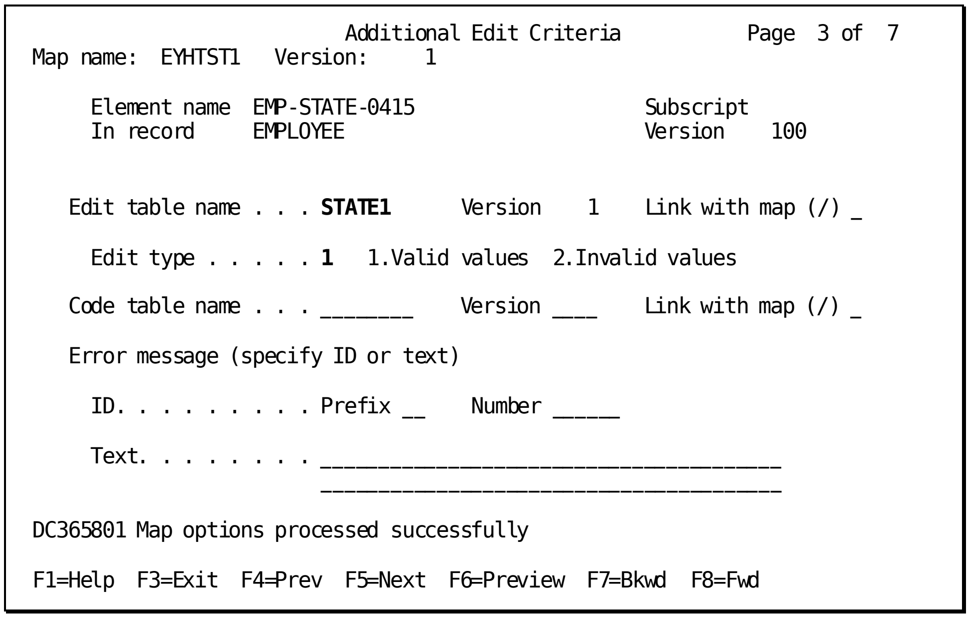
To suppress use of a built-in table without using another table, the developer must disable automatic editing for the field.

The map developer overrides the use of a built-in table (if any) by specifying a stand-alone table for the field. Stand-alone tables are created in the data dictionary by the DDDL TABLE statement.

Specifying a Stand-alone Table

The developer uses the online or batch compiler to specify a stand-alone table:

The Edit table name and Code table name prompts on the Additional Edit Criteria screen can be used to specify the name of a table to be used for the field. For example, the *state1* table is specified as an unlinked edit table of valid values in the following sample screen.


```
                             Additional Edit Criteria          Page  3 of  7
Map name:  EYHTST1   Version:     1

     Element name  EMP-STATE-0415                    Subscript
     In record     EMPLOYEE                          Version    100


   Edit table name . . . STATE1      Version    1    Link with map (/) _

     Edit type . . . . . 1   1.Valid values  2.Invalid values

   Code table name . . . ________    Version ____    Link with map (/) _

   Error message (specify ID or text)

     ID. . . . . . . . . Prefix __    Number _____

     Text. . . . . . . . ________________________________________
                         ________________________________________

DC365801 Map options processed successfully

F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F7=Bkwd  F8=Fwd
```

Suppressing a Stand-alone Table

To suppress use of a specified stand-alone table without specifying another table, the map developer can eliminate the name of the table from the Additional Edit Criteria screen by performing either of the following actions:

- Pressing the ERASE EOF key to erase the table name
- Typing blanks over the table name

Specifying a Stand-alone Table using the Batch Compiler

The EDIT TABLE and CODE TABLE clauses of the batch compiler MFLD statement can be used to specify the name of the table to be used for the field. For example, the DEPTEDIT table is specified as an unlinked edit table of valid values in the following sample MFLD statement:

```
ADD MFLD DEPT-ID
    DFLD ID-0410
    EDIT TABLE IS DEPTEDIT
         NOLINK
         USAGE IS VALIDATE.
```

Alternate Names

Type of table	Alternate name	Reason
Built-in	Tightly coupled	Because each table is part of the element for which it is defined
Stand-alone	Loosely coupled	Because they are separate from the record elements with which they are associated

Linked VS. Unlinked

When the map developer specifies a stand-alone table for use during automatic editing, the table must be designated either linked or unlinked as described:

If the table is	This happens
Linked	The table is included in the map load module with which it is associated.
Unlinked	The table is loaded at runtime by the map load module with which it is associated.

Enabling a table as an unlinked table eliminates the need to regenerate maps that use the table, should the table definition change. A stand-alone table is specified as linked or unlinked on the same screen or in the same batch MFLD statement that names the table for a field.

Note: For more information about the definition of built-in and stand-alone table, see the appendix “Generating Edit and Code Tables".

Error-Handling Criteria

What is Error-handling?

Error handling alerts the terminal operator if an input error occurs.

A field is considered to be in error if any of the following conditions exist:

- The input (or the encoded equivalent) does not conform to the internal picture.
- The input does not conform to the external picture.
- Numeric input is truncated on the left (high-order truncation).
- No input is supplied for a required field.
- The input is determined to be invalid, based on the edit table used for the field.
- The input does not match any decoded values in the code table used for the field, and no catchall value is defined for decoded values.
- The input is determined to be in error according to criteria defined by a user-written edit module.

Note: For more information about the use of the automatic editing to establish correct/incorrect input conditions for error handling, see the chapter "Automatic Editing Criteria”.

Note: For more information about the definition and use of user-written edit modules, see the appendix "User-Written Edit Modules".

What Error-handling Criteria can be Defined?

The following table lists the error-handling criteria that can be defined for each map. Each criterion is described separately:

Criteria	Use
Attributes for correct input	Identifies correct input and input that has not been edited
Attributes for incorrect input	Identifies erroneous input
Error message	Provides the terminal operator with any messages associated with the field in error
A terminal alarm	Informs the terminal operator that an input error has occurred

Attributes for Correct and/or Incorrect Input

Definition

Attributes that are used when the mapped display is initially mapped out are assigned individually to fields when the fields are defined. Attributes used to draw attention to correct and/or incorrect data are assigned to an entire map when the map is defined. These error-handling attributes override the field-level attributes when an edit error is redisplayed.

How to Define Attributes

Either the online or batch compiler can be used to define attributes for correct and/or incorrect fields:

- *Page two of the General Options screen* can be used to establish attributes for use by error handling. The developer selects attributes on this screen to establish those attributes for use with error-handling.
- The *batch compiler MAP statement ON EDIT ERROR clause* can be used to establish attributes for error handling. The following specifications can be made for the ON EDIT ERROR clause:
  - The *INCORRECT FIELDS ATTRIBUTES specification* names attributes for fields that contain incorrect input.
  - The *CORRECT FIELDS ATTRIBUTES specification* names attributes for fields that contain correct input.

    The list of available attributes is detailed in "Attributes for Fields".

Attributes that are defined for correct fields are also used to redisplay variable fields for which editing was not performed.

Error Messages

Default Error Message

Error handling provides a default error message for any field. The message has the following format:

```
ERROR AT row, column
```

How to Override the Default Message

The developer can override the default error message for a field by specifying an error message for use by error-handling. The online or batch compiler can be used to define an error message:

- The **Error message** prompt on the Additional Edit Criteria screen can be used to establish an error message for the field being defined. The error message can be supplied at the time the map field is defined, can be defined in the data dictionary, or can be the default message.

```
                         Additional Edit Criteria          Page  3 of  7
Map name:  EYHTST1   Version:    1

     Element name  EMP-STATE-0415                    Subscript
     In record     EMPLOYEE                          Version    100


   Edit table name . . . _______     Version ___     Link with map (/) _

     Edit type . . . . . _   1.Valid values  2.Invalid values

   Code table name . . . _______     Version ___     Link with map (/) _

   Error message (specify ID or text)

     ID. . . . . . . . . Prefix __    Number _____

     Text. . . . . . . .  NOT A VALID STATE CODE
                         ________________________________________

DC365801 Map options processed successfully

F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F7=Bkwd  F8=Fwd
```

In the previous screen, the developer specifies a message on the line after the Text prompt. The message, **NOT A VALID STATE CODE**, is specified for the data field that follows STATE in this example.

- The *ERROR MESSAGE clause of the batch compiler MFLD statement* can be used to establish an error message for the field being defined. You can either accept the default message, or define the error message in the data dictionary.

  For example, an error message is defined for the EMP-STATE map field in the following sample MFLD statement:

```
ADD MFLD EMP-STATE
    REQUIRED
    EXTERNAL PICTURE IS 'XX'
    ERROR MESSAGE IS
    'NOT A VALID STATE'.
```

Defining a Message Field for a Map

An error message displays an input error only if a message field is defined for the map. The developer defines a message field for a map by using the online or batch compiler:

- *The ELEMENT NAME and SUBSCRIPT prompts on the second page of the Field Definition screen* are used as follows:
  - *ELEMENT NAME*—The developer types the keyword $MESSAGE (or $M) to establish a variable field as a message field.
  - *SUBSCRIPT*—The developer optionally specifies the maximum number of characters that the message field can contain in the SUB field (default is 80).

    Note: When the developer enters either $MESSAGE or $M in the ELEMENT NAME field and presses <Enter>, the literal, SUBSCRIPT, changes to LENGTH and irrelevant fields are darkened, as illustrated in the following sample screen.

```
                                  Field Definition                   Page  1 of  7
 Map name:  EYHTST1   Version:     1
 ...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80



_...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80
 Field at row  21   column  80                                     Drop field (/) _

      Element name: $message                        Subscript 80
      In record                                        Version

      Edit Picture

      Display intensity  1  1. Normal    2.  Bright         3. Hidden
      At end of field    3  1. Auto-tab  2.  Lock keyboard  3. Take no action

      Unprotected (/) . . . . .  /       Required (/). . . . . .  _
      Automatically edited (/)   _       Skipped by tab key (/)   _

 DC366004 Specify the variable field and any attributes

 F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F8=Fwd
```

```
                                   Field Definition                   Page  1 of  7
 Map name:  EYHTST1   Version:     1
 ...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80



_
_______________________________________________________________________________
 ...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80
 Field at row  21   column  80                                     Drop field (/) _

      Element name: $MESSAGE                        Subscript 80




      Display intensity  2  1. Normal    2.  Bright         3. Hidden
      At end of field    3  1. Auto-tab  2.  Lock keyboard  3. Take no action

      Unprotected (/) . . . . .          Required (/). . . . . .  _
      Automatically edited (/)   _       Skipped by tab key (/)   _

 DC366004 Specify the variable field and any attributes

 F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F8=Fwd
```

- The **MESSAGE LENGTH clause of the batch compiler MFLD statement** is used to establish a variable panel field as a message field and specify the maximum number of characters that the message field can contain.

  For example, a message field (MESS1) is defined for a map in the following sample MFLD statement:

```
ADD MFLD MESS1
    MESSAGE LENGTH 80.
```

How are Messages Displayed?

The message field for a map displays the error message for a field in error if the dialog or program that uses the map redisplays the map with error-handling attributes. When several data fields contain incorrect input, the message field displays as many error messages as possible, in order of occurrence from top to bottom, right to left, of the incorrect data fields. A blank is displayed between each error message.

Alarm Status on Input Error

You can enable or disable the error alarm feature using the following:

- The *Sound alarm on error* prompt on the General Options screen
- The *ON EDIT ERROR* clause of the batch compiler MAP statement

Automatic Editing at Runtime

If automatic editing is not enabled for a field, data in the field is handled as follows:

- **On mapin**, data that has been changed (or for which the MDT is set on) is justified, padded, or zeroed as specified by each map field definition. The data is then moved to program variable storage if the Transmit data entry field on the Map Read/Write Options screen is **/** for input.
- **On mapout**, data is moved directly to the output buffer from program variable storage. The data is then transmitted to the screen if DATA is Y (YES) for output.

Automatic editing is performed on mapin and mapout only if editing is enabled at the map level and only for fields for which editing is specifically enabled.

Note: For more information about enabling automatic editing for a map and its fields, see Enabling Automatic Editing and Error Handling (see page 53).

When enabled for a map and field, automatic editing validates and edits data supplied by the operator. Each field is evaluated according to specifications made for the field when the field was defined. Automatic editing operations that take place during mapin and mapout operations are presented in this section.

Mapin Operations

Automatic editing operates on a single field at a time. Fields for which automatic editing is enabled are evaluated in order of occurrence on the map. Automatic editing evaluates and edits each field, as illustrated in the following figure.

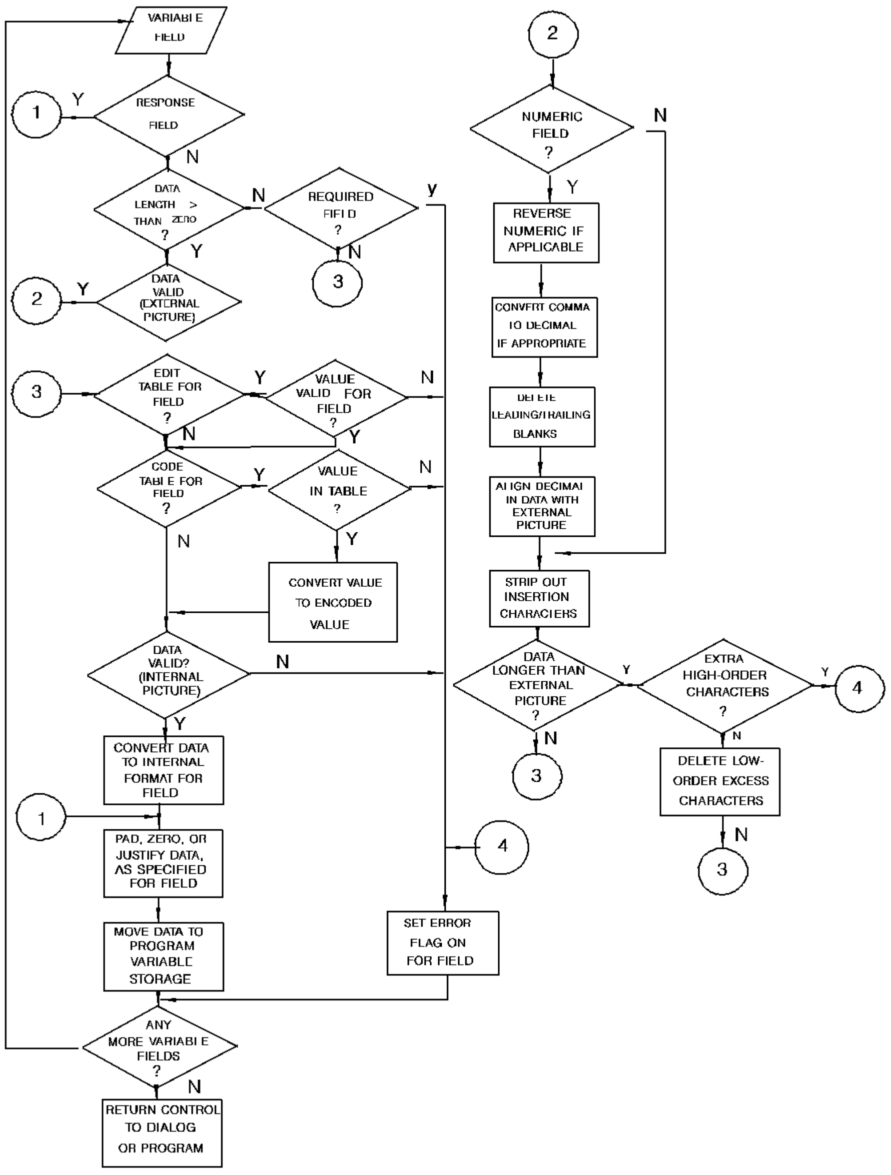
A given automatic editing operation must be completed without errors before automatic editing can advance to the next editing operation for that field. Automatic editing stops editing a field when an input error is detected in the field; the data is not moved to storage.

Automatic editing completes editing for a field in which no input error is detected by performing the following steps:

- The edited data is moved to the CA ADS record buffers or the program variable storage associated with the map
- Automatic editing either begins for the next field to be edited or finishes for the map:
  - Automatic editing begins for the next field on the map to be edited until all designated fields have been evaluated by automatic editing
  - Automatic editing is completed for the map if all fields for which automatic editing is enabled have been edited
- Automatic editing converts alphabetic characters to upper case if TRANSLATE TO UPPER CASE was specified for the field
- Automatic editing removes trailing underscore characters if UNDERSCORE BLANK FIELD was specified for the field

Automatic Editing on Mapin for Non-pageable Maps, Headers, and Footers

The following figure illustrates how automatic editing is handled on mapin operations for a non-pageable map and the header and footer areas of a pageable map.


VARIABLE FIELD
RESPONSE FIELD
DATA LENGTH > THAN ZERO ?
REQUIRED FIELD ?
DATA VALID (EXTERNAL PICTURE)
EDIT TABLE FOR FIELD ?
VALUE VALID FOR FIELD ?
CODE TABLE FOR FIELD ?
VALUE IN TABLE ?
CONVERT VALUE TO ENCODED VALUE
DATA VALID? (INTERNAL PICTURE)
CONVERT DATA TO INTERNAL FORMAT FOR FIELD
PAD, ZERO, OR JUSTIFY DATA, AS SPECIFIED FOR FIELD
MOVE DATA TO PROGRAM VARIABLE STORAGE
SET ERROR FLAG ON FOR FIELD
ANY MORE VARIABLE FIELDS ?
RETURN CONTROL TO DIALOG OR PROGRAM
NUMERIC FIELD ?
REVERSE NUMERIC IF APPLICABLE
CONVERT COMMA TO DECIMAL IF APPROPRIATE
DELETE LEADING/TRAILING BLANKS
ALIGN DECIMAL IN DATA WITH EXTERNAL PICTURE
STRIP OUT INSERTION CHARACTERS
DATA LONGER THAN EXTERNAL PICTURE ?
EXTRA HIGH-ORDER CHARACTERS ?
DELETE LOW-ORDER EXCESS CHARACTERS
Y
N
1
2
3
4

Automatic Editing on Mapin for Pageable Maps

Automatic editing on mapin occurs as soon as the GET DETAIL statement in the process code is initiated. The code must check automatic editing for the results. The previous illustrated procedure is implemented however, it takes place after the GET DETAIL statement is encountered in the process code.

Additionally, automatic editing error messages found on input (mapin - after the GET DETAIL statement has been issued) are *immediately* moved into the message buffer area and cannot be suppressed.

How the Online Compiler Responds to Errors

If input is found to be in error during editing, the online compiler responds as follows:

1. The field is identified as being in error.
2. Automatic editing ends for the field.
3. Automatic editing either continues with the next appropriate field on the map or relinquishes control to the program that requested the mapin, as described:
   - **If there is another data field on the map to be edited**, editing begins for that field.
   - **If there are no more fields on the map to be edited**, control returns to the CA ADS dialog or the application program:
     - **Control returns to the CA ADS dialog** that requested the mapin, as instructed by the EXECUTE ON EDIT ERRORS specification for the dialog.

       If **EXECUTE ON EDIT ERRORS is YES**, control passes to the appropriate response process.

       If **EXECUTE ON EDIT ERRORS is NO**, control does not pass to the response process until the operator either corrects all input errors detected by automatic editing or terminates the dialog.
     - **Control returns to the application program** that requested the mapin.

   Note: For more information about how dialogs and programs respond to input errors in maps, see Error Handling at Runtime (see page 86).

Considerations

The following considerations apply to runtime automatic editing:

- Automatic editing does not evaluate data in a response field
- Only data fields for which the MDT is set on are transmitted to program variable storage. The MDT can be set:
  - If the operator has entered a value in the field
  - If the program has been modified
  - If the MDT option was specified for the field at map definition
- Input must be supplied in a required field; failure to supply input constitutes an input error
- Input does not have to be supplied in a field that is not a required field

Determining if a Field is Changed or Erased

Map inquiry statements issued by the CA ADS dialog or application program that uses the map determine that data is changed or erased when the ERASE EOF key is pressed:

- **A field is identified as changed** when ERASE EOF erases the contents of the field in either of the following cases:
  - A pad character is defined for an alphanumeric field
  - ZEROED WHEN NULL is specified for a numeric field
- **A field is identified as erased** when ERASE EOF is pressed for the field while the cursor is at the first position of an alphanumeric field for which no pad character is defined.

Mapout Operations

Automatic editing operates on a single field at a time. Automatic editing processes each field as illustrated in the following figure. A given automatic editing operation must be completed without errors for a field before automatic editing can advance to the next editing operation for that field.

A translation character can be defined at system generation to be output when invalid data is found on mapout. If the translate character is defined as a null or a blank, no translation is performed. The default translation character values is the at character (@). Data that contains invalid character values (such as packed data that cannot be unpacked or bit data) is converted to a single @ character; the @ character is displayed in the field when the map is displayed.

The code table (if any) for a field is used to convert an encoded stored value to a decoded display value. The dialog or program that uses the map abends if an unanticipated stored value is evaluated by a code table that does not include the keywords NOT FOUND as an encoded value.

Automatic editing places blanks in numeric input if the *Blank when zero* option is on for the field and the input contains only zeros.

Automatic editing converts input to external format from internal format by processing the external picture for the field from right to left. If an error occurs during this phase, the task requesting the mapout is abended.

Automatic editing places underscores in a blank field if UNDERSCORE BLANK FIELD was selected for the field.

Edit tables are not used in mapout operations.

Error Handling at Runtime

When is a Field in Error?

A field is identified as being in error when automatic editing detects an error in that field on mapin. User-written edit modules, CA ADS dialogs, and application programs can perform their own editing and validation and specify whether a field is in error.

Defining Attributes for Redisplay

The CA ADS dialog or application program that uses a map can redisplay a map that contains fields in error.

Attributes can be defined for the redisplay of correct and incorrect fields. Such error-handling attributes can be used to draw the operator's attention to input errors. Error messages can be specified for fields in error. Additionally, the terminal alarm can be sounded when a map contains input errors. Specifications for the redisplay of maps with input errors are made as follows:

- The *mapping facility* can be used to make error-handling specifications for a map when the map is defined
- The *CA ADS dialog or application program* that issues the mapout can modify error-handling specifications for the map

Error-handling attributes defined by the mapping facility are available for use when automatic editing and error-handling are enabled for the map (regardless of the individual field settings).

Note: For information about how to enable automatic editing and error-handling for a map, see Enabling Automatic Editing and Error Handling (see page 53).

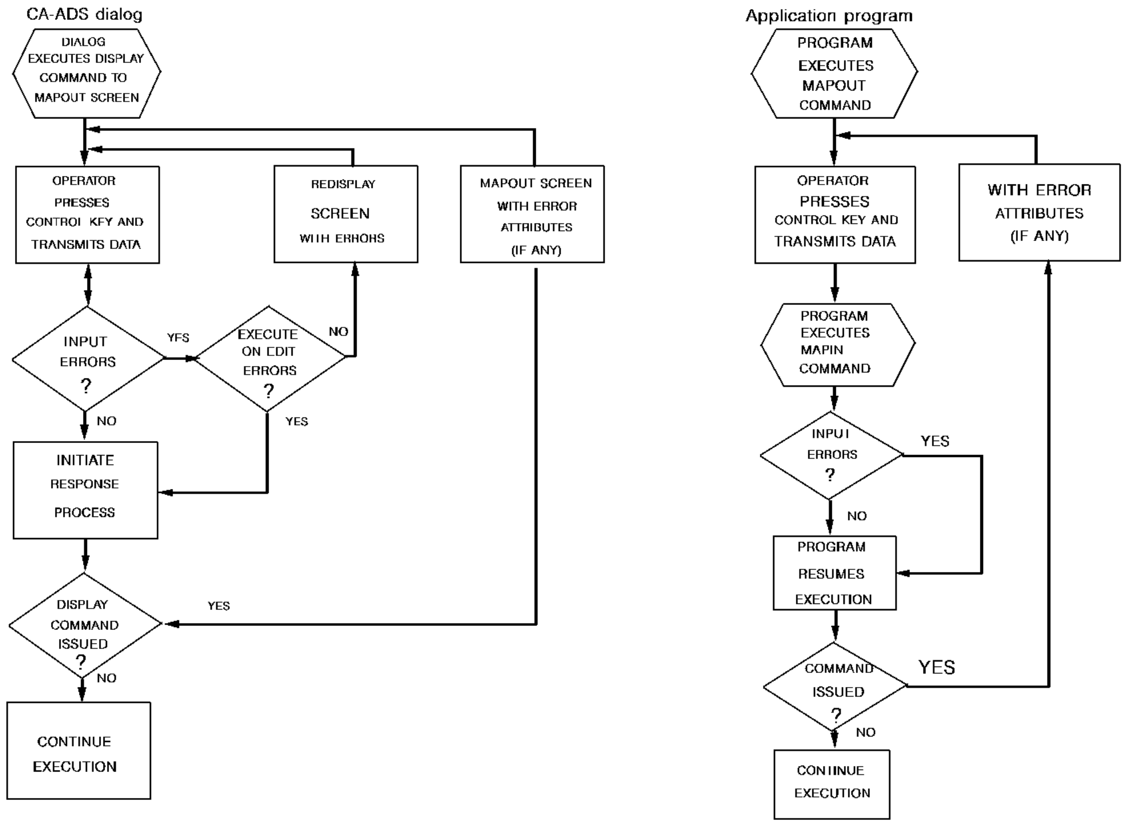
The ways in which CA ADS dialogs and application programs redisplay maps are contrasted in the following figure. Dialogs and programs can be set up to handle input errors as follows.

- A *CA ADS dialog* EXECUTE ON EDIT ERRORS YES/NO specification determines how the dialog executes if fields are in error:
  - *YES* specifies that control passes to the appropriate response process. The response process can include statements to determine if errors have been detected on mapin, to set additional fields in error, and so forth.

    A DISPLAY command must be used to redisplay the map for the operator. The map is redisplayed according to current error-handling specifications.
  - *NO* specifies that control is not passed to the response process. The map is redisplayed according to current error-handling specifications. The terminal operator must correct all map fields that are in error before control passes to the appropriate response process.

  Note: For more information about the CA ADS features and syntax, see the *CA ADS Reference Guide*.
- An *application program* includes Data Manipulation Language (DML) statements to determine if errors have been detected on mapin, to set additional fields in error, and so forth. A DML statement can issue a mapout to redisplay the map according to current error-handling specifications.

Dialogs, Programs, and Input Errors


CA-ADS dialog
DIALOG EXECUTES DISPLAY COMMAND TO MAPOUT SCREEN
OPERATOR PRESSES CONTROL KEY AND TRANSMITS DATA
REDISPLAY SCREEN WITH ERRORS
MAPOUT SCREEN WITH ERROR ATTRIBUTES (IF ANY)
INPUT ERRORS ?
YES
EXECUTE ON EDIT ERRORS ?
NO
YES
NO
INITIATE RESPONSE PROCESS
DISPLAY COMMAND ISSUED ?
YES
NO
CONTINUE EXECUTION
Application program
PROGRAM EXECUTES MAPOUT COMMAND
OPERATOR PRESSES CONTROL KEY AND TRANSMITS DATA
WITH ERROR ATTRIBUTES (IF ANY)
PROGRAM EXECUTES MAPIN COMMAND
INPUT ERRORS ?
YES
NO
PROGRAM RESUMES EXECUTION
COMMAND ISSUED ?
YES
NO
CONTINUE EXECUTION

An application program should not issue mapout requests with the NEWPAGE (ERASE) specification during error-handling; NEWPAGE (ERASE) maps out all map fields, including literal fields, and changes MDT settings.

Note: For more information about the development of application programs that interact with maps, see the appropriate *CA IDMS DML Reference Guide*.

Steps Performed by Runtime Mapping

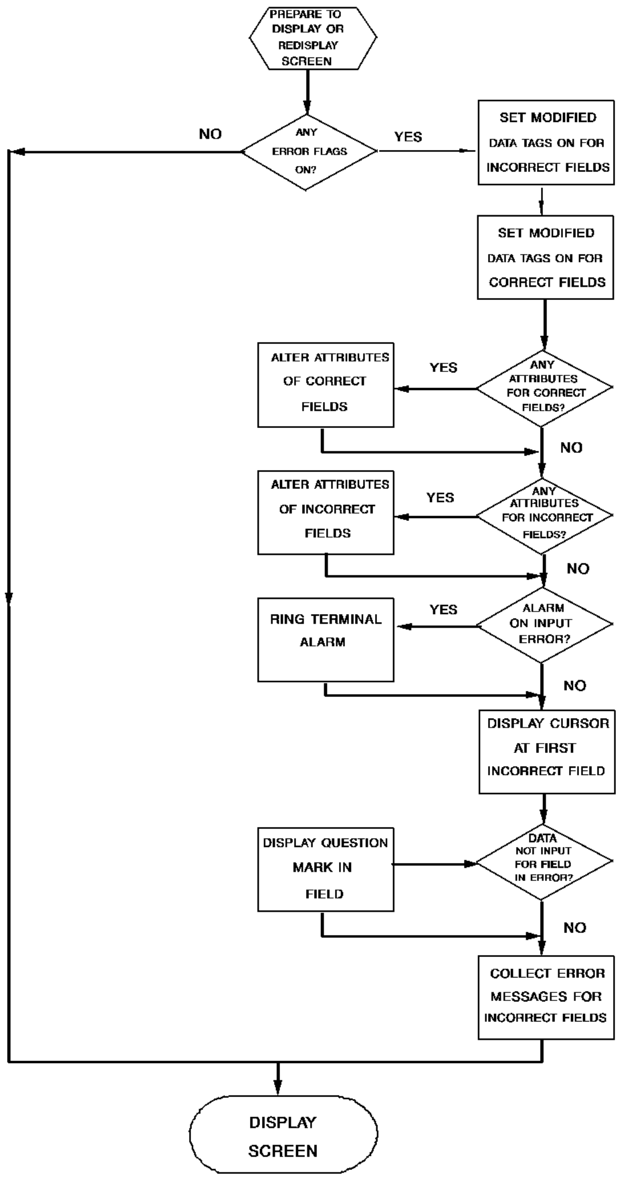
When a CA ADS dialog or application program redisplays a map for which automatic editing and error-handling are enabled, current error-handling attributes are automatically used for the field.

The list and diagram that follow illustrate the steps performed by runtime mapping:

- Modified data tags are set for correct fields.
- Correct-field attributes (if any) take effect for fields that are not in error or that were not edited.
- Incorrect-field attributes (if any) take effect only for fields that are in error.
- The cursor is displayed at the first map field in error.

- Blank required fields or any edited field for which data was not transmitted are redisplayed with a question mark (?) character.
- Data that is not in error is moved from the I/O buffer to program variable storage on a mapin operation if DATA is Y (YES) on mapin for the field.
- Data that is in error is not moved from the I/O buffer to protected variable storage on a mapin operation.
- On an *initial display* of a map by a CA ADS dialog, all literals and data fields are transmitted even if a field is in error. However, *in all other* cases, during a mapout operation, if any field is flagged as being in error, then, for all fields (both correct and incorrect), only attribute bytes are transmitted back to the screen; no data is moved from program variable storage to the screen.

Error Handling at Runtime


PREPARE TO DISPLAY OR REDISPLAY SCREEN
ANY ERROR FLAGS ON?
NO
YES
SET MODIFIED DATA TAGS ON FOR INCORRECT FIELDS
SET MODIFIED DATA TAGS ON FOR CORRECT FIELDS
ANY ATTRIBUTES FOR CORRECT FIELDS?
YES
ALTER ATTRIBUTES OF CORRECT FIELDS
NO
ANY ATTRIBUTES FOR INCORRECT FIELDS?
YES
ALTER ATTRIBUTES OF INCORRECT FIELDS
NO
ALARM ON INPUT ERROR?
YES
RING TERMINAL ALARM
NO
DISPLAY CURSOR AT FIRST INCORRECT FIELD
DATA NOT INPUT FOR FIELD IN ERROR?
DISPLAY QUESTION MARK IN FIELD
NO
COLLECT ERROR MESSAGES FOR INCORRECT FIELDS
DISPLAY SCREEN

Chapter 5: Pageable Maps

This chapter discusses about pageable maps, map-paging sessions, and dialog and program operations.

This section contains the following topics:

Overview

What is a Pageable Map?

A *pageable map* is a map that can contain unlimited occurrences of a set of map fields. Each occurrence of the set of fields is called a *detail occurrence*.

A pageable map can contain more detail occurrences than can fit on the terminal operator's screen at one time. The runtime system stores detail occurrences sequentially in the order in which they are created by pageable map commands and divides them into pages based on the number of occurrences that can fit on the terminal operator's screen. One page of occurrences can be displayed on the screen at any one time.

Example of a Pageable Map

For example, a pageable map might display information about a department and list all the employees within the department. The set of map fields related to employee information occurs once for each employee to be listed. These detail occurrences of employee information are created at runtime by pageable map commands and can be displayed to the terminal operator one page at a time.

What's in this Section?

This section discusses the use and definition of pageable maps by presenting the following topics:

- Areas of pageable maps
- Map paging sessions
- Dialog and program operations
- Runtime considerations
- Creating pageable maps

Areas of Pageable Maps

A pageable map is divided into three areas, as illustrated in the following screen:

```
EMPLOYEE PERSONAL DATA


EMPLOYEE NAME:  JANE             FERNDALE
EMPLOYEE NUMBER:  0032              SOCIAL SECURITY NUMBER:  034-56-7890
EMPLOYEE NAME:  TOM              FITZHUGH
EMPLOYEE NUMBER:  0081              SOCIAL SECURITY NUMBER:  112-34-5678
EMPLOYEE NAME:  GEORGE           FONRAD
EMPLOYEE NUMBER:  0045              SOCIAL SECURITY NUMBER:  092-34-7890
EMPLOYEE NAME:  ROBIN            GARDNER
EMPLOYEE NUMBER:  0053              SOCIAL SECURITY NUMBER:  022-34-4444
EMPLOYEE NAME:  JENNIFER         GARFIELD
EMPLOYEE NUMBER:  0003              SOCIAL SECURITY NUMBER:  021-99-4516




PAGE:  0004
```

Three Areas of a Pageable Map

- The **header area** (optional) is a rectangular area located across the top of the screen that contains one or more rows of map fields associated with header information. The header area information is displayed whenever the map is displayed.
- The **footer area** (optional) is a rectangular area located across the bottom of the screen that contains one or more rows of map fields associated with footer information. The footer information is displayed whenever the map is displayed.
- The **detail area** (required) is a rectangular area located across the middle of the screen that contains the *detail occurrence* for the map.

  The set of fields in the detail occurrence is defined in the detail area only once. At runtime, the number of detail occurrences that are displayed in the detail area depends on the space available on the screen after accounting for the header and footer information.

Examples of each Area

For example, a pageable map used to display a department record and all associated employee records might contain the following information:

- **Header area**—The title of the map and department information
- **Footer area**—A message field, a page field to display the current page number, and literal fields with information about how to page through the map
- **Detail area**—Detail occurrences of employee information

What is a Map Page?

The term **map page** refers to a runtime display made up of the header and footer map fields and a page of detail occurrences. The page of occurrences that is displayed at any given time is determined by the value of the system $PAGE field. For example, if a department with 25 employees is displayed on a pageable map that can hold a maximum of 10 employee occurrences, the value in the system $PAGE field determines the occurrences that are displayed:

- **If $PAGE equals 1**, occurrences 1 through 10 are displayed.
- **If $PAGE equals 2**, occurrences 11 through 20 are displayed.
- **If $PAGE equals 3**, occurrences 21 through 25 are displayed.

The value in the $PAGE field can be specified by the operator or by the dialog or program. The current value in the system $PAGE field can be displayed on a map by associating $PAGE with a field on the pageable map that is associated with the system $PAGE field.

Map-Paging Sessions

What is a Map-paging Session?

When a CA ADS dialog or an application program uses a pageable map at runtime, a *map-paging session* takes place. CA ADS dialog or application program commands build and display detail occurrences for map pages during a map-paging session. The operator can update information in data fields and can access different map pages by pressing control keys or specifying page numbers.

The fields in a detail occurrence are defined once when the map is created. At runtime, these fields are repeated as many times on a page as possible to fill the detail area. Each repetition of the detail occurrence represents a record occurrence.

For example, the following screens contrast the definition and run time detail occurrences for a pageable map:

- **At definition time**, the fields in the detail area are associated with dictionary elements or with system-supplied fields.
- **At runtime**, commands in the CA ADS dialog premap process or the application program move stored data to the appropriate work record elements before displaying the map. When the map is displayed, the variable field associated with WK-ID-0415 displays the identification number of a different employee in each detail occurrence. Each detail occurrence displays information about the particular employee identified in the WK-ID-0415 field.

Definition Time

```
EMPLOYEE PERSONAL DATA

EMPLOYEE NAME:  ______________________________1
EMPLOYEE NUMBER: ______2     SOCIAL SECURITY NUMBER: ________3









______________________________________________________________4


PAGE: _____5
```

1. Associated with WK-NAME-0415
2. Associated with WK-ID-0415
3. Associated with WK-SS-NO-0415
4. Associated with system-supplied $Message field
5. Associated with system-supplied $Page field

Runtime

```
EMPLOYEE PERSONAL DATA


EMPLOYEE NAME:  JANE             FERNDALE
EMPLOYEE NUMBER:  0032              SOCIAL SECURITY NUMBER:  034-56-7890
EMPLOYEE NAME:  TOM              FITZHUGH
EMPLOYEE NUMBER:  0081              SOCIAL SECURITY NUMBER:  112-34-5678
EMPLOYEE NAME:  GEORGE           FONRAD
EMPLOYEE NUMBER:  0045              SOCIAL SECURITY NUMBER:  092-34-7890
EMPLOYEE NAME:  ROBIN            GARDNER
EMPLOYEE NUMBER:  0053              SOCIAL SECURITY NUMBER:  022-34-4444
EMPLOYEE NAME:  JENNIFER         GARFIELD
EMPLOYEE NUMBER:  0003              SOCIAL SECURITY NUMBER:  021-99-4516




PAGE:  0004
```

Sequence of Events in a Map-paging Session

1. **The map-paging session is initiated** according to specifications or commands in the CA ADS dialog or application program that uses the pageable map.

   More information

   - For more information about CA ADS statements for pageable maps, see the *CA ADS Reference Guide*.
   - For more information about PL/I statements for pageable maps, see Appendix G, "PL/I DML Statements for Pageable Maps".
   - For more information about Assembler or COBOL statements, see the *CA IDMS DML Reference Guide for Assembler* or the *CA IDMS DML Reference Guide for COBOL*.

2. **Detail occurrences are created** by statements in the dialog or program. Occurrences are stored sequentially in the order that they are created and are divided into pages based on the number of detail occurrences that can fit on the terminal screen at one time. A detail occurrence is displayed on the terminal screen only when the map page to which the occurrence belongs is displayed.

3. **A map page is displayed** by the CA ADS runtime system or the DC runtime system as a result of one of the following:

   - **The first detail occurrence on the second page of occurrences is constructed**. The first page of detail occurrences in a map-paging session is automatically displayed. The CA ADS process or program module that creates detail occurrences continues to execute and can create additional detail occurrences.
   - **A display command is issued** by the dialog (except when the command immediately follows the display of the first page of occurrences, as described previously).
   - **A display is done automatically** when the dialog has no premap process

   The CA ADS runtime system (which executes CA ADS dialogs) and the DC runtime system (that executes application programs) are both referenced by the term **program runtime system** in the remainder of this section. The term **runtime mapping system** refers to a separate part of the runtime system that handles only mapping functions at runtime.

   Note: For more information about display commands for pageable maps, see Building and Displaying Fields (see page 105).

4. **The terminal operator optionally modifies map data fields** in the header or footer areas or in any of the detail occurrences of the current map page. Map-field modifications are subject to restrictions specified at map-definition or map runtime:

   - **At map-definition time**, fields can be protected from operator input.
   - **At map runtime**, modifications can be restricted by specifications made in the CA ADS dialog or application program:
     - The paging mode option (UPDATE/BROWSE) determines whether modifications can be made to a pageable map.
     - Map modification commands determine whether modifications can be made to individual fields.

   Note:

   - For more information about paging mode options, see "Map-Paging Session Options".
   - For more information about modification commands, see "Map Inquiry and Modification".

5. **The terminal operator optionally makes a paging request** to specify the next map page to be displayed in either of the following ways:

   - **By pressing a control key** associated with paging forward or backward one page:

     – <PF8> pages the detail area forward one page.

     – <PF7> pages the detail area backward one page.

     These control key settings are system generation options.

   - **By specifying an integer value** in the page field (if any) on the map.

   - **By typing first or last** over the page field on the map.

     – **First** brings the operator to the first page of detail occurrences.

     – **Last** brings the operator to the last page of detail occurrences.

6. **The terminal operator presses a control key**, initiating the following:

   - **The internal representation of data fields is updated** to reflect changes made by the terminal operator if the operator pressed any key other than <Clear>, <PA1>, <PA2>, <PA3>.

   - **The $PAGE value is updated** if a paging request was made in any of the following ways:

     – The operator pressed a key associated with paging forward or backward

     – The operator specified a value in the page field (if any) on the map

     – The dialog or program set the value of the $PAGE field on the previous mapin operation

   - **The flow of control is determined** by the paging-type option specified for the dialog or program

     Note: For more information about paging-type specifications,see Building and Displaying Fields (see page 105).

7. **Modified detail occurrences are retrieved** by commands in the dialog or program. Data is updated to program variable storage according to specifications made for each field.

   Note: For information about retrieval commands for pageable maps, see Retrieving Modified Data (see page 108).

8. **Detail occurrences are modified** by either the program runtime system or the mapping runtime system, as specified by the paging-type for the map-paging session.

   Note: For information about paging-type options, see "Building and Displaying Fields".

9. **Additional detail occurrences are created** at any time by commands in the dialog or program. New detail occurrences are stored at the end of the set of detail occurrences in the session scratch record.

   Note: For information about commands that create new detail occurrences, see Building and Displaying Fields (see page 105).

10. **A page of the map is displayed** by either the program runtime system or the mapping runtime system, as specified by the paging-type for the map-paging session.

11. **The map-paging session is terminated** according to specifications or commands in the CA ADS dialog or application program that uses the pageable map.

Dialog and Program Operations

Pageable maps are used at runtime by CA ADS dialogs and application programs. Specifications for a dialog or program determine how a pageable map can be used at runtime. For example, options specified for a map-paging session determine whether the operator can page backward or update information on the map.

Specifications and statements in a dialog or program perform the following functions at runtime. These specifications and statements are discussed separately as follows:

- Establish map-paging session options
- Build and display fields
- Retrieve modified data

Note: The CA ADS runtime system (which executes CA ADS dialogs) and the DC runtime system (that executes application programs) are both referenced by the term *program runtime system* in the remainder of this section. The term *runtime mapping system* refers to a separate part of the runtime system that handles only mapping functions at runtime.

Map-Paging Session Options

Putting Options into Effect

CA ADS

CA ADS issues a #STRTPAG and #ENDPAG request for you under the conditions specified in the following table:

Options	**Conditions**
Specified	When a pageable map is associated with a dialog

Options	Conditions
Retained	Across dialogs when *all* of the following conditions are met: ■ The dialog that passes control and the dialog that receives control are associated with the same pageable map. ■ The dialog that passes control and the dialog that receives control are defined with the same map-paging session options. Control is passed by means of a LINK, INVOKE, or RETURN command.
Ended	When *any* of the following conditions are met: ■ The application terminates normally ■ The application aborts ■ The application passes control to another dialog under any of the following conditions: ■ The dialog receiving control is associated with a different pageable map than the one that initiated the map paging session ■ The dialog receiving control has different map paging dialog options than the dialog that initiated the map paging session ■ The dialog that initiated the map paging session issues the TRANSFER command, either by way of a CA ADS PROCESS statement or an EXECUTE NEXT function ■ The dialog that initiated the map paging session returns control to a higher level ■ POP or POPTOP is issued and the menu receiving control is at a higher level than the dialog that started the paging session

Application Programs

With COBOL, PL/1, and Assembler, the program must explicitly issue the paging requests.

Options	Conditions
Specified	In the DML statement that initiates the paging session: ■ By the *COBOL* or *PL/I* STARTPAGE statement ■ By the *Assembler* #STRTPAG macro
Retained	Across program branches if no DML command to initiate a new paging session or terminate the existing session is encountered.

Options	Conditions
Ended	By a DML statement that either explicitly terminates the current paging session or begins a new paging session and implicitly terminates the current paging session: ■ The *COBOL* or *PL/I* ENDPAGE statement explicitly terminates a map-paging session; the STARTPAGE statement (above) implicitly terminates the current map-paging session. ■ The *Assembler* #ENDPAG macro explicitly terminates a map-paging session; the #STRTPAG macro (above) implicitly terminates the current map-paging session.

Specifying Paging and Update Requests

Paging-type Specification

The paging-type specification for a dialog or program determines whether the program or the runtime mapping system handles paging and update requests. Paging and update requests are made when the operator presses a control key:

- A **paging request** to display a different page of the map is made if either of the following cases applies when the operator presses a control key:
  - The control key is associated with paging forward or backward.
  - The value in the $PAGE field has been altered by the operator or the dialog/program that uses the map, and the control key is not <Clear>, <PA1>, <PA2>, or <PA3> (which do not transmit data).
- An **update request** (to update operator modifications to the scratch record for the paging session) is made when the MDT is set on for fields and the operator presses a control key other than <Clear>, <PA1>, <PA2>, or <PA3>.

Three Types of Paging

One paging-type option must be specified for a dialog or program that uses a pageable map. The paging options affect the flow of control when an operator presses a control key during a paging session as described in the following table:

Option	Affect on flow of control
NOWAIT	(default) Specifies that the runtime mapping system automatically handles all paging and update transactions. Control is passed to the program runtime system only when neither an update nor a paging request is made when the operator presses a control key.

Option	Affect on flow of control
WAIT	Specifies that runtime mapping automatically handles paging transactions that do not cause data to be updated. Control is passed to the program runtime system when an update or non-paging request is made.
RETURN	Specifies that the mapping runtime system does not handle any terminal transactions in the paging session. Control is passed to the program runtime system whenever the operator presses a control key.

Note: Runtime mapping does not update program variable storage unless a MAP IN command is issued. In cases where the operator can update data, it is recommended that WAIT or RETURN be specified for the map-paging session so that data can be retrieved as it is updated.

CA ADS automatically handles all MAPIN and MAPOUT commands.

Paging Type	Paging request		Non-paging request	
	No MDT set	Any MDT set **	No MDT set	Any MDT set **
NOWAIT	Runtime mapping displays the requested map page	Runtime mapping displays the requested map page	Control passes to the program runtime system	Runtime mapping redisplays the same map page
WAIT	Runtime mapping displays the requested map page	Control passes to the program runtime system	Control passes to the program runtime system	Control passes to the program runtime system
RETURN	Control passes to the program runtime system	Control passes to the program runtime system	Control passes to the program runtime system	Control passes to the program runtime system

* If <Clear>, <PA1>, <PA2>, or <PA3> is pressed, and that key is not associated with backward or forward paging, refer instead to the Non-paging Request heading.

** If <Clear>, <PA1>, <PA2>, or <PA3> is pressed, refer to the No MDT Set column under the same heading.

How to Specify the Paging-type

The paging-type is specified for a CA ADS dialog or application program as follows:

Language	Option/Clause
CA ADS	■ NOWAIT/WAIT/RETURN option—Map Specifications screen in ADSC ■ PAGING TYPE clause
COBOL or PL/I	NOWAIT/WAIT/RETURN clause—STARTPAGE statement
Assembler	TYPE=NOWAIT/WAIT/RETURN clause—#STRTPAG macro

Note

- For more information on CA ADS specifications for pageable maps, see the *CA ADS Reference Guide*.
- For more information about PL/I statements for pageable maps, see Appendix G, "PL/I DML Statements for Pageable Maps".
- For more information about COBOL or Assembler pageable map statements, see the *CA IDMS DML Reference Guide for Assembler* or the *CA IDMS DML Reference Guide for COBOL*.

Backpaging Capability

Definition

The backpaging specification for a dialog or program determines whether the terminal operator can display a previous map page during a map-paging session. The following considerations apply to the backpaging option:

- *If backpaging is allowed* (default), detail occurrences of previous pages must be retained during the map-paging session.
- *If backpaging is not allowed*, the previous page of detail occurrences is deleted when a new map page is displayed.

How to Enable Backpaging

Backpaging is enabled/disabled for a CA ADS dialog or application program by one of the following specifications:

Language	Option/Clause
CA ADS	BACKPAGE(YES/NO) option—Map Specifications screen in ADSC
COBOL or PL/I	BACKPAGE/NOBACKPAGE clause—STARTPAGE statement

Language	Option/Clause
Assembler	BACKPAG=YES/NO clause—#STRTPAG macro

Note:

- For information about CA ADS specifications for pageable maps, see the CA *ADS Reference Guide*.
- For information about PL/I statements for pageable maps, see CA IDMS DML *Reference Guide for PL/I.*
- For more information about COBOL or Assembler pageable map statements, see the *CA IDMS DML Reference Guide for Assembler* or the *CA IDMS DML Reference Guide for COBOL*.

Paging Mode

Definition

The paging mode specification for a dialog or program determines whether the terminal operator can modify variable map fields:

- *UPDATE* specifies that the terminal operator can modify variable map fields, subject to restrictions specified for the map either at map-definition time or by the dialog or program that uses the map.
- *BROWSE* specifies that the terminal operator can modify only the page and response fields (if any) of the map. The MDTs for variable fields on a map can be set only according to specifications made either in the map-definition or by the dialog or program that uses the map.

Note: UPDATE cannot be specified if backpaging is not allowed and NOWAIT is specified as the paging-type.

How to Specify the Paging Mode

The paging mode is specified for a CA ADS dialog or application program as follows:

Language	Option/Clause
CA ADS	■ UPDATE/BROWSE option—Map Specifications screen in ADSC ■ PAGING MODE clause—BACKPAGE (Yes/No)
COBOL or PL/I	UPDATE/BROWSE clause of the STARTPAGE statement
Assembler	FLAG=UPDATE/BROWSE clause of the #STRTPAG macro

Note:

- For information about CA ADS specifications for pageable maps, see the CA *ADS Reference Guide.*
- For information about PL/I statements for pageable maps, see the appendix "PL/I DML Statements for Pageable Maps".
- For more information about COBOL or Assembler pageable map statements, see either the CA IDMS DML Reference Guide for Assembler or the *CA IDMS DML Reference Guide for COBOL*.

Building and Displaying Fields

Statements in CA ADS dialogs and application programs can be used to create and display header and footer fields during a map-paging session when execution is under dialog or program control. Execution is under dialog or program control when:

- The first page of detail occurrences has not been displayed
- The operator presses a control key that passes control from the mapping runtime system to the program runtime system, as specified by the paging-type specification for the session.

  Note: For information about the paging-type specification, see Building and Displaying Fields (see page 105) earlier in this section.

Building Fields

CA ADS and Data Manipulation Language (DML) statements that build individual detail occurrences from stored values or operator modifications are as follows:

Language	Option/Clause
CA ADS	PUT DETAIL statement
COBOL or PL/I	MAP OUT DETAIL statement
Assembler	#MREQ OUT DETAIL=YES statement

How it Works

The header and footer areas of the map are stored in the scratch area and are built automatically when the first PUT DETAIL command is executed. As a result, any modifications to the header or footer are ignored after that point.

As the program runtime system builds detail occurrences for a pageable map, it stores the occurrences sequentially in the order in which they are created.

More information:

- For information about CA ADS specifications for pageable maps, see the CA *ADS Reference Guide*.
- For information about PL/I statements for pageable maps, see Appendix G, "PL/I DML Statements for Pageable Maps."
- For more information about COBOL or Assembler pageable map statements, see the CA IDMS DML Reference Guide for Assembler or the CA *IDMS DML Reference Guide for COBOL*.

Displaying Fields

If the AUTODISPLAY option is on for the dialog, the program runtime system automatically displays the first page of a pageable map when the first detail occurrence of the second page of occurrences is created. The first page of a pageable map consists of fields in the header and footer areas and the first page of detail occurrences.

The program runtime system continues to build detail occurrences after the first page of the map is displayed, if necessary. After the first page is displayed, control is passed to the terminal operator as described as follows:

- **CA ADS**—Control is passed to the terminal operator when a DISPLAY command is issued after the final detail occurrence is built for the map.
- **COBOL, PL/I,** and **Assembler programs**—Control is passed to the terminal operator immediately after the first page of detail occurrences is displayed. The terminal operator can page through the map while additional detail occurrences are being built. A mapin operation can be initiated after all detail occurrences for a pageable map are built.

Requesting Display of Detail Occurrences

The following table lists the statements that request display of a page of detail occurrences and pass control to the terminal operator:

Language	Option/Clause
CA ADS	DISPLAY statement
COBOL or PL/I	MAP OUT RESUME statement
Assembler	#MREQ OUT RESUME statement

How Statements are used

The previous statements are used in the following ways:

- *To initiate a mapout when mapout operations are under the control of the program runtime system,* as determined by the paging-type option for the paging session.

  Note: For more information, see Building and Displaying Fields (see page 105).

- *To display the first page of a pageable map* (application program only) when fewer detail occurrences are built than can be displayed on the first page of the map. COBOL and PL/I programs issue a status code and Assembler programs return a value when the first page of a pageable map is displayed. If the appropriate status code or value has not been received when the runtime system finishes building detail occurrences for the map, a MAP OUT RESUME or #MREQ OUT RESUME statement must be used to display the first page of the map.

Summary of Commands

Language	Create Single Occurrence	Display Page
CA ADS	PUT DETAIL	The PUT DETAIL statement that creates the first detail occurrence of the second page causes the first page of occurrences to be displayed. A DISPLAY statement causes the specified page of occurrences to be displayed.
COBOL PL/I	MAP OUT DETAIL	The MAP OUT DETAIL statement that creates the first occurrence of the second page causes the first page of occurrences to be displayed. A MAP OUT RESUME statement causes the specified page of occurrences to be displayed.
Assembler	#MREQ OUT DETAIL	The #MREQ OUT DETAIL statement that creates the first occurrence of the second page causes the first page of occurrences to be displayed. A #MREQ OUT RESUME statement causes the specified page of occurrences to be displayed.

Note:

- For information about CA ADS specifications for pageable maps, see the *CA ADS Reference Guide*.
- For information about PL/I statements for pageable maps, see the appendix "PL/I DML Statements for Pageable Maps".
- For more information about COBOL or Assembler pageable map statements, see either the *CA IDMS DML Reference Guide for Assembler* or the *CA IDMS DML Reference Guide for COBOL*.

Retrieving Modified Data

What is Retrieved?

Modified data in fields in the header and footer area for a pageable map can be retrieved on mapin:

- The *CA ADS* runtime system automatically retrieves modified values from the header or footer.
- The *COBOL* or *PL/I* MAP IN HEADER statement is used to retrieve a modified value in either the header or footer area
- The *Assembler* #MREQ MAP IN HEADER=YES statement is used to retrieve a modified value in either the header or footer area

The retrieved value from each modified field (MDT set on) in the header or footer area is updated to program variable storage if DATA is Y (YES) for the field.

The mapping runtime system updates a scratch record when operator modifications are made to fields in detail occurrences but does not update program variable storage. Therefore, CA ADS dialogs and application programs that use pageable maps in the UPDATE paging mode must include statements that update program variable storage when necessary.

CA ADS and the DMLs provide statements that retrieve modified detail occurrences; either all data fields or only those fields in the occurrence for which the MDT is set on can be retrieved. The retrieved value for each modified field (MDT set on) in the detail occurrence is moved to program variable storage if DATA is specified as Y (YES) for the field on mapin.

CA ADS and DML commands that retrieve modified detail occurrences are as follows:

Language	Retrieve from Header/Footer Area	Retrieve from Detail Occurrence
CA ADS	Fields automatically retrieved	GET DETAIL statement

Language	Retrieve from Header/Footer Area	Retrieve from Detail Occurrence
COBOL PL/I	MAP IN HEADER statement	MAP IN DETAIL statement
Assembler	#MREQ IN HEADER=YES statement	#MREQ IN DETAIL=YES statement **Note**: The HEADER specification in each statement retrieves data from both the header and footer areas.

Note:

- For information about CA ADS specifications for pageable maps, see the *CA ADS Reference Guide*.
- For information about PL/I statements for pageable maps, see the appendix "PL/I DML Statements for Pageable Maps."
- For more information about COBOL or Assembler pageable map statements, see the *CA IDMS DML Reference Guide for Assembler* or the *CA IDMS DML Reference Guide for COBOL*.

Runtime Considerations

The detail occurrence for a pageable map cannot occupy more lines than are available in the detail area at runtime.

Size Constraints on Maps

A pageable map can be displayed on screens of varying sizes. The following constraints apply:

- A map cannot be displayed on a screen that is narrower than the map.
- A pageable map can only be displayed on a screen if the runtime detail area is large enough to hold at least one complete detail occurrence.
- The footer (if available) will adjust to the different devices. For example, if the footer starts three lines from the bottom of the screen, the footer will start on:
  - Line 22 of a 24X80
  - Line 30 of a 32X80
  - Line 41 of a 43X80
  - Line 25 of a 27X132

Constraints on System-supplied Fields

The following considerations apply to system-supplied fields on a pageable map:

- A maximum of one *message field* can be defined on a map. If defined, the message field for a pageable map should be defined as follows:
  - *In the header or footer area*, if the most important messages for the map are generated by the ADS DISPLAY MESSAGE command.
  - *In the detail occurrence*, if the most important messages for the map are generated by the ADS PUT DETAIL MESSAGE command. At runtime, the single message field defined in the detail occurrence is mapped out once in each occurrence in the detail area.
- A *page field* can be defined in the header or footer area of a pageable map. The page field is associated with the system $PAGE field when the map is defined by using the mapping facility. At runtime, the page field displays the number of the current map page. If the page field is unprotected, the operator can key a page number, *first*, or *last* in the field to request display of a page.
- A *response field* can be defined in the header or footer area of a pageable map. A response field is meaningful only when the map is used by a CA ADS dialog.

Creating Pageable Maps

Overview

Pageable maps are created by using either the online or batch compiler of the CA IDMS mapping facility. Specifications made during map-definition establish the map as a pageable map and define the following pageable map features:

- Header area (optional)
- Detail area
- Detail occurrence
- Footer area (optional)

Note: For more information about the areas and detail occurrence of a pageable map, see Areas of Pageable Maps (see page 92).

Using the Online Compiler

Use the online compiler to state the specifications for a pageable map as follows:

- Select the *Pageable* option on the first General Options screen
- Specify *the boundaries of the detail area* by using the *Pageable Options* screen in the Field Definition process as described as follows:
  - If the field or literal is the only one in the detail area, select *Option 1*
  - If the field is the first of two or more fields in the detail area, select *Option 2*
  - If the field is the last field or literal in the detail area, select *Option 3*. The detail occurrence will end at the last character position in this field.
  - If the field is the first field or literal in the footer area, select *Option 4*. This defines the end of the detail area and the beginning of the footer.

After the map has been compiled using the Compile action on the Main Menu screen, you can use it in MAPC.

Note: For more information about using the online mapping compiler screens, see "Online Mapping Compiler Reference".

Using the Batch Compiler

The map developer makes pageable map specifications for a map by using clauses of batch compiler statements:

- *PAGEABLE clause of the MAP statement*—Designates that the map is a pageable map
- *DETAIL START clause of the PFLD or MFLD (for MAP AUTOPANEL) statements*—Establishes:
  - *The end of the header area* (if any) on the line immediately above the line that contains the attribute byte for the field assigned the DETAIL START specification
  - *The start of the detail area* on the line that contains the attribute byte of the field being defined
  - *The first field of the detail occurrence* on the line that contains the attribute byte of the field being defined

The field assigned the DETAIL START specification must begin on a new line (that is, it cannot begin on a line that contains characters for a field in the header area).

- *DETAIL END clause of the PFLD or MFLD (for MAP AUTOPANEL) statements*—Establishes that the detail occurrence for the map is to end at the final character position of the current field
- *FOOTER START clause of the PFLD or MFLD (for MAP AUTOPANEL) statement*—This clause is optional. If it is used, it establishes:
  - *The start of the footer area* on the line that contains the attribute byte for the field. The footer area ends at the end of the screen.
  - *The end of the detail area* on the line immediately above the line that contains the attribute byte of the field assigned the FOOTER START specification.

  The field assigned the FOOTER START specification cannot begin on a line that contains characters for a field in the detail occurrence. If assigned, the FOOTER START specification must be assigned to a following field the field assigned the DETAIL END specification.

Considerations

DETAIL START, DETAIL END, and FOOTER START can only be assigned to fields after the owner map has been designated as pageable. Only one pageable map area specification can be made on any given row of the map. For example, the DETAIL END and FOOTER START specifications cannot be assigned to different fields on the same row.

A map is not ready for use until the map utility has been used to generate a map load module for the completed map-definition.

Note: For more information about using the batch compiler, see "Batch Compiler Coding Considerations".

Chapter 6: The Help Facility

This chapter discusses about the Help facility.

This section contains the following topics:

Overview

The help system in the Mapping facility allows you to create help messages for an entire map or a specific field on a map.

Help created using the help system has the following advantages over user-written help:

- Processing of the help request is transparent to the dialog or program that uses the map.
- During a help session, the help system preserves the map attributes and any data that has been entered, but not yet stored, until the help session is over. When the session is over, the system restores that information.
- Help can be added to an existing system without changing any code and without having to recompile any dialogs or programs.

This chapter provides step-by-step instructions on how to create the text and associate it with a map or field as well as a brief overview of how to use the help system.

Note: Physical Terminals (PTERMs) should be defined with the READBUFFER option. PTERMs defined with NOREADBUFFER may not function properly. For more information about the READBUFFER PTERM option, see the *CA IDMS System Generation Guide*.

Terminology

You should be aware of the following definitions before you begin working on the help system:

Term	Definition
Field-level help	Help that applies to a specific field on a map.

Term	Definition
Map-level help	Help that applies to an entire map.
Text module	A module in IDD that contains the text of either a map-level or a field-level help message. Each help message is stored in a separate text module.
Help load module	The load module that contains each map- and field-level help text module associated with a map. There is only one help load module per map.
Half-window format	Help that displays on the half of the screen that does not contain the cursor. With field-level help this allows the user to view simultaneously, the help text and the field in question.
Full-window format	Help that displays across the entire screen.

Creating Map-Level Help

Summary of Steps

To create map-level help, perform the steps listed as follows. The steps are described in detail on the following pages:

- Create and store the text for the message in IDD
- Associate that text with the map using the mapping compiler
- Compile the map
- Optionally, you can test the results using the DC/UCF system task, SHOWMAP.

Creating the Text of the Help Message

Steps

To create and store the help text, perform the following steps:

1. Sign on to IDD
2. Enter the following all at once, as illustrated on the following sample screen:

   ```
   ADD MODULE your-maphelp-text-module LANGUAGE IS HELP MODULE SOURCE FOLLOWS
        The text of your help message
   MSEND.
   ```

   Important: If a line of help text exceeds 72 characters, it will be truncated when it is displayed as help for the map.

3. Save the module

Sample Screen

```
                          IDD 15.0 ONLINE      NO ERRORS      DICT=SYSDICT      1
ADD MODULE MAP-HELP-EXAMPLE LANGUAGE IS HELP MODULE SOURCE FOLLOWS

   THIS IS HELP FOR THE ENTIRE MAP.
MSEND.
```

Associating the Help Text with a Map

Naming Conventions

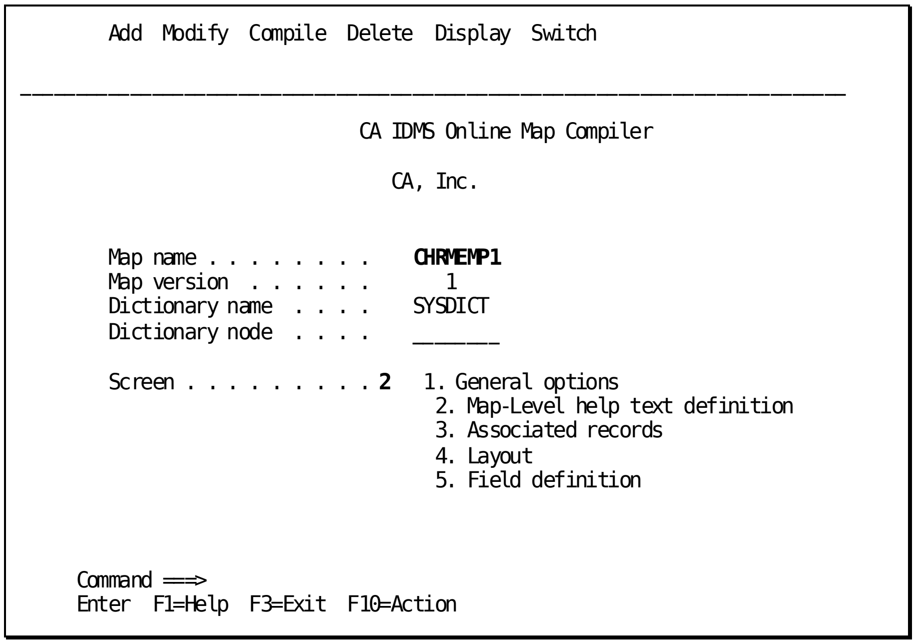
When an application is being migrated at the load module level, both the map module and the help module associated with it must be moved. Therefore, it is suggested that you maintain consistent naming conventions that will make it easy to identify the connection between a map and its associated help load module. In examples provided, the fourth character is used to identify whether the entity is a map (M) or a help load module (H).

Steps

After you create the help text, you must associate it with the appropriate map using either the online map compiler, MAPC, or the batch compiler, RHDCMAP1. An example using MAPC follows:

Note: The following steps assume that the map with which you are associating the help already exists. If not, create the map first following the instructions in "Using the Online Complier".

1. Access MAPC
2. On the main menu, enter the name of the map with which you want to associate the help and select the Map-Level help text definition option, as shown:



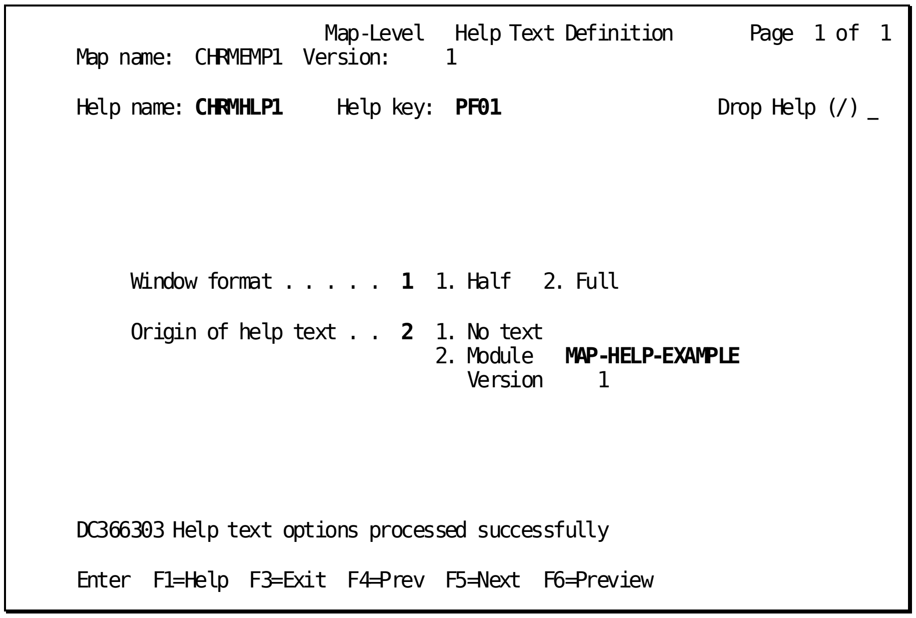
```
       Add  Modify  Compile  Delete  Display  Switch

 ____________________________________________________________

                        CA IDMS Online Map Compiler

                          CA, Inc.


       Map name . . . . . . . .    CHRMEMP1
       Map version  . . . . . .       1
       Dictionary name  . . . .    SYSDICT
       Dictionary node  . . . .    ________

       Screen . . . . . . . . . 2   1. General options
                                     2. Map-Level help text definition
                                     3. Associated records
                                     4. Layout
                                     5. Field definition



     Command ===>
     Enter  F1=Help  F3=Exit  F10=Action
```

3. On the following Map-Level Help Text Definition screen:

   - Enter the name of the load module that contains all the help for the map in the Help Name field.
   - Specify the PF key that will be used to access help for the map.

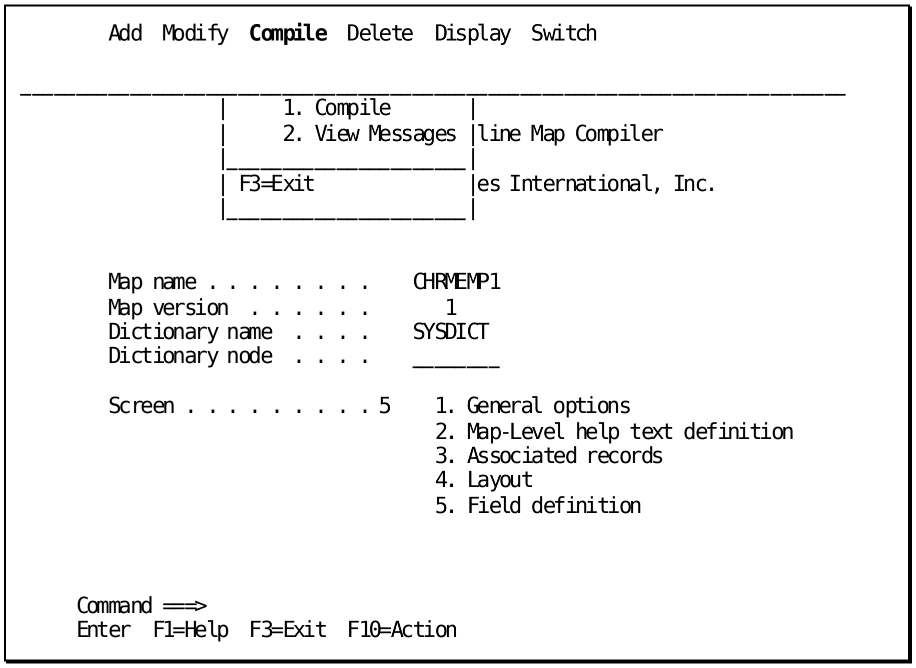
     Important. If a value is not entered in the Help key field, the default is the SYSGEN OLM statement value, which defaults to <PF1>. The PF key value assigned for help supersedes that defined for any other purpose. For example, if <PF1> is defined to a dialog as invoking a response, and <PF1> is also defined as the help key, the help system will get control.

- Specify whether the help should be displayed in a full or half window in the Window format field.
- In the Origin of help text field, specify option 2 and provide the name of the IDD module that contains the help text.
- Press <Enter>.



```
                          Map-Level   Help Text Definition       Page  1 of  1
  Map name:  CHRMEMP1  Version:     1

  Help name: CHRMHLP1     Help key:  PF01                  Drop Help (/) _




       Window format . . . . .  1  1. Half   2. Full

       Origin of help text . .  2  1. No text
                                   2. Module   MAP-HELP-EXAMPLE
                                      Version     1




  DC366303 Help text options processed successfully

  Enter  F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview
```

4. Compile the map using the Compile option on the main menu:



```
      Add  Modify  Compile  Delete  Display  Switch

 _____________________________________________________________________________
                   |     1. Compile        |
                   |     2. View Messages  |line Map Compiler
                   |_______________________|
                   | F3=Exit               |es International, Inc.
                   |_______________________|


      Map name . . . . . . . .    CHRMEMP1
      Map version  . . . . . .       1
      Dictionary name  . . . .    SYSDICT
      Dictionary node  . . . .    ________

      Screen . . . . . . . . . 5    1. General options
                                    2. Map-Level help text definition
                                    3. Associated records
                                    4. Layout
                                    5. Field definition



   Command ===>
   Enter  F1=Help  F3=Exit  F10=Action
```

Testing the Results

Steps

1. After you compile the map, you can test the results using the DC/UCF system task, SHOWMAP, as shown:

```
    V81  ENTER NEXT TASK CODE:
     showmap chrmemp1
```

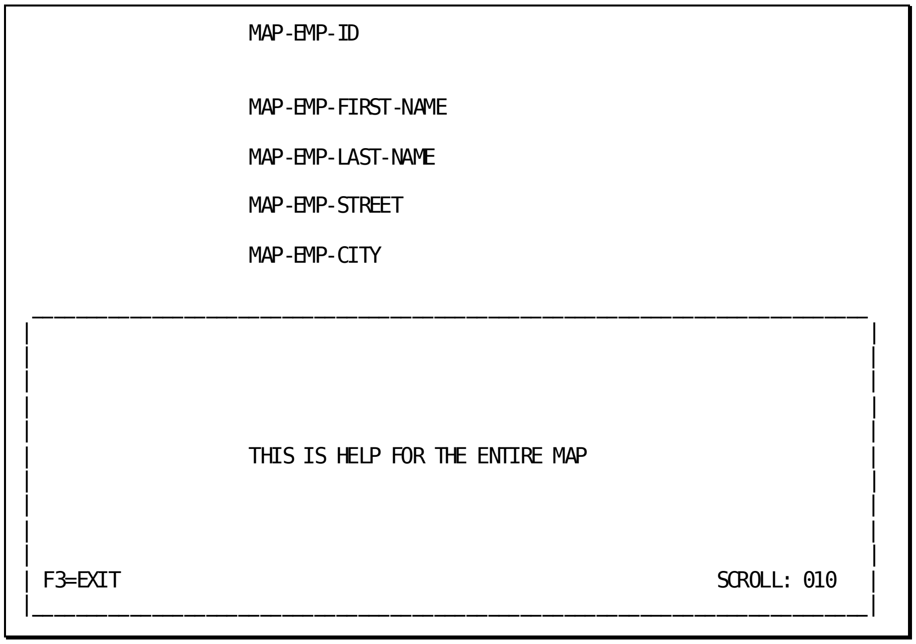
2. When the map displays, place the cursor anywhere on the map and press the PF key defined for help for the map:

```
                     EMP-ID
                                      #   <--{cursor}

                     EMP-FIRST-NAME

                     EMP-LAST-NAME

                     EMP-STREET

                     EMP-CITY

                     EMP-STATE

                     EMP-ZIP-FIRST-FIVE

                     EMP-PHONE

                     STATUS



      NEXT RESPONSE
```

3. Help will be displayed as described as follows:

   ■ If the cursor is on a field for which field-level help has been defined, or is an occurrence of a field for which field-level help has been defined, field-level help is displayed

   ■ For all other fields and other areas of the map, map-level help is displayed



```
                    MAP-EMP-ID


                    MAP-EMP-FIRST-NAME

                    MAP-EMP-LAST-NAME

                    MAP-EMP-STREET

                    MAP-EMP-CITY

 _________________________________________________________________
|                                                                 |
|                                                                 |
|                                                                 |
|                                                                 |
|                  THIS IS HELP FOR THE ENTIRE MAP                |
|                                                                 |
|                                                                 |
|                                                                 |
|                                                                 |
| F3=EXIT                                            SCROLL: 010  |
|_________________________________________________________________|
```

Creating Field-Level Help

Summary of Steps

To create field-level help, perform the following steps which are described in detail on the following pages:

■ Create and store the text for the message in IDD

■ Associate that text with the field using either the online map compiler, MAPC, or the batch compiler, RHDCMAP1. An example using MAPC follows:

  – Access the layout screen

  – Select the appropriate field

  – Use the Field-level Help Text Definition screen to specify the name of the help load module and text module

■ Compile the map

■ Optionally, you can test the results using the DC/UCF system task, SHOWMAP.

Creating the Text of the Help Message

To create and store the help text, perform the following steps:

Steps

1. Sign on to IDD
2. Enter the following:
   `ADD MODULE` *`your-fieldhelp-text-module`* `LANGUAGE IS HELP MODULE SOURCE FOLLOWS`
   `The text of your help message`
   `MSEND.`

   Important: If a line of help text exceeds 72 characters, it will be truncated when it is displayed as help for the map.
3. Save the module

Sample Screen

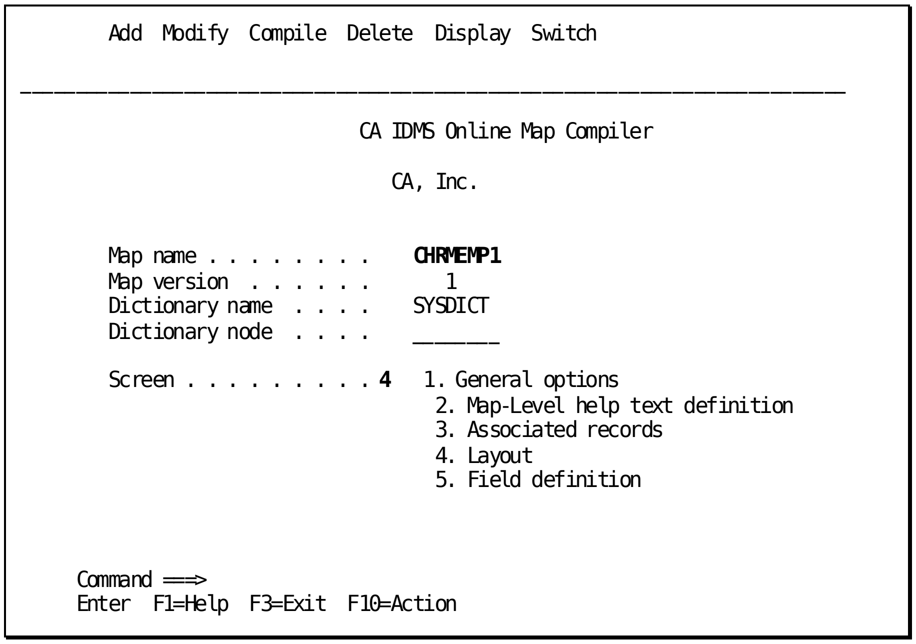
```
                          IDD 15.0 ONLINE     NO ERRORS     DICT=SYSDICT     1
ADD MODULE FIELD-HELP-EXAMPLE LANGUAGE IS HELP MODULE SOURCE FOLLOWS

   THIS IS HELP FOR A SINGLE FIELD
MSEND.
```

Associating the Help Text with a Field

Steps

After you create the help text, you must associate it with the appropriate field using either the online map compiler, MAPC, or the batch compiler, RHDCMAP1. An example using MAPC follows:

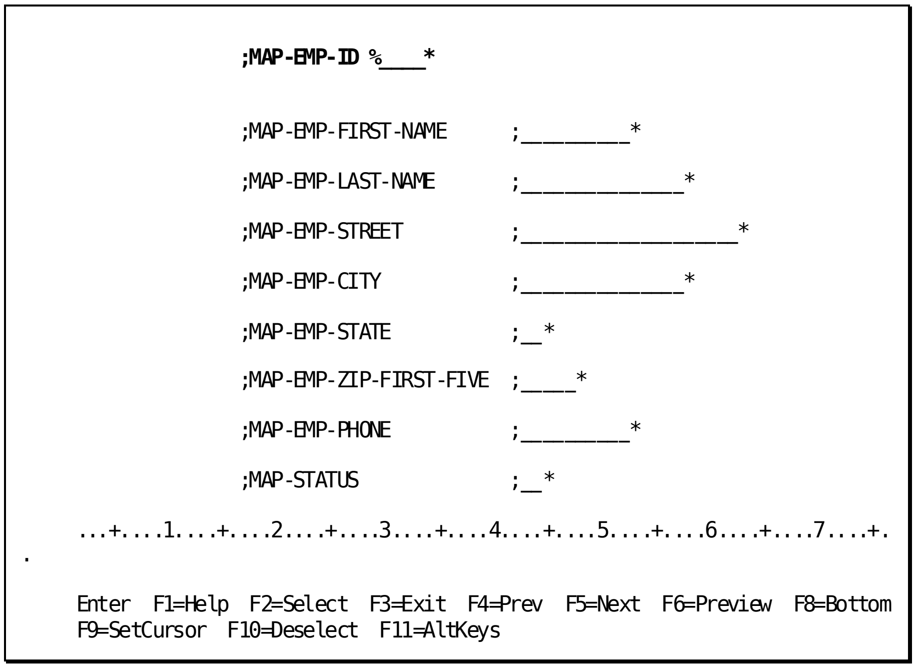
1. Access MAPC
2. On the main menu, enter the name of the map with which you want to associate the help and select the Layout option.



```
        Add  Modify  Compile  Delete  Display  Switch
___________________________________________________________________________
                              CA IDMS Online Map Compiler

                                 CA, Inc.


        Map name . . . . . . . .    CHRMEMP1
        Map version  . . . . . .       1
        Dictionary name  . . . .    SYSDICT
        Dictionary node  . . . .    ________

        Screen . . . . . . . . . 4   1. General options
                                      2. Map-Level help text definition
                                      3. Associated records
                                      4. Layout
                                      5. Field definition



     Command ===>
     Enter  F1=Help  F3=Exit  F10=Action
```

3. On the Layout screen, identify which field you want to associate the help with by overtyping the field mark with a selection character, in this case %, as shown, or by placing the cursor on the selected field and pressing <PF2>:

   Note: If the field is an OCCURS field, you need only define help once; that message will automatically display when the PF key is placed on any occurrence of the field.



```
                  ;MAP-EMP-ID %____*

                  ;MAP-EMP-FIRST-NAME       ;_________*

                  ;MAP-EMP-LAST-NAME        ;______________*

                  ;MAP-EMP-STREET           ;___________________*

                  ;MAP-EMP-CITY             ;______________*

                  ;MAP-EMP-STATE            ;__*

                  ;MAP-EMP-ZIP-FIRST-FIVE   ;____*

                  ;MAP-EMP-PHONE            ;_________*

                  ;MAP-STATUS               ;__*

      ...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.

      Enter  F1=Help  F2=Select  F3=Exit  F4=Prev  F5=Next  F6=Preview  F8=Bottom
      F9=SetCursor  F10=Deselect  F11=AltKeys
```

4. From the layout screen, press <PF5> to move to the Field Definition screen, then overtype the page number with **4** to move to the Field-Level Help Text Definition screen.

   On the Field-level Help Text Definition screen shown:

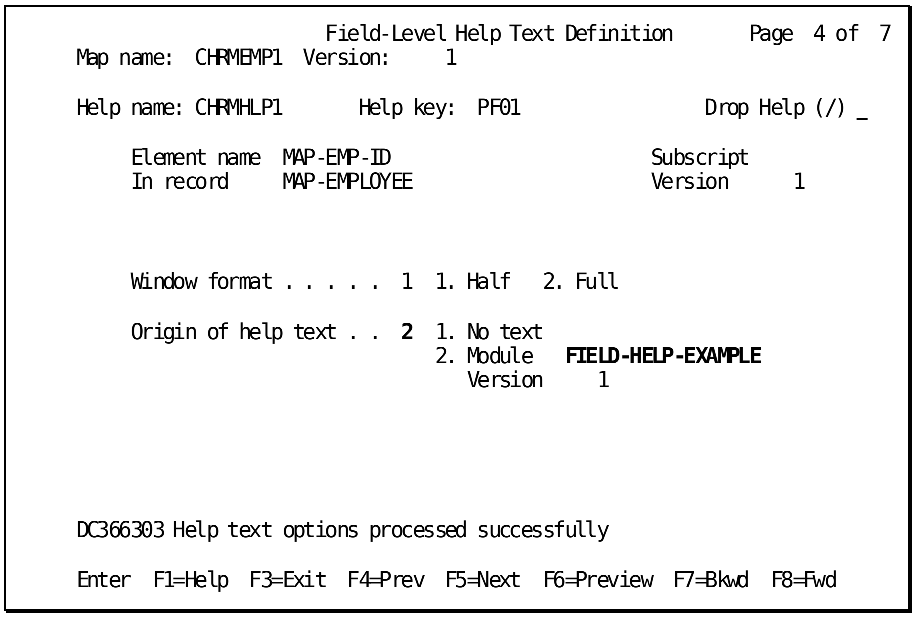
   - Enter the name of the load module that contains all the help for the map in the Help Name field.

     Important: Only one help load module exists per map.

   - Specify the PF key that will be used to access help for the map.

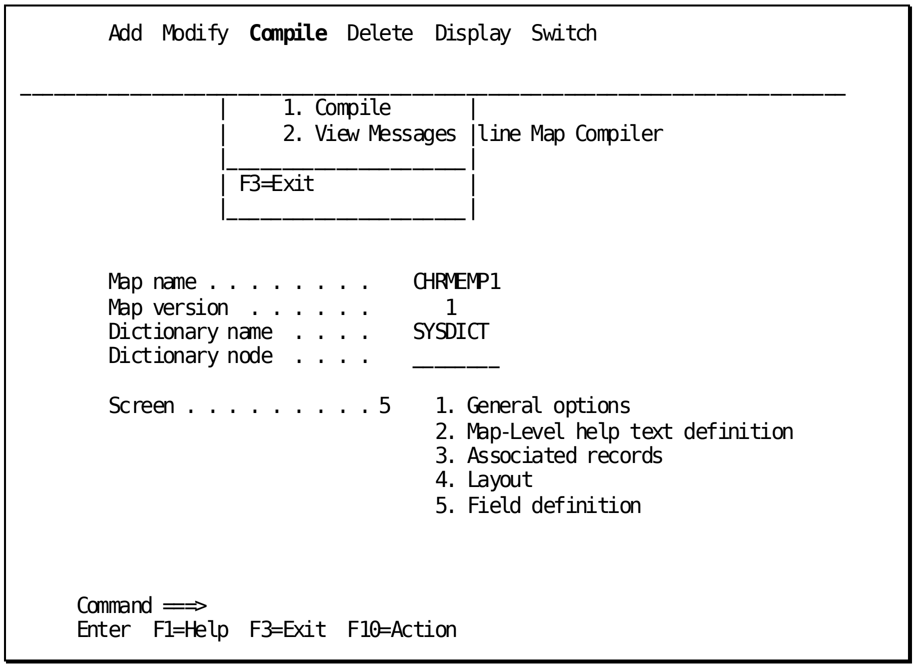
     Important: If a value is not entered in the Help key field, the default is the SYSGEN OLM statement value, which defaults to <PF1>. The PF key value assigned for help supersedes that defined for any other purpose. For example, if <PF1> is defined to a dialog as invoking a response, and <PF1> is also defined as the help key, the help system will get control.

- Specify whether the help should be displayed in a full or half window in the Window format field.
- In the Origin of help text field, specify option 2 and provide the name of the IDD module that contains the help text.
- Press <Enter>.



```
                        Field-Level Help Text Definition      Page  4 of  7
Map name:  CHRMEMP1  Version:     1

Help name: CHRMHLP1      Help key:  PF01               Drop Help (/) _

     Element name  MAP-EMP-ID                        Subscript
     In record     MAP-EMPLOYEE                      Version     1



     Window format . . . . .  1  1. Half   2. Full

     Origin of help text . .  2  1. No text
                                 2. Module   FIELD-HELP-EXAMPLE
                                    Version     1





DC366303 Help text options processed successfully

Enter  F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F7=Bkwd  F8=Fwd
```

5. Compile the map using the Compile option on the main menu:



```
        Add  Modify  Compile  Delete  Display  Switch

_____________________________________________________________________________
                  |     1. Compile        |
                  |     2. View Messages  |line Map Compiler
                  |_______________________|
                  | F3=Exit               |
                  |_______________________|


        Map name . . . . . . . .    CHRMEMP1
        Map version  . . . . . .       1
        Dictionary name  . . . .    SYSDICT
        Dictionary node  . . . .    ________

        Screen . . . . . . . . . 5    1. General options
                                      2. Map-Level help text definition
                                      3. Associated records
                                      4. Layout
                                      5. Field definition



     Command ===>
     Enter  F1=Help  F3=Exit  F10=Action
```

Testing the Results

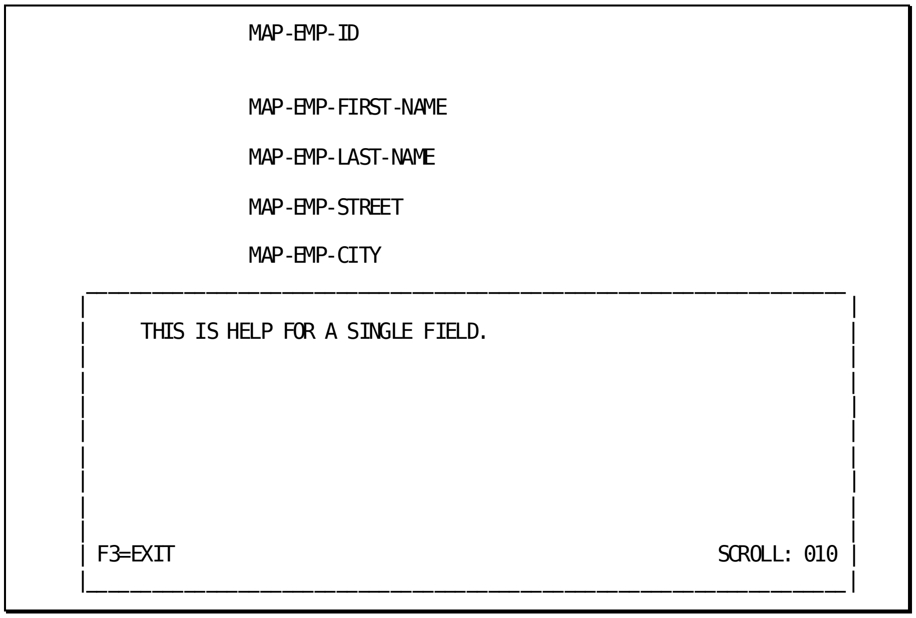
1. After you compile the map, you can see a prototype of the help text by using the DC/UCF system task, SHOWMAP.

```
   V81  ENTER NEXT TASK CODE:
    showmap chrmemp1
```

2. Place the cursor on the field to display the help text associated with it:

```
                    EMP-ID #  <--{cursor}

                    EMP-FIRST-NAME

                    EMP-LAST-NAME

                    EMP-STREET

                    EMP-CITY

                    EMP-STATE

                    EMP-ZIP-FIRST-FIVE

                    EMP-PHONE

                    STATUS



    NEXT RESPONSE
```

3. When you press the PF key associated with help for the map, the help text is displayed for the field:



```
                    MAP-EMP-ID


                    MAP-EMP-FIRST-NAME

                    MAP-EMP-LAST-NAME

                    MAP-EMP-STREET

                    MAP-EMP-CITY
      ______________________________________________________________
      |                                                            |
      |    THIS IS HELP FOR A SINGLE FIELD.                        |
      |                                                            |
      |                                                            |
      |                                                            |
      |                                                            |
      |                                                            |
      |                                                            |
      |                                                            |
      | F3=EXIT                                      SCROLL: 010   |
      |____________________________________________________________|
```

Using the Help System

Overview

To use the help system the user simply presses the key specified for Help at the bottom of the screen.

Foreign Language Support

The Help facility provides a scrolling feature to use if there is more than one page of information.

The following table specifies the commands that can be entered for each language:

Language	Top	First	Bottom	End	Last	Next
English	TOP	FIR	BOT	END	LAS	
	TO	FI	BO	EN	LA	
	T	F	B	E	L	

Language	Top	First	Bottom	End	Last	Next
German	SPI	ERS	BOD		LEZ	NIE
	SP	ER	BO		LE	NI
	I		B		L	N
French		ANE		ZYX		
		AN		ZY		
		A		Z		
Spanish	PRI		ULT	TER		
	PR		UL	TE		
	P		U			

Chapter 7: Runtime Considerations

This chapter discusses about the runtime considerations.

This section contains the following topics:

Overview

This section provides an overview of the use of map load modules by application programs and dialogs. The following topics are presented in this section:

- Mapout and mapin operations
- Map inquiry and modification
- Message field considerations
- Attributes

Note:

- Mapping mode data transmission by CA ADS dialogs is presented in the *CA ADS Reference Guide*.
- Mapping mode of data transmission by application programs is presented in the *CA IDMS DML Reference Guide for COBOL*, *CA IDMS DML Reference Guide for Assembler*, and *CA IDMS DML Reference Guide for PL/I*.
- For a listing of runtime error codes and messages, see the *CA IDMS Messages and Codes Guide*.

Mapout and Mapin Operations

Overview of Activities

The following activities occur on mapout and mapin:

- *On mapout*, data from program storage is transmitted to the terminal if DATA is Y (YES) for mapout:
  - *Literal and variable fields* are both transmitted:
    - The CA ADS runtime system transmits both literal and variable fields when NEWPAGE is specified in the ADSO sysgen statement for CA ADS dialogs.
    - An application program transmits both literal and variable fields when NEWPAGE is specified in the statement that issues the mapout.
  - *Literal fields only* are transmitted by an application program if LITERALS is specified in the DML statement that issues the mapout.
  - *Variable fields only* are transmitted as follows:
    - The ADS runtime system transmits only variable fields when a dialog's map is already displayed as the result of a previous mapout.
    - An application program transmits only variable fields if neither NEWPAGE nor LITERALS is specified in the statement that issues the mapout.
  - *Neither literals nor variables* are transmitted on mapout if there is a field found to be in error.
- *On mapin*, data which has been modified or for which the MDT has been set is transmitted to program variable storage if DATA is Y (YES) for mapin and the field is not found to be in error.

Note: For information about mapout and mapin of pageable maps, see the chapter "Creating Pageable Maps".

CA ADS Dialogs

CA ADS dialogs request mapout and mapin operations as described in the following table:

This operation	Is performed when
Mapout	■ *No premap process is executed* at the beginning of a dialog because: ■ There is no premap ■ Or, the ENTRY POINT was set to MAP *A DISPLAY statement is executed* in a premap or response process
Mapin	The operator presses a control key to initiate an I/O response process in the CA ADS runtime system

Note: For more information about CA ADS mapout and mapin commands, see the *CA ADS Reference Guide.*

Other Languages

IDMS/UCF COBOL, Assembler, and PL/I programs specify mapout and mapin by using CA IDMS DML statements specific to each language:

COBOL and PL/I

This operation	Requests
MAPOUT	The transmission of data from application program storage to the terminal.
MAPIN	A transmission of data from the terminal to application program storage.
MAPOUTIN	A mapout operation followed by a mapin operation.

Assembler

This operation	Requests
#MREQ OUT	A transfer of data from application program storage to the terminal.

This operation	Requests
#MREQ IN	A transfer of data from the terminal to application program storage.
#MREQ OUTIN	A mapout operation followed by a mapin operation.

The MAPOUTIN and #MREQ OUTIN statements are used in conversational (rather than pseudo-conversational) programs.

Note: For more information about DML mapout and mapin commands, see the appropriate *CA IDMS DML Reference Guide for COBOL*, *Assembler*, or *PL/I*.

Map Inquiry and Modification

CA ADS and the CA IDMS Data Manipulation Languages (DMLs) provide statements that allow the application developer to inquire about and modify maps:

Statements

Inquiry Statements

Inquiry statements can be used to examine certain results of a mapin operation, such as whether any data fields have been changed, truncated, or erased.

Modification Statements

Modification statements can be used to modify options specified during map-definition, such as display color or intensity, and to modify the result of a previous mapping operation.

Summary of Statements

The following table lists the statements used to examine mapin results or to make temporary or permanent changes to a map:

Language Statements	Inquiry Statements	Modification Statements
CA ADS	IF statement in conjunction with conditional global variables	Attributes MODIFY MAP
COBOL	INQUIRE MAP	DML MODIFY MAP
PL/I	DML INQUIRE MAP	DML MODIFY MAP

Language Statements	Inquiry Statements	Modification Statements
Assembler	DML #MAPINQ	DML #MAPMOD

Temporary VS Permanent Modifications

The difference between temporary and permanent map modifications is:

- *Temporary modifications* apply to the next mapout operation only.
- *Permanent modifications* remain in effect for all mapout operations for the map until explicitly revoked by one of the following actions:
  - A subsequent, overriding map modification statement is issued for the map at runtime
  - The map control block is reinitialized (as when the map is used by CA ADS dialog or CA IDMS application program)
  - The application program (task) that uses the map terminates
  - The dialog that uses the map either terminates or becomes inactive in the current application thread

Note: If both temporary and permanent modifications are specified, the temporary changes override the permanent changes for the first mapout operation only.

Write Control Characters (WCC)

If CA ADS or DML map modification statement specifies a write control character (WCC) for the map, all WCC options that the map defines are overridden; the following specified default values are used for unspecified WCC options:

Language	Default Value	For this Option
CA ADS	NOMDT	RESETMDT/NOMDT
COBOL	NOKBD	RESETKBD/NOKBD
PL/1	NOALARM	ALARM/NOALARM
	NOPRT	STARTPRT/NOPRT
Assembler	RESETMDT	RESETMDT/NOMDT
	RESETKBD	RESETKBD/NOKBD
	NOALARM	ALARM/NOALARM
	NOPRT	STARTPRT/NOPRT

Message Field Considerations

Displaying Error Messages

A maximum of one message field can be defined on any given map; a message field can be of any length. When several data fields submit incorrect input, the message field displays as many error messages as possible, in order of occurrence (from top to bottom and left to right) of the incorrect data fields.

Messages in the Detail Area

A message field that is defined in the detail area of a pageable map is mapped out once in each occurrence of the detail occurrence in the detail area. Messages generated by the CA ADS DISPLAY MESSAGE statement are not displayed if the message field is defined in the detail occurrence for the map.

Message Sources

A message field displays messages generated from either of the following sources:

- *The error-handling capability* of the CA IDMS mapping facility
- *The CA ADS dialog or application program* that uses the map

If more than one error message is to be displayed in the message field, as many messages are displayed as will fit in the message field. A space character separates messages.

Default Error Message

If an error occurs in a field for which no message has been defined, the default error message is used. The message has the following format:

```
ERROR AT row,column
```

If a Message Field is not Defined

When a message is sent to a map for which no message field is defined, dialogs and application programs react as follows:

- *A CA ADS dialog* displays the message on the default CA ADS message screen (unless the error occurs in a pageable map session, in which case the message is ignored)
- *A CA IDMS application program* ignores the message; processing continues and the operator does not view the message

Attributes

Conflicts

The runtime mapping system does not allow conflicting attributes to be associated with a single map field. If an attribute specified in a map modification command conflicts with a previously established attribute, the new attribute overrides the existing attribute. For example, specification of BRIGHT overrides a previously established DARK attribute; specification of UNDERSCORE overrides a previous REVERSE-VIDEO specification.

How Attributes are Determined

The attributes used for fields are determined as follows:

- *Attributes specified for individual fields* are used for the particular fields when a map is mapped out.
- *Attributes specified for error-handling* override the attributes specified for data fields on a map when the map is redisplayed with error-handling attributes:
  - *Correct field attributes* are used for all data fields that contain correct input and for fields that were not edited on mapin.
  - *Incorrect field attributes* are used for all data fields that contain input errors.

The actual attributes for individual fields and for error-handling are determined at runtime:

- *Temporary modifications* specified for the given field by CA ADS or DML statements take priority over any contradicting attributes previously specified for the field.
- *Permanent modifications* made by CA ADS or DML statements override any contradicting attributes previously defined for the field.
- *Attributes defined for the field by the online mapping compiler or the batch compiler* are used when neither temporary nor permanent modifications override them.

Neither temporary nor permanent modifications alter the map definition or the map load module.

Chapter 8: Online Compiler Overview

This chapter discusses about the online compiler overview.

This section contains the following topics:

Overview

Using the online compiler, developers can define, compile, modify, and delete CA IDMS maps in an online environment. The online compiler performs the following functions:

- *Requests user specifications* for defining, copying, modifying, and deleting map-related entity occurrences through a series of screens
- *Constructs and stores map entity occurrences*, based on the map developer specifications
- *Compiles or deletes map load modules*

Although the online compiler performs most operations that are available through the batch compiler and utility, there are still specific reasons to use the batch compiler and utility. For example, a developer would use the batch compiler to generate device groupings for the map and the batch utility to decompile a map.

The online compiler can modify maps that were created by either the online compiler or the batch compiler and batch utility (unless the map defines device groupings, since maps for multiple devices cannot be simultaneously represented by the online compiler).

Note: For more information about the batch compiler and utility, see the chapter “Batch Compiler and Batch Utility Overview".

What's in this Chapter?

This chapter discusses:

- How to initiate an online compiler session
- What the screens look like
- How to use the action bar
- How to use the function keys

Accessing the Online Compiler

From CA IDMS

Specify the appropriate CA IDMS task code for your site, for example, MAPC. Task codes are defined at system generation and can vary from site to site.

Directly from Another Task

If the task is executing under the transfer control facility, Specify the appropriate CA IDMS task code (for example, MAPC) in conjunction with the SWITCH activity

From the SWITCH Pull Down Menu

Specify the task code of the task to which you want to transfer.

The SWITCH facility enables the map developer to transfer directly from one CA IDMS task to another. For example, the developer can transfer between the online compiler, the CA ADS application generator, the CA ADS dialog generator, and online IDD. When control is transferred from a task, the current session of that task is suspended if necessary. A task can have several suspended sessions.

Information specified for a map is maintained in a queue record during an online session. When a new session is initiated, as when the online compiler is invoked from CA IDMS, a new queue record is built. When a suspended session is invoked, the queue record from the suspended session is used.

Note: For more information about how to use the transfer control facility, see the *CA IDMS Common Facilities Guide*.

Using the Online Compiler

Overview

Online compiler screens prompt a map developer for information about a map and, in some cases, are used to specify a course of action during the map definition process.

What Screens are Used?

The following six primary screens can be accessed during an online compiler session. The Main Menu displays automatically when the compiler is invoked, but the other screens must be invoked either from the Main Menu screen or from each other. Additionally, some of the screens have more than one page on which you can enter additional information:

- The *Main Menu screen* establishes basic information about the map such as name and version number.

  The Main Menu screen also contains the *action bar* which can be used instead of entering a command on the command line. The developer can use the action bar to initiate an add, copy, modify, display, delete, or compile of a map, or to move to another task.

- The *General Options screens* (two pages) establish the options that apply to the map such as the type of map, display and print options, and attributes for re-displayed fields.

- The *Map-level Help Text screen* establishes the connection between the map and the IDD module that contains the help text.

- The *Associated Records screen* establishes the records associated with the map.

  The Autopaint option is also accessed from this screen.

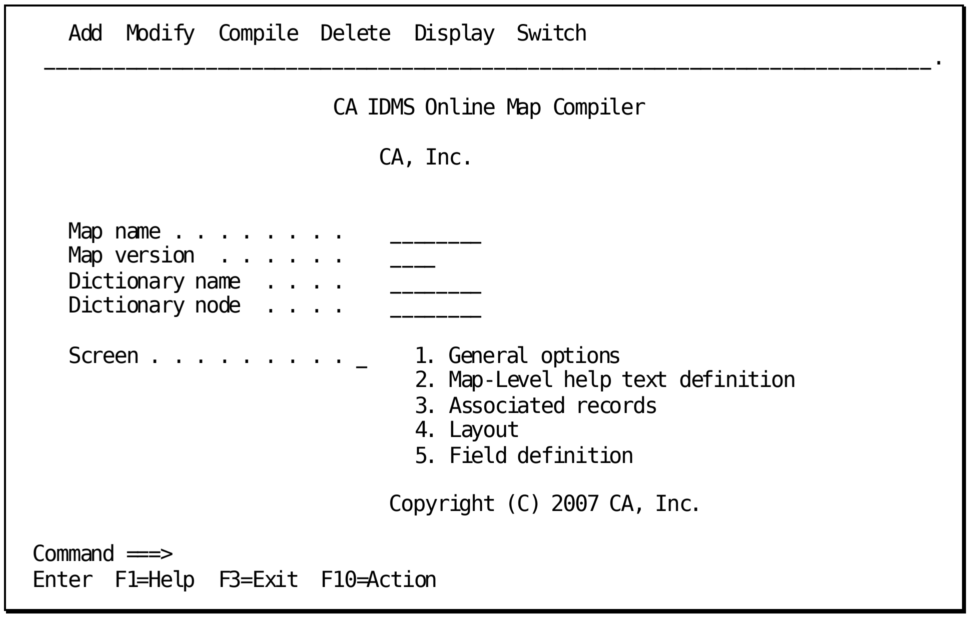
- The *Layout screen* is used to position map fields on the map.

- The *Field Definition screen* establishes field-specific options. Information is collected using seven separate pages as follows:

  - Field Definition
  - Map Read/Write Options
  - Additional Edit Criteria
  - Field-level Help
  - Device-dependent Options
  - User-defined Edit Modules
  - Pageable Options

Using the Main Menu Screen

When a developer signs on to the online compiler, the first screen displayed is the *Main Menu* screen as shown. This screen is used both to provide basic information about and to initiate action on a map.


```
   Add  Modify  Compile  Delete  Display  Switch
 ____________________________________________________________________________.

                           CA IDMS Online Map Compiler

                               CA, Inc.


   Map name . . . . . . . .    ________
   Map version  . . . . . .    ____
   Dictionary name  . . . .    ________
   Dictionary node  . . . .    ________

   Screen . . . . . . . . . _    1. General options
                                 2. Map-Level help text definition
                                 3. Associated records
                                 4. Layout
                                 5. Field definition

                                Copyright (C) 2007 CA, Inc.

Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

How are the Areas Used?

The three areas of the screen are used as follows:

- The *action bar* at the top of the screen can be used as an alternative to entering a command on the command line. The developer can use the action bar to initiate an add, modify, display, compile, or delete of a map or to switch to another task.

  When a developer selects an action, a pull-down window displays. The window contains options related to the action.

  Note: For more information about the action bar, see "Using the action bar (see page 139)".

- The *selection area* in the middle of the screen prompts the developer to supply basic information related to the map and to specify which screen should be displayed next.

- The *command line* close to the bottom of the screen can be used instead of the action bar to perform a particular action. Rather than moving the cursor to the action bar, the developer can enter a command beside the arrow.

- The *function keys* at the bottom of the screen are the keys the developer can use to move from screen to screen within the compiler as well as to display help and map images.

Using the Action Bar

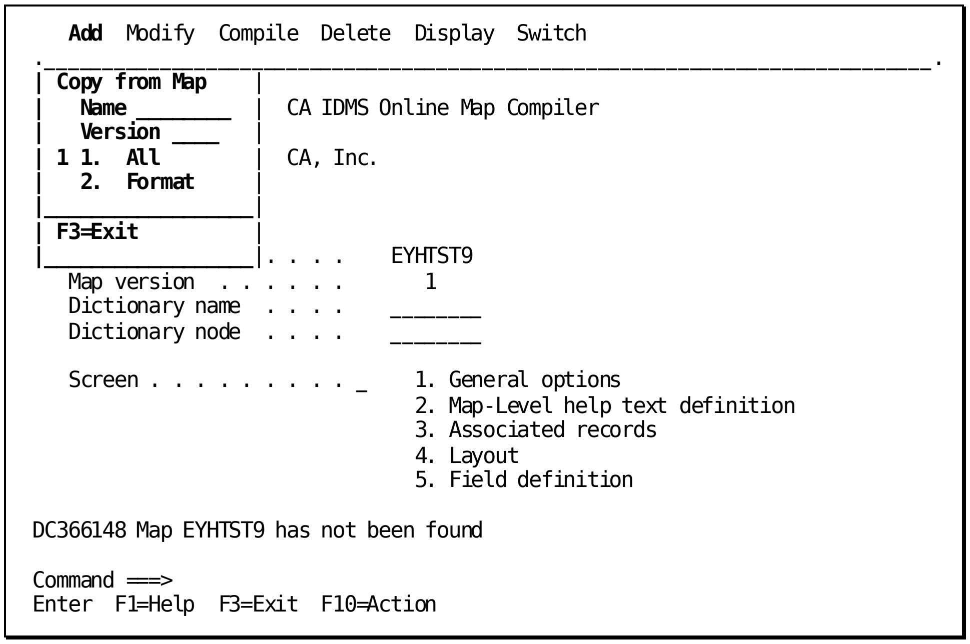
What's in this Section?

This section explains the actions that can be performed using the action bar at the top of the Main Menu screen. It presents each pull-down window and describes how to use it.

Using the Defaults

The first entry on a pull down menu is always the default. To select the default, press [Enter].

ADD



```
   Add  Modify  Compile  Delete  Display  Switch
 .___________________________________________________________________________.
 | Copy from Map    |
 |   Name ________  |  CA IDMS Online Map Compiler
 |   Version ____   |
 | 1 1.  All        |  CA, Inc.
 |   2.  Format     |
 |__________________|
 | F3=Exit          |
 |__________________|. . . .    EYHTST9
   Map version  . . . . . .        1
   Dictionary name  . . . .     ________
   Dictionary node  . . . .     ________

   Screen . . . . . . . . . _    1. General options
                                 2. Map-Level help text definition
                                 3. Associated records
                                 4. Layout
                                 5. Field definition

 DC366148 Map EYHTST9 has not been found

 Command ===>
 Enter  F1=Help  F3=Exit  F10=Action
```

How to use the Window

- Before accessing the ADD function, you must supply the name of the map you are creating and optionally the version number. If a version number is not entered, the dictionary-set default will be displayed.
- If you want to copy an existing map, enter the name and version of that map.
- If you are copying a map, indicate if you want to copy the layout and the records and elements associated with the original map (1) or just the layout (2).
- To confirm your choice, press [Enter].

MODIFY

```
   Add  Modify  Compile  Delete  Display  Switch
 .____________________________________________________________________________.
      |     1. Checkout        |
      |     2. Release         | DC Online Map Compiler
      |     3. List Checkout   |
      |________________________|
      | F3=Exit                | CA, Inc.
      |________________________|


   Map name . . . . . . . .    EYHTST9
   Map version  . . . . . .       1
   Dictionary name  . . . .    ________
   Dictionary node  . . . .    ________

   Screen . . . . . . . . . _    1. General options
                                 2. Map-Level help text definition
                                 3. Associated records
                                 4. Layout
                                 5. Field definition



 Command ===>
 Enter  F1=Help  F3=Exit  F10=Action
```

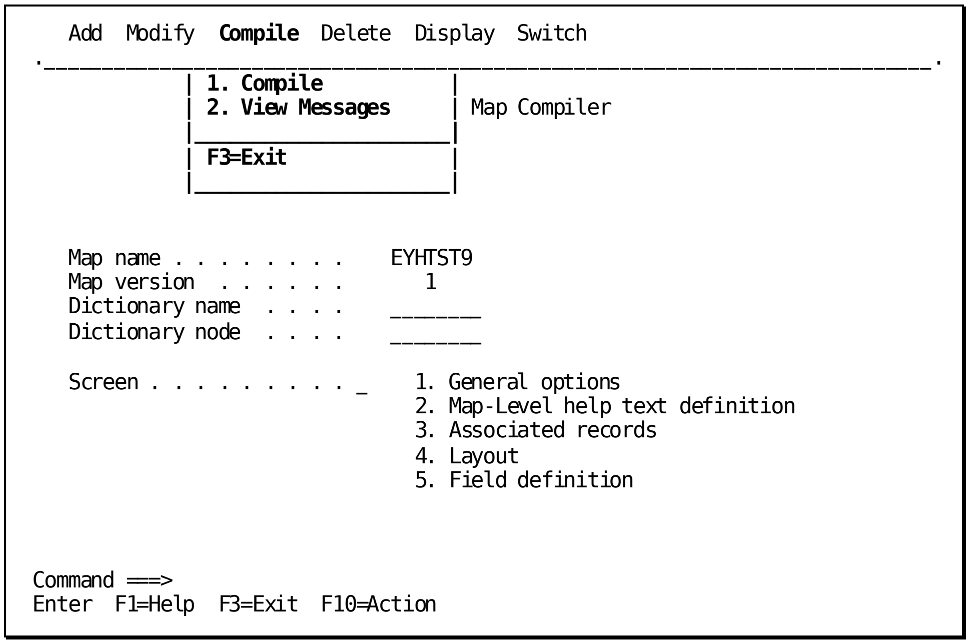
How to use the Window

The MODIFY action controls the *checkin/checkout* procedures of the online compiler as follows:

- **Checkout**—Allows the developer working on a map to have sole access to it. The map is protected from additional updates until it has been checked in.
- **Release**—Releases the developer's hold on the map and allows updates by other developers. If changes have been made since the map was checked out, and the map has not been re-compiled, a warning message will be displayed stating that the changes exist in a work file.

- **List Checkout**—Displays a list of all maps that are checked out to the user ID that is signed on. For each map, it identifies the following:
  - Map name
  - Which version of the map is checked out
  - Which dictionary the map belongs to

COMPILE

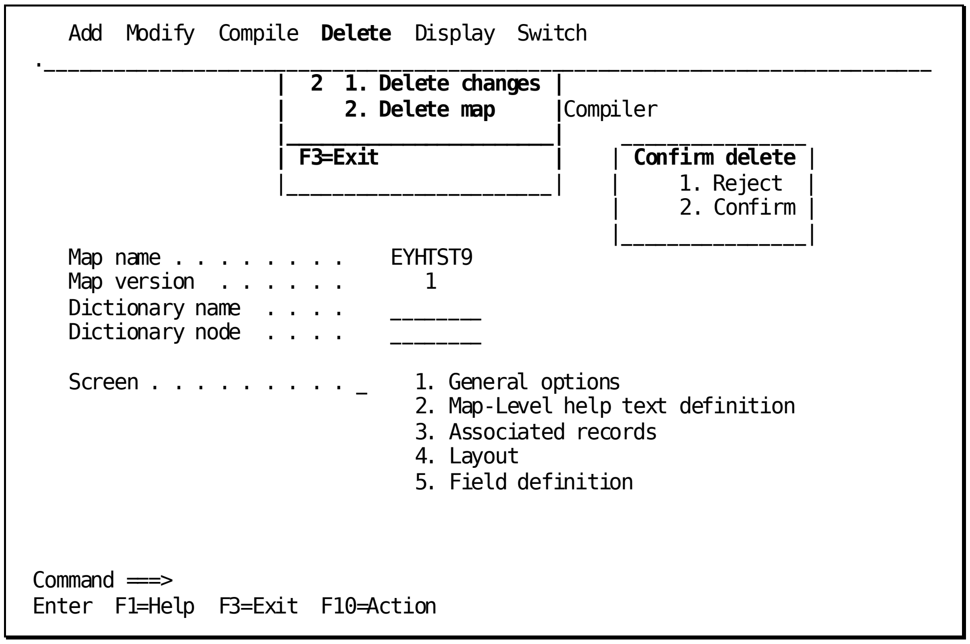


```
   Add  Modify  Compile  Delete  Display  Switch
 .______________________________________________________________________________.
              | 1. Compile            |
              | 2. View Messages      | Map Compiler
              |_______________________|
              | F3=Exit               |
              |_______________________|


   Map name . . . . . . . .    EYHTST9
   Map version  . . . . . .       1
   Dictionary name  . . . .    ________
   Dictionary node  . . . .    ________

   Screen . . . . . . . . . _    1. General options
                                 2. Map-Level help text definition
                                 3. Associated records
                                 4. Layout
                                 5. Field definition



 Command ===>
 Enter  F1=Help  F3=Exit  F10=Action
```

How to use the Window

From this window, you can either compile a map or view the messages from a previous compile.

DELETE

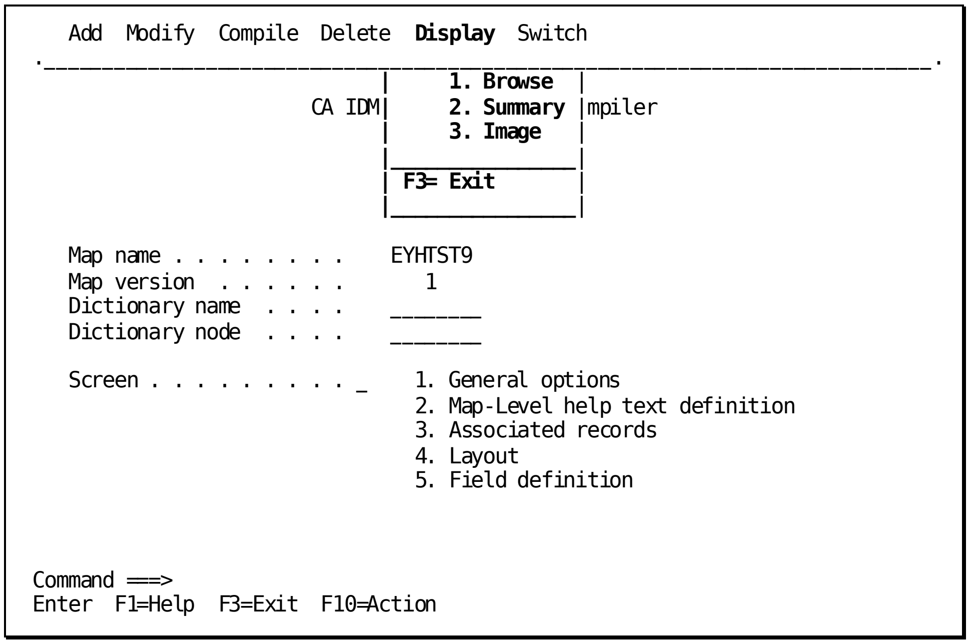


```
   Add  Modify  Compile  Delete  Display  Switch
.____________________________________________________________________________
                     |  2  1. Delete changes |
                     |     2. Delete map     |Compiler
                     |_______________________|     ________________
                     | F3=Exit               |    | Confirm delete |
                     |_______________________|    |     1. Reject  |
                                                  |     2. Confirm |
                                                  |________________|
   Map name . . . . . . . .    EYHTST9
   Map version  . . . . . .       1
   Dictionary name  . . . .    ________
   Dictionary node  . . . .    ________

   Screen . . . . . . . . . _    1. General options
                                 2. Map-Level help text definition
                                 3. Associated records
                                 4. Layout
                                 5. Field definition


Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

How to use the Window

You can either delete the map entirely or delete only the changes made since the last compile. As a safety measure, if you choose to delete the map completely, a second window will be displayed that prompts you to confirm the delete.

DISPLAY



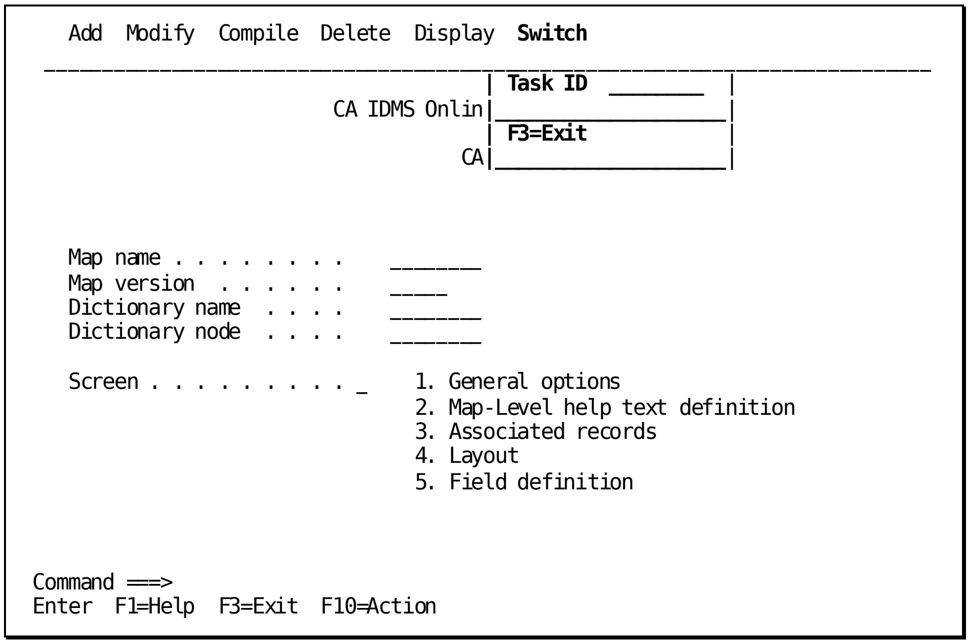
```
   Add  Modify  Compile  Delete  Display  Switch
._____________________________________________________________________________.
                             |     1. Browse   |
                       CA IDM|     2. Summary  |mpiler
                             |     3. Image    |
                             |_________________|
                             | F3= Exit        |
                             |_________________|

   Map name . . . . . . . .    EYHTST9
   Map version  . . . . . .       1
   Dictionary name  . . . .    ________
   Dictionary node  . . . .    ________

   Screen . . . . . . . . . _    1. General options
                                 2. Map-Level help text definition
                                 3. Associated records
                                 4. Layout
                                 5. Field definition



Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

How to use the Window

You can display information about a map in any of three ways:

- **Browse**—Displays all the screens associated with a map. The information pertains to the map as it was last compiled.
- **Summary**—Displays a one-page summary report of the map containing vital statistics on the map such as when it was created, updated, compiled, and who created and modified it.
- **Image**—Displays the map as it would look to an end-user. If the user is currently modifying the map, IMAGE includes the *uncompiled* modifications.

SWITCH



```
   Add  Modify  Compile  Delete  Display  Switch
 _____________________________________________________________________________
                                       | Task ID  ________  |
                         CA IDMS Onlin|____________________|
                                       | F3=Exit            |
                                    CA|____________________|



   Map name . . . . . . . .    ________
   Map version  . . . . . .    _____
   Dictionary name  . . . .    ________
   Dictionary node  . . . .    ________

   Screen . . . . . . . . . _    1. General options
                                 2. Map-Level help text definition
                                 3. Associated records
                                 4. Layout
                                 5. Field definition



 Command ===>
 Enter  F1=Help  F3=Exit  F10=Action
```

How to use the Window

From this window you can leave the online compiler and move to another CA IDMS task by entering the task code in the Task ID field.

Overview of a session

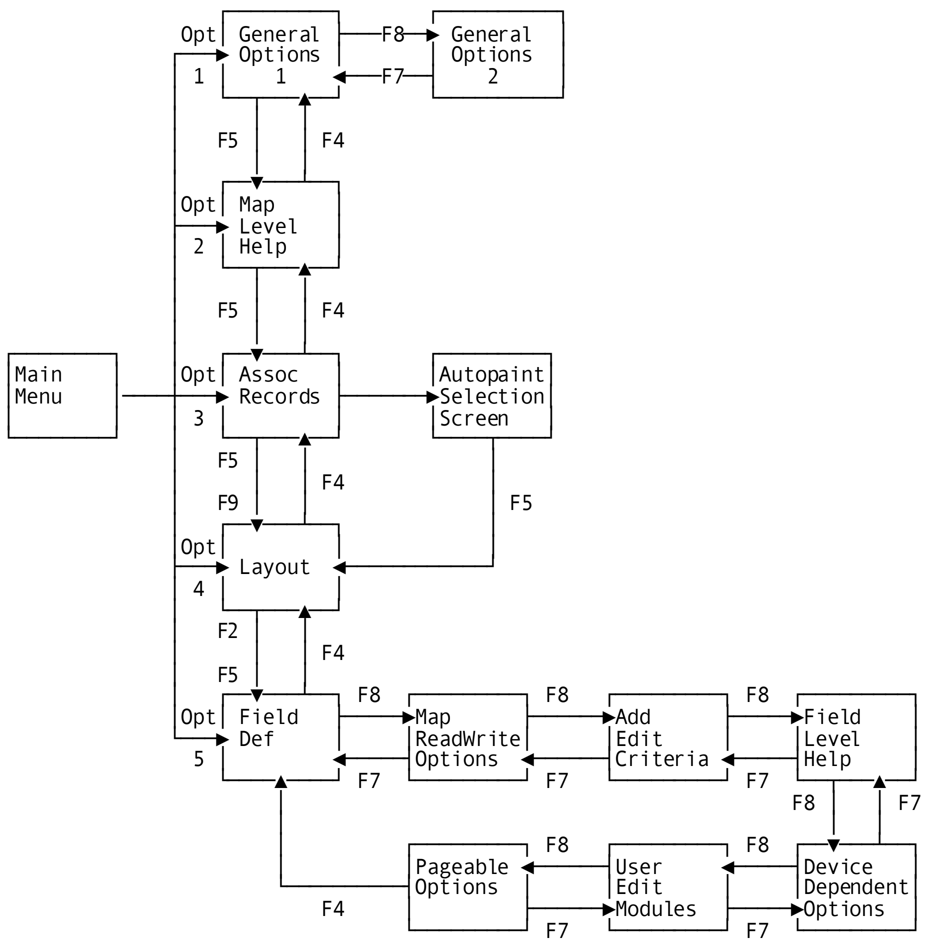
The following figure provides an overview of the online compiler screens and the relationships among them.

Note: For more information about how to use the function keys, see Using the function keys.

In the drawing,

- **Opt n** refers to the screen options on the Main Menu screen
- **Fn** refers to the function key used to move between the screens Online Mapping Compiler Reference.

Note: For more information about each screen, see Using the function keys.



Notes

- When using the Field Definition screens (Option 5), you can move between pages by overtyping the page number at the top of the screen.
- If you are providing information about a literal, you can only use the Field Definition and Pageable Options screens.

Using the function keys

Use

To move from screen to screen or from page to page within a screen, you must use the function keys. Function keys are displayed at the bottom of each screen and may vary depending on the screen.

Most keys have the same function through out the MAPC compiler; <F1> is help, <F4> returns to the previous function screen, etc. Some keys keys have different functions in the Layout screen. For example, <F10> resets all fields selected for edit. The following is a list of the *main* keys and their function within the Layout screen. In addition, *alternate keys* are used to tailor the layout of the screen and consequently, are displayed only on the Layout screen. To toggle between the main keys and the alternate keys, press <PF11> (AltKeys).

Main keys

Key	Name(s)	Function
F1	Help	Displays help information for the function
F2	Select	Identifies a field on which further action will be taken
F3	Exit	Processes the information and returns to the previous function
F4	Prev	Returns to previous function as listed on the main menu screen
F5	Next	Moves to next function as listed on the main menu screen
F6	Preview	Displays the map layout as it would look to the end user
F7	Top	Displays the top of the map layout
F8	Bottom	Moves to the bottom of the map layout
F9	SetCursor	Sets initial cursor position on the map
F10	Deselect	Resets all selected fields for edit so they won't be edited

Key	Name(s)	Function
F11	AltKeys	Toggles between the Main and the Alternate function keys which are used on the Layout screen

Alternate Keys

Key	Name(s)	Function
F1	Help	Displays help information for the function
F2	Mark	Identifies a field on which further action will be taken
F3	Copy	Copies the marked field or block to the location of the cursor
F4	Move	Moves the marked field or block to the location of the cursor
F5	Delete	Deletes the marked field or block
F6	Preview	Displays the map layout as it would look to the end user
F7	Top	Displays the top of the map layout
F8	Bottom	Moves to the bottom of the map layout
F9	Propagate	Copies a field on every line to the cursor
F10	ClrMark	Erase(s) mark from fields
F11	MainKeys	Toggles between the Main and Alternate function keys

How to Move, Copy, and Delete Text

The following information gives detailed instructions on how to use the Move, Copy, and Delete alternate function keys on a field, a line, and on blocks of lines.

Move Key (F4)

- To move a field:
  1. Cursor to the field to be moved
  2. Press Mark (F2)
  3. Cursor to the new position
  4. Press Move (F4)

- To move a line:
  1. Press Mark (F2) twice on the line to be moved
  2. Cursor to the new line position
  3. Press Move (F4)
- To move a block of lines:
  1. Press Mark (F2) on the top and bottom lines of the text to be moved
  2. Cursor to the top line of the new position
  3. Press Move (F4)

Copy Key (F3)

- To copy a field:
  1. Cursor to the field to be copied
  2. Press Mark (F2)
  3. Cursor to the new position
  4. Press Copy (F3)
- To copy a line:
  1. Press Mark (F2) twice on the line to be copied
  2. Cursor to the new line position
  3. Press Copy (F3)
- To copy a block of lines:
  1. Press Copy (F3) on the top and bottom lines of the text to be copied
  2. Cursor to the top line of the new position
  3. Press Copy (F3)

Delete Key (F5)

- To delete a field:
  1. Cursor to the field to be deleted
  2. Press Mark (F2)
  3. Press Delete (F5)

- To delete a line:
  1. Press Mark (F2) twice on the line to be deleted
  2. Press Delete (F5)
- To delete a block of lines:
  1. Press Mark (F2) on the top and bottom lines of the text to be deleted
  2. Press Delete (F5)

Chapter 9: Online Mapping Compiler Reference

This chapter discusses about the online mapping compiler reference.

This section contains the following topics:

Overview

This section describes each screen used to create a map. The screens are presented in the order in which they typically would be used:

- Main Menu
- General Options - Two screens
- Map-Level Help
- Associated Records
- Layout
- Field Definition
  - Map Read/Write Options
  - Additional Edit Criteria
  - Field-Level Help
  - Device-Dependent Options
  - User-Defined Edit Modules
  - Pageable Maps

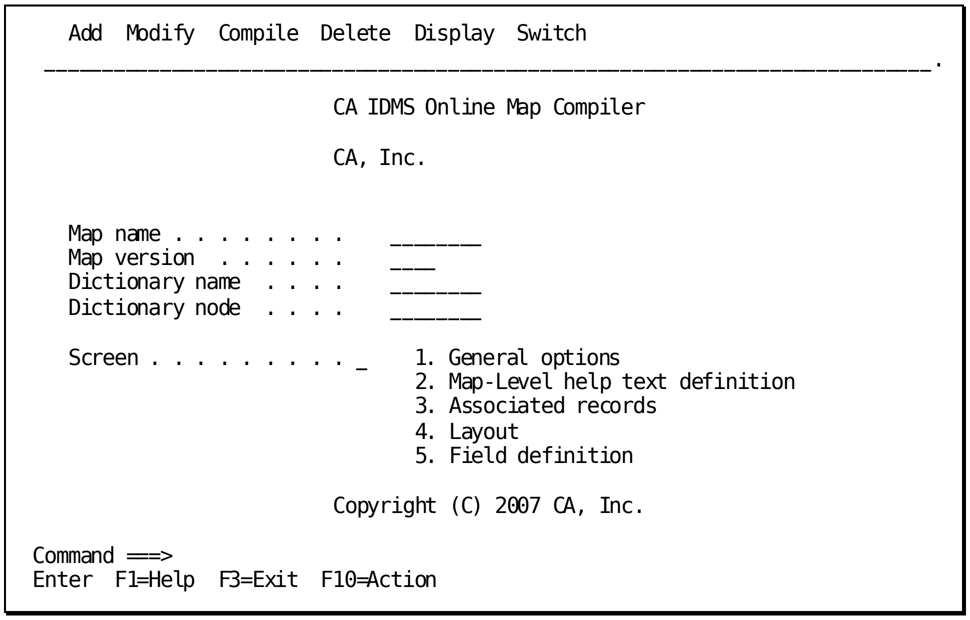
The Main Menu Screen

Description

The Main Menu is the first screen displayed in a session. It is used to provide basic information about the map such as the name and version number and to initiate the map definition session.

After a map has been added, it also can be deleted or compiled from this screen.

Sample Screen



```
   Add  Modify  Compile  Delete  Display  Switch
 _____________________________________________________________________________.
                         CA IDMS Online Map Compiler

                         CA, Inc.


   Map name . . . . . . . .    ________
   Map version  . . . . . .    ____
   Dictionary name  . . . .    ________
   Dictionary node  . . . .    ________

   Screen . . . . . . . . . _    1. General options
                                 2. Map-Level help text definition
                                 3. Associated records
                                 4. Layout
                                 5. Field definition

                         Copyright (C) 2007 CA, Inc.

Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

Field Descriptions

Map name	1 - 8 character name of the map being defined, modified, deleted, or compiled. Map name: ■ Must begin with an alphanumeric or national character such as a pound sign (#), at sign (@), or dollar sign ($) ■ Cannot contain embedded period or blanks
Map version	The version number of the map being defined; must be in the range 1 - 9999. Default is the data dictionary default version number as defined in the DDDL SET OPTIONS statement.

Dictionary name	The dictionary used to store and retrieve the map and load modules. When you sign on, the dictionary is the one specified in your user profile if there is one. If a dictionary is not specified in your profile, the primary dictionary for the CA IDMS system or node is the default. In either case, the dictionary name can be overridden by issuing the DCUF SET DICTNAME command.
Dictionary node	The node name of the dictionary in the distributed database system network. The developer uses the DICTNODE option to specify the location of the dictionary and the name of the node that controls the load area where the map load module is stored. Default is the node specified in the most recently issued DCUF SET DICTNODE command in the current CA IDMS session. If no DCUF SET DICTNODE command has been issued, the local node is the default node. Changing the dictionary name or node modifies the mode for the current session.
Screen	A list of the screens that can be accessed to enter more information about the map.

Notes:

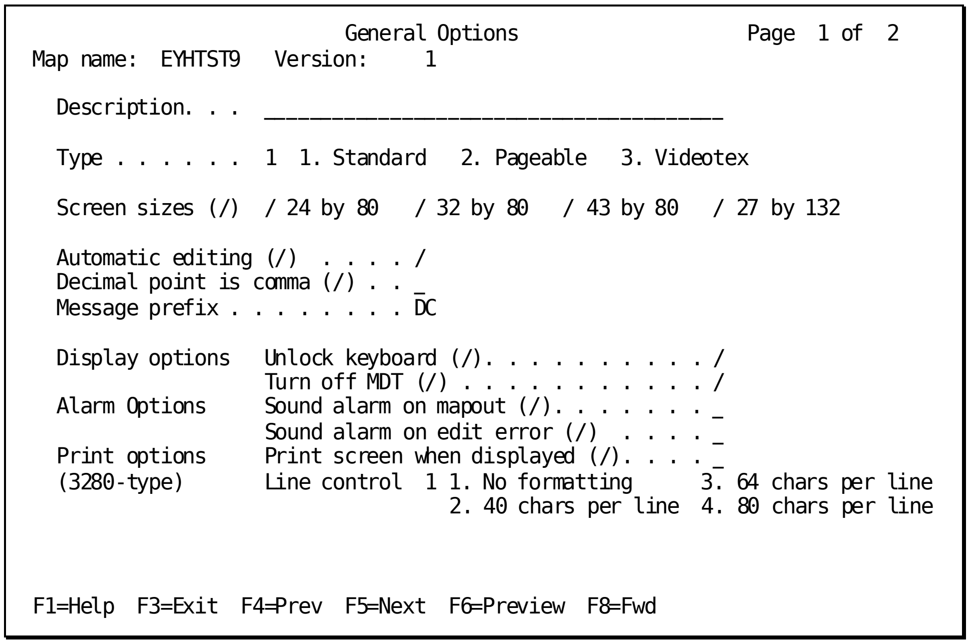
- For more information about the action bar, see "Using the Action Bar".
- For more information about the function keys, see "Using the Function Keys".

General Options—Page 1

Description

This screen is the first of two screens used to enter general information about the map. Information entered here includes the title of the map, device information, display and print options as well as indicators for automatic editing, decimal point handling, and the message prefix.

Sample Screen



```
                          General Options                   Page  1 of  2
Map name:  EYHTST9   Version:    1

  Description. . .  ________________________________________

  Type . . . . . .  1  1. Standard   2. Pageable   3. Videotex

  Screen sizes (/)  / 24 by 80   / 32 by 80   / 43 by 80   / 27 by 132

  Automatic editing (/)  . . . . /
  Decimal point is comma (/) . . _
  Message prefix . . . . . . . . DC

  Display options   Unlock keyboard (/). . . . . . . . . . /
                    Turn off MDT (/) . . . . . . . . . . . /
  Alarm Options     Sound alarm on mapout (/). . . . . . . _
                    Sound alarm on edit error (/)  . . . . _
  Print options     Print screen when displayed (/). . . . _
  (3280-type)       Line control  1 1. No formatting     3. 64 chars per line
                                    2. 40 chars per line  4. 80 chars per line



F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F8=Fwd
```

Field Definitions

Description	The title of the map. This is for documentation purposes only.
Type	The type of map. For example, a standard one-page map, a pageable map, or a videotex map. Videotex refers to the French display devices that are connected to the national phone system.

Screen sizes	The terminal screen sizes on which the map can be used. At least one must be selected; a maximum of four can be selected. The default device specifications are determined by the screen size of the device on which the map is being defined. Device specifications must be specified when you create the map; they cannot be changed online. To change the specifications, you must use the batch compiler and utility. **Note:** For more information about using the batch compiler, see Compiler Action Verbs (see page 194).
Automatic editing	Indicates whether automatic editing and error-handling are enabled for the map: ■ **/** (default)—Globally enables automatic editing and error-handling for the map. ■ **Blank**—Globally disables automatic editing and error-handling for the map; editing and error-handling criteria (if any) defined for map fields are ignored. **Note:** For more information about enabling and disabling automatic editing, see "Enabling Automatic editing and error handling."
Decimal point is comma	Specifies the character to be used as the decimal point for numeric fields on the map: ■ **/**—Specifies that the comma (,) character is used as the decimal point in numeric fields, in accordance with international format. An external picture for the field also must comply with international format, with the comma as the decimal point. ■ **Blank**—Specifies that the period (.) character is used as the decimal point in numeric fields. The default setting for the *Decimal point is comma* prompt is determined at system generation.
Message prefix	Defines the prefix for messages for the map.

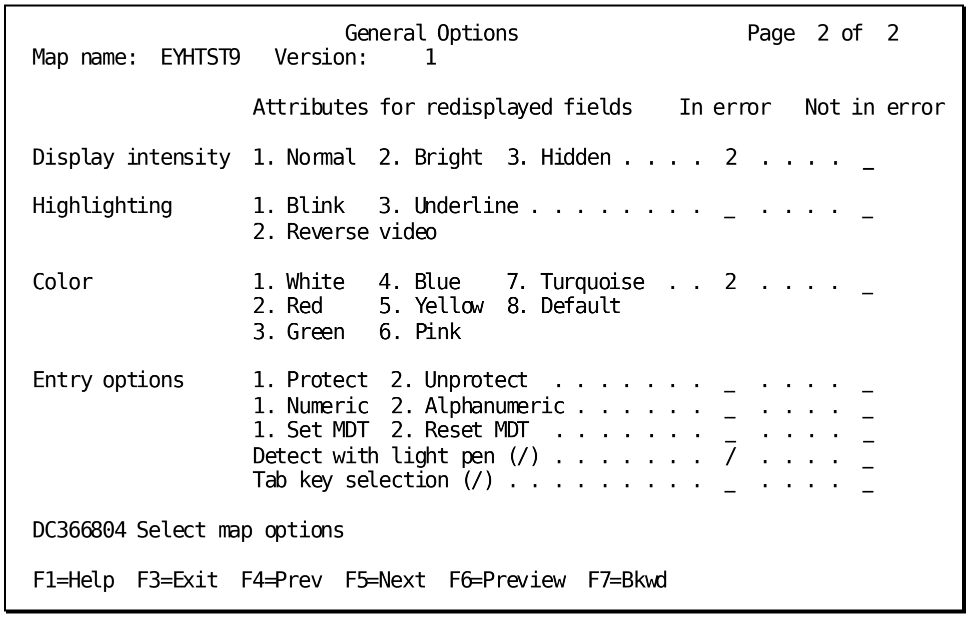
Display options	**Unlock keyboard**—Specifies whether the keyboard is unlocked after a mapout operation: ■ **/** (default)—Specifies that the keyboard is unlocked. ■ **Blank**—Specifies that the keyboard remains locked until the operator presses the RESET key. **Turn off MDT**—Specifies whether modified data tags (MDTs) for data fields are reset when the map is mapped out: ■ **/** (default)—Specifies that all MDTs are reset (turned off) when the map is mapped out. ■ **Blank**—Specifies that MDTs are left unchanged when the map is mapped out. The *Set modified data tag* specification for individual fields on the Map Read/Write Options screen overrides this field.
Alarm options	**Sound alarm on mapout**—Specifies whether the terminal alarm sounds when the map is mapped out: ■ **/**—Specifies that the alarm is sounded. This specification is meaningful only if the terminal is equipped with a hardware alarm. ■ **Blank** (default)—Specifies that the alarm is not sounded. **Sound alarm on edit error**—Specifies whether the terminal alarm sounds when the map contains edit errors: ■ **/**—Specifies that the alarm is sounded. This specification is meaningful only if the terminal is equipped with a hardware alarm. ■ **Blank** (default)—Specifies that the alarm is not sounded.
Print options (3280-type)	**Print screen when displayed**—Specifies whether a 3280-type printer prints the screen on a mapout: ■ **/**—Specifies that the printer starts printing on mapout. ■ **Blank** (default)—Specifies that the printer does not print. **Line control**—Defines line control for 3280-type printers. ■ No formatting—Specifies that new-line characters are used in the data stream. ■ 40 chars per line—Specifies that data is divided into 40-character lines. ■ 64 chars per line—Specifies that data is divided into 64-character lines. ■ 80 chars per line—Specifies that data is divided into 80-character lines.

General Options—Page 2

Description

This screen is the second of two screens used to enter general information for the map. This screen is used to specify attributes for fields that are redisplayed during an error cycle.

Sample Screen



```
                          General Options                   Page  2 of  2
Map name:  EYHTST9   Version:     1

                   Attributes for redisplayed fields    In error   Not in error

Display intensity  1. Normal  2. Bright  3. Hidden . . . .  2  . . . .  _

Highlighting       1. Blink   3. Underline . . . . . . . .  _  . . . .  _
                   2. Reverse video

Color              1. White   4. Blue    7. Turquoise  . .  2  . . . .  _
                   2. Red     5. Yellow  8. Default
                   3. Green   6. Pink

Entry options      1. Protect  2. Unprotect  . . . . . . .  _  . . . .  _
                   1. Numeric  2. Alphanumeric . . . . . .  _  . . . .  _
                   1. Set MDT  2. Reset MDT  . . . . . . .  _  . . . .  _
                   Detect with light pen (/) . . . . . . .  /  . . . .  _
                   Tab key selection (/) . . . . . . . . .  _  . . . .  _

DC366804 Select map options

F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F7=Bkwd
```

Field Definitions

Note: For the following fields, specify the options for fields in error and those not in error.

Display intensity	Specifies whether the field is displayed and whether a displayed field appears at normal or bright intensity: *Normal*—Specifies that the field is displayed at normal intensity. *Bright*—Specifies that the field is displayed at brighter than normal intensity; the field appears highlighted on the screen. *Hidden*—Specifies that the field is not displayed. Data written to the field from program storage or entered by the operator is not visible. Password fields are often attributed the Hidden attribute.

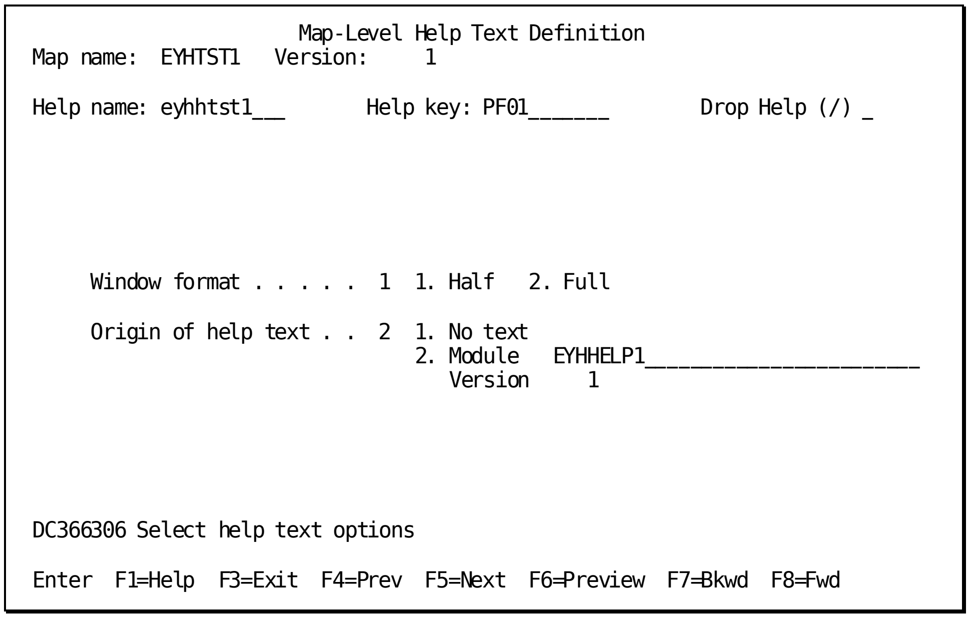
Highlighting	For 3279 machines only: specifies whether the field blinks, is displayed in reverse video, or is underlined.
Color	Specifies the color of the field, or of the background if Reverse video is specified for the field. The developer can only specify one display color. Default specifies that the default display color for the terminal is used. Attributes other than default take effect only when the map is displayed at a 3279-type terminal.
Entry options	■ *Protect/Unprotect*—Identifies whether the field is protected from data entry (1) or open to data entry (2). ■ *Numeric/Alphanumeric*—Specifies whether the field is numeric (1) or alphanumeric (2). ■ *Set MDT/Reset MDT*—Specifies whether the modified data tag should be set automatically during a mapout operation or only when the contents of the field are altered by the terminal operator. ■ *Detect with light pen*—Specifies whether the field can be detected with a light pen. ■ *Tab key selection*—Specifies whether the operator can use &tab. to move the cursor to the field at runtime. When this option is on, NOSKIP and UNPROTECTED are specified for a field.

Map-level Help Text Definition

Description

This screen is used to associate help text previously defined in IDD with a map. You can also specify what type of window is used to display help.

Sample Screen


```
                       Map-Level Help Text Definition
Map name:  EYHTST1   Version:     1

Help name: eyhhtst1___      Help key: PF01_______      Drop Help (/) _





     Window format . . . . .  1  1. Half   2. Full

     Origin of help text . .  2  1. No text
                                 2. Module   EYHHELP1________________________
                                    Version     1




DC366306 Select help text options

Enter  F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F7=Bkwd  F8=Fwd
```

Field Definitions

Help name	The name of the help load module that holds all the compiled help text associated with the map. The name must be a unique entity name.
Drop Help	Used to indicate if the help should be disassociated from the map.
Window format	A code used to indicate if the help text is displayed in a half (1) or full (2) window.
Origin of help text	Specify the name of the IDD module that contains the help text for this map.

Associated Records

Description

This screen is used to enter the schema or work records to be used by the map, and optionally specifies role names for records.

Accessing the Autopaint Feature

The autopaint feature which lets you create a map automatically, is also initiated from this screen.

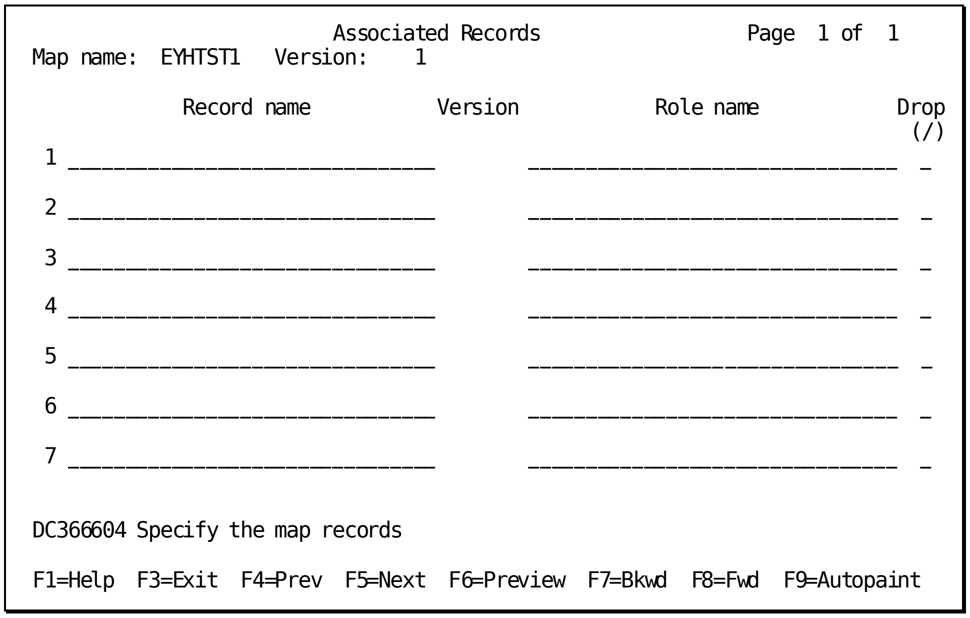
To use the autopaint feature:

- Enter the record information and press <F9>.

  The autopaint feature displays a screen that lists each record and its elements.
- Select the elements you want to be included on the map.
- To view the newly created screen, use the Layout screen.

Note: For more information about the Autopaint feature, see "A Sample Session".

Sample Screen



```
                               Associated Records                 Page  1 of  1
Map name:  EYHTST1  Version:    1

            Record name              Version            Role name              Drop
                                                                                (/)
 1 ______________________________        ______________________________  _

 2 ______________________________        ______________________________  _

 3 ______________________________        ______________________________  _

 4 ______________________________        ______________________________  _

 5 ______________________________        ______________________________  _

 6 ______________________________        ______________________________  _

 7 ______________________________        ______________________________  _


DC366604 Specify the map records

F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F7=Bkwd  F8=Fwd  F9=Autopaint
```

Field Definitions

Record name	Identifies the names of records that contain elements referenced by the map. If a logical record is used, the developer names all of the records containing elements that are part of the logical record and that are used in the map-definition. The name of the logical record is later specified in the dialog or program using the map.
Version	Identifies the version number of each specified record. If not specified, the default version is the value in the IDD OPTION FOR DICTIONARY DEFAULT FOR EXISTING VERSION. If the IDD EXISTING VERSION option is not set, the default version is 1. The developer can use the version field to specify a different version of a record for an existing map. Overstriking the field with a new version number deletes the old record version from the map. The new record version is added to the map. All map fields associated with elements in the old record are associated with the corresponding elements in the new record, by element name. If the new record version omits any elements associated with the previous record version, the related map fields are deleted from the map. If an external picture has been explicitly defined for a map field, changing the record version number does not change the external picture of the element, even if the internal picture is different in the new record version.
Role name	Specifies a role name that is to be used for the record at runtime. Role names typically are used when a given record type is to be used in more than one context. For example, the developer might specify the EMPLOYEE record layout twice for a map that uses the EMPLOYEE record for both employee-related and manager-related fields on a single map: ■ One specification of the EMPLOYEE record would not include a role name for the record. ■ The second specification of the EMPLOYEE record would include a valid role name for the record (for example, MANAGER). The role name must be used in subsequent references to the record in the map-definition. A role name can be established in either of the following two ways: ■ It can be previously defined for the record in the subschema used by the dialog or program. The online compiler will not verify the subschema role name; it must be provided by the user at map-definition. ■ It can be unique to the map, established at map-definition by specifying it on this screen.

Drop	Allows the developer to disassociate the the selected record or role name from the map. To disassociate a record from a map, type a slash (**/**) beside the record or role name. All map data fields are disassociated from the map when it is dropped.

Layout

Description

This screen is used to format the map. If the Autopaint feature has been used, the layout is automatically displayed. Otherwise, the screen will be blank.

Effects of Screen Size on Map Design

Developing or modifying a map on a device that is larger than the smallest screen size specified on the first page of the General Options screen, can present certain problems.

If you try to define a field on the Layout screen outside of the boundaries of the smallest device specifications size, the Mapping facility indicates the error by discarding the misplaced fields and echoing the Layout screen. When the errors are corrected, you can proceed through the map-definition sequence.

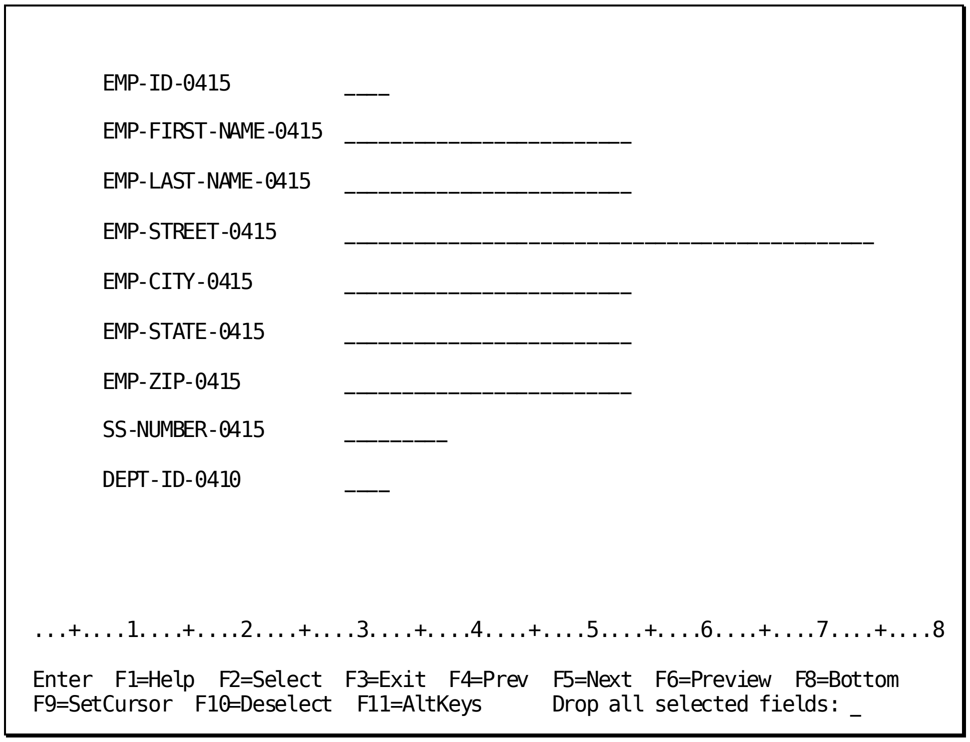
For example, a map that is to be displayed on a 24X80 screen size can be modified on a 32X80 terminal. However, if you try to establish any fields in rows 25 through 32, the online compiler discards these fields and echo the Layout screen.

How to Change the Map

If you want to rearrange the fields or alter them in any way, press <F11> to display the alternate keys. The alternate keys are used to maneuver the fields.

Note: For more information about what function each alternate key performs, see "Using the Function Keys."

Sample Screen



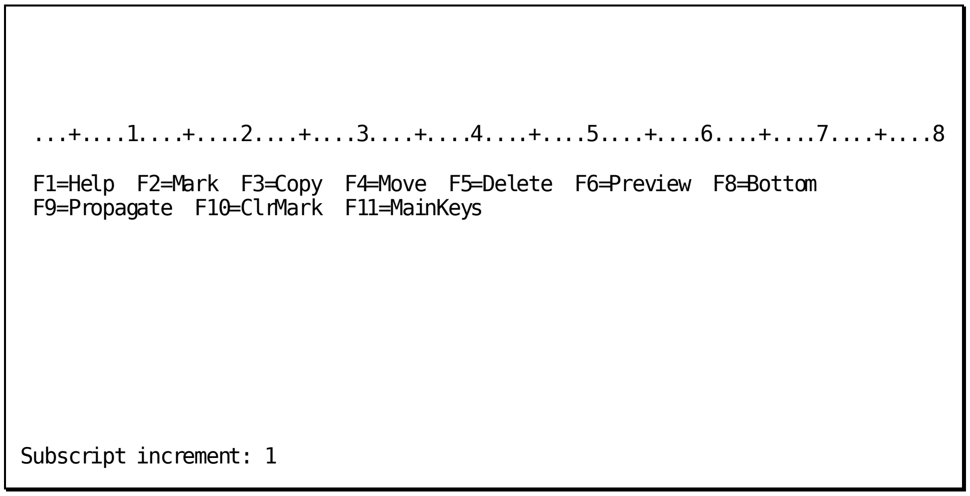
```
      EMP-ID-0415          ____

      EMP-FIRST-NAME-0415  ________________________

      EMP-LAST-NAME-0415   ________________________

      EMP-STREET-0415      ____________________________________________

      EMP-CITY-0415        ________________________

      EMP-STATE-0415       ________________________

      EMP-ZIP-0415         ________________________

      SS-NUMBER-0415       _________

      DEPT-ID-0410         ____



...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8

Enter  F1=Help  F2=Select  F3=Exit  F4=Prev  F5=Next  F6=Preview  F8=Bottom
F9=SetCursor  F10=Deselect  F11=AltKeys      Drop all selected fields: _
```

To Delete All Selected Fields

Position the cursor at the beginning of the field to be marked for deletion. Press <PF2> and move the cursor to the end of the section to be marked. Press <PF2> and all the fields within the section of the same type as the first selected field, that is, data or literal, are marked. Or you can type the field select character over each attribute byte of each field to be selected.

Enter any non-blank character into a new field on the last line. You are warned that all selected fields will be deleted unless the drop field is cleared or a key other than <Enter> is pressed.

If you want to manipulate the fields, press <F11> and the screen is displayed with the *Alternate Function* keys, as shown:



```
 ...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8

 F1=Help  F2=Mark  F3=Copy  F4=Move  F5=Delete  F6=Preview  F8=Bottom
 F9=Propagate  F10=ClrMark  F11=MainKeys









Subscript increment: 1
```

To Propagate a Field

From the *Alternate Function keys* screen, put the cursor under any data or literal field and press <F9>. The field above the cursor is copied onto each lower line until it is copied to the cursor's line, until a field that would overlap a copied field is encountered, or until the maximum subscript is reached.

If the field to be propagated is an occurring data field and its subscript is to be incremented by a number other than 1, you must enter the subscript increment into a new field on the last line before pressing <F9>.

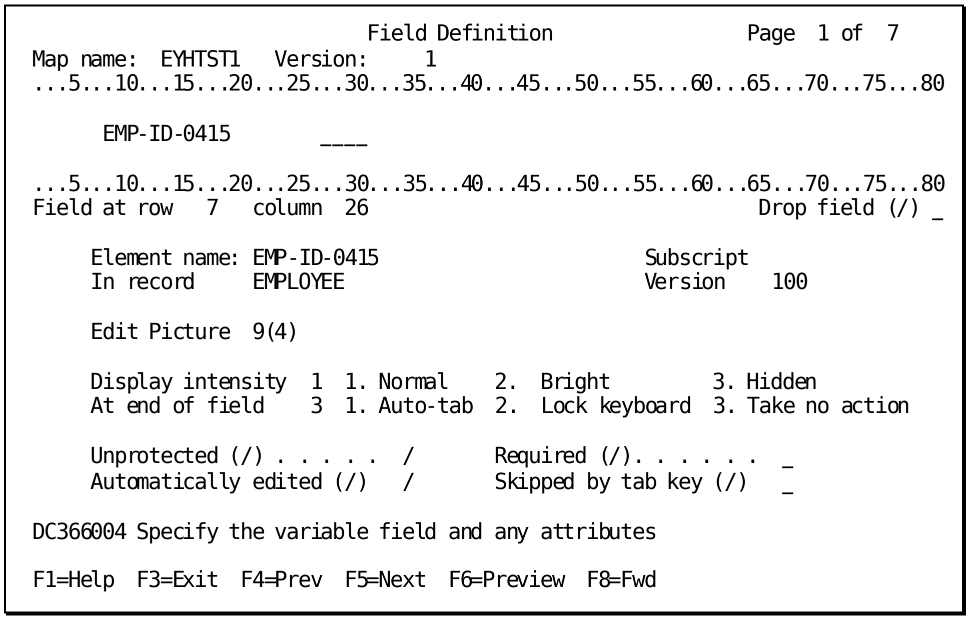
Field Definition Screens

Field Definition

Description

This is one of seven screens used to specify information about a particular field. This screen is used to enter a miscellaneous assortment of information about a field.

Sample Screen



```
                           Field Definition                Page  1 of  7
Map name:  EYHTST1   Version:     1
...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80

      EMP-ID-0415        ____

...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80
Field at row   7   column  26                                 Drop field (/) _

     Element name: EMP-ID-0415                      Subscript
     In record     EMPLOYEE                         Version    100

     Edit Picture  9(4)

     Display intensity  1  1. Normal    2.  Bright        3. Hidden
     At end of field    3  1. Auto-tab  2.  Lock keyboard  3. Take no action

     Unprotected (/) . . . . .  /      Required (/). . . . . .  _
     Automatically edited (/)   /      Skipped by tab key (/)   _

DC366004 Specify the variable field and any attributes

F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F8=Fwd
```

Field Definitions

Field at row/column	Specifies the location of the field on the map.
Drop field	Specifies whether the field should be deleted.
Element name	The name of the record element associated with the field.
In record	The name of the record with which the element is associated. The record must be previously defined for the map using the Associated Records screen.
Subscript	Subscript for field with OCCURS clause.
Version	The version number of the record.

Edit picture	Enables automatic editing and establishes an external picture for the field for use by automatic editing. A developer can specify one of the following to establish a particular external picture: ■ *An external picture*, such as XX/XX/XX or XXX-XX-XXXX ■ *The word INTERNAL*, which requests that the map use the external picture associated with the record element (or the picture constructed for the field) When automatic editing has been enabled for the field by a different specification, the external picture associated with the record element (or the picture constructed for the field) is displayed for Edit picture. **Note:** For more information about external pictures and automatic editing, see the chapter "Automatic Editing and Error Handling."
Display intensity	Specifies whether the field is displayed and whether a displayed field appears at normal or bright intensity: *Normal*—Specifies that the field is displayed at normal intensity. *Bright*—Specifies that the field is displayed at brighter than normal intensity; the field appears highlighted on the screen. *Hidden*—Specifies that the field is not displayed. Data written to the field from program storage or entered by the operator is not visible. Password fields are often attributed the Hidden attribute.
At end of field	■ Auto-tab ■ Lock keyboard ■ Take no action

Unprotected	Specifies whether the field accepts operator input: ■ **/**—Specifies that the field is open to data entry or modification. Data in an unprotected data field is transmitted to program variable storage on mapin if all of the following conditions are true: ■ Modifications have been made to the field (the MDT is set on). ■ Automatic editing does not detect an input error in the data. ■ The Transmit data entry option has been chosen on the Map Read/Write Options screen. ■ *Blank*—Specifies that the field does not accept user input. Any attempt to enter, modify, or delete data in the field is physically restricted by a 3270-type terminal. If MDT is set programmatically data is read. Data in a Protected field on some glass TTY terminals can be overridden by the terminal operator; however, operator modifications are ignored on mapin.
Required	Indicates if data must be entered in the field.
Automatically edited	Indicates if automatic editing is enabled. ■ **/**—Enables automatic editing for the field. Automatic editing is to be performed for the field if automatic editing is also enabled for the entire map on the first Associated Records screen. ■ *Blank* (default)—Disables automatic editing for the field. Automatic editing is not performed for the field, even if automatic editing is enabled for the entire map on the first General Options screen. Editing can be enabled for a field by entering a slash (**/**) in the *Automatically edited* field or by supplying an external picture, edit table name, or code table name on the Additional Edit Criteria screen. The most recent specification takes precedence and determines whether automatic editing is enabled or disabled. For example, if the *Automatically edited* field is blank, and the developer later names an edit table for the field, automatic editing is enabled. If the format of the record element associated with the data field is not DISPLAY, editing must be enabled so that conversion to DISPLAY format is performed.

Skipped by tab key	■ **/**—Specifies that the operator cannot use tabto position the cursor on this field at runtime; the cursor skips over this field and is positioned on the next unprotected field. Choosing this option specifies NUMERIC and PROTECTED for the field. ■ *Blank* (default for variable fields)—Specifies that the cursor is positioned at the start of the field when the operator presses the tab key at runtime. Choosing this option specifies UNPROTECTED for a field.

Map Read/write Options

Description

This is one of seven screens used to enter information for a specific field. This screen specifies how fields are handled on the mapin and mapout operations.

Sample Screen

```
                          Map Read/Write Options            Page  2 of  7
Map name:  EYHTST1   Version:     1

     Element name  EMP-ID-0415                       Subscript
     In record     EMPLOYEE                          Version    100

Map Read      Transmit data entry (/) . . . . . . . . /
options       Zero when null (/). . . . . . . . . . . /
              Translate to upper case (/) . . . . . . _
              Justify data. . . . . . . . . . . . . . 1  1. Left   2. Right
              Pad character format  . Display . . . . _
                                      Hexadecimal . . __

Map Write     Blank when zero (/) . . . . . . . . . . _
options       Underscore blank fields (/) . . . . . . _
              Display without trailing blanks . . . . _
              Set modified data tag (/) . . . . . . . _
              Transmit. . . . . . 1 1. Data and attribute byte  3. Erase field
                                    2. Attribute byte only      4. Nothing

DC366404 Select input/output edit options

F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F7=Bkwd  F8=Fwd
```

Field Definitions

Map Read options

Identifies characteristics of the field that pertain to mapin operations.

Transmit data entry

Specifies whether the transmitted contents (if any) of the field are to be moved automatically into program variable storage on a mapin operation:

/ (default)

Specifies that data is automatically moved into program variable storage if MDT is set on for the field before a mapin operation.

Blank

Specifies that the data contained in the field is not moved automatically into program variable storage, even if the MDT is set on.

Zero when null

Specifies whether the numeric field is to be filled with zeros when automatic editing is enabled for a numeric field and the terminal operator nulls (erases) the contents of the field, as by pressing the ERASE EOF key. The following options are available:

/ (default)

Specifies that the field is filled with zeros of the appropriate data type when automatic editing is enabled for a given field and map and the operator nulls the entire field.

Blank

Specifies that data already contained inthe buffer is retained when the entire field is filled with nulls.

Translate to upper case

Specifies if the field should be translated to upper case upon mapin.

Justify data

Specifies how operator input is to be aligned for transmission to variable storage:

Left

Specifies that input is left-justified.

Right

Specifies that input is right-justified.

Pad character format

Specifies a pad character for an alphanumeric field in character or hexadecimal format.

No pad character is used for a field in either of the following cases:

- The developer does not specify a pad character.
- The developer cancels a pad character for a field by pressing the ERASE EOF key for the PAD CHAR field in which the pad character was specified, and does not specify another pad character.

Unwanted data can be stored for a field for which no pad character is defined. For example, the following values are stored for a field if JOHNSON is mapped out, the operator presses the ERASE EOF key to erase the field, and the operator then types SMITH:

- *If no pad character is defined for the field*, SMITHON is stored for the field. The operator would have to key blanks over ON to eliminate these characters from the data.
- *If a pad character is defined for the field*, SMITH is stored for the field.

Map Write options

Blank when zero

Indicates how a numeric fieldbeing edited is to be mapped out when automatic editing is enabled for the field and the value for the field is 0:

/

Specifies that blanks are displayed in the field.

Blank (default)

Specifies that zeros are displayed in the field.

Underscore blank fields

Specifies that blank fields on a map are underscored. On mapin, trailing underscores are removed.

Display without trailing blanks

Specifies whether trailing blanks are to be eliminated from the field being edited before it is displayed:

/

Specifies that the contents of the field are displayed without trailing blanks (if any). Old data may remain in the field after operator alterations if NEWPAGE is NO in either the CA ADS sysgen statement or the DML statement that issues the mapout in an application program.

Blank

Specifies that the contents of the field are displayed with trailing blanks, if any.

Set modified data tag

Specifies whether the modified data tag is set automatically during a mapout.

Transmit

Specifies how the contents of the field are to be moved on a mapout.

Data and attribute byte

Specifies that data and the attribute byte is transmitted.

Attribute byte only

Specifies that only the attribute byte for the field is transmitted to the screen; data in the record buffer is not sent to the terminal.

Erase field

Specifies that data is not transmitted to the screen; the field is initialized to null or low values, depending on whether the field is numeric or alphanumeric.

Nothing

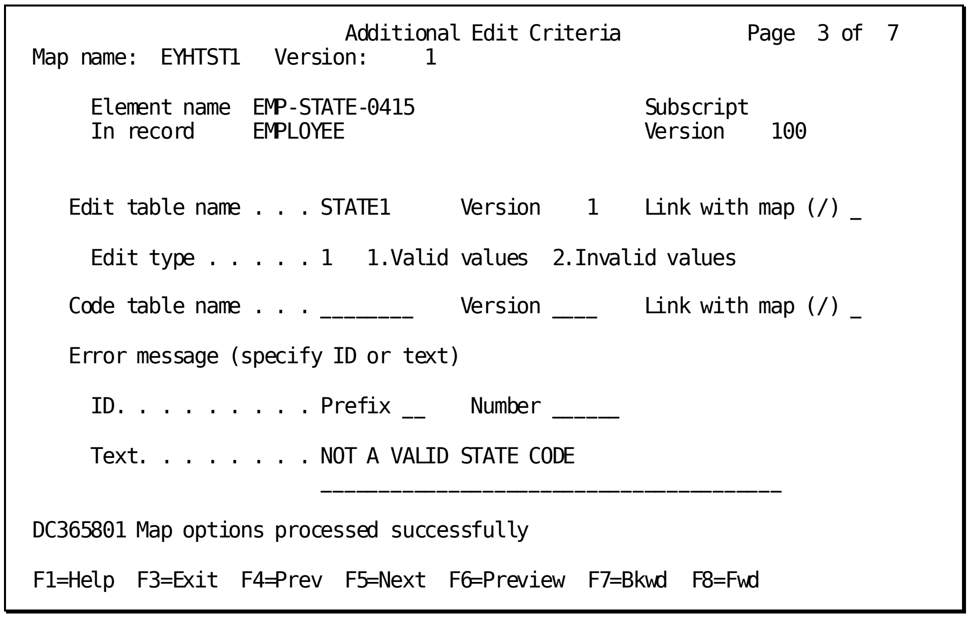
Specifies that neither data nor attribute byte is transmitted. Any data previously in the field continues to display.

Additional Edit Criteria

Description

This is one of seven screens used to enter information for a specific field. This screen is used to enter the edit table, code table, and error message information for a field.

Sample Screen



```
                           Additional Edit Criteria          Page  3 of  7
Map name:  EYHTST1   Version:     1

     Element name  EMP-STATE-0415                  Subscript
     In record     EMPLOYEE                        Version    100


   Edit table name . . . STATE1      Version    1    Link with map (/) _

     Edit type . . . . . 1   1.Valid values  2.Invalid values

   Code table name . . . ________    Version ____    Link with map (/) _

   Error message (specify ID or text)

     ID. . . . . . . . . Prefix __     Number _____

     Text. . . . . . . . NOT A VALID STATE CODE
                         ________________________________________

DC365801 Map options processed successfully

F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F7=Bkwd  F8=Fwd
```

Field Definitions

Edit table name	Enables automatic editing and specifies the name of an existing stand-alone table to be used as the edit table. If editing is enabled elsewhere for the field and no edit table is supplied, the default edit table is the built-in table (if any) defined in the associated record element. If a table name is not supplied here or in the record element, no edit table is used. ■ *Version*—Specifies the version of the edit table, default is 1. ■ *Link with map*—Specifies whether the edit table is linked as part of the map load module or is loaded dynamically at runtime: ■ **/**—Specifies that the table is linked to the map load module that uses it. This is useful for tables that contain items that cannot be used readily by another record element. ■ *Blank* (default)—Specifies that the table is loaded dynamically. This specification is useful when the contents of a table change frequently. **Note:** Edit tables can only be associated with a group element if the group is made up of DISPLAY elements. For more information about the use of edit tables, see "Edit and Code Tables".
Edit type	Indicates whether the table is one of valid or invalid tables: **Blank** (default)—Specifies the edit type defaults to the TYPE parameter for the table in the IDD. Tables are defined as valid or invalid depending on the IDD 'TABLE... TYPE IS EDIT VALID' or 'TABLE... TYPE IS EDIT INVALID'. ■ *Valid values*—Specifies the table contains valid values for the field. An error occurs when the operator inputs a value that is not contained in the table. ■ *Invalid values*—Specifies the table contains invalid or incorrect values. An error occurs when the operator inputs a value that is contained in the table.

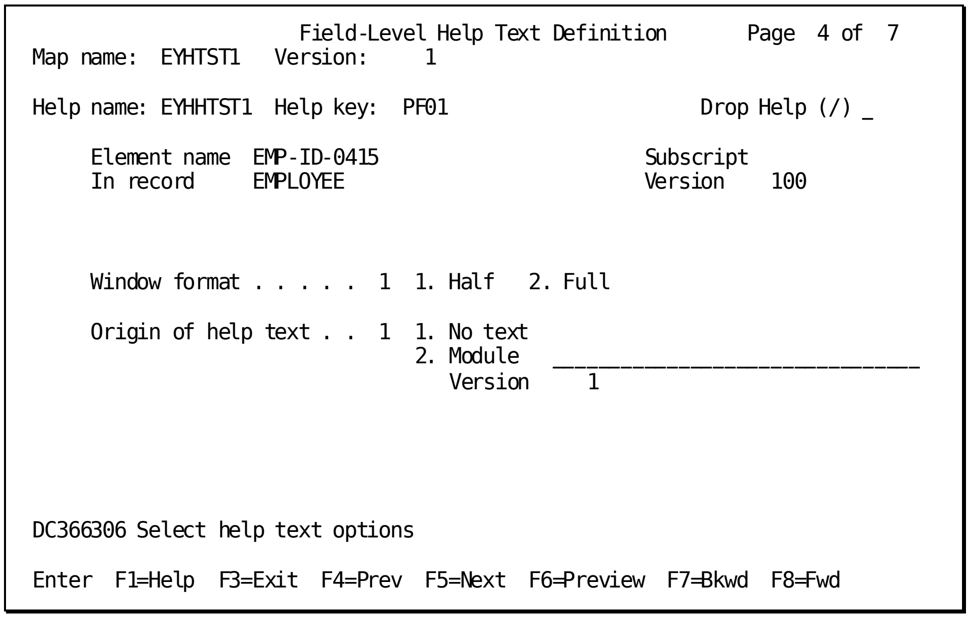
Code table name	Enables automatic editing and specifies the name of an existing stand-alone table to be used as the code table for the field being edited. If editing is enabled elsewhere for the field but the a code table name is not provided, code table is the table (if any) defined in the record element associated with the field. If a code table name is not provided and does not exist in the record element, no code table is used. *Version*—Specifies the version of the code table, default is 1. *Link with map*—Specifies whether the code table is linked as part of the map load module or is loaded dynamically at runtime: ■ **/**—Specifies that the table is linked to the map load module that uses it. This is useful for tables that contain items that cannot be used readily by another record element. ■ *Blank* (default)—Specifies that the table is loaded dynamically. This specification is useful when the contents of a table change frequently. **Note:** Code tables can only be associated with a group element if the group is made up of DISPLAY elements. For more information about the use of code tables, see the chapter "Edit and Code Tables."
Error message	Used to specify the number of an existing message or the developer-written text of the error message that is displayed for the field. The default is the message prefix specified on the General Options screen. If no prefix is specified, DC is used.

Field-level Help Text Definition

Description

This is one of seven screens used to enter information for a specific field. This screen is used to specify help information for a field.

Sample Screen



```
                       Field-Level Help Text Definition      Page  4 of  7
Map name:  EYHTST1   Version:     1

Help name: EYHHTST1  Help key:  PF01                     Drop Help (/) _

     Element name  EMP-ID-0415                      Subscript
     In record     EMPLOYEE                         Version    100



     Window format . . . . .  1  1. Half   2. Full

     Origin of help text . .  1  1. No text
                                 2. Module  _______________________________
                                    Version     1




DC366306 Select help text options

Enter  F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F7=Bkwd  F8=Fwd
```

Field Definitions

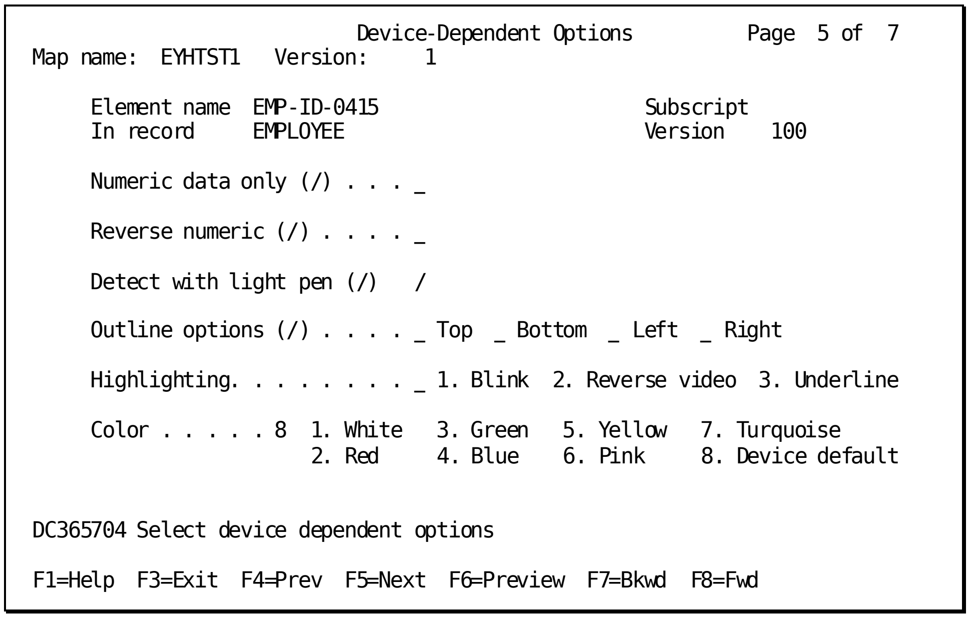
Help name	The name of the load module that contains all the compiled help information for the map and the associated fields.
Drop Help	Used to indicate if the help should be deleted.
Window format	tfcode used to indicate if the help text is displayed in a half (1) or full (2) window.
Origin of help text	Specify the name of the IDD module that contains the help text for this field.

Device-dependent Options

Description

This is one of seven screens used to enter information for a specific field.

Sample Screen



```
                            Device-Dependent Options         Page  5 of  7
 Map name:  EYHTST1   Version:     1

      Element name  EMP-ID-0415                        Subscript
      In record     EMPLOYEE                           Version    100

      Numeric data only (/) . . . _

      Reverse numeric (/) . . . . _

      Detect with light pen (/)   /

      Outline options (/) . . . . _ Top  _ Bottom  _ Left  _ Right

      Highlighting. . . . . . . . _ 1. Blink  2. Reverse video  3. Underline

      Color . . . . . 8  1. White   3. Green   5. Yellow   7. Turquoise
                         2. Red     4. Blue    6. Pink     8. Device default


 DC365704 Select device dependent options

 F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F7=Bkwd  F8=Fwd
```

Field Definitions

Numeric data only	Specifies whether any character or just numeric characters can be entered: ■ If the terminal is not equipped with the NUMERIC hardware feature, the operator can enter any character in the field. To restrict input to numeric characters in this case, a numeric external picture must be assigned to the field and automatic editing must be enabled for the field and map. ■ If the terminal is equipped with the NUMERIC feature, the field accepts only numeric input. Numeric input includes: ■ Numeric digits in the range 0—9 ■ The decimal point ■ The minus sign

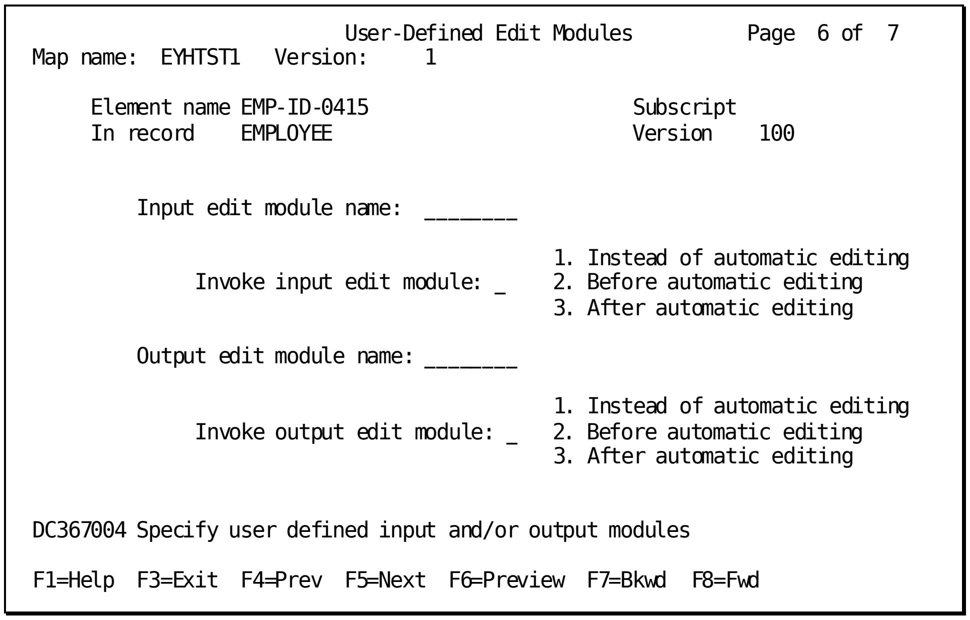
Reverse numeric	Specifies whether the contents of a numeric field are reversed on mapin and again on mapout. This option is used for numeric fields when hardware modifications cause input to be entered from right to left. REVERSE NUMERIC is the default for new fields when NUMERIC FIELD ORDER IS REVERSED is specified in the OLM statement at system generation. A blank in this field indicates that the field is not reversed on mapin or mapout, which is the default for new fields when NUMERIC FIELD ORDER IS STANDARD is specified at system generation.
Detect with light pen	Specifies whether the field can be detected with a light pen.
Outline options	Enables one or more of the following outline options if the terminal supports field outlining: *Top*—Draws a line above the current field. The line: ■ Starts above the first displayable character position in the field ■ Ends either at the delimiter of the current field (for delimited fields) or the start of the next field *Bottom*—Draws a line following the current field. The line: ■ Starts following the first displayable character position in the field ■ Ends either at the delimiter of the current field (for delimited fields) or the start of the next field *Left*—Draws a line to the left of the field. *Right*—Draws a line to the right of the field.
Highlighting	Specifies whether the field will blink, be displayed in reverse video, or be underlined.
Color	The runtime color of the field, or of the background if Reverse video is specified. The developer can specify only one display color. Device default specifies that the default display color for the terminal is used. Color attributes other than device default take effect only when the map is displayed at a 3279-type terminal.

User-defined Edit Modules

Description

This is one of seven screens used to enter information for a specific field. This screen is used to specify the name of the input and output edit modules. Additionally, it is used to indicate when the edit module is invoked.

Sample Screen



```
                              User-Defined Edit Modules          Page  6 of  7
  Map name:  EYHTST1   Version:     1

       Element name EMP-ID-0415                       Subscript
       In record    EMPLOYEE                          Version    100


           Input edit module name:  ________

                                                    1. Instead of automatic editing
                Invoke input edit module: _         2. Before automatic editing
                                                    3. After automatic editing

           Output edit module name: ________

                                                    1. Instead of automatic editing
                Invoke output edit module: _        2. Before automatic editing
                                                    3. After automatic editing


  DC367004 Specify user defined input and/or output modules

  F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F7=Bkwd  F8=Fwd
```

Field Definitions

Input edit module name	The name of a user-written edit module to process input after transmission on a mapin operation. **Note:** For more information about the use of user-written edit modules, see the appendix "User-Written Edit Modules."
Invoke input edit module	Specifies if the edit module should be invoked instead of, before, or after automatic editing.
Output edit module name	The name of a user-written output edit module used before display on an output operation.
Invoke output edit module	Specifies if the edit module should be invoked instead of, before, or after automatic editing.

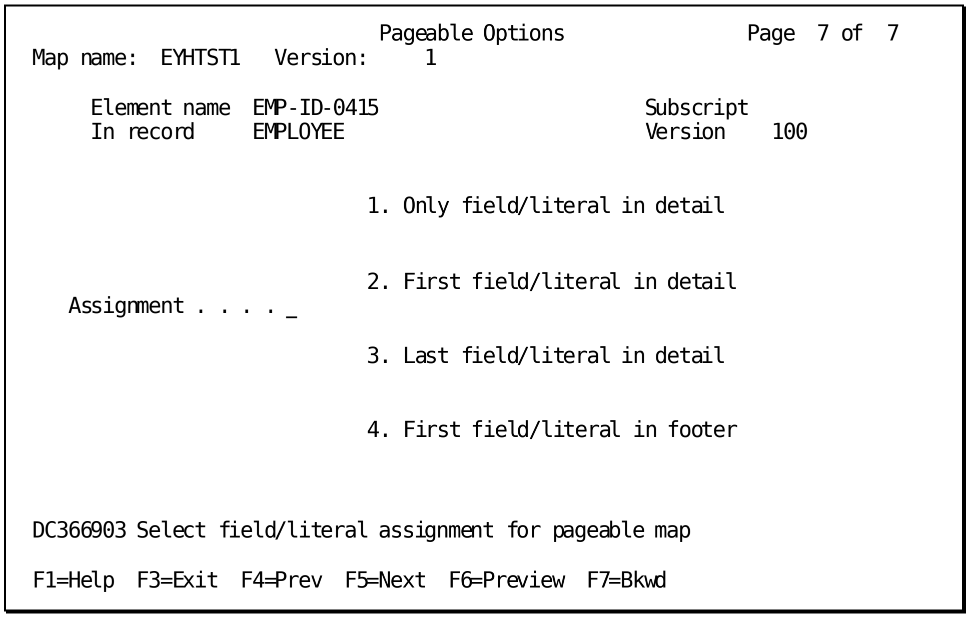
Pageable Options

Description

This is one of seven screens used to enter information about a specific field. This screen is used if the field is either:

- The only field/literal in the detail area
- The first field/literal in the detail area
- The last field/literal in the detail occurrence
- The first field/literal in the footer

Sample Screen



```
                              Pageable Options              Page  7 of  7
Map name:  EYHTST1   Version:    1

     Element name  EMP-ID-0415                        Subscript
     In record     EMPLOYEE                           Version    100


                             1. Only field/literal in detail


                             2. First field/literal in detail
   Assignment . . . . _

                             3. Last field/literal in detail


                             4. First field/literal in footer


DC366903 Select field/literal assignment for pageable map

F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F7=Bkwd
```

Field Definitions

Assignment

The information entered in these fields defines the specifications for a pageable map. The information you provide describes the element in the Element name field at the top of the screen:

- *Only field/literal in detail*—If the element is the *only* field or literal in the detail area, enter 1.
- *First field/literal in detail*—If the element is the *first* field or literal in the detail area, enter 2.
- *Last field/literal in detail*—If the element is the *last field or literal in the detail occurrence*, enter 3. The detail occurrence ends at the last character position in the field.
- *First field/literal in footer*—If the element is the *first field or literal in the footer area*, enter 4.

This field defines the beginning of the footer area and the end of the detail area.

Chapter 10: Batch Compiler and Batch Utility Overview

This chapter discusses about the batch compiler and batch utility overview.

This section contains the following topics:

Overview

The batch compiler and batch utility allow users to define, generate, modify, and delete maps in a batch environment. Together, they provide an alternative to the online mapping compiler described earlier in this manual.

To generate maps using the batch component of the mapping facility, it is necessary to be familiar with the following topics which are discussed separately in this section:

- Functions performed by the batch compiler and utility
- Functions performed by the map and panel entity occurrences

Compiler and Utility Functions

The CA IDMS mapping facility compiler and utility provide the following capabilities:

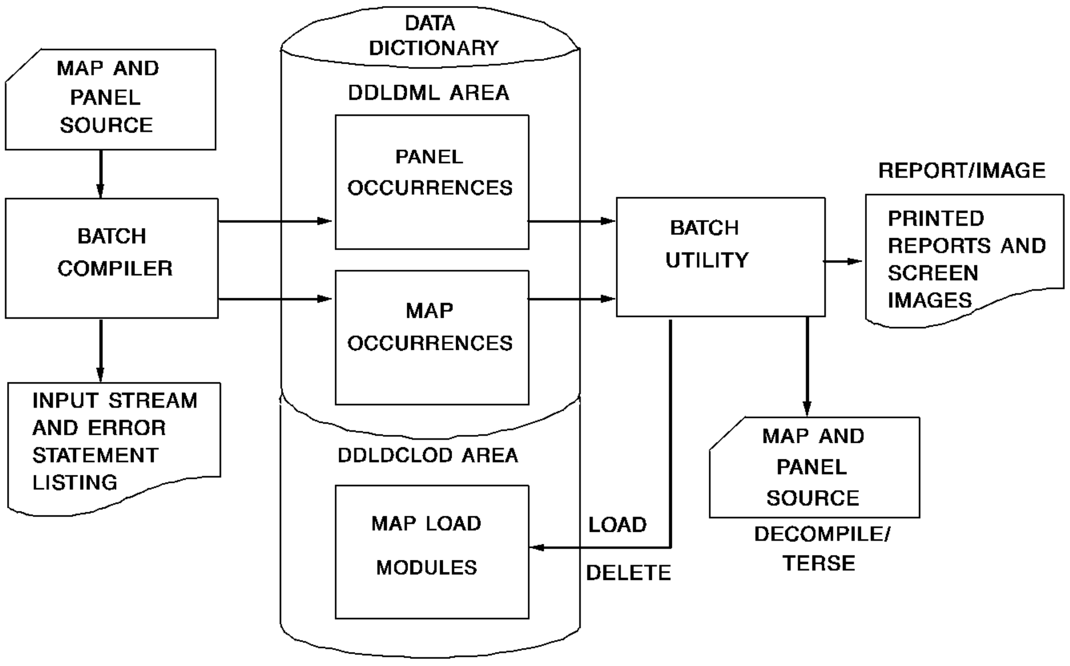
Component	Capability
Batch compiler (RHDCMAP1)	■ *Accepts, validates, and compiles mapping language statements* that are written by a map developer or generated by the decompile process of the map utility ■ *Populates the data dictionary with entity occurrences* that are generated when input mapping language statements are compiled ■ *Provides input statement listings* that present information and provide error messages (if any) about the compile operation

Component	Capability
Batch utility (RHDCMPUT)	Uses entity occurrences stored in the data dictionary to perform the following functions: ■ *Generate map load modules* used by CA ADS dialogs and programs ■ *Produce reports* about map-related entity occurrences created by the online mapping compiler or the batch compiler ■ *Decompile maps* into executable mapping language statements ■ *Delete map load modules* from the load area

A map load module is the version of the map that can be used by CA ADS dialogs and by the *CA IDMS System Generation Guide* programs. Necessary map-related entity occurrences must exist in the data dictionary before the map utility can generate a new or modified map load module.

Batch Compiler and Utility Functions

Functions performed by the batch compiler and utility are illustrated in the following figure:



Panels and Maps

What is a Map?

From the terminal operator's point of view, a map is a screen display that is used for input and output operations. From a map developer's point of view, the screen display viewed by the operator is the product of a map load module that is generated from entity occurrences in the data dictionary.

What Does the Batch Compiler Do?

The batch compiler is used to establish map-related entity occurrences in the data dictionary. The following entity occurrences are generated and maintained by the batch compiler:

Type of Occurrence	Specifies	Performs these functions
PANEL and related PFLD (panel field) occurrences	The screen layout for a map	The panel typically: Specifies the device types on which the screen layout can display and establishes any special display conditions necessary for each device type Assembles panel field occurrences that store the following information for individual fields: ■ Location by row and column ■ Characteristics such as display intensity or color ■ Values for literal fields

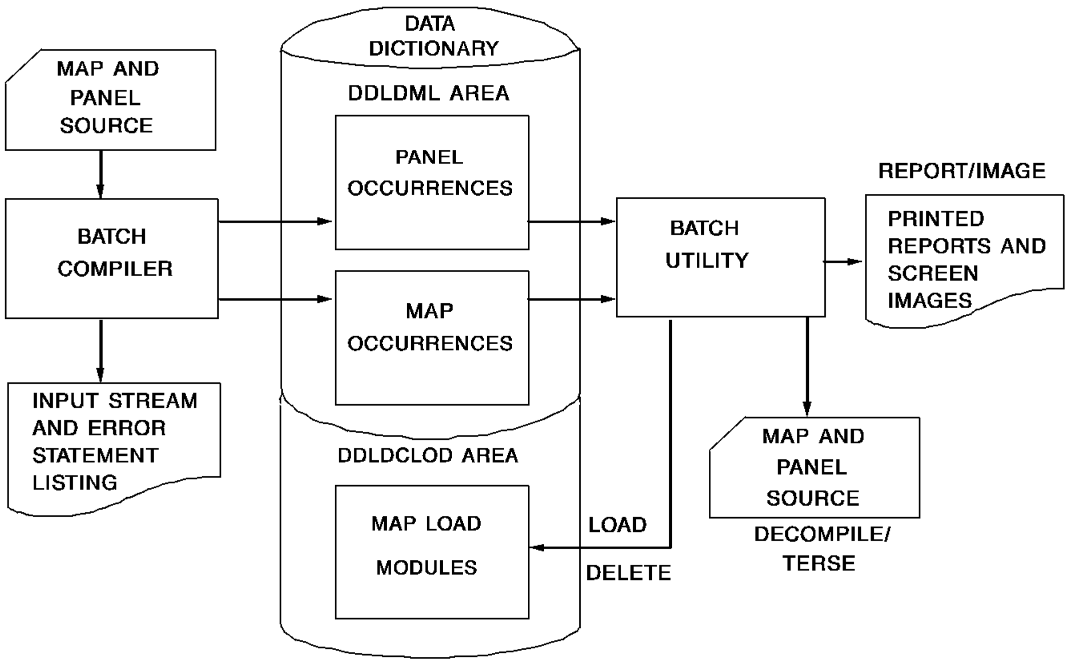
Type of Occurrence	Specifies	Performs these functions
MAP and related MFLD (map field) occurrences	Usage information for a map	The map typically: Specifies a PANEL occurrence on which the map occurrence is to be based Specifies runtime events that occur when a map load module generated from the MAP and PANEL occurrences is executed, such as whether the keyboard is locked or unlocked Specifies whether automatic editing and error-handling are available at runtime Selects PFLD occurrences from the associated PANEL occurrence and specifies the following information for the fields in MFLD (map field) occurrences: ■ Type of field usage (that is, data, literal, message, page, or response field) ■ Predefined schema or work record elements to be linked with data fields. ■ Automatic editing, input, and output information for data fields

Mapping Language Statements

Two sets of mapping language statements that can be used to generate PANEL, PFLD, MAP, and MFLD occurrences are presented in "Batch Compiler Statements".

Panels, Maps, and Record Elements

The following figure illustrates the relationship between panels, maps, and record elements.



A panel defines a screen layout and a map associates panel fields with elements in records.

Chapter 11: Batch Compiler Coding Considerations

This chapter discusses about the batch compiler coding considerations.

This section contains the following topics:

Overview

The coding considerations that are presented in this section apply when developers use the batch compiler to generate, modify, or delete map-related entity occurrences in the data dictionary. The following topics are presented:

- Compiler security
- Compiler signon
- Compiler directives
- Compiler statement coding requirements
- Compiler statement sequencing
- Compiler action verbs

Compiler Security

Batch compiler security prohibits unauthorized map developers from adding, modifying, and/or deleting map-related entity occurrences. The batch compiler performs a security check whenever the map developer using the batch compiler specifies the name of a map to be added, modified, or deleted. If the security check fails, the map developer cannot perform the specified action.

Security is established by using the IDD Data Dictionary Definition Language (DDDL) and can be applied at the compiler level and at the map level. The two levels of security are discussed separately as follows:

Security at the Compiler Level

Security at the batch compiler level restricts the actions that a map developer can specify for any map. Security at the batch compiler level is governed by the following IDD statements:

- **SET OPTIONS ... SECURITY FOR IDMS DC. IS ON/OFF**

  Specifies whether security is in effect for CA IDMS entity types accessed by the compiler. If CA IDMS security is off, the map developer immediately passes the compiler level security check. If security is on and the map developer has not used the SIGNON statement (presented later in this section) to provide signon information, the developer immediately fails the security check. Otherwise, the map developer passes or fails the security check based on information specified by the USER statement (see the following description).

- **ADD/MOD USER** *user-name* **... INCLUDE/EXCLUDE DELETE/DISPLAY MAP/PANEL**

  Specifies the actions that the map developer has the authority to perform. The map developer passes or fails the security check depending on whether the developer has authority for the specified action.

Note: For more information about the SET OPTIONS and USER statements, see the *CA IDMS IDD DDDL Reference Guide*.

If the map developer fails the compiler-level security check, an error message is generated. If the developer passes the security check, the batch compiler performs a security check at the map-specific level.

Security at the Map Level

Security at the map level restricts the actions that a map developer can perform for the named map. If no MAP-098 record exists for the named map (that is, if no MAP statement has been issued in IDD or at system generation for the named map), the map developer immediately passes the security check. Otherwise, security is governed by the following two clauses of the MAP statement in IDD:

- **PUBLIC ACCESS FOR ALL/NONE/UPDATE/MODIFY/REPLACE/DELETE/DISPLAY**

  Specifies the actions that any user can specify for the named map. If the PUBLIC ACCESS clause is not included in the MAP statement or if the action requested by the map developer is allowed for any user, the developer immediately passes the security check. Otherwise, the developer passes or fails the security check based on information specified by the INCLUDE/EXCLUDE USER clause of the MAP statement (see the following description).

- **INCLUDE/EXCLUDE USER user-name REGISTERED FOR PUBLIC ACCESS/ALL/UPDATE/MODIFY/REPLACE/DELETE/DISPLAY**

  Specifies the actions that the map developer has the authority to specify for the named map. If the developer's user name is not included in the MAP statement, the developer immediately fails the security check. Otherwise, the map developer passes or fails the security check depending on whether the developer has authority for the action specified for the named map.

If the map developer fails the map-level security check, an error message is generated. If the developer passes the security check, the batch compiler initiates compile operations for the specified map.

Note: For more information about the MAP statement, see the *CA IDMS IDD DDDL Reference Guide*.

Compiler Signon

The batch compiler SIGNON statement can be included in batch source statements to provide signon information. A maximum of one SIGNON statement can be included for a given batch compiler job. If included, the SIGNON statement must be positioned in batch source statements as follows:

- *SIGNON must follow introductory compiler directives* (that is, ICTL, OCTL, and ISEQ directives). Compiler directives are presented later in this section.
- *SIGNON must precede compiler statements* (that is PANEL, MAP, MAP AUTOPANEL, MFLD, and PFLD statements). Compiler statements are presented in the chapter "Batch Compiler Statements".

The SIGNON statement integrates the batch compiler with IDD security features. If activated, IDD security is used to protect panel and map data dictionary occurrences from unauthorized modification and deletion. Security information included in the signon information determines the authority granted to the developer for data dictionary access.

Note: For more information about batch compiler security, see Compiler Security (see page 187).

Syntax

```
►►─── SIGnon ───────────────────────────────────────────────────────────────►

►──┬─ USER ─┬──────┬─ user-name ─┬──────────────────────────────────────┬─┬─ . ─►◄
   │        ├─ IS ─┤             └─ PASSword ─┬──────┬─ password ─┘     │
   │        └─ = ──┘                          ├─ IS ─┤                  │
   │                                          └─ = ──┘                  │
   └─ REVerse NUMeric ──────────────────────────────────────────────────┘
```

USER IS *user-name-aq*

Specifies the existing 1- through 32-character name of the map developer as specified by a DDDL USER statement. If the name contains embedded blanks, the string must be enclosed in quotes.

PASSWORD IS *password*

Specifies the 1- through 8-character password (if any) of the developer. A password is defined for a developer in the OCF USER statement that establishes the corresponding *user-name* in the SIGNON statement.

REVERSE NUMERIC

Specifies that the reverse-numeric display option is to be given to all numeric map fields that are generated or modified in the batch run. The REVERSE NUMERIC clause specified in the SIGNON statement overrides any conflicting MFLD REVERSE NUMERIC specifications (if any) in the accompanying batch source statements.

Note: A developer who does not supply a *user-name* and *password* where applicable when IDD security is enabled for the batch compiler can access only entities that are available for public access.

Compiler Directives

The compiler directive statements provided by the compiler mapping language are placed in compiler source statements to specify information for use during compilation and report generation. Available compiler directive statements are listed in the following table:

Statement	**Function**
ICTL=*(start-column-n, end-column-n)*	Directs the compiler to scan only the column range specified for meaningful data. The default specification is 1-80.
OCTL=*(line-count-n)*	Specifies the number of lines to appear on each page of the panel and map reports.
ISEQ=*(start-column-n, end-column-n)*	Directs the compiler to perform sequence checking on source statements falling within the specified column range.
EJECT	Directs the compiler to continue the printing of the output report on a new page.
SPACE *space-count-n*	Directs the compiler to skip from 1 through 9 blank lines between lines of the output report. Only one blank is allowed between SPACE and the integer specified.
comment-text	Directs the compiler to interpret characters following the asterisk as comment text. Comments always start with an asterisk but can be terminated by another asterisk or by the end of the card image.

Considerations

Each compiler directive in compiler source statements must occupy a line by itself. The following considerations apply to the use of compiler directives:

- *ICTL, OCTL, and ISEQ statements* must precede all statements in mapping language source statements.
- *EJECT and SPACE statements* can be coded anywhere in mapping language source statements.

Compiler Statement Coding Requirements

The following general coding requirements apply when using batch compiler source statements:

- Statements and keywords can start in any column. The compiler scans the column range specified in the ICTL compiler directive statement for meaningful data. The default column range is from 1 through 80.
- One or more blanks or commas are required between keywords.
- A statement can be coded on more than one line, but keywords cannot be split across lines.
- A period terminates a statement unless the period is contained in a comment or a quoted literal string.
- Any quoted literal can be continued from one line to another. The final character of the first line must be in the end column, as specified in the ICTL compiler directive statement discussed in the previous table. The first character in the continued literal must begin in the start column of the next line. No continuation character is required.

Compiler Statement Sequencing

The batch compiler provides two sets of mapping language statements, as described in "Batch Compiler Statements". Each set of mapping language syntax defines source statements that create and maintain panel, panel field, map, and map field entity occurrences in the data dictionary:

- *MAP AUTOPANEL and MFLD statements* are used to define map and map field occurrences explicitly. Panel and panel field occurrences are generated automatically by the mapping facility when MAP AUTOPANEL and associated MFLD statements generate map and map field occurrences.
- *PANEL, PFLD, MAP, and MFLD statements* are used to define panel, panel field, map, and map field occurrences explicitly.

Sequencing requirements for each set of mapping language syntax are listed separately as follows.

MAP AUTOPANEL and MFLD Statement Sequencing

A MAP AUTOPANEL statement must be followed immediately by the MFLD statements that define its related fields. The following considerations apply when the developer prepares source composed of MAP AUTOPANEL and MFLD statements:

- The map occurrence generated by the most recently executed MAP statement is established as current for subsequent MFLD statements.
- Associated panel and panel field occurrences are created when map and map field entity occurrences are generated by MAP AUTOPANEL and associated MFLD statements.

Sample Statement

The following abbreviated sample statements illustrate MAP AUTOPANEL and MFLD statement sequencing:

```
ADD MAP ONEMAP
    AUTOPANEL
    ADD MFLD
    ADD MFLD
    ADD MFLD
    ADD MFLD
```

In the previous example, entity occurrences for a map named ONEMAP and occurrences for related map fields are generated. Associated panel and panel field occurrences are generated by the batch compiler from information specified in the MAP AUTOPANEL and MFLD statements.

PANEL, PFLD, MAP, and MFLD Statement Sequencing

The following order of specification applies when the developer prepares source composed of PANEL, PFLD, MAP, and MFLD statements:

- A PANEL statement must be followed immediately by the PFLD statements that define its related fields. The panel occurrence generated by the most recently executed PANEL statement is established as current both for subsequent PFLD statements and subsequent MAP statements.
- A MAP statement must be followed immediately by the MFLD statements that define its related fields. The following considerations apply:
  - The panel occurrence named by the MAP statement must exist in the data dictionary before the MAP statement is compiled.
  - The map occurrence generated by the most recently executed MAP statement is established as current for subsequent MFLD statements.

Sample Statements

The following abbreviated sample statements illustrate PANEL, PFLD, MAP, and MFLD statement sequencing:

```
ADD PANEL NEWPANEL
    ADD PFLD EMP-ID
    ADD PFLD EMP-FNAME
    ADD PFLD EMP-LNAME
    ADD PFLD DEPT-CODE

ADD MAP NEWMAP
    PANEL IS NEWPANEL
    ADD MFLD EMP-ID
    ADD MFLD EMP-FNAME
    ADD MFLD EMP-LNAME
    ADD MFLD DEPT-CODE
```

In the previous example, the NEWPANEL panel occurrence is defined before related panel fields are defined; the NEWPANEL panel occurrence becomes current for the map occurrence generated by the ADD MAP NEWMAP statement.

It is not necessary to generate panel occurrences in the same compiler run as related map occurrences.

Compiler Action Verbs

Where can Verbs be Used?

The compiler provides the verbs ADD, MODIFY, and DELETE that specify the action the map compiler should perform on the accompanying mapping source statements. Each verb can be specified for any of the following statements in the CA IDMS mapping language syntax:

This Statement	Generates...
MAP AUTOPANEL	Map and panel occurrences
MFLD	(MAP AUTOPANEL only) Map field and panel field occurrences
PFLD	Panel field occurrences
MAP	Map occurrences
MFLD	Map field occurrences

Note: For more information about the previous mapping language statements, see "Batch Compiler Statements".

What do the Verbs Do?

The operations performed by the ADD, MODIFY, and DELETE verbs are as follows:

- *ADD* establishes a new occurrence in the data dictionary. If the occurrence specified by an ADD verb already exists in the data dictionary, the statement that contains the verb is flagged as an error.
- *MODIFY* changes an existing occurrence in the data dictionary. The tables on the following pages summarize considerations that apply when using the MODIFY verb for automatic and manual panel definition.
- *DELETE* removes an existing occurrence from the data dictionary. Deleted occurrences cannot be reconstructed from map load modules that were generated from those occurrences. The tables on the following pages summarize considerations that apply when using the DELETE verb.

Defaults

When a verb is not specified with a CA IDMS mapping statement, the following defaults apply:

- *ADD* is the default verb when the occurrence specified by the statement does not already exist in the data dictionary.
- *MODIFY* is the default verb if the occurrence specified by the statement already exists in the data dictionary.

Important: DELETE must be specified explicitly; it is never the default action.

Modifying a DEVICE Specification

Different versions of the same panel can be established for a variety of screen sizes by using the DEVICES clause of the MAP AUTOPANEL or the PANEL statement. Since information specified by the DEVICES clause can affect many other MAP clauses, the MODIFY verb does not modify the DEVICES clause. To change a DEVICES specification, the developer should use the following procedure:

1. Decompile the panel and all associated maps by using a map utility process and save the resulting output.
2. Delete the maps affected by the change. This action implicitly deletes all map fields that belong to the deleted maps.
3. Delete the panel occurrence itself. This action implicitly deletes all associated panel fields.
4. Add the revised panel occurrence with the new DEVICES specification, followed by the associated panel field, map, and map field occurrences.

The MODIFY Verb

Statements for Automatic Panel Definition

Statement	Description
MODIFY MAP AUTOPANEL	Modifies a map occurrence in the data dictionary and/or establishes currency for subsequent MFLD statements. The related panel occurrence is updated only if the panel occurrence has not been used to generate additional map occurrences. The map occurrence must be respecified in its entirety; no previously defined specifications are retained since MODIFY MAP AUTOPANEL functions as implicit DELETE and ADD operations.
MODIFY MFLD	Modifies a map field occurrence and its related panel field occurrence in the data dictionary. The map field occurrence must be respecified in its entirety; no previously defined specifications are retained since MODIFY MFLD (for MAP AUTOPANEL) functions as implicit DELETE and ADD operations.

Statements for Manual Panel Definition

Statement	Description
MODIFY PANEL	Modifies a panel occurrence in the data dictionary and/or establishes panel currency for subsequent PFLD statements. Any existing specifications, except for a DEVICES specification, can be modified. Previously omitted specifications can be added.
MODIFY PFLD	Modifies a panel field occurrence in the data dictionary. Previously omitted specifications can be added; however, a new FOR specification can be added only if the newly specified device is already defined in the DEVICES clause in the owner PANEL statement.

Statement	Description
MODIFY MAP	Modifies a map occurrence in the data dictionary and/or establishes map currency for subsequent MFLD statements. Any existing specification can be modified. Previously omitted specifications can be added; however, a new ORIGIN FOR specification cannot be added unless the newly specified device is already defined in the DEVICES clause of the owner panel statement. A modified USING RECORDS clause must rename all previously named records in the order in which they were originally named, followed by new record specifications.
MODIFY MFLD	Modifies a map field occurrence in the data dictionary. Any existing specification can be modified. Previously omitted specifications can be added.

The DELETE Verb

Statements for Automatic Panel Definition

Statement	Description
DELETE MAP AUTOPANEL	Deletes a map occurrence and all related map field occurrences from the data dictionary. The related panel and panel field occurrences that have been generated from the map are deleted from the data dictionary.
DELETE MFLD	DELETE MFLD is not a legal option within automatic panel definition syntax. A MODIFY MAP AUTOPANEL statement followed by MODIFY MFLD statements can be used to respecify a map and panel; the MFLD statements specify the fields to be created.

Statements for Manual Panel Definition

Statement	Description
DELETE PANEL	Deletes a panel occurrence and all related panel field occurrences from the data dictionary. All map occurrences derived from the panel occurrence can be deleted.
DELETE PFLD	Deletes a panel field occurrence from the owner panel occurrence in the data dictionary.

Statement	Description
DELETE MAP	Deletes a map occurrence and all related map field occurrences from the data dictionary and dissociates the map from the panel occurrence. If the panel is not associated with any map occurrences, the panel and panel field occurrences are also deleted from the data dictionary.
DELETE MFLD	Deletes a map field occurrence from the owner map occurrence in the data dictionary.

Chapter 12: Batch Compiler Statements

This chapter discusses about the batch compiler statements.

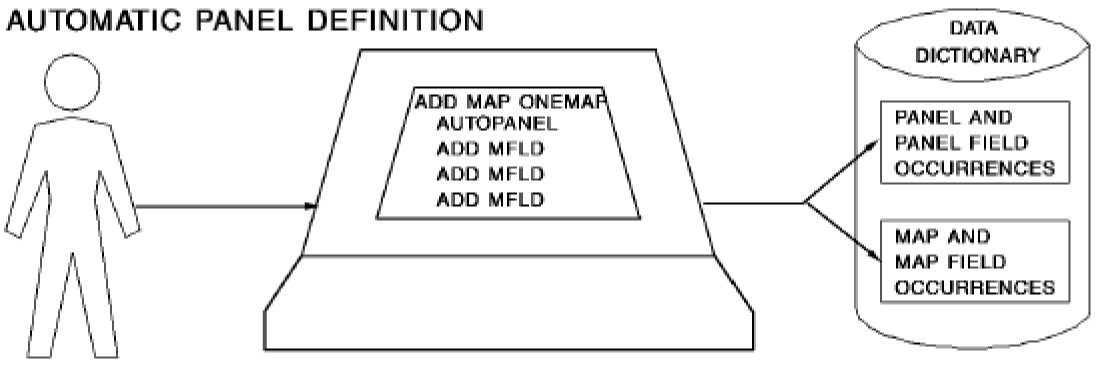
This section contains the following topics:

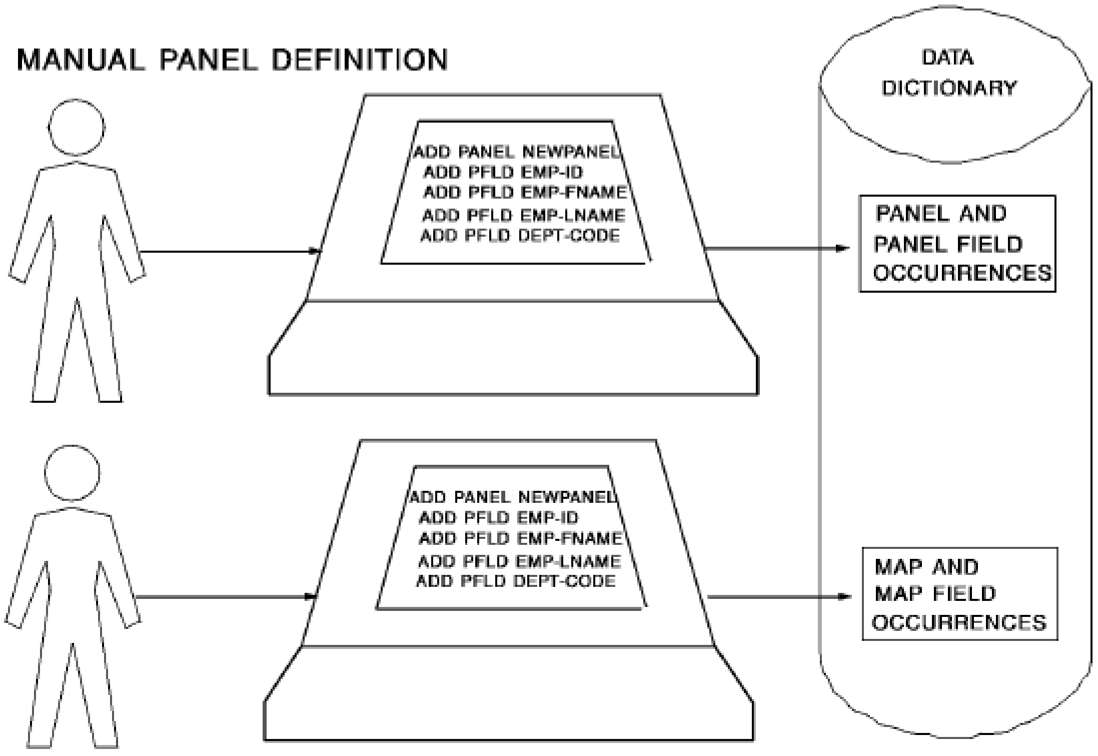
Overview

The batch compiler provides two sets of compiler statements for creating maps and panels:

- Statements that automatically define panels—These statements are used to create, modify, and delete map and map field occurrences in the data dictionary. Panel and panel field occurrences are created and updated automatically when map and map field occurrences are created and updated. The following statements are used for automatic panel definition:
  - The MAP AUTOPANEL statement defines panel and map occurrences.
  - The MFLD statement for MAP AUTOPANEL defines panel field and map field occurrences.
- Statements that manually define panels—These statements are used to create, modify, and delete map, map field, panel, and panel field occurrences. The following statements are used for manual panel definition:
  - The PANEL statement defines and generates a panel occurrence.
  - The PFLD statement defines and generates a panel field occurrence.
  - The MAP statement defines and generates a map occurrence.
  - The MFLD statement defines and generates a map field occurrence.

Automatic panel definition is contrasted with manual panel definition in the following figures:


AUTOMATIC PANEL DEFINITION
ADD MAP ONEMAP
AUTOPANEL
ADD MFLD
ADD MFLD
ADD MFLD
DATA DICTIONARY
PANEL AND PANEL FIELD OCCURRENCES
MAP AND MAP FIELD OCCURRENCES



What's in this Section?

The statements used to define panels and statements automatically and manually are presented separately below, following a discussion of the attributes list which can be used when defining a panel either automatically or manually.

Attributes List

Attributes that can be assigned to any given field are defined in the *attributes-list* parameter. This parameter appears in the following clauses:

- *ON EDIT ERROR INCORRECT FIELDS ATTRIBUTES clause* of the MAP statement
- *ON EDIT ERROR CORRECT FIELDS ATTRIBUTES clause* of the MAP statement ON EDIT ERROR specification
- *ATTRIBUTES clause* of the PFLD statement
- *ATTRIBUTES clause* of the MFLD (for MAP AUTOPANEL) statement

How to use the Attributes List

The same list of attributes is available for use in each mapping language clause that allows a map developer to specify attributes for a field. For clarity and convenience, the complete syntax for *attributes-list* is presented only once in this section.

Important: The default values apply only to the ATTRIBUTES clauses of the PFLD statement and the MFLD (for MAP AUTOPANEL) statement. No default values apply to the INCORRECT FIELDS ATTRIBUTES and CORRECT FIELDS ATTRIBUTES clauses.

Syntax

```
►►───┬─────────────────────┬─────────────────────────────────────────────────►◄
     ├─┬─ ALPHAnumeric ─┬──┤
     │ └─ NUMeric ──────┘  │
     ├─┬─ PROTected ────┬──┤
     │ └─ UNPROTected ──┘  │
     ├─── SKIP ────────────┤
     ├─┬─ DETECTable ───┬──┤
     │ └─ NONDETECTable ┘  │
     ├─┬─ DISPlay ──────┬──┤
     │ ├─ DARK ─────────┤  │
     │ └─ BRIGHT ───────┘  │
     ├─┬─ MDT ──────────┬──┤
     │ └─ NOMDT ────────┘  │
     ├─┬─ BLINK ────────┬──┤
     │ └─ NOBLINK ──────┘  │
     ├─┬─ REVerse-video ┬──┤
     │ └─ NORMal-video ─┘  │
     ├─┬─ UNDERscore ───┬──┤
     │ └─ NOUNDERscore ─┘  │
     ├─┬─ ALLLine ──────┬──┤
     │ └─ NOLIne ───────┘  │
     ├─┬─ LEFTLine ─────┬──┤
     │ └─ NOLEFTLine ───┘  │
     ├─┬─ RIGHTLine ────┬──┤
     │ └─ NORIGHTLine ──┘  │
     ├─┬─ BOTTOMLine ───┬──┤
     │ └─ NOBOTTOMLine ─┘  │
     ├─┬─ TOPLine ──────┬──┤
     │ └─ NOTOPLine ────┘  │
     └─┬─ BLue ─────────┬──┘
       ├─ RED ──────────┤
       ├─ PINk ─────────┤
       ├─ GREen ────────┤
       ├─ TURquoise ────┤
       ├─ YELlow ───────┤
       ├─ WHIte ────────┤
       └─ NOColor ──────┘
```

Parameters

ALPHAnumeric/NUMeric

Specifies the characters that can be entered in the field:

- *ALPHANUMERIC* (default for variable fields) specifies that the operator can enter any character.
- *NUMERIC* specifies that the operator can enter characters as follows:
  - *If the terminal is not equipped with the NUMERIC hardware feature,* the operator can enter any character in the field. To restrict input to numeric characters in this case, a numeric external picture must be assigned to the field and automatic editing must be enabled for the field and map, as specified in "Automatic Editing and Error Handling".
  - *If the terminal is equipped with the NUMERIC feature*, the field accepts only numeric input.

  Numeric input can include the following characters:

  - Numeric digits in the range 0 through 9
  - The decimal point (.)
  - The minus sign (-)

PROTected/UNPROTected

Indicates whether the field can accept operator input:

- *PROTECTED* (default for literal fields) specifies that the field is input protected. Any attempt to enter, modify, or delete data in the field is physically restricted by a 3270-type terminal. Data in a PROTECTED field on some glass TTY terminals can be overwritten by the operator; however, operator modifications are ignored on mapin. If PROTECTED is specified with either the ADD MFLD (for MAP AUTOPANEL) or the ADD PFLD statement, the DELIMIT/NODELIMIT clause of either statement defaults to NODELIMIT.
- *UNPROTECTED* (default for variable fields) specifies that the field is open to data entry or modification. Data in an unprotected data field is transmitted to program variable storage on mapin if all of the following conditions are true:
  - Modifications have been made to the field (the MDT is set on).
  - Automatic editing does not detect an input error in the data.
  - DATA is set to Y (YES) for input in the MFLD statement for the field.

  If UNPROTECTED is specified with an ADD operation, the DELIMIT/NODELIMIT clause defaults to DELIMIT.

SKIP

Specifies that the operator cannot use the TAB key to position the cursor on the field; the cursor is advanced to the next UNPROTECTED field. Indicating SKIP for a field specifies NUMERIC and PROTECTED for the field.

DETECTable/NONDETECTable

Specifies whether the field is detectable by the selector light pen:

- *DETECTABLE* specifies that the field is detectable by light pens.
- *NONDETECTABLE* (default) specifies that the field is not detectable by selector light pens. NONDETECTABLE does not apply to the INCORRECT FIELDS ATTRIBUTES or CORRECT FIELDS ATTRIBUTES clause.

DISPlay/DARK/BRIGHT

Indicates whether the field is displayed and, if displayed, whether it appears at normal or bright intensity:

- *DISPLAY* (default) specifies that the field is displayed at normal intensity.
- *DARK* specifies that the field is not displayed. Data written to the field from program variable storage or entered by the operator is not visible on the screen.
- *BRIGHT* specifies that the field is displayed at high intensity; the field appears highlighted at runtime.

BRIGHT fields are always DETECTABLE; DARK fields can never be DETECTABLE.

MDT/NOMDT

Data fields only; indicates whether the modified data tag (MDT) is set on automatically for the field on a mapout operation:

- *MDT* specifies that the modified data tag is set on automatically on mapout.
- *NOMDT* (default) specifies that on mapout the MDT is not automatically set on; the MDT is set on only when the contents of the field are altered by a terminal operator.

BLINK/NOBLINK

Specifies whether the field is to blink at runtime:

- *BLINK* specifies that the field blinks. The BLINK attribute takes effect only when the map is displayed at a 3279-type terminal.
- *NOBLINK* (default) specifies that the field does not blink.

REVerse-video/NORMal-video

Indicates whether the field is displayed in reverse or normal video:

- *REVERSE-VIDEO* specifies that the color of the characters in the field and of the background are reversed. The REVERSE-VIDEO attribute takes effect only when the map is displayed at a 3279-type terminal.
- *NORMAL-VIDEO* (default) specifies that the color of the characters in the field and of the background are not reversed.

UNDERscore/NOUNDERscore

Indicates whether the field is underscored:

- *UNDERSCORE* specifies that the field is underscored. The UNDERSCORE attribute takes effect only when the map is displayed at a 3279-type terminal.
- *NOUNDERSCORE* (default) specifies that the field is not underscored.

ALLLine/NOLIne

Enables or disables field outlining for the entire field.

- *ALLLine* draws a line on the top, bottom, left, and right of the field.

  This is equivalent to selecting all four outline options on the Device-Dependent Options screen of MAPC Field Definition. No other outline options should be specified.
- *NOLINE* indicates no outlining should occur. This is the default.

  No other outline options should be specified.

LEFTLine/NOLEFTLine

Enables or disables field outlining to the left of the field.

- *LEFTLINE* draws a line to the left of the field.

  This is equivalent to selecting the Left option on the MAPC Device-Dependent Options screen of Field Definition.

- *NOLEFTLINE* indicates no line should appear to the left of the field.

RIGHTLine/NORIGHTLine

Enables or disables field outlining to the right of the field.

- *RIGHTLINE* draws a line to the right of the field.

  This is equivalent to selecting the Right option on the MAPC Device-Dependent Options screen of Field Definition.

- *NORIGHTLINE* indicates no line should appear to the right of the field.

BOTTOMLine/NOBOTTOMLine

Enables or disables field outlining below the field.

- *BOTTOMLINE* draws a line below the field.

  This is equivalent to selecting the Bottom option on the MAPC Device-Dependent Options screen of Field Definition.

- *NOBOTTOMLINE* indicates no line should appear below the field.

TOPLine/NOTOPLine

Enables or disables field outlining above the field.

- *TOPLINE* draws a line above the field.

  This is equivalent to selecting the Top option on the MAPC Device-Dependent Options screen of Field Definition.

- *NOTOPLINE* indicates no line should appear above the field.

In order for the above field outlining options to draw lines in selected locations around the field when the MAP is displayed, the terminal or emulator being used must support field outlining.

More information:

For more information about specifying Outline options with MAPC, see Online Mapping Compiler Reference: Device-dependent Options.

BLue/RED/PINk/GREen/TURquoise/YELlow/WHIte/NOColor

Specifies the runtime color of the field, or of the background if REVERSE-VIDEO is specified for the field. Only one display color can be specified for a given field. NOCOLOR (default) specifies that the default display color for the terminal is used. Color attributes other than NOCOLOR take effect only when the map is displayed at a 3279-type terminal.

Note: BLINK, UNDERSCORE, and REVERSE-VIDEO are mutually exclusive. For example, neither REVERSE VIDEO nor UNDERSCORE can be assigned to a field for which the BLINK attribute is defined.

Statements for Automatic Panel Definition

Overview

The developer explicitly defines map and map field occurrences when using statements that automatically define panels. Related panel and panel field occurrences are generated and updated automatically. The compiler action verbs ADD, MODIFY, and DELETE define the overall purpose of the mapping statements.

Note: For more information about the ADD, MODIFY, and DELETE verbs, see Compiler Action Verbs.

Statements You can use

- The *MAP AUTOPANEL statement* defines and generates a map occurrence and automatically generates a related panel occurrence for the map occurrence. The AUTOPANEL clause is always included in a MAP statement that automatically generates a panel occurrence.
- The *MFLD statement* defines and generates a map field occurrence and automatically generates a related panel field occurrence for the map field occurrence.

A MAP AUTOPANEL statement must be followed immediately by the MFLD statements that define its related fields. The MAP AUTOPANEL and MFLD statements are presented separately as follows.

MAP AUTOPANEL Statement Syntax

A MAP AUTOPANEL statement typically is used to perform the following functions:

- Create or maintain a map occurrence and its associated panel occurrence in the data dictionary
- Identify the map occurrence with a unique combination of name and version number
- Identify the associated panel occurrence with a unique name composed of the name of the map and the suffix -AUTOPANEL
- Specify the particular devices suitable for the map at runtime
- Identify the records and roles referenced by map data fields
- Enable global automatic editing and error-handling, specifying correct-field and incorrect-field attributes
- Specify various terminal hardware control functions (such as alarm or numeric options) to be invoked during mapout operations

```
►►─┬──────────┬─ MAP map-name ─┬──────────────────────────────┬─────────────►
   ├─ ADD ────┤                └─ VERSion ─┬──────┬─ version ─┘
   ├─ MODIFY ─┤                            ├─ IS ─┤
   └─ DELETE ─┘                            └─ = ──┘

 ►─┬─────────────────────────────────────────────┬──────────────────────────►
   └─ DATETIME ─┬──────┬─ date-time-stamp ─┘
                ├─ IS ─┤
                └─ = ──┘

 ►─┬───────────────────────────────────────────────────┬────────────────────►
   └─ MSG PREFIX ─┬──────┬─┬─ DC ◄ ────────────┬─┘
                  ├─ IS ─┤ └─ message-prefix ──┘
                  └─ = ──┘

 ►── AUTOPANEL DEVices = ─┬─ (device-code) ──────────────────┬──────────────►
                          ├─ (24x80, 32x80, 43x80, 27x132) ◄ ┤
                          └─ ALL ────────────────────────────┘

 ►─┬───────────────────────────────────────┬─┬─ RESident ──────┬────────────►
   └─ SYStem ─┬──────┬─ dc-version ─┘ └─ NONRESident ◄ ─┘
              ├─ IS ─┤
              └─ = ──┘

 ►─┬────────────────────────────────────────────────────────────────────────►
   └─ USING ─┬──────────────┬───────────────────────────────────────────────
             ├─ RECORDS ────┤
             └─ REC ────────┘

 ►──────────────────────────────────────────────────────────────────────┬───►
  ────── ( ─▼─ record-name ─┬───────────┬─┬───────────────────────┬─┴── ) ──┘
                            └─ version ─┘ └─ ROLEname role-name ──┘
```

```
►──┬─EDIT ◄──┬──┬──────────────────────────────┬──────────────────────►
   └─NOEDIT ─┘  └─ CURSOR at panel-field-name ─┘

►──┬─────────────────────────────────────┬───┬──────────────────────────────┬──►
   └─ RESET ◄ ──┬──┬──────────────────┬──┘   └─┬─ LOCK ────┬──┬────────────┬─┘
   └─ NORESET ──┘  └──┬─ MODIFIED ─┬──┘        └─ UNLOCK ◄─┘  └─┬─ KEYBOARD ─┬─┘
                      └─ MOD ──────┘                            └─ KEY ──────┘

►──┬─ ALARM ──────┬──┬─ STARTPRT ──┬──┬─ NLCR ◄ ─┬──┬─ PAGeable ──────┬──►
   └─ NOALARM ◄ ──┘  └─ NOPRT ◄ ───┘  ├─ 40CR ───┤  └─ NONPAGeable ◄ ─┘
                                      ├─ 64CR ───┤
                                      └─ 80CR ───┘

►──┬────────────────────────────────────────────────────┬──►
   └─ DECimal point ──┬──────┬──┬─ Comma ─────┬─────────┘
                      ├─ IS ─┤  └─ Period ◄ ──┘
                      └─ = ──┘

►──┬──────────────────────────────────────────────────────────────►
   └─ HELP ──┬─ NO ───────────────────────────────────────────────
             └─ LOAD MODule ──┬──────┬── module-name ─────────────
                              ├─ IS ─┤
                              └─ = ──┘

─►──────────────────────────────────────────────────────────────────►─
 .─────────────────────────────────────────────────────────────────
  ─────────────────────────────────────────────────────────────────
   └─ SOUrce ──┬─ NONE ──────────────────────────────────────────────
               └─ MODule module-name ──┬──────────────────────────────────┬──
                                       └─ version ──┬──────┬── version ──┘
                                                    ├─ IS ─┤
                                                    └─ = ──┘

─►──┬─────────────────────────┬──────────────────────────────────►
    ──────────────────────┬───┘
    ────────────────────┬─┘
    ──┬─ HALF screen ◄ ─┤
      └─ FULL screen ───┘

►──────────────────────────────────────────────────────────────────►─
  ─────────────────────────────────────────────────────────────────
  ──┬──────────────────────────────┬
    └─ HELPKEY ──┬──────┬── PFnn ──┘
                 ├─ IS ─┤
                 └─ = ──┘

►──┬────────────────────────────────────────────────────────────────────►
   └─ ON edit ERROR ──┬──────────────────────────────────────────────────┬
                      └─ INCORRECT fields ATTRibutes = ( attributes-list ) ─┘

►──────────────────────────────────────────────────────────────────►
  ──┬────────────────────────────────────────────────────┬
    └─ CORRECT fields ATTRibutes = ( attributes-list ) ──┘

►──┬──────────────────────────────┬──────────────────────────────►
   ──┬──────────────────────────┬─┘
     └─ SOUND ──┬─ ALARM ──────┬┘
                └─ NOALARM ◄ ──┘

►──┬─────────────────────────────────────────────────────────────────┬─ . ─►◄
   └─ ORIGIN for ──┬─ ALL ────────────────────────┬──┬──────┬─( row  column )─┘
                   └─ ( ─▼─ device-code ─┴─ ) ────┘  ├─ IS ─┤
                                                     └─ = ──┘
```

Parameters

ADD/MODIFY/DELETE

Specifies the action taken with regard to the MAP statement. ADD, MODIFY, and DELETE access for a map is subject to security restrictions specified for the batch compiler and individual maps, as outlined in "Compiler Security".

MAP map-name

Specifies the unique 1- through 8-character name for the map being defined, modified, or deleted. The following considerations apply to the composition of *map-name*:

- *Map-name* can consist of any alphanumeric or special characters.
- *Map-name* must begin with an alphanumeric or national character; for example, pound sign (#), at sign (@), or dollar sign ($).
- *Map-name* must not contain embedded period or blank characters.

VERSION IS version-n

Optionally specifies a version number to further identify the map. *Version-n* must be in the range 1 through 9999. If omitted, *version-n* defaults to the data dictionary version default, as defined by the Data Dictionary Definition Language (DDDL) SET OPTIONS statement.

DATETIME IS date-time-stamp

The map compiler DATETIME clause is returned in map source statements when you use the map utility to decompile a map.

If you use the DATETIME option to decompile a map from one DC system and add it to another system:

- *Do not change decompiled map source statements.* If you change statements, unpredictable errors will occur at runtime when you access the map.
- *Define identical record element descriptions on each system.* You can accomplish this by using IDD.

MSG PREFIX IS message-prefix

Defines the two-character prefix to be used as the default prefix for any MFLD in the map that is defined using the ERROR MESSAGE clause.

AUTOPANEL

Specifies that panel and panel field occurrences are generated automatically when MAP and associated MFLD statements generate map and map field occurrences.

DEVICES=

(**device-code-a**) **/(24X80,32X80,43X80,27X132)/ALL** specifies the devices with which the map can be used:

- **(***Device-code-a***)** specifies devices (screen size) with which the map can be used. Valid screen sizes are 12X40, 12X80, 24X80, 32X80, 43X80, and 27X132.

  Commas must be used to separate *device-code-a* specifications when more than one device is chosen. Device specifications in the DEVICES clause must be enclosed in parentheses; for example, DEVICES=(12X40,24X80,43X80).

- **(24X80,32X80,43X80,27X132)** is the default specification given to the map.
- **ALL** specifies that the map can be used with all valid screen sizes.

To reserve a map field for use on only a subset of the devices specified in the DEVICES clause, FOR clauses can be included in an MFLD statement for MAP AUTOPANEL. FOR clauses can also be used to specify values or attributes for a field displayed on specific devices.

The MODIFY verb does not update the DEVICES specification.

Note: For more information, see "Compiler Action Verbs".

SYSTEM IS dc-version-n

Specifies the version number of a CA IDMS system with which the map is associated. *Dc-version-n* is the 1- through 4-character identifier assigned to the system at system generation.

RESIDENT/NONRESIDENT

Indicates whether the map load module is resident in storage at system runtime:

- **RESIDENT** specifies that the map load module is resident. This is useful for frequently used maps.
- **NONRESIDENT** (default) specifies that the map load module is not resident; the load module is loaded dynamically when required for a program mapping request.

USING RECORDS

(**record-name /**(**record-name version-n**) **ROLENAME role-name**

Specifies the list of predefined schema and/or work records used by the map and optionally specifies role names for records:

- **Record-name** identifies the name of a record that contains elements referenced by the map. If *record-name* is not unique in the data dictionary, the version number of the necessary schema or work record must be supplied; the default value for *version-n* is specified at system generation.

  If a logical record is being used, the developer names the records containing elements that are part of the logical record and that are used in the map definition. The logical record name is later specified by the dialog or program using the map.

- **ROLENAME role-name** specifies the role name used for the record at runtime. Role names are needed when a given record type is referenced in more than one context. For example, the developer might specify the EMPLOYEE record layout twice for a map that uses the EMPLOYEE record for both employee-related and manager-related fields on a single map:

  - One specification of the EMPLOYEE record would not include a role name for the record.
  - The second specification of the EMPLOYEE record would include a valid role name for the record (for example, MANAGER). The role name must be used in subsequent references to the record in the map-definition.

  The specified role name can be established in two ways:

  - The role name can be previously defined for the record by a logical record definition in the subschema used by the program or dialog.
  - The role name can be unique to the map, defined at map definition time on the Associated Records screen or via the batch compiler.

EDIT/NOEDIT

Indicates whether automatic editing and error-handling are enabled for the map, as follows:

- **EDIT** (default) globally enables automatic editing and error-handling for the map.
- **NOEDIT** globally disables automatic editing and error-handling for the map; editing and error-handling criteria (if any) defined for map fields are ignored.

Note: For more information about enabling and disabling automatic editing, see "Enabling Automatic Editing and Error Handling".

RESET/NORESET MODIFIED

Indicates whether the modified data tags (MDTs) for data fields are reset automatically on a mapout operation:

- **RESET** (default) specifies that all MDTs are reset (turned off) when the map is mapped out.
- **NORESET** specifies that MDTs are left unchanged when the map is mapped out.

The MDT/NOMDT specification in the MFLD ATTRIBUTES clause for a field overrides the RESET/NORESET specification for that field if the map-level and field-level specifications differ. If MDT is chosen for a field, the MDT is set on regardless of the RESET MDT specification.

Note: For more information about the MDT/NOMDT setting, see "Attributes for Fields".

LOCK/UNLOCK KEYBOARD

Specifies whether the keyboard unlocks automatically after a mapout operation:

- **LOCK** specifies that the keyboard remains locked until the operator presses the RESET key.
- **UNLOCK** (default) specifies that the keyboard is unlocked after a mapout.

ALARM/NOALARM

Indicates whether a terminal alarm sounds automatically on a mapout operation:

- **ALARM** specifies that the terminal alarm sounds on a mapout operation. This specification is meaningful only if the terminal is equipped with a hardware alarm.
- **NOALARM** (default) specifies that the terminal alarm does not sound on mapout.

STARTPRT/NOPRT

Specifies whether the contents of the printer terminal buffer should be printed automatically upon completion of data transmission on a mapout operation:

- **STARTPRT** specifies that the contents of the printer terminal buffer are printed. This specification is meaningful only for mapping operations associated with 3280-type printers.
- **NOPRT** (default) specifies that the contents of the printer terminal buffer are not printed.

NLCR/40CR/64CR/80CR

Specifies character-per-line formatting for printer output:

- **NLCR** (default) specifies that no line formatting is performed on the printed output. Printing skips to a new line only when new line (NL) and carriage return (CR) characters are encountered.
- **40CR** specifies that the buffer contents are printed at 40 characters per line.
- **64CR** specifies that the buffer contents are printed at 64 characters per line.
- **80CR** specifies that the buffer contents are printed at 80 characters per line.

These specifications are applicable only if the STARTPRT clause is specified for the map.

PAGEABLE/NONPAGEABLE

Specifies whether the map is pageable:

- **PAGEABLE** specifies that the map is pageable. A pageable map is a map that can display more than one page of information at runtime.
- **NONPAGEABLE** (default) specifies that the map is not a pageable map.

Note: For more information about pageable maps, see "Pageable Maps".

DECIMAL POINT IS COMMA/PERIOD

Specifies the decimal point character for numeric fields on the map:

- **COMMA** specifies that the comma (,) is used as the decimal point, in accordance with international format. An external picture for the field also must be specified in international format, with the comma as the decimal point.
- **PERIOD** (default) specifies that the period (.) is used as the decimal point.

HELP

Specifies whether help will be implemented for the map.

NO/LOAD MODule module name

If there is Help for the map, the name of the load module that contains all the help source for the map.

HELPKEY IS PFnn

The PFKey designated as the Help key for the map.

SOUrce NONE/MODule module-name

The name of the IDD module that contains the help text for the map.

If module name is specified, you can optionally specify:

- The version number
- Whether the help is displayed on a full or half screen

ON EDIT ERROR

Defines incorrect-field attributes, correct-field attributes, and alarm status for use when a dialog or map redisplays a map that contains input errors. The following clauses assign error-handling criteria:

- **INCORRECT FIELDS ATTRIBUTES=(attributes-list)** specifies attributes that are assigned to incorrect fields when an edit error occurs. Typically, incorrect fields are given an attribute such as BRIGHT or BLINK to draw the operator's attention to the erroneous data. No default attributes are defined.

  Note: Syntax for the **attributes-list** is discussed in Attributes List (see page 201).

- **CORRECT FIELDS ATTRIBUTES=(attributes-list)** specifies attributes that are assigned to correct and unedited fields when an edit error occurs. No default attributes are defined.

  Note: For more information about syntax for the **attributes-list**, see Attributes List (see page 201).

- **SOUND ALARM/NOALARM** specifies whether a terminal alarm sounds on input error:
  - **ALARM** indicates that the alarm is sounded. This option is meaningful only when a terminal is equipped with a hardware alarm.
  - **NOALARM** (default) indicates that the alarm is not sounded.

For example, a dialog or program can include code to redisplay a map when an error is detected in a field on mapin. When the display is mapped back out, incorrect-field attributes take effect for fields that are in error, and correct-field attributes take effect for fields that are not in error. The terminal operator can correct the errors and resubmit the map.

Note:

- For information about the use of error-handling specifications, see Error-handling Criteria (see page 76).
- For information about how dialogs and programs override specifications made in the ON EDIT ERROR clause, see Map Inquiry and Modification (see page 130).

ORIGIN FOR (device-code)/ALL IS (row column)

Positions the origin of the runtime map at a row/column location on specified devices:

- **Device-code** names one device. Available *device-code* specifications are 12X40, 12X80, 24X80, 32X80, 43X80, and 27X132. The specified device must be defined in the DEVICES clause of the MAP statement. More than one ORIGIN FOR *device-code* clause can be included in a single MAP statement.

  Parentheses are required when a device code(s) is specified.
- **ALL** names all devices defined in the DEVICES clause of the MAP statement.
- **Row column** specifies the coordinates at which the upper left-hand corner of the runtime map is plotted for all devices specified in the ORIGIN FOR specification. Only one *row column* specification can be made for a given ORIGIN FOR clause; if specified, it must be enclosed in parentheses. If not specified, *column* defaults to 1.

  Parentheses are required around the row column coordinates.

Examples

The following are the examples of the MAP AUTOPANEL statement:

Example 1

Adding a Map Occurrence

The following sample MAP AUTOPANEL statement adds a map occurrence named MEALS:

```
        ADD MAP MEALS VERSION IS 2
            AUTOPANEL DEVICES=(24X80)
            USING RECORDS MEALS-REC VERSION 1
            NOEDIT.
```

The MEALS-REC schema record is used by the sample map occurrence. Automatic editing is disabled by the NOEDIT specification. A panel occurrence is generated automatically for the MEALS map occurrence and given the name MEALS-AUTOPANEL.

Example 2

Modifying a Map Occurrence

The following sample MAP AUTOPANEL statement modifies the map occurrence established in the previous example:

```
        MOD MAP MEALS VERSION IS 2
            AUTOPANEL DEVICES=(24X80)
            USING RECORDS MEALS-REC VERSION 1
            EDIT
            ON ERROR
               INCORRECT ATTRIBUTES (BRIGHT)
               CORRECT ATTRIBUTES (DISPLAY).
```

Automatic editing is enabled by the EDIT clause. Attributes for the redisplay of incorrect and correct fields are added to the map. The related MEALS-AUTOPANEL panel is also modified by this sample statement unless the panel has been used as the basis for other map occurrences.

Example 3

Positioning a Map on a Device

The following sample MAP AUTOPANEL statement defines a map occurrence that can be displayed on two different devices at runtime:

```
ADD MAP SEATS
    AUTOPANEL DEVICES=(12X40,24X80)
    USING RECORDS PASS-REC VERSION 1
    ORIGIN FOR 12X40 IS 5,5
    ORIGIN FOR 24X80 IS 10,20.
```

The upper left-hand corner of the runtime SEATS map is positioned at row 5, column 5 on 12X40 devices. The upper left-hand corner of the runtime map is at row 10, column 20 on 24X80 devices. For more information about the placement of maps on different devices, see "Positioning Maps on Different Devices".

Example 4

Deleting a Map Occurrence

The following sample MAP AUTOPANEL statement deletes the MEALS map occurrence version 2 and simultaneously deletes any map field occurrences that belong to the map:

```
DEL MAP MEALS VERSION IS 2.
```

The related MEALS-AUTOPANEL panel occurrence and its panel field occurrences are deleted from the data dictionary by this sample statement unless the panel has been used as the basis for other map occurrences.

MFLD Statement Syntax

Functions Performed

An MFLD statement for MAP AUTOPANEL is used to add a map field to a map by performing the following functions:

- Creating and maintaining a single map field occurrence and associated panel field occurrence for the specified map and panel occurrences in the most recent MAP AUTOPANEL statement.
- Identifying the map field occurrence in the data dictionary with a name that is unique within the owner map occurrence.
- Identifying the panel field occurrence with a name composed of a 5-digit identifier with the prefix AUTOPF. The name AUTOPF00001 is assigned to the first panel field occurrence generated for the map, AUTOPF00002 is assigned to the second panel field occurrence generated for the map, and so forth.
- Specifying characteristics for the field, such as the following:
  - Field occurrences for multiply-occurring fields
  - Screen locations of the field by row and column
  - Physical attributes of the field, such as display color
  - Field values for literal fields
  - Delimit characteristics

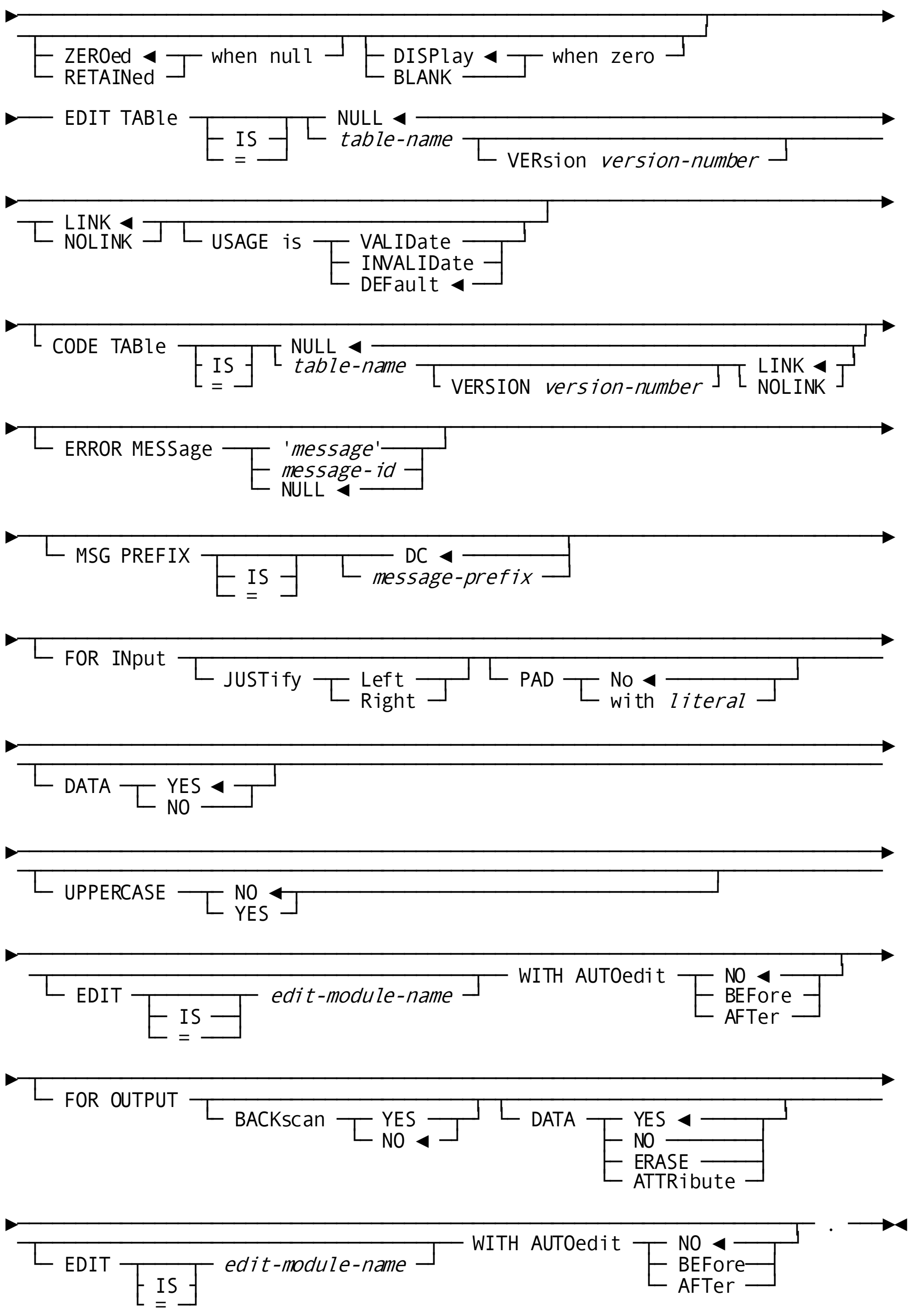
- Variable field type, as follows:
  - The *DFLD specification* establishes the field as a data field and relates it to a single existing record element in the data dictionary. Additional specifications, such as automatic editing criteria, can be made for data fields.
  - The *MESSAGE LENGTH specification* establishes the field as a message field and defines the length of the field. establishes the field as a page field and defines the length of the field.
  - The *RESPONSE LENGTH specification* establishes the field as a response field and defines the length of the field. A response field is meaningful only when the map is used by a CA ADS dialog.

```
►►─┬────────────┬─ MFld ───────────────────────────────────────────────────►
   ├─ ADD ──────┤
   └─ MODify ───┘

►─┬──────────────────────────────────────────────┬─────────────────────────►
  └─ OCCURS ─┬─ 1 ◄ ───────────────────────────┬─┘
             └─ occurrence-count times ────────┘

►─┬────────────────────────────────────────────────────────────────────────►
  ├─ FOR ALL ◄ ────────────────────────────────────────────────────────────
  │                ┌──── , ─────┐            │
  └─ FOR ( ─▼─ device-code ─┴── ) ───────────┘

►──────────────────────────────────────────────────────────────────────────►
 ─┬───────────────────────────────────────────┬────────────────────────────
  └─ AT ─┬─ ANYwhere ─────────────────────────┤
         │  ┌──────────────────────────────┐  │
         └─▼─ ( row ─┬─ 1 ◄ ────────┬─ ) ──┴──┘
                     └─ column ─────┘

►──────────────────────────────────────────────────────────────────────────►
 ─┬─────────────────────────────────────────────┬──────────────────────────
  └─ ATTRibutes = ─┬─ NONE ─────────────────────┤
                   └─ ( attributes-list ) ──────┘

►──────────────────────────────────────────────────────────────────────────►
 ─┬──────────────────────────────────────────┬─────────────────────────────
  ├─ DELIMit ─┬────────┬─┬─ SKIP ◄ ─┬────────┤
  │           ├─ IS ───┤ └─ NOSKIP ─┘        │
  │           └─ = ────┘                     │
  └─ NODELIMit ──────────────────────────────┘

►──────────────────────────────────────────────────────────────────────────►
 ─┬─────────────────────────────────────────────────┬──────────────────────
  └─ PAGing type ─┬────────┬─┬─ DETail STart ──────┤
                  ├─ IS ───┤ ├─ DETail ONLY ───────┤
                  └─ = ────┘ ├─ DETail ENd ────────┤
                             ├─ FOOTer STart ──────┤
                             └─ NULl ◄ ────────────┘
```

```
►──────────────────────────────────────────────────────────────────────────┬─►
  └─ VALue ─┬──────┬─┬─ 'data-value' ──────────────────────────────────┬─┘
            ├─ IS ─┤ │          ┌──────────────────────────────────┐  │
            └─ = ──┘ └─( ─▼─┬─────────────────────┬─ 'data-value' ─┴─ )─┘
                             └─ (occurrence count) ┘

►─┬─ CURSOR ──────┬──────────────────────────────────────────────────────────►
  └─ NOCURSOR ◄ ──┘

►─┬─ LITeral ◄ ──────────────────────────┬───────────────────────────────────►◄
  ├─ MESSage LENgth ─┬─ length ─┬────────┤
  │                  └─ 80 ◄ ───┘        │
  ├─ PAGE LENgth 4 ──────────────────────┤
  ├─ RESPonse LENgth ─┬─ length ─┬───────┤
  │                   └─ 80 ◄ ───┘       │
  └─ DFld dfld-specifications ───────────┘
```

Expansion of dfld-specifications

```
►►── data-field-name ─┬──────────────────────────┬───────────────────────────►
                      └─ ( subscript-number ) ───┘

►─┬──────────────────────────────────────────────────────────┬───────────────►
  └─ OF ─┬─ record-name ─┬─────────────────────────────────┬─┤
         │               └─ VERsion version-number ────────┘ │
         └─ role-name ───────────────────────────────────────┘

►─┬──────────────────────────────────────────────────────────────────────────►
  └─ HELP ───────────────────────────────────────────────────────────────────

►────────────────────────────────────────────────────────────────────────────►
 ──── SOUrce ─┬─ NONE ───────────────────────────────────────────────────────
              └─ MODule module-name ─┬───────────────────────────────────────
                                     └─ VERsion ─┬──────┬─ version-number ─┘
                                                 ├─ IS ─┤
                                                 └─ = ──┘

►──────────────────────────────────────────────────────────────────────────┬─►
 ───────────────────────────────────────────────────────────────────────┬─┘
 ───────────────────────────────────────────────────────────────────────┘
   ┌─ HALF screen ◄─┐
   └─ FULL screen ──┘

►─┬─ REQuired ─────┬─────────────────────────────────────────────────────────►
  └─ OPTional ◄ ───┘

►─┬───────────────────────────────────────────────┬──────────────────────────►
  └─ REVerse NUMeric ─┬──────┬─┬─ Yes ─┬──────────┘
                      ├─ IS ─┤ └─ No ──┘
                      └─ = ──┘

►─┬────────────────────────────────────────────────────────────┬─────────────►
  └─ UNDERSCORE when blank ─┬─ No ◄ ───────────────────────────┘
                            └─ Yes ─┘

►─┬─ NOEDIT ◄ ───────────────────────────────────────────────────────────────►
  └─ EXTernal PICture ─┬──────┬─┬─ 'picture' ─┬──────────────────────────────
                       ├─ IS ─┤ └─ INTernal ──┘
                       └─ = ──┘
```


ZEROed ◄
RETAINed
when null
DISPlay ◄
BLANK
when zero
EDIT TABle
IS
=
NULL ◄
table-name
VERsion *version-number*
LINK ◄
NOLINK
USAGE is
VALIDate
INVALIDate
DEFault ◄
CODE TABle
IS
=
NULL ◄
table-name
VERSION *version-number*
LINK ◄
NOLINK
ERROR MESSage
'*message*'
message-id
NULL ◄
MSG PREFIX
IS
=
DC ◄
message-prefix
FOR INput
JUSTify
Left
Right
PAD
No ◄
with *literal*
DATA
YES ◄
NO
UPPERCASE
NO ◄
YES
EDIT
IS
=
edit-module-name
WITH AUTOedit
NO ◄
BEFore
AFTer
FOR OUTPUT
BACKscan
YES
NO ◄
DATA
YES ◄
NO
ERASE
ATTRibute
EDIT
IS
=
edit-module-name
WITH AUTOedit
NO ◄
BEFore
AFTer
.

Parameters

ADD/MODIFY

Specifies the action taken with regard to the MFLD statement. The DELETE verb cannot be specified for MFLD statements for MAP AUTOPANEL.

Note: For information about how to remove a field occurrencefrom a map occurrence that was defined by a MAP AUTOPANEL statement, see "Compiler Action Verbs (see page 194)."

MFLD

Introduces the clauses that define a map field and associated panel field occurrence.

OCCURS 1/occurrence-count TIMES

Specifies the number of times the field is to appear on the map; the default is 1.

FOR ALL /(*device-code-a*)

Associates the specified screen sizes with field specifications established by subsequent AT, ATTRIBUTES, DELIMIT, and VALUE clauses. If the map is used with more than one screen size, multiple FOR specifications can be included in the MFLD statement to establish different information for each screen size.

For a more detailed description of this latter use of the FOR specification, see "Defining Versions of Maps for Different Devices (see page 256)."

A field is associated with specific devices as follows:

ALL

Specifies that subsequent clauses of the MFLD statement apply to all screen sizes specified in the related DEVICES clause.

(Device-code-a)

Specifies one or more devices. Subsequent clauses of the MFLD statement apply only to the designated screen sizes. More than one *device-code-a* specification can be included in a FOR clause. The number of valid *device-code-a* specifications depends on the number of screen types declared in the related DEVICES specification.

Valid screen sizes are 12X40, 12X80, 24X80, 32X80, 43X80, and 27X132. Device specifications must be enclosed in parentheses and separated by commas; for example, FOR (12X40,12X80).

AT ANYWHERE/ (row,1/column)

Specifies the screen coordinate of the attribute byte for a field by row and column. The coordinate establishes the location of a runtime field on a given screen. An attribute byte is a nondisplayable character that precedes the displayed field and defines the field's attributes.

The field itself is displayed starting at the coordinate that immediately follows the nondisplayable attribute byte. For example, a field displays starting in coordinate (5,11) for an AT (5,10) specification.

The following considerations apply to the placement of attribute bytes and fields:

- Specifying the coordinates for the final column of a row places the first displayable character for a field in the first column of the next row.
- Specifying the coordinates for the final column of the final row on a screen places the first displayable character for a field in the first column of the first row (1,1).
- Specifying coordinates that cause a field to exceed the remaining length on a given row results in a field that is split at the end of the screen and wrapped around to either the next row or the top of the screen, depending on the row in which the coordinates were placed.

Screen coordinates are designated as follows:

ANYWHERE

Specifies that the field can appear anywhere on the screen. ANYWHERE is meaningful only with mapin operations for which the requesting program reads extraneous data. Extraneous data is data that is not associated with a field at a specific row/column location.

(Row-n,column-n)

Specifies the row and column coordinates for the attribute byte for the field:

- *Row-n* identifies a horizontal position on the screen.
- *Column-n* identifies a vertical position on the screen; the default column is 1.

The following considerations apply when positioning multiply-occurring fields:

- Each occurrence of the field requires its own *row-n*,*column-n* specification; multiple *row-n*,*column-n* specifications can be made in one AT clause, if necessary.
- If there are more *row-n*,*column-n* specifications than multiply-occurring fields specified in the OCCURS clause, the compiler input statement listing returns an error message.
- AT specifications can occur in any order; they are assigned to corresponding OCCURS values in order of iteration.

ATTRIBUTES=NONE/(attributes-list)

Specifies the attributes for the field. Only one ATTRIBUTES clause can occur in a given MFLD statement. ATTRIBUTES specifications apply to all occurrences of the field. Valid specifications are as follows:

- **NONE** removes all attribute specifications from the map and panel field occurrences being defined by the MFLD statement. The following runtime considerations apply when NONE is specified for a field:
  - The field is displayed with the attributes defined for the preceding field if the preceding field is not delimited.
  - The field is displayed with the default display attributes provided by the device if the preceding field is delimited or if there is no preceding field.
  - The field is displayed beginning in the column position specified in the AT clause, since there is no attribute character for the field.
- **(Attributes-list)** specifies a list of attribute that apply to the field.

  Note: For more information about available attributes, see "Attributes List (see page 201)."

DELIMIT/NODELIMIT

Specifies whether a delimit character is placed after the final position of a data field:

- **DELIMIT IS SKIP/NOSKIP** specifies that an internal delimit character is placed after the final position of the field, as determined by the external picture of the associated record element. The action of the cursor when it reaches the delimit character and the disposition of excess characters are determined by one of the following specifications:
  - **SKIP** (default) specifies that the cursor is advanced automatically to the start of the next UNPROTECTED field when operator input reaches the delimit character. If there are no more UNPROTECTED fields on the map, the cursor is placed at the start of the current field. Characters typed after the internal delimit character is reached are placed in the field to which the cursor advances. SKIP is the default if DELIMIT is specified.
  - **NOSKIP** specifies that the cursor remains at the delimit character when operator input reaches the end of the field. Subsequently typed input locks the keyboard until the operator presses the RESET key. The TAB key advances the cursor to the next UNPROTECTED field.
- **NODELIMIT** specifies that no internal delimit character is assigned to the field. The operator is not informed when input reaches the end of the field, and can continue typing until the attribute byte of the next field is reached. On mapin, the external picture of the record element associated with the field determines the amount of operator input that is stored. Input that exceeds the length of the external picture is ignored; a CA ADS dialog or application program can include commands to inquire whether extraneous data has been input for a NODELIMIT field.

PAGING TYPE IS DETAIL ONLY/START/DETAIL END/FOOTER START/NULL

(pageable maps only). Specifies whether the field begins and/or ends the detail occurrence or an area on the pageable map:

- *DETAIL ONLY* performs the following functions:
  - *Begins the detail area* on the line that contains the attribute byte of the field being defined.
  - *Begins the detail occurrence* on the line that contains the attribute byte of the field being defined.
  - *Ends the header area* (if any) on the line immediately above the line that contains the attribute byte for the field assigned the DETAIL ONLY specification.
  - *Ends the detail occurrence* for the map at the final character of the field assigned the DETAIL ONLY specification.

  The field assigned the DETAIL ONLY specification must begin on a new line (that is, it cannot begin on a line that contains characters for a field in the header area).
- *DETAIL START* performs the following functions:
  - *Begins the detail area* on the line that contains the attribute byte of the field being defined.
  - *Begins the detail occurrence* on the line that contains the attribute byte of the field being defined.
  - *Ends the header area* (if any) on the line immediately above the line that contains the attribute byte for the field assigned the DETAIL START specification.

  The field assigned the DETAIL START specification must begin on a new line (that is, it cannot begin on a line that contains characters for a field in the header area).
- *DETAIL END* specifies that the detail occurrence for the map is to end at the final character position of the current field. The detail area for the map is not terminated by DETAIL END; FOOTER START (below) can be used to terminate the detail area.
- *FOOTER START* performs the following functions:
  - *Begins the footer area* on the line that contains the attribute for the field. The footer area ends at the end of the screen.
  - *Ends the detail area* on the line immediately above the line that contains the attribute byte of the field assigned the FOOTER START specification.

The field assigned the FOOTER START specification must begin on a new line (that is, it cannot begin on a line that contains characters for a field in the DETAIL END field). If assigned, the FOOTER START specification must be made for a field following the field assigned the DETAIL END specification.

- *NULL* (default) specifies that the field does not begin or end a detail or an area on the map. The NULL setting can be used to override a previous DETAIL START, DETAIL END, or FOOTER START specification for a field.

Note: For more information about the areas of pageable maps, see "Pageable Maps (see page 91)."

VALUE IS data-value/((occurrence-count) data-value)

Supplies string values to literal fields:

- **Data-value** assigns a value to a singly-occurring literal field. The specified value must be enclosed in quotation marks.
- **((Occurrence-count) data-value)** assigns a value to a literal field or assigns discrete values to multiple occurrences of a literal field:
  - **(Occurrence-count)** identifies one or more field occurrences by order of iteration. Values are assigned to the literal fields specified in the AT clause by order of iteration of the row/column specifications rather than by their order of display on the mapped screen. For example, with a literal field that occurs four times, the specification VALUE IS ((2) 'ABC' (1) 'DEF' (1) 'GHI') assigns the value ABC to the first and second field occurrences, DEF to the third field occurrence, and GHI to the fourth field occurrence.
  - **Data-value** specifies the value assigned to each occurrence or group of literal field occurrences. The supplied value must be enclosed in quotation marks.

A maximum of 256 characters can be specified for a literal field in *data-value*

CURSOR/NOCURSOR

Specifies the mapout location of the cursor at runtime:

- **CURSOR** specifies that the cursor is located at the start of the indicated field when the map is mapped out.
- **NOCURSOR** (default) specifies that the cursor is not located at the start of the indicated field when the map is mapped out.

If CURSOR is specified in more than one MFLD statement for a given map, the runtime cursor is positioned at the field for which CURSOR was last specified at compile time. If CURSOR is not specified for any field on the map, the default runtime cursor location is the first UNPROTECTED field on the screen, or at coordinate 1,1 if there are no UNPROTECTED fields.

LITERAL

Specifies that the field is a literal field. A literal field is given a string value in the VALUE IS clause. If LITERAL, MESSAGE LENGTH, PAGE LENGTH, RESPONSE LENGTH, or DFLD is not specified in the MFLD statement, LITERAL is the default.

MESSAGE LENGTH length

Specifies that the field is a message field. If included, *length* must be an integer greater than or equal to 1 and less than or equal to the total number of character positions on the smallest screen for which the map is intended. The default for *length-n* is 80.

PAGE LENGTH 4

Specifies that the field is a page field.

RESPONSE LENGTH length

CA ADS only. Specifies that the field is a response field. If included, *length* must be an integer in the range 1 through 32. The default is 8.

DFLD data-field-name (subscript) OF record-name/role-name

Specifies that the field is a data field and associates the field with the record element named by *data-field-name*:

- *Data-field-name* specifies an element that is already defined to the data dictionary by means of the CA IDMS/DB schema compiler or the IDD DDDL compiler.
- *Subscript-n* specifies the subscript of the record element if the element is multiply-occurring.
- *OF record-name/role-name* names the record or role to which the associated element belongs:
  - *Record-name VERSION version-n* specifies the name of a schema or work record that is already defined in the data dictionary and is specified in the USING RECORDS clause for the map. A record name for an element must be specified when the same *data-field-name* occurs in more than one record used by the map.

    VERSION version-n must be used to specify the version number of the record if *record-name* is not unique in the map.
  - *Role-name* specifies the role name of a record. The role name can be previously defined for a record in the subschema used in the program or dialog, or the role name can be a unique name that is established at map definition in the ROLENAME clause of the MAP statement.

The length of a data field is not specified in the DFLD clause; the length is determined in one of the following ways:

- *The EXTERNAL PICTURE clause of the MFLD statement* (described as follows) can determine the length of a data field, as follows:
  - *If EXTERNAL PICTURE explicitly specifies an external picture* for the field, that external picture determines the length of the field.
  - *If EXTERNAL PICTURE specifies INTERNAL*, an external picture is constructed from the internal picture specified for the element named by *data-field-name*, and that external picture determines the length of the field.
- *The external picture associated with the record element* (if any) determines the length of the field if the MFLD statement does not specify an external picture.
- *The external picture derived from the internal picture specified for the field* (if automatic editing is enabled for the map and field) determines the length of the field if the record element definition does not specify an external picture.

  A data field can contain as many character positions as are available on the smallest screen for which the map is intended, minus one character position for the attribute byte for the field.

Remaining MFLD clauses supply automatic editing and error-handling information for the field being defined. For more information about the features enabled by the clauses listed, see the chapter "Automatic Editing and Error Handling." The following MFLD clauses apply only when DFLD is specified for the field:

Note: For more information about the features enabled by the clauses listed, see the chapter "Automatic Editing and Error Handling."

HELP

Specifies whether help will be implemented for the field.

SOUrce NONE/MODule module-name

The name of the IDD module that contains the help text for the field.

If module name is specified, you can optionally specify:

- The version number
- Whether the help is displayed on a full or half screen

REQUIRED/OPTIONAL

Indicates whether operator input is required in the field:

- **REQUIRED** specifies that input is required. An input error occurs if the terminal operator does not enter data for the field.
- **OPTIONAL** (default) specifies that input is optional.

REVERSE NUMERIC IS YES/NO

Specifies whether the contents of a numeric field are reversed on mapin and again on mapout:

- **YES** specifies that data for the field is reversed on mapin, and again on mapout. REVERSE NUMERIC is used for numeric fields when hardware modifications cause input to be entered from right to left. YES is the default for new fields when REVERSE NUMERIC is specified in the SIGNON statement for the batch run.
- **NO** specifies that data for the field is not reversed on mapin or on mapout. The NO specification is overridden when the developer specifies REVERSE NUMERIC in the SIGNON statement for the batch run.

UNDERSCORE when blank NO/YES

Indicates if the field should be underscored if it is blank. On mapin, trailing underscores are removed.

EXTERNAL PICTURE IS/NOEDIT

Indicates whether the field is processed by automatic editing and error-handling, and establishes an external picture for use in editing. The following options are available:

- **NOEDIT** (default) disables automatic editing and error-handling for the field.
- **EXTERNAL PICTURE IS** enables automatic editing for the field and specifies the external picture for the field during automatic editing, the action to be taken by automatic editing should the terminal operator erase the field, and/or the display format for NUMERIC fields that contain only zeros, as follows:
  - **Picture/INTERNAL** specifies the external picture for the field.

Note: For information about external pictures, see External Pictures.

Picture specifies an actual external picture, such as XX/XX/XX or XXX-XX-XXXX.

- **INTERNAL** (default) requests that the map use the external picture defined for the associated record element (or the picture constructed for the field).
- **ZEROED/RETAINED WHEN NULL** specifies the action taken when automatic editing is enabled for a numeric field and the terminal operator erases the contents of the field (for example, by pressing the ERASE EOF key).

  ZEROED (default) requests the field be filled with zeros of the appropriate data type when automatic editing is enabled for the given map and field and the operator erases the contents of the field.

  RETAINED indicates that the data contained in the buffer should be retained when the operator erases the contents of the field.
- **DISPLAY/BLANK WHEN ZERO** indicates the action taken when automatic editing is enabled for a numeric field that contains only zeros.

  DISPLAY (default) requests that the zeros be displayed.

  BLANK requests that blanks be displayed instead of zeros.

Note: For a group data field, the only valid external picture is an alphanumeric one. Note also that edit and code tables are not supported for group data fields.

EDIT TABLE IS table-name/NULL

Specifies the edit table used for the field if automatic editing is enabled for the field and map:

■ **Table-name** specifies the name of an existing stand-alone table used as the edit table for the field.

Note: For more information about the use of edit tables, see "Edit and Code Tables".

If editing is enabled elsewhere for the field and no edit table is named by using the EDIT TABLE IS clause, the default edit table is the built-in table (if any) defined in the associated record element. If no edit table is named by the EDIT TABLE IS clause or defined in the element definition, no edit table is used. The following clauses can be included in an EDIT TABLE IS specification when the edit table is named:

- **VERSION version-n** specifies the version of the edit table used. The default is 1.
- **LINK/NOLINK** specifies whether the edit table is linked as part of the map load module or is loaded dynamically at runtime:

  LINK (default) indicates that the named edit table is to be linked as part of the map load module. The LINK specification is particularly useful for tables that contain items that cannot be used readily by another record element.

NOLINK indicates that the table is loaded dynamically at runtime. NOLINK is useful when the contents of a table change frequently.

- **USAGE IS VALIDATE/INVALIDATE/ DEFAULT** indicates whether the edit table is a table of valid values or invalid values:

  VALIDATE specifies that the table contains valid values. An error occurs when the terminal operator enters a value that does not appear in the table.

  INVALIDATE specifies that the table contains invalid values. An error occurs when the terminal operator enters a value that appears in the table.

  DEFAULT (default) specifies that the VALID or INVALID specification given in the table definition should be used for the table.

■ **NULL** (default) specifies that no stand-alone edit table is used for the field at runtime. NULL does not suppress use of the built-in edit table for the field.

CODE TABLE IS table-name/NULL

Specifies the code table used for the field if automatic editing is enabled for the field and map:

■ **Table-name** specifies the name of an existing stand-alone code table used for the field if automatic editing is enabled for the field.

Note: For more information about the use of code tables, see "Edit and Code Tables".

If editing is enabled elsewhere for the field, and no code table is named by using the CODE TABLE IS clause, the default code table is the built-in code table (if any) defined in the associated record element. If no code table is named by the CODE TABLE IS clause or defined in the element definition, no edit table is used. The following clauses can be included in a CODE TABLE IS specification when a code table is named:

- **VERSION version-n** specifies the version of the code table used. The default is one.
- **LINK/NOLINK** specifies whether the code table is linked as part of the map load module or is loaded dynamically at run time:

  LINK (default) indicates that the named code table is to be linked as part of the map load module. LINK is useful for tables that contain items that cannot be used readily by another record element.

  NOLINK indicates that the table is loaded dynamically at runtime. NOLINK is useful when the contents of a table change frequently.

■ **NULL** (default) specifies that no stand-alone code table is used for the field at runtime. NULL does not suppress use of the built-in code table for the field.

ERROR MESSAGE message/message-id/NULL

Defines the error message returned on mapout if the field is in error. The message is displayed in the message field defined for the map. If the map has no message field, the message is not displayed and processing continues normally (with CA ADS, the message is displayed on the CA ADS default message screen). The developer can specify any one of the following options:

- **Message** specifies the text of the message displayed if the field is in error. *Message* must be enclosed in quotation marks.
- **Message-id** specifies the 6-digit message identifier of the data dictionary message displayed if the field is in error. The map compiler adds the prefix DC to this 6-digit identifier to construct the actual identifier of the data dictionary message.
- **NULL** (default) specifies that the default error message is used if the field contains incorrect input. The default message has the following format:

  ERROR AT *row,column*

MSG PREFIX IS message-prefix

Defines the two-character prefix to be used to locate the message in the dictionary of the message defined in the previous ERROR MESSAGE parameter when the field is found to be in error at runtime. The value defaults to the value specified in the MSG PREFIX parameter in the MAP statement.

FOR INPUT

Specifies functions performed for data on a mapin operation:

JUSTIFY LEFT/RIGHT

Specifies how operator input is to be aligned for transmission to program variable storage:

- **LEFT** (default) specifies that input is left-justified.
- **RIGHT** specifies that input is right-justified.

PAD NO/WITH literal

Specifies a pad character for an alphanumeric field in character or hexadecimal format:

- **NO** (default) specifies that data is not padded.
- **WITH literal** specifies the pad character for a field in character or hexadecimal format. Character literals are specified as C'*c*', where *c* denotes a character literal. Pad characters in hexadecimal format are specified as X'*nn*', where *nn* denotes a two-digit hexadecimal value. Hexadecimal format is recommended when specifying the blank character as the pad character for a field.

Pad characters are used to avoid unwanted data being stored for a field on mapin. For example, a data field containing the name JOHNSON is mapped out. The operator presses the ERASE EOF key to erase the field, types in SMITH, and presses Enter. The value mapped to variable storage depends on whether a pad character is defined for the field:

- **If no pad character is defined for the field**, SMITHON is stored for the field. The operator would have to key blanks over ON to eliminate these characters from the data.
- **If a blank pad character is defined for the field**, SMITH is stored for the field.

DATA YES/NO

Indicates whether the transmitted contents (if any) of the field are to be moved automatically into program variable storage on a mapin operation:

- **YES** (default) specifies that the contents of the field are automatically moved into program variable storage if transmitted from the terminal. Data is transmitted from the terminal when the MDT is set on for the field before a mapin operation.
- **NO** specifies that data contained in the field is not moved automatically into program variable storage, even if the MDT is set on.

UPPERcase YES/NO

Indicates if field should be displayed in uppercase.

EDIT IS edit-module-name WITH AUTOEDIT

Optionally specifies the name of an existing user-written edit module to process input after transmission on a mapin operation. The relationship between the user edit module and automatic editing is determined by the WITH AUTOEDIT clause as follows:

- **NO** (default) specifies that automatic editing is not performed for the field; only the user-written edit module is used.
- **BEFORE** specifies that automatic editing is performed immediately before the user edit module edits the input.
- **AFTER** specifies that automatic editing is performed immediately after the user edit module edits the input.

User edit modules are discussed in "Automatic Editing and Error Handling".

FOR OUTPUT

Specifies functions to be performed for data prior to a mapout operation:

BACKSCAN YES/NO

Indicates whether trailing blanks are to be eliminated from the field prior to display:

- **YES** specifies that the contents of the field are displayed without trailing blanks (if any). Old data may remain in the field after operator alterations if NEWPAGE is set to NO in either the CA ADS sysgen statement or the DML statement that issues the mapout.
- **NO** (default) specifies that the contents of the field are displayed with trailing blanks (if any).

DATA YES/NO/ERASE/ATTRIBUTE

Indicates whether data in program variable storage is to be transmitted to the screen on a mapout operation:

- **YES** (default) specifies that data is transmitted.
- **NO** specifies that neither data nor the attribute byte are transmitted. Any data previously in the field continues to display.
- **ERASE** specifies that data is not transmitted; the field on the screen is initialized to null or low values, depending on whether the field is numeric or alphanumeric.
- **ATTRIBUTE** specifies that only the attribute byte for the field is transmitted; data in the record buffer is not sent to the terminal.

EDIT IS edit-module-name WITH AUTOEDIT

Optionally specifies the name of an existing user-written edit module to process data before display on a mapout operation. The relationship between the user edit module and automatic editing is determined by the WITH AUTOEDIT clause as described as follows:

- **NO** (default) specifies that automatic editing is not performed; only the user-written edit module is used.
- **BEFORE** specifies that automatic editing is performed immediately before the user edit module edits the field.
- **AFTER** specifies that automatic editing is performed immediately after the user edit module edits the field.

User-written edit modules are discussed in the chapter "Automatic Editing and Error Handling."

Examples

The following examples illustrate use of the MFLD statement.

Example 1

Adding a Map Field Occurrence to a Map Occurrence

The sample MAP AUTOPANEL and MFLD statements add two fields to the MEALS map as shown:

```
ADD MAP MEALS VERSION IS 10
    AUTOPANEL DEVICES=(24X80)
      USING RECORDS MEALS-REC VERSION 1
      NOEDIT.

    ADD MFLD
        AT (7,7)
        ATTRIBUTES (PROTECTED BRIGHT)
        VALUE IS 'FIRST CLASS'.

    ADD MFLD
        AT (10,7)
        ATTRIBUTES (PROTECTED BRIGHT)
        VALUE IS 'TOURIST CLASS'.
```

Example 2

Deleting a Map Field Occurrence from a Map Occurrence

The MAP AUTOPANEL and MFLD statements eliminate the TOURIST CLASS field from the MEALS map as shown:

```
MOD MAP MEALS VERSION IS 10
    AUTOPANEL DEVICES=(24X80)
    USING RECORDS MEALS-REC VERSION 1
    EDIT
    ON ERROR
       INCORRECT ATTRIBUTES (BRIGHT)
       CORRECT ATTRIBUTES (DISPLAY).

  ADD MFLD
      AT (7,7)
      ATTRIBUTES (PROTECTED BRIGHT)
      VALUE IS 'FIRST CLASS'.
```

The FIRST CLASS map and panel fields are retained for the MEALS map.

Statements for Manual Panel Definition

The developer explicitly defines panel, panel field, map, and map field occurrences when using compiler statements that manually define panels. The compiler action verbs ADD, MODIFY, and DELETE define the overall purpose of the mapping statements.

Note: For information about the ADD, MODIFY, and DELETE verbs, see "Compiler Action Verbs".

Statements You can use

- The *PANEL statement* defines and generates a panel occurrence.
- The *PFLD statement* defines and generates a panel field occurrence.
- The *MAP statement* defines and generates a map occurrence that is associated with a specific panel occurrence.
- The *MFLD statement* defines and generates a map field occurrence. The map field occurrence is associated with a specific panel field occurrence from the panel occurrence named by the owner MAP statement.

Conditions

A PANEL statement must be followed immediately by the PFLD statements that define its related fields. A panel occurrence defined by PANEL and PFLD statements must exist in the data dictionary before MAP and MFLD statements can be used to generate a map occurrence based on the panel record. A MAP statement must be followed immediately by the MFLD statements that define its related fields.

The PANEL, PFLD, MAP, and MFLD statements and their clauses are presented separately as follows:

PANEL Statement Syntax

The PANEL statement typically is used to perform the following functions:

- Create or maintain a panel occurrence in the data dictionary
- Identify the panel occurrence with a unique combination of name and version number
- Specify the particular devices suitable for use with the panel

PFLD statements that immediately follow a PANEL statement establish fields for that panel; a panel generally has several panel fields.

```
►►─┬──────────┬─ PANel panel-name ─┬──────────────────────────────────┬─►
   ├─ ADD ────┤                    └ VERsion ─┬──────┬─ version number ┘
   ├─ MODify ─┤                               ├─ IS ─┤
   └─ DELete ─┘                               └─ = ──┘

 ►── DEVices = ─┬─ ( device-code ) ────────────────────────── . ──────────►◄
                ├─ (24x80, 32x80, 43x80, 27x132) ◄─┤
                └─ ALL ────────────────────────────┘
```

Parameters

ADD/MODIFY/DELETE

Specifies the action taken with regard to the PANEL specification. For information about these verbs, see "Compiler Action Verbs." ADD, MODIFY, and DELETE access for a panel is subject to security restrictions specified for the batch compiler and individual maps, as outlined in "Compiler Security".

PANEL panel-name

Supplies a 1- through 32-character name for the panel. The following considerations apply to the composition.

alphanumeric or special characters. character; for example, pound sign (#), at sign (@), or dollar sign ($). or blank characters.

VERSION IS version

Optionally specifies a version number to further identify the map. *Version* must be in the range 1 through 9999. If omitted, *version* defaults to the data dictionary default as defined by the Data Dictionary Definition Language (DDDL) SET OPTIONS statement.

DEVICES=(device-code)/(24X80,32X80,43X80,27X132)/ALL

Specifies the devices with which the map can be used:

- (**Device-code-a**) specifies a device (screen size) with which the map can be used.

  Valid screen sizes are 12X40, 12X80, 24X80, 32X80, 43X80, and 27X132. Commas must be used to separate *device-code-a* specifications when more than one device is specified. Device specifications in the DEVICES clause must be enclosed in parentheses; for example, DEVICES=(12X40,24X80,43X80).

- **(24X80,32X80,43X80,27X132)** is the default specification given to the map.

- **ALL** specifies that the map can be used with all valid screen sizes.

Using a Field on a Subset of Devices

To reserve a map field for use on only a subset of the devices specified in the DEVICES clause, FOR clauses can be included in a PFLD statement for the field. FOR clauses are also used to specify values or attributes to be used when a field is displayed on specific devices.

Effect of the MODIFY Verb

The MODIFY verb does not update the DEVICES specification.

Note: For more information, see "Compiler Action Verbs".

Examples

The following are the examples of the PANEL statement:

Example 1

Adding a Panel Occurrence

The sample PANEL statement adds the MEALS-PANEL panel occurrence to the data dictionary as shown:

```
     ADD PANEL MEALS-PANEL VERSION IS 2
         DEVICES=(24X80).
```

The sample MEALS-PANEL panel defined by this sample PANEL statement is defined for 24X80 screens.

Example 2

Modifying a Panel Occurrence

The sample PANEL statement modifies the MEALS-PANEL panel in the data dictionary as shown:

```
     MOD PANEL MEALS-PANEL VERSION IS 2
         DEVICES=(24X80,43X80).
```

Existing panel fields (if any) defined for the MEALS-PANEL panel are retained by the panel.

Example 3

Deleting a Panel Occurrence

The sample PANEL statement deletes the MEALS-PANEL panel and all associated panel fields (if any) from the data dictionary as shown:

```
     DEL PANEL MEALS-PANEL VERSION IS 2.
```

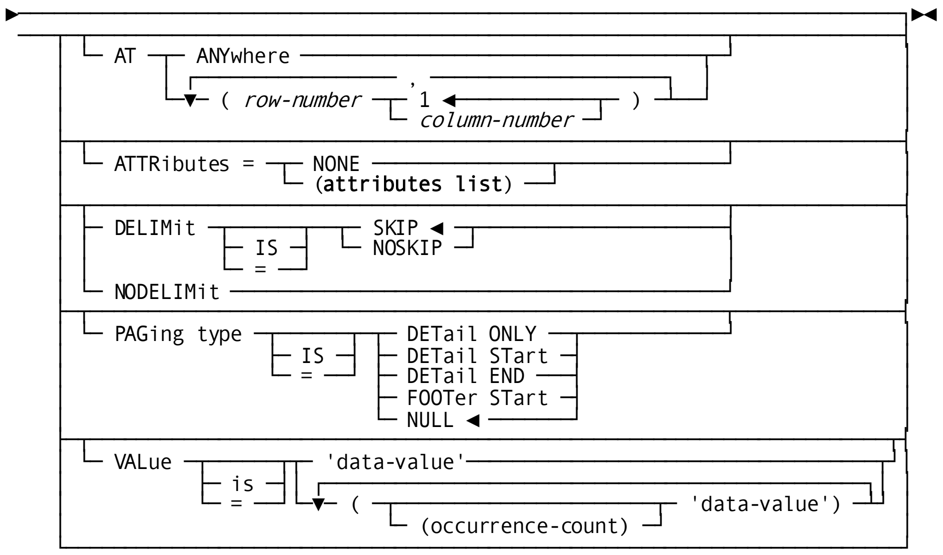
PFLD Statement Syntax

Functions Performed

A PFLD statement typically is used to perform the following functions:

- Create and maintain a single panel field occurrence for the panel established in the most recent PANEL statement.
- Identify the panel field occurrence in the data dictionary with a name that is unique within the owner panel occurrence.
- Specify characteristics for the field, such as the following:
  - Field occurrences for multiply-occurring fields
  - Screen locations of the panel field by row and column
  - Physical attributes of the field, such as display color
  - Field values for literal fields
  - Delimit characteristics

```
►►─┬──────────┬─ PFld panel-field-name ──────────────────────────►
   ├─ ADD ────┤
   ├─ MODify ─┤
   └─ DELete ─┘

 ►─┬─────────────────────────────────────────────┬──────────────►
   └─ OCCURS ─┬─ 1 ◄ ─────────────────────┬──────┘
              └─ occurrence-count times ──┘

   ┌───────────────────────── , ──────────────────────────────┐
 ►─▼─┬──────────────────────────────────────────────────────────┴─►
     ├─ FOR ALL ◄ ──────────────────────────────────────────────
     └─ FOR ( ─┬──── , ─────┐── ) ─┘
               └─ device-code ) ─┘
```



Parameters

ADD/MODIFY/DELETE

Specifies the action taken with regard to the MFLD statement.

PFLD

Introduces the clauses that define a panel field and associated panel field occurrence.

OCCURS 1/occurrence-count TIMES

Specifies the number of times the field is to appear on the panel; the default is 1.

FOR ALL /(*device-code-a*)

Associates the specified screen sizes with field specifications established by subsequent AT, ATTRIBUTES, DELIMIT, and VALUE clauses. If the panel is used with more than one screen size, multiple FOR specifications can be included in the PFLD statement to establish different information for each screen size; for a more detailed description of this latter use of the FOR specification, see "Defining Versions of Maps for Different Devices".

A field is associated with specific devices as follows:

- **ALL** specifies that subsequent clauses of the PFLD statement apply to all screen sizes specified in the related DEVICES clause.
- **(Device-code-a)** specifies one or more devices. Subsequent clauses of the PFLD statement apply only to the designated screen sizes. More than one *device-code-a* specification can be included in a FOR clause. The number of valid *device-code-a* specifications depends on the number of screen types declared in the related DEVICES specification.

  Valid screen sizes are 12X40, 12X80, 24X80, 32X80, 43X80, and 27X132. Device specifications must be enclosed in parentheses and separated by commas; for example, FOR (12X40,12X80).

AT ANYWHERE/ (row,1/column)

Specifies the screen coordinate of the attribute byte for a field by row and column. The coordinate establishes the location of a runtime field on a given screen. An attribute byte is a nondisplayable character that precedes the displayed field and defines the field's attributes.

The field itself is displayed starting at the coordinate that immediately follows the nondisplayable attribute byte. For example, a field displays starting in coordinate (5,11) for an AT (5,10) specification.

The following considerations apply to the placement of attribute bytes and fields:

- Specifying the coordinates for the final column of a row places the first displayable character for a field in the first column of the next row.
- Specifying the coordinates for the final column of the final row on a screen places the first displayable character for a field in the first column of the first row (1,1).
- Specifying coordinates that cause a field to exceed the remaining length on a given row results in a field that is split at the end of the screen and wrapped around to either the next row or the top of the screen, depending on the row in which the coordinates were placed.

Screen coordinates are designated as follows:

- **ANYWHERE** specifies that the field can appear anywhere on the screen. ANYWHERE is meaningful only with mapin operations for which the requesting program reads extraneous data. Extraneous data is data that is not associated with a field at a specific row/column location.
- **(Row-n,column-n)** specifies the row and column coordinates for the attribute byte for the field:
  - *Row-n* identifies a horizontal position on the screen.
  - *Column-n* identifies a vertical position on the screen; the default column is 1.

The following considerations apply when positioning multiply-occurring fields:

- Each occurrence of the field requires its own *row-n*,*column-n* specification; multiple *row-n*,*column-n* specifications can be made in one AT clause, if necessary.
- If there are more *row-n*,*column-n* specifications than multiply-occurring fields specified in the OCCURS clause, the compiler input statement listing returns an error message.
- AT specifications can occur in any order; they are assigned to corresponding OCCURS values in order of iteration.

ATTRIBUTES=NONE/(attributes-list)

Specifies the attributes for the field. Only one ATTRIBUTES clause can occur in a given MFLD statement. ATTRIBUTES specifications apply to all occurrences of the field. Valid specifications are as follows:

- **NONE** removes all attribute specifications from the map and panel field occurrences being defined by the MFLD statement. The following runtime considerations apply when NONE is specified for a field:
  - The field is displayed with the attributes defined for the preceding field if the preceding field is not delimited.
  - The field is displayed with the default display attributes provided by the device if the preceding field is delimited or if there is no preceding field.
  - The field is displayed beginning in the column position specified in the AT clause, since there is no attribute character for the field.
- **(Attributes-list)** specifies a list of attributes that apply to the field.

  Note: For more information about available attributes, see "Attributes List (see page 201)".

DELIMIT/NODELIMIT

Specifies whether a delimit character is placed after the final position of a data field:

- **DELIMIT IS SKIP/NOSKIP** specifies that an internal delimit character is placed after the final position of the field, as determined by the external picture of the associated record element. The action of the cursor when it reaches the delimit character and the disposition of excess characters are determined by one of the following specifications:
  - **SKIP** (default) specifies that the cursor is advanced automatically to the start of the next UNPROTECTED field when operator input reaches the delimit character. If there are no more UNPROTECTED fields on the map, the cursor is placed at the start of the current field. Characters typed after the internal delimit character is reached are placed in the field to which the cursor advances. SKIP is the default if DELIMIT is specified.
  - **NOSKIP** specifies that the cursor remains at the delimit character when operator input reaches the end of the field. Subsequently typed input locks the keyboard until the operator presses the RESET key. The TAB key advances the cursor to the next UNPROTECTED field.
- **NODELIMIT** specifies that no internal delimit character is assigned to the field. The operator is not informed when input reaches the end of the field, and can continue typing until the attribute byte of the next field is reached. On mapin, the external picture of the record element associated with the field determines the amount of operator input that is stored. Input that exceeds the length of the external picture is ignored; a CA ADS dialog or application program can include commands to inquire whether extraneous data has been input for a NODELIMIT field.

PAGING TYPE IS DETAIL ONLY/START/DETAIL END/FOOTER START/NULL

(pageable maps only) Specifies whether the field begins and/or ends the detail occurrence or an area on the pageable map:

- **DETAIL ONLY** performs the following functions:
  - *Begins the detail area* on the line that contains the attribute byte of the field being defined.
  - *Begins the detail occurrence* on the line that contains the attribute byte of the field being defined.
  - *Ends the header area* (if any) on the line immediately above the line that contains the attribute byte for the field assigned the DETAIL ONLY specification.
  - *Ends the detail occurrence* for the map at the final character of the field assigned the DETAIL ONLY specification.

  The field assigned the DETAIL ONLY specification must begin on a new line (that is, it cannot begin on a line that contains characters for a field in the header area).

- **DETAIL START** performs the following functions:
  - *Begins the detail area* on the line that contains the attribute byte of the field being defined.
  - *Begins the detail occurrence* on the line that contains the attribute byte of the field being defined.
  - *Ends the header area* (if any) on the line immediately above the line that contains the attribute byte for the field assigned the DETAIL START specification.

  The field assigned the DETAIL START specification must begin on a new line (that is, it cannot begin on a line that contains characters for a field in the header area).
- **DETAIL END** specifies that the detail occurrence for the map is to end at the final character position of the current field. The detail area for the map is not terminated by DETAIL END; FOOTER START (below) can be used to terminate the detail area.
- **FOOTER START** performs the following functions:
  - **Begins the footer area** on the line that contains the attribute for the field. The footer area ends at the end of the screen.
  - **Ends the detail area** on the line immediately above the line that contains the attribute byte of the field assigned the FOOTER START specification.

  The field assigned the FOOTER START specification must begin on a new line (that is, it cannot begin on a line that contains characters for a field in the DETAIL END field). If assigned, the FOOTER START specification must be made for a field below the field assigned the DETAIL END specification.
- **NULL** (default) specifies that the field does not begin or end a detail or an area on the map. The NULL setting can be used to override a previous DETAIL START, DETAIL END, or FOOTER START specification for a field.

Note: For more information about the areas of pageable maps, see "Pageable Maps".

VALUE IS data-value/((occurrence-count) data-value)

Supplies string values to literal fields:

- **Data-value** assigns a value to a singly-occurring literal field. The specified value must be enclosed in quotation marks.
- **((Occurrence-count) data-value)** assigns a value to a literal field or assigns discrete values to multiple occurrences of a literal field:
  - **(Occurrence-count)** identifies one or more field occurrences by order of iteration. Values are assigned to the literal fields specified in the AT clause by order of iteration of the row/column specifications rather than by their order of display on the mapped screen. For example, with a literal field that occurs four times, the specification VALUE IS ((2) 'ABC' (1) 'DEF' (1) 'GHI') assigns the value ABC to the first and second field occurrences, DEF to the third field occurrence, and GHI to the fourth field occurrence.
  - **Data-value** specifies the value assigned to each occurrence or group of literal field occurrences. The supplied value must be enclosed in quotation marks.

A maximum of 256 characters can be specified for a literal field in *data-value*.

Examples

The following examples illustrate use of the PFLD statement.

Example 1

Adding a Panel Field Occurrence to a Panel Occurrence

The sample PANEL and PFLD statements add the literal FIRST-CLASS panel field to the MEALS-PANEL panel as shown:

```
ADD PANEL MEALS-PANEL VERSION IS 2
    DEVICES=(24X80).

    ADD PFLD FIRST-CLASS
        AT (7,7)
        ATTRIBUTES (PROTECTED BRIGHT)
        VALUE IS 'FIRST CLASS'.
```

Example 2

Modifying a Panel Occurrence

The sample PANEL and PFLD statements add the MEALS-HEAD2 panel field to the MEALS-PANEL panel as shown:

```
MOD PANEL MEALS-PANEL VERSION IS 2.

    ADD PFLD MEALS-HEAD2
        AT (3,25)
        ATTRIBUTES (PROTECTED BRIGHT)
        VALUE IS 'MEALS SELECTION SCREEN'.
```

Panel fields (if any) that are already defined for MEALS-PANEL are retained by the panel occurrence.

MAP Statement Syntax

Functions Performed

A MAP statement typically is used to perform the following functions:

- Create or maintain a map occurrence in the data dictionary
- Identify the map occurrence with a unique combination of name and version number
- Identify the existing panel occurrence on which the map is based
- Identify the records or roles referenced by map data fields
- Enable global automatic editing and error-handling, specifying correct-field and incorrect-field attributes
- Specify various terminal hardware control functions (such as alarm or numeric options) to be invoked during mapout operations

The actual map is constructed with MFLD statements, presented later in this section, by selecting fields from the panel occurrence named in the MAP statement and by associating the variable panel fields with record elements defined in the data dictionary.

```
►►─┬──────────┬─ MAP map-name ─┬──────────────────────────────┬──────────►
   ├─ ADD ────┤                └─ VERsion ─┬──────┬─ version ─┘
   ├─ MODIFY ─┤                            ├─ IS ─┤
   └─ DELETE ─┘                            └─ = ──┘

 ►─┬─────────────────────────────────────────────┬────────────────────────►
   └─ DATETIME ─┬──────┬─ date-time-stamp ───────┘
                ├─ IS ─┤
                └─ = ──┘

 ►─┬─────────────────────────────────────────────────┬────────────────────►
   └─ MSG PREFIX ─┬──────┬─┬─ DC ◄ ─────────────┬────┘
                  ├─ IS ─┤ └─ message-prefix ───┘
                  └─ = ──┘

 ►─┬────────────────────────────────────────────────────────────────┬─────►
   └─ PANel ─┬──────┬─ panel-name ─┬───────────────────────────────┬┘
             ├─ IS ─┤              └─ VERsion ─┬──────┬─ version ──┘
             └─ = ──┘                          ├─ IS ─┤
                                               └─ = ──┘

 ►─┬──────────────────────────────────────┬─┬─ RESident ──────┬───────────►
   └─ SYStem ─┬──────┬─ dc-version ───────┘ └─ NONRESident ◄ ─┘
              ├─ IS ─┤
              └─ = ──┘

 ►─┬──────────────────────────────────────────────────────────────────────►
   └─ USING ─┬──────────────┬─────────────────────────────────────────────
             └─┬─ RECORDS ──┤
               └─ REC ──────┘

 ►─────────────────────────────────────────────────────────────────────┬──►
                ┌──────────────────────────────────────────────────┐   │
 ─────── ( ─▼─ record-name ─┬────────────┬─┬────────────────────────┬┴─ ) ─┘
                            └─ version ──┘ └─ ROLEname role-name ───┘

 ►─┬─ EDIT ◄ ──┬─┬─────────────────────────────────┬──────────────────────►
   └─ NOEDIT ──┘ └─ CURSOR at panel-field-name ────┘

 ►─┬─────────────────────────────────────┬─┬──────────────────────────────┬►
   ├─ RESET ◄ ──┬─┬─────────────────┬────┘ ├─ LOCK ────┬─┬────────────────┤
   └─ NORESET ──┘ └─┬─ MODIFIED ────┤      └─ UNLOCK ◄ ┘ └─┬─ KEYBOARD ───┤
                    └─ MOD ─────────┘                      └─ KEY ────────┘

 ►─┬─ ALARM ──────┬─┬─ STARTPRT ──┬─┬─ NLCR ◄ ──┬─┬─ PAGeable ──────┬─────►
   └─ NOALARM ◄ ──┘ └─ NOPRT ◄ ───┘ ├─ 40CR ────┤ └─ NONPAGeable ◄ ─┘
                                    ├─ 64CR ────┤
                                    └─ 80CR ────┘

 ►─┬─────────────────────────────────────────────────┬────────────────────►
   └─ DECimal point ─┬──────┬─┬─ Comma ─────┬────────┘
                     ├─ IS ─┤ └─ Period ◄ ──┘
                     └─ = ──┘

 ►─┬──────────────────────────────────────────────────────────────────────►
   └─ HELP ─┬─ NO ────────────────────────────────────────────────────────
            └─ LOAD MODule ─┬──────┬─ module-name ────────────────────────
                            ├─ IS ─┤
                            └─ = ──┘

─►────────────────────────────────────────────────────────────────────────►─
 .────────────────────────────────────────────────────────────────────────
  ─────────────────────────────────────────────────────────────────────────
   └─ SOUrce ─┬─ NONE ────────────────────────────────────────────────────
              └─ MODule module-name ─┬────────────────────────────────────
                                     └─ version ─┬──────┬─ version ─┘
                                                 ├─ IS ─┤
                                                 └─ = ──┘
```

```
─►─────────────────────────┬──────────────────────────────────────────────────►
 ──────────────────────────┘
 ──────────────────────┬─┘
 ────────────────────┬─┘
 ─┬─ HALF screen ◄ ─┬┘
  └─ FULL screen ───┘

►─────────────────────────────────────────────────────────────────────────────►─
 ─────────────────────────────────────────────────────────────────────────────
 ─────────────────────────────────────────────────────────────────────────────
  └─ HELPKEY ──┬──────┬── PFnn ──┘
               ├─ IS ─┤
               └─ = ──┘

►─┬───────────────────────────────────────────────────────────────────────────►
  └─ ON edit ERROR ──┬──────────────────────────────────────────────────────┬
                     └─ INCORRECT fields ATTRibutes = ( attributes-list ) ─┘

►─────────────────────────────────────────────────────────────────────────────►
 ────────────────────────────────────────────────────────────────────────────
  └─ CORRECT fields ATTRibutes = ( attributes-list ) ─┘

►───────────────────────────┬─────────────────────────────────────────────────►
 ───────────────────────────┘
  └─ SOUND ─┬─ ALARM ─────┬┘
            └─ NOALARM ◄ ─┘

►─┬──────────────────────────────────────────────────────────────────┬─ . ─►◄
  └─ ORIGIN for ─┬─ ALL ──────────────────────┬─┬──────┬─( row  column )─┘
                 │      ┌─────────────┐       │ ├─ IS ─┤
                 └─ ( ─▼─ device-code ─┴─ ) ──┘ └─ = ──┘
```

Parameters

ADD/MODIFY/DELETE

Specifies the action taken with regard to the MAP statement. ADD, MODIFY, and DELETE access for a map is subject to security restrictions specified for the batch compiler and individual maps, as outlined in "Compiler Security".

MAP map-name

Specifies the unique 1- through 8-character name for the map being defined, modified, or deleted. The following considerations apply to the composition of *map-name*:

- *Map-name* can consist of any alphanumeric or special characters.
- *Map-name* must begin with an alphanumeric or national character; for example, pound sign (#), at sign (@), or dollar sign ($).
- *Map-name* must not contain embedded period or blank characters.

VERSION IS version-n

Optionally specifies a version number to further identify the map. *Version-n* must be in the range 1 through 9999. If omitted, *version-n* defaults to the data dictionary version default, as defined by the Data Dictionary Definition Language (DDDL) SET OPTIONS statement.

DATETIME IS date-time-stamp

The map compiler DATETIME clause is returned in map source statements when you use the map utility to decompile a map.

If you use the DATETIME option to decompile a map from one DC system and add it to another system:

- *Do not change decompiled map source statements.* If you change statements, unpredictable errors will occur at runtime when you access the map.
- *Define identical record element descriptions on each system.* You can accomplish this by using IDD.

MSG PREFIX IS message-prefix

Defines the two-character prefix to be used as the default prefix for any MFLD in the map that is defined using the ERROR MESSAGE clause.

PANEL panel-name

Specifies the name of the panel with which the map is associated. The panel occurrence must already be defined in the data dictionary.

VERsion is version

Optionally specifies a version number to further identify the panel occurrence. If omitted, *version* defaults to the data dictionary version default as defined by the Data Dictionary Definition Language (DDDL) SET OPTIONS statement.

SYSTEM IS dc-version-n

Specifies the version number of a CA IDMS system with which the map is associated. *Dc-version-n* is the 1- through 4-character identifier assigned to the system at system generation.

RESIDENT/NONRESIDENT

Indicates whether the map load module is resident in storage at system runtime:

- **RESIDENT** specifies that the map load module is resident. This is useful for frequently used maps.
- **NONRESIDENT** (default) specifies that the map load module is not resident; the load module is loaded dynamically when required for a program mapping request.

USING RECORDS

(record-name/(record-name version-n) ROLENAME role-name)

Specifies the list of predefined schema and/or work records used by the map and optionally specifies role names for records:

- **Record-name** identifies the name of a record that contains elements referenced by the map. If *record-name* is not unique in the data dictionary, the version number of the necessary schema or work record must be supplied; the default value for *version-n* is specified at system generation.

  If a logical record is being used, the developer names the records containing elements that are part of the logical record and that are used in the map definition. The logical record name is later specified by the dialog or program using the map.

- **ROLENAME role-name** specifies the role name used for the record at runtime. Role names are needed when a given record type is referenced in more than one context. For example, the developer might specify the EMPLOYEE record layout twice for a map that uses the EMPLOYEE record for both employee-related and manager-related fields on a single map:

  - One specification of the EMPLOYEE record would not include a role name for the record.
  - The second specification of the EMPLOYEE record would include a valid role name for the record (for example, MANAGER). The role name must be used in subsequent references to the record in the map-definition.

  The specified role name can be established in two ways:

  - The role name can be previously defined for the record by a logical record definition in the subschema used by the program or dialog.
  - The role name can be unique to the map, defined at map definition time on the Associated Records screen or via the batch compiler.

EDIT/NOEDIT

Indicates whether automatic editing and error-handling are enabled for the map, as follows:

- **EDIT** (default) globally enables automatic editing and error-handling for the map.
- **NOEDIT** globally disables automatic editing and error-handling for the map; editing and error-handling criteria (if any) defined for map fields are ignored.

Note: For more information about enabling and disabling automatic editing, see "Enabling Automatic Editing and Error Handling".

RESET/NORESET MODIFIED

Indicates whether the modified data tags (MDTs) for data fields are reset automatically on a mapout operation:

- **RESET** (default) specifies that all MDTs are reset (turned off) when the map is mapped out.
- **NORESET** specifies that MDTs are left unchanged when the map is mapped out.

The MDT/NOMDT specification in the MFLD ATTRIBUTES clause for a field overrides the RESET/NORESET specification for that field if the map-level and field-level specifications differ. If MDT is chosen for a field, the MDT is set on regardless of the RESET MDT specification.

Note: For more information about the MDT/NOMDT setting, see "Attributes for Fields".

LOCK/UNLOCK KEYBOARD

Specifies whether the keyboard unlocks automatically after a mapout operation:

- **LOCK** specifies that the keyboard remains locked until the operator presses the RESET key.
- **UNLOCK** (default) specifies that the keyboard is unlocked after a mapout.

ALARM/NOALARM

Indicates whether a terminal alarm sounds automatically on a mapout operation:

- **ALARM** specifies that the terminal alarm sounds on a mapout operation. This specification is meaningful only if the terminal is equipped with a hardware alarm.
- **NOALARM** (default) specifies that the terminal alarm does not sound on mapout.

STARTPRT/NOPRT

Specifies whether the contents of the printer terminal buffer should be printed automatically upon completion of data transmission on a mapout operation:

- **STARTPRT** specifies that the contents of the printer terminal buffer are printed. This specification is meaningful only for mapping operations associated with 3280-type printers.
- **NOPRT** (default) specifies that the contents of the printer terminal buffer are not printed.

NLCR/40CR/64CR/80CR

Specifies character-per-line formatting for printer output:

- **NLCR** (default) specifies that no line formatting is performed on the printed output. Printing skips to a new line only when new line (NL) and carriage return (CR) characters are encountered.
- **40CR** specifies that the buffer contents are printed at 40 characters per line.
- **64CR** specifies that the buffer contents are printed at 64 characters per line.
- **80CR** specifies that the buffer contents are printed at 80 characters per line.

These specifications are applicable only if the STARTPRT clause is specified for the map.

PAGEABLE/NONPAGEABLE

Specifies whether the map is pageable:

- **PAGEABLE** specifies that the map is pageable. A pageable map is a map that can display more than one page of information at runtime.
- **NONPAGEABLE** (default) specifies that the map is not a pageable map.

Note: For more information about pageable maps, see "Pageable Maps".

DECIMAL POINT IS COMMA/PERIOD

Specifies the decimal point character for numeric fields on the map:

- **COMMA** specifies that the comma (,) is used as the decimal point, in accordance with international format. An external picture for the field also must be specified in international format, with the comma as the decimal point.
- **PERIOD** (default) specifies that the period (.) is used as the decimal point.

HELP

Specifies whether help will be implemented for the map.

NO/LOAD MODule module name

If there is Help for the map, the name of the load module that contains all the help source for the map.

HELPKEY IS PFnn

The PFKey designated as the Help key for the map.

SOUrce NONE/MODule module-name

The name of the IDD module that contains the help text for the map.

If module name is specified, you can optionally specify:

- The version number
- Whether the help is displayed on a full or half screen

ON EDIT ERROR

Defines incorrect-field attributes, correct-field attributes, and alarm status for use when a dialog or map redisplays a map that contains input errors. The following clauses assign error-handling criteria:

- **INCORRECT FIELDS ATTRIBUTES=(attributes-list)** specifies attributes that are assigned to incorrect fields when an edit error occurs. Typically, incorrect fields are given an attribute such as BRIGHT or BLINK to draw the operator's attention to the erroneous data. No default attributes are defined.

  Note: For more information about syntax for the **attributes-list**, see Attributes List.

- **CORRECT FIELDS ATTRIBUTES=(attributes-list)** specifies attributes that are assigned to correct and unedited fields when an edit error occurs. No default attributes are defined.

  Syntax for the **attributes-list** is discussed in Attributes List.

- **SOUND ALARM/NOALARM** specifies whether a terminal alarm sounds on input error:

  - **ALARM** indicates that the alarm is sounded. This option is meaningful only when a terminal is equipped with a hardware alarm.
  - **NOALARM** (default) indicates that the alarm is not sounded.

For example, a dialog or program can include code to redisplay a map when an error is detected in a field on mapin. When the display is mapped back out, incorrect-field attributes take effect for fields that are in error, and correct-field attributes take effect for fields that are not in error. The terminal operator can correct the errors and resubmit the map.

Notes:

- For more information about the use of error-handling specifications, see "Error-handling Criteria."
- For more information about how dialogs and programs override specifications made in the ON EDIT ERROR clause, see "Map Inquiry and Modification."

ORIGIN FOR (device-code)/ALL IS (row column)

Positions the origin of the runtime map at a row/column location on specified devices:

- **Device-code** names one device. Available *device-code* specifications are 12X40, 12X80, 24X80, 32X80, 43X80, and 27X132. The specified device must be defined in the DEVICES clause of the MAP statement. More than one ORIGIN FOR *device-code* clause can be included in a single MAP statement.

  Parentheses are required when a device code(s) is specified.

- **ALL** names all devices defined in the DEVICES clause of the MAP statement.

- **Row column** specifies the coordinates at which the upper left-hand corner of the runtime map is plotted for all devices specified in the ORIGIN FOR specification. Only one *row column* specification can be made for a given ORIGIN FOR clause; if specified, it must be enclosed in parentheses. If not specified, *column* defaults to 1.

  Parentheses are required around the row column coordinates.

Chapter 13: Batch Compiler Execution and JCL

This chapter discusses about batch compiler execution and JCL.

This section contains the following topics:

Overview

The developer submits batch compiler statements in JCL to create and maintain map and panel entity occurrences in the data dictionary according to instructions provided by mapping statements. Reports provided by the batch compiler inform the map developer of the outcome of the compile, and provide diagnostic and error messages when necessary.

This section describes special coding features provided by the batch compiler, presents the JCL required to run the programs that add, modify, or delete a map or panel occurrence and provides information about compiler reports and messages.

Special Coding Features of the Batch Compiler

Clauses provided by the batch compiler can be used to define special versions of maps for different devices and to position and center batch-defined maps on different devices.

Each of these special coding features of the batch compiler is presented as follows.

Defining Versions of Maps for Different Devices

Supported Screen Sizes

The following terminal screen (that is, **device**) sizes are supported by the batch compiler:

- 12X40
- 12X80
- 24X80
- 32X80
- 43X80
- 27x132

Defining Device-independent Maps

A map load module that can be used with more than one device type is said to be device-independent. A map developer uses the batch compiler to define a device independent map by specifying different screen layouts for each device on which the map can be displayed. For example, it might be necessary to define shorter literal fields for a 12X40 device than for a 43X80 device.

The DEVICES clause of the MAP AUTOPANEL or PANEL statement is used to specify the list of devices that is valid for the map being defined. The list of devices can be subdivided into device groupings; a different screen layout can be specified for each device grouping.

Defining Device Groupings

Device groupings are established at the field level by using MFLD (for MAP AUTOPANEL) or PFLD statements. Use any of the following techniques to establish device groupings:

- **Include multiple FOR specifications within one MFLD or PFLD statement**, as shown:

  ```
  ADD PANEL JOB-DATA
      DEVICES=(12X40,24X80,32X80).

      ADD PFLD NUM-POSITIONS.
          ATTRIBUTES=(BRIGHT,BLUE)
               FOR (12X40)
                   AT (2,40)
                   VALUE IS '# POSITIONS'
               FOR (24X80)
                   AT (3,80)
                   VALUE IS 'NUMBER OF POSITIONS'
               FOR (32X80)
                   AT (3,80)
                   VALUE IS 'TOTAL NUMBER OF POSITIONS'.
  ```

  The FOR clauses in the previous example establish the following device groupings:

  - **The first FOR clause** establishes a device grouping valid for 12X40 devices. At runtime, the literal field displays as follows:
    - The attribute byte for the field is at coordinate 2,40; the literal begins at the next coordinate (3,1)
    - The literal is # POSITIONS
  - **The second FOR clause** establishes a device grouping valid for 24X80 devices. At runtime, the literal field displays as follows:
    - The attribute byte for the field is at coordinate 3,80; the literal begins at the next coordinate (4,1)
    - The literal is NUMBER OF POSITIONS
  - **The third FOR clause** establishes a device grouping valid for 32X80 devices. At runtime, the literal field displays as follows:
    - The attribute byte for the field is at coordinate 3,80; the literal begins at the next coordinate (4,1)
    - The literal is TOTAL NUMBER OF POSITIONS

Clauses before the first FOR clause in an MFLD or PFLD statement apply to each subsequently established device grouping. In the previous example, the ATTRIBUTES clause displays each literal field established in the PFLD statement in bright blue.

- **Specify multiple screen sizes in a single FOR clause of an MFLD or PFLD statement**, as shown:

```
ADD MAP EMP-INFO
    AUTOPANEL DEVICES=(24X80,32X80,43X80).

    ADD MFLD
            FOR (24X80,32X80)
            AT (14,15)
            VALUE IS 'DEPT NAME'.
```

  The FOR clause in the previous sample MFLD statement establishes a device grouping made up of two device types (24X80 and 32X80). A third device grouping (43X80) is implicitly created; however, no value is assigned to the latter device grouping by this MFLD statement.

- **Specify different screen sizes in individual MFLD or PFLD statements**, as shown:

```
ADD MAP CEXME413
    AUTOPANEL DEVICES=(24X80,32X80,43X80)
              USING (EMPOSITION 100).

    ADD MFLD
              FOR (24X80)
              AT (14,15)
              DFLD SALARY-AMOUNT-0420.
    ADD MFLD
              FOR (32X80)
              AT (20,15)
              DFLD SALARY-AMOUNT-0420.
```

  Each MFLD (for MAP AUTOPANEL) statement in the previous example associates a screen field with the SALARY-AMOUNT-0420 record element by establishing a device grouping:

  - **The first MFLD statement** establishes a device grouping valid for 24X80 devices. At runtime, the attribute byte for the field is at coordinate 14,15; the value begins at the next coordinate (14,16).
  - **The second MFLD statement** establishes a device grouping valid for 32X80 devices. At runtime, the attribute byte for the field is at coordinate 20,15; the value begins at the next coordinate (20,16).

  A third device grouping (43X80) is implicitly created in the previous example; however, values for SALARY-AMOUNT-0420 are not displayed on 43X80 devices.

- **Specify a combination of the previous options**, as shown:

```
ADD MAP CEXME413 VERSION 2
    AUTOPANEL DEVICES=(24X80,32X80,43X80)
              USING (EMPOSITION 100).

     ADD MFLD
              FOR (24X80,32X80,43X80)
              AT (10,15)
              DFLD SALARY-GRADE-0420.
     ADD MFLD
              FOR (24X80)
              AT (14,15)
              DFLD SALARY-AMOUNT-0420.
     ADD MFLD
              FOR (32X80)
              AT (20,15)
              DFLD SALARY-AMOUNT-0420.
```

Reconciling Conflicting Specifications

The batch compiler reconciles any conflicting device grouping specifications made by the map developer so that any given screen size belongs to only one device grouping. Since 24X80 and 32X80 are each specified in two different device grouping specifications in the previous sample statements, the batch compiler subdivides the first device grouping and assigns the specifications made for the grouping to each of the three device groupings (24X80, 32X80, and 43X80) established by the second and third MFLD statements.

For example, the following device groupings are established by the previous sample statements:

- A device grouping for 24X80 devices displays the following values at runtime:
  - SALARY-GRADE-0420 values are displayed at 10,15.
  - SALARY-AMOUNT-0420 values are displayed at 14,15.

- A device grouping for 32X80 devices displays the following values at runtime:
  - SALARY-GRADE-0420 values are displayed at 10,15.
  - SALARY-AMOUNT-0420 values are displayed at 20,15.
- A device grouping for 43X80 devices displays runtime values for the SALARY-GRADE-0420 element at 10,15.

The batch compiler subdivides conflicting device grouping specifications only as much as necessary to insure that each screen size belongs to a maximum of one device grouping. For example, two device groupings are established when the following sample statements are compiled:

```
ADD MAP CEXMJKD2
       AUTOPANEL DEVICES = (24X80, 32X80, 43X80)
       USING (INSURANCE-PLAN 100).
   ADD MFLD
         FOR (24X80,32X80)
         AT (14,15)
         DFLD COMPANY-NAME-0435.
   ADD MFLD
         FOR (24X80,32X80,43X80)
         AT (15,15)
         DFLD GROUP-NUMBER-0435.
```

The three screen sizes specified in the DEVICES clause of the previous MAP AUTOPANEL statement are divided into two device groupings:

- The device grouping for 24X80 and 32X80 devices displays values for the COMPANY-NAME-0435 and GROUP-NUMBER-0435 elements.
- The device grouping for 43X80 devices displays values only for the GROUP-NUMBER-0435 element.

An AT specification for a device grouping must be valid for the smallest device (screen size) in the device grouping. For example, AT (16,32) cannot be specified for a device grouping that contains the 12X40 device type.

The ORIGIN FOR clause also creates device groupings, as explained in Positioning Maps on Different Devices (see page 261).

Effects of Device Groupings

The use of device groupings to achieve device independence can affect batch compiler performance, batch utility panel and map reports, and map load module overhead:

- The **batch compiler** validates row and column specifications based on the smallest screen size within each device grouping. A row/column specification of (20,25), for example, would be acceptable for the (24X80,32X80,43X80) device grouping but would not be acceptable for the (12X40,24X80,32X80) device grouping since the specified screen location cannot be accommodated on the 12X40 screen.
- The **batch utility** produces panel and map reports that list as many screen formats as there are device groupings associated with the specified panel/map. Each format illustrates only the smallest screen size within a device grouping. For example, if the device groupings (12X40,43X80), (24X80,32X80), and (12X80) have been defined for panel/map fields associated with the panel, the map utility report displays three screen formats, with screen sizes of 12X40, 24X80, and 12X80.
- A **map load module** contains a separate panel occurrence for each device group, even if there is only one panel field specification that is different.

Maps defined with device groupings cannot be displayed or edited by using the online mapping compiler since that compiler can only display one screen layout for a given map-definition.

Positioning Maps on Different Devices

The origin for a map is its upper left-hand coordinate.

The ORIGIN FOR clause of the MAP or MAP AUTOPANEL statement centers or repositions an entire runtime map on a terminal screen by performing the following functions:

- Specifies one or more devices (by screen size) and creates a device grouping
- Specifies the coordinate at which the origin of the panel/map is placed on the designated devices at runtime

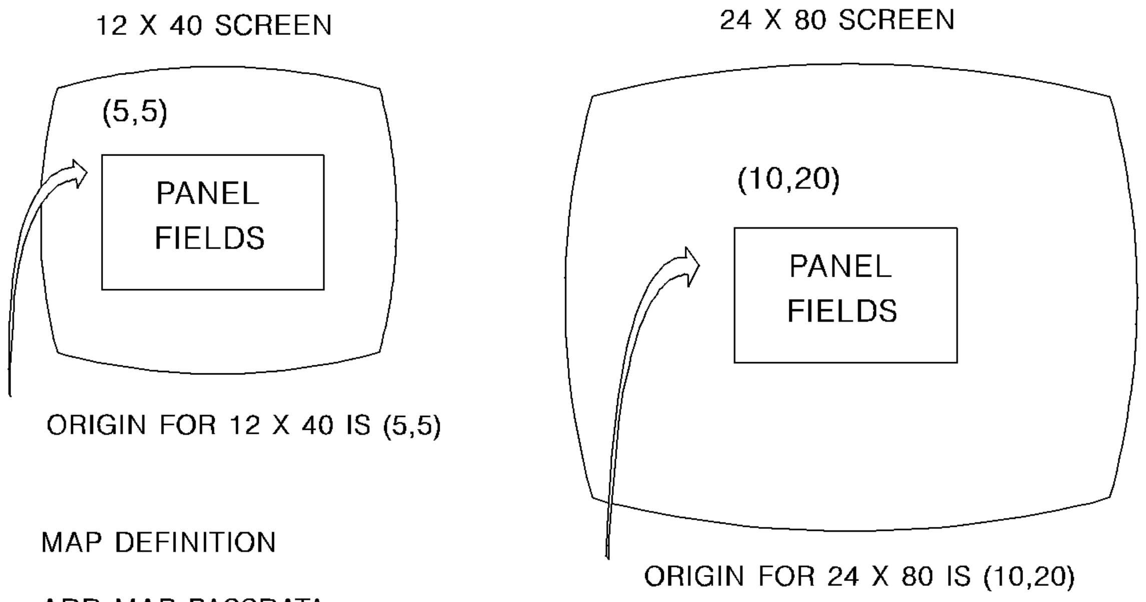
For example, the origin for the map defined by the following sample MAP statement would be located at coordinate 10,20 when displayed on 24X80 devices and at coordinate 30,20 on 43X80 devices at runtime:

```
ADD MAP EMPDATA
    AUTOPANEL DEVICES=(24X80,43X80)
    ORIGIN FOR (24X80) IS (10,20)
    ORIGIN FOR (43X80) IS (30,20).
```

Map Positioning for Different-size Devices

The effects of an ORIGIN FOR specification on devices of two different sizes are illustrated in the following figure:

The origin of the panel/map is positioned at row 5, column 5 for the smaller device. The origin is positioned at row 10, column 20 for the larger device.



```
MAP DEFINITION

ADD MAP PASSDATA
      PANEL IS RESERVATION DATA
      ORIGIN FOR 12 X 40 IS 5,5
      ORIGIN FOR 24 X 80 IS 10,20.
```

Batch Compiler JCL

The map compiler accepts compiler statements that were either written by the map developer or generated by the DECOMPILE or TERSE process of the map utility. The resulting entity occurrences are placed in the DDLDML area of the data dictionary and can be used:

- *By the batch utility*, to generate map load modules for application programs
- *By DML processors*, to interpret and expand mapping requests coded in CA IDMS application programs
- *By the CA ADS compiler,* to interpret and expand mapping requests in CA ADS dialogs

The batch compiler executes in update mode. Run the batch compiler through the central version or as a local CA IDMS/DB program with active journal files to protect the integrity of the data dictionary.

z/OS JCL

```
//RHDCMAP1 EXEC PGM=RHDCMAP1,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=idms.sysmsg.ddldcmsg,DISP=SHR
//SYSLST   DD SYSOUT=A
//SYSPCH   DD DUMMY
//SYSIDMS  DD *
DMCL=dmcl-name
DICTNAME=appldict
sysidms parameters
//SYSIPT   DD *
source statements
```

idms.dba.loadlib	Data set name of CA IDMS load library that contains the DMCL and database table load modules
idms.custom.loadlib	Data set name of the CA IDMS load library containing customized CA IDMS executable modules
idms.cagjload	Data set name of CA IDMS load library that contains the CA IDMS executable modules that do not require customization
idms.sysctl	Data set name of SYSCTL file
idms.sysmsg.ddldcmsg	Data set name of the system message area

dmcl-name	The name of the dictionary the DMLF precompiler should access
appldict	The name of the application dictionary that should be accessed
sysidms parameters	A list of SYSIDMS parameters for this job.

Note: For more information about the SYSIDMS parameters, see the CA IDMS Common Facilities Guide.

Local Mode

To execute in local mode, perform these steps:

- Remove the sysctl DD statement.
- Add the following statements:

```
//dictdb  DD DSN=idms.appldict.ddldml,DISP=SHR
//sysjrnl DD DSN=idms.tapejrnl,DISP=(NEW,CATLG),UNIT=tape
```

dictdb	DDname of application dictionary definition area
idms.appldict.ddldml	Data set name of application data dictionary DDLDML area
sysjrnl	DDname of first tape journal file
idms.tapejrnl	Data set name of first tape journal file

z/VSE JCL

```
// UPSI   b
// DLBL   SYSIDMS,'#SYSIPT'
// EXTENT sys020,nnnnnn,,,ssss,llll
// ASSGN  sys020,DISK,VOL=nnnnnn,SHR
// EXEC  RHDCMAP1
sysidms parameters
/*
```

b	Appropriate UPSI switch, 1-8 characters, if specified in the IDMSOPTI module
#SYSIPT	If #SYSIPT is used, the individual parameters must be listed in the SYSIDMS parameters statement. This can also be defined as a disk dataset, in which case #SYSIPT is replaced by the name of the file containing the parameters and the parameters are not listed.

sys020	Logical unit assignment of output file
nnnnnn	Volume serial number of disk storage device
ssss	Relative starting track
llll	Number of tracks
sysidms parameters	A list of SYSIDMS parameters for this job.

Note: For more information about the SYSIDMS parameters, see the *CA IDMS Common Facilities Guide*.

Local Mode

To execute in local mode, perform these steps:

- Remove the UPSI specification.
- Add the following statements:

```
// DLBL   dictdb,'idms.dictdb',,DA
// EXTENT sys005,nnnnnn
// ASSGN  sys005,DISK,VOL=nnnnnn,SHR
// TLBL   sysjrnl,'idms.tapejrnl',,nnnnnn,,f
// ASSGN  sys009,TAPE,VOL=nnnnnn
```

dictdb	Filename of application data dictionary
idms.dictdb	File-id of application data dictionary
sys005	Logical unit assignment of data dictionary
sysjrnl	Filename of tape journal file
idms.tapejrnl	File-id of tape journal file
f	File number of tape journal file
sys009	Logical unit assignment of tape journal file

z/VM JCL

```
FILEDEF SYSPCH DUMMY
FILEDEF SYSLST PRINTER
FILEDEF CDMSLIB DISK IDMSLIB LOADLIB A6
FILEDEF SYSIDMS DISK sysidms input a
FILEDEF SYSIPT DISK comp input a
GLOBAL LOADLIB IDMSLIB
OSRUN RHDCMAP1
```

sysidms input a	Filename, filetype, and filemode of the file containing the SYSIDMS input parameters.
comp input a	Filename, filetype, and filemode of the file that contains batch compiler source statements.

Note: For more information about the SYSIDMS parameters, see the *CA IDMS Common Facilities Guide*.

Local Mode

To execute in local mode, perform these steps:

1. Specify that RHDCMAP1 is executing in local mode by performing one of the following:
   - Link RHDCMAP1 with an IDMSOPTI program that specifies local execution mode.
   - Modify the OSRUN statement:

     ```
     OSRUN RHDCMAP1,PARM='*LOCAL*'
     ```

     Note: This option is valid only if the OSRUN command is issued from a System Product interpreter or an EXEC2 file.
   - Specify *LOCAL* as the first input parameter of the file identified by comp input a.
2. Add the following statements before the OSRUN statement:

   ```
   FILEDEF dictdb DISK dictdb dictfile d
                  (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
   FILEDEF j1jrnl DISK j1jrnl jrnlfile k
                  (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
   ```

dictdb	DDname of the application data dictionary file
dictdb dictfile d	Filename, filetype, and filemode of the application data dictionary file
pppp	Page size of the file

nnnn	Number of pages in the file
j1jrnl	DDname of the first disk journal file
j1jrnl jrnlfile k	Filename, filetype, and filemode of the first disk journal file

Compiler Reports and Messages

Compiler syntax and input validation messages are returned to the user on the compiler input statement report. Diagnostic and error messages are provided on the input statement report when source statements input to the map compiler contain errors. Each type of message is discussed as follows.

Diagnostic Messages

The following diagnostic information is provided on an input statement report when errors are encountered:

- **The word ERROR or WARNING** following the statement number of the erroneous source line
- **A dollar sign** ($), positioned:
  - Under the first character of the source line that could not be successfully processed
  - Under the period that terminates a map source statement to indicate that a logic error has occurred
- **The action taken by the compiler** for each mapping language statement, as follows:
  - **ADDED—**The statement is accepted and the definition is added in the data dictionary.
  - **MODIFIED—**The statement is accepted and the definition is modified in the data dictionary.
  - **DELETED—**The statement is accepted and the definition is deleted from the data dictionary.
  - **NO ACTION—**The statement is rejected. The NO ACTION message implies the following:
    - At least one E-level error has been detected.
    - No occurrence is added, modified, or deleted in the data dictionary for the statement.

Error Messages

Error messages appear on a separate page at the end of the input statement listing. The batch compiler lists error messages in the following format:

```
STMT      SVRTY         ERROR       FOUND       MESSAGE
nnnn   severity-level   nnnnnn    found-text  message-text
```

The error messages provide the following information:

- **STMT** *(nnnn)* identifies the compiler-generated statement number for each line that contains an error.
- **SVRTY** *(severity-level)* identifies the severity level of the input error. The following severity levels can be returned:
  - **W** (warning)**—**Flags potential problems in the source code; the statement containing the error is processed. W-level messages provide information about potential problems and do not necessarily indicate errors.
  - **E** (error)**—**Identifies erroneous input; the statement containing the error has been rejected. The error must be corrected before the code can be compiled.
  - **F** (fatal)**—**Identifies errors that affect more than one map source statement. The error must be corrected before the code can be compiled.
- **ERROR** *(nnnnnn)* provides the 6-digit identifying number for the error message provided in the MESSAGE column described as follows.
- **FOUND** *(found-text)* identifies the erroneous portion of the given input line. Either of the following items are listed in this column:
  - The first eight characters of the parameter in error
  - A period (.) character to indicate that an end-of-statement error has occurred
- **MESSAGE (message-text)** specifies the nature of the problem encountered by the map compiler. For a detailed description of error codes and messages, see the *CA IDMS Messages and Codes Guide*.

The error message page of the compiler input-statement listing also specifies the number of coding errors encountered. The listed number does not necessarily represent the actual number of errors in the source code. For example, rejection of a PANEL, PFLD, MAP, or MFLD statement might cause the map compiler to reject subsequent correct statements simply because the proper currency has not been established. The following section was left out because the previous sample compiled without reports It should be inserted again.

Sample Report

A sample input-statement listing that contains diagnostic and error messages is provided in the following figure. The source for this example contains only three actual errors:

- The ATTRIBUTES specifications in line 10 are not enclosed in parentheses.
- The VALUE specification in line 17 is not enclosed in quotation marks.
- The MFLD statement in line 29 is missing the required AT specification.

The batch compiler returns 16 errors as a result of the actual errors in the sample compile.

```
CAGJF0 CA IDMS/DC MAPPING COMPILER PHASE 1        CA IDMS/DC IS A PROPRIETARY SOFTWARE PRODUCT              DATE     TIME   PAGE
               VERSION nn.n                   LICENSED FROM CA                                mm/dd/yy  113808     1
   STMT   ITEM   NUMBER                             -- MAPPING INPUT STATEMENT LISTING --
      1   MAP        1          ADD MAP CEXME028
      2                         AUTOPANEL
      3                         USING ((EMPLOYEE 100))
      4                         EDIT
      5                         RESET UNLOCK NOALARM NOPRT NLCR.                                VERSION     1    *** ADDED ***
      6
      7   PFLD       1          ADD MFLD
      8                             FOR (24X80, 32X80, 43X80, 27X132)
      9                             AT (4,25)
     10                             ATTR = BRIGHT PROTECTED
   *** ERROR ***                           $
   *** ERROR ***                           $
   *** UNKNOWN KEYWORD ***                 $
   *** UNKNOWN KEYWORD ***                        $
     11                             VALUE IS 'EMPLOYEE PHONE INFORMATION'
   *** UNKNOWN KEYWORD ***          $
   *** UNKNOWN KEYWORD ***                $
   *** UNKNOWN KEYWORD ***                   $
   *** UNKNOWN KEYWORD ***                             $
   *** UNKNOWN KEYWORD ***                                   $
   *** UNKNOWN KEYWORD ***                                              $
     12                             LITERAL.                                                     *** NO ACTION ***
     13
     14   PFLD       2          ADD MFLD
     15                             FOR (24X80, 32X80, 43X80, 27X132)
     16                             AT (8,25)
     17                             VALUE IS EMPLOYEE ID
   *** ERROR ***                             $
   *** ERROR ***                             $
   *** UNKNOWN KEYWORD ***                   $
   *** UNKNOWN KEYWORD ***                            $
     18                             LITERAL.                                                     *** NO ACTION ***
     19
     20   PFLD       3          ADD MFLD
     21                             AT (8,45)
     22   MFLD       3              DFLD EMP-ID-0415
     23                             OF EMPLOYEE VER 100
     24                             EXT PIC IS INT.                                              *** ADDED ***
     25
     26   PFLD       4          ADD MFLD
     27                             VALUE IS 'PHONE NUMBER'
     28                             LITERAL.
   *** ERROR ***                    $
   *** ERROR ***                    $
          MFLD       4                                                                           *** NO ACTION ***
     29
     30   PFLD       5          ADD MFLD
     31                             AT (12,45)
     32   MFLD       5              DFLD EMP-PHONE-0415
     33                             OF EMPLOYEE VER 100
     34                             EXT PIC IS INT.                                              *** ADDED ***
```

```
CAGJF0 CA IDMS/DC MAPPING COMPILER PHASE 1        CA IDMS/DC IS A PROPRIETARY SOFTWARE PRODUCT             DATE     TIME   PAGE
               VERSION nn.n                   LICENSED FROM CA                                mm/dd/yy  113808     2
STMT  SVRTY  ERROR   FOUND    MESSAGE
   10   E   386005  BRIGHT    ATTRIBUTE CLAUSE REQUIRES A PARENTHESIZED LIST IN PANEL FIELD STATEMENT
   10   E   388001  BRIGHT    INVALID PANEL FIELD NAME SPECIFIED IN MAP FIELD STATEMENT
   10   E   388017  BRIGHT    INVALID KEYWORD IN MAP FIELD STATEMENT
   10   E   388017  PROTECTE  INVALID KEYWORD IN MAP FIELD STATEMENT
   11   E   388017  VALUE     INVALID KEYWORD IN MAP FIELD STATEMENT
   11   E   388017  IS        INVALID KEYWORD IN MAP FIELD STATEMENT
   11   E   388017  'EMPLOYE  INVALID KEYWORD IN MAP FIELD STATEMENT
   11   E   388017  PHONE     INVALID KEYWORD IN MAP FIELD STATEMENT
   11   E   388017  INFORMAT  INVALID KEYWORD IN MAP FIELD STATEMENT
   11   E   388017  '         INVALID KEYWORD IN MAP FIELD STATEMENT
   17   E   386008  EMPLOYEE  VALUE CLAUSE FOR PANEL FIELD MUST BE FOLLOWED BY A STRING OR PARENTHESIZED LIST
   17   E   388001  EMPLOYEE  INVALID PANEL FIELD NAME SPECIFIED IN MAP FIELD STATEMENT
   17   E   388017  EMPLOYEE  INVALID KEYWORD IN MAP FIELD STATEMENT
   17   E   388017  ID        INVALID KEYWORD IN MAP FIELD STATEMENT
   28   E   386027  LITERAL   AT LEAST ONE 'AT' CLAUSE IS REQUIRED FOR ADD OF PANEL FIELD
   28   E   388001  LITERAL   INVALID PANEL FIELD NAME SPECIFIED IN MAP FIELD STATEMENT
    16  ERRORS
  PROCESS=LOAD
  MAP=CEXME028,VERSION=00001
```

Note: CAGJF0 in the upper left-hand corner of the Error Message page is the release number.

Chapter 14: Batch Utility Reference

This chapter contains the following topics:

This section contains the following topics:

Overview

The batch utility uses information defined in the data dictionary to perform the following activities:

- Generate map load modules
- Produce map and panel reports
- Produce a facsimile of a map and panel on a terminal screen
- Decompile map and panel occurrences generated by either the batch or the online compiler into map source code that can be used as input to the batch compiler
- Delete map load modules

The batch utility also is used to migrate a map from one dictionary to another. Decompiled batch source code from one dictionary can be compiled on another dictionary, and a new load module can be generated from the newly compiled map code.

The statements and JCL necessary to use the batch mapping utility are presented as follows.

Batch Utility Statements

The following statements control batch utility operations:

- The *PROCESS statement* specifies the operations to be performed.
- The *PANEL statement* specifies the panel occurrences to be processed.
- The *MAP statement* specifies the map occurrences to be processed.

Syntax for each of these utility statements is presented, followed by examples of their use.

Note: No SIGNON to dictionary card is used because no DDLDML area updates are possible from the batch utility.

Batch Utility Statements

The PROCESS statement specifies the actions to be taken by the batch utility. The following general rules apply when coding PROCESS statements:

- One or more PROCESS statements must be submitted in each batch utility run.
- When multiple PROCESS statements are specified, each successive PROCESS statement overrides the previous one.
- A PROCESS statement specifies the action to be taken for the utility PANEL and MAP statements that follow it.

```
                        ┌────────── , ───────────────────────────────────┐
►►─── PROCESS = ───▼─┬─── LOAD ───────────────────────────────────────┴─►◄
                     ├─┬─ REPORT ─┬───────────────────────────────────┤
                     │ └─ IMAGE ──┘                                   │
                     ├─┬─ DECOMPILE ─┬─┬──────────────────────────┬───┤
                     │ └─ TERSE ─────┘ └─ , DATETIME = ┬ YES ┬────┘   │
                     │                                 └ NO ─┘        │
                     ├────── ALL ─────────────────────────────────────┤
                     └────── DELETE ──────────────────────────────────┘
```

Parameters

LOAD

Generates a map load module and stores the module in the DDLDCLOD area of the data dictionary. LOAD applies only to MAP statements; at least one MAP statement must follow a PROCESS LOAD statement.

REPORT/IMAGE

Generates a report and/or a copy of the screen format for all specified map and panel occurrences:

- *REPORT* prints a screen image and report for the specified map and panel occurrences as shown in the following figure.
- *IMAGE* prints a screen image for the specified map and panel occurrences.

DECOMPILE/TERSE

Produces source code from data dictionary map and panel occurrences:

- *DECOMPILE* provides all specifications made for the named map and/or panel occurrences
- *TERSE* provides only non-default specifications for the named map and/or panel occurrences

Output for either DECOMPILE or TERSE is written to SYSPCH and consists of executable source code suitable for processing by the map compiler. The decompile process does not affect the load module.

DATETIME date-time-stamp

DATETIME has the following options:

- YES—Includes the map's date/time stamp in the decompiled map's source code. The date/time stamp is returned in the DATETIME clause in the newly decompiled map source statements.
- NO (default)—Decompiles the map without retaining the map's date/time stamp.

Note: DATETIME is only an option for decompile operations (PROCESS=DECOMPILE or PROCESS=TERSE).

DELETE

Logically deletes map load modules from the DDLDCLOD area. The actual deletion is performed at CA IDMS startup. DELETE applies only to MAP statements; at least one MAP statement must follow a PROCESS DELETE statement. The DELETE operation has no effect on map occurrences in the DDLDML area. The map compiler must be used to delete map or panel occurrences.

ALL

Requests that LOAD, REPORT, and DECOMPILE be performed. ALL applies only to MAP statements; at least one MAP statement must follow a PROCESS ALL statement.

Note: When multiple processes are specified, each must be separated by a comma, as shown:

PROCESS=REPORT,LOAD

If there are spaces between items, this will result in the rest of the line to be ignored.

Sample Report and Code

As a result of specifying DECOMPILE, REPORT, the map utility produces a report, a screen image, and mapping language source code as shown on the following pages.

```
EJECT
MAP CEXME028 VERSION     1
    AUTOPANEL
    DEVICES = (24X80, 32X80, 43X80, 27X132)
    NONRESIDENT
    USING ( (EMPLOYEE    100))
    EDIT
    RESET UNLOCK NOALARM NOPRT NLCR
    NONPAGEABLE
    DECIMAL POINT IS PERIOD
    HELP NO
    ON EDIT ERROR
        SOUND NOALARM.
         SPACE 2
MFLD
    FOR (24X80, 32X80, 43X80, 27X132)
    AT (    4,   25)
    ATTRIBUTES = (ALPHANUMERIC,PROTECTED,DETECTABLE,BRIGHT,NOMDT,
                  NOBLINK,NORMAL-VIDEO,NOUNDERSCORE,
                  NOCOLOR)
    NODELIMIT
    VALUE = ((    1) 'EMPLOYEE PHONE INFORMATION')
    NOCURSOR
    LITERAL.
         SPACE 2

MFLD
    FOR (24X80, 32X80, 43X80, 27X132)
    AT (    8,   25)
    ATTRIBUTES = (NUMERIC,PROTECTED,NONDETECTABLE,DISPLAY,NOMDT,
                  NOBLINK,NORMAL-VIDEO,NOUNDERSCORE,
                  NOCOLOR)
    NODELIMIT
    VALUE = ((    1) 'EMPLOYEE ID')
    NOCURSOR
    LITERAL.
         SPACE 2
```

```
MFLD
    FOR (24X80, 32X80, 43X80, 27X132)
    AT (   8,  45)
    ATTRIBUTES = (ALPHANUMERIC,UNPROTECTED,NONDETECTABLE,DISPLAY,NOMDT,
                 NOBLINK,NORMAL-VIDEO,NOUNDERSCORE,
                 NOCOLOR)
    DELIMIT = SKIP
    NOCURSOR

    DFLD EMP-ID-0415
      OF EMPLOYEE VER    100
        HELP SOURCE NONE
        OPTIONAL
        REVERSE NUMERIC IS NO
    EXTERNAL PICTURE IS INTERNAL
        ZEROED WHEN NULL
        DISPLAY WHEN ZERO
    FOR INPUT
        JUSTIFY LEFT
        PAD NO
        DATA YES
        UPPER NO
    FOR OUTPUT
        DATA YES
        BACKSCAN  NO.
         SPACE 2
MFLD
    FOR (24X80, 32X80, 43X80, 27X132)
    AT (  12,  25)
    ATTRIBUTES = (NUMERIC,PROTECTED,NONDETECTABLE,DISPLAY,NOMDT,
                 NOBLINK,NORMAL-VIDEO,NOUNDERSCORE,
                 NOCOLOR)
    NODELIMIT
    VALUE = ((   1) 'PHONE NUMBER')
    NOCURSOR
    LITERAL.
         SPACE 2
MFLD
    FOR (24X80, 32X80, 43X80, 27X132)
    AT (  12,  45)
    ATTRIBUTES = (ALPHANUMERIC,UNPROTECTED,NONDETECTABLE,DISPLAY,NOMDT,
                 NOBLINK,NORMAL-VIDEO,NOUNDERSCORE,
                 NOCOLOR)
    DELIMIT = SKIP
    NOCURSOR

    DFLD EMP-PHONE-0415
      OF EMPLOYEE VER    100
        HELP SOURCE NONE
        OPTIONAL
        REVERSE NUMERIC IS NO
    EXTERNAL PICTURE IS INTERNAL
        ZEROED WHEN NULL
        DISPLAY WHEN ZERO
    FOR INPUT
        JUSTIFY LEFT
        PAD NO
        DATA YES
        UPPER NO
    FOR OUTPUT
        DATA YES
        BACKSCAN  NO.
CAGJF0                           CA IDMS/DC  MAP UTILITY                         DATE: mm/dd/yy  TIME: 171634 PAGE     3

     THE FOLLOWING SYMBOLS REPRESENT ATTRIBUTE CHARACTERS:
          U - UNPROTECTED ALPHANUMERIC FIELD
          P - PROTECTED ALPHANUMERIC FIELD
          N - UNPROTECTED NUMERIC FIELD
          S - AUTOSKIP FIELD (PROTECTED AND NUMERIC)
```

```
CAGJF0                              CA IDMS/DC  MAP UTILITY                              DATE:  mm/dd/yy  TIME: 171634 PAGE     4
REPORT FOR MAP   CEXME028                         VERSION     1     COMPILE DATE: mm/dd/yy        COMPILE TIME: 171458
DEVICES: 24X80,  32X80,  43X80,  27X132
    5   10   15   20   25   30   35   40   45   50   55   60   65   70   75   80   85   90   95  100  105  110  115  120  125  130
**********************************************************************************************************************************



                        PEMPLOYEE PHONE INFORMATION



                        SEMPLOYEE ID        U....S
                                    (CURSOR) -



                        SPHONE NUMBER       U.........S










**********************************************************************************************************************************
    5   10   15   20   25   30   35   40   45   50   55   60   65   70   75   80   85   90   95  100  105  110  115  120  125  130
CAGJF0                              CA IDMS/DC  MAP UTILITY                             DATE: mm/dd/yy  TIME: 171634 PAGE     5
REPORT FOR MAP   CEXME028                         VERSION     1     COMPILE DATE: mm/dd/yy        COMPILE TIME: 171458
DEVICES: 24X80,  32X80,  43X80,  27X132
USING RECORDS:
      EMPLOYEE                         VERSION   100
WCC: NOALARM, UNLOCK KEYBOARD, RESET MODIFIED, NOPRT, NLCR
PANEL CEXME028-AUTOPANEL               VERSION     1
PFLD: AUTOPF00001                            AT ( 4,25)
         ATTRIBUTES = (ALPHANUMERIC,PROTECTED,DETECTABLE,BRIGHT,NOMDT,
                       NOBLINK,NORMAL-VIDEO,NOUNDERSCORE,
                       NOCOLOR)
         NODELIMIT
      LITERAL STRING
PFLD: AUTOPF00002                            AT ( 8,25)
         ATTRIBUTES = (NUMERIC,PROTECTED,NONDETECTABLE,DISPLAY,NOMDT,
                       NOBLINK,NORMAL-VIDEO,NOUNDERSCORE,
                       NOCOLOR)
         NODELIMIT
      LITERAL STRING
PFLD: AUTOPF00003                            AT ( 8,45)
         ATTRIBUTES = (ALPHANUMERIC,UNPROTECTED,NONDETECTABLE,DISPLAY,NOMDT,
                       NOBLINK,NORMAL-VIDEO,NOUNDERSCORE,
                       NOCOLOR)
         DELIMIT SKIP
DFLD: EMP-ID-0415                                    OF EMPLOYEE
         HELP SOURCE NONE
         OPTIONAL
         EXTERNAL PICTURE IS INTERNAL
             ZEROED WHEN NULL  DISPLAY WHEN ZERO
         INPUT:  JUSTIFY LEFT, UPPER NO, DATA YES, PAD NO
         OUTPUT: BACKSCAN NO, DATA YES
PFLD: AUTOPF00004                            AT (12,25)
         ATTRIBUTES = (NUMERIC,PROTECTED,NONDETECTABLE,DISPLAY,NOMDT,
                       NOBLINK,NORMAL-VIDEO,NOUNDERSCORE,
                       NOCOLOR)
         NODELIMIT
      LITERAL STRING
```

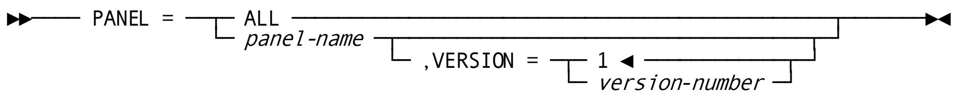
```
PFLD: AUTOPF00005                               AT (12,45)
         ATTRIBUTES = (ALPHANUMERIC,UNPROTECTED,NONDETECTABLE,DISPLAY,NOMDT,
                       NOBLINK,NORMAL-VIDEO,NOUNDERSCORE,
                       NOCOLOR)
         DELIMIT SKIP
DFLD: EMP-PHONE-0415                                    OF EMPLOYEE
         HELP SOURCE NONE
         OPTIONAL
         EXTERNAL PICTURE IS  INTERNAL
             ZEROED WHEN NULL  DISPLAY WHEN ZERO
         INPUT:  JUSTIFY LEFT, UPPER NO, DATA YES, PAD NO
         OUTPUT: BACKSCAN NO, DATA YES
MAP LOAD MODULE GENERATED IN LOAD AREA FOR CEXME028, SIZE =    424


 END OF CA IDMS/DC MAP UTILITY
```

PANEL Statement

A PANEL statement designates a panel occurrence to be processed by a PROCESS statement. The following processes apply to panel occurrences:

- *REPORT* prints a screen image and report for specified panel occurrences.
- *IMAGE* prints a screen image for specified panel occurrences.
- *DECOMPILE* produces the default and non-default compiler source statements for specified panel occurrences. DECOMPILE also produces its output as card images with syntax in fixed locations, one statement clause per line. Therefore, use it when the definition may be modified.
- *TERSE* produces the non-default compiler source statements for specified panel occurrences. Because TERSE fills each line, it is not recommended if you want to edit the output. It should be used when the map is being backed up or migrated.

Syntax



Parameters

ALL

Specifies that all panel occurrences in the data dictionary are to be processed as specified in the PROCESS statement that precedes the PANEL statement. For example, the following sample code prints a screen image for each panel in the data dictionary:

```
PROCESS=IMAGE
PANEL=ALL
```

VERSION=*version-n*

Specifies a panel occurrence to be processed as indicated in the PROCESS statement that precedes the PANEL statement. For example, the following sample code produces a screen image and report for the CEXLEJKD panel:

```
PROCESS=REPORT
PANEL=CEXLEJKD
```

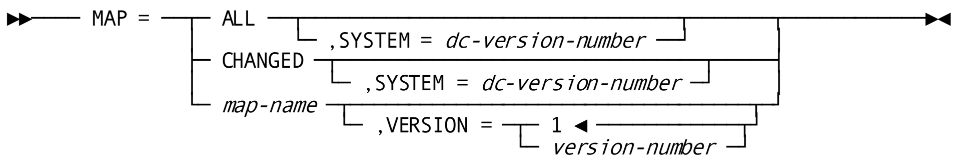
VERSION=*version-n*

Optionally specifies the version number of the panel occurrence. The default is 1.

MAP Statement

A MAP statement is submitted after a PROCESS statement to designate the map occurrences to be processed.

Syntax MAP Statement



Parameters

ALL,SYSTEM=dc-version-n

Specifies that all map occurrences in the data dictionary are to be processed as specified in the PROCESS statement that precedes the MAP

statement. For example, the sample code shown below loads all maps in the data dictionary:

```
PROCESS=LOAD
MAP=ALL
```

MAP=ALL cannot be specified with PROCESS=DELETE; this restriction prevents the inadvertent deletion of map load modules.

SYSTEM=dc-version-n optionally specifies that only map occurrences associated with the specified CA IDMS system are processed. For example, the sample code shown below loads all maps in system 7:

```
PROCESS=LOAD
MAP=ALL,SYSTEM=07
```

CHANGED,SYSTEM=dc-version-n

Specifies that modified map occurrences are to be processed as specified in the preceding PROCESS statement. All map occurrences in the data dictionary that have been modified since they were last generated are processed. For example, the sample code shown below loads all modified and ungenerated maps in the dictionary, generates a report for each map, and finally decompiles each map:

```
PROCESS=ALL
MAP=CHANGED
```

MAP=CHANGED cannot be specified with PROCESS=DELETE; this restriction prevents the inadvertent deletion of map load modules.

SYSTEM=dc-version-n optionally specifies that only modified map occurrences that are associated with the specified CA IDMS system are processed.

Map-name,VERSION=version-n

Specifies that the named map occurrence is to be processed as specified in the PROCESS statement that precedes the MAP statement. For example, the following sample code decompiles map CEXME104:

```
PROCESS=DECOMPILE
MAP=CEXME1_4,VERSION=1
```

VERSION=version-n (default is 1) optionally specifies the version number of the map occurrence; the default is 1.

Considerations

The MAP clause also applies to corresponding panel occurrences when both of the following conditions apply:

- The MAP statement names a map occurrence created by batch statements for automatic panel definition or by the online mapping compiler.
- The MAP statement modifies a REPORT, IMAGE, DECOMPILE, TERSE, and/or DELETE operation, as specified in the PROCESS statement that precedes the MAP statement.

A PANEL statement should not be included when the MAP clause conforms to both of the conditions listed above.

Multiple processes and multiple MAP and PANEL statements can be included in a single batch utility run, as shown below:

```
PROCESS=REPORT,DECOMPILE
MAP=MAINTMAP
MAP=CUSTMAP
MAP=ORDMAP
PROCESS=REPORT
PANEL=MAINTPAN
PANEL=CUSTPAN
PANEL=ORDPAN
PROCESS=DELETE
MAP=METMAP
```

The sample code shown above requests the following operations:

- REPORT and DECOMPILE operations for maps MAINTMAP, CUSTMAP, and ORDMAP
- REPORT operations for maps MAINTPAN, CUSTPAN, and ORDPAN
- DELETE operation for the METMAP load module

Batch Utility JCL

The batch utility executes in update mode. Run the batch utility through the central version or as a local CA IDMS/DB program with active journal files to protect the integrity of the data dictionary.

z/OS JCL

```
//RHDCMPUT EXEC PGM=RHDCMPUT,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=idms.sysmsg.ddldcmsg,DISP=SHR
//SYSLST   DD SYSOUT=A
//SYSPCH   DD SYSOUT=A
//SYSIDMS  DD *
DMCL=dmcl-name
DICTNAME=appldict
sysidms parameters
//SYSIPT   DD *
control statements
```

idms.dba.loadlib	Data set name of load library that contains the DMCL and database name table load modules
idms.custom.loadlib	Data set name of the load library that contains the customized CA IDMS executable modules
idms.cagjload	Data set name of load library that contains the CA IDMS executable modules that do not require customization
idms.sysctl	Data set name of SYSCTL file
idms.sysmsg.ddldcmsg	Data set name of the system message area
sysidms parameters	A list of the SYSIDMS parameters that pertain to this job.
dmcl-name	The name of the dictionary the DMLF precompiler should access
appldict	The name of the application dictionary that should be accessed

Note: For more information about the SYSIDMS parameter, see the *CA IDMS Common Facilities Guide*.

Local Mode

To execute the batch utility in local mode:

- Remove the sysctl DD statement.
- Add the following statements:

```
//dictdb   DD DSN=idms.appldict.ddldml,DISP=OLD
//dloddb   DD DSN=idms.appldict.ddldclod,DISP=OLD
//sysjrnl DD DSN=idms.tapejrnl,DISP=(NEW,CATLG),UNIT=tape
```

dictdb	DDname of application dictionary definition area
idms.appldict.ddldml	Data set name of application dictionary definition area
dloddb	DDname of application dictionary definition load library area
idms.appldict.ddldclod	Data set name of application dictionary definition load library area
sysjrnl	DDname of the tape journal file
idms.tapejrnl	Data set name of the tape journal file

z/VSE JCL

```
// UPSI   b
// DLBL   SYSIDMS,'#SYSIPT'
// EXTENT sys020,nnnnnn,,,ssss,llll
// ASSGN  sys020,DISK,VOL=nnnnnn,SHR
// EXEC   RHDCMPUT
SYSIDMS parameters
/*
```

b	Appropriate UPSI switch, 1-8 characters, if specified in the IDMSOPTI module
#SYSIPT	If #SYSIPT is used, the individual parameters must be listed in the following SYSIDMS parameters statement. Can also be defined as a disk dataset, where #SYSIPT is replaced with the name of the file containing the SYSIDMS parameters and the parameters are not listed separately
user.output	File-id of output file
sys020	Logical unit assignment of output file
nnnnnn	Volume serial number of disk storage device
ssss	Relative starting track
llll	Number of tracks
SYSIDMS parameters	A list of the SYSIDMS parameters that pertain to this job.

Note: For more information about the SYSIDMS parameter, see the *CA IDMS Common Facilities Guide*.

z/VM JCL

```
FILEDEF SYSLST PRINTER
FILEDEF SYSPCH DISK syspch output a
FILEDEF SYSIDMS DISK sysidms input a
FILEDEF SYSIPT DISK util input a
GLOBAL LOADLIB IDMSLIB
OSRUN RHDCMPUT
```

syspch output a	Filename, filetype, and filemode of the card-image output file
sysidms input a	Filename, filetype, and filemode of the file that contains the SYSIDMS parameters
util input a	Filename, filetype, and filemode of the file containing batch utility control statements

Note: For more information about the SYSIDMS parameter, see the *CA IDMS Common Facilities Guide*.

Local Mode

To execute the batch utility in local mode:

1. Specify that RHDCMPUT is executing in local mode by performing one of the following:

   - Link RHDCMPUT with an IDMSOPTI program that specifies local execution mode

   - Modify the OSRUN statement:

     ```
     OSRUN RHDCMPUT PARM='*LOCAL*'
     ```

     Note: This option is valid only if the OSRUN command is issued from a System Product interpreter or an EXEC2 file.

   - Specify *LOCAL* as the first input parameter of the file identified by util input a

2. Add the following statements before the OSRUN statement:

   ```
   FILEDEF dictdb DISK dictdb dictfile d
                  (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
   FILEDEF j1jrnl DISK j1jrnl jrnlfile k
                  (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
   FILEDEF dloddb DISK dloddb dictfile f
                  (RECFM F LRECL pppp BLKSIZE pppp XTENT nnnn
   ```

dictdb	DDname of the application data dictionary
dictdb dictfile d	Filename, filetype, filemode of the application data dictionary file

pppp	Page size of the file
nnnn	Number of pages in the file
j1jrnl	DDname of the first disk journal file
jljrnl jrnlfile k	Filename, filetype, and filemode of the first disk journal file
dloddb	DDname of the data dictionary load area
dloddb dictfile f	Filename, filetype, and filemode of the data dictionary load area

Sample JCL

The input job stream for the batch utility with batch utility statements and JCL is shown here. The map, CEXME028, which was defined using the online compiler, is decompiled by the sample code; the resulting source is sent to a card-image data set; and a report for the map is produced. Executable code for the map CEXME028 is compiled and loaded into the DDLDCLOD area of the data dictionary.

```
//MPUT81    EXEC PGM=RHDCMPUT,REGION=4096K
//STEPLIB   DD DSN=DBDC.SYSTEM81.R150.NTWKLOAD,DISP=SHR
//          DD DSN=DBDC.SYSTEM81.CUSTOM.LOADLIB,DISP=SHR
//          DD DSN=DIST.CAGJF0.CAGJLOAD,DISP=SHR
//SYSCTL    DD DSN=DBDC.SYSTEM81.SYSCTL,DISP=SHR
//SYSLST    DD SYSOUT=*
//SYSPCH    DD SYSOUT=*
//SYSIDMS   DD *
  DICTNAME=TSTDICT
//SYSIPT    DD *
  PROCESS=ALL
  MAP=CEXME028,VERSION=00001
```

Error Messages

If errors are present in batch utility source statements, the utility returns diagnostic messages in the listing generated with each run. Messages issued by the batch utility are displayed in the following format:

```
nnnnnn    severity-level    message-text
```

The following information is represented by the previous format:

- *Nnnnnn* represents the 6-digit identifying number for the error message.
- *Severity-level* represents the severity associated with the message:
  - *W (warning)* identifies potential problems in the source code. The statement that contains the potential error is processed and processing continues.
  - *E (error)* identifies erroneous input. The statement that contains the error is rejected and processing continues. The source code must be corrected before the code can be successfully compiled.
  - *F (fatal)* identifies errors that affect more than one map source statement. Processing is terminated and the errors must be corrected before the source code can be compiled.
- *Message-text* represents the error message that applies to the given error.

Appendix A: Integrated Data Dictionary Mapping Entities

This appendix describes about integrated data dictionary mapping entities.

This section contains the following topics:

Overview

The CA IDMS mapping compilers make extensive use of the data dictionary as a source of information and as a storage location for map-related entity occurrences and map load modules. By using this central resource, the mapping compilers share information with other CA products, thus promoting data integrity and stability.

This appendix focuses on the integration and interaction of the mapping compilers with other CA data management products. The following topics are presented:

- Data dictionary entities used by the mapping compilers
- Data dictionary entities updated by the mapping compilers
- Critical changes to data dictionary entities and related recompilation requirements
- Coordinated use of the batch and online mapping compilers

Data Dictionary Entities Used by the Mapping Compilers

When is the Data Dictionary Used?

The batch and online compilers retrieve information from and update information in the data dictionary at map compilation and map runtime:

- At *map compilation*, the batch and online compilers use map-related and table data dictionary entities, as follows:
  - Both map compilers verify that each specified occurrence is defined in the data dictionary.
  - Both compilers access information from record elements, such as the internal picture, external picture, or edit table defined for or associated with a map definition.
- At *program runtime*, application programs invoke map load modules as needed. A map load module can, in turn, invoke stand-alone tables and map help load modules stored in the load area.

Entities used by Compilers

The following data dictionary entities are used by the mapping compilers:

- Element occurrences
- Record occurrences
- Map occurrences
- Message occurrences
- Table occurrences
- Map and table load modules
- Help modules

The establishment, use, and/or modification of each of these entity occurrences is discussed, following a brief discussion of the builder codes used to identify the owner of many entity occurrences.

Builder Codes

What is a Builder Code?

When an entity occurrence is established in the data dictionary, a builder code is assigned to the occurrence. The builder code designates the component of the CA data management system that owns the entity and is allowed to make structural modifications to the entity. If the entity occurrence is subsequently used by a different product, the builder code may change to reflect the current use of the entity.

Some components, such as CA-IDD, access the builder code when modifications to an entity occurrence are attempted. Builder codes and their associated components are shown in the following table:

Builder Code	Input Source
C	CA IDMS mapping compilers
D	DDDL compiler
G	CA IDMS-CV/DC sysgen compiler (source records)
R	CA IDMS-CV/DC sysgen compiler (object records)
S	Schema compiler
V	Subschema compiler
M	DML processors
A	CA ADS dialog generator
P	CA ADS application generator
X	CA IDMSDIRL utility

For example, a CA-IDD-built occurrence has a builder code of D, which specifies that IDD owns the occurrence and that modifications can be made to the occurrence.

Copying a CA-IDD-built Occurrence

Consequences of copying a CA-IDD-built occurrence are described as follows:

- *If the occurrence is copied into a schema*, the builder code changes to S. A builder code of S indicates to the DDDL compiler that only nonstructural modifications, such as commentary and documentational entries, can be made by the DDDL compiler.
- *If the occurrence is copied into both a map and a schema*, the builder code changes to S.
- *If the occurrence is copied into a CA IDMS map* by the mapping compilers, the builder code changes to C. A builder code of C indicates to the DDDL compiler that limited modifications can be made.

When a schema or a CA IDMS map is deleted, the builder code changes back to D and modifications can then be made by the DDDL compiler.

Note: For more information about builder codes, see the CA IDMS System *Generation Guide*.

Element Occurrences

What is an Element?

A record element is a logical subdivision of a record; an element cannot be addressed without first addressing the record to which the element belongs.

Element occurrences define group or elementary data items that can be used in CA-IDD-built records or CA IDMS/DB schema-built records. Elements used by maps must be defined in the dictionary and must be members of records prior to map compilation.

Establishing Element Occurrences

Element occurrences can be established in the data dictionary in one of the following ways:

- The *ADD ELEMENT statement submitted to the DDDL compiler* establishes a free-standing element that can be included in a CA-IDD-built record or that can be included automatically in a schema-built record.
- The *RECORD statement submitted to the DDDL compiler*, in conjunction with the RECORD ELEMENT substatement or the COBOL substatement, establishes a CA-IDD-built record and all elements that participate in that record. If the element specified in the substatement matches an existing element record, the existing record is used. Otherwise, a new element occurrence is created.
- The *RECORD DESCRIPTION section of schema DDL* submitted to the schema compiler establishes CA IDMS/DB schema-built records in the data dictionary. All elements that participate in the schema records are automatically established in the data dictionary.

Elements are the building blocks that form records. The batch and online compilers use element occurrences as they appear in records rather than as free-standing items.

How the Information is Used

Information defined for the record element in the data dictionary is used by either the batch or online compiler:

- The *internal picture* for the element is used to derive characteristics of the element, such as length or usage.
- The *external picture* for the element is used to determine the external characteristics of a field if both of the following conditions are true:
  - The external picture is defined at the element level and not overridden at the map-field level.
  - Automatic editing is enabled for the map field.
- The *edit table* is used by automatic editing and error-handling in editing data if both of the following conditions are true:
  - The edit table is defined at the record level and not overridden at the map-field level.
  - Automatic editing is enabled for the map field.
- The *code table* is used by automatic editing in encoding and decoding data if both of the following conditions are true:
  - The code table is defined at the record level and not overridden at the map-field level.
  - Automatic editing is enabled for the map field.

Considerations

An element occurrence defined in the data dictionary can be modified or deleted. The following considerations apply to the deletion of element occurrences:

- An element that participates in a record cannot be deleted.
- An element that participates in a group element structure cannot be deleted.

Critical changes to dictionary-defined record elements, such as a change in an element's picture, necessitate regenerating maps and recompiling programs that use those maps (For more information, see "Critical Changes (see page 304)".).

Record Occurrences

What is a Record Occurrence?

A record occurrence is the basic addressable unit of data in CA IDMS/DB. A record consists of a fixed or variable number of characters subdivided into units called elements. Records that are used by maps must be defined in the dictionary prior to map compilation.

Establishing a Record Occurrence

Record occurrences can be established in the data dictionary in one of the following ways:

- The *ADD RECORD statement submitted to the DDDL compiler* establishes a record entity occurrence in the data dictionary.

  Note: For more information about the ADD RECORD statement and other DDDL statements, see the *CA IDMS IDD DDDL Reference Guide*.

- The *RECORD DESCRIPTION SECTION of the schema DDL* submitted to the schema compiler establishes record entity occurrences in the data dictionary.

  Note: For more information about the DDL schema compiler, see the *CA IDMS Utilities Guide*.

Considerations

The following considerations apply to the modification or deletion of records:

- If a record participates in a CA IDMS/DB schema, the record cannot be deleted from the dictionary. Additionally, no structural modifications can be made to the record; however, commentary and documentational entries can be added or modified.
- A record synonym that participates in a CA IDMS/DB schema, subschema, or CA IDMS map cannot be removed or excluded from the dictionary.
- A view id (established by a DDDL PROGRAM statement) that is active in a CA IDMS/DB subschema cannot be removed from the dictionary.

- A record that participates in a map can be modified, with the following exceptions:
  - A record element that participates in a map cannot be removed from the record or replaced. This restriction applies also to group elements that contain subordinate elements that participate in maps. REPLACE RECORD ELEMENTS implicitly performs a remove and an insert operation.

    Several DDDL commands cause the removal or replacement of record elements. If the record elements or subordinate elements to be removed or replaced participate in maps, the operation is not performed. If the record elements or subordinate elements do not participate in maps, the operation is performed as usual. The following commands perform a remove operation:

    - REMOVE RECORD ELEMENT
    - REPLACE RECORD ELEMENT
    - REMOVE ALL
    - COBOL substatement

    The following commands perform a replace operation:

    - REBUILD RECORD ELEMENTS followed by RECORD ELEMENT substatements
    - REPLACE RECORD ELEMENT
  - The occurrence count of the OCCURS clause cannot be decreased if the decrease causes a map field to be made obsolete. For example, if the twelfth occurrence of a record element is used by a map, and an attempt is made to modify the record and decrease the occurrence count to eleven, an error occurs and the modification is not performed. This rule applies to group record elements as well as to elementary record elements.
- Certain modifications to map-owned occurrences require that the map be recompiled if the map is to reflect the changes in the occurrence. Other modifications require that the map, as well as the programs using the map, be recompiled.

  Note: For more information, see Critical Changes.

Panel Occurrences

Panel occurrences stored in the data dictionary define display screens. Panel occurrences are associated with map occurrences and can be established in the data dictionary in any of the following four ways:

- The *CA IDMS online mapping compiler* establishes panel occurrences in the dictionary when it creates a map. Panels established with this option are identified by the -OLMPANEL suffix.
- The *ADD PANEL statement submitted to the CA IDMS batch compiler* establishes panel occurrences in the data dictionary.
- The *ADD MFLD statement (for MAP AUTOPANEL) submitted to the CA IDMS batch compiler* establishes panel occurrences in the data dictionary. Panels established with this option are identified by the -AUTOPANEL suffix.
- The *ADD PANEL statement submitted to the DDDL compiler* establishes panel occurrences in the data dictionary. Such panels are documentational only.

  Note: For more information about the DDDL ADD PANEL statement, see the *CA IDMS IDD DDDL Reference Guide*.

Panel occurrences are stored in the DDLDML area of the data dictionary.

Considerations

The CA IDMS mapping facility batch and online compilers can be used to modify or delete panel occurrences. The following considerations apply to the modification or deletion of panel occurrences:

- A panel record created by batch compiler statements for manual panel definition is not deleted when maps associated with the panel are deleted.
- A panel occurrence generated either by batch compiler automatic panel definition or by the online compiler is affected by the deletion of an associated map:
  - *The panel occurrence is automatically deleted* when the panel occurrence has not been associated with any other map occurrences.
  - *The panel occurrence is not affected* when the panel occurrence has been associated with other map occurrences.

    Important: If a map was originally compiled using the online compiler, and then decompiled and recompiled in batch, panel occurrences will not be automatically deleted.
- A panel occurrence associated with one or more maps cannot be deleted until all associated maps have been deleted.
- A panel occurrence established by the CA IDMS mapping compilers cannot be modified or deleted by the DDDL compiler, except to modify comments.

Changing a DEVICE Specification

The DEVICES specification of a panel cannot be modified. To change a DEVICES specification, the following procedure must be used:

1. Decompile the panel and associated maps, saving the decompiled source for later use.
2. Delete all maps associated with the panel.
3. Delete the panel occurrence itself.
4. Define a new panel with the updated DEVICES specification by using the decompiled source.
5. Define the associated maps using the decompiled source.

Map Occurrences

Map occurrences stored in the data dictionary associate record elements with panel field occurrences for existing panel occurrences.

Establishing Map Occurrences

Map occurrences can be established in the data dictionary in any of the following three ways:

- By using the CA IDMS online mapping compiler
- By using the ADD MAP or ADD MAP AUTOPANEL statements of the CA IDMS batch compiler
- By using the ADD MAP statement of the CA-IDD DDDL compiler (such maps are documentational only);

  Note: For more information, see the *CA IDMS IDD DDDL Reference Guide*.

Map occurrences are stored in the DDLDML area of the data dictionary.

The CA IDMS mapping facility online and batch compilers can modify or delete map occurrences from the data dictionary.

Considerations

The following considerations apply to the modification or deletion of map occurrences:

- Deleting a map occurrence that was generated by batch compiler statements for manual panel definition does not affect the associated panel occurrence.
- Deleting a map occurrence that was generated either by either the online or batch compiler statements for automatic panel definition affects the associated panel occurrence:
  - The associated panel occurrence is deleted unless the panel has been associated with additional map occurrences.
  - The associated panel occurrence is not affected if the panel has been associated with additional map occurrences.
- Map occurrences that are created by the online or batch compilers cannot be deleted by the DDDL compiler; the DDDL compiler can be used only to modify comments.

Message Occurrences

Message occurrences stored in the data dictionary define informational messages. These messages can be established only through the ADD MESSAGE statement submitted to the DDDL compiler. Such messages can be modified at any time through the DDDL compiler.

Note: For more information, see the *CA IDMS IDD DDDL Reference Guide*.

Messages stored in the data dictionary can be associated with maps by the mapping facility batch or online compiler. A CA ADS dialog or application program can include statements that cause a map to be redisplayed due to input errors. Error messages are displayed for each field in error in the message field (if any) defined for the map. If a map has no message field, error messages are not displayed and processing continues (with CA ADS the message is displayed on the CA ADS default message screen).

A message can be specified in a map field occurrence generated by the CA IDMS mapping facility and thus be included in a map load module. This type of message does not constitute a message occurrence and can only be modified by using the mapping facility online or batch compilers.

Table Occurrences

Table occurrences stored in the data dictionary define edit and code tables that are used by the automatic editing feature of the CA IDMS mapping facility. Edit tables contain lists of single values and/or ranges of values against which data field values in a map are verified. Code tables contain lists of values according to which data field values are encoded and record element values are decoded.

Types of Tables

Three types of edit and code tables exist in the data dictionary:

- A *built-in table* is created by the EDIT/CODE TABLE clause in either the RECORD ELEMENT or the COBOL substatement of the DDDL RECORD statement. A built-in table is associated with an individual record element in the dictionary.
- A *linked stand-alone table* is created by the DDDL ADD TABLE statement of CA-IDD. A stand-alone table is not associated with a particular record element in the dictionary. The term *linked* indicates that a copy of such a table is incorporated into the map load module.
- An *unlinked stand-alone table* is created and generated by the DDDL ADD TABLE statement of CA IDD. A stand-alone table is not associated with a particular record element in the dictionary. The term *unlinked* indicates that such a table is dynamically loaded as a separate load module at program runtime.

A built-in table is also called a tightly coupled table since the table is associated with an individual record element in the dictionary. A stand-alone table is also called a loosely coupled table since the table is not associated with a particular record element in the dictionary.

Considerations

A table can be modified or deleted at any time by an authorized user once the table is established in the data dictionary. The following considerations apply to the modification or deletion of a table occurrence:

- *When a built-in table is modified*, any map load modules that use the record element in which the table is defined must be recompiled. This step is necessary because an element and its tables are incorporated into map load modules that specify the element. It is not necessary to recompile dialogs or programs that use the recompiled maps.
- *When a linked stand-alone table is modified*, any maps that use the table must be recompiled. This step is necessary because a linked stand-alone table is incorporated into map load modules that specify the table. It is not necessary to recompile dialogs or programs that use the recompiled maps.
- *When an unlinked stand-alone table is modified*, it is not necessary to recompile maps that access the table.

Notes:

- For more information about edit and code tables, see the appendix "Generating Edit and Code Tables."
- For more information about the DDDL RECORD or ADD TABLE statements, see the *CA IDMS IDD DDDL Reference Guide*.

Map and Table Load Module Occurrences

Establishing Load Modules

Load module occurrences in the data dictionary define modules that can be used by the CA IDMS/DB central version, CA IDMS, and CA ADS Load modules are stored in the DDLDCLOD area of the data dictionary. Map, table, and help load modules are established in the data dictionary in the following ways:

- The *GENERATE clause of the ADD TABLE statement submitted to the DDDL compiler* establishes a loosely coupled edit or code table as a load module in the data dictionary.
- The *Compile action on the Main Menu* establishes a map as a load module in the data dictionary.

  Note: The help load module is created when its corresponding map is compiled.
- The *PROCESS=LOAD statement submitted to the CA IDMS batch utility* establishes a map as a load module in the data dictionary.

Considerations

The following considerations apply to the modification or deletion of a load module:

- Once a load module has been stored in the data dictionary, only a user with ALL authority can delete the load module or have it punched to a SYSPCH file.
- When a previously loaded map load module is recompiled, the DCMT VARY PROGRAM *map-name* NEW COPY command must be used to ensure that the new version of the module is loaded in either of the following cases:
  - The batch utility is used to recompile the module.
  - MAPC is used to recompile the module and the OLM sysgen option NEW COPY IS N is defined for the online compiler.

  For either of the previously specified cases, if the DCMT VARY PROGRAM command is not used and the previous version of the map load module has not been deleted or overlayed, the old version of the map load module is used.
- When a map is recompiled and the sysgen option is NEW COPY IS N, the load module should also be varied NEW COPY.
- When a stand-alone edit or code table description is modified, it must be recompiled by the GENERATE option of the CA-IDD DDDL TABLE statement if the table load module is to reflect changes in the table definition.
- When a stand-alone edit or code table load module is modified, the map load module that uses that table as a linked table must be recompiled if it is to reflect changes in the table.

Data Dictionary Entities Updated by Mapping Compilers

At map generation, the batch and online compilers update the dictionary by adding, modifying, or deleting map occurrences and load modules and by establishing links between maps and dictionary entities that the maps access. For example, a record is updated to reflect the use of that record by the map. While the map occurrence exists in the dictionary, the record cannot be deleted.

The mapping facility batch and online compilers establish and maintain map and panel occurrences by updating the following data dictionary records:

- **MAP-098** occurrences represent map occurrences.
- **MAPRCD-125** occurrences relate CA IDMS maps to a schema or work record that the map uses.
- **MAPFLD-124** occurrences represent map field occurrences.
- **PANELFLD-121** occurrences represent panel field occurrences.

- **PFLD-DATA-147** occurrences are logical extensions of PANELFLD-121 occurrences that contain device-dependence tables for panel fields.
- **PROG-051** occurrences represent maps as programs in the data dictionary; a flag in the record indicates that the record is a map.
- **MODMAP-195** occurrences relate CA IDMS maps to edit and code tables and to modules containing help text.

These data dictionary records are shown in the following table along with the CA IDMS statements and operations that update each record in the data dictionary:

Map Compiler Statements

	PANEL	PFLD	MAP	MAP MFLD	MFLD AUTO-PANEL	(AUTO-PANEL)
MAP-098			X		X	
MAPPRCD-125			X		X	
MAPFLD-124				X		X
PANEL-118	X					
PANELFLD-121		X				X
PFLD-DATA-147		X				X
PROG-051						
DDLDCLOD AREA						
Map Utility Load statement						X

Online Mapping Compiler Screens

	Initial Definit	Added Records	Correct /Incorrect	Field Select	Field Edit	Extend Field Edit
MAP-098	X	X	X	F		

	Initial Definit	**Added Records**	**Correct /Incorrect**	**Field Select**	**Field Edit**	**Extend Field Edit**
MAPRCD-125	X	X				
MAPFLD-124				F	X	X
PANEL-118	X					
PANELFLD-12 1		X		F	X	X
PFLDDATA-14 7		X		F	X	X
PROG-051				G		
DDLDCLOD AREA				G		

Notes:

- F indicates records updated on FINISH or GENERATE.
- G indicates records updated on GENERATE only.

Note: For more information about data dictionary records, see the *CA IDMS Dictionary Structure Reference Guide*.

Critical Changes

A *critical change* is one that requires entities that use or are used by the changed entity to be recompiled. The *date/time stamps* will be in conflict until all necessary entities are recompiled. If the date/time stamp for a map load module conflicts with the date/time stamp of a dialog or program that uses the map load module, an error results.

What is a Critical Change?

The following types of changes, which are considered critical, update the date/time stamps for panel and map occurrences:

- Adding a variable field to a map/panel
- Deleting a variable field from a map/panel
- Changing a pageable map to a nonpageable map, or vice versa
- Changing the version of a record

To update the date/time stamp for the map load module and incorporate the critical changes that were made, recompile the map load module.

What to Recompile

The following entities must be recompiled when a map load module is recompiled due to a critical change:

- CA ADS dialogs (if any) that use the map
- Application programs (if any) that use the map

The developer can identify the dialogs and programs that have been compiled against a map by displaying the map occurrence with CA-IDD.

When a developer copies a map and optionally edits existing field definitions for the newly copied map, it is not considered a critical change. Therefore, the date/time stamp is the same for the original and the copied map. Alternative maps must all have the same date/time stamp.

Important: If a developer copies a map and then *moves the fields to different positions*, it is considered a critical change.

Note: For more information about alternative maps, see "Alternative Maps".

IDD produces a list of maps that must be recompiled when CA IDD is used to modify a record such that map recompilation is necessary. The list of maps is followed by a message that informs the developer if dialogs and programs that use the map need to be recompiled when the map is recompiled.

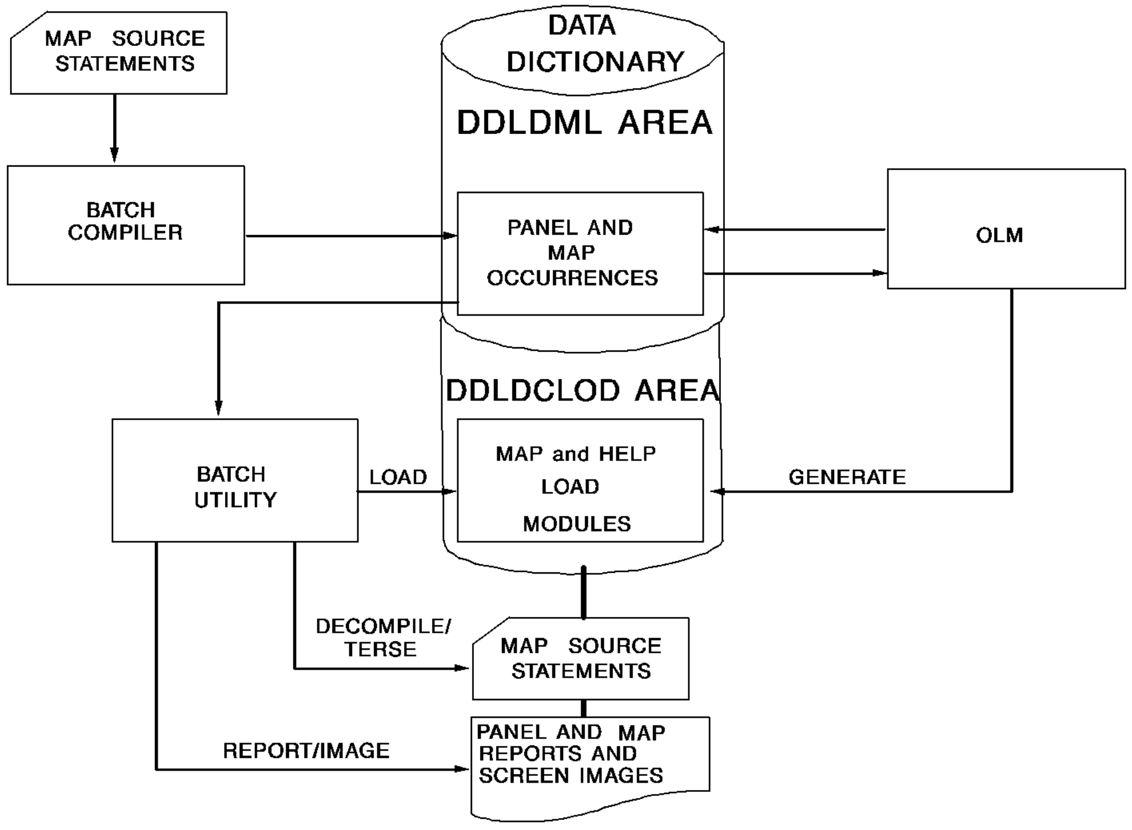
IDD modifications and regeneration/recompilation requirements for maps and programs are summarized in the following table.

Note: For more information about modification of records and record elements, see the *CA IDMS IDD DDDL Reference Guide*.

IDD DDDL Modification	**Map Regeneration** Required for Maps	**Dialog/Program Regeneration** Required for Dialog/Programs
PICTURE	X	X
USAGE	X	X
REDEFINES	X	X
OCCURS *count*	X	X
SIGN	X	X
EDIT TABLE	X	
CODE TABLE	X	
EXTERNAL PICTURE	X	
RECORD ELEMENT specification	X	X

Coordinated Use of the Online and Batch Compilers

Use of the online mapping compiler, the batch compiler, and the batch utility can be coordinated to develop and maintain maps. The relationships among the data dictionary, batch compiler, batch utility, and the online compiler are illustrated in the following figure:



For example, a map developer might perform the following sequence of actions to coordinate the use of the online compiler, the batch compiler, and the batch utility:

1. Develop a new map by using the online compiler
2. Revise the map in response to a major revision of a data dictionary entity by performing the following actions:
   a. Obtain map source statements by using the DECOMPILE or TERSE process of the batch utility
   b. Delete the occurrences and load module for the map by using the online compiler
   c. Update the decompiled source statements, as necessary
   d. Compile the updated source code, redefining the map
   e. Generate a load module for the updated map by using the LOAD process of the batch utility

The following considerations apply to the coordinated use of the mapping compilers:

- **The online compiler** can be used to modify or delete a batch-generated map under the following conditions:
  - The name of the map panel is composed of the map name and the suffix -OLMPANEL.
  - The map does not define device groupings.
  - The map does not contain device specifications for devices smaller than 24X80.
- The **batch compiler and utility** can be used to decompile, revise, and recompile a map created by the online compiler. Panels created by the online compiler are processed automatically by the map utility REPORT and DECOMPILE processes; the panel need not be explicitly named.

  DECOMPILE or TERSE move the definition of a map from one dictionary to another. The map can then be recompiled on the target dictionary. A map load module can be moved from one load area to another (or to a load library) using the DDDL PUNCH command, but programs and dialogs cannot be compiled against a map for which there is no source definition in the DDLDML area of the dictionary.

Appendix B: Using Glass TTY Terminals

This appendix discusses about using glass TTY terminals.

This section contains the following topics:

Overview

Most maps prepared by using the online or batch compiler and utility can be displayed on a visual-display teletypewriter terminal (glass TTY). Before attempting to map to or from a glass TTY, the user must prepare a device independence table for the given TTY, as described throughout this appendix.

Note: Key names and control codes discussed in this appendix do not apply to all glass TTY terminals. Documentation of a given TTY terminal should be consulted to verify the keys and codes used by that terminal.

Types of Tables

Several different device independence tables can be created at a site, as described as follows:

- A unique device independence table may have to be created for each type of glass TTY terminal available at a site. In other cases, a given device independence table can be used for a group of terminals.
- Different device independence tables can be created for the same type of TTY if, for example, some users need to see different attribute byte symbols on their TTY screens or are accustomed to associating different functions with particular function keys.

At the beginning of a terminal session, the operator loads a specific device independence table by using the DCUF SET SCREEN statement. The RHDCTAPR module uses the specified TTY device independence table during runtime.

Note: For more information about the DCUF SET SCREEN statement, see the *CA IDMS System Generation Guide*.

Steps

To create a device independence table, the user must perform the following steps:

1. Prepare statements that establish data conversion information for the given TTY.
2. Assemble and link those statements into an RHDCTTBL module and execute the module.

These steps are discussed separately, following information about the TTY environment and the restrictions imposed on mapping by TTY limitations.

Note: Throughout this appendix, both 3270- and 3279-type terminals are referred to as 3270-type terminals.

TTY Environment

Cursor Position

Cursor position on mapout is specified in the map-definition. A mapout operation writes a map to the TTY and positions the cursor as it would be positioned on a 3270-type terminal.

Attribute Byte

Each field on mapout is physically preceded by an invisible attribute byte. A display symbol for the attribute byte can be defined in a device independence table. Attribute byte symbols can be used to mark the location of the field and to inform the operator whether the field is unprotected, delimited, blank and protected, or in error. Default attribute byte symbols are presented in the following table:

Default Symbol	Meaning
+	Unprotected fields
!	Blank, protected field
*	Delimited field
?	A field in error

The attribute byte symbols that are defined in a device independence table provide the terminal operator with information about a given field. The symbols that are presented in this table can be overridden when a device independence table is generated.

Protected Fields

TTY terminals do not physically protect fields; the operator can key characters into any location on the map. Data keyed into a field that is defined as an UNPROTECTED field is transmitted and processed as usual. Data is ignored on mapin if it is typed into a field that is designated as PROTECTED or into a portion of the map on which no fields are defined.

Keys

Each terminal defines the keys or key sequences that cause the cursor to be moved on the screen. Terminal-defined key associations must be repeated in a device independence table that is generated for a given terminal. The key assignments presented in the following table are typical for some glass TTY terminals. Documentation for any given terminal should be consulted for the cursor-movement key assignments that are valid for that terminal.

Typical Cursor Movement Keys

A device independence table specifies the terminal-defined keys or key sequences that are used to move the cursor on the screen. The TTY key associations presented in this table are typical of some glass TTY terminals. Documentation for any given terminal should be consulted for the cursor-movement key assignments that are valid for that terminal.

Default TTY key	Function
<Ctrl>–H	Cursor-left
<Ctrl>–J	Cursor-down
<Ctrl>–K	Cursor-up
<Ctrl>–L	Cursor-right
<Ctrl>–<Home>	Home

TTY control keys can be defined to act like 3270-type attention keys. Attention key assignments that are typical for some glass TTY terminals are listed in the following table. Documentation for any given terminal should be consulted for the attention key assignments that are valid for that terminal.

3270 Key	Typical function of the 3270 key	Default TTY Key	Statement = hex-value-a
<Enter>	Send data to host	Return	CENTER=0D
<Clear>	Return to higher level	<Ctrl>–Z	CCLEAR=1A
<PF1>	Help	<Ctrl>–F	CPF1=06
<PF2>		<Ctrl>–I	CPF2=09
<PF3>		<Ctrl>–R	CPF3=12
<PF4>		<Ctrl>–S	CPF4=13
<PF5>		<Ctrl>–T	CPF5=14
<PF6>		<Ctrl>–U	CPF6=15
<PF7>	Display previous page	<Ctrl>–V	CPF7=16
<PF8>	Display next page	<Ctrl>–W	CPF8=17
<PF9>	Swap screens	<Ctrl>–X	CPF9=18
<PA1>	Refresh screen	<Ctrl>–Y	CPA1=19

The operator presses a TTY key or key sequence to invoke the 3270-type function associated with the key in the device independence table. The right-hand column of this table presents the RHDCTTBL statements that establish these particular key relationships.

Restrictions

TTY limitations impose the following restrictions on maps:

- Maps that contain fields that wrap around from the bottom to the top of the screen are not supported. A map is not a wraparound map if only the cursor returns to the upper left from the lower right coordinate at runtime.
- The last position on the screen is unavailable. For example, data cannot be mapped to or from the position 24,80 on a 24X80 screen.
- Maps with overlapping fields are not supported.

The following restrictions should also be noted:

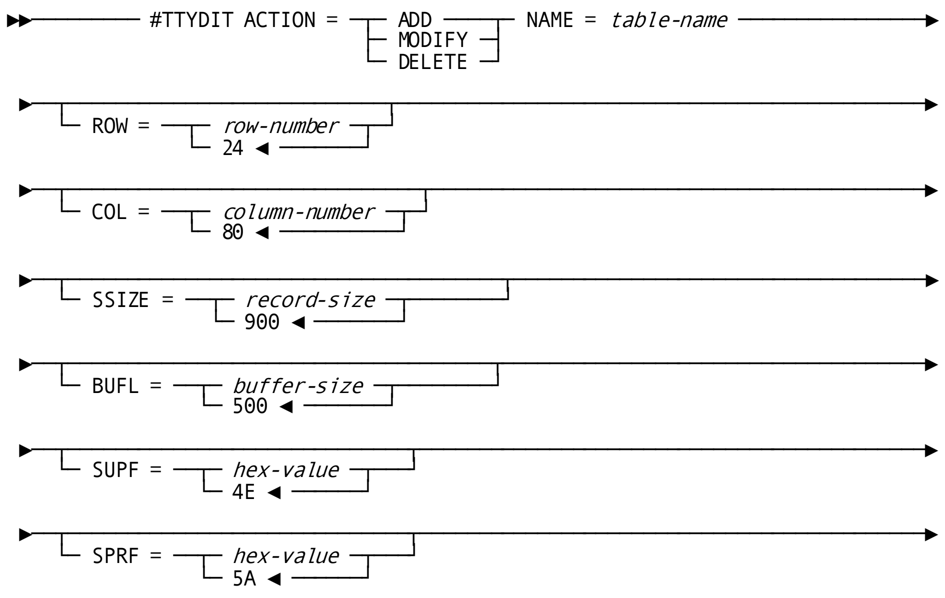
- TTY keys that do not put control codes into the data stream (such as SHIFT/CLEAR and CTRL/Q) should not be used, since RHDCTAPR registers user activity according to the control codes it receives.
- The online mapping compiler can only be run at a 3270-type terminal.
- A map generated through either batch or the online mapping compiler can be displayed on TTY and 3270-type terminals under the following conditions:
  - A map that is wider than a given screen can not display on that screen.
  - A map can only display at a terminal if the screen size for the terminal is specified in the map-definition. A TTY usually has a size of 24X80.
  - A map can only display at a TTY terminal if the existing device independence table for the TTY terminal is specified at the beginning of the terminal session.

Preparing Device Independence Statements

The #TTYDIT macro is the core of the RHDCTTBL utility used to create TTY tables. Through it, the user defines the TTY environment and establishes protocol information.

Protocol coding is accomplished by editing the RHDCTTBL program which contains the #TTYDIT macro. The RHDCTTBL program is delivered with the tape and is installed in the source library. Assembling the RHDCTTBL program creates the TTY protocol device independence table. Executing the RHDCTTBL program adds the device independence table to the load area.

Sample assembly and execution JCL are provided later in this section. Syntax for #TTYDIT macro statements is shown:



```
►─┬──────────────────────────────────────────────────────────►
  └─ DELM = ──┬─ hex-value ─┬┘
              └─ 5C ◄ ──────┘

►─┬──────────────────────────────────────────────────────────►
  └─ FERR = ──┬─ hex-value ─┬┘
              └─ 6F ◄ ──────┘

►─┬──────────────────────────────────────────────────────────►
  └─ HMRW = ──┬─ row-number ─┬┘
              └─ 01 ◄ ───────┘

►─┬──────────────────────────────────────────────────────────►
  └─ HMCL = ──┬─ column-number ─┬┘
              └─ 01 ◄ ──────────┘

►─┬──────────────────────────────────────────────────────────►
  └─ KYBD = ──┬─ N ──┬┘
              └─ Y ◄ ┘

►─┬──────────────────────────────────────────────────────────►
  └─ ASKI = ──┬─ N ──┬┘
              └─ Y ◄ ┘

►─┬──────────────────────────────────────────────────────────►
  └─ ALRM = ──┬─ hex-value ─┬┘
              └─ 07 ◄ ──────┘

►─┬──────────────────────────────────────────────────────────►
  └─ UNLK = ──┬─ hex-value ─┬┘
              └─ 0E ◄ ──────┘

►─┬──────────────────────────────────────────────────────────►
  └─ CTHM = ──┬─ hex-value ─┬┘
              └─ 1E ◄ ──────┘

►─┬──────────────────────────────────────────────────────────►
  └─ UPLN = ──┬─ hex-value ─┬┘
              └─ 0B ◄ ──────┘

►─┬──────────────────────────────────────────────────────────►
  └─ DNLN = ──┬─ hex-value ─┬┘
              └─ 0A ◄ ──────┘

►─┬──────────────────────────────────────────────────────────►
  └─ FRSP = ──┬─ hex-value ─┬┘
              └─ 0C ◄ ──────┘

►─┬──────────────────────────────────────────────────────────►
  └─ BKSP = ──┬─ hex-value ─┬┘
              └─ 08 ◄ ──────┘

►─┬──────────────────────────────────────────────────────────►
  └─ ESC = ──┬─ hex-value ─┬┘
             └─ 1B ◄ ──────┘

►─┬──────────────────────────────────────────────────────────►
  └─ CCLEAR = ──┬─ hex-value ─┬┘
                └─ 1A ◄ ──────┘

►─┬──────────────────────────────────────────────────────────►
  └─ CENTER = ──┬─ hex-value ─┬┘
                └─ 0D ◄ ──────┘
```

```
►─┬──────────────────────┬──────────────────────────────────────────────────►
  └─ CPOSn = hex-value ──┘

►─┬─────────────────────┬───────────────────────────────────────────────────►
  └─ CPFn = hex-value ──┘

►─┬─────────────────────┬───────────────────────────────────────────────────►
  └─ CPAn = hex-value ──┘

►─┬──────────────────────┬──────────────────────────────────────────────────►
  └─ PCURn = hex-value ──┘

►─┬──────────────────────────────────┬──────────────────────────────────────►
  └─ PCLR = ──┬─ hex-value ─┬────────┘
              └─ 1A ◄ ──────┘

►─┬───────────────────────┬─────────────────────────────────────────────────►
  └─ PDROW = ──┬─ Y ─┬────┘
               └─ N ◄┘

►─┬───────────────────────────────────┬─────────────────────────────────────►
  └─ ROWDELM = ──┬─ hex-value ─┬──────┘
                 └─ 00 ◄ ──────┘

►─┬───────────────────────┬─────────────────────────────────────────────────►
  └─ PDCOL = ──┬─ Y ─┬────┘
               └─ N ◄┘

►─┬────────────────────────────────────┬────────────────────────────────────►
  └─ COLDELM = ──┬─ hex-value ─┬───────┘
                 └─ 00 ◄ ──────┘

►─┬─────────────────────────┬──────────────────────────────────────────────►◄
  └─ DECIMAL = ──┬─ Y ─┬────┘
                 └─ N ◄┘
```

#TTYDIT ACTION=ADD/MODIFY/DELETE

Specifies the action to be taken:

- ADD specifies that a new device independence table is to be created. ADD is the default specification if no table with the specified name exists.
- MODIFY specifies that the named device independence table is to be modified. MODIFY is the default specification if the named table already exists.
- DELETE specifies that the named device independence table is to be deleted.

NAME= table-name

Specifies a 3-character name suffix for the table. The table is known to CA IDMS as $TTY@*table-name*. The default for *table-name* is ADM.

ROW=row-n

Specifies the number of screen rows on the given type of glass TTY terminal. The default for *row-n* is 24.

COL=column-n

Specifies the number of screen columns on the given type of glass TTY terminal. The default for *column-n* is 80.

SSIZE=record-size-n

Specifies the size, in bytes, of the area in program variable storage that stores all data to be transmitted. The default for *record-size-n* is 900.

BUFL=buffer-size-n

Specifies the size, in bytes, of the TTY inbound buffer. The default for *buffer-size-n* is 500.

SUPF=hex-value-a

Specifies the character attribute symbol that marks the start of an unprotected field on the TTY screen. The value for *hex-value-a* must be supplied in hexadecimal format. The default for *hex-value-a* is the plus sign (+); hex 4E.

SPRF=hex-value-a

Specifies the character attribute symbol that marks the start of a blank protected field on the TTY screen. The default is 5A.

Note: Any character, including a space character, can be specified for SUPF, SPRF, DELM, and/or FERR. The same character can be specified for more than one attribute symbol.

DELM=hex-value-a

Specifies the character attribute symbol that marks the location of the delimit character on a delimited field. The value for *hex-value-a* must be supplied in hexadecimal format. The default for *hex-value-a* is the asterisk (*); hex 5C.

FERR=hex-value-a

Specifies the character attribute symbol that marks a data field containing erroneous input (as determined by automatic editing or by a user edit module). The value for *hex-value-a* The default for *hex-value-a* is the question mark (?); hex 6F.

HMRW=row-number-n

Specifies the 2-digit coordinate for the cursor home row. The default for *row-number-n* is 01.

HMCL=column-number-n

Specifies the 2-digit coordinate for the cursor home column. The default for *column-number-n* is 01.

KYBD=N/Y

Specifies whether the host should send the control code defined by the UNLK statement to the terminal to unlock the keyboard. Y (the default) specifies that it is necessary to send an unlock character to the terminal. Any other response suppresses the unlock code and should be used when the keyboard does not lock or when the terminal does not recognize an unlock code.

ASKI=N/Y

Specifies whether control codes in subsequent statements specify ASCII or EBCDIC codes; it is not permissible to mix ASCII and EBCDIC codes in a given table specification. Y (the default) indicates that all control codes in the table specification are in ASCII; any other response indicates that all subsequent control codes are in EBCDIC.

ALRM=hex-value-a

Specifies the hex control code to ring the terminal alarm. The default for *hex-value-a* is 07 (the ASCII mnemonic for this code is BEL).

UNLK=hex-value-a

Specifies the hex control code to unlock the keyboard. The default for *hex-value-a* is 0E (the ASCII mnemonic for this code is SO).

CTHM=hex-value-a

Specifies the hex control code that returns the cursor to the home position, as defined by the HMRW= and HMCL= statements presented previously. The default for *hex-value-a* is 1E (the ASCII mnemonic for this code is RS).

UPLN=hex-value-a

Specifies the hex control code for upward cursor movement (cursor up). The default for *hex-value-a* is 0B (the ASCII mnemonic for this code is VT).

DNLN=hex-value-a

Specifies the hex control code for downward cursor movement (cursor down). The default for *hex-value-a* is 0A (the ASCII mnemonic for this code is LF).

FRSP=hex-value-a

Specifies the hex control code for forward cursor movement (cursor right). The default for *hex-value-a* is 0C (the ASCII mnemonic for this code is FF).

BKSP=hex-value-a

Specifies the hex control code for backward cursor movement (cursor left). The default for *hex-value-a* is 08 (the ASCII mnemonic for this code is BS).

ESC=hex-value-a

Specifies the hex control code for the ESC (ESCAPE) key. The default for *hex-value-a* is 1B (the ASCII mnemonic for this code is ESC).

CCLEAR=hex-value-a

Specifies the hex control code that functions as the 3270 clear key aid byte. The default for *hex-value-a* is 1A (the ASCII mnemonic for this code is SUB).

CENTER=hex-value-a

Specifies the hex control code for the RETURN (ENTER) key. The default for *hex-value-a* is 0D (the ASCII mnemonic for this code is CR).

CPOSn= hex-value-a

Where *n* is an integer from 1 to 80, specifies the hex control codes for absolute cursor positions. Each code specifies a unique row/column screen location.

CPFn=hex-value-a

Where *n* is an integer from 1 to 24, specifies hex control codes for control keys. The control code specified for CPF1 is translated to respond as though PF1 were pressed on a 3270-type terminal; the control code for CPF2 functions as PF2; the control code for CPF3 functions as PF3; and so forth.

Default values for *hex-value-a* are presented in the table located at the end of this section. To override either a default or user-specified setting, the user must specify both a new code and a null value for the given setting.

CPAn=hex-value-a

Where *n* is an integer from 1 to 3, specifies hex control codes for control keys that correspond to the PA keys on 3270-type terminals. The default value for CPA1 is 19, which corresponds to CTRL/Y in ASCII. There are no defaults for CPA2 and CPA3. To override either a default or user-specified setting, the user must specify both a new code and a null value for the given setting.

PCURn=hex-value-

Where *n* is an integer from 1 to 3, specifies the hex control codes for the leading control bytes in the cursor positioning protocol. The value specified for PCUR1 is the first byte in the protocol; PCUR2 specifies the second byte; and PCUR3 specifies the third byte (if any). If all three PCUR statements are assigned values, there will be three leading control bytes and PCURLEN must be set to 3. The default for PCUR1 is 1B (ESC in ASCII) and the default for PCUR2 is 3D (= in ASCII). There is no default for PCUR3. The leading position cursor protocol is combined with the row/column delimit protocol (if any) to position the cursor during execution of the application program.

PCLR=hex-value-a

Specifies the hexadecimal control code that clears the screen. This is the protocol transmitted to the TTY terminal. The default for *hex-value-a* is 1A; (the ASCII mnemonic for this code is SUB).

PDROW=Y/N

Specifies whether protocol is needed to delimit the row parameter of the position cursor protocol. N (the default) indicates that protocol is not required.

ROWDELM=hex-value-a

Specifies the hex control code that is the protocol required to delimit the row parameter of the position cursor protocol. This value is used only if PDROW is specified as Y. The default for *hex-value-a* is 00.

PDCOL=Y/N

Specifies whether protocol is needed to delimit the column parameter of the position cursor protocol. N (the default) indicates that protocol is not required.

COLDELM=hex-value-a

Specifies the hex control code that is the protocol required to delimit the column parameter of the position cursor protocol. This value is used only if PDCOL is specified as Y. The default for *hex-value-a* is 00.

DECIMAL=Y/N

Specifies whether the protocol to position the cursor requires decimal numbers for the row/column parameters. N (the default) indicates that row/column is specified in hexadecimal format and that CPOS*n* values are to be used. Y indicates that row/column is specified in decimal format and that specified CPOS*n* values are not to be used. If Y is specified, the row/column numbers are converted to decimal numbers from the 3270 data stream by the glass TTY runtime system.

Absolute Cursor Positions

Each default hex-value-a value in the following table specifies a single row/column screen position for internal use. The ASCII values presented in this table are typical absolute cursor position values. Documentation for any given terminal should be consulted for the absolute cursor position values that are valid for that terminal.

CPOS=	hex -value -a	CPOS=	hex -value -a	CPOS=	hex -value -a	CPOS=	hex -value -a
1	20	21	34	41	48	61	5C
2	21	22	35	42	49	62	5D
3	22	23	36	43	4A	63	5E
4	23	24	37	44	4B	64	5F
5	24	25	38	45	4C	65	60
6	25	26	39	46	4D	66	61
7	26	27	3A	47	4E	67	62
8	27	28	3B	48	4F	68	63
9	28	29	3C	49	50	69	64
10	29	30	3D	50	51	70	65
11	2A	31	3E	51	52	71	66
12	2B	32	3F	52	53	72	67
13	2C	33	40	53	54	73	68
14	2D	34	41	54	55	74	69
15	2E	35	42	55	56	75	6A
16	2F	36	43	56	57	76	6B
17	30	37	44	57	58	77	6C
18	31	38	45	58	59	78	6E
19	32	39	46	59	5A	79	6E
20	33	40	47	60	5B	80	6F

RHDCTTBL JCL and Execution

RHDCTTBL must be assembled and linked each time it is executed to create or update a device independence table. The JCL necessary to assemble, link, and execute the RHDCTTBL module is presented as follows:

Note: This is an SMP module. See the section on system modification in the CA IDMS installation guide for your operating system.

z/OS JCL

```
//            EXEC ASMFCL
//ASM.SYSLIB DD DSN=yourHLQ.CAGJMAC,DISP=SHR
//            DD DSN=sysl.maclib,DISP=SHR
//ASM.SYSIN  DD *
   rhdcttbl source statements
//LKED.SYSLMOD  DD DSN=idms.loadlib,DISP=SHR
//LKED.SYSIN    DD *
  INCLUDE SYSLMOD(RHDCTTBL)
  INCLUDE SYSLMOD(IDMSUTIO)
  INCLUDE SYSLMOD(IDMS)
  INCLUDE SYSLMOD(IDMSDATE)
ENTRY TTBLEP1
NAME RHDCTTBL(R)
// EXEC PGM=RHDCTTBL
//STEPLIB DD DSN=idms.custom.loadlib,DISP=SHR
//         DD DSN=idms.cagjload,DISP=SHR
//SYSLST  DD SYSOUT=A
//sysctl  DD DSN=idms.sysctl,DISP=SHR
```

yourHLQ.CAGJMAC	Data set name of CA IDMS macro library
sysl.maclib	Data set name of system macro library
idms.loadlib	Data set name of CA IDMS load library
sysctl	DDname of the SYSCTL file
idms.sysctl	Data set name of SYSCTL

z/VSE JCL

```
// OPTION CATAL
 PHASE RHDCTTBL,*
// EXEC ASMA90
    rhdcttbl source statements
/*
 INCLUDE IDMSUTIO
 INCLUDE IDMSDATE
 INCLUDE IDMS
 ENTRY TTBLEP1
// EXEC LNKEDT
/*
// UPSI b
// EXEC RHDCTTBL
```

b

Appropriate UPSI switch, 1-8 characters, if specified in the IDMSOPTI module

z/VM JCL

```
GLOBAL MACLIB IDMSLIB CMSLIB OSMACRO
FILEDEF TEXT DISK ttbl TEXT a3
ASSEMBLE usersrc (NODECK OBJECT PRINT NOTERM
TXTLIB DEL utextlib ttbl
TXTLIB ADD utextlib ttbl
FILEDEF SYSLST PRINTER
FILEDEF SYSLMOD DISK uloadlib LOADLIB a6
                             (RECFM V LRECL 1024 BLKSIZE 1024
FILEDEF objlib DISK utextlib TXTLIB a
FILEDEF objlib1 DISK IDMSLIB1 TXTLIB A2
GLOBAL LOADLIB uloadlib
LKED linkctl (LET LIST MAP XREF NCAL RENT NOTERM PRINT SIZE 512K 64K

linkage editor control statements (linkctl)

INCLUDE objlib (ttbl)
INCLUDE objlib1 (IDMSUTIO)
INCLUDE objlib1 (IDMS)
INCLUDE objlib1 (IDMSDATE)
ENTRY TTBLEP1
NAME RHDCTTBL(R)


FILEDEF CDMSLIB DISK IDMSLIB LOADLIB A6
GLOBAL LOADLIB IDMSLIB
OSRUN RHDCTTBL
```

ttbl TEXT a3	Filename, filetype, and filemode of the file that contains the generated assembled text
usersrc	Filename of the file containing user source code
utextlib	Filename of the user text library
uloadlib LOADLIB a6	Filename, filetype, and filemode of the user load library
objlib	DDname of the user object library
objlib1	DDname of the first CA IDMS/DB object library
linkctl	Filename of the file that contains the linkage editor control statements

Appendix C: User-Written Edit Modules

This appendix discusses about user-written edit modules.

This section contains the following topics:

Overview

Definition

A user-written edit module is a program module that can be used to supplement or replace automatic editing and error handling on mapin and/or mapout. A user-written edit module should be used only when it performs an operation that is not available through the automatic editing and error handling features that the CA IDMS mapping facility provides.

Steps

The following steps are involved in writing and using a user-written edit module:

1. Planning and coding the module
2. Preprocessing, compiling, and link editing the module
3. Specifying the module for use by the CA IDMS mapping facility

Specifying an Existing Module

An existing module is specified for use during runtime mapping by using either the online compiler or the batch compiler of the CA IDMS mapping facility:

- The *online mapping compiler* User-Defined Edit Module screen can be used to specify a user-written edit module, as explained in the documentation of that screen in "Batch Compiler Statements".
- The *batch compiler* MFLD (for MAP AUTOPANEL) and MFLD statements can be used to specify a user-written edit module, as explained in the documentation of either statement in “Batch Compiler Statements".

User-written edit modules must be written in Assembler. Preprocessing, compiling, and link editing of Assembler modules is detailed in the *CA IDMS DML Reference Guide for Assembler*. Planning and coding considerations that apply to all user-written edit modules are presented as follows, followed by considerations that apply specifically to either mapin or mapout operations.

Coding Considerations

User-written edit modules for the CA IDMS mapping facility are not CA IDMS/DC. exits; CA IDMS/DC controls the execution of the user-written edit module.

Since user-written edit modules are considered subroutines by Runtime mapping, they do not receive storage protection in the same manner as application programs. It is advisable to conform to the register usage demonstrated in this appendix when coding and testing user edit modules so that user edit modules do not modify system storage areas.

The following points, which are discussed separately, should be considered when planning a user-written edit module for mapin or mapout operations:

- Registers immediately prior to user edit-module execution
- System macros that must be included in user edit modules
- System DSECTS that are used by user edit modules

Registers Immediately Prior to User Edit Module

Registers 2-12

The online compiler saves the contents of registers two through twelve in a save area prior to calling a user-written edit module. The save area is configured as shown in the following table. The register values stored in the save area are passed back to the appropriate registers after the user-written edit module finishes executing.

Word	Contents
1	The return value from the #START macro (discussed later in this section); register 12 points to this word on entry to the user edit module
2	The value from register 11
3	The value from register 12 (not the register 12 value passed to the user edit module)
4	The value from register 2
5	The value from register 3
6	The value from register 4
7	The value from register 5
8	The value from register 6
9	The value from register 7
10	The value from register 8
11	The value from register 9
12	The value from register 10
13	The address of word 1 of this save area
14	The start of the 18-word save area for the user edit module; register 13 points to this word on entry to the edit module

Registers 1 and 13-15

Registers 1 and registers 13 through 15 are used for communication:

- **R1** contains the address of the parameter list passed by Runtime mapping to the user-written edit module.
- **R13** contains the address of the 18-word save area reserved for the user-written edit module to use.
- **R14** contains the address to return at end of processing.
- **R15** contains the address of the user-written edit module entry point.

System Macros

User-written edit modules should begin with the #START macro and exit with the #RTN macro. Modules that do not include these macros must include code that performs the equivalent functions.

The #START and #RTN macros are presented separately here.

#START Macro

The #START macro indicates the start of a routine. #START sets up addressability for the issuing program, using register 12 as the base register. #START must be the first instruction in a user-written edit module that uses CA IDMS calling conventions.

Syntax

```
►►── label─┬───────────────┬────────────────────────────►◄
           └─ INTernal ─┘
```

Label

Specifies a label for the entry point established by the #START macro. It is necessary to specify a label in a #START instruction for a user-written edit module.

INTERNAL

Prevents the #START macro from generating an entry point. INTERNAL *should not* be specified in a user-written edit module.

Sample Instruction

The following sample instruction establishes an entry point with the name XTEP1:

XTEP1 #START

Note: For more information about the #START macro and the MPMODE parameter, consult the *CA IDMS System Operations Guide*.

#RTN Macro

The #RTN macro terminates a routine and returns control to the calling routine. #RTN loads the return address from the TCE stack into register 14, adjusts register 13 to point to the top of the currently available position in the stack, and issues a BR 14 instruction. A sample of the code generated by #RTN is shown:

```
SH     R13,=H'4'
L      R13,0(,R13)
L      R14,0(,R13)
BR     R14
```

#RTN must be the last instruction executed in a user-written edit module that uses CA IDMS calling conventions.

Syntax

```
►►─── label ──────────────────────────────────────────────────►◄
```

Specification of a label is optional. The following sample instruction establishes the label RTRN1 for a #RTN macro:

```
RTRN1 #RTN
```

System DSECTs

The following table lists the system DSECTS used during execution of a user-written edit module and the copy book names used to name the DSECTS:

DSECT Module	Copy book name	Purpose
IBH	COPY #IBHDS	Provides the terminal input buffer header in the buffer chain
MCE	COPY #MCHDS	Provides the map control element for the map
MRB	COPY #MRBDS	Provides the map request block and map request element for the map
PTE	COPY #PTEDS, #PTXDS	Provides the physical terminal element

The macro library installed with CA IDMS contains all necessary DSECT definitions.

Note: When running your User-Written Edit Modules in SYSTEM MODE, R9 and R10 must point to the TCE and CSA at the time of any DC request. #CSADS must be copied in order to utilize it.

Note: For more information on DSECT definitions, see the CA ADS DSECT *Reference Guide*.

Input Modules for Mapin Operations

When performed on input, a user-written edit module determines how data is prepared prior to storage. The module receives data from a map field or from automatic editing. The following considerations affect the coding of input modules:

- Format of data received and output by input modules
- Parameters passed to input modules
- Macros available for input modules

Each of these topics is discussed separately, followed by an example of a user-written edit module for mapin operations.

Format of Data

A user-written edit module can be performed before, instead of, or after automatic editing. The point at which a user-written edit module is performed affects the format of data that the module receives and outputs.

Input Data

The user-written edit module must be prepared to receive data in the format that is appropriate to the point at which the module executes:

- *When executed before or instead of automatic editing*, a user-written edit module for mapin operations receives data in external format for the field.
- *When executed after automatic editing*, a user-written edit module for mapin operations receives data in internal format for the field.

Output Data

The user-written edit module must output data in the format that is appropriate to the point at which the module executes:

- *When executed before automatic editing*, a user-written edit module for mapin operations must output data in external format for the field. Runtime mapping repoints the address of the map field to the first byte, permitting automatic editing to edit the map field into internal format.
- *When executed instead of or after automatic editing*, a user-written edit module for mapin operations must output data in internal format for the field.

Parameters Passed to Input Modules

Runtime mapping loads the specified user-written edit module from the CA IDMS/DC. load library. The address words placed in the parameter list summarized in the following table are referenced from CA IDMS/DC off the address passed in register 1.

Address words 9 and 10 are passed in the parameter list for online maps only. They are not passed for file maps.

Address word 9 points to the first byte of input data for the field in the work area. If the user module is used in conjunction with automatic editing, this is the work area that must be referenced.

- If the user edit module is *executed before* automatic editing and the user edit module changes the input, modifications should be made to the data to which word 9 points This data will be passed to automatic editing.
- If the user edit module is *executed after* automatic editing, word 9 points to the data that is passed by automatic editing

If automatic editing is not used, the user edit module can access data for the field using either address word 4, which references the data in the data stream, or address word 9, which references the data after it is moved to the work area. (IBH DSECT)

Address Word	Data Element
1	Data field in target data record (output)
2	Header for the next input buffer in the buffer chain (IBH DSECT)
3	End of current input buffer
4	Start of data for the field in the input buffer (data stream)
5	Map Request Element for the field (MRE DSECT)
6	Map Control Element for the field (MCE DSECT)
7	Physical Terminal Element (PTE DSECT)
8	Map Request Block (MRB DSECT)
9	Start of data in the field, in the work area (start of input)
10	Last byte of data for the field, in the work area (end of input)

Note: Address words 9 and 10 are passed in the parameter list of online maps only.

Input data can be placed in a single input buffer or can overflow in a chain of noncontiguous input buffers. The length, in bytes, of the unedited input data is stored in the MREINLEN field in the Map Request Element (address word 5). The fourth address word points to the first byte of the input data in the current input buffer.

Macros for Input Modules

The following system macros are available for use in user-written input modules:

Macro	Marks a field ...
#SET MRETERR	In error by setting the internal MRETERR flag on for the field.
#SET MRECHNG	As changed by setting the internal MRECHNG flag on for the field. If automatic editing is not enabled, the MRECHNG flag will not be set; the user must supply code to set and later test the flag.
#SET MREERAS	As erased by setting the internal MREERAS flag on for the field.
#SET MRETRUN	As truncated by setting the internal MRETRUN flag on for the field.
#SET MRETDIF	As containing data that is different than the data in the record buffer.

Sample Input Module

What the Sample Input Module does

The following sample edit module verifies dates supplied by the terminal operator and, if necessary, strips out slashes to make the operator's input conform to *mmddyy* format. The sample edit module verifies that the date conforms to the following rules:

- The date must be numeric.
- The month must be from 01 to 12.
- If the month is 04, 06, 09, or 11, the date must not be greater than 30.
- If the month is 02, one of the following two rules must be true:
  - If a leap year, the day must not be greater than 29.
  - If not a leap year, the day must not be greater than 28.
- For all other months, the day must not be greater than 31.

If the Date is in Error

If the sample edit module determines that the date is valid, the date is transposed from *mmddyy* to *yymmdd* format for storage. If the date is in error, an error indicator is sent to the map. The erroneous date appears on the map as ?00000 if the map is redisplayed to the user for correction.

The following sample edit module is not reentrant.

The #MOPT macro that is included in this sample module generates register equates for use in coding the module, sets up a CSECT name for the module, and includes the name of the macro and its date/time stamp in future listings of the module. Use of the #MOPT macro is optional.

Edit modules can be either SYSTEM or USER MODE programs; the majority are USER MODE. To specify a user-written edit module mode, use the #MOPT macro. Set the ENV parameter to USER for the USER MODE program or to SYS or SYSTEM for SYSTEM MODE. If you do not set the ENV parameter, the defaults imply that it is set to USER.

If a DC request is issued, USER MODE programs need to be link edited with IDMSBALI or need to issue a #BALI macro within the user-written edit module. If a DC request is not issued in the program, there is no need to link edit the program with IDMSBALI or to issue a #BALI macro within the code.

You don't need to link edit SYSTEM MODE edit modules with IDMSBALI or have the program contact the #BALI macro. However, the CSA DSECT must be copied into your code if you issue any DC request. If you issue a DC request, R9 and R10 must point to the TCE and CSA at the time of the request.

```
         COPY  #CSADS
         COPY  #MRBDS
         #MOPT CSECT=CSYPDTE0,ENV=SYS
DTE0NTRY #START MPMODE=ANY
**********************************************************************
*  R1 POINTS TO AN TEN WORD PARAMETER LIST FOR MAPIN AS FOLLOWS:     *
*                                                                    *
*  WORD1   0(R1) ———▶ ADDRESS OF DATA FIELD IN TARGET DATA RECORD    *
*  WORD2   4(R1) ———▶ ADDRESS OF HEADER FOR NEXT INPUT BUFFER        *
*  WORD3   8(R1) ———▶ ADDRESS OF END OF CURRENT INPUT BUFFER         *
*  WORD4  12(R1) ———▶ ADDRESS OF START OF INPUT DATA FOR FIELD       *
*  WORD5  16(R1) ———▶ ADDRESS OF MRE                                 *
*  WORD6  20(R1) ———▶ ADDRESS OF MCE                                 *
*  WORD7  24(R1) ———▶ ADDRESS OF PTE                                 *
*  WORD8  28(R1) ———▶ ADDRESS OF MRB                                 *
*  WORD9  32(R1) ———▶ ADDRESS OF START OF DATA IN THE WORK AREA      *
*  WORD10 36(R1) ———▶ ADDRESS OF LAST BYTE OF DATA IN THE WORK AREA  *
**********************************************************************
         L     R2,0(R1)            GET ADDRESS OF DATA FIELD
         L     R3,32(R1)           GET ADDRESS OF INPUT FIELD
         L     R4,16(R1)           GET ADDRESS OF MAP REQUEST ELMT
         USING CSA,R10             BASES DC CSA STORAGE
         USING MRE,R4              BASES MAPPING MRE BLOCK
         LH    R6,MREINLEN         R6 <- LENGTH OF INPUT FIELD
CHKLEN8  LA    R5,8                R5 <- 8
         CR    R5,R6               LENGTH=8?
         BNE   CHKLEN6             NO, CHECK FOR LENGTH=6
         CLI   2(R3),C'/'          FORMAT OF XX/.....?
         BNE   SETERROR            NO, SET INPUT ERROR
         CLI   5(R3),C'/'          FORMAT OF XX/XX/..?
         BNE   SETERROR            NO, SET INPUT ERROR
         MVC   2(2,R3),3(R3)       MOVE DAYS OVER
         MVC   4(2,R3),6(R3)       MOVE YEAR OVER
         B     CHKNUMS             GO CHECK FOR NUMERIC CHARACTERS
CHKLEN6  LA    R5,6                R5 <- 6
         CR    R5,R6               LENGTH=6?
         BNE   SETERROR            NO, SET INPUT ERROR
CHKNUMS  LR    R7,R3               R7 —▶ FIRST CHARACTER OF DATE
         LA    R8,6(,R7)           R8 —▶ FIRST CHARACTER PAST DATE
CHKNLOOP CLI   0(R7),C'0'          CHARACTER LOWER THAN 'F0'?
         BL    SETERROR            YES, SET INPUT ERROR
         CLI   0(R7),C'9'          CHARACTER HIGHER THAN X'F9'?
         BH    SETERROR            YES, SET INPUT ERROR
         LA    R7,1(,R7)           INCREMENT CHARACTER POINTER
         CR    R7,R8               END OF DATE?
         BL    CHKNLOOP            NO, CHECK NEXT DATE CHARACTER
         CLC   0(6,R3),=C'000000'  IS DATE ZERO?
         BE    FLIPDATE            YES, MOVE DATE TO USER RECORD
         CLC   0(6,R3),=C'999999'  IS DATE ALL NINES?
         BE    FLIPDATE            YES, MOVE DATE TO USER RECORD
         CLC   0(2,R3),=C'12'      MONTH > 12?
         BH    SETERROR
         CLC   0(2,R3),=C'00'      MONTH = 0?
         BE    SETERROR
         CLC   2(2,R3),=C'00'      DAY = 0?
         BE    SETERROR
         CLC   4(2,R3),=C'00'      YEAR = 0?

         BE    SETERROR
         CLC   0(2,R3),=C'04'      APRIL?
         BE    CHK30DAY
         CLC   0(2,R3),=C'06'      JUNE?
         BE    CHK30DAY
         CLC   0(2,R3),=C'09'      SEPTEMBER?
         BE    CHK30DAY
         CLC   0(2,R3),=C'11'      NOVEMBER?
         BE    CHK30DAY
         CLC   0(2,R3),=C'02'      FEBRUARY?
         BE    CHKLEAP             YES, CHECK FOR LEAP YEAR
         CLC   2(2,R3),=C'31'      31 DAYS OR FEWER FOR OTHER MONTHS
         BH    SETERROR            IF BAD, SET INPUT ERROR
         B     FLIPDATE            MOVE DATE TO USER RECORD
CHK30DAY CLC   2(2,R3),=C'30'      30 DAYS OR FEWER FOR SOME MONTHS
         BH    SETERROR            IF BAD, SET INPUT ERROR
         B     FLIPDATE            MOVE DATE TO USER RECORD
CHKLEAP  #GETSTK =(2)              GET 2 WORDS OF STORAGE (R11 BASED)
         PACK  0(8,R11),4(2,R3)    PACK THE YEAR
         CVB   R9,0(,R11)          R9 <- CONVERTED BINARY YEAR
         SR    R8,R8               R8 <- ZERO
         LA    R15,4               R15 <- 4
         DR    R8,R15              DIVIDE YEAR BY 4
         LTR   R8,R8               ZERO REMAINDER?
         BZ    CHK29DAY            YES, CHECK FOR 29 OR FEWER DAYS
         CLC   2(2,R3),=C'28'      28 DAYS OR FEWER FOR NON-LEAP YEAR
         BH    SETERROR            IF BAD, SET INPUT ERROR
         B     FLIPDATE            MOVE DATE TO USER RECORD
CHK29DAY CLC   2(2,R3),=C'29'      29 DAYS OR FEWER FOR LEAP YEAR
         BH    SETERROR            IF BAD, SET INPUT ERROR
FLIPDATE MVC   0(2,R2),4(R3)       MOVE YEAR TO USER RECORD
         MVC   2(4,R2),0(R3)       MOVE MONTH AND DAY TO USER RECORD
         B     RETURN              RETURN
SETERROR #SET  MRETERR             INDICATE INPUT ERROR TO USER
RETURN   #RTN
         END   DTE0NTRY
```

Output Modules for Mapout Operations

When performed on output, a user-written edit module determines how data is prepared prior to mapout. The module receives data from program variable storage or from automatic editing. The following considerations affect the coding of output modules:

- Format of data received and output by output modules
- Parameters passed to output modules
- Macros available for output modules

Each of these topics is discussed separately, followed by an example of a user-written edit module for mapout operations.

Format of Data

A user-written edit module can be performed before, instead of, or after automatic editing. The point at which a user-written edit module is performed affects the format of data that is received by and output from the module.

Input Data

A user-written edit module for mapout operations must be prepared to receive data in the format that is appropriate to the point at which the module executes:

- *When executed before or instead of automatic editing*, a user-written edit module receives data in internal format for the field.
- *When executed after automatic editing*, a user-written edit module receives data in external format for the field. Data is only passed to the user edit module after the data has been edited and automatic editing criteria indicate that the data is correct.

Output Data

A user-written edit module for mapout operations must output data in the format that is appropriate to the point at which the module executes:

- *When executed before automatic editing*, a user-written edit module must output data in internal format for the field. Output data must be returned to program variable storage from which the data originally came (address word 1).
- *When executed instead of automatic editing*, a user-written edit module must output data in external format for the field. Output data must be placed in the target field in the output buffer (address word 2).
- *When executed after automatic editing*, a user-written edit module must output data in external format for the field (address word 2).

Parameters Passed to Output Modules

Runtime mapping loads a user-written edit module from the CA IDMS/DC. load library. The address words that are placed in the parameter list summarized in the following table are referenced from CA IDMS/DC off the address passed in register 1.

Address Word	Data Element
1	Data field in the application program data record
2	Target field in the output buffer for edited data
3	Map Request Element for the data field (MRE DSECT)
4	Map Control Element for the data field (MCE DSECT)
5	Physical Terminal Element (PTE DSECT)
6	Map Request Block (MRB DSECT)

The online compiler inserts an attribute byte in the first position of the output screen buffer.

If the user-written edit module is executed instead of or after automatic editing, register 1 must be set to point past the final character of the edited data.

Control returns to runtime mapping after the user edit module and automatic editing (if executed) finish editing the data. The value in register 1 is used to scan for and eliminate trailing blanks from the edited data if requested by either the BACKSCAN option of the online compiler Field Edit screen or the BACKSCAN clause of the batch mapping MFLD statement.

Macros for Output Modules

The system macro **#SET MRETERR** is available for use in user-written output modules. #SET MRETERR indicates that a field contains incorrect data by setting the internal MRETERR flag on for the field.

Sample Output Module

The following sample edit module transposes a 6-digit date from *yymmdd* to *mmddyy* format. The #MOPT macro included in this sample module generates register equates for use in coding the module, sets up a CSECT name for the module, and includes the name of the macro and its date/time stamp in future listings of the module. Use of the #MOPT macro is optional.

Edit modules can be either SYSTEM or USER MODE programs; the majority are USER MODE. To specify a user-written edit module mode, use the #MOPT macro. Set the ENV parameter to USER for the USER MODE program or to SYS or SYSTEM for SYSTEM MODE. If you do not set the ENV parameter, the defaults imply that it is set to USER.

If a DC request is issued, USER MODE programs need to be link edited with IDMSBALI or need to issue a #BALI macro within the user-written edit module. If a DC request is not issued in the program, there is no need to link edit the program with IDMSBALI or to issue a #BALI macro within the code.

You don't need to link edit SYSTEM MODE edit modules with IDMSBALI or have the program contact the #BALI macro. However, the CSA DSECT must be copied into your code if you issue any DC request. If you issue a DC request, R9 and R10 must point to the TCE and CSA at the time of the request.

Note: It is recommended that this module be performed before automaticediting so that an external picture can be used to insert slashes (/) into the date to make it *mm/dd/yy*.

```
         COPY  #CSADS
         COPY  #MRBDS
         #MOPT CSECT=CSYPDTE2,ENV=SYS
DTE2NTRY #START MPMODE=ANY
*********************************************************************
*  R1 POINTS TO A SIX WORD PARAMETER LIST FOR MAPOUT AS FOLLOWS:    *
*                                                                   *
*  WORD1   0(R1) ——► ADDRESS OF DATA IN USER'S RECORD BUFFER        *
*  WORD2   4(R1) ——► ADDRESS OF NEXT POSITION IN OUTPUT BUFFER      *
*  WORD3   8(R1) ——► ADDRESS OF MRE                                 *
*  WORD4  12(R1) ——► ADDRESS OF MCE                                 *
*  WORD5  16(R1) ——► ADDRESS OF PTE                                 *
*  WORD6  20(R1) ——► ADDRESS OF MRB                                 *
*********************************************************************
         L     R2,0(R1)              GET ADDRESS OF DATA FIELD
         L     R3,4(R1)              GET ADDRESS OF NEXT OUTBUF POS
         L     R4,8(R1)              GET ADDRESS OF MRE
         USING MRE,R4
         USING CSA,R10
         #TEST MRETERR,OFF=FLIPDATE  ARE WE IN AN ERROR CYCLE ?
         MVC   0(6,R3),=C'?00000'    MOVE ERROR FIELD TO BUFFER
         B     RETURN
FLIPDATE MVC   0(2,R3),2(R2)         MOVE MONTH
         MVC   2(2,R3),4(R2)         MOVE DAY
         MVC   4(2,R3),0(R2)         MOVE YEAR
         MVC   0(6,R2),0(R3)         MOVE REVERSED FIELD INTO BUFFER
RETURN   LA    R1,6(R3)              INCREMENT USER BUFFER POINTER
         #RTN
         END   DTE2NTRY
```

Appendix D: Generating Edit and Code Tables

This appendix describes about generating edit and code tables.

This section contains the following topics:

How to Define Tables

Edit and code tables used by automatic editing are defined in the data dictionary through IDD DDDL statements:

- *Stand-alone* edit and code tables are defined by the TABLE statement. Stand-alone tables can be specified for use with any record element at map-definition time. Stand-alone tables are also called loosely coupled tables.
- *Built-in* edit and code tables are defined by substatements of the RECORD statement. Built-in tables belong to and can be used only for the record with which they are defined. Built-in tables are also called tightly coupled tables.

Stand-alone and built-in tables are presented as follows.

Note: Edit and code tables can only be associated with a group element if the group is made up of DISPLAY elements.

Stand-Alone Tables

Overview

Stand-alone tables can be associated with any record element at map-definition time. Stand-alone tables are typically used to list values when:

- The values are subject to change
- The values can be used by several record elements

A stand-alone table is defined and generated by the TABLE statement of DDDL. Clauses in the TABLE statement determine the search technique, arrangement of values, and the type of values for a table as follows.

Search Technique

The *search technique used for the table* is determined by the SEARCH IS LINEAR/BINARY clause, as follows:

Clause	Description
SEARCH IS LINEAR	Specifies that a linear search algorithm is used for the table. The following considerations apply to linear searches: ■ A linear search progresses sequentially through the table; the first value in the table is examined first, the second value second, and so forth, until either the target value is found or the end of the table is reached. ■ The arrangement of values in the table, as established by the SORTED/UNSORTED parameter discussed later in this section, determines the order of values in the table and, thus, the order of the search.

Clause	Description
SEARCH IS BINARY	Specifies that the table is searched with a binary search algorithm. The following considerations apply to binary searches: ■ A binary search compares the target value against the table's midpoint value and determines which half of the table contains the target value. The selected portion then is halved in the same way; this process is repeated until either the target value is found or the end of the table is reached. ■ When SEARCH IS BINARY is specified, values in the table are kept in sorted order, regardless of the SORTED/UNSORTED specification for the table. The SORTED/UNSORTED parameter is discussed later in this section. ■ Binary searching cannot be performed on an edit table that contains ranges, since binary searching requires all values in the table are the same length. ■ A binary search in a code table can be specified for either an encoded value or a decoded value (as specified in the ON ENCODE/DECODE parameter of the SEARCH IS BINARY clause), but not for both. If binary searching is performed for encode values, decode values are searched linearly; if binary searching is performed for decode values, encode values are searched linearly.

Arrangement of Values in the Table

The arrangement of values in the table is determined by the TABLE IS SORTED/UNSORTED clause, as follows:

Clause	Description
TABLE IS SORTED	Specifies that values in the table are sorted in ascending order according to the EBCDIC collating sequence. The following considerations apply: ■ An *edit table* of ranges is sorted according to the lowest value in the range ■ A *code table* is sorted according to its encoded values
TABLE IS UNSORTED	Specifies that values in the table are not sorted; the table is stored in the order of its appearance in the defining TABLE statement.

Type of Values in the Table

The type of values in the table is specified by the TABLE DATA IS NUMERIC/ALPHANUMERIC clause. This specification affects the results of a table search. For example, the validity of the value 20b (where the b character denotes the blank character) depends on the type of values specified for the table:

- The entry is valid if the edit table is an ALPHANUMERIC table of valid values (20b falls alphabetically in the range 100 through 200)
- The entry is invalid if the edit table is a NUMERIC table of valid values (20 falls numerically outside the range 100 through 200)

Where are Load Modules Stored?

Load modules for stand-alone tables are stored in the DDLDCLOD area of the data dictionary.

Note: For more information about DDDL syntax and syntax rules, see the *CA IDMS IDD DDDL Reference Guide*.

Linked vs Unlinked

The map developer specifies whether a stand-alone table is linked or unlinked when enabling the table:

- A **linked table** is included in the map load module with which it is associated. Map load modules that use a linked table must be regenerated when changes are made to the table.
- An **unlinked table** is loaded at runtime by the map load module with which it is associated. It is unnecessary to regenerate map load modules that use an unlinked table when the table is changed.

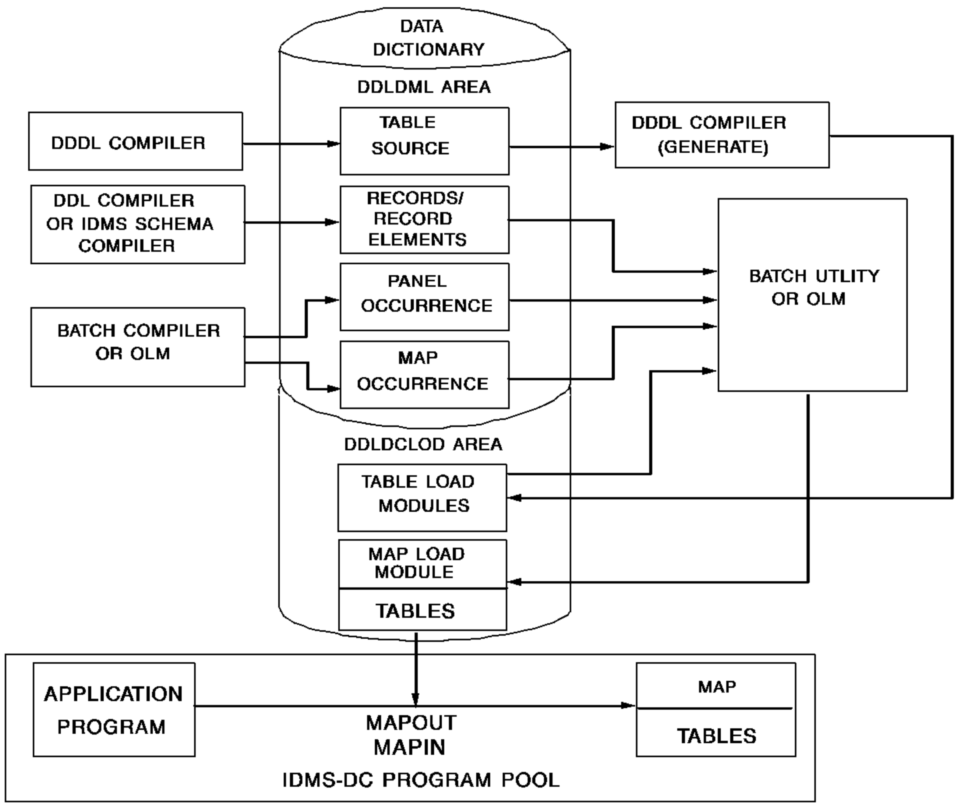
It is often preferable to enable stand-alone tables as unlinked tables since stand-alone tables are typically used as general-purpose tables for several record elements.

Compiling, Generating, Loading of maps

Linked Stand-alone Tables

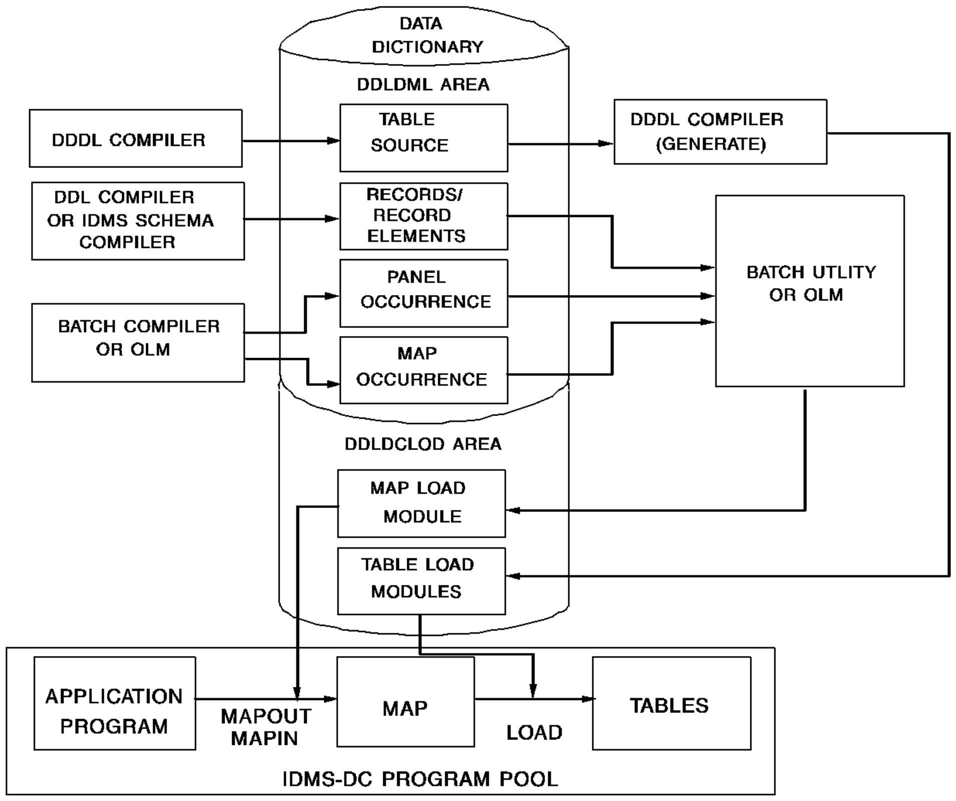
The compilation and runtime loading of a map that uses linked stand-alone tables are illustrated in the following drawing.

A linked stand-alone table becomes part of a map load module that uses it; the map load module must be recompiled if a linked table is changed.


DATA DICTIONARY
DDLDML AREA
DDDL COMPILER
TABLE SOURCE
DDDL COMPILER (GENERATE)
DDL COMPILER OR IDMS SCHEMA COMPILER
RECORDS/ RECORD ELEMENTS
PANEL OCCURRENCE
BATCH UTLITY OR OLM
BATCH COMPILER OR OLM
MAP OCCURRENCE
DDLDCLOD AREA
TABLE LOAD MODULES
MAP LOAD MODULE
TABLES
APPLICATION PROGRAM
MAPOUT
MAPIN
MAP
TABLES
IDMS-DC PROGRAM POOL

Unlinked Stand-alone Tables

An unlinked stand-alone table is used by a map, but is not part of the map load module; changes to an unlinked table do not affect map load modules that use the table.



Examples

The following examples demonstrate the use of DDDL statements to define and generate stand-alone tables.

Example 1

The following sample TABLE statement adds a stand-alone edit table of valid values to the dictionary; the search technique is linear and the table is unsorted:

```
ADD TABLE DEPTEDIT
TYPE IS EDIT VALID
SEARCH IS LINEAR
TABLE IS UNSORTED
VALUES ARE ( SHIPPING PERSONNEL ACCOUNTING
     MARKETING 'OFFICE SERVICES' )
GENERATE
          .
```

Example 2

The following sample TABLE statement adds a stand-alone edit table of valid values to the dictionary; the search technique is binary:

```
ADD TABLE NAMEEDIT
TYPE IS EDIT VALID
SEARCH IS BINARY
VALUES ARE
 ( ADAMS
   AGASSIZ
   BACH
     .
     .
     .

   XERXES
   YEATS
   ZENO )
GENERATE
 .
```

Example 3

The following sample TABLE statement adds a stand-alone code table to the dictionary; the search technique is linear and the table is unsorted:

```
ADD TABLE DEPTCODE
TYPE IS CODE
SEARCH IS LINEAR
TABLE IS UNSORTED
VALUES ARE
( 01          SHIPPING
  02          PERSONNEL
  03          ACCOUNTING
  04          MARKETING
  05         'OFFICE SERVICES'
  00          NOT FOUND
  NOT FOUND   MISSING  )
GENERATE
 .
```

Use of the NOT FOUND Condition

The following examples illustrate the use of the NOT FOUND condition in the value list of a code table.

Example 4a

When NOT FOUND (a condition to be acted upon) is used in the encode column of a code table, the following occurs:

```
VALUES ARE
( 100         MATHEMATICS
  200         ENGLISH
  300         SCIENCE
  NOT FOUND   INVALID-DEPT-NO  )
```

- On mapout, NOT FOUND is used as a catchall. At mapout, any value other than 100, 200, or 300 will match the NOT FOUND condition in the table. The corresponding value, INVALID-DEPT-NO will be moved to the map field.
- On mapin, NOT FOUND produces automatic editing errors under the following conditions:
  - If the value entered does not match a decoded value and if there is no catchall value.
  - If the value entered in the map field is INVALID-DEPT-NO the corresponding value is NOT FOUND instead of real value. As a result, the table is re-searched but no match is found.

Example 4b

When NOT FOUND is used in the decode column of the Code Table's value list, the following processing occurs:

```
VALUES ARE
( 100         MATHEMATICS
  200         ENGLISH
  300         SCIENCE
  000         NOT FOUND   )
```

- NOT FOUND is used as a catchall on map-ins. Any value entered in the map field, other than Mathematics, English, or Science will match the NOT FOUND condition and its corresponding value, 000, will be moved to the buffer.
- On mapout, NOT FOUND causes a program to abort as described:
  - If a value in the buffer does not match an encoded value or a catchall, the user program will abort with the message, MAPPING DATA ERROR
  - If 000 is the value in the buffer, its corresponding value is NOT FOUND. The encode values are then re-searched looking for a NOT FOUND condition. When there is no match, the application aborts.

Example 4c

When NOT FOUND is used in both the encode and decode column of a code table's value list as shown, the following processing occurs:

```
VALUES ARE
( 100         MATHEMATICS
  200         ENGLISH
  300         SCIENCE
  000         NOT FOUND
  NOT FOUND   INVALID-DEPT-NO  )
```

- On mapout and mapin, the NOT FOUND condition is used as a catchall:
  - Any value other than ENGLISH, MATHEMATICS, or SCIENCE, that is entered in the map field will match the NOT FOUND condition in the decoded column and its corresponding value, 000, will be moved to the buffer on mapin.
  - On mapin, if an invalid department number is entered in the map field, a match is found in the decode column and its corresponding value, NOT FOUND, is moved to the buffer. The decode column is then re-searched looking for NOT FOUND. When it is found, its corresponding value, 000, is moved to the map field.
  - Any value other than 100, 200, 300, or 000 will match the NOT FOUND condition in the encode column, and its corresponding value, INVALID-DEPT-NO, is moved to the buffer.
  - On mapout, if 000 is in the buffer, there is a match in the encode column and its corresponding value is NOT FOUND. The encode column is then re-searched looking for NOT FOUND. A match is found and its corresponding value, INVALID-DEPT-NO, is moved to the map field.

Built-In tables

Overview

A built-in table belongs to the record element with which it is generated and can only be used by that element. Built-in tables typically are used to list values in the following cases:

- The values are unlikely to change before the record element needs to be modified
- Few values are included in the table
- The values are specific to a particular element; the table is unlikely to be needed for another element

The RECORD ELEMENT or COBOL substatement of the DDDL RECORD statement is used to add or replace an element definition in a record. When either substatement is executed, edit and/or code tables (if any) defined in the substatement are generated as built-in tables for the added or replaced element. Either of the following clauses in a RECORD ELEMENT or COBOL substatement establishes a built-in table for an element:

- EDIT TABLE
- CODE TABLE

Note: For more information about DDDL syntax and syntax rules, see the *CA IDMS IDD DDDL Reference Guide*.

Rules for Built-in Tables

The following rules apply to built-in tables:

- Built-in tables are always searched in a linear fashion.
- The values in the table are maintained in unsorted order.

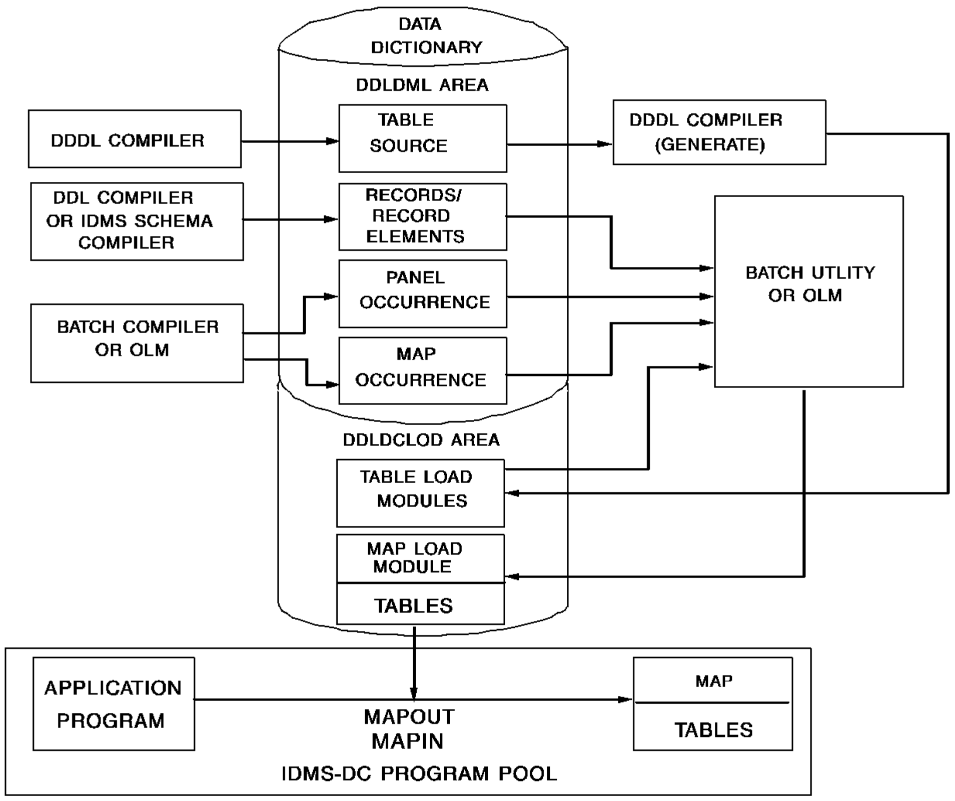
A built-in table is part of each map load module that is associated with the element that contains the table. A map load module must be regenerated if it is to reflect changes made to a constituent built-in table.

Compiling, Generating, and Loading

Built-in Tables

The compilation, generation, and runtime loading of a map that uses built-in tables are illustrated in the following figure.

A built-in table is part of a map load module that uses it; the map load module must be recompiled if a constituent built-in table is changed.



Examples

The following examples demonstrate the use of DDDL statements to define and generate built-in tables.

Example 1

The following sample RECORD ELEMENT statement adds the DAY-EL element to the DATE-REC record and defines a built-in edit table of valid values for the element:

```
ADD RECORD DATE-REC
    .
    RECORD ELEMENT DAY-EL
    EDIT VALID TABLE IS ( SUNDAY MONDAY TUESDAY
         WEDNESDAY THURSDAY FRIDAY SATURDAY )
    .
```

Example 2

The following sample RECORD ELEMENT statement adds MONTH-EL element to the DATE-REC record and defines a built-in code table for the element:

```
ADD RECORD DATE-REC
    .
    RECORD ELEMENT MONTH-EL
    CODE TABLE IS
       ( 01 JANUARY
         02 FEBRUARY
         03 MARCH
         04 APRIL
         05 MAY
         06 JUNE
         07 JULY
         08 AUGUST
         09 SEPTEMBER
         10 OCTOBER
         11 NOVEMBER
         12 DECEMBER )
    .
```

Appendix E: Estimating Pageable Map Storage

This appendix describes about estimating pageable map storage.

This section contains the following topics:

Definition

A pageable map can map out an unlimited number of variable fields at runtime. A runtime pageable map often provides more than one page of information. The terminal operator can page through the map during a paging session to view information that does not fit on the current map page.

Note: For more information on pageable maps, see the chapter "Pageable Maps”.

The amount of storage available during a runtime paging session is specified at system generation. Storage is allocated dynamically at runtime, as needed. This appendix presents a method for determining the amount of storage to specify at system generation. It is advisable to apply this method to several actual or intended pageable maps, and to specify the largest amount of storage in the system generation program.

Calculations Used

The amount of storage to reserve for a paging session is determined by multiplying the following values:

- The amount of storage per map page
- The maximum number of pages per pageable map

The methods for determining these values are presented separately.

Note: The value obtained for paging session storage must be rounded to the nearest integer when specified in the OLM statement at system generation.

Estimating the Amount of Storage per Map Page

Overview of Calculations

The amount of storage, in bytes, for a runtime map page is calculated by multiplying the following values:

- The amount of storage per detail occurrence
- The number of detail occurrences that can be mapped out in the detail area for the map at runtime

The methods for determining each of the previous values are presented separately here.

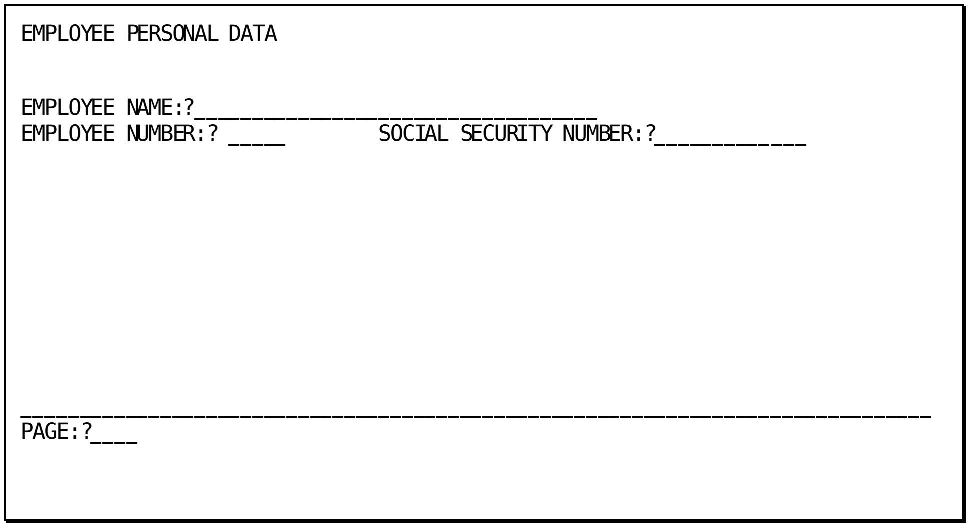
Amount of Storage per Detail Occurrence

The amount of storage per detail occurrence, in bytes, is calculated by adding the following values:

- The length (in bytes) of all variable fields in the detail occurrence
- The number of variable fields in the detail occurrence multiplied by 40 (40 is the number of bytes of overhead for each variable field)
- The number of bytes of overhead for the detail occurrence (this value is always 28)

Example

The fields on the following screen were used as the basis for the calculations.



```
EMPLOYEE PERSONAL DATA


EMPLOYEE NAME:?__________________________________
EMPLOYEE NUMBER:? _____        SOCIAL SECURITY NUMBER:?_____________











_____________________________________________________________________________
PAGE:?____
```

The following calculations were used to estimate the amount of storage required for the screen:

```
Length of all variable fields ......... . 38

Number of variable fields (x40).........120

Overhead............................. .. 28

              ____________________________________________

   Storage per detail occurrence.........186
```

Number of Detail Occurrences per Detail Area

The number of detail occurrences that can be mapped out in the detail area for the map at runtime is determined by:

1. Dividing the number of screen lines available for the detail area at runtime by the number of screen lines in the detail occurrence defined for the detail area.
2. Rounding the resulting value down to the next integer if an integer value is not obtained in the division operation.

Determining the Number of Lines Available at Runtime

The number of screen lines available for the detail area at runtime depends on the number of lines that are available on a given terminal screen. To find the number of lines available for the detail area at runtime, perform the following steps:

1. Add the number of lines reserved for the header area to the number of lines for the footer area.
2. Subtract the sum from the number of screen lines on the terminal.

Example

The fields on the following screen were used as the basis for our calculations.

```
EMPLOYEE PERSONAL DATA


EMPLOYEE NAME:?______________________________
EMPLOYEE NUMBER:? _____        SOCIAL SECURITY NUMBER:?_____________








____________________________________________________________________________
PAGE:?_____
```

How Many Detail Occurrences in a Detail Area?

The following calculations were used to estimate how many detail occurrences can be mapped out to the detail area at runtime for the map above:

```
          runtime screen lines..............10

          Lines in detail occurrence........  2

    ______________________________________________

          Times detail occurrence mapped out...5
```

The number of screen lines at runtime and the number of lines in the detail occurrence determine the number of detail occurrences per detail area. Multiplying this value with the integer amount of storage per detail occurrence produces the amount of storage needed for a map page.

Determining the Number of Pages per Pageable Map

The maximum number of pages that a pageable map can have in a paging session is calculated by multiplying the following runtime values:

- *The maximum number of times the detail occurrence can be repeated for the entire map*, as determined by the data the pageable map is to retrieve.

  For example, a pageable map is designed to display the names and identification numbers of all employees that belong to the department selected by the terminal operator at runtime. The maximum number of times the detail occurrence can be repeated for the map is determined by the number of employees in the largest available department.

- *The number of times the detail occurrence can be mapped out in the detail area for the map*, as calculated from the following values:

  - The number of screen lines available for the detail area at runtime
  - The number of screen lines in the detail occurrence

  The method for determining the number of times the detail occurrence can be mapped out in the detail area is presented in Estimating the Amount of Storage per Map Page earlier in this section.

Important: It is advisable to plan for a reasonable amount of short-term growth when estimating the number of pages per pageable map.

Appendix F: Alternative Maps

This appendix describes about alterative maps.

This section contains the following topics:

Overview

Alternative maps can be used in any application in which a dialog or program shows different copies of the same map to different users. The use of alternative maps displays an alternative copy of a map to a user if information in the user's signon record indicates that an alternative copy of maps should be displayed.

Each of the following methods for preparing alternative maps is discussed as follows:

- *Alternative copies of maps* are generated by using the CA IDMS mapping facility.
- *Tables of alternative maps* are generated and assigned to user signon records at system generation.

Generating Alternative Maps

When you copy an existing map to make an alternative map, only noncritical changes can be made.

Important: The following are considered critical changes:

- Adding a variable field
- Deleting a variable field
- Changing a nonpageable map to a pageable map, and vice versa
- Changing record versions
- Moving fields around on a map

Steps

The developer can use the online compiler to generate similar copies of a map by following these steps:

1. *Access the compiler*
2. Access *the Main Menu screen*:
   - Specify the pertinent information about the map being created, such as the name and version number of the map and dictionary
   - Select the *ADD option from the action bar* at the top of the screen.
   - Specify the name and version number (if necessary) of the map to be copied
   - Indicate whether all record information should be copied, or just the map layout
3. *On the Layout screen*, perform any of the following optional steps:
   - Change literal fields as necessary
   - Use the select-field character (default is the percent sign (%)) to select fields (if any) to which noncritical specifications are made, such as those listed:
     - An attribute specification, such as DARK (for fields that should not be shown to certain users) or PROTECTED (for fields that should not be altered)
     - The name of an edit or code table (particularly useful in a multilingual environment)
4. *On the Field Definition screens,* make noncritical changes to fields as appropriate
5. Return to the *Main Menu screen* and select the *Compile option from the action bar* at the top of the screen to generate data dictionary occurrences

Important: The version numbers of the original map and the copy must be the same.

Generating and Assigning Alternate Map Tables

Alternative map tables are generated and assigned to particular user signon records at system generation. Alternative maps are not shown to users unless the necessary information is defined in the system generation program.

The generation of alternative map tables and the assignment of alternative map tables to users are presented separately as follows.

Generating Map Tables

The system generation MAPTYPE statement is used to generate an alternative map table as described:

- *A unique maptype name* is specified in the MAPTYPE statement. The maptype name identifies an alternative map table. For example, the SPANISH maptype name could be used to identify a table of Spanish-language alternative maps.
- *A table of corresponding map copies* is built for the table by using clauses of the MAPTYPE statement. Each MAP *map-name-1* MAPS TO *map-name-2* clause in the MAPTYPE statement associates an original map with an appropriate copy.

Sample Alternative Map Table

```
ADD MAPTYPE SPANISH
    MAP ENGMAP01 MAPS TO SPNMAP01
    MAP ENGMAP02 MAPS TO SPNMAP02
    MAP ENGMAP03 MAPS TO SPNMAP03
    MAP ENGMAP04 MAPS TO SPNMAP04.
```

In the previous example, map SPNMAP01 is available for display to a Spanish-speaking operator when the operator runs a dialog or program that names the ENGMAP01 map.

The use of Wildcards

A generic case can be established in a MAPTYPE statement when the names of alternative maps are related in a consistent and predictable manner. A single MAP *map-name-1* MAPS TO *map-name-2* clause can be used to construct an alternative map table when a generic case can be named by the clause. The question mark (?) is used as a mask character when specifying generic map names in a MAPTYPE statement.

For example, the sample SPANISH alternative map table that is constructed in the previous example can also be constructed by the following sample statement:

```
ADD MAPTYPE SPANISH
    MAP ENGMAP?? MAPS TO SPNMAP??.
```

The sample SPANISH alternative map table built by the previous statement contains the names of all maps that meet the following criteria:

- The *name of the original copy* begins with the characters ENGMAP.
- The *name of the alternative copy* is configured as follows:
  - The name begins with the characters SPNMAP.
  - The name ends with the same final characters that end the name of the corresponding original map.
- The *date/time stamp* is the same for both copies.

For example, map ENGMAP04 would be added to the SPANISH alternative map table and associated with map SPNMAP04 by the sample MAPTYPE statement. Either of the following cases would cause ENGMAP04 and its alternative copy to not be added to the SPANISH alternative map table:

- The map named SPNMAP04 has a different date/time stamp than ENGMAP04.
- The alternative copy has been given a name that does not conform to the generic case.

Note: For more information about the MAPTYPE statement, see the *CA IDMS System Generation Guide*.

Assigning Map Tables to Users

A maptype can be assigned to a user at every signon by adding a MAPTYPE command to the user profile. For example:

- In the Sysgen Compiler, enter:

  ```
  ADD MAPTYPE SPANISH
      MAP ENGMAP?? MAPS TO SPNMAP??.
  ```

- In OCF, enter:

  ```
  CREATE USER PROFILE LMG01_PROFILE
      ATTRIBUTE
         MAPTYPE = SPANISH;

  CREATE USER LMG01
      PROFILE LMG01_PROFILE ... ;
  ```

Note: For more information about USER PROFILES, see the *CA IDMS Security Administration Guide*.

Appendix G: PL/I DML Statements for Pageable Maps

This appendix discusses about PL/I DML statements for pageable maps.

This section contains the following topics:

Overview

Pageable maps are defined by using the CA IDMS UCF mapping facility. A CA ADS dialog or program that uses a pageable map must include statements to handle storage and display of fields on the pageable map at runtime. This appendix presents the PL/I DML statements that enable a program to use pageable maps.

Notes:

- For more information about the definition and use of pageable maps, see the chapter "Pageable Maps."
- For more information about CA ADS statements for pageable maps, see the *CA ADS Reference Guide*.
- For more information about COBOL DML statements, see the *CA IDMS DML Reference Guide for COBOL* and for Assembler DML statements, see the *CA IDMS DML Reference Guide for Assembler*.

PL/I DML Statements

PL/I DML statements used for pageable maps are listed in the following table and described separately on the following pages:

Clause	Use
DECLARE MAP	Specifies that mapping mode terminal I/O is being used and names the map used in the program.
MAP IN	Requests a transfer of modified pageable map data to program storage by specifying either the DETAIL or HEADER clause.

Clause	Use
MAP OUT	Creates detail occurrences for a pageable map and/or requests display of map pages when the DETAIL or RESUME clause is specified.
STARTPAGE SESSION	Begins a map-paging session and specifies options for the session.
ENDPAGE SESSION	Terminates a map paging session.

Note: For more information about PL/I, see the *CA IDMS DML Reference Guide for PL/I*.

More information:

Pageable Maps (see page 91)

DECLARE MAP

A DECLARE MAP statement must be included in a PL/I program to name each map used by the program.

Syntax

```
►►─── DECLARE (map-name MAP ─┬──────────────────────────┬─ ) ──────────────►
                             └ VERSION version-number ──┘

►─┬───────────────────────────────────────────────────────┬─ ; ──────────►◄
  └ TYPE ( ─┬─ STANDARD ◄ ─┬─ ) ─┬────────────┬─┘
            └─ EXTENDED ───┘     └─ PAGING ───┘
```

Map-name MAP

Specifies the name of a predefined map to be used by the program. *Map-name* must be a 1- to 8-character name of an existing map load module.

VERSION 1/version-n

Optionally identifies the version of the named map. *Version-n* is a numeric constant of the map version desired. The default is 1.

TYPE STANDARD/EXTENDED

Specifies the attributes of the named map:

STANDARD (default)

Indicates that map attributes are those available on standard 3270-type terminals.

EXTENDED

Indicates that map attributes include those available on 3279-type terminals. Mapping features such as color, blinking fields, and reverse video can be used for applications running under 3279-type terminals.

PAGING

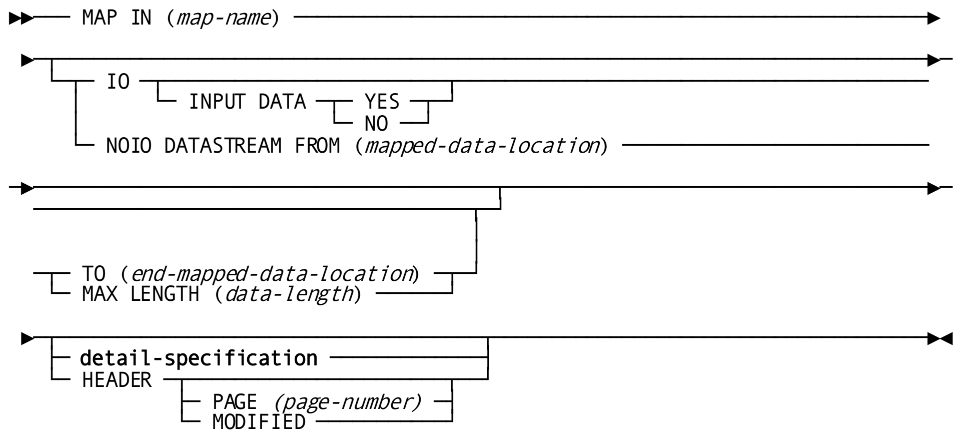
Specifies that the named map is a pageable map.

MAP IN

The MAP IN statement requests a transfer of data to program storage. After completion of a MAP IN function, the ERROR-STATUS field of the IDMS-DC communications block indicates the outcome of a pageable-map operation:

Message	Problem
4664	The requested node for a header or detail was either not present or not updated.
4668	No more modified detail occurrences require mapin.
4672	The scratch record containing the requested detail could not be accessed (internal error).

Syntax


MAP IN (*map-name*)
IO
INPUT DATA
YES
NO
NOIO DATASTREAM FROM (*mapped-data-location*)
TO (*end-mapped-data-location*)
MAX LENGTH (*data-length*)
detail-specification
HEADER
PAGE (*page-number*)
MODIFIED

```
►►─┬─────────────────────────────────────────────────────────────────────┬─
   └─ DETAIL ─┬─ NEXT ◄ ──────────────────────────────────────────────────┤
              ├─ FIRST ─┬──────────────────────────────────────────────┤
              │         └─ RETURNKEY (data-field) ─┘                    │
              ├─ KEY (key-name) ────────────────────────────────────────┤
              ├─ SEQUENCE_NUMBER (sequence-field) ─┬────────────────────┤
              │                                    └ RETURNKEY (data-field)─┘
              └─ RETURNKEY (data-field) ────────────────────────────────┘

─►─┬─────────────────────────────────┬──────────────────────────────────►◄
   └─┬───────────────────────┬─┬────────────┬─┘
     └─ PAGE (page-number) ─┘ └─ MODIFIED ─┘
```

DETAIL

Specifies that the MAP IN operation is to retrieve data from a modified detail occurrence (MDT set on). The contents of all data fields in the detail occurrence are retrieved unless MODIFIED is specified for the MAP IN DETAIL statement; MODIFIED retrieves only modified fields.

The retrieved detail occurrence is specified by one of the following clauses:

NEXT (default)

Retrieves the next sequential modified detail occurrence. An end-of-data condition is returned in either of the following cases:

- No detail occurrences have been modified
- All modified detail occurrences have been mapped in already

FIRST RETURNKEY IS *data-field-name-v*

Retrieves the first available modified detail occurrence.

RETURNKEY IS *data-field-name-v* optionally specifies the name of a variable field in which to store the 4-byte key (if any) associated with the retrieved detail occurrence. If no value is associated with the detail occurrence, *data-field-name-v* is set to 0. *Data-field-name-v* must be a 4-byte value, but does not have to be a binary fullword.

Note: A value is associated with a detail occurrence by using the KEY IS parameter in a MAP OUT DETAIL command for that occurrence.

An end-of-data condition results if all modified detail occurrences have been retrieved already.

KEY IS *key-v*

Specifies the modified detail occurrence to retrieve based on the value associated with the detail occurrence. (A value is associated with a detail occurrence by using the KEY IS parameter in the MAP OUT DETAIL command for that occurrence.) *Key-v* is the name of a 4-byte field.

A detail-not-found condition is returned in either of the following cases:

- The specified occurrence is not a modified detail occurrence
- No detail occurrence with the specified value is found

SEQUENCE_NUMBER IS *data-field-name-v* RETURNKEY IS *data-field-name*

Specifies the sequence number of the retrieved data occurrence. Detail occurrences are built by the application program, and are stored in the sequence in which they are created. *Data-field-name* is a binary fullword.

RETURNKEY IS *data-field-name-v* optionally specifies the name of a variable field to store the 4-byte value (if any) associated with the retrieved detail occurrence. If no value is associated with the detail occurrence, *data-field-name-v* is set to zero. *Data-field-name-v* must be a 4-byte value, but does not have to be a binary fullword.

A detail-not-found condition is returned in either of the following cases:

- The specified occurrence is not a modified detail occurrence.
- No detail occurrence with the specified value is found.
- **RETURNKEY IS** *data-field-name* performs the same operation as the NEXT clause (described previously) and specifies the name of a variable field to store the 4-byte value (if any) associated with the retrieved detail occurrence. (A value is associated with a detail occurrence by using the KEY IS parameter in a MAP OUT DETAIL command for that occurrence.) If no value is associated with the detail occurrence, *data-field-name-v* is set to 0. *Data-field-name-v* must be a 4-byte value, but does not have to be a binary fullword.

HEADER

Specifies that the MAP IN operation is to retrieve the contents of data fields in the header and footer areas. The contents of all data fields in the header and footer areas are retrieved unless MODIFIED is specified for the MAP IN HEADER statement; MODIFIED retrieves only modified fields.

MODIFIED

Specifies that only modified fields (MDT set on) are retrieved in the MAP IN operation.

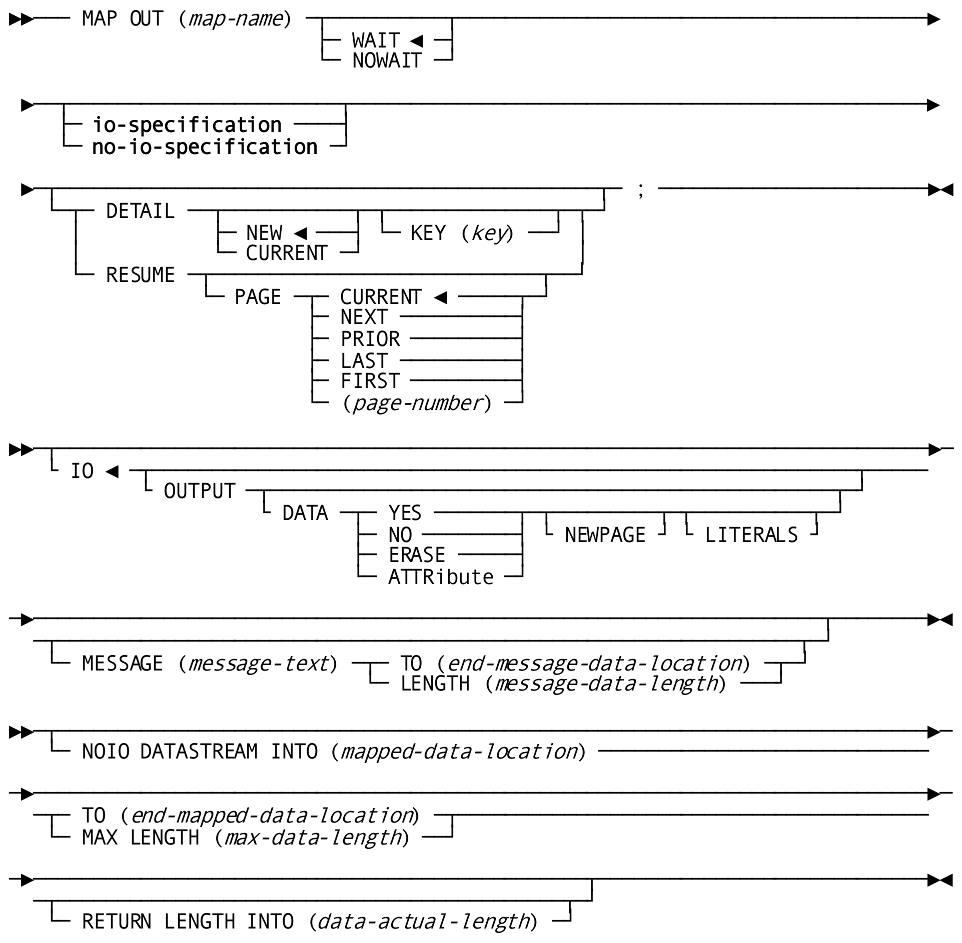
MAP OUT

The MAP OUT statement is used to create or modify detail occurrences for a pageable map or to request that a map page be transmitted to the terminal screen. After completion of a MAP OUT function, the ERROR-STATUS field of the CA IDMS/DC communications block indicates the outcome of a pageable-map operation using the following error messages:

Message	Description
4664	There is no current detail occurrence to be updated (MAP OUT DETAIL CURRENT only). No action is taken.

Message	Description
4668	The amount of storage defined for pageable maps at system generation is insufficient. No action is taken. This and subsequent MAP OUT DETAIL commands are ignored.
4672	No detail occurrence, footer, or header fields exist to be mapped out by a MAPOUT RESUME command.
4676	The first screen page has been transmitted to the terminal.

Syntax



DETAIL

Specifies that the MAP OUT command is to create or modify a detail occurrence for a pageable map, and optionally associates a numeric key value with the occurrence:

NEW/CURRENT

Specifies whether the detail is to be created or modified:

- **NEW** (default) creates a detail occurrence of a pageable map. Detail occurrences are displayed in the order in which they are created.
- **CURRENT** modifies the detail occurrence that was referenced by the most recent MAP IN DETAIL or MAP OUT DETAIL statement.

KEY IS *key-v* (optional)

Specifies a value associated with the created or modified detail occurrence. The value is not displayed on the terminal screen. *Key-v* is the name of a variable data field that contains the key of a database record associated with the detail occurrence.

The specified value is stored as a 4-byte value. When the KEY IS parameter is used with a MAP OUT DETAIL CURRENT command, the specified value replaces the value (if any) previously associated with the detail occurrence.

RESUME PAGE IS

Specifies that a page of detail occurrences is mapped out to the terminal from the session scratch record. Detail occurrences in the scratch record are divided into pages at runtime based on the number of detail occurrences that can fit on the screen at one time.

The page of occurrences displayed is determined by the PAGE IS clause:

- **CURRENT** (default) specifies that the current page is redisplayed. If no page has been displayed, the first page of the pageable map is displayed.
- **NEXT** specifies that the page that follows the current page is displayed. If no page follows the current page, the current page is redisplayed.
- **PRIOR** specifies that the page that precedes the current page is displayed. If no page precedes the current page, the current page is redisplayed.
- **FIRST** specifies that the first available page of detail occurrences is displayed.
- **LAST** specifies that the page of detail occurrences with the highest available page number is displayed. field in which is stored the binary fullword number of the displayed page. A page number is stored in the variable field by a preceding MAP IN PAGE IS *data-field-name-v* statement that names the same numeric variable field.

STARTPAGE

A STARTPAGE statement initiates the paging session. It can be followed by any number of DML commands, including MAP IN and MAP OUT commands. The map-paging session is terminated by an ENDPAGE command (or by another STARTPAGE command, if one is encountered before an ENDPAGE command).

After completion of a STARTPAGE function, the ERROR-STATUS field of the IDMS-DC communications block indicates the outcome of a pageable-map operation:

Message Number	Description
4604	A paging session was already in progress when this STARTPAGE command was received. An implied ENDPAGE was processed before this STARTPAGE was successfully executed.

Syntax

```
►►─── STARTPAGE session (map-name) ──────────────────────────────────────►

 ►─┬───────────┬─┬──────────────┬─┬──────────┬─┬───────────────┬─ ; ───►◄
   ├ WAIT ─────┤ └ BACKPAGE ◄ ─┤ ├ UPDATE ◄ ─┤ ├ AUTODISPLAY ◄ ─┤
   ├ NOWAIT ◄ ─┤   NOBACKPAGE ─┘ └ BROWSE ───┘ └ NOAUTODISPLAY ─┘
   └ RETURN ───┘
```

STARTPAGE SESSION *map-name*

Specifies the beginning of a pageable map session, specifies the name of the pageable map used for the session, and specifies the map paging options in effect for the session. The STARTPAGE command must precede any commands (such as MAP IN) that specify operations performed using the map.

NOWAIT/WAIT/RETURN

Specifies the runtime flow of control when the operator presses a control key:

- *NOWAIT* (default) specifies that runtime mapping automatically handles all paging and update transactions. Control is passed to the program only when neither an update nor paging request is made when the operator presses a control key.
- *WAIT* specifies that runtime mapping automatically handles paging transactions that do not update data. Control is passed to the program when the operator presses a key that requests an update or a nonpaging operation.
- *RETURN* specifies that runtime mapping does not handle any terminal transactions in the paging session. Control is passed to the program when the operator presses a control key.

Runtime mapping does not update program variable storage unless the application program issues a MAP IN command. In cases where the operator can update data, it is recommended that WAIT or RETURN be specified for the paging session so that data can be retrieved as it is updated.

BACKPAGE/NOBACKPAGE

Specifies whether the terminal operator can display a previous map page:

- *BACKPAGE* (default) specifies that the operator can display previous pages of detail occurrences.
- *NOBACKPAGE* specifies that the operator cannot display any page of detail occurrences with a page number lower than the current page number. Modifications made on a given page of the map must be retrieved by a MAP IN statement in the application program prior to a MAP OUT RESUME statement. The previous page of detail occurrences is deleted from the session scratch record when a new map page is displayed.

UPDATE/BROWSE

Specifies whether the terminal operator can modify map data fields:

- *UPDATE* (default) specifies that the terminal operator can modify variable map fields, subject to restrictions specified for the map either at map-definition time or by statements in the program.
- *BROWSE* specifies that the terminal operator can modify only the page and response fields (if any) of the map. The MDTs for variable fields on the map can be set on only according to specifications made either in the map-definition or by statements in the program.

NOBACKPAGE cannot be assigned if UPDATE and NOWAIT are specified for the session.

ENDPAGE

The ENDPAGE statement terminates a map-paging session, clears the scratch record for the session, and clears the map paging options for the completed session. A STARTPAGE/ENDPAGE pair enclose commands that handle a pageable map at runtime.

Syntax

```
►►─── ENDPAGE session ; ──────────────────────────────────────────────►◄
```

Index