Natural

Editors

Version 9.3.1

April 2025

ADABAS & NATURAL

This document applies to Natural Version 9.3.1 and all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: NATUX-NNATEDITORS-931-20250409

Table of Contents

Preface

This documentation describes all editors available in Natural.

For a tutorial on using the editors, see the *First Steps* documentation.

For information on Unicode and code page support for Natural editors, see *Editors in the SPoD Environment* in the *Unicode and Code Page Support* documentation.

The *Editors* documentation is organized under the following headings:

NaturalONE as the Default Development Environment	Provides description about NaturalONE as the default development environment and related edit functions.
Program Editor	Describes the program editor which is used to create and modify Natural programs, subprograms, subroutines, classes, copycodes, helproutines, functions and text objects.
Data Area Editor	Describes the data area editor which is used to create and modify local, global and parameter data areas.
Map Editor	Describes the map editor which is used to create and modify maps (screen layouts).
DDM Services	Describes DDM Services which are used to create, maintain and delete Natural data definition modules (DDMs).

1 About this Documentation

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
`Monospace font`	Identifies service names and locations in the format *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
`Italic`	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at **https://documentation.softwareag.com**.

In addition, you can also access the cloud product documentation via **https://www.softwareag.cloud**. Navigate to the desired product and then, depending on your solution, go to “Developer Center”, “User Center” or “Documentation”.

Product Training

You can find helpful product training material on our Learning Portal at **https://knowledge.softwareag.com**.

Tech Community

You can collaborate with Software AG experts on our Tech Community website at **https://tech-community.softwareag.com**. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software AG news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at **https://github.com/softwareag** and **https://hub.docker.com/publishers/softwareag** and discover additional Software AG resources.

Product Support

Support for Software AG products is provided to licensed customers via our Empower Portal at **https://empower.softwareag.com**. Many services on this portal require that you have an account. If you do not yet have one, you can request it at **https://empower.softwareag.com/register**. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

I Disabled Natural Editors

2 Disabled Natural Editors

Since Natural Version 9.1, the following Natural editors have been disabled (deactivated) by default in a local z/OS mainframe and UNIX environment:

Details

- Program editor
- Data area editor
- Map editor

This change does not affect the following:

- DDM editor (SYSDDM utility)
- Software AG Editor
- Batch mode processing (batch programs will run as usual)
- APIs and user exits that access editor interfaces

You can use the `TECH` system command (or corresponding API `USR2026N`) and the `*EDITOR` system variable to find out whether Natural editors are disabled in the current Natural environment.

With Natural version 9 on Mainframe, Linux and Cloud, NaturalONE, and the Natural Development Server are integrated into Natural. The Natural editors (program editor, data area editor and map editor) are disabled. As NaturalONE and Natural Development Server licenses are integrated with Natural, they can be activated with the Natural (NAT) license key. This does not apply to Natural for Windows.

Disabled Natural editors are substituted by the editors provided with NaturalONE. For this purpose, NaturalONE and Natural Development Server are supplied free of charge for mainframe customers who move their development environment to NaturalONE.

However, an environment with Natural ISPF installed, allows for using all Natural editors.

If Natural editors are disabled in the current Natural environment, the editor-related functions of the following commands and features are no longer supported and a corresponding error message is returned when you request an edit function:

- `EDIT` system command,
- `LIST` system command,
- `SCAN` system command,
- SYSEXT utility.

II NaturalONE as the Default Development Environment

3 NaturalONE as the Default Development Environment

Since Natural Version 9.1 for z/OS and Linux, NaturalONE is the default development environment. The Natural Development Server is integrated into Natural as well.

You can activate both NaturalONE and Natural Development Server with the Natural (NAT) license key.

Note: This does not apply to Natural for Windows.

With NaturalONE the edit functionality is provided in a modern Eclipse environment. The former "character interface" or "green screen" editors (program editor, data area editor, and map editor) in a local z/OS or Linux environment are disabled.

You can use the `TECH` system command (or corresponding API `USR2026N`) and the `*EDITOR` system variable to find out whether Natural editors are disabled in the current Natural environment.

Details

This change does not affect the following:

- DDM editor (SYSDDM utility)
- Software AG Editor
- Batch mode processing (batch programs will run as usual)
- APIs and user exits that access editor interfaces

You can use the `TECH` system command (or corresponding API `USR2026N`) and the `*EDITOR` system variable to find out whether Natural editors are disabled in the current Natural environment.

If Natural editors are disabled in the current Natural environment, the editor-related functions of the following commands and features are no longer supported and a corresponding error message is returned when you request an edit function:

- `EDIT` system command
- `LIST` system command
- `SCAN` system command
- SYSEXT utility.

III Program Editor

The Natural program editor is used to create and modify the source code of a Natural object of the type program, subprogram, subroutine, helproutine, copycode, text, class or function.

Note: The Natural program editor has been disabled in your environment by default. For more information, see *NaturalONE as the Default Development Environment* in the *Editors* documentation.

Related Topic:

For information on Unicode and code page support for Natural editors, see *Editors in the SPoD Environment* in the *Unicode and Code Page Support* documentation.

4 Invoking the Program Editor

To invoke the program editor

- Use the system command `EDIT` as described in the *System Commands* documentation.

  When the program editor is invoked, an editor screen similar to the example below appears:

```
>> -----------Columns 001 072 <<  Program SAGDEMO  Lines  14     User SAG     ↩
Command ===>                                        Mode   Struct Lib  SAGTEST ↩
****** ***************************** top of data ****************************
000010 ** Example 'SAGDEMO': DISPLAY                                          ↩
000020 ************************************************************************
000030 DEFINE DATA LOCAL                                                      ↩
000040 1 VIEWEMP VIEW OF EMPLOYEES                                            ↩
000050   2 PERSONNEL-ID                                                       ↩
000060   2 NAME                                                               ↩
000070   2 BIRTH                                                              ↩
000080   2 JOB-TITLE                                                          ↩
000090 END-DEFINE                                                             ↩
000100 *                                                                      ↩
000110 READ (3) VIEWEMP BY BIRTH                                              ↩
000120   DISPLAY PERSONNEL-ID NAME JOB-TITLE                                  ↩
000130 END-READ                                                               ↩
000140 END                                                                    ↩
****** **************************** bottom of data ****************************
                                                                             ↩
                                                                             ↩
                                                                             ↩
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Save  Exit  Run   Rfind Stow  -     +     Check Home  Undo  Canc
```

The editor screen contains the following items (from top to bottom): the **top information line**, the **editor command line**, the **prefix area**, the **editing area** and PF-key lines (described in *Displaying and Modifying PA/PF-Key Assignments*). These items are explained in the following sections.

5 Terminating the Program Editor

To terminate the program editor

- In the command line (`Command ===>`), enter one of the following editor commands:

  ```
  EXIT
  ```

  or

  ```
  CANCEL
  ```

  or

  ```
  .
  ```

  (a period)

  Or:

  On the editor screen, press PF3 or PF12.

Before leaving the editor, depending on your editor profile settings, the **EXIT Function** window appears as described in the following section.

Exit Function

If the editor profile option **Prompt Window for Exit Function** or **Prompt Window for Cancel Function** is set to `Y`, any time you execute the exit function (PF3 or `EXIT` editor command) or the cancel function (PF12 or `CANCEL` editor command), the **EXIT Function** prompt window is invoked. This window provides the following options:

Option	Explanation
Save and Exit	Leaves the editor after saving all modifications made to the current source code.
Exit without Saving	Leaves the editor without saving any modifications made to the current source code since it was last saved.
Resume Function	Neither leaves the editor nor saves any modifications; the prompt window is closed and the current function is resumed.

When **Prompt Window for Exit Function** or **Prompt Window for Cancel Function** is set to `N`, the corresponding exit or cancel function terminates the editor immediately without displaying the prompt window. Any changes made to the current source since you last saved the source are *not* saved.

For details on saving a source, see *Saving and Cataloging Sources*. For details on setting editor profile options, see the section *Editor Profile*.

6 Top Information Line

The top information line of the editor screen is indicated by double greater than signs (>>). It contains the following items (from left to right):

- **Columns**

  The first and the last column currently displayed. You can enter text beyond the 72nd column, but to ensure that data can be handled by other platforms you are advised to use only columns 1-72.

- **Object Type**

  The type of object currently in the source work area. If no object type or object name is specified when the program editor is invoked, object type **Program** is displayed by default.

  The object type can be changed by using the `SET TYPE` editor command.

- **Object Name**

  The name of the object currently in the source work area. No name is displayed if the source work area is empty or if the current source code has not yet been saved as a source object (see also *Saving and Cataloging Sources*).

- **Lines**

  The total number of lines currently used by the editor (source lines and information lines).

- **User**

  The ID of the current user.

7 Editor Command Line

The editor command line is indicated by **Command = = =>**. In the command line, you can enter one of the following:

- Any Natural system command.

  For example: The system command `CHECK` can be used for checking the syntax of source code and `SAVE` for saving source code (see also ***Saving and Cataloging Sources***).

  For other system commands related to maintaining and using object sources, see *Managing Applications with Natural Objects* in the *System Commands* documentation.
- An editor command (see ***Editor Commands*** and ***Editor Commands for Scrolling***).
- The name of a Natural program to be executed.

Additionally, the command line contains the following information (from left to right):

- **Mode**

  The programming mode: structured (`Struct`) or reporting (`Report`) currently in effect. When a Natural object is read into the source work area, the mode is set to the one which was in effect when the object was saved (see also ***Saving and Cataloging Sources***).

  For information on the differences between structured and reporting mode, see *Purpose of Programming Modes* in the *Programming Guide*.

  To switch from reporting to structured mode or vice versa, enter an `S` in the first position of `Report` or an `R` in the first position of `Struct` and press ENTER.

  If you want to check whether structured mode has been activated or switch structured mode on, you can also set the programming mode for the current Natural session as described in *Setting/Changing the Programming Mode* in the *Programming Guide*.

- **Lib**

  The library to which you are currently logged on.

8 Prefix Area

The leftmost six columns of the editor screen are referred to as the prefix area. The prefix area can contain the following:

- A 6-digit line number (for example, `000010`) for a line that contains source code.

  Note: For technical reasons six digits are shown, but only four digits are processed internally.

- A series of apostrophes (`''''''`) for a blank source line. It appears when the insert mode is active. Once you enter text in this line and press ENTER, the apostrophes are replaced by a line number.
- A series of asterisks (`******`) for an empty editing area. You then need to activate the insert mode to enter text by using one of the `I` (insert) line commands described in the relevant section.
- Text that indicates the contents or state of a line. For example:

  `=prof>` indicates that this line contains information on the current editor profile settings.

  `=cols>` indicates that this line shows the current column positions.

  `==chg>` indicates that a character string was replaced in this line.

  `.X` indicates that this line is marked with a label.

To hide text in the prefix area and reset pending commands

- In the command line, enter the following editor command:

```
RESET
```

Entering Line Commands

In addition to showing the number of a source line or text, the prefix area is used for entering **line commands** as described in the relevant section.

You can protect the prefix area to avoid overwriting of source lines by using the `PROTECT` editor command described in *Editor Commands*.

9 Editing Area

The editing area is either empty or contains source code that was last read into the source work area with the command `EDIT` or `READ` as shown in the example of a program in *Invoking the Program Editor*.

When you read in the source of an existing object, the entire source code is loaded into the source work area and is available for editing. However, depending on the size of the source, the editing area may not show all of the lines that belong to the source. In this case, you have to scroll down the source (see *Editor Commands for Scrolling*) to go to the line you want to view or modify.

In addition, if you use **split-screen mode** (see the relevant section), the editing area displays fewer lines of source code.

You can use multiple methods to create or modify source code:

- Type in or update text directly in the relevant source line.
- Use one or more **line commands** as described in the relevant section.

  A line command, for example, is used to insert or delete a line or copy variable or field definitions within the current source.
- Use one or more editor commands or alternative PF keys as described in *Editor Commands for Scrolling*, *Editor Commands* and *Displaying and Modifying PA/PF-Key Assignments*.

  An editor command, for example, is used to scroll in the source, find and replace text strings, undo an edit action, invoke the editor profile facility or display another object from which you can copy variable or field definitions.

Inserting Text

This section provides example instructions for using a line command to insert text into the current source.

To insert text into a source

1 In the leftmost column of the **prefix area**, next to the required source line, enter an `I` (insert) command (see *Line Commands*) thus temporarily overwriting the characters in the prefix area.

  For example, enter the following line command:

```
I5
```

  Insert mode is activated as indicated by a series of apostrophes (`' ' ' ' ' '`) and five blank source lines are inserted below the line in which you entered the line command. The cursor is positioned at the beginning of the first blank source line.

2 In the blank source lines, type in text. Lines that are left blank are eliminated from the source when you press ENTER.

The apostrophes are replaced by line numbers indicating that the new text has been incorporated into the source.

Renumbering of Source Lines

When you add lines to the source code of an object, Natural numbers the added lines in increments of ten. When you insert lines in source code, Natural automatically renumbers all lines contained in the source code in increments of ten beginning with `000010` at the first line.

You can change the number of increments by using the system command `RENUMBER` described in the *System Commands* documentation.

See also *Renumbering of Source-Code Line Number References* in the *Programming Guide*.

Copying and Pasting Text

This section provides instructions for copying text within the current source by using copy and paste functionality. For instructions on copying text from another Natural object into the current source, see ***To display and copy definitions with SPLIT***.

To cut or copy and paste text to a source

1 Cut or copy text to the clipboard.

2 Add the number of source lines required to place the text you cut or copied to the clipboard, for example, by performing the following steps:

- Insert a line with the `I` line command.
- Enter a character in the new line.
- Copy the new line with the `Rn` line command where *n* is the repetition factor.

3 In the command line, enter the following:

```
PROTECT ON
```

This command protects the **prefix area** to avoid overwriting of source lines if a pasted line contains more than 72 characters.

4 Position the cursor in the first position of the first source line where you want to paste the text.

5 Choose the paste function of your terminal emulation or press CTRL+V.

The text placed on the clipboard is pasted in the source lines from the cursor position to the end of the source.

If a pasted line starts with a 6-digit line number copied from a source, you can remove the line number with the following steps.

6 In the command line, enter the following:

```
PROTECT OFF
```

The prefix area is no longer protected and you can enter a line command.

7 Next to the pasted line that contains a copied line number, enter the following line command:

```
(7
```

The line is shifted left by 7 columns thus deleting the line number.

Copying or Moving Text with a Data Window

You can use a data window to copy or move text that does not start or end at the beginning or end of a line. Data window functions can be performed by using line commands and/or editor commands.

When you define a window, all text on your screen between start and end of the window become part of the window.

Example of Using a Data Window

This section provides an example of defining and moving text with a data window by using either line commands or corresponding editor commands.

The example refers to the text shown in Step 1 and assumes that you want to move the whole sentence starting `Note that when...` (line 80) to follow the first sentence of the displayed text ending `...copy operations` (line 30).

To define and move a window using line commands

1 Type in text as shown below:

```
>> -----------Columns 001 072 <<     Text WINEX    Lines  10     User MMO     ↩
Command ===>                                        Mode   Report Lib  SAGTEST ↩
****** ***************************** top of data ****************************
000010 Copy a Window with Text                                                ↩
000020                                                                        ↩
000030 You can specify a window with text for move or copy operations. This   ↩
000040 allows you to copy or move text that does not start or end at the      ↩
000050 beginning or end of a line. This function can be performed using       ↩
000060 line commands and/or editor commands.                                  ↩
000070                                                                        ↩
000080 Below are some examples of copying windows with text. Note that when   ↩
000090 you define a window, all text on your screen between start and end of  ↩
000100 the window become part of the window. Available line commands are:     ↩
****** **************************** bottom of data ****************************
                                                                              ↩
                                                                              ↩
                                                                              ↩
                                                                              ↩
                                                                              ↩
                                                                              ↩
                                                                              ↩
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Save  Exit  Run   Rfind Stow             Check Home  Undo  Canc
```

2 Type the line command `WS` in line 80, the first line of text to be moved, place the cursor in the required column (`N` of the word `Note`) and press ENTER.

The message `WS55` appears in the prefix area of line 80, indicating the column number selected:

```
>> ----------Block is pending <<     Text WINEX    Lines  10     User MMO
Command ===>                                     Mode   Report Lib  SAGTEST
****** ***************************** top of data ******************************
000010 Copy a Window with Text
000020
000030 You can specify a window with text for move or copy operations. This
000040 allows you to copy or move text that does not start or end at the
000050 beginning or end of a line. This function can be performed using
000060 line commands and/or editor commands.
000070
WS55   Below are some examples of copying windows with text. Note that when
000090 you define a window, all text on your screen between start and end of
000100 the window become part of the window. Available line commands are:
****** **************************** bottom of data ****************************









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Save  Exit  Run   Rfind Stow             Check Home  Undo  Canc
```

3 Type the line command `WE` in line 100, the last line of text to be moved, move the cursor to the last column to be moved (full stop (`.`) after `window`) and press ENTER.

The message `WE37` appears in the prefix area of line 100:

```
>> ----------Block is pending <<     Text WINEX    Lines  10     User MMO
Command ===>                                     Mode   Report Lib  SAGTEST
****** ***************************** top of data ******************************
000010 Copy a Window with Text
000020
000030 You can specify a window with text for move or copy operations. This
000040 allows you to copy or move text that does not start or end at the
000050 beginning or end of a line. This function can be performed using
000060 line commands and/or editor commands.
000070
WS55   Below are some examples of copying windows with text. Note that when
000090 you define a window, all text on your screen between start and end of
WE37   the window become part of the window. Available line commands are:
****** **************************** bottom of data ****************************




Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Save  Exit  Run   Rfind Stow             Check Home  Undo  Canc
```

4 Type the line command `WM` in line 30 (the text will be moved to the following line, line 40), and move the cursor to the column at which line 30 is to be split (the blank before the word `This`). Press ENTER.

The specified text section is moved:

```
>> -----------Columns 001 072 <<     Text WINEX    Lines  13     User MMO
Command ===>                                      Mode   Report Lib  SAGTEST
****** ***************************** top of data ******************************
000010 Copy a Window with Text
000020
000030 You can specify a window with text for move or copy operations.
000040 Note that when
000050 you define a window, all text on your screen between start and end of
000060 the window become part of the window.
000070 This
000080 allows you to copy or move text that does not start or end at the
000090 beginning or end of a line. This function can be performed using
000100 line commands and/or editor commands.
000110
000120 Below are some examples of copying windows with text.
000130                                        Available line commands are:
****** *************************** bottom of data ******************************




Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Save  Exit  Run   Rfind Stow             Check Home  Undo  Canc
```

You can achieve the same result described in the instructions above by using the command sequence indicated below.

To define and move a window using editor commands

1 Type in the text shown in **Step 1** above.

2 In the command line, enter the following:

```
WINDOW 80 100 55 37;MWINDOW 30 64
```

The specified text section is moved as illustrated in **Step 4** above.

Finding and Replacing Text

This section provides instructions for finding and replacing a character string in the current source.

To find and replace a character string in a source

1 Search for a string by using the following editor command:

```
FIND 'string'
```

where *string* is any alphanumeric character string.

The first occurrence of the specified string is found.

2 Replace the string found by using the following command:

```
CHANGE 'string'  'new-string'
```

Or:

Search for the next occurrence of the string by pressing PF5.

Depending on the command used, the first occurrence of the string is either replaced or the next occurrence (if any) of the string is found.

3 Replace the string found by pressing PF17.

Or:

Search for the next occurrence of the string by pressing PF5.

Depending on the PF key used, the next occurrence of the string is either replaced or the next but one occurrence (if any) of the string is found.

Formatting Source Code

You can format source code by indenting source lines. Indentation is performed differently for sources created in reporting mode than for sources created in structured mode.

To format source code

- In the command line, enter the following system command:

  ```
  STRUCT
  ```

  The lines in the source code are indented.

For details, see STRUCT in the *System Commands* documentation.

Checking Source Code

To check the current source code for syntax errors

- On the editor screen, press PF9.

  Or:

  In the command line, enter the following:

  ```
  CHECK
  ```

  The source code currently contained in the source work area is checked for syntax errors. If a syntax error is detected, an appropriate error message appears indicating the line number affected and the erroneous definition.

10 Split-Screen Mode

In split-screen mode, you can use one half of the screen for editing an object (editing section) and at the same time have another Natural object displayed in the other half (display section). In addition, you can copy the definitions shown in the display section.

The following types of Natural object can be displayed in split-screen mode:

DDMs (data definition modules), data areas, programs, subprograms, subroutines, helproutines, copycodes, texts, maps, classes, functions and Predict program descriptions.

The instructions below show how to display the definitions of another Natural object and copy them into the object you are currently editing.

To display and copy definitions with SPLIT

1 In the current source, enter the SPLIT editor command (described in *Editor Commands*) to display the source of the object from which you want to copy text, for example:

```
SPLIT P LDATEST1
```

Split-screen mode is set with the current source (in the example below program PGMTEST1) in the editing section (upper half) and a local data area (LDATEST1) in the display section (lower half) of the editor screen as shown in the example below:

```
   >> -----------Columns 001 072  <<  Program PGMTEST1 Lines  187    User SAG ↩
   Command ===>                                                  Lib  SYSLIB ↩
   001720   IF (RC NE 0)                                                     ↩
   001730    then                                                           ↩
   001740     reset CMD_LINE_2                                              ↩
   001750     IF (s_prog ne ' ')                                            ↩
   001760      then                                                         ↩
   001770       assign CMD_LINE_PGM = S_PROG                                ↩
   001780       assign CMD_LINE_LIB = S_LIB                                 ↩
   001790     END-IF                                                        ↩
   001800   END-IF                                                          ↩
   >> -----------Columns 001 072  <<    Local LDATEST1 Lines  158    User SAG ↩
   Command ===>                                                  Lib  SYSLIB ↩
   ****** ***************************** top of data ↩
*****************************
   000001 DEFINE DATA LOCAL                                                ↩
   000002 1 NPF_F_READ(A1)                                                 ↩
   000003 CONST                                                            ↩
   000004 <'R'>                                                            ↩
   000005 1 NPF_F_WRITE(A1)                                                ↩
   000006 CONST                                                            ↩
   000007 <'W'>                                                            ↩
   ↩
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
        Help  Save  Exit  Run   Rfind Stow  -     +     Check Home  Undo  Canc
```

Notes:

1. Because it is not possible to modify the definitions in the display section, not all editor commands are available. PF keys are reserved for the editing section. Thus, commands meant for the display section must be issued using the command line.
2. The SWAP editor command can be used to move the cursor between the command lines of the display and editing sections.

2 In the display section, mark the block of lines to be copied from LDATEST1, for example, by entering the line command CC next to the first and the last line of the required block.

3 In the editing section of PGMTEST1, next to the line below which you want to place the block of lines, enter the line command A.

The marked block of lines is copied into PGMTEST1 below the line where you entered the line command.

4 If required, terminate split-screen mode by entering the SPLIT END editor command.

11 Editor Commands

ADVANCE | AORDER | AUTOSAVE | BNDS | CANCEL | CAPS | CENTER | CHANGE | COLS | CWINDOW | DELETE | DWINDOW | DX | DY | DX-Y | EMPTY | EX | EY | EX-Y | EXCLUDE | EXIT | FIND | FLIP | HEX | HOME | INCLUDE | JLEFT | JRIGHT | JUSTIFY | LABEL | LC | LIMIT | LOCATE | LOG | MASK | MWINDOW | NEXT | ORDER | POINT | POWER | PROF | PROFILE | PROTECT | RCHANGE | RESET | RFIND | SET TYPE | SHIFT | SORT | SPLIT | SWAP | TABS | UC | UNDO | WINDOW | X | XSWAP | Y | **Common Command Options**

This section describes all editor commands that can be used to modify a source and summarizes commonly used editor command options. In addition to the editor commands listed above, you can use **editor commands to scroll** through a source and **line commands** to manipulate single or multiple source lines.

Using an Editor Command

- You enter an editor command in the **command line** of the editor screen. Depending on the configuration of your installation, an editor command can be entered in lower case. In this section, however, all commands are shown in upper case to distinguish them as commands.
- You can enter several editor commands in the same input operation if you separate them with a semicolon (;).
- For explanations of the syntax symbols used in this section, refer to *System Command Syntax* in the *System Commands* documentation.
- Some frequently used commands can be issued using **PF keys** as indicated in this section.

ADVANCE

```
ADVANCE [ ON  ]
        [ OFF ]
        [ PAGE ]
```

This command is used to specify whether the cursor moves to the next line automatically after a line update.

ON	The cursor moves to the next line after an update.
OFF	The cursor does not move to the next line after an update.
PAGE	The line containing the cursor is placed at the top of the editing area after an update.

The ADVANCE command issued without a parameter has the same effect as ADVANCE ON.

AORDER

```
AORDER [ ON  ]
       [ OFF ]
```

This command is used to specify whether text is to be automatically justified within the set boundaries.

The AORDER command issued without a parameter has the same effect as AORDER ON. The base setting can be changed by editing your profile.

AUTOSAVE

```
{ AUTOSAVE } [ ON  ]
{ ASAVE    } [ OFF ]
```

This command is used to specify whether the editor executes an automatic SAVE command when you issue the EXIT command.

The AUTOSAVE command issued without a parameter has the same effect as AUTOSAVE ON.

BNDS

```
BNDS [ n m ]
     [ n   ]
```

This command is used to restrict the effect of certain commands to a specific range of columns.

These boundaries apply to the editor commands FIND, CHANGE, CENTER, ORDER, JLEFT and JRIGHT, and their corresponding line commands (if available) such as TC, TO, LJ and RJ.

n	The number of the column at which the left boundary is to be placed.
m	The number of the column at which the right boundary is to be placed.

If *n* and *m* are omitted, the boundaries are set at the first and last column of the editing area.

You can issue the BNDS line command to see the current boundary settings.

CANCEL

```
CANCEL
```

Alternative PF key: PF12

This command cancels all changes made since you last saved the source (see also ***Saving and Cataloging Sources***) and leaves the editor. Depending on your editor profile settings, you are prompted to save your changes or leave without saving (see also ***Exit Function***).

CAPS

```
CAPS [ ON  ]
     [ OFF ]
     [ PGM ]
```

This command is used to switch upper-case translation on or off. This command only applies to lines which are created or modified after the command is issued.

ON	Text in line is translated to upper case.
OFF	Text in line is not translated; that is, it remains as entered.
PGM	Text in line is translated to upper case (except for comments, which remain as entered).

The CAPS command issued without a parameter has the same effect as CAPS ON.

CENTER

```
CENTER { ALL }
       { n   }
       { n m }
```

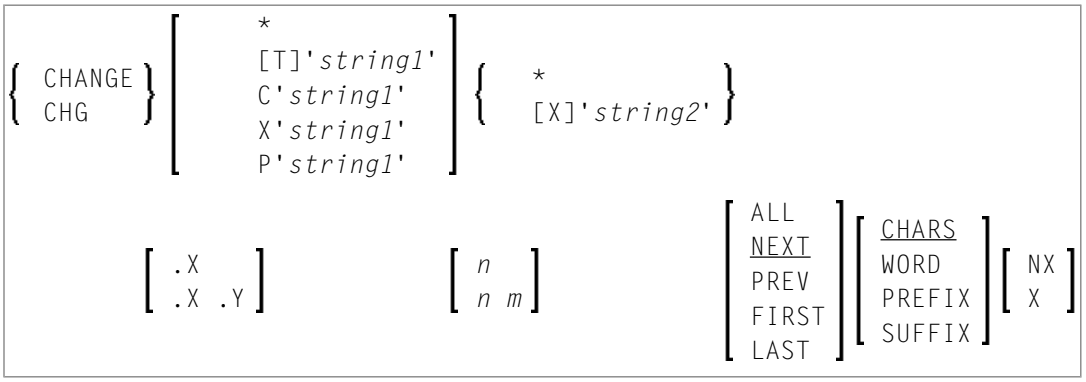
This command is used to center text.

ALL	Centers the text of all lines.
n	Centers the text from line *n* to the last line.
n m	Centers the text from line *n* to line *m*.

The CENTER command applies only within the horizontal boundaries as set with the BNDS editor command.

For centering, you can also use the line commands TC and TCC.

CHANGE



This command is used to replace a character string (*string1*) by another character string (*string2*).

If you want an apostrophe to be part of *string1* or *string2*, you must write it as two apostrophes.

You can specify the string to be replaced (*string1*) as described in the following section.

T'*string1*'	Replaces *string1* irrespective of whether it occurs in lower case or upper case. This is the default.
'*string1*'	Same as T'*string1*'.
C'*string1*'	Replaces *string1* only if it occurs exactly as specified.
X'*string1*'	Replaces the string that corresponds to the specified hexadecimal character string *string1*. Replace it by the hexadecimal string *string2*.
P'*string1*'	Replaces *string1* which includes the following wildcard characters:

=	any character
§	alphabetic character

	#	numeric character
	$	special character
	^	non-blank character
	-	non-numeric character
	<	lower-case letter
	>	upper-case letter
*	Uses the character string specified in a previous command, for example, FIND, CHANGE or EXCLUDE.	
.X .X .Y	See *Line Specifications* for an explanation.	
n *n m*	See *Column Specifications* for an explanation.	
ALL NEXT PREV FIRST LAST	See *Direction of Operation* for an explanation.	
CHARS WORD PREFIX SUFFIX	See *Special Occurrences* for an explanation.	
NX X	See *Displayed or Non-Displayed Lines* for an explanation.	

This section covers the following topics:

- Using CHANGE Together with Other Commands

Using CHANGE Together with Other Commands

You can use the command RCHANGE to repeat the execution of a CHANGE command.

To search the entire source code for a character string and then decide occurrence by occurrence whether to replace it by another character string, you can use a combination of the commands FIND and CHANGE and the PF keys assigned to the commands RFIND and RCHANGE as described in *Finding and Replacing Text*.

Examples of the CHANGE Command

Example 1:

```
CHG 'LOW' 'HIGH'
```

This command replaces the first occurrence of LOW by HIGH, regardless of upper or lower case.

Example 2:

```
CHG C'OPS' 'SPF' .X .Y 28 32 ALL
```

This command changes OPS (exactly as entered here) into SPF; it changes all occurrences in the block of lines labeled with .X and .Y and between columns 28 and 32.

Example 3:

```
CHG C'NAME' 'APPL' .X .Y ALL PREFIX NX
```

This command changes all occurrences of a string that begins with NAME (exactly as entered here) into APPL in all displayed lines in the block of lines labeled with .X and .Y.

Example 4:

```
CHG * 'NEW'
```

This command replaces the next occurrence of the string specified in the last CHANGE command by the string NEW.

Example 5:

```
CHG 'OLD' *
```

This command replaces the next occurrence of the string OLD by the same new string as specified in the last CHANGE command.

COLS

This command displays a line at the top of the editing area showing column positions.

You can also use the line command `COLS` to display the column positions.

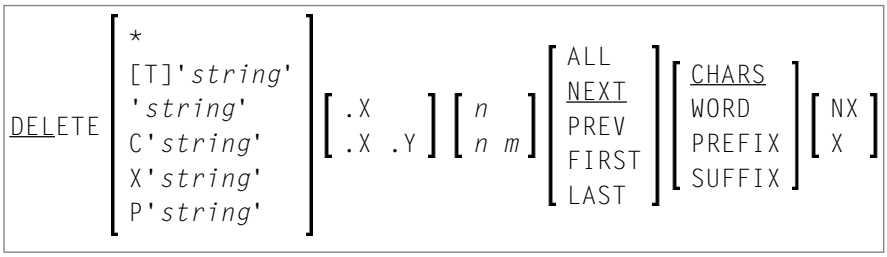
CWINDOW

This command is used to copy a data window according to the command parameters.

n	The number of the line in which the data window is to be inserted.
m	The number of the column in which the data window is to be inserted.

See also *Copying and Moving Text with a Data Window*.

DELETE



This command is used to delete lines.

You can specify that only lines which contain a specified character *string* are to be deleted as described in the following section.

<table>
<tr><td>T'string'</td><td>Deletes lines that contain the string irrespective of lower case or upper case.

This is the default.</td></tr>
<tr><td>'string'</td><td>Same as T'string'.</td></tr>
<tr><td>C'string'</td><td>Deletes lines that contain the string exactly as specified.</td></tr>
<tr><td>X'string'</td><td>Deletes lines that contain the string which corresponds to the specified hexadecimal character string.</td></tr>
<tr><td>P'string'</td><td>Deletes lines that contain the string which includes the following wildcard characters:
<table><tr><td></td><td></td></tr><tr><td>=</td><td>any character</td></tr><tr><td>§</td><td>alphabetic character</td></tr><tr><td>#</td><td>numeric character</td></tr><tr><td>$</td><td>special character</td></tr><tr><td>^</td><td>non-blank character</td></tr><tr><td>-</td><td>non-numeric character</td></tr><tr><td><</td><td>lower-case letter</td></tr><tr><td>></td><td>upper-case letter</td></tr></table></td></tr>
<tr><td>*</td><td>Uses the search string specified in a previous command, for example, FIND, CHANGE or EXCLUDE.</td></tr>
<tr><td>.X</td><td>See Line Specifications for an explanation.</td></tr>
<tr><td>.X .Y</td><td></td></tr>
<tr><td>n
n m</td><td>See Column Specifications for an explanation.</td></tr>
<tr><td>ALL
NEXT
PREV
FIRST
LAST</td><td>See Direction of Operation for an explanation.</td></tr>
<tr><td>CHARS
WORD
PREFIX
SUFFIX</td><td>See Special Occurrences for an explanation.</td></tr>
</table>

NX X	See *Displayed or Non-Displayed Lines* for an explanation.

If you enter the DELETE command without any parameters, the current line is deleted.

You can also use the line command D, D*n* or DD to delete lines.

Example 1:

```
DEL C'NAME' 1 20 ALL PREFIX NX
```

This command deletes all lines that contain the string NAME (in upper case exactly as entered here) as a prefix to a word in all lines not excluded from display if NAME occurs between columns 1 and 20.

Example 2:

```
DEL C'Abc' .X .Y 10 30 ALL
```

This command deletes all lines that contain the string Abc (exactly as entered here) between columns 10 and 30 within the block of lines labeled with .X and .Y

DWINDOW

```
DWINDOW
```

This command is used to delete the last defined data window.

DX, DY, DX-Y

```
DX
DY
DX-Y
```

These commands are used to delete marked lines.

- The DX command deletes the line marked with the .X label.
- The DY command deletes the line marked with the .Y label.
- The DX-Y command deletes all lines between the .X and .Y labels.

EMPTY

```
EMPTY [ ON  ]
      [ OFF ]
```

This command controls the deletion of blank lines.

OFF	Blank lines are not deleted.
ON	Blank lines are deleted.

The EMPTY command issued without a parameter has the same effect as EMPTY ON.

EX, EY, EX-Y

```
EX
EY
EX-Y
```

These commands are used to delete lines in a source.

- The EX command deletes *all lines* preceding the line marked with the .X label.
- The EY command deletes *all lines* following the line marked with the .Y label.
- The EX-Y command deletes *all lines* preceding the .X label and following the .Y label.

EXCLUDE

```
EXCLUDE [ *                ] [ .X    ] [ n   ] [ ALL   ] [ CHARS  ]
        [ [T]'string'      ] [ .X .Y ] [ n m ] [ NEXT  ] [ WORD   ]
        [ 'string'         ]                   [ PREV  ] [ PREFIX ]
        [ C'string'        ]                   [ FIRST ] [ SUFFIX ]
        [ X'string'        ]                   [ LAST  ]
        [ P'string'        ]
```

This command is used to exclude lines from being displayed.

You can specify that only lines which contain a specified character *string* are to be excluded from display as described in the following section.

<table>
<tr><td>T'string'</td><td>Excludes lines that contain the string irrespective of lower case or upper case. This is the default.</td></tr>
<tr><td>'string'</td><td>Same as T'string'.</td></tr>
<tr><td>C'string'</td><td>Excludes lines that contain the string exactly as specified.</td></tr>
<tr><td>X'string'</td><td>Excludes lines that contain the string which corresponds to the specified hexadecimal character string.</td></tr>
<tr><td>P'string'</td><td>Excludes lines that contain the string which includes the following wildcard characters:
<table>
<tr><td></td><td></td></tr>
<tr><td>=</td><td>any character</td></tr>
<tr><td>§</td><td>alphabetic character</td></tr>
<tr><td>#</td><td>numeric character</td></tr>
<tr><td>$</td><td>special character</td></tr>
<tr><td>^</td><td>non-blank character</td></tr>
<tr><td>-</td><td>non-numeric character</td></tr>
<tr><td><</td><td>lower-case letter</td></tr>
<tr><td>></td><td>upper-case letter</td></tr>
</table></td></tr>
<tr><td>*</td><td>Uses the search string specified in a previous command, for example, EXCLUDE, FIND or CHANGE.</td></tr>
<tr><td>.X</td><td>See Line Specifications for an explanation.</td></tr>
<tr><td>.X .Y</td><td></td></tr>
<tr><td>n
n m</td><td>See Column Specifications for an explanation.</td></tr>
<tr><td>ALL
NEXT
PREV
FIRST
LAST</td><td>See Direction of Operation for an explanation.</td></tr>
<tr><td>CHARS
WORD</td><td>See Special Occurrences for an explanation.</td></tr>
</table>

PREFIX SUFFIX	

If you enter the EXCLUDE command without any parameters, the current line is excluded from display.

You can use the INCLUDE editor command to re-display excluded lines.

Example 1:

```
EXCLUDE .X .Y
```

This command excludes lines from the line labeled with .X to the line labeled with .Y.

Example 2:

```
EXCLUDE C'NAME' ALL PREFIX
```

This command excludes from display all lines which contain strings that begin with NAME (in upper case as entered here).

EXIT

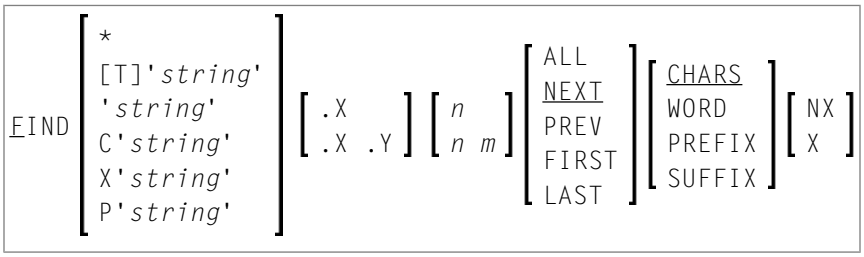
EXIT

Alternative PF key: PF3

This command is used to leave the editor. If any changes have been made since you last saved the source (see also *Saving and Cataloging Sources*), you are prompted to save your changes or leave without saving, depending on your editor profile settings (see also *Exit Function*).

Note: If AUTOSAVE is set to ON, you will not be prompted before exiting the session; your changes will then be saved automatically.

FIND



This command is used to search for a specific character *string*. The cursor is placed at the beginning of the first *string* found. If the line containing the *string* was excluded from display, it is displayed when found.

If you want an apostrophe to be part of the *string*, you must write it as two apostrophes.

You can specify the *string* as described in the following section.

T'*string*'	Searches for the *string* irrespective of lower case or upper case. This is the default.
'*string*'	Same as T'*string*'.
C'*string*'	Searches for the *string* exactly as specified.
X'*string*'	Searches for the string that corresponds to the specified hexadecimal character *string*.
P'*string*'	Searches for a *string* which includes the following wildcard characters: = any character § alphabetic character # numeric character $ special character ^ non-blank character - non-numeric character < lower-case letter > upper-case letter
*	Searches for the *string* specified in the previous command, for example, FIND, DELETE or EXCLUDE.
.X	See ***Line Specifications*** for an explanation.
.X .Y	

n *n m*	See ***Column Specifications*** for an explanation.
ALL NEXT PREV FIRST LAST	See ***Direction of Operation*** for an explanation.
CHARS WORD PREFIX SUFFIX	See ***Special Occurrences*** for an explanation.
NX X	See ***Displayed or Non-Displayed Lines*** for an explanation.

Example 1:

```
F C'NAME' .X .Y ALL PREFIX X
```

This command searches for any occurrence of `NAME` exactly as entered here as a prefix of a word in any excluded line within the block of lines labeled with `.X` and `.Y`.

Example 2:

```
F C'HILITE' X PREV
```

This command searches for the previous occurrence of `HILITE` exactly as entered here in any excluded line.

You can use the `RFIND` command to repeat the execution of a `FIND` command.

Example 3:

```
F P'RCV#' .X .Z 20 30
```

This command searches for any 4-character string that begins with `RCV` and whose fourth character is numeric. It searches within the block of lines labeled with `.X` and `.Z` and between columns 20 to 30.

Example 4:

```
F X'6C' SUFFIX NX
```

This command searches for the character with hexadecimal representation `6C`. Only those occurrences of the character that are at the end of word are found. The search is valid for non-excluded lines only.

Example 5:

```
F '''w'
```

This command searches for the following character string: `'w`

Example 6:

```
F 'r''w'
```

This command searches for the following character string: `r'w`

Example 7:

```
F ''''
```

This command searches for an apostrophe (').

FLIP

```
FLIP
```

This command is used to toggle the PF-key display between PF1 to PF12 and PF13 to PF24.

HEX

```
HEX [ ON  ]
    [ OFF ]
```

This command is used to switch hexadecimal display mode on or off.

HOME

```
HOME
```

This command returns the cursor to the command field after the next ENTER.

INCLUDE

```
INCLUDE [ *                ] [ .X     ] [ n   ] [ ALL   ] [ CHARS  ]
        [ [T]'string'      ] [ .X .Y  ] [ n m ] [ NEXT  ] [ WORD   ]
        [ 'string'         ]                   [ PREV  ] [ PREFIX ]
        [ C'string'        ]                   [ FIRST ] [ SUFFIX ]
        [ X'string'        ]                   [ LAST  ]
        [ P'string'        ]
```

This command is used to re-display lines that were excluded from display by an `EXCLUDE` command. The command takes the same parameters as the `EXCLUDE` command.

If you enter the `INCLUDE` command without any parameters, it includes the first line of an excluded block of lines.

JLEFT

```
JLEFT { ALL }
      { n   }
      { n m }
```

This command is used to align text left-justified.

`ALL`	Aligns the text of all lines.
n	Aligns the text from line *n* to the last line.
n m	Aligns the text from line *n* to line *m*.

The `JLEFT` command applies only within the horizontal boundaries as set with the `BNDS` command.

For left justification, you can also use the line commands `LJ` and `LJJ`.

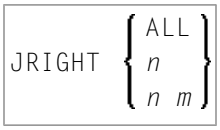
See also the JRIGHT command.

Example:

```
BNDS 10;JLEFT 15 20
```

The text between column 10 and the rightmost column of your screen in lines 15 to 20 is left-aligned to column 10.

JRIGHT



This command is used to align text right-justified.

ALL	Aligns the text of all lines.
n	Aligns the text from line *n* to the last line.
n m	Aligns the text from line *n* to line *m*.

The JRIGHT command applies only within the horizontal boundaries as set with the BNDS command.

For right justification, you can also use the line commands RJ and RJJ.

See also the JLEFT command.

Example 1:

```
BNDS 4 40;JRIGHT 6 18
```

The text between columns 4 to 40 in lines 6 to 18 is right-aligned to column 40.

Example 2:

```
BNDS 10;JRIGHT 15
```

The text to the right of column 10 from line 15 to the last line is right-aligned to the rightmost column of your editing screen.

JUSTIFY

```
          ┌ LEFT  ┐
JUSTIFY   { RIGHT }
          └ BOTH  ┘
```

This command is used to set the justification mode for the line commands `TO` and `TOO`.

`TO` and `TOO` are used to join source lines with subsequent lines. Both commands apply only within the horizontal boundaries as set with the `BNDS` command.

`LEFT`	Aligns text to the left boundary.
`RIGHT`	Aligns text to the right boundary.
`BOTH`	Aligns text to both boundaries.

Example:

You set the horizontal boundaries to columns 10 and 60 and activate left justification with the following command:

```
BNDS 10 60;JUSTIFY LEFT
```

When you then mark a line with a `TO` line command (or a block of lines with two `TOO` line commands), the text between columns 10 and 60 in the marked line(s) is left-aligned to column 10.

LABEL

```
LABEL .label
```

This command is used to mark the current line (that is, the line which is currently at the top of the editing area) with the specified *label*.

The *label* is a string of 1 to 4 alphanumerical characters.

Example:

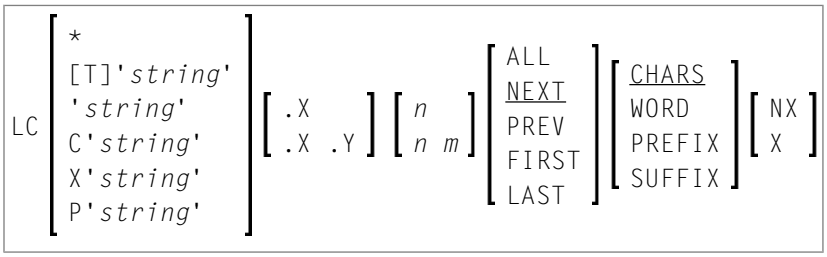
Use the following command to label the current line with `.X`:

```
LABEL .X
```

You can also mark a block of lines with two labels. For example, to mark a block with labels `.X` and `.Y`, you first mark the current line (assuming it is the first line of the block to be marked) with `.X` as shown in the example above; then you scroll until the last line of the block is the current line; then you issue the `LABEL .Y` command to mark that line with `.Y`.

You can also use the *.label* line command to mark a line with a label.

LC



This command is used to change one or more lines to lower case.

You can specify that only lines which contain a specified character *string* are to be changed to lower case. If you want an apostrophe to be part of the *string*, you must write it as two apostrophes.

You can specify the *string* as described in the following section.

`T'`*string*`'`	Changes lines which contain the *string* irrespective of lower case or upper case. This is the default.
`'`*string*`'`	Same as `T'`*string*`'`.
`C'`*string*`'`	Changes lines which contain the *string* exactly as specified.
`X'`*string*`'`	Changes lines which contain the string that corresponds to the specified hexadecimal character *string*.
`P'`*string*`'`	Changes lines which contains a *string* that includes the following wildcard characters:

=	any character
§	alphabetic character
#	numeric character
$	special character
^	non-blank character
-	non-numeric character
<	lower-case letter

	>	upper-case letter
*	Changes lines which contain the *string* specified in a previous command, for example, LC, DELETE or EXCLUDE.	
.X	See ***Line Specifications*** for an explanation.	
.X .Y		
n *n m*	See ***Column Specifications*** for an explanation.	
ALL NEXT PREV FIRST LAST	See ***Direction of Operation*** for an explanation.	
CHARS WORD PREFIX SUFFIX	See ***Special Occurrences*** for an explanation.	
NX X	See ***Displayed or Non-Displayed Lines*** for an explanation.	

If you enter the LC command without any parameters, the current line is changed to lower case.

Example:

```
LC C'NAME' .X .Y ALL PREFIX NX
```

This command changes to lower case all displayed lines within the block of lines labeled with .X and .Y if they contain the string NAME (in upper case as entered here) as prefix to a word.

LIMIT

```
LIMIT [n]
```

With this command, you specify the maximum number of lines to be searched with a `FIND` or `RFIND` command. The parameter *n* is the number of lines to be searched.

LOCATE

```
[LOCATE] { 0       }
         { n       }
         { .label  }
```

This command is used to scroll a specific line to the top of the editing area (that is, make it the current line).

The command provides the following options:

0	Makes the first line of the source code current.
n	Makes line *n* current.
.label	Makes the line labeled with *.label* current.

Examples:

```
LOC 32
```

Places line number 32 at the top of the editing area.

```
32
```

Same as above.

```
LOC .X
```

Places the line labeled with `.X` at the top of the editing area.

LOG

This command activates or deactivates the internal log file.

The log file is a history of all modifications made in the editor since session begin. When the log file is active, each time you press ENTER, the changes made since the previous ENTER are recorded in the log file. When using the UNDO command, you can consecutively back out changes made since the beginning of the editor session.

Important: All entries in the log file are cleared when you clear the source work area or read in the source of another Natural object, or when you terminate the program editor session.

MASK

This command activates or deactivates the mask function. When the mask function is active, each time you insert a line in the editor, a predefined line of text is entered instead of a blank line. The mask line is defined using the MASK line command, described in the following paragraph. The mask function is useful when you must write several lines of code which are identical or very similar.

To define and use a mask line

1 Enter the MASK line command in any source line and press ENTER.

A blank line indicated by =mask> appears above the line in which you entered the command.

2 In the blank line, type in the text you want to define as a mask line and press ENTER.

The mask line is now available for the current source until you update the mask with a new mask line or until you deactivate the mask function.

3 Enter the MASK ON editor command.

The mask function is activated. The defined mask line now appears in all lines added through a line insert operation.

4 Enter an insert line command, for example: I2

Two new lines are inserted into the source with the text of the mask line. The text of a mask line appears in all lines added with an insert command.

5 Modify the text in the new lines. If you do not modify the text, any inserted line is deleted the next time you press ENTER.

The `MASK OFF` command deactivates the mask function but does not delete the contents of the mask line.

MWINDOW

```
MWINDOW [ n   ]
        [ n m ]
```

This command is used to move a data window according to the command parameters.

n	The number of the line in which the data window is to be inserted.
m	The number of the column in which the data window is to be inserted.

See also *Copying and Moving Text with a Data Window*.

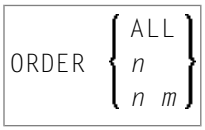
NEXT

```
NEXT [ *           ]
     [ object-name ]
```

This command is used to display the *next* parallel editing session, assuming two or more editing sessions are running concurrently and if the profile parameter `EDTRB` (see the *Parameter Reference* documentation) is set. The following command parameters are optional:

*	Displays a list of all concurrently running sessions for selection.
object-name	Calls directly by name a concurrently running editing session.

ORDER



This command is used to join source lines.

ALL	Joins all lines.
n	Joins the lines from line *n* to the last line.
n m	Joins lines from line *n* to line *m*.

The ORDER command applies only within the horizontal boundaries as set with the BNDS command.

Within the set boundaries, the lines are concatenated and are filled to the greatest possible extent; words that do not fit into one line are automatically placed in the next line.

To join source lines, you can also use the line commands TF, TO and TOO.

POINT

This command places the line marked by the line command NZ at the top of the editing area.

POWER

This command switches the editor to insert mode. You are presented with a blank screen into which you can enter one or more lines of text. After entry, press ENTER and the text is inserted into the first line of the editing area.

PROF

```
PROF [n]
```

This command displays your editor profile at the top of the editor screen.

With *n* you specify additional lines to be displayed. Possible values for *n* are:

6	Displays your editor profile and all tab positions (as specified by the TABS command).
7	Displays same as 6, plus the mask line (as specified by the MASK command).
8	Displays same as 7, plus boundaries (as specified by the BNDS command).
9	Displays same as 8, plus column numbers (as specified by the COLS command).

PROFILE

```
PROFILE
```

This command invokes the editor profile facility. It enables you to modify your editor defaults for current and future sessions. The editor profile facility is described in more detail in section *Modifying Profile Settings for Permanent Use*.

PROTECT

```
PROTECT [INS]
        [ON ]
        [OFF]
```

This command is used to protect the **prefix area**.

INS	Protects the **prefix area** of lines added using the insert line command.
ON	Activates protection.
OFF	Deactivates protection.

RCHANGE

```
RCHANGE
```

This command repeats the last CHANGE command.

RESET

```
RESET
```

This command resets all pending editor and line commands and deletes all line labels.

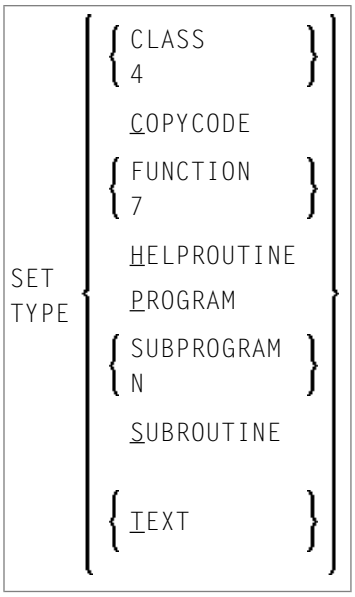
RFIND

```
RFIND
```

Alternative PF key: PF5

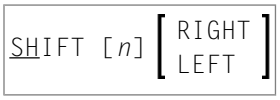
This command repeats the last FIND command.

SET TYPE



This command changes the type of the object currently in the source work area.

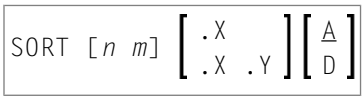
SHIFT



This command shifts a block of lines between the `.X` and `.Y` labels to the right or left by *n* columns (or up to the last non-blank character). The default shift is five columns to the right.

n	The number of columns the lines are to be shifted (default value is 5).
RIGHT	Shifts block of lines to the right (default).
LEFT	Shifts block of lines to the left.

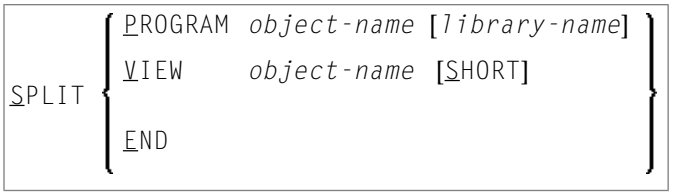
SORT



The SORT command sorts lines in the editor in ascending or descending [a] order. If you enter SORT without any parameters, the command sorts all text in the object in ascending order.

n m	Sorts from column *n* to column *m*.
.X	Sorts from the line labeled with .X to the end of the source.
.X .Y	Sorts from the line labeled with .X to the line labeled with .Y (where .X and .Y represent any string of up to four characters).
A	Sorts text in ascending order (A to Z).
D	Sorts text in descending order (Z to A).

SPLIT



This command sets **split-screen mode** and displays the source of another object on the editor screen.

PROGRAM	Displays a program, subprogram, subroutine, helproutine, data area (global, local, parameter), copycode, text, map, class or function.
VIEW	Displays a view (DDM, as defined in Predict or SYSDDM). If SHORT is specified, the view is listed in short form (that is, only the Adabas short names and corresponding Natural field names are displayed) without any field headers or field edit mask information.
END	Terminates split-screen mode.

With PROGRAM or VIEW, an asterisk (*) can be used for *object-name* to display a list of all available objects. If the an asterisk (*) is preceded by one or more characters, only those objects whose names begin with these characters are displayed.

For an example of using SPLIT, see *To display and copy definitions with SPLIT*.

SWAP

```
SWAP
```

The SWAP command toggles between two objects in **split-screen mode** (see the relevant section). During this operation, the cursor switches from one object to the other.

TABS

```
      ┌ ON [tab-character]                                    ┐
      │ OFF                                                   │
TABS  │ ┌ LEFT    ┐                                           │
      │ │ RIGHT   │ [tab-character] [column...]               │
      │ │ DECIMAL │                                           │
      └ └         ┘                                           ┘
```

This command is used to control tabulator settings.

You can enable or disable logical or physical tabulation using the command TABS ON or TABS OFF. Tabulation is also enabled by any command that changes a tabulation setting.

For example, the following command enables logical tabulation with the ampersand sign (&) as the logical tabulation character:

```
TABS &
```

You set tab positions using the TABS command. For example, the following command sets tabs in columns 10, 20 and 30:

```
TABS 10 20 30
```

You can enter text and automatically move it to a specific tab position by preceding it with a logical tabulation character. One tabulation character moves the text to the next tab position, two tabulation characters move the text to the second tab position, and so on.

To display the current TABS command settings, issue the PROF command.

To display the current tab positions, issue the TABS line command.

Apart from tab positions, you can specify the following parameters with the TABS command:

LEFT	Places the text left-justified at the tab position.
RIGHT	Places the text right-justified at the tab position.
DECIMAL	Places the text so that the decimal point in the text is at the tab position.

Multiple tab characters are possible to tabulate text in a specific column: issue the TABS line command and type over each asterisk (*) marking the tab positions with another special character. Any input preceded by any of these special characters is tabulated in the corresponding column. You can type an L (for LEFT), an R (for RIGHT) or a D (for DECIMAL) after each tabulation character to specify placement of the text for the tab position.

In the following examples of tabulation, the ampersand (&) is assumed to be the tabulation character; the COLS line command has been issued to display column positions.

Example 1 - Tab Positions

The command:

```
TABS 10 20 40 LEFT
```

activates logical tabs with tabulation columns 10, 20, and 40 with left justification. After you press ENTER, the input text line

```
&abc &def &ghi
```

is displayed as follows:

```
=cols> ----+----1----+----2----+----3----+----4----+----5----+----6
               abc       def                ghi
```

Example 2 - TABS RIGHT

The command

```
TABS RIGHT
```

activates logical tabs with right justification. After you press ENTER, the input text line

```
&abc &def &ghi
```

is displayed as follows:

```
=cols> ----+----1----+----2----+----3----+----4----+----5----+----6
              abc       def                 ghi
```

Example 3 - TABS DECIMAL

The command

```
TABS DECIMAL
```

activates logical tabs with justification of the decimal point in the tab position. After you press ENTER, the input text line

```
&15.27$ &16.3 EUR &13 IS
```

is displayed as follows:

```
=cols> ----+----1----+----2----+----3----+----4----+----5----+----6
              15.27$    16.3 EUR             13 IS
```

Example 4 - Mixed Justification

Issue the following command:

```
TABS 10 20 30 40 50
```

Then issue the `TABS` line command. This displays the current tab positions as follows:

```
=tabs>          *         *         *         *         *
```

Type an `L`, `R` or `D` next to each tab position as required (unmarked tab positions assume the value of the last `TAB` command):

```
=tabs>          *R        *D        *D        *D        *L
```

After you press ENTER, the input text line

```
&start &0.01 &0.02 &0.03 &end
```

is displayed as follows:

```
=cols> ----+----1----+----2----+----3----+----4----+----5----+----6
            start        0.01      0.02      0.03       end
```

Example 5 - Multiple Tab Symbols

Replace the asterisks in the `=tabs>` line by other special characters and specify left justification for each one as follows:

```
=tabs>           ]L        &L        #L        $L        =L
```

After you press ENTER, the input text line

```
=first$second#third&fourth]fifth
```

is displayed as follows:

```
=cols> ----+----1----+----2----+----3----+----4----+----5----+----6
                                                      first
                                          second
                                    third
                           fourth
                  fifth
```

Example 6 - Using a Blank as Tabulation Symbol

Issue the command

```
TABS ' '
```

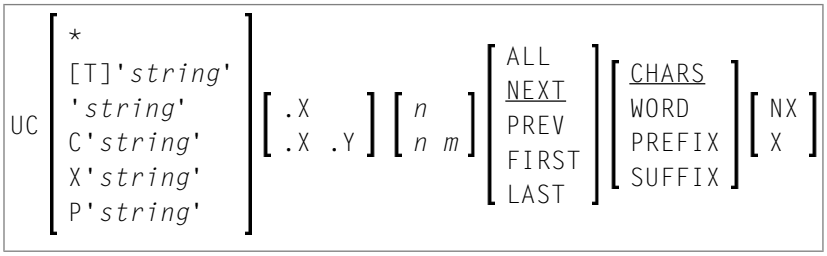
which activates tabulation with one blank as the tabulation character. This means that words separated by one blank are tabulated. After you press ENTER, the input text line

```
this is a blank tabulation
```

is displayed as follows:

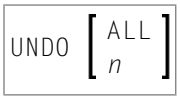
```
=cols> ----+----1----+----2----+----3----+----4----+----5----+----6
               this      is        a         blank     tabulation
```

UC



The UC command converts one or more lines to upper case. It uses the same parameters as the LC command. If you enter the UC command without parameters, it changes the current line to upper case.

UNDO



Alternative PF key: PF11

If the log file is active (see the LOG command), the UNDO command backs out all changes made since the last time you pressed ENTER. Repeated use of the UNDO command backs out consecutive changes in reverse order. You can thus back out all changes one by one until you restore the member to its original status at session begin.

Important: All entries in the log file are cleared when you clear the source work area or read in the source of another Natural object, or when you terminate the program editor session.

You can specify the following parameters with the `UNDO` command:

`ALL`	Backs out all modifications made in the current editor session.
n	Backs out the last *n* modifications.

WINDOW

```
        ⎧ line1 line2                    ⎫
WINDOW  ⎨ line1 line2 column1            ⎬
        ⎩ line1 line2 column1 column2    ⎭
```

This command is used to define a data window to be copied or moved. The starting line and column and the end line and column of the window are specified in the command parameters. At least *line1* and *line2* are required.

line1 line2	Defines a window starting at column 1 of *line1* and ending in the last column of *line2*.
line1 line2 column1	Defines a window starting at *column1* of *line1* and ending at the last column of *line2*.
line1 line2 column1 column2	Defines a window starting at *column1* of *line1* and ending at *column2* of *line2*.

See also *Copying and Moving Text with a Data Window*.

X

```
X
```

This command places the line marked by the line command `.X` at the top of the editing area.

XSWAP

```
XSWAP
```

This command is used to exchange displayed lines with excluded lines. Lines are excluded using the EXCLUDE command.

Y

```
Y
```

This command places the line marked by the line command .Y at the top of the editing area.

Common Command Options

There are some options which are available with several editor commands. These options are described in the following section.

- Line Specifications
- Column Specifications
- Displayed or Non-Displayed Lines
- Direction of Operation
- Special Occurrences

Line Specifications

The following options can be used to restrict the effect of an editor command to a line or a block of lines labeled with the .X and .Y line commands:

.X	The editor command affects only the line labeled with .X. Exception: SORT **editor command**.
.X .Y	The editor command affects only the block of lines from the line labeled with .X to the line labeled with .Y. If you use this option, you must also supply the parameter ALL with the command.

Note: .X and .Y can also be any label of 1 to 4 alphabetic characters (see also the LABEL editor command and the .label line command).

Column Specifications

The following options can be used to restrict the effect of an editor command to a certain range of columns. The column numbers refer to the actual source-code columns; the line numbers preceding the source code are not counted. So, if you specify column 1 with a command, this may physically be the 8th column of your screen, but it is in fact the 1st column of the source code you are editing.

n	The command affects only lines in which the specified string begins in column *n* (that is, the first character of the string must be in column *n*).
n m	The command affects only lines in which the specified string occurs anywhere between columns *n* and *m*.

Displayed or Non-Displayed Lines

The following options can be used to specify that only excluded or only included lines are to be affected by an editor command:

NX	The command affects only non-excluded lines; that is, lines which are currently being displayed.
X	The command affects only excluded lines; that is, lines which are currently *not* being displayed as specified by the EXCLUDE command. An excluded line remains excluded from display if an editor command function is performed on it.

Direction of Operation

The following options can be used to specify the direction in which an editor command is to operate:

NEXT	The command affects the next line (starting from the cursor position) in which the specified *string* occurs. This is the default setting.
PREV	The command affects the line that contains the previous occurrence of the specified *string*.
FIRST	The command affects the first line in which the specified *string* occurs.
LAST	The command affects the last line in which the specified *string* occurs.
ALL	The command affects all lines in which the specified *string* occurs.

Special Occurrences

The following options can be used to specify whether only special occurrences of the specified *string* are to be affected by an editor command:

CHARS	The command affects any line in which the specified *string* occurs. This is the default setting.
WORD	The command affects only those lines in which the specified *string* forms a word.
PREFIX	The command affects only those lines in which the specified *string* is the beginning of a word.
SUFFIX	The command affects only those lines in which the specified *string* is the end of a word.

12 Editor Commands for Scrolling

This section describes the editor commands and alternative PF keys (if available) that can be used to scroll in the source code currently contained in the editing area.

You enter an editor command in the command line of the editor screen. Depending on the configuration of your installation, an editor command can be entered in lower case. In this section, however, all commands are shown in upper case to distinguish them as commands.

In the following table, an underlined text portion represents an acceptable command abbreviation.

Command	Function
BOTTOM or ++	Scrolls to the end of the source.
TOP or --	Scrolls to the beginning of the source.
DOWN or + or ENTER key or PF8	Scrolls forwards by the amount specified by the **scroll mode**.
DOWN *n*	Scrolls forwards by *n* lines.
+*n*	Scrolls forwards by *n* lines.

Command	Function
UP or - or PF7	Scrolls backwards by the amount specified by the **scroll mode**.
UP *n*	Scrolls backwards by *n* lines.
-*n*	Scrolls backwards by *n* lines.
LEFT	Scrolls to the left by the number of columns that fit on the screen.
LEFT *n*	Scrolls to the left by *n* columns.
LEFT MAX	Scrolls the maximum number of columns to the left edge of the screen.
RIGHT	Scrolls to the right by the number of columns that fit on the screen.
RIGHT *n*	Scrolls to the right by *n* columns.
RIGHT MAX	Scrolls the maximum number of columns to the right edge of the screen.
FIX *n*	Specifies the number of columns *n*, starting with column 1, to remain in display when scrolling to the right.

13 Line Commands

This section describes the line commands provided by the program editor. A line command always applies to the source line in which you enter it or to a block of lines marked by multiple line commands.

Depending on the configuration of your installation, line commands can be entered in lower case. In this section, however, all commands are shown in upper case to distinguish them as commands.

To execute a line command

1 In the leftmost column of the **prefix area**, next to the required source line, type a line command over the characters contained in these columns, and press ENTER.

It is not possible to enter a line command if either of the following applies:

- Insert mode is active (indicated by a series of apostrophes (' ' ' ' ' '). In this case, toggle the insert mode to non-insert mode.
- The **prefix area** is protected. In this case, deactivate protection with the `PROTECT` editor command.

Or:

In the command line, enter a line command preceded by a colon (:), place the cursor in the line to which you want to apply the command, and press ENTER. For example:

In the command line, enter `:I`, place the cursor in the first line and press ENTER. A new line is then inserted below the first line.

2 If required, enter the `RESET` editor command to reset a pending line command and delete all line labels.

Command	Function
)	Moves this line right by two columns.
)*n*	Moves this line right by *n* columns, irrespective of any other text in the line: you may lose text in the moved line.
))*n*	Marks the first line of a block of lines to be moved right by *n* columns. A second))*n* is required to mark the last line of the block. The function is performed when the second))*n* is specified. The block is moved regardless of any other text in the block: you may lose text in the moved block.
(	Moves this line left by two columns.
(*n*	Moves this line left by *n* columns regardless of any other text: you may lose text in the moved lines.
((*n*	Marks the first line of a block of lines to be moved left by *n* columns. A second ((*n* is required to mark the last line of the block. The function is performed when the second ((*n* is specified.
<	Moves the text in this line left by two columns.
>	Moves the text in this line right by two columns.
>*n*	Moves the text in this line right by *n* columns or up to last non-blank character: no text is lost.
>>*n*	Marks the first line in a block of lines to be moved to the right by *n* columns or up to last non-blank character. A second >> is required to mark the last line of the block. The function is performed when the second >> is specified.
<*n*	Moves the text in this line left by *n* columns or up to first non-blank character.
<<*n*	Marks the first line in a block of lines to be moved to the left by *n* columns or up to first non-blank character. A second << is required to mark the last line of the block. The function is performed when the second << is specified.
A	Marks the target line for a move (M, M*n*, MM) or copy (C, C*n*, CC) line command. The moved or copied line(s) are inserted *after* this line.
B	Marks the target line for a move (M, M*n*, MM) or copy (C, C*n*, CC) line command. The moved or copied line(s) are inserted *before* this line.
BNDS	Displays the boundary positions in this line. See also the BNDS editor command.
C	Copies this line to the position indicated by an A, B or O line command.
C*n*	Copies this line and the next *n-1* lines to the position indicated by an A, B or O line command.
CC	Marks the first line of a block of lines to be copied. A second CC command is required to mark the last line of the block to be copied. The function is performed when the second CC is specified. The lines are copied to the position indicated by an A, B or O line command.
CX	Copies the line labeled with .X. Inserts text after this line.
CY	Copies the line labeled with .Y. Inserts text after this line.
CX-Y	Copies the block of lines from the line labeled with .X to the line labeled with .Y. Inserts text after this line.
COLS	Displays the column positions in this line.

Command	Function
D	Deletes this line.
D*n*	Deletes the current line and the next *n-1* lines.
DD	Marks the first line of a block to be deleted. A second DD command is required to mark the last line of the block to be deleted. The deletion is performed when the second DD is entered.
DX	Deletes the line labeled with .X.
DY	Deletes the line labeled with .Y.
DX-Y	Deletes the block of lines from the line labeled with .X to the line labeled with .Y.
F	Includes the first excluded line.
F*n*	Includes the first *n* excluded lines.
I	Inserts one line. The editor switches to insert mode indicated by ''''''. This means if you type text or enter a blank in the new line and press ENTER, a new line is automatically inserted and the cursor placed in it. If you enter no new text in an inserted line and press ENTER, the editor leaves insert mode and the blank line is deleted. You can also fill an inserted line with a predefined content (see the MASK editor command).
I*n*	Inserts *n* lines. You can type text in the new lines. When you press ENTER, unused lines are deleted but one blank line remains with the cursor in it (editor stays in insert mode).
.I(*obj*,*pos*,*n*)	Inserts any object contained in the current library in the current system file into the editor screen. **Note:** This command is entered in the first position of a source line, not in the prefix area. The *pos* entry indicates the line at which the include operation is to begin. For example, setting *pos* to 20 causes the insertion to begin with the 20th line in the source. The *n* entry indicates the number of lines to be inserted. If the object is a Natural map, an INPUT USING MAP statement (see *INPUT Syntax 2 - Using Predefined Map Layout* in the *Statements* documentation) with all defined variables is automatically included in the current line. If the object is a data area, the entire data area is included, except comment lines. Only data areas that have been saved and cataloged with the STOW system command, can be included in the source contained in the editing area. If the object is an adapter, a PROCESS PAGE USING (see *Syntax 2 - PROCESS PAGE USING* in the *Statements* documentation) with all defined variables is automatically included in the current line.
.I(*)	Invokes a selection list of objects in the current library.
J	Joins the next line with this one. You can specify how many of the characters of the following line are to be joined by placing the cursor at the point in the line where it is to

Command	Function
	be separated and pressing ENTER. To join the entire line, place the cursor outside the line to be joined. This command is identical to the TJ command.
L*n*	Includes the last *n* excluded lines.
LC	Changes the letters in this line to lower case.
LC*n*	Changes the letters in this line and in the next *n-1* lines to lower case.
LCC	Marks the first line of a block of lines in which to change all letters to lower case. A second LCC is required to mark the last line in the block. The function is performed when the second LCC is specified.
LJ	Justifies the text within the set boundaries in this line with the left boundary.
LJJ	Marks the first line of a block of lines within the set boundaries to be justified to the left. A second LJJ command is required to mark the last line of the block to be justified. The justification is performed after the second LJJ command has been issued.
M	Moves this line to the position indicated by an A, B or O line command.
M*n*	Moves this line and the next *n-1* lines to the position indicated by an A, B or O line command.
MM	Marks the first line of a block of lines to be moved. A second MM command is required to mark the last line of the block to be moved. The function is performed when the second MM is specified. The lines are moved to the position indicated by an A, B or O line command.
MASK	Inserts a blank line in the editing area in which you can define a mask. This line is inserted whenever the I*n* line command is used to create one or more new lines. See also the MASK editor command and ***To define and use a mask line***.
MX	Moves the line labeled with .X. Inserts it after this line.
MY	Moves the line labeled with .Y. Inserts it after this line.
MX-Y	Moves the block of lines from the line labeled with .X to the line labeled with .Y. Inserts it after this line.
N	Modifications made in this line do not take effect when ENTER is pressed.
NZ	Places this line at the top of the editing area when the POINT editor command is issued.
O	Marks this line as the target line for a move (M, M*n*, MM) or copy (C, C*n* CC) line command. The moved or copied line(s) are merged with this line, that is, blank characters in the line are overlaid.
O*n*	Marks this line and the next *n-1* lines as the target lines for a move (M, M*n*, MM) or copy (C, C*n* CC) line command. The moved or copied lines are merged with these lines, that is, blank characters in the lines are overlaid.
OO	Marks the first line of a block of target lines for a move (M, M*n*, MM) or copy (C, C*n* CC) line command. A second OO command is required to mark the last line of the block of target lines. The moved or copied line(s) are merged with these lines, that is, blank characters in the lines are overlaid.
R	Repeats this line once.
R*n*	Repeats this line *n* times.

Command	Function
RR	Marks the first line of a block of lines to be repeated. A second RR command is required to mark the last line of the block to be repeated. The repeat operation is performed when the second RR is entered.
RR*n*	Repeats the block of lines *n* times.
RJ	Justifies the text within the set boundaries in this line with the right boundary.
RJJ	Marks the first line of a block of lines within the set boundaries to be justified to the right. A second RJJ command is required to mark the last line of the block to be justified. The justification is performed when the second RJJ is entered.
S	Splits this line into two lines beginning at the cursor position. Type in the command, move the cursor to the position where the line is to be split, and press ENTER.
T	Scrolls the source to make the marked line the top line.
TABS	Displays the tab positions in this line. See also the TABS editor command.
TC	Centers the text within the set boundaries in this line.
TCC	Marks the first line of a block of lines within the set boundaries to be centered. A second TCC command is required to mark the last line of the block of the centered. The centering is performed when the second TCC command is entered.
TE	Excludes all lines below this line from display and switches to insert mode. The excluded lines appear again when you press ENTER.
TF	Joins this line with the following lines up to the next blank line. The bounds settings can be used to restrict the columns affected (see the BNDS editor command).
TF*n*	This line command can be entered with a numerical value specifying the right boundary. For example, the line command TF5 orders text with column 5.
TI	Inverts the sequence of all characters in the current line and within the set boundaries.
TII	Marks the first line of a block of lines to be inverted within set boundaries. Requires a second TII to mark the last line of the block. The function is performed when the second TII is specified.
TJ	Joins the next line with this one. Same as the J line command.
TO	Joins this line with the next one.
TOO	Marks the first line of a block of lines within the set boundaries to be joined. A second TOO command is required to mark the last line of the block to be joined. The function is performed when the second TOO is entered.
TS	Splits this line into two lines at the cursor position; a blank line is also automatically inserted, but deleted if unused (identical to the S line command).
UC	Changes all letters in this line to upper case.
UC*n*	Changes all letters in this line and in the next *n-1* lines to upper case.
UCC	Marks the first line of a block of lines in which to change all letters to upper case. A second UCC is required to mark the last line of the block. The function is performed when the second UCC is specified.
W	Opens a window with one line.
W*n*	Opens a window with *n* lines.

Command	Function
WC	Copies the data window. The cursor position marks the column at which this line is to be split to insert the copied text.
WC*n*	Splits this line in column *n*, and copies the text between the two parts of the line.
WE	Marks the end of the data window. Works in the same way as WS. If the window is to start and end in the same line, replace the WS command by the WE command. The editor acknowledges the set window with message WW in the prefix area. See also ***Copying and Moving Text with a Data Window***.
WM	Moves the data window. Works in the same way as WC, but the original text is deleted after the copy operation. See also ***Copying and Moving Text with a Data Window***.
WM*n*	Splits this line in column *n*, and moves the text between the two parts of the line.
WS	Marks the start of the data window. The cursor position marks the column from which text is read. If the cursor is not in the line for which the command is entered, column 1 is taken. See also ***Copying and Moving Text with a Data Window***.
WS*n*	Specifies that the data window starts in column *n* of this line.
X	Excludes this line from display.
X*n*	Excludes this line and the following *n* lines from display.
XX	Marks the first line of the block of lines to be excluded from display. A second XX is required to mark the second line of the block. The function is performed when the second XX is specified.
.X	Marks this line with .X. See also .*label* below.
.Y	Marks this line with .Y. See also .*label* below.
.*label*	Marks this line with .*label* where *label* is a string of 1 to 4 alphabetic characters. For example: line command .X names this line .X, and .Y names this line .Y. See also the LABEL editor command.

14 Editor Profile

Each user has an editor profile with parameters which can be set according to individual needs. The first time you invoke the editor, it uses the default settings determined by your administrator.

The options provided to display and modify your default editor profile settings are described in the following section.

Displaying and Hiding Profile Settings

To display or hide the current editor profile settings

1 In the command line of the editor, enter the following:

```
PROF
```

(See also the PROF editor command.)

The following lines appear at the top of the editing area:

```
 >> -----------Columns 001 072 <<  Program SAGDEMO  Lines  19     User SAG    ↩

Command ===>                                          Mode   Struct Lib  SAGTEST
****** ***************************** top of data ******************************
=prof> date: 30/01/08 16:07:57 user: MMO      init size: 14 size: 14
=prof> var   - 250,..recovery off (100 0)...autosave off... empty line off
=prof> mask off.caps on .hex off  nulls on  std.autoren on  std auto order off
=prof> log off .mso on .fix off .escape on  + .  tabs off
=prof> advance on .protect off.limit off
```

2 You can hide the display of the profile settings by entering the following in the command line of the editor:

```
RESET
```

The profile settings disappear. The current profile settings are retained.

The individual items of the current editor profile and the editor commands that can be used to temporarily change an item (if modifiable) are described in the following section.

Modifying Profile Settings for Temporary Use

You can modify your default editor profile settings for the duration of the current editor session or until you change the settings again.

To change your editor profile settings for the current session

- For the profile item you want to change, use the appropriate editor command listed in the following table. For example, enter `CAPS OFF` if you want the text to be translated into upper case.

  The vertical bar used in the syntax of the editor commands separates alternatives. For details on the editor commands listed, refer to *Editor Commands*.

Profile Item	Description	Editor Command
`date`	Current date and time. Non-modifiable item.	None
`user`	Current logon user. Non-modifiable item.	None
`init size`	Number of lines in the source code when the editor was invoked. Non-modifiable item.	None
`size`	Current number of lines in the source code, excluding information lines (for example profile lines and message lines). Non-modifiable item.	None
`var`	Current line length. Non-modifiable item.	None
`autosave`	Activates or deactivates automatic save when the `EXIT` editor command is issued.	`AUTOSAVE`
`empty line`	Specifies whether lines containing only blank characters are to be deleted automatically.	`EMPTY ON|OFF`
`mask`	Activates or deactivates the mask line function.	`MASK ON|OFF`
`caps`	Specifies whether text is to be translated into upper case.	`CAPS ON|OFF|PGM`
`hex`	Specifies whether characters are to be displayed in hexadecimal format.	`HEX ON|OFF`

Profile Item	Description	Editor Command
autoren	Specifies that automatic renumbering of source lines is activated. Non-modifiable item.	None
auto order	Automatically justifies text within defined boundaries.	AORDER ON|OFF
log	Enables or disables the log file. When enabled, the UNDO editor command can be used to backout last changes.	LOG ON|OFF
mso	Indicates that multiple-session operations are allowed. A multiple-session operation is an operation in which data is exchanged between two editing sessions, for example, when copying text from one object to another in **split-screen mode**. Non-modifiable item.	None
fix	Specifies whether a fixed number of columns is displayed and how many columns are to be fixed.	FIX ON|OFF *n*
tabs	Activates or deactivates tabulation.	TABS ON|OFF
advance	Specifies whether the cursor moves to the next line automatically after a line update.	ADVANCE ON|OFF|PAGE
protect	Specifies protection of line numbers.	PROTECT ON|OFF|INS
limit	Specifies the maximum number of lines to be searched by a FIND or RFIND editor command.	LIMIT *n*

Modifying Profile Settings for Permanent Use

You can use the profile facility of the program editor to change your profile settings not only for the current session as described earlier, but also for future editor sessions. Settings modified with the editor profile facility are valid for each new Natural session or until you change them again. These settings can be overridden for the duration of the current session by using the editor commands described in *Modifying Profile Settings for Temporary Use*.

To modify your editor profile settings for permanent use

- In the command line of the program editor, enter the following:

  ```
  PROFILE
  ```

  The **Main Menu** of the program editor profile facility appears with the following options:

Option	Function
Save	Saves the current profile.
Modify	Invokes input screens for modifying the default settings for PA/PF keys and editor commands: see *Modifying Editor Default Settings*.
Read	Reads the editor profile of another user which you specify in the **Profile Name** field. The settings from the external profile can then be saved under your profile name.
Technical Info	Displays information on the object currently being edited and the system environment: see *Displaying Technical Information*.

This section covers the following topics:

- Editor Profile Commands
- Modifying Editor Default Settings

Editor Profile Commands

The direct commands and alternative PF keys available in the editor profile facility are described in the following section. You enter a direct command in the command line (**Command ===>**) at the bottom of the editor profile screen.

Command	PF Key	Function
CANCEL	PF12	Cancels the current modification and displays the previous screens. Any modifications made to the profile have no effect for the current session.
EXIT	PF3	Leaves the current screen and either returns to the previous screen or invokes the **EXIT Function** prompt window (see also *Exit Function*).
FLIP	None	Toggles between PF1 - PF12 and PF13 - PF24.
READ	PF6	Reads the profile parameters for the user ID currently contained in the **Profile Name** field. Any modifications made so far, but not yet saved, are overwritten (valid only for the **Main Menu**).
SAVE	PF5	Saves all currently valid profile parameters both for the current session and on the database. However, it does *not* leave the current function (valid only for the **Main Menu**).

Modifying Editor Default Settings

If you select **Modify** from the **Main Menu** or press PF4, the **Modify Defaults** screen is displayed with the following options:

Option	Function
PA/PF-Keys	Modifies the PA/PF-key assignments: see *Displaying and Modifying PA/PF-Key Assignments*.
Commands	Modifies the command defaults: see *Modifying Command Defaults*.
Find	Modifies the parameter settings of the FIND command: see *Modifying FIND Command Defaults*.
General	Enables or disables the **EXIT Function** prompt window and modifies the number of lines displayed before error line: see *Modifying General Defaults*.

Displaying and Modifying PA/PF-Key Assignments

If you select **PA/PF-Keys** from the **Main Menu** or press PF2, the **Modify PF/PA-Keys** screen is displayed with the current PA/PF-key assignments. To modify a PF-key assignment, replace the name of an existing command (editor command, system command or program name) by another command (maximum five characters).

To modify a PA-key or PF-key assignment

- In the input field next to the required PA or PF key, enter the name of a valid editor or system command, or an existing program. You can enter a maximum of five characters.

The default PA/PF-key assignments for the editor are:

PF Key	Command	Function
PF1	HELP	Invokes the Natural online help facility.
PF2	SAVE	Executes the system command SAVE which saves the source code currently in the source work area as a source object in the current library in the current system file. See also *Saving and Cataloging Sources*.
PF3	EXIT	Exits the editor. See also *Exit Function*.
PF4	RUN	Checks and runs the program in the editing area.
PF5	RFIND	Repeats the last FIND command.
PF6	STOW	Executes the system command STOW which saves and catalogs the source code currently in the source work area. See also *Saving and Cataloging Sources*.
PF7	UP or -	Scrolls upwards.
PF8	DOWN or +	Scrolls downwards.
PF9	CHECK	Checks whether the source code currently in the source work area contains any syntax errors.
PF10	HOME	Places the cursor in the command line.

PF Key	Command	Function
PF11	UNDO	Backs out the last change made to the source.
PF12	CANCEL	Exits the editor. See also *Exit Function*.
PF13	PROFILE	Invokes the editor profile facility.
PF14	RESET	Resets all pending editor and line commands and deletes all line labels.
PF15	SWAP	Toggles the cursor between the upper and lower half of a split screen if split-screen mode is set.
PF16	LAST	Executes the system command LAST (see the *System Commands* documentation) which recalls the last command issued and places it in the command line.
PF17	RCHANGE	Repeats the last CHANGE editor command.
PF18	FLIP	Toggles the PF-key display between PF1 - PF12 and PF13 - PF24.
PF19	TOP or --	Scrolls to the beginning of the source.
PF20	BOTTOM or ++	Scrolls to the end of the source.

To modify a PF-key assignment, replace the name of an existing command (editor command, system command or program name) by another command (maximum five characters).

Modifying Command Defaults

If you select **Commands** from the **Main Menu** or press PF5, you can enable or disable selected editor commands and specify default characters for editor commands. The input fields contained on the screen are explained in the table below. For more information on a specific field, enter a question mark (?) in the field and press ENTER.

Field	Description
aorder	Enables or disables the autoorder function.
autosave	Enables or disables the autosave function.
caps	Specifies whether text is to be translated into upper case.
cols	Specifies whether a line is to be displayed that shows the current column positions.
decimal character	Specifies the character used to mark decimal positions in numbers used in tabs. See also *Example 3 - TABS DECIMAL* of the TABS editor command.
empty	Specifies whether lines containing only blank characters are to be deleted automatically.
fix	Specifies whether the number of columns entered in **fixlen** (see below) remain in display when scrolling to the right.

Field	Description
fixlen	Only applies if **fix** (see above) is set to `ON`. Specifies the number of columns starting with column 1 to remain in display when scrolling to the right.
hex	Specifies whether characters are to be displayed in hexadecimal format.
justify	Enables or disables justification or specifies the justification type.
limit	Specifies the maximum number of lines to be searched by `FIND` or `RFIND` editor command.
log	Enables or disables the log file. When enabled, the `UNDO` editor command can be used to back out the last change.
mask line	Enables or disables the mask line function.
message line	Enables or disables message output.
mso	Indicates that multiple-session operations are allowed such as copying between split-screen sessions. Non-modifiable item.
scroll mode	Specifies how scrolling is to be performed (whole page, half page, to cursor).
tabs	Enables or disables tabulation.
tabulator character	Specifies the logical tabulation character used to automatically move input to a specific tab position.

Modifying FIND Command Defaults

If you select **Find** from the **Main Menu** or press PF8, a screen is displayed with the current parameter settings. These settings are used whenever you issue the `FIND` editor command without parameters from the command line of the program editor. You can modify these values in a window before the search is performed.

The following parameters can be modified:

Parameter	Function
FIND string	Specifies the string to be searched for.
search from/to col	Specifies the start and end numbers of the range of columns to be searched.
search from/to label	Specifies the start and end labels of the marked block of lines to be searched.

Modifying General Defaults

If you select **General** from the **Main Menu**, a screen is displayed with the current defaults. You can modify these settings as required. A short description of each setting follows:

Option	Function
Prompt Window for Exit Function	Specifies whether a prompt window is to appear when leaving the program editor with the **exit function** (see the relevant section).
Prompt Window for Cancel Function	Specifies whether a prompt window is to appear when leaving the program editor with the cancel function. See also ***Exit Function***.
Lines displayed before error line	Specifies the number of lines to be displayed at the top of the page preceding the line containing a syntax error. If you specify 0 (zero), the line containing the syntax error is placed in the first line of the editing area.

Displaying Technical Information

If you select **Technical Info** from the **Main Menu** or press PF7, a screen is displayed with information about the object currently being edited and the computing environment. The following items are provided:

Item	Description
User ID	ID of the current user.
Current library	Name of the library to which you are currently logged on.
Current program	Name of the object currently in the editing area.
Current type	Type of object (for example, program) currently in the editing area.
Object type	Internal type code of the current object.
Mode	Programming mode (see also **Mode**).
Operating system	Name of the operating system installed.
Steplib	Name of the library concatenated with the current library.

15 Editor Buffer Pool Settings

It may be necessary to increase the size of the buffer pool above the default setting of 400 KB. See the profile parameter description of EDTBPSIZE in the *Parameter Reference* documentation.

Multiple Editor Sessions

When you open a new editing session from the program editor, the old session is stored in the buffer pool if the profile parameter EDTRB (see the *Parameter Reference* documentation) is set. When you leave the new session with the cancel or exit function (see also ***Exit Function***), the old session is freed from the buffer pool. The NEXT editor command can be used to jump from one active session to another.

If the EDTRB parameter is switched OFF, the ring buffer is not used, and the new session replaces the current session. If the editor profile parameter AUTOSAVE (see *Editor Commands*) is switched on, the contents of the corresponding session are saved to FUSER.

If you intend to use the LOG editor command, we recommend that you increase the size of the buffer pool to 1 MB. Changes are generally stored in the buffer-pool editor.

16 Saving and Cataloging Sources

You can save the source code currently in the source work area as a source object and also as a cataloged object, which are stored in a Natural library in a Natural system file.

For the naming conventions that apply when saving or cataloging sources, refer to *Object Naming Conventions* in the *Using Natural* documentation.

To save source code as a source object

- On the editor screen, enter the system command `SAVE` according to the syntax rules described in the *System Commands* documentation.

  Or:

  On the editor screen, press PF2.

To save source code as a cataloged object

- On the editor screen, enter the system command `CATALOG` according to the syntax rules described in the *System Commands* documentation.

To save source code as a source object *and* a cataloged object

- On the editor screen, enter the system command `STOW` according to the syntax rules described in the *System Commands* documentation.

  Or:

  On the editor screen, press PF6.

Note: You must supply an object name if you save or catalog a new source or if you want to copy the current source. Otherwise, an appropriate message appears or a window prompts you for the name.

IV Data Area Editor

17 Data Area Editor

The Natural data area editor is used to create and modify a data area. A data area is a Natural object of the type global data area (GDA), local data area (LDA) or parameter data area (PDA). For information on using a data area, see *Data Areas* in the *Programming Guide*.

A data area contains data element definitions, such as user-defined variables, constants and database fields referenced with a data view in a data definition module (DDM), which can be used by one or more Natural objects. You can also create copycode from a data area. Note that data views from a DDM cannot be defined in PDAs.

Notes:

1. The data area editor has been disabled in your environment by default. For more information, see ***NaturalONE as the Default Development Environment*** in the *Editors* documentation.
2. It is recommended to stay with NaturalONE for editing Natural sources (incl. data areas). In case the editors are directly used on the Natural environment, data area sources which are stored in the `DEFINE DATA` format, will be automatically converted into the internal format before editing. This format can only be interpreted by the data area editor itself and cannot be parsed by the Natural compiler directly. When the source is saved with the data area editor, the internal format will be generated. In this case, the original source layout (e.g. indentations for comments and `INIT` values) will get lost when the data area will be downloaded to the NaturalONE client again.

Related Topic:

- *Editors in the SPoD Environment* in the *Unicode and Code Page Support* documentation

Invoking the Data Area Editor

You invoke the data area editor with the Natural system command `EDIT` described in the *System Commands* documentation.

To invoke the data area editor for a new data area

- Issue the `EDIT` command specifying the type of data area (`GLOBAL`, `LOCAL` or `PARAMETER`) you want to create.

  For example:

  ```
  EDIT LOCAL
  ```

  An editor screen with an empty editing area appears for an LDA (indicated by `LOCAL` in the top information line of the screen) similar to the example shown in the following instructions.

To invoke the data area editor for an existing data area

- Issue the command `EDIT` specifying the name of a data area that has been stored as a source object in your current Natural environment.

  For example:

  ```
  EDIT LDA1
  ```

  An editor screen similar to the example below appears which contains the source of the local data area `LDA1`:

```
                    Press <ESC> to enter command mode                    ↩
 Mem: LDA1     Lib: SAGTEST  Type: LOCAL      Size:   1662  Line:    0 of:  36
C T    Comment                                                            ↩
  *    *** Top of Data Area ***                                           ↩
  V  1 EMPLOYEES_VIEW                              EMPLOYEES              ↩
     2 PERSONNEL-ID                      A          8                     ↩
  G  2 FULL-NAME                                                         ↩
     3 FIRST-NAME                        A         20                     ↩
     3 MIDDLE-I                          A          1                     ↩
     3 NAME                              A         20                     ↩
     2 MIDDLE-NAME                       A         20                     ↩
     2 MAR-STAT                          A          1                     ↩
     2 SEX                               A          1                     ↩
     2 BIRTH                             D                                ↩
     2 N@BIRTH                           I          2                     ↩
  G  2 FULL-ADDRESS                                                      ↩
  M  3 ADDRESS-LINE                      A         20 (1:191)             ↩
     3 CITY                              A         20                     ↩
     3 ZIP                               A         10                     ↩
     3 POST-CODE                         A         10                     ↩
     3 COUNTRY                           A          3                     ↩
  G  2 TELEPHONE                                                         ↩
  F 1 HELP     F 2 CHOICE   F 3 QUIT     F 4 SAVE     F 5 STOW     F 6 CHECK  ↩
  F 7 READ     F 8 CLEAR    F 9 MEM TYPE F10 GEN      F11 FLD TYPE F12
```

The editor screen is organized in the following sections (from top to bottom):

Section	Explanation
Command line	Used to issue an editor or a system command or execute a program as described in ***Edit Mode and Command Mode***. This line only appears if command mode is set. Otherwise, a message regarding command mode is displayed instead.
Information line	Contains the following information (from left to right) about the data area currently on the editor screen:
	Mem: — The name of the data area or `empty` for a new data area that has not yet been saved as a source object with the `SAVE` or `STOW` system command.
	Lib: — The library to which you are currently logged on.
	Type: — The type of data area: `LOCAL`, `GLOBAL` or `PARAMETER`. The type can be changed by using the editor command `SET TYPE`.
	Size: — The size (number of characters) of the current source.
	Line: — The number of the current (highlighted) source line.
	of: — The total number of lines contained in the source work area.
Editing area	Contains the source of a data area or appears empty for a new data area: see *Using the Editing Area*. You can only scroll in the source or modify the source if edit mode is set: see ***Edit Mode and Command Mode***.
Function-key lines	Contains the function keys (F keys) available to execute an editor command: see ***Editor Commands and Function Keys***.
Message line	Appears if an error occurs, in which case the message line temporarily overwrites the first function-key line with an appropriate error message.

Edit Mode and Command Mode

The data area editor operates in two different modes: edit mode and command mode.

In edit mode, you can scroll up or down in the source of the current data area, use the line commands required for creating or modifying source lines, and press all F keys available on the screen.

In command mode, you can enter or select an editor or a system command or execute a program. In command mode, you cannot modify the source of the data area.

By default, the data area editor is in edit mode when you invoke it.

To toggle between edit and command mode

- Press ESC.

  If command mode is set, the editor command line appears, which is indicated by **Command:** in the top left corner of the editor screen.

  The current mode is kept for the duration of the Natural session.

To enter a direct command or a program name

- In the command line, enter one of the following:
  - Any Natural system command.

    For example: The system command `CHECK` can be used for checking the syntax of source code and `SAVE` for saving source code (see also ***Storing and Cataloging a Data Area***).

    For other system commands related to maintaining and using object sources, see *Managing Applications with Natural Objects* in the *System Commands* documentation.
  - The name of a Natural program to be executed.
  - An editor command. All editor commands available are described in ***Editor Commands and Function Keys***.

To select a direct command from a menu

- In the command line, enter an `M` (menu).

  The following command menus appear:

  - **Commands**

    When you select this menu (selected by default), a window with a subset of most frequently used commands appears. You can select and execute one of the following commands: **Check**, **Clear**, **Fld-type**, **Gen**, **Read**, **Save**, **Type** or **Stow**. For explanations of these commands, see the corresponding direct commands described in ***Editor Commands and Function Keys***.
  - **Direct Command Line**

    When you select this menu, all command menus are closed and the command line appears.

- **Quit**

  When you select this menu, the data area editor is terminated. Any changes made to the current data area since the last `SAVE` or `STOW` command are *not* saved.

Editing Area

The editing area is either empty or contains source code that was last read into the source work area with the command `EDIT` or `READ` as shown in the example in *Invoking the Data Area Editor*.

When you read in the source of an existing object, the entire source code is loaded into the source work area and is available for editing. However, depending on the size of the source, the editing area may not show all of the lines that belong to the source. In this case, you have to scroll down in the source to go to the line you want to view or modify as described in *Viewing and Selecting Source Lines*.

Columns in the Editing Area

The editing area of the editor screen is organized in columns where all attributes that belong to a variable or field definition are maintained in one line.

The columns contained in the editing area are described in the following section. The contents of the columns depend on the values entered in the **Definition** or **Redefine** window (see *Input Fields in the Definition or Redefine Window*) when creating or modifying a variable or field. The contents also depend on whether a counter field (C* variable) was created from a field.

Note: A column heading can change or disappear depending on the type of variable or field contained in the current source line.

Column Heading	Explanation
C	The command column in which you can enter one of the line commands required to create or modify source lines. See also *Line Commands*.
T	The type of variable or field. Possible types are:

Column Heading	Explanation	
	B	**Block** A data block within a GDA.
	C	**Constant** or **Counter Variable** A user-defined constant (not applicable to PDAs) or a counter field (C* variable). A counter field is used for a multiple-value field or a periodic group within a view (DDM).
	F	**Data Field**: filler character. The filler bytes that can be denoted within a field or variable being redefined.
	G	**Group** A group within a view (DDM).
	M	**Multiple Field** A multiple-value field within a view (DDM).
	O	**Object Handle** The handle of an object.
	P	**Periodic Group** A periodic group within a view (DDM).
	R	**Redefined Field** The redefinition of a variable or field.
	V	Not applicable to PDAs. **View** A view definition created from a DDM.
	blank	**Data Field** A user-defined variable or field, or a group structure (not within a view).
	*	**Comment**. A comment field.
L	The level number of the variable or field (1 - 99). Variables which are not within a hierarchical structure and view definitions must be assigned level 1. Level numbers cannot be used with data block definitions.	

Column Heading	Explanation
Name of *variable-field-type* or **Comment**	The name of the variable or field. This column heading changes according to the **type of variable or field** currently selected as indicated in the description of column **T**. For a view, in addition to **Name of** *variable-field-type*, the **Name of DDM** column appears. For a block, in addition to **Name of** *variable-field-type*, the **Name of Parent Block** column appears.
F	The Natural data format of the variable or field.
Length	The length of the variable or field.
Index/Comment	The array indices and/or comment of the variable or field.
M	The **M** (Miscellaneous) column contains an X if an edit mask, header, and/or initial value is defined for a variable or field. This column does not appear for a view, or a group, periodic group or multiple-value field within a view.

Viewing and Selecting Source Lines

In edit mode, you can scroll up or down in the current data area to view all lines of the source currently contained in the source work area, select a line for modification, or position at the line where you want to insert new lines.

To scroll in a source and select a line

- Scroll up or down one line with UP ARROW and DOWN ARROW respectively.

  Or:

  Press HOME or END to access the corresponding first (topmost) line or the last (bottommost) line contained in the source work area.

  Or:

  Move the current source line up or down one screen page at a time with PAGE UP or PAGE DOWN respectively. Check your file *SAGtermcap* for entries if you have no access to any of these keys.

  The line in which the cursor is positioned is the current line, which is selected (highlighted) and can be modified.

Creating and Modifying Source Lines

In edit mode, you can create or modify the variable or field definition in the current source line by using the appropriate line command described in *Line Commands*.

The following are example instructions for using a line command to add single or multiple lines to a source or modify a line.

To add a line for a new definition other than a view, redefinition or counter field

1 In the **C** column, next to the selected line below which you want to place the new definition, type in the following line command:

```
I
```

If the data area is empty, enter the `I` next to **Top of Data Area**.

A selection window appears.

2 Select one of the following:

Data Field:	For a user-defined variable or a database field.
Block:	For a data block within a GDA described in the *Programming Guide*.
Constant:	For a user-defined constant described in *User-Defined Constants* in the *Programming Guide*.
Handle:	For an object handle described in *Using Classes and Objects* in the *Programming Guide*.
Structure:	For a hierarchical group structure (not within a view).
Comment:	For a comment field.

A **Definition** window appears for the selected type of variable or field in which you can enter the required attribute definitions. For explanations of the input fields contained in this window, see *Input Fields in the Definition or Redefine Window*.

To add lines for a view definition

1 In the **C** column, next to the selected line below which you want to place the view, type in the following line command:

```
V
```

If the data area is empty, enter `V` next to **Top of Data Area**.

A **View Definition** window appears.

2 Enter the name of the view to be created and the DDM from which to create the view. The specified DDM must be contained in the current Natural library in the current system file.

A **DDM** selection window appears with a list of all fields defined in the specified DDM.

3 Scroll through the list with the arrow (cursor) keys and enter an `X` next to each DDM field you want to select for the view.

Or:

In the **DDM** selection window, enter an `A` next to any of the DDM fields to select all DDM fields for the view.

You can deselect a field by replacing the `X` with a blank character.

If no periodic group or multiple-value field is selected, all fields are copied into the data area.

If you selected a periodic group or multiple-value field, an **Occurrences Definition** window appears.

4 In the **From:** field(s), enter the lower bounds of the one- or two-dimensional array to be used in the view, and, in the **To:** field(s), enter the upper bounds.

The periodic group or multiple-value field is copied into the data area with the specified occurrences as subordinate fields of a hierarchical view definition structure where the view name is at level 1.

To modify the contents of a line

- In the **C** column, next to the selected line, type in the following line command:

```
E
```

The **Definition** window (or **Redefine** window for a redefinition) appears for the variable or field definition contained in the selected line.

Input Fields in the Definition or Redefine Window

You create and modify a variable or field in the **Definition** window or the **Redefine** window for a redefinition. The input fields contained in this window depend on the type of variable or field selected. The table below lists and describes all possible input fields.

The definitions you enter in the **Definition** or **Redefine** window are checked for syntax errors.

For explanations of the variable or field attributes mentioned in this section, see also *User-Defined Variables* and *Field Definitions* in the *Programming Guide* and `DEFINE DATA` in the *Statements* documentation. The values used in the `DEFINE DATA` statement correspond to the values used in a data area.

Input Field	Explanation
Level	The level number (1 - 99) of the variable or field. Variables or fields which are not within a hierarchical structure must be assigned level 1. View definitions must be assigned level 1. Level numbers cannot be used with data block definitions.
Name	The name of one of the following: ■ User-defined variable or database field. ■ User-defined constant. ■ Structure (group definition, not within a view). ■ Handle of object. ■ Block and parent block (GDAs only). ■ View (DDM). ■ Multiple-value field (within a view). ■ Periodic group (within a view). ■ Counter field (C* variable) for a multiple-value field or a periodic group within a view. For valid names, see *Naming Conventions for User-Defined Variables* in the *Using Natural* documentation. For a user-defined constant, see also `CONSTANT` in the *Statements* documentation.
Format	The Natural data format of the variable or field. For valid formats, see *Format and Length of User-Defined Variables* and *Special Formats* in the *Programming Guide*. For a counter field (C* variable), you can specify the Natural data format/length I2 or I4 (the default setting is N3 for no format/length).
Length	The length of the variable or field. For valid lengths, see *Format and Length of User-Defined Variables* in the *Programming Guide*. No length is permitted for the Natural data formats C, D, T and L. You can define dynamic variables by specifying `DYNAMIC` in the **Length** field. For a counter field (C* variable), you can specify the Natural data format/length I2 or I4 (the default setting is N3 for no format/length).
Arraydefinition	The array definition of the variable or field. You can define the upper and lower bounds of a one- or two-dimensional array as demonstrated in ***Examples of Array Definitions***.
Edit Mask	Not applicable to PDAs. The edit mask of the variable or field to be used when the variable or field is displayed with an I/O statement.

<table>
<tr><th>Input Field</th><th>Explanation</th></tr>
<tr><td></td><td>Enter a valid value without using apostrophes or parentheses as shown in Examples of Edit Mask Definitions. For valid input values, see the corresponding session parameter EM described in the Parameter Reference documentation.</td></tr>
<tr><td>Header Definition</td><td>Not applicable to PDAs.

The header to be produced for the variable or field in a DISPLAY statement.

Enter any alphanumeric character string without using apostrophes or parentheses as shown in Example of a Header Definition. For further information on headers, see the corresponding session parameter HD described in the Parameter Reference documentation.</td></tr>
<tr><td>Initialization</td><td>Not applicable to PDAs.

The initial value assigned to a variable or field or the array occurrence(s) defined for a variable or field. You can specify one of the following initialization modes:

<table>
<tr><td>F</td><td>Free form mode.
If you enter F, the Free Form Initialization window appears, in which you can enter your initial value definitions according to the common Natural syntax definitions in a DEFINE DATA statement. You can assign the same initial value to a whole range of field occurrences at a time.

See also Examples of Initial Value Assignments in Free Mode.

For detailed information on defining initial values, see Initial-Value Definition Initial/Constant Values for an Array in the Statements documentation.</td></tr>
<tr><td>S</td><td>Single-value mode.
If you enter S, the Single Value Initialization window appears, in which you can enter a single initial value rather than an initialization clause.

Values are entered based on the variable or field type. You only enter the required variable or field value; any further specifications necessary (including apostrophes for alphanumeric variables or fields, or value prefixes such as H for hexadecimal, D for date and T for time) are generated automatically. For example, to specify an initial value of H'F1F2' for a binary variable (B2), enter F1F2. The data editor generates: INIT <H'F1F2'>.

If the variable or field is an array, all elements of the array are listed. A value for each element can be entered (optional).</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Input Field</th><th>Explanation</th></tr>
<tr><td></td><td>Caution: Changing the initialization type deletes all previously entered initialization information.

See also the sections Initial Values (and the RESET Statement) and Initial Values for Arrays in the Programming Guide.</td></tr>
<tr><td>Value Clause</td><td>Only applies to PDAs.

Determines the way in which the value of a variable or field specified as a parameter in a CALLNAT statement is passed from a program to an invoked object (for example, a subprogram).

Valid input values:

<table>
<tr><td>blank</td><td>Indicates the parameter specification call-by-reference (default).</td></tr>
<tr><td>V</td><td>Indicates the parameter specification call-by-value.</td></tr>
<tr><td>R</td><td>Indicates the parameter specification call-by-value-result.</td></tr>
</table>
For detailed information, see the corresponding options BY VALUE and BY VALUE RESULT described for the DEFINE DATA statement in Parameter Data Definition, and operand2 described for the CALLNAT statement in the Statements documentation.</td></tr>
<tr><td>Optional Param</td><td>Only applies to PDAs.

Determines whether the variable or field value is passed as a parameter according to the setting of Value Clause (see above):
<table>
<tr><td></td><td></td></tr>
<tr><td>N</td><td>A parameter must be passed (default).</td></tr>
<tr><td>Y</td><td>A parameter can be passed.</td></tr>
</table>
For detailed information, see the corresponding option OPTIONAL described for the DEFINE DATA statement in Parameter Data Definition and operand2 described for the CALLNAT statement in the Statements documentation.</td></tr>
<tr><td>Indexdefinition</td><td>The array definition of a variable of the type structure (group).

This input field can be used to define the upper and lower bounds for an array, to supply initial values for a variable or to supply an edit mask for a variable.
See also Examples of Array Definitions.</td></tr>
<tr><td>Comment</td><td>Commentary text.</td></tr>
</table>

Examples of Array Definitions:

```
(2,2)    /* 2 dimensions, 2 occurrences
(2,2,2)  /* 3 dimensions, 2 occurrences
(1:10,2)
(-1:3,2)
```

Examples of Edit Mask Definitions:

```
999.99
XXX...XX
MM.DD.YY
```

Example of a Header Definition:

```
HEADER TEXT
```

Examples of Initial Value Assignments in Free Mode:

```
INIT<3>
INIT<'ABC'>
INIT<H'F1F2'> /* binary variable (B2)
CONST<12>
INIT ALL<'ABC'>
```

Line Commands

You enter a line command in the **C** column of a source line when edit mode is set. The command entered may not be indicated in the column.

For a selection list of all available line commands, press F2.

All line commands provided by the data area editor are described in the following table. The expression “edit block” used in the table, denotes a marked block of source lines.

<table>
<tr><th>Command</th><th>Function</th></tr>
<tr><td>C

(Copy)</td><td>Copies one or more lines. If C is entered within an edit block, all lines of this block are copied.

See also To cut/copy and paste an edit block.</td></tr>
<tr><td>D

(Delete)</td><td>Deletes one or more lines.

If D is entered within an edit block, all lines within this block are deleted. For additional information, see the line commands C (Copy) and P (Paste).

See also To cut/copy and paste an edit block.</td></tr>
<tr><td>E

(Edit)</td><td>Depending on the variable or field type selected, opens the Definition or Redefine window (see also Input Fields in the Definition or Redefine Window), in which you can modify the attributes of the variable or field contained in this line.

For a field within a view definition, you can only modify the following: Level (except periodic groups and multiple-value fields), Edit Mask, Header Definition, Indexdefinition and Comment.</td></tr>
<tr><td>H

(Unmark block)</td><td>Unmarks the current edit block. This command must be issued from within the edit block.</td></tr>
<tr><td>I

(Insert line)</td><td>Inserts a line for a new definition as described in To add a line for a new definition other than a view, redefinition or counter field.</td></tr>
<tr><td>M

(Add comment)</td><td>Adds comment mark(s) to the selected line or the marked edit block. These lines are ignored by a compiler check. This may be useful for testing purposes.</td></tr>
<tr><td>N

(Remove comment)</td><td>Removes comment mark(s) from the selected line or the marked edit block.</td></tr>
<tr><td>P

(Paste)</td><td>Pastes one or more lines into the data area after the current line.

See also To cut/copy and paste an edit block.</td></tr>
<tr><td>R

(Redefine)</td><td>Redefines the variable or field contained in this line as single variable or a group of variables.

A window appears in which you can select one of the following redefine options:

<table>
<tr><td>Data Field</td><td>Redefines the variable or field as a single user-defined variable.</td></tr>
<tr><td>Filler</td><td>Redefines the variable or field using the filler option (nX). With the filler option, n filler bytes can be denoted within the variable or field being redefined, where n can be up to 10 digits (1 GB). The definition of trailing filler bytes is optional.</td></tr>
</table>
</td></tr>
</table>

Command	Function
	<table><tr><td>**Structure**</td><td>Redefines the variable or field as a structure (group).</td></tr></table> <table><tr><td>**Comment**</td><td>Redefines the variable or field as a comment.</td></tr></table> Depending on the redefine option selected, either a **Redefine** or a **Definition** window appears, in which you can enter the definitions required (see also *Input Fields in the Definition or Redefine Window*). The data area editor keeps track of the number of free bytes still available for the redefinition. If there are no free bytes, the redefine function ends. Depending on the redefine option used, the data area editor automatically adds the lines required for the redefinition above the variable or field that is redefined.
S (Show)	Displays the **Definition** or **Redefine** window which contains all definitions of the variable or field contained in this line (see also *Input Fields in the Definition or Redefine Window*). Changes are not possible. If an initialization has been specified, a separate window containing initialization information is also displayed. Information scrolling is possible.
V (Insert view)	Inserts a view definition as described in *To add lines for a view definition*.
X (Mark block start)	Marks the beginning of an edit block.
Y (Mark block end)	Marks the end of an edit block.
Z (Mark group)	Marks an entire group structure as an edit block. The edit block starts at the current line and includes all consecutive lines with levels less than the current line. For example, if Z is entered in a view line, the entire view is marked as an edit block.
* (Generate counter)	Not applicable to PDAs. Generates a counter field (C* variable) from the multiple-value field or periodic group contained in this line. The counter field is placed in the line above which you entered the command. A counter field is used to retrieve the number of occurrences of a multiple-value field or a period group. See also *Referencing the Internal Count for a Database Array (C* Notation)* in the *Programming Guide*.

The following are instructions for using line commands to move or copy a block of lines or a single line within the source.

To cut/copy and paste an edit block

1 Next to the first line of the block of lines (or the single line) to be cut or copied, enter the following line command:

```
X
```

The line is marked.

2 Scroll up or down to the last line of the block of lines and enter the following line command:

```
Y
```

For a single line, you enter the line command `Y` in the same line, in which you entered the line command `X`.

All lines that belong to the block are marked (highlighted) thus representing the edit block.

3 Within the edit block, enter one of the following line commands:

```
C
```

to copy the line(s), or

```
D
```

to delete (cut) the line(s).

4 Position the cursor in the line below which you want to paste the line(s) and enter the following line command:

```
P
```

The line(s) are pasted into the source.

5 You can unmark the edit block by entering the following line command within the edit block:

```
H
```

The edit block is unmarked and only the current line is highlighted.

Editor Commands and Function Keys

This section describes the editor commands and editor-specific system commands that can be entered in the command line or selected from the **Commands** menu in command mode.

In addition, this section describes alternative F keys that can be used in edit mode. In command mode, you can only use F1.

For explanations of the syntax symbols used in the editor commands, refer to *System Command Syntax* in the *System Commands* documentation.

Command	F Key	Function
n/a	F1	Provides help information on editor features. See also ***Help Information and Selection Options***. **Note:** When you enter `HELP` you will invoke the help function for error messages as described for the system command `HELP` in the *System Commands* documentation.
n/a	F2	Invokes a selection window with valid input values for the input field in which the cursor is positioned.
`CATALOG [object-name]`	n/a	Executes the system command `CATALOG` which **checks** and catalogs the current data area definition. You must supply an object name with the command if you catalog a new data area definition or if you want to copy the current data area. Otherwise, an appropriate message appears. See also ***Storing and Cataloging a Data Area***.
`CHECK`	F6	Executes the system command `CHECK` which checks the syntax of the current data area definition. A window informs you that a syntax check is in process. If a syntax error is found, the line containing the error becomes the current line, and the error is displayed in the message line. If no errors are found, a corresponding message is displayed.
`CLEAR`	F8	Executes the system command `CLEAR` which clears the source work area. Changes to the data area currently contained in the source work area are lost if they were not previously saved.
`FLD TYPE`	F11	`FLD TYPE` is available from the **Commands** menu only (you cannot enter it in the command line). Invokes a window in which you can change the type of the current variable or comment. You can select one of the following types: **D** — Data field **B** — Block

Command	F Key	Function
		C Constant **H** Handle **S** Structure * Comment The type of a field within a view definition cannot be changed. **Caution:** When changing types, some attribute definitions of a variable or comment can be lost.
GENERATE *object-name*	F10	Generates a Natural object of the type copycode from the current data area definition thus overwriting the contents of the source work area. Changes made to the data area since the last SAVE or STOW are lost. The copycode object is stored as a source object in the specified Natural library in the current system file under the *object-name* supplied with the command.
QUIT or . (a period)	F3	Terminates the data area editor. Any changes made since the last SAVE or STOW command are lost.
READ *object-name*	F7	Executes the system command READ which reads an existing data area definition into the source work area. The data area must be stored as a source object with the specified *object-name* in the current Natural library in the current system file.
SAVE [*object-name*]	F4	Executes the system command SAVE which saves the current data area definition. You must supply an object name if you save a new data area definition or if you want to copy the current data area. Otherwise, an appropriate message appears or input window appears. If you press F4, a window always prompts you for name of the data area and the library. If the current object and library names are correct, no entry is required. See also ***Storing and Cataloging a Data Area***.
SCAN *scan-value*	n/a	Scans the data area for a character string (*scan-value*) in the **Name of** *variable-field-type* column (default) and/or the **M** column of the editor screen, depending on whether the SET SCAN command was executed earlier. The line in which the *scan-value* is found is highlighted. If the first instance of *scan-value* is highlighted, you can go to the next instance by pressing ENTER.

<table>
<tr><th>Command</th><th>F Key</th><th>Function</th></tr>
<tr><td></td><td></td><td>The scan is performed from the first to the last line in the source work area and wraps around to the beginning after the last line is reached.

Note: The SCAN command performs an exact search for the specified scan-value. This should be taken into account when searching for DBCS (Double Byte Character Set) characters.</td></tr>
<tr><td>SET ABS [ON|OFF]</td><td>n/a</td><td>Determines whether the SCAN command operates in absolute or non-absolute mode.

<table>
<tr><td>ON</td><td>The SCAN command operates in absolute mode, which means that the value to be scanned needs not be delimited by blanks or special characters.</td></tr>
<tr><td>OFF</td><td>The SCAN command operates in non-absolute mode, which means that the value to be scanned must be delimited by blanks or special characters.</td></tr>
</table>
The default is OFF.</td></tr>
<tr><td>SET SCAN [COMMENT|NAME|LEVEL]</td><td>n/a</td><td>Determines the column(s) in which the SCAN command searches for a scan-value:

<table>
<tr><td>COMMENT</td><td>The Index/Comment column is scanned.</td></tr>
<tr><td>NAME</td><td>The Name of variable-field-type (or Comment) column is scanned.</td></tr>
<tr><td>LEVEL</td><td>Scans within a hierarchical group structure.</td></tr>
</table>
The default is NAME.</td></tr>
<tr><td>STOW [object-name]</td><td>F5</td><td>Executes the system command STOW which saves and catalogs the current data area definition.

You must supply an object name if you STOW a new data area or if you want to copy the current data area.

See also Storing and Cataloging a Data Area.</td></tr>
<tr><td>SET TYPE G|L|A</td><td>F9</td><td>Changes the type of the current data area:

<table>
<tr><td>G</td><td>Global data area</td></tr>
<tr><td>L</td><td>Local data area</td></tr>
</table></td></tr>
</table>

Command	F Key	Function	
		A	Parameter data area

Storing and Cataloging a Data Area

Before a data area can be used in a Natural program (or another object), the data area must be saved as a source object and/or a cataloged object in the current Natural environment.

The commands used for saving and cataloging the current data area definition are described in the following section.

You must supply an object name if you save or catalog a new data area definition or if you want to copy the current data area. Otherwise, an appropriate message appears or a window prompts you for the name. For the naming conventions that apply when saving or cataloging a data area, refer to *Object Naming Conventions* in the *Using Natural* documentation.

To save a data area as a source object

- Enter the system command `SAVE` according to the syntax rules described in the *System Commands* documentation.

  Or:

  Press F4.

  Or:

  From the **Commands** menu, choose **Save**.

  The source of the data area is stored as a source object in the specified Natural library in the current system file. The data area is not checked for syntax errors.

To save a data area as a cataloged object

- Enter the system command `CATALOG` according to the syntax rules described in the *System Commands* documentation.

  The source of the data area is checked for syntax errors. If no errors are found, it is saved as a cataloged object in the specified Natural library in the current system file.

To save a data area as a source object *and* a cataloged object

- Enter the system command `STOW` according to the syntax rules described in the *System Commands* documentation.

Or:

Press F5.

Or:

From the **Commands** menu, choose **Stow**.

The source of the data area is checked for syntax errors. If no errors are found, it is saved as a source object and a cataloged object in the specified Natural library in the current system file.

Help Information and Selection Options

You can use the help system to obtain help information on editor functions and input fields. In addition, in edit mode, you can use the help system to choose a valid input value or a line command from a list.

To display help information

1 In edit mode, position the cursor in the **C** column and press F1 for a summary of all editor functions available.

Or:

Place the cursor on the field about which you require further information and press F1 for instructions on using this field.

A window appears with help information.

You can use the arrow (cursor) keys to scroll up or down in the window.

2 Choose ENTER or ESC to close the help window.

To select a valid value or appropriate line command

- Place the cursor on the field for which you want to select a valid input value or execute a line command and press F2.

  If applicable, a selection window appears from which you can select an input value or a line command.

V Map Editor

18 Map Editor

The Natural map editor is used to create a Natural object of type map. A map is a screen layout that can be referenced in a Natural object such as a program by using either an `INPUT USING MAP` statement (for input maps) or a `WRITE USING MAP` statement (for output maps).

A map contains text fields and data fields. Text fields are literal strings and data fields are variables. Data fields can be either user-defined variables or Natural system variables.

Once a map has been created, it can be stored as a source object and a cataloged object in a library in a Natural system file.

Notes:

1. Using Natural Studio in a Windows environment, the map editor supports fields with Unicode format and Unicode strings. However, when reading the source of a Unicode map into the editing area of a map editor in a local Linux or mainframe environment, all Unicode strings will be removed from the source.
2. The map editor has been disabled in your environment by default. For more information, see *NaturalONE as the Default Development Environment* in the *Editors* documentation.

Creating a Map

There are four major steps involved in the creation of a map:

1. Definition of the map profile (that is, the field delimiters, format settings, context settings and filler characters to be used). A menu is provided from which you select the desired items.
2. Definition of the map. A map can be defined in two different ways:
   - First define a prototype map, next make the corresponding data definitions in the object that references the map, then integrate the map into the application.

     Fields can be defined directly on the map editing screen. Each field is assigned a default name. Subsequently, when the corresponding data definitions have been made in the respective object, these data definitions can be assigned to the map fields (**post assignment**).
   - Define a map using existing data definitions.

     If data definitions already exist in an object that references the map, the map fields can be created by using the data definitions contained in this object. In this case, all characteristics of the data definitions are copied into the map.
3. Definition of the fields to be used in the map. Map fields can be created by either typing the field definitions directly in the map editing screen or by selecting data definitions from another Natural object (see also *Selecting Data Definitions*).
4. The saving and/or cataloging of the map definition. Once a map has been defined as described in the previous steps, it can be saved as a source object and/or cataloged object in the current

library and Natural system file. Once saved as a source object, the map can be read and modified during a subsequent map editor session. Once saved as a cataloged object, a map can be invoked from a Natural program.

Invoking the Map Editor

You invoke the Natural map editor to create a new map or edit an existing one as described in the following section.

Creating a New Map

From the Natural main menu, select **Direct** and press ENTER to invoke the **Direct Command** window.

Note: You can also invoke the **Direct Command** window by selecting **<DIRECT COMMAND>** from an object selection list that is displayed after you have selected a library from the Natural main menu.

In the **Direct Command window**, enter the following command:

```
EDIT MAP
```

or, in short form

```
E M
```

Press ENTER. The **MAP EDITOR** menu will be displayed:

```
 ..............................................................................
                    NATURAL MAP EDITOR (Esc to select field)                  .
Create   Modify    Erase     Drag      Info OFF  Lines     Ops. Map  Quit     .
 ..............................................................................
```

Editing an Existing Map

From the Natural main menu, select **Direct** and press ENTER to invoke the **Direct Command** window.

In this window, enter the following command:

```
EDIT map-name
```

or, in short form

```
E map-name
```

If you do not remember the name of the map you want to edit, select **Library** from the Natural main menu and press ENTER. A list of all available libraries will be displayed.

From the list, select the desired library with the cursor, and press ENTER. A list of all objects in this library will be displayed.

Note: File designations are listed under the column **Type**.

With the cursor keys, scroll through the list until the desired map appears on the list. Then mark it with an E (Edit) in the column before the object name and press ENTER. The map editor will then be invoked for the selected map.

Note: The F2 key invokes a window listing valid functions (for example, **C** for Check, **D** for Read, **E** for Edit) for this specific object for the item **Ops. Map**.

Regardless of the map invoked, you are prompted at the bottom of the screen to use the cursor keys to select a field and to press ENTER. The selected field is highlighted.

Note: The rightmost position of the bottom line displays the column and row number of the highlighted field.

Press ESC and the **MAP EDITOR** menu is invoked.

Note: To change the programming mode (structured/reporting) for an existing map, enter the following commands in the **Direct Command** window:

```
READ map-name
```

```
GLOBALS SM=ON/OFF
```

```
SAVE map-name
```

Map Editor Menu

The **MAP EDITOR** menu shown earlier is the main menu of the map editor. The menu items are described in the following section.

- Create
- Modify
- Erase
- Drag
- Info ON/OFF
- Lines
- Ops. Map
- Quit

Create

If you press ENTER, a list containing the following items is displayed:

- **A** - Parameter Data Area
- **G** - Global Data Area
- **H** - Helproutine
- **L** - Local Data Area
- **M** - Map
- **N** - Subprogram
- **P** - Program
- **S** - Subroutine
- **T** - Text Constant
- **U** - User Defined
- **V** - View (DDM) Defined
- **1** - Parameter Defined
- **2** - Local Defined

These items are used to select fields/variables from an object for an object of this class. The object class can also be invoked by entering `C` and the object class abbreviation (key-sensitive). For more detailed information, see the section *Selecting Data Definitions*.

Modify

Modify enables you to modify a selected field. The selected field is the current field and it is highlighted.

A window displaying field attributes (extended field editing) is displayed in which the contents of these attributes can be modified.

Erase

Erase enables you to delete the current field. You are then prompted:

```
Delete field (Y/N)?
```

All responses are key-sensitive; caution is recommended.

If the current field is an array field, the entire array (not just the field) will be deleted.

Drag

Drag enables you to move the current field to any unoccupied position on the screen. The selected field can be moved without restriction, using the cursor.

Once the field is positioned and ENTER is pressed, this field position takes effect.

Info ON/OFF

Info ON/OFF is used to switch the display of the **Extended Field Editing** window on and off (toggle switch). **OFF** is the default value.

The ENTER key or I (key-sensitive) is used to switch between **ON** and **OFF**.

Lines

Lines invokes a selection list from which you can select the following line-specific functions:

```
Insert After
Erase Line
Copy After
Duplicate Line
Move After
Split Line
Join Line
```

These functions can be selected to perform operations on an entire line (not a single field) in a map. All operations are performed on the current line.

Note: These functions are self-explanatory and prompts appear at the bottom of the screen for each item selected.

Ops. Map

Ops. Map (map operations) invokes the following selection list:

```
C Check Map
E Edit  Map
V Reverse  Map
K Key Rules
L List  Map
P Prof. Map
R Read  Map
S Save  Map
T Test  Map
W Stow  Map
```

The list contains the following items:

Item	Description
Check Map	Causes syntax checking and generation of source code.
Edit Map	Invokes the map editing screen for modification of an existing map definition. The map editor will start a new edit session.
Reverse Map	This feature only applies if the default code page supports languages that are written from right-to-left (RTL) such as Hebrew. Reverses the screen direction of a map from left-to-right (this is the default) to right-to-left, or vice versa. This is only a visual change; the physical field position defined in the map source is retained. The screen direction remains reversed until you select **Reverse Map** again.
Key Rules	Invokes editing of function-key-related processing rules.
List Map	Generates source code and lists it.
Prof. Map	Invokes a map profile window, which is described under *Map Profile*.
Read Map	Invokes the map editing screen to read an existing map definition.
Save Map	Performs a source code generation check and then saves the map. The map definition is saved in source form in the Natural library.
Test Map	The current map definition is tested to ensure that it can be executed successfully. This includes testing of all processing rules and help facilities.
Stow Map	Performs a source code generation check, as well as a save and catalog of a map definition. The map definition is cataloged and also saved in source form in the current Natural library.

Quit

Quit terminates the map editor session.

If you have edited the map (not saved), selected **Quit** and pressed ENTER, the following prompt appears:

```
Modifications have not been saved, quit anyway Y/N ?
```

Note: Replies are key-sensitive; caution is recommended.

If the editor session is terminated, the Natural main menu is displayed.

Creating a Text Constant

Select **Text Constant** from the **Create** list and press ENTER.

Depending on the map type invoked, a screen appears.

At the bottom of the screen, you are always prompted to position the cursor and enter text.

Position the cursor to the start of an empty field.

Note: You cannot overwrite existing fields.

Enter the text. The first character entered causes the line to be highlighted. Highlighting indicates the maximum space available for text entry. Characters can be entered or deleted until you press ENTER.

Note: The ESC key cancels text entry.

Press PF2 to select an attribute and color to be used for the text entered. Use the UP-ARROW and DOWN-ARROW keys to scroll through and select one of the available attributes/colors or use the corresponding abbreviation (for example, `B` for blinking or `RE` for red) and press ENTER. Using the LEFT-ARROW and RIGHT-ARROW keys, you can toggle between attribute and color definition.

Text entry is now complete.

Note: If you want to create and/or define a data variable after having defined the text, see the section *Using Natural System Variables in a Map*.

Creating a User-Defined Variable

Select **User Defined** from the field list and press ENTER.

An **Extended Field Editing** window similar to the example below appears:

```
Extended Field Editing
Field :
Format: A Len:            AL:             PM:                   ZP: N  SG: N
Rules : 0 Rule Editing: N Array:          Array Editing: N      Mode:
AD:           CD:         CV:                                   DY: N  HE: N
EM:
```

A message appears at the bottom of the screen, prompting you to: `Position cursor and press Enter or format char.`

Position the cursor to the start of a field position and press ENTER.

A selection list appears, with all valid Natural data formats such as **A** for data type alphanumeric.

Note: If you know the Natural data format, you can enter it directly in the field position where you placed the cursor. The display of available data formats is thus avoided.

Select the Natural data format required and press ENTER.

The Natural data format is now entered in the **Extended Field Editing** window and a default name (for example, #1) is assigned to the field.

There are two length fields displayed on this screen:

1. **AL** (alphanumeric length): the display length.
2. **Len**: the internal length of the field.

The lengths of user-defined variables are defined by performing the following steps:

1. Enter the length of the first field (for example, Alphan. Len...).
2. When the field length definition is complete, press ENTER.
3. You are then prompted to enter a field name for the variable. Having selected the appropriate field name, press ENTER.

The definition function for the first field is now complete.

The cursor moves automatically to the second length field (Len...). Change the length field definition or use the TAB key. Continue this process to define all other pertinent information to be used for the field being defined.

For further information on these fields, see *Modifying a User-Defined Variable - Field Editing*.

When this definition is complete, press ENTER.

Using Natural System Variables in a Map

Natural system variables can also be specified in a map definition.

A Natural system variable can be selected with a **Create** > **User Defined** field.

Select * (System) from the field list and select the system variable from the list provided.

The Natural data format of the specified system variable is inserted in the field definition form.

Modifying a User-Defined Variable - Field Editing

The map editor is used to define a field with all its attributes.

Select **Create** or **Modify** from the **MAP EDITOR** menu and press ENTER. An **Extended Field Editing** window similar to the example below appears, which displays all attributes for the current field:

```
Extended Field Editing
Field :
Format: A Len:            AL:             PM:                   ZP: N  SG: N
Rules : 0 Rule Editing: N Array:          Array Editing: N      Mode:
AD:           CD:         CV:                                   DY: N  HE: N
EM:
```

With the **Extended Field Editing** window, any selected field can be modified. A selected field is the current field, which is highlighted.

The field attributes contained in the **Extended Field Editing** window are explained in the following table:

Field Attribute	Explanation
Field	The field name. Field name assignment is related to the method with which the field was originally defined. If the field was taken from a data definition in another Natural object, it is assigned the same name as the field definition in this object. If the field was specified as a Natural system variable, it is assigned the name of the specified variable.

<table>
<tr><th>Field Attribute</th><th>Explanation</th></tr>
<tr><td></td><td>If the field is neither of the above, it is assigned a dummy name. You must assign a name to such a field prior to map execution.

The name of a field can be changed. However, a prefix must not be used for a field which did not have a prefix assigned previously. To obtain a prefixed field name, select the field from the data definition in another Natural object. You are prompted to enter a name. If modifications have been made you must press ENTER to continue. Otherwise, you can move through the field attributes using the TAB key.

Note: Duplicate field names are only allowed for fields defined as output-only fields.</td></tr>
<tr><td>Format</td><td>The Natural data format of the field. These can be changed by overwriting the current entry.</td></tr>
<tr><td>Len</td><td>The internal program length of the variable.</td></tr>
<tr><td>AL or NL or FL or DF</td><td>The length to be used when displaying the field. The field label depends on the Natural data format entered:

<table><tr><td>AL</td><td>for formats A (alphanumeric), L (logical) and T (time).</td></tr><tr><td>NL</td><td>for formats B (binary), I (integer), N (numeric) and P (packed numeric).</td></tr><tr><td>FL</td><td>for format F (floating point).</td></tr><tr><td>DF</td><td>for format D (date).</td></tr></table></td></tr>
<tr><td>PM</td><td>Print Mode.

This input field corresponds to the session parameter PM. For detailed information on using this field and valid input values, see PM - Print Mode in the Parameter Reference.</td></tr>
<tr><td>ZP</td><td>Zero printing. You can only enter a value in ZP if the field is numeric or a time system variable.

This input field corresponds to the session parameter ZP. For detailed information on using this field and valid input values, see ZP - Zero Printing in the Parameter Reference.</td></tr>
<tr><td>SG</td><td>Sign position. You can only enter a value in SG if the field is numeric or a time system variable.

This input field corresponds to the session parameter SG. For detailed information on using this field and valid input values, see SG - Sign Position in the Parameter Reference.</td></tr>
<tr><td>Rules</td><td>The number of processing rules currently defined for the field.</td></tr>
<tr><td>Rule Editing</td><td>Editing of processing rules; you are prompted:

<code>Inline Processing Rule Editing (Y, <PF2>=EDIT)?</code>

Note: The source code used to define a processing rule is entered/edited in the same way as with the Natural program editor.</td></tr>
<tr><td>Array</td><td>Indicates whether the field is an array or not (blank).</td></tr>
<tr><td>Array Editing</td><td>Editing of arrays; you are prompted:

<code>Array Editing (Y, <PF2>=EDIT)?</code></td></tr>
</table>

Field Attribute	Explanation
Mode	Indicates how the field was created: **Data** - Field was copied from the data definition in another Natural object (except DDM). **Sys** - Field is a system variable. **Undef** - Field was created directly on the screen and has a dummy name. **User** - The name of the field was created by extended field editing. **View** - Field was copied from the field definition in a DDM.
AD	Field attributes. This field corresponds to the session parameter AD. For detailed information on using this field and valid input values, see the relevant sections (referenced below) in *AD - Attribute Definition* in the *Parameter Reference* documentation. You are prompted: `Attribute definitions (<PF2>=EDIT)` An Attribute definition window, containing the following items is displayed: ■ Representation See *Field Representation*. ■ Alignment See *Field Alignment*. ■ I/O Characteristics See *Field Input/Output Characteristics*. ■ Mandatory Characters See *Mandatory Input*. ■ Length Characteristics See *Length of Input Value*. ■ Upper/Lower Case See *Field Upper/Lower Case Characteristics*. ■ Filler Character See *Filler Character*. This function incorporates a toggle feature. If the attribute definition character displayed is correct, use the ESC key and the character is not changed. To change the attribute characteristics of a field, select the desired attribute and press ENTER; the modification is inserted in the **AD**

Field Attribute	Explanation
	definition. Press ESC to exit this function. Each item selected invokes an attribute window. These windows are self-explanatory.
CD	Color attributes; you are prompted: `Color definitions (<PF2>=EDIT <CSR-UP/DN>=Select (Esc=Cancel Enter=OK))` Press PF2 to edit the color definition, use the cursor to select a color and confirm with ENTER.
CV	Control variable for dynamic field attributes; you are prompted: `(<PF2>=Edit Rank if Array)` The name of a variable which contains the attributes to be used for this field. This variable must be defined with Natural data format C in the program. The control variable also contains a MODIFIED data tag, which indicates whether the field has been modified following map execution. A single control variable can be applied to several map fields, in which case the MODIFIED data tag is set if any of the fields referencing the control variable has been modified.
DY	Dynamic string attributes; you are prompted: `(Y,<PF2>=Edit)` This parameter is used to define certain characters contained in the text string of an alphanumeric variable to control the attribute setting. For detailed information on valid input values, see *DY - Dynamic Attributes* in the *Parameter Reference*.
HE	The **HE** option is used to assign a helproutine or a help map to the map field. A helproutine or help map is then invoked at execution time when a help request is made for the map field. For detailed information, see the description of the HE session parameter in *HE Helproutine* in the *Parameter Reference*. If you enter a `Y` in the **HE** field or place the cursor in the field and press PF2, a window prompts you to enter the name of a helproutine or help map and parameters to be passed to this helproutine or help map. The syntax that applies to specifying names and parameters in the **HE** field corresponds to the syntax of the `HE` session parameter described in *HE Parameter Syntax* (*Parameter Reference*). In addition to the syntax explanations provided there, the following applies when using the map editor: *operand1*: ■ If a variable name is specified which corresponds to the name of a map field, this field must be in the Natural data format/length A8. ■ If a variable name is specified for which no map field yet exists, a map parameter with that name is automatically defined in the Natural data format/length field A8. *operand2*:

Field Attribute	Explanation
	■ If a variable name is specified for which no map field yet exists, a map parameter with that name is automatically defined in the Natural data format/length N7. Removing a parameter from the **HE** field implies that the parameter is also removed from the map, unless the parameter is a map field or is associated with any other map field as a help parameter or **Starting from** value.
EM	Edit mask to be used for the field.

Three of the above items are of special interest:

- Rule Editing - Processing Rules
- Array Editing
- AD - Attribute Definition

Rule Editing - Processing Rules

Field-Related Processing Rules

Three types of processing rules can be defined:

- Inline processing rules
- Predict free rules
- Predict automatic rules

Inline processing rules are defined within a map source and do not have a name assigned. The availability of Predict is not required for inline rules.

Note: Field-related inline processing rules can also be executed on a field-by-field basis; for further information, see the section ***Field-Sensitive Processing***.

Predict free rules have a name assigned and are stored in the Predict Dictionary. You cannot modify an existing Predict free rule (this can only be done in Predict); however, you can read a free rule into the editor, modify it, and store it under a different name to create a new free rule.

You can access Predict free rules that are stored in either a local or a remote Predict environment.

Predict free rules that are stored in a remote environment on a Linux or a mainframe host can be accessed by using a Natural RPC server.

For information on how to connect to a Predict server, see the profile parameter `USEDIC` (*Parameter Reference*) and *Dictionary Server Assignments* in the section *Overview of Configuration File Parameters* (*Configuration Utility* documentation).

Inline rules can become Predict free rules (and vice versa) if you assign/remove the rule name.

Predict automatic rules apply to database fields and are defined by the Predict administrator. If a field is created from the data definition in another Natural object, and if the field is a database field, all automatic rules for that field are linked to the map definition. All automatic rules are concatenated and treated as a single map rule.

The rank of the automatic rules is defined in the map profile settings (default 1).

Automatic rules cannot be modified using the map editor. They can, however, be assigned a different rank by either using the command `P=n` or just overwriting the old rank.

An ampersand (&) within the source code of a processing rule is dynamically replaced by the fully-qualified name of the field for which the rule is defined. Array indexes are not affected by the replacement; you must explicitly specify the index notation after the ampersand (&) as shown in the following example.

Examples:

```
IF &    = ' ' THEN REINPUT 'ENTER NAME' MARK *&   /* For a scalar field
IF &(1) = ' ' THEN MOVE 'X' TO &(*)               /* For an array field
```

The field name notation `&.field-name` within the source code of a processing rule allows you to have DDM-specific rules that cross-check the integrity of values between database fields, without having to explicitly qualify the fields with a view name. As *field-name* you specify the name of the database field as defined in the DDM, and at compilation time, Natural dynamically qualifies the field by replacing the ampersand (&) with the corresponding view name. This allows you to use the same processing rule for specific fields, regardless of which view the fields are taken from.

Function-Key-Related Processing Rules

Two types of function-key-related processing rules can be defined:

- Inline processing rules
- Predict free rules

Function-key-related processing rules can be used to assign activities to program sensitive function keys during map processing. For function keys which already have a command assigned by the program, this command is executed without any rule processing.

Example:

```
IF *PF-KEY = 'PF3'
   ESCAPE ROUTINE
END-IF
```

When this rule is executed, map processing is terminated without further rule processing.

Processing-Rule Ranks

A field can have up to 100 processing rules (Rank 0 to 99). At map execution time, the processing rules are executed in ascending order by rank and screen position of the field. PF-key processing rules are always assumed to have the first screen position.

For optimum performance, the following assignments are recommended when assigning ranks to processing rules:

Rank	Processing Rule
0	Termination rule
1 - 4	Automatic rules
5 - 24	Format checking
25 - 44	Value checking for individual fields
45 - 64	Value cross-checking between fields
65 - 84	Database access
85 - 99	Special purpose

Processing-Rule Editing

Note: When modifying or adding a processing rule, keep in mind that you must recatalog the map(s) that reference this rule so that the changes become effective.

To edit field-related processing rules, select **Rule Editing** in the **Extended Field Editing** window. To edit function-key-related processing rules, select **Key Rules** from the **Ops. Map** selection list.

The following window containing the options Rules and Fields appears:

```
··Current Field: PERSONNEL.STREET·····································
·                     R U L E   E D I T I N G (Esc = Quit)             ·
·Rules                                  Fields                         ·
·····································································

                                      Hobby:  XXXXXXXXXXXXXXXXXXXXXXXXXXXX
Address
-
State        :  XX
Zip          :  99999
City         :  XXXXXXXXXXXXXXX
Street/Number:  XXXXXXXXXXXXXXXXXXX  99999
                                                                        ↩
 Create or modify a rule for this field
```

Select one of these options with the cursor and press ENTER.

If you select Fields, a list of variables used in the current map appears (for information purposes only).

If you select Rules, a selection list of existing rules for the current field appears.

On each list, the Predict rules are identified by their names, the inline rules by their first three source code lines.

There are the following ways to define processing rules for a field or key:

- Create a new processing rule
  - define a new rule
  - modify an existing rule and save it under a new name
- Assign an existing rule and
  - edit or
  - move or
  - copy or
  - unlink the rule.

To define a new processing rule

1 Select Create.

An empty rule editor is displayed.

2 Enter the rule. (Use source code in the same way as in the Natural program editor.)

3 If you want the rule to be a Predict free rule, name it. If you want the rule to be an inline rule, do not name it before saving it.

4 Save the rule, see the section *Commands for Editing Processing Rules*.

Important: Once you have saved the rule, you can only modify it in Predict.

To modify an existing processing rule

1 In the Rule Editor header, enter the name of a Predict rule in the field Rule and press ENTER.

The rule is displayed in the rule editor.

2 Modify the rule. (Use source code in the same way as in the Natural program editor.)

3 Rename the rule and save it, see section *Commands for Editing Processing Rules*.

Important: Once you have saved the rule, you can only modify it in Predict.

To assign an existing rule

1 Select an existing rule from the list.

2 Press ENTER.

A window with the following options is displayed: edit, move, copy, unlink.

- **Edit**
  Select edit to modify the rule. The name and contents of the rule (if it is a free rule) are displayed in the editor. See Step 2 of the section to modify an existing processing rule.
- **Move**
  Select move to modify the rule's rank. When you press ENTER a list is displayed from which you can select the new rank.
- **Copy**
  Select copy to copy the rule but assign it a new rank. When you press ENTER a list is displayed from which you can select the new rank.

- **Unlink**
  Select unlink to remove the rule from the field.
- Quit with ESC.

Note: If rules are written referencing a database statement, a label should be used, not a line reference number.

After the desired field processing rule has been entered, issue the command

```
P=nn
```

where *nn* is the processing rule rank.

This command saves the rule automatically.

Commands for Editing Processing Rules

In the processing-rule editor, processing rules can be selected for editing by using the following commands in the editor command line:

Command	Function
P *nn*	Select rule with rank *nn*.
P*	Select the rule from selection list.
P	Advance to the next rule defined for the field.
P=*nn*	Assign the rule on the current rank to rank *nn* and save it automatically.
U	Unlink (Delete)
.	End processing-rule editing and save the rule.

Array Editing

To invoke array editing, use the TAB key to select the item **Array Editing** in the field attributes window. Replace the character N by Y (key sensitive) or press PF2 and the **Array Definition** window appears:

```
·Array Definition·····························································
Name #1                                      Upper Bnds 1_____ 1_____ 1_____
-------------------------------------------------------------------------------
Dimensions                    Occurrences    Starting from     Spacing
1 . Index vertical            1__            _________         0   Lines
0 . Index horizontal          1__            _________         1   Columns
0 . Index (H/V) V             1__            _________         0   Cls/Ls
·····························································
```

This window displays the fields as defined on the map. If changes are not required, press TAB to proceed to the next field; if you want to return to a previous field, press SHIFT+TAB.

Note: The map editor does not support X-arrays (eXtensible arrays).

The **Array Definition** window contains the following entries:

Entry	Explanation
Upper Bnds	Indicates the upper bounds of the array; that is, the highest occurrence in (from left to right) the first, second and third dimension. If a field defined in a program is used to define the map array, the upper bounds of that field (user-defined variable or database field), as defined in the program, are used; these cannot be overwritten on the array definition screen. If the map array is derived from the data definition in another Natural object, the dimensions of the map array must not exceed the dimensions shown in this field. If the map array is not derived from the data definition in another Natural object, the dimensions of the map array must not exceed the dimensions as defined in the Natural program.
Dimensions	An array can have up to three dimensions. The order in which the dimensions of the array are mapped to the map layout is determined by the values entered to the left of the Index operands; the abbreviations used are: **H**=Horizontal and **V**=Vertical.
Occurrences	The number of occurrences to be defined for a dimension.
Starting from	The starting index value for a dimension. A numeric value can be used, or a variable name can be used to indicate that the actual value is supplied in the Natural program which invokes the map definition.
Spacing	The number of blank lines (for vertical dimensions) or blank columns (for horizontal dimensions) to be inserted between each dimension occurrence.

Enter the desired information and press ENTER. You are returned to the next item (**AD**) listed in the field attributes window.

Examples of Array Definitions

Example 1:

A one-dimensional array consisting of 10 vertical occurrences with 2 blank lines to be inserted between each occurrence.

```
·Array Definition·····························································
Name #1                                      Upper Bnds 10____ 1_____ 1_____
-------------------------------------------------------------------------------
Dimensions                    Occurrences    Starting from     Spacing
1 . Index vertical            10_            _________         2   Lines
0 . Index horizontal          1__            _________         1   Columns
0 . Index (H/V) V             1__            _________         0   Cls/Ls
·····························································
```

Example 2:

Same as example 1 except that the array is to be horizontal.

```
·Array Definition·····························································
Name #1                                      Upper Bnds 10____ 1_____ 1_____
-------------------------------------------------------------------------------
Dimensions                    Occurrences    Starting from     Spacing
0 . Index vertical            1__            ________          0   Lines
1 . Index horizontal          10_            ________          1   Columns
0 . Index (H/V) V             1__            ________          0   Cls/Ls
·····························································
```

Example 3:

A two-dimensional array. The first dimension consists of 10 vertical occurrences with 1 blank line between each occurrence. The second dimension consists of 5 horizontal occurrences with 2 blank columns between each occurrence.

```
·Array Definition·····························································
Name #1                                      Upper Bnds 10____ 5_____ 1_____
-------------------------------------------------------------------------------
Dimensions                    Occurrences    Starting from     Spacing
1 . Index vertical            10_            ________          1   Lines
2 . Index horizontal          5__            ________          2   Columns
0 . Index (H/V) V             1__            ________          0   Cls/Ls
·····························································
```

Example 4:

Same as example 3 except that the order of the dimensions is reversed.

```
·Array Definition·····························································
Name #1                                      Upper Bnds 5_____ 10____ 1_____
-------------------------------------------------------------------------------
Dimensions                    Occurrences    Starting from     Spacing
2 . Index vertical            10_            ________          1   Lines
1 . Index horizontal          5__            ________          2   Columns
0 . Index (H/V) V             1__            ________          0   Cls/Ls
·····························································
```

Example 5:

A three-dimensional array. The first dimension consists of 3 vertical occurrences with 1 blank line between each occurrence. The second dimension consists of 5 horizontal occurrences with 2 blank columns between each occurrence. The third dimension consists of 2 occurrences, expanded vertically within each occurrence of the first dimension.

```
·Array Definition·····························································
Name #1                                    Upper Bnds 3_____ 5_____ 2_____
-------------------------------------------------------------------------------
Dimensions                    Occurrences    Starting from     Spacing
1 . Index vertical            3__            _________         1   Lines
2 . Index horizontal          5__            _________         2   Columns
3 . Index (H/V) V             2__            _________         0   Cls/Ls
·····························································
```

Example 6:

An example of using **Starting from**. The first dimension consists of 10 vertical occurrences starting from the index `I`. `I` is defined in the map editor with Natural data format/length N7 by default. The second dimension consists of 5 horizontal occurrences starting from the index 3.

```
·Array Definition·····························································
Name #1                                    Upper Bnds 10____ 5_____ 1_____
-------------------------------------------------------------------------------
Dimensions                    Occurrences    Starting from     Spacing
1 . Index vertical            10_            I________         1   Lines
2 . Index horizontal          5__            3________         2   Columns
0 . Index (H/V) V             1__            _________         0   Cls/Ls
·····························································
```

Example 7:

An example of making a two-dimensional display from a one-dimensional array. The array consists of 40 elements. It is displayed in two columns with 20 lines each. This is achieved by specifying 0 as the horizontal index.

```
·Array Definition·····························································
Name #1                                      Upper Bnds 40____ 1_____ 1_____
------------------------------------------------------------------------------
Dimensions                      Occurrences    Starting from     Spacing
1 . Index vertical              20_            ________          0   Lines
0 . Index horizontal            2__            ________          10  Columns
0 . Index (H/V) V               1__            ________          0   Cls/Ls
·····························································
```

AD - Attribute Definition

Attribute definition editing is performed as follows. Use the TAB key to move to the **AD** item in the field attribute window.

Press PF2 to invoke the following **Attribute Definition** window:

```
Children     :  99                        Years-Educ:  99
Family Status:  XXXXXXXXXX                Years-Comp:  99
Sex          :  X                         Vacation-D:  99
                                          Sick-Days :  99

                                    Hobby:  XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Address
-
State        :  XX         ··Attribute Definition···
Zip          :  99999      ·Representation         ·
City         :  XXXXXXXXXXX·Alignment              ·
Street/Number:  XXXXXXXXXXX·I/O Characteristics    ·
                           ·Mandatory Characters   ·
                           ·Length Characteristics ·
                           ·Upper/Lower Case       ·
                           ·Filler Character       ·
··Extended Field Editing·····························································
·Field= PERSONNEL.STREET                                                      ·
·Format= A Len= 20    Alphan. Len= 20    PM=                                  ·
·Rules:  0 Rule Editing? N Array:          Array Editing? N       Mode= Data   ·
·AD= FHWOIL      CD=    CV=                                 DY= ->  HE= ->    ·
·EM=                                                                          ·
·····························································
```

Select the desired item with the cursor keys.

Press ENTER and an additional selection window for each item selected appears.

Note: If the attribute definition character displayed is correct, use the ESC key and this character is not changed. The ESC key terminates AD editing.

Selecting Data Definitions

A map field can be defined by selecting a data definition from another Natural object. Data definitions are either field definitions in a Natural DDM (data definition module) or variable definitions.

To select a data definition, first select the object class. Valid object classes are:

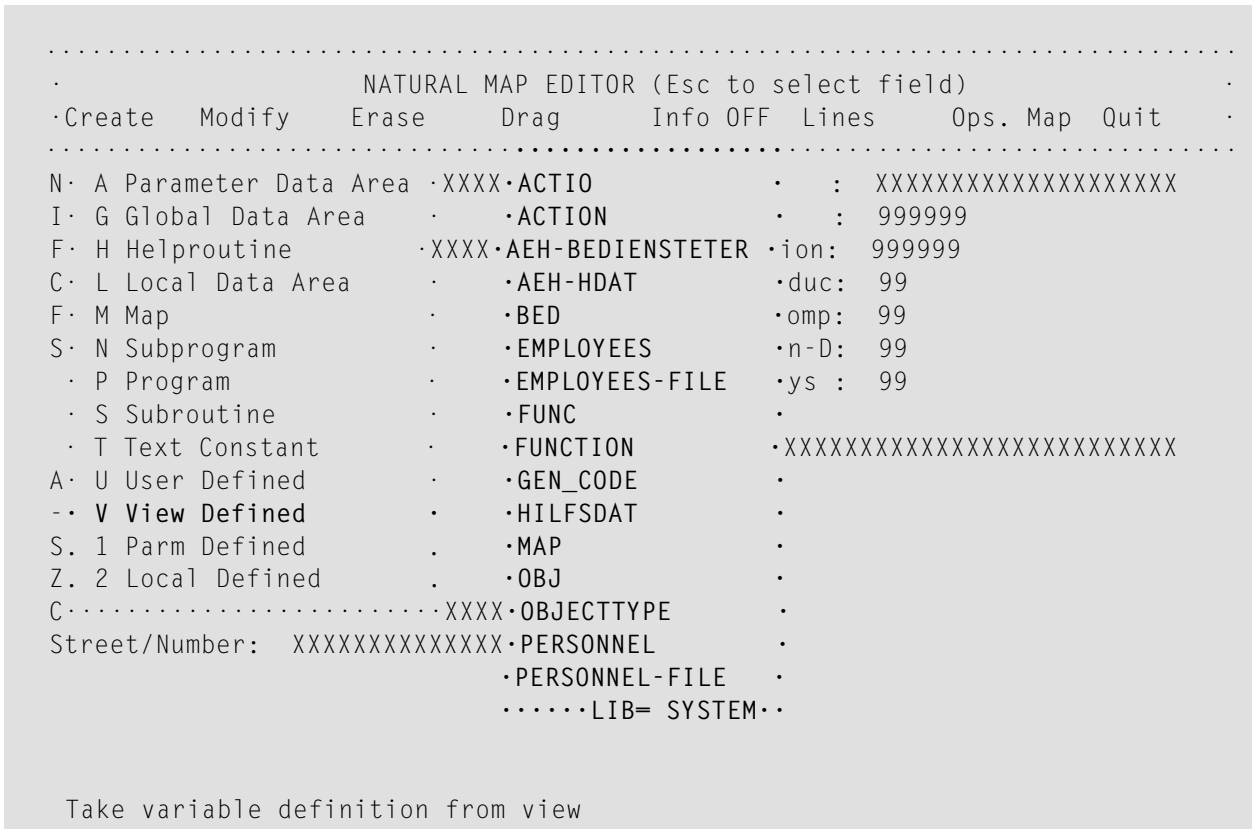
Object Class	Description
A	Parameter Data Area
G	Global Data Area
H	Helproutine
L	Local Data Area
M	Map
N	Subprogram
P	Program
S	Subroutine
V	View (DDM)

Programs, subroutines, subprograms and helproutines can only be used if they contain a `DEFINE DATA` statement.

For demonstration purposes, a Natural object of type DDM will be used. From the **MAP EDITOR** menu, select **Create** and then **View Defined**. Alternatively, enter `C` (for **Create**) and then `V` (for **View Defined**).

From the pop-up window that opens, select the library that contains the required DDM.

The following library list appears on the screen:



Select the desired DDM (for example, PERSONNEL) and press ENTER. The selected DDM is displayed in a window:

```
             *** Personnel Data Detail Display Function ***

Person Data                              Employment Data
-                                        -
Name         :  XXXXXXXXXXXXXXXXXXX      Job       :  XXXXXXXXXXXXXXXXXXX
Initial      :  X                        Salary    :  999999
First-Name   :  XXXXXXXXXXXXXX           Commission:  999999
Children     :  99                       Years-Educ:  99
Family Status:  XXXXXXXXXX               Years-Comp:  99
Sex          :  X                        Vacation-D:  99
                                         Sick-Days :  99

                                    Hobby:  XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Address
-
··PERSONEL·····························································
·  1  AA  PERSONNEL-NUMBER                 N  8.0    D                    ·
·  1  AA  PERSONAL-NUMMER                  N  8.0    D                    ·
·  1  AA  NUMERO-PERSONNEL                 N  8.0    D                    ·
·G 1  G1  PERSON                                                          ·
·  2  BA  NAME                             A   20  N D                    ·
·····························································

HD=PERSONNEL/NUMBER
```

This window displays, for the highlighted field, three additional information lines at the bottom of the screen.

These lines display the following information for the current field:

- Edit mask
- Header
- Comments

The field name consists of the DDM name concatenated with the field name by a period, for example:

personal.personnel-number

Select the fields which are to be included in your map and press ENTER.

The library window, listing the selected field in the "Field= " line appears and you are prompted to:

```
Position cursor and press Enter.
```

Position the cursor to the start of a field position and press ENTER.

Continue modifying fields as described under *Creating a User-Defined Variable*.

Repeat this procedure until all fields required for your map have been defined.

Defining Fields for a Parameter or Local Data Definition

New parameters or local variables can be created or existing ones can be modified by selecting the item **Parm Defined** or **Local Defined** from the **Create** menu and pressing ENTER.

The following topics are covered below:

- Parameter Definitions
- Local Data Definitions

Parameter Definitions

If you select the **Parm Defined** item, the following window appears:

```
··PARAMETER·····································································
·<CREATE>                                                                  ·
·TEST1                                    L     001:003                    ·
·TEST2                                    L     001:003,001:004            ·
·TEST3                                    L     001:003,001:004,001:005    ·
·····························································································
```

With this function, new parameters can be added and existing parameters can be modified.

If you want to modify an existing parameter (for example, TEST3), select it with your cursor and press ENTER. A window pops up prompting you to select the desired action: you can **Edit** or **Delete** the selected parameter, or **Cancel** the action.

If you select **Edit**, the following window appears, in which you can edit the selected parameter:

```
··PARAMETER·····································································
·Name.....: TEST3                                                          ·
·Format...: L                                                              ·
·                                                                          ·
·Dimension: 3                                                              ·
·           Lower Bnds  :  Upper Bnds                                      ·
·1. Index.:          1              3                                      ·
·2. Index.:          1              4                                      ·
·3. Index.:          1              5                                      ·
·····························································································
```

If you want to create a new parameter, select the **<CREATE>** option. The same window appears, but this time it is empty so that you can make the specifications for the parameter to be created.

When you leave the above window, a further window pops up prompting you whether you want to either save your modifications/specifications or cancel the action.

Local Data Definitions

If you select the "Local Defined" item, the following window appears:

```
··LOCAL·····························································
·<CREATE>                                                           ·
·TESTA                                    L     001:003             ·
·TESTB                                    L     001:003,001:004     ·
·TESTC                                    L     001:003,001:004,001:005     ·
·····························································
```

With this function, new local variables can be added and existing variables can be modified. Local variables can be used to pass values from one processing rule to another.

If you want to modify an existing local variable (for example, TESTC), select it with your cursor and press ENTER. A window pops up prompting you to define the desired action: you can Edit the selected variable, Delete it, or Cancel the function.

If you define the action Edit, the following window appears, in which you can edit the selected local variable:

```
··LOCAL·····························································
·Name.....: TESTC                                                   ·
·Format...: L                                                       ·
·                                                                   ·
·Dimension: 3                                                       ·
·           Lower Bnds  :  Upper Bnds                               ·
·1. Index.:          1              3                               ·
·2. Index.:          1              4                               ·
·3. Index.:          1              5                               ·
·····························································
```

If you want to create a new local variable, select the <CREATE> option. The same window appears, but this time it is empty so that you can make the specifications for the variable to be created.

When you leave the above window, a further window pops up prompting you whether you want to either save your modifications/specifications or cancel the function.

Map Profile

This section describes the process of defining the map settings (profile) for a map.

The Map Profile is invoked by selecting "Prof. Map" from the "Ops. Map" selection list. The following screen is displayed:

```
.....................................................................................
.                    NATURAL MAP EDITOR (Esc to select field)                       .
.Cr..Map Settings.....................................................................
   .Format                     Context                                             .
   .-------------------------  -------------------------                           .
Add.Page Size .....:  24                                                           .
-  .Line Size .....:  79       WRITE Statement ...: N                              .
Sta.Layout .......:                                                                .
Zip. dynamic ......: N         Helproutine ......:                                 .
Cit.Zero Print ....: N         Help Parameter ....:                                .
Str.Upper Case ....: Y          as field default .: N                              .
   .Documentation Skip ...: N         Help Text .........: N                        ↩
.
   .Decimal Char ..: .          Position Line ....:  0    Column :   0            .
   .Standard Keys .: N         AutoRuleRank ......:  1                           .
   .Right Justify .: Y                                                           .
   .Print Mode ....:           Filler Characters                                   .
   .                           -------------------------                           .
   .Control Var ...:           Optional, Partial .: _                              .
   .                           Required, Partial .: _                              .
   .Field Sensitive: N         Optional, Complete : _                              .
                               Required, Complete : _                              .
  .....................................................................................

Modify map profile
```

This map profile screen comprises map profile settings and filler characters.

The following topics are covered below:

- Map Profile Settings

- Filler Characters

Map Profile Settings

The following map profile settings can be used:

Entry	Explanation
Page Size	The number of map lines to be edited (1 - 250); if "Std Keys" (see below) is set to "Y", the number of lines is restricted to 3 - 250. For a map which is output with a `WRITE` statement (see the entry "WRITE Statement" in the Context column of the Map Settings), you specify the number of lines of the logical page output with the `WRITE` statement, not the map size. Thus, the map may be output several times on one page.
Line Size	The number of map columns to be edited (5 - 249).
Layout dynamic	The name of a map source definition which contains a predefined layout. **Y** - Specifies the layout to be dynamic. The dynamically used layout does not become a fixed part of the map at compilation time, but is executed at runtime. Thus, subsequent modifications of a layout map become effective for all maps using that layout map. If the layout map includes user-defined variables, you have to define these parameters in the map using the layout map. Input fields and modifiable fields in the layout map are not open at runtime. Parameters can be added by pressing F9 within the Field and Variable Definitions function. **N -** Specifies the layout to be static. The static layout is copied into the source area when a map is initialized. Filler characters are not transferred; "N" is the default setting.
Zero Print	**Y** - displays a field value of all zeros as one zero only. **N** - displays a zero value as blanks. This value is copied into the field definition when a new field is created and can be modified for individual fields using the extended field editing function.
Upper Case	**Y -** indicates that all input entered for fields at map execution time is to be converted to upper case. **N -** indicates that no lower to upper case conversion is to be performed. This value is copied into the field definition when a new field is created, and can be modified for individual fields using the extended field editing function.
Manual Skip	**Y** - Does *not* automatically move the cursor to the next field in the map at execution time even if the current field is completely filled. **N** - Moves the cursor automatically to the next field in the map at execution time when the current field is completely filled; "N" is the default setting.
Decimal Char	The character to be used as the decimal notation character. This character can only be changed with the `GLOBALS` command.

Entry	Explanation
Standard Keys	**Y -** leaves the last two lines of the map empty so that function-key specifications can be entered at execution time. **N -** causes all lines to be used for the map.
Right Justify	The type of field justification to be used for numeric and alphanumeric fields taken from data definitions in other Natural objects: **Y** - right justified **N** - left justified
Print Mode	The default print mode for variables: **C -** indicates that an alternative character set is to be used (special character table). **I -** indicates inverse print direction. **I C -** indicates standard print direction. This value is copied into the field definition when a new field is created.
Control Var	The name of a control variable, the content of which determines the attribute characteristics of fields and texts that have the attribute definition AD=Y or (Y). The maximum length of a control variable is limited to 8 characters. The control variable referenced in the map must be defined in the program using that map.
Field Sensitive	**Y** - specifies that processing rules attached to map fields are executed on a field-by-field basis; that is, immediately after you have left a given field. **N** - specifies that no field sensitivity is to be defined for the map, which means that the map is not to be processed until you have entered all necessary values in the map fields and pressed ENTER (or any F key).
WRITE Statement	**Y** - Marking this field with **Y** produces a `WRITE` statement at the end of the map definition process. The resulting map may then be invoked from a Natural program using a `WRITE USING FORM` statement. Empty lines at the end of the map are automatically deleted so that the map can be output several times on one page. **N** - Marking this field with **N** will cause the result of the map definition process to be an `INPUT` statement. The resulting map may then be invoked from a Natural program using an `INPUT` statement.
Helproutine	The name of a helproutine which is invoked at runtime when the help function is invoked for this map (global help for map). For detailed explanation of the syntax, see the parameter "Help" in the section *Modifying a User-Defined Variable - Field Editing*.
Help Parameter	The help parameter which is invoked at execution time when the help function is invoked. **Note:** A maximum of 20 help parameters are possible. If you enter more, they are ignored.
as field default	**Y** - specifies that the helproutine for the map is to apply as default to each individual field on the map, which means that the name of each field is passed individually to the helproutine. **N** - specifies that the name of the map is passed to the helproutine.

Entry	Explanation
Help Text	**Y** - specifies that this map is actually help text; default = "N".
Position Line Column	The position where the help map is to appear on the screen at execution time.
AutoRuleRank	The rank (priority) assigned to Predict automatic rules when they are linked to the map during field definition. Default is 1.

Filler Characters

Filler characters can be assigned to indicate whether information for a field is mandatory and whether the field must be completely filled:

Field Type	Explanation
Optional Partial	Is not mandatory, need not be completely filled.
Required Partial	Mandatory, need not be completely filled (AD=E).
Optional Complete	Is not mandatory, must be completely filled (AD=G).
Required Complete	Mandatory, must be completely filled (AD=EG).

Filler characters may also be defined for individual fields using the extended field editing function. For definition of field types, see also the session parameter `AD`.

Post Assignment

A field which has been previously defined in the screen layout of a map may be assigned the field name and field attributes of a DDM field definition or a `DEFINE DATA` definition.

Note: Duplicate field names are only allowed for fields defined as "output only fields".

A map field which has been created from a DDM must be redefined by using the appropriate DDM field definition from this DDM or a `DEFINE DATA` definition. However, this is only possible if it is the same database field.

Post assignment of a DDM field definition can only be done by deleting the respective field definition from the map and selecting a new DDM field definition from a DDM or a `DEFINE DATA` definition (see also ***Selecting Data Definitions***).

Post assignment cannot be used for view arrays (in a DDM field definition or a `DEFINE DATA` definition) if one or more dimensions of that array are smaller than the dimensions of the array in the layout.

Field-Sensitive Processing

In the map profile, you can specify whether a map is to be processed after you have entered all necessary values in the map fields and pressed ENTER, or whether the processing rules attached to the map fields are to be handled on a field-by-field basis.

When handled on a field-by-field basis, the processing rules attached to a field are executed immediately as soon as you have filled this field entirely or you move the cursor to another position; they are executed in the same order as without field sensitivity.

The following topics are covered below:

Advantages of Field-Sensitive Processing

The advantages of field-sensitive processing are described in the following section.

Note: To exploit the advantages of field-sensitive processing, you need to adapt your existing processing rules accordingly. For more information on processing rules, see ***Rule Editing - Processing Rules***.

Dynamic Fillings of Fields (Based on User Input)

Without field-sensitive processing, the application would wait until all the fields have been filled in and you have pressed ENTER, before checking if the data entered are valid. In the worst case, you would have wasted time and effort filling in all the fields, before being informed of the error.

However, with field-sensitive processing, as soon as you have entirely filled a field or leave the field by either pressing END or SHIFT+END or moving the cursor with one of the cursor movement keys, the data input is checked immediately, and other fields can be pre-filled automatically (via `REINPUT FULL`) with data based on the value entered in the previous field. Depending on this value, these data may also be the result of a database query.

The information under which condition a field was left can be retrieved from the `*PF-KEY` system variable, which contains either "FULL" (if the field was entirely filled) or the name of the key most recently pressed as shown in the following table.

Pressed Key	*PF-KEY
LEFT-ARROW	LEFT
RIGHT-ARROW	RIGT
UP-ARROW	UP
DOWN-ARROW	DOWN
PAGE-UP	PGUP
PAGE-DOWN	PGDN
TAB	TAB
HOME	HOME
END	END
BACKTAB	BTAB
ENTER	ENTR
Field is full	FULL

Improved User Guidance

With field-sensitive processing, you can be guided from field to field, depending on the values entered. The cursor can skip fields that are pre-filled and take you to the next input field.

As messages are displayed immediately after a field has been checked, you are informed more promptly and more precisely as to what to do.

Security Based on User Input

Depending on the value entered in a particular field, you can be prompted, for example, for a special password to access the requested information.

Rapid Data Entry

All these mechanisms described above make interaction with the application much faster and more efficient.

Defining a Map as Field-Sensitive

To be able to use field-sensitive processing, you first need to define a map as being field-sensitive by setting Field Sensitive to "Y" in the Map Profile. A window appears asking you whether field sensitivity should be with or without automatic field recognition:

```
·····························NATURAL MAP EDITOR (Esc to select field)·····
·Cr··Map Settings·····Format·····Context·····
Add·Page Size .....:  24
-  ·Line Size .....:  79       WRITE·····
Sta·Layout ........:                 · Automatic Field Recognition ? ·
Zip· dynamic ......: N         Help ·          YES   NO             ·
Cit·Zero Print ....: N         Help ·····
Str·Upper Case ....: Y          as field default .: N
   ·Documentation Skip ...: N        Help Text .........: N
·
   ·Decimal Char ..: .          Position Line ....:  0    Column:   0
   ·Standard Keys .: N         AutoRuleRank ......:  1
   ·Right Justify .: Y
   ·Print Mode ....:           Filler Characters
   ·                           ·····
   ·Control Var ...:           Optional, Partial .: _
   ·                           Required, Partial .: _
   ·Field Sensitive: Y         Optional, Complete : _
                               Required, Complete : _
   ·····

Modify map profile
```

Field sensitivity within a Natural map can be achieved in two different ways:

1. With automatic field recognition, which means that you need not code any conditions for activating the right processing rule.

   Note: If you upload such a map to the mainframe, the Natural mainframe map editor does not understand the syntax of the uploaded map, because of new syntax extensions.

2. Without automatic field recognition, which means that you are responsible for being informed about which field was left with the last keystroke. You can achieve this by using the system variables `*CURS-COL` and `*CURS-LINE` in your Natural program.

   Note: If you upload such a map to the mainframe, `STOW` the map on the mainframe, and the same source code will be generated.

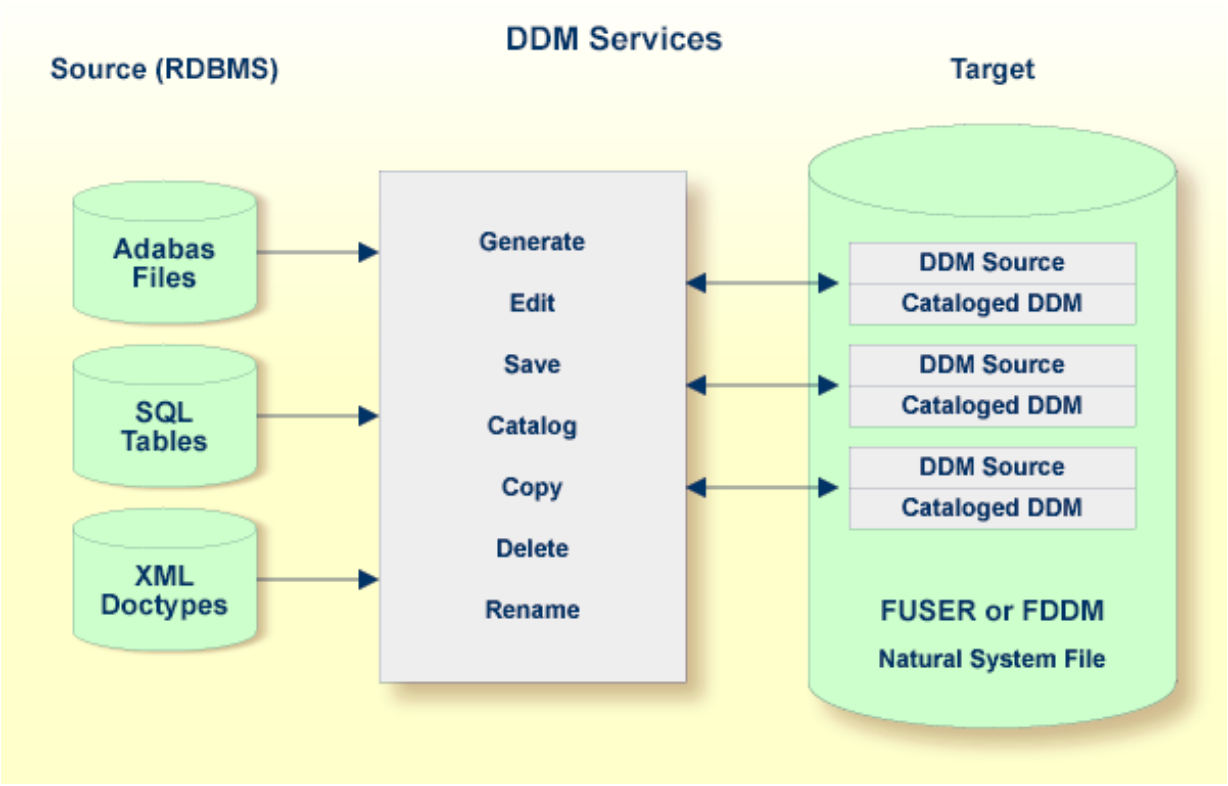
VI DDM Services

DDM Services are used to create, maintain and delete Natural data definition modules (DDMs).

19 Principles of Operation

DDM Services are used to create a Natural DDM from a database file or from another DDM. A Natural object (for example, a program or a data area) can only access a database file if a corresponding DDM has been created for this file and saved as a source object and a cataloged object. For further details on Natural DDMs, see *Data Definition Modules - DDMs* in the *Programming Guide*.

The following graphic illustrates the main features and basic principles of operation when processing DDMs with DDM Services:



Storing DDMs - FDDM System File

DDMs reside either in individual Natural user libraries or system libraries contained in the system file FUSER, FNAT or FDDM. The FDDM system file is a separate container that collects all DDMs available in your Natural system environment. The FDDM system file is activated by setting the Natural profile parameter `FDDM` in the NATPARM parameter file.

Caution: If the FDDM system file has been activated, you can only store and access DDMs in the FDDM system file; DDMs contained in libraries in the FNAT or FUSER system file can then no longer be accessed.

DDMs are not restricted to files stored in an Adabas database. Some options described in the *DDM Services* documentation only apply to Adabas and can be ignored if a different database management system is used.

Related Topics:

- *System File FDDM* in the *Operations* documentation
- *FDDM - Natural System File for DDMs* in the *Parameter Reference* documentation

Restrictions of Use

The section below describes the restrictions that can apply to using DDM Services functions under:

- Predict
- Natural Security

Predict

To guarantee data integrity for DDMs defined in Predict, the Predict administrator can restrict the use of DDM Services functions for editing, copying, creating, deleting and renaming, for which equivalent functions are provided by Predict.

In principle, we strongly recommend that you do not use *any* DDM Services functions that can alternatively be performed by Predict.

For further information, contact your Predict system administrator.

Natural Security

Access to DDMs can be restricted when Natural Security has been installed. Within the DDM security profile, there may be a definition of whether a DDM may be modified only by specific users (DDM modifiers) or the owners of the security profile.

For further information, see *Protecting DDMs On Linux And Windows* in the *Natural Security* documentation.

20 Invoking and Terminating DDM Services

This section provides instructions for invoking and terminating DDM Services.

To invoke DDM Services

- From the Natural main menu, choose **Services** and, from the selection window, choose **DDM Services** or enter a `D`.

  Or:

  From the Natural main menu, choose **Direct** and, in the **Direct Command** window, enter the following:

  ```
  SYSDDM
  ```

  The **DDM Services** screen appears as shown in the example below:

```
                                                                               ↩
 2004-09-09                       DDM Services             Library: SYSTEM   ↩
 17:15:21               V 6.1.1 Pl 9   Software AG 2004     DBID   :          ↩
 User: SAG                                                 FNR    :          ↩
+-------------------------------------------------------------------------------+
¦   Library         DDM Maintenance           Services Profile         Quit   ↩
 ¦
+-------------------------------------------------------------------------------+
                                                                               ↩
                                                                               ↩
                                                                               ↩
                                                                               ↩
                                                                               ↩
                                                                               ↩
                                                                               ↩
                                                                               ↩
Logon to DDM library                                                           ↩
```

The menus provided on the **DDM Services** screen are explained in the relevant sections of the *DDM Services* documentation.

To terminate DDM Services

- On the **DDM Services** screen, choose **Quit**.

  The Natural main menu appears.

21 Using DDM Maintenance Functions

The functions provided in the DDM maintenance window are used to create (copy), display, edit, rename or delete a DDM.

Listing and Performing Maintenance Functions

This section describes how to list and perform DDM maintenance functions for the DDMs listed in the DDM maintenance window.

For a description of the **<CREATE>** menu option, see the section *Creating DDMs*.

To list and perform a DDM maintenance function

1 In the DDM maintenance window, position the cursor at the DDM you want to process and press F2.

A selection window appears that lists the valid functions and corresponding function codes:

```
+----------+
¦ C Copy   ¦
¦ D Delete ¦
¦ E Edit   ¦
¦ L List   ¦
¦ N Rename ¦
+----------+
```

For an explanation of the functions, see *Description of Maintenance Functions*.

2 Enter a function code for the DDM to be processed by using either of the following methods:

- In the selection window, position the cursor at the function code that corresponds to the required function and press ENTER.
- Or:
  Press ENTER to leave the selection window, enter a valid function code next to the required DDM, and press ENTER again.

The function code is entered in the DDM maintenance window, next to the DDM to be processed.

Depending on the function code entered, the corresponding function is either performed immediately or a window appears asking you to either confirm the action or enter a DDM name.

You can also enter more than one function code. In this case, enter all function codes and press ENTER to process one DDM after the other.

For an example of how to perform a function, see *Copying DDMs*.

Description of Maintenance Functions

The functions provided in the DDM maintenance window apply to the DDMs contained in the current Natural library and/or system file.

The following table describes the functions and function codes that can be entered in the DDM maintenance window.

Function Code	Function
C	Copy DDM. Copies the source and cataloged object of a DDM. If Predict is installed: In addition, copies XREF data if XREF entries exists for the DDM in the FDIC system file.
D	Delete DDM. Deletes the source and cataloged object of a DDM.
E	Edit DDM. Invokes the DDM editor and reads a DDM source into the editing area. See also ***Using the DDM Editor***.
L	List DDM. Invokes the **List** screen and displays the DDM source. (This function does not invoke the DDM editor.) For information on the columns of fields displayed on the **List** screen, see ***Columns of Field Attributes***.
R	Rename DDM. Renames the source and cataloged object of a DDM. If Predict is installed: In addition, copies XREF data if XREF entries exists for the DDM in the FDIC system file. For the naming conventions that apply when renaming a DDM, refer to *Object Naming Conventions* in the *Using Natural* documentation.

22 Creating DDMs

This section describes how to create a DDM by either copying DDMs or creating DDMs directly from the field definitions in a database. In addition, it provides information on how to generate multiple DDMs from an SQL database.

Copying DDMs

This section describes how to create a new DDM from an existing one.

If you want to copy DDMs between different libraries, database files, and/or hardware platforms, see also *Maintaining DDMs in Different Environments*.

Predict Installations:

If XREF entries exists for the DDM in the FDIC system file, they will also be copied for the new DDM.

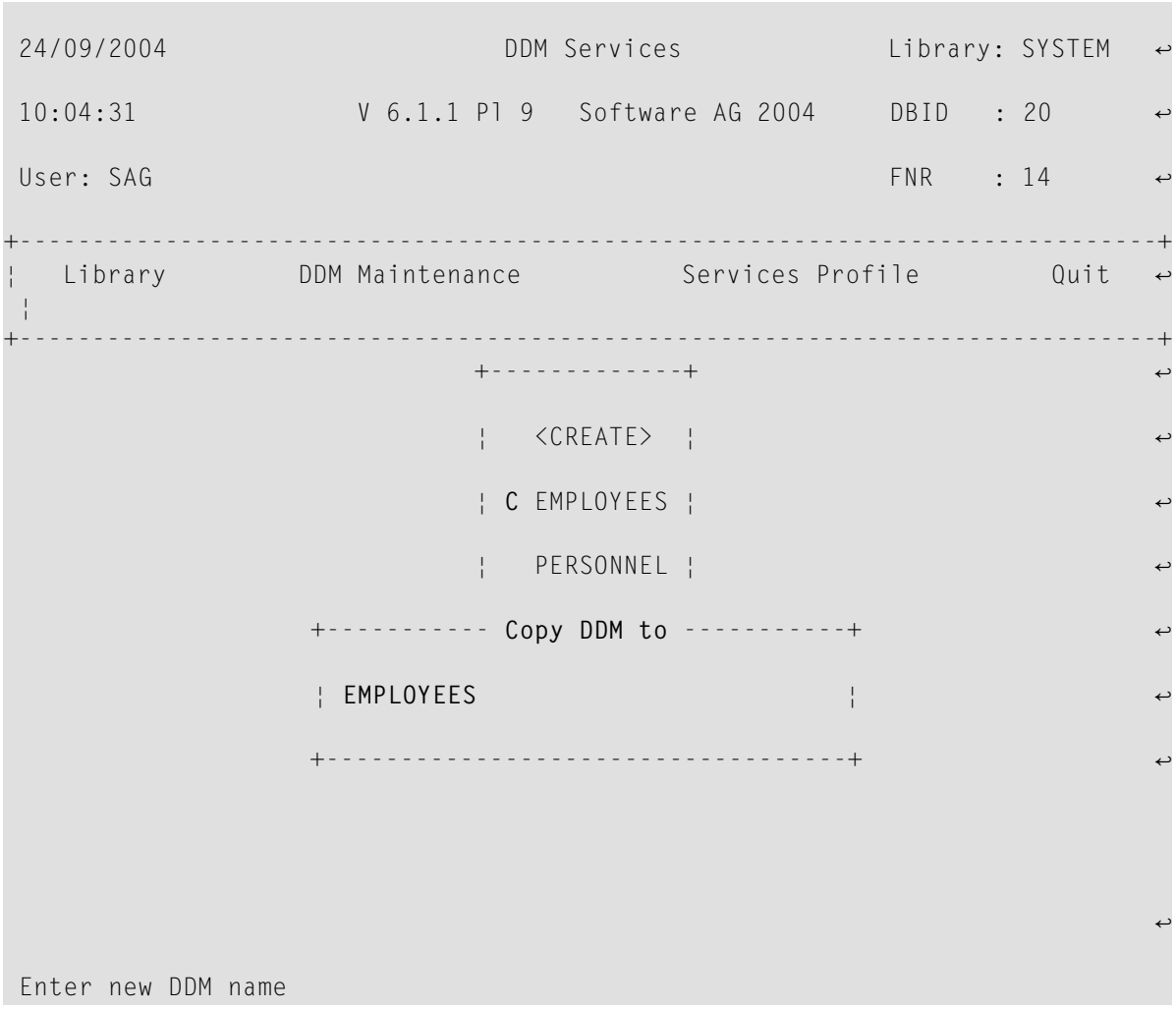
To copy a DDM

1 On the **DDM Services** screen, choose **DDM Maintenance**.

The DDM maintenance window appears with a list of all DDMs available.

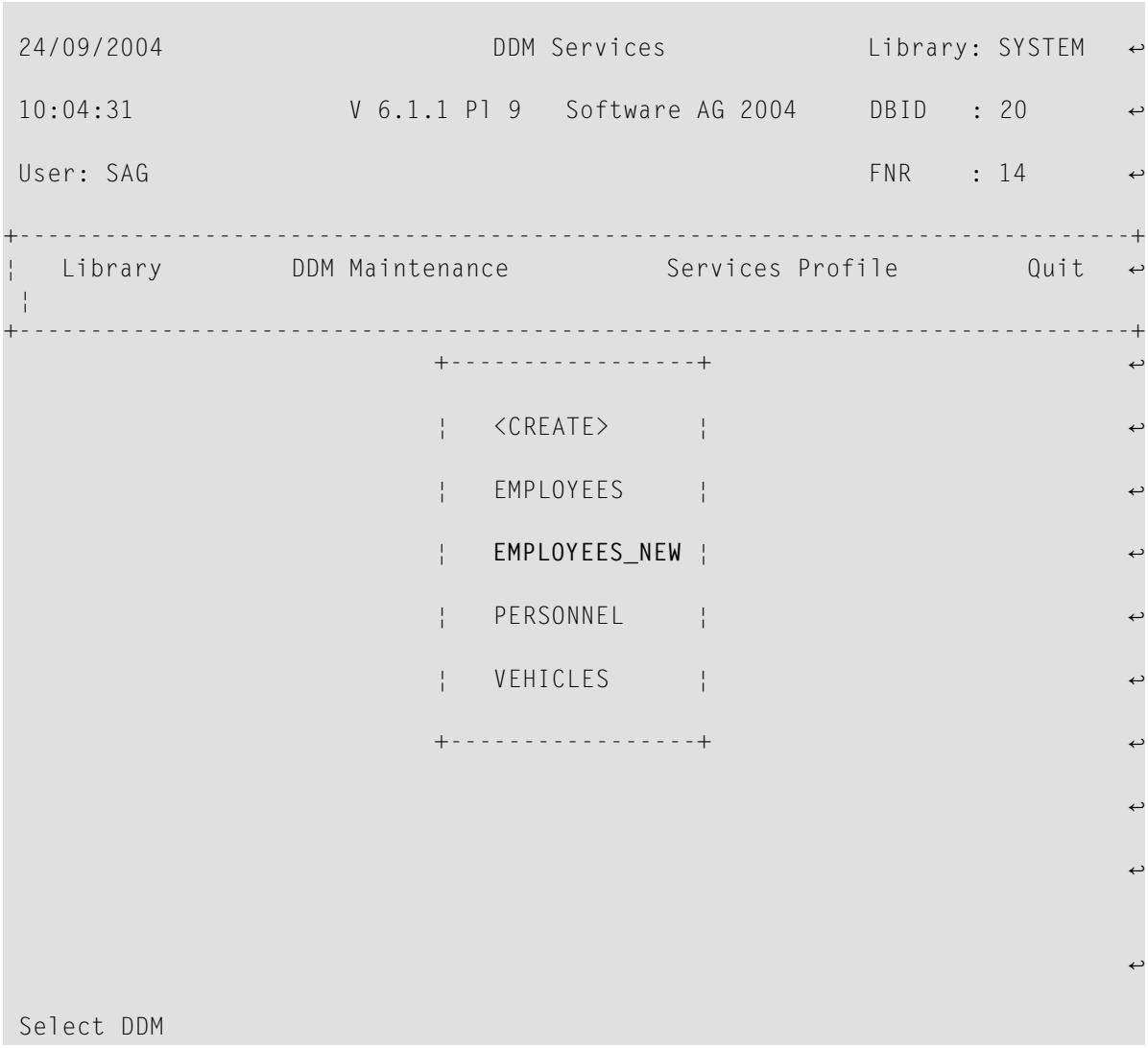
2 In the DDM maintenance window, position the cursor at the DDM you want to copy (in the example below: `EMPLOYEES`), enter the function code `C` and press ENTER.

The **Copy DDM to** window appears preset to the name of the DDM to be copied as shown in the example below:



3 In the **Copy DDM to** window, replace the preset name by a new one and press ENTER.

The new DDM (here: EMPLOYEES_NEW) is saved as a source and a cataloged object and is added to the list of DDMs:



<CREATE> from Adabas

This section describes how to create a new DDM from an Adabas database by using the **<CREATE>** menu option.

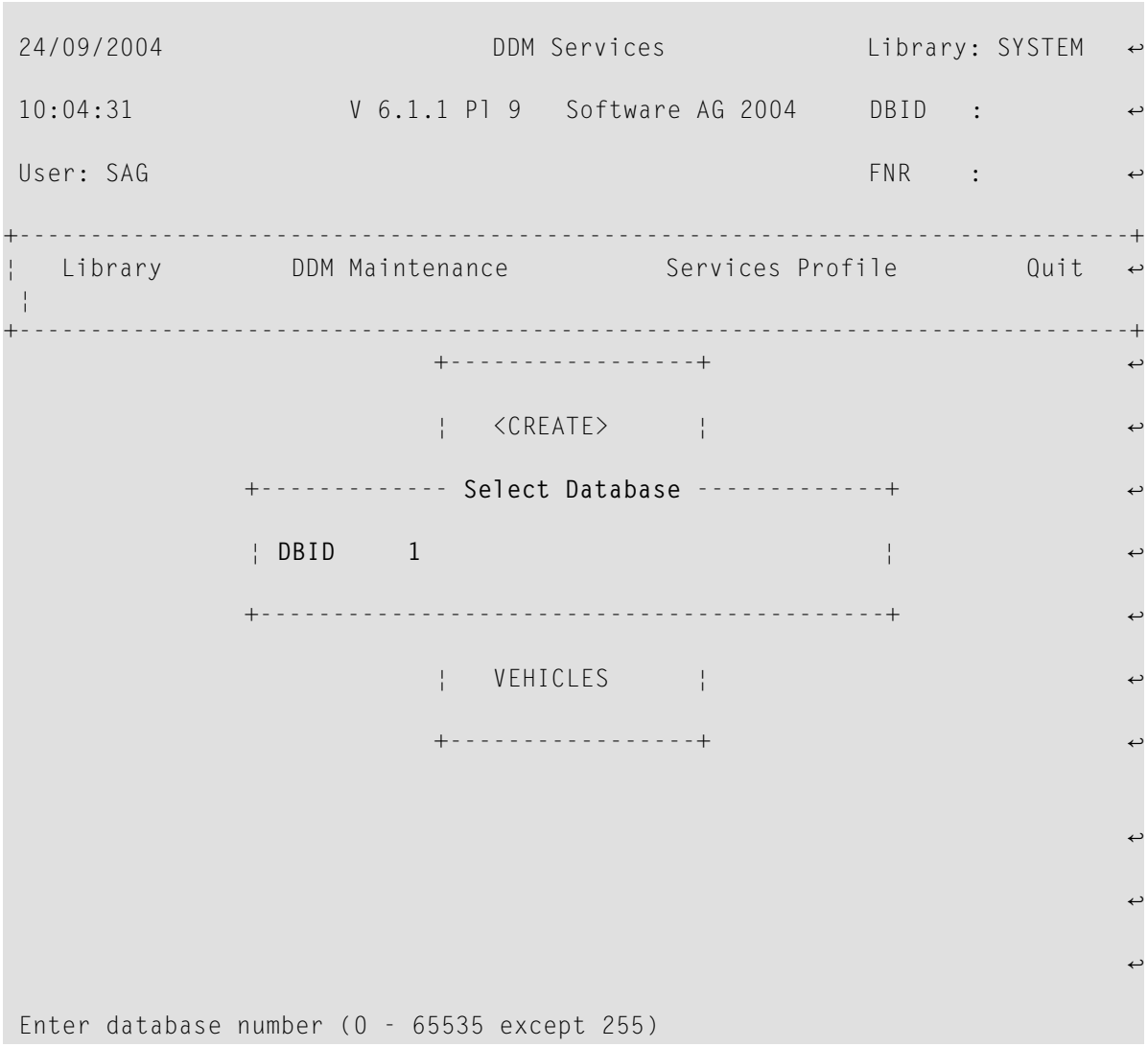
To create a DDM from an Adabas database

1 On the **DDM Services** screen, choose **DDM Maintenance**.

The DDM maintenance window appears.

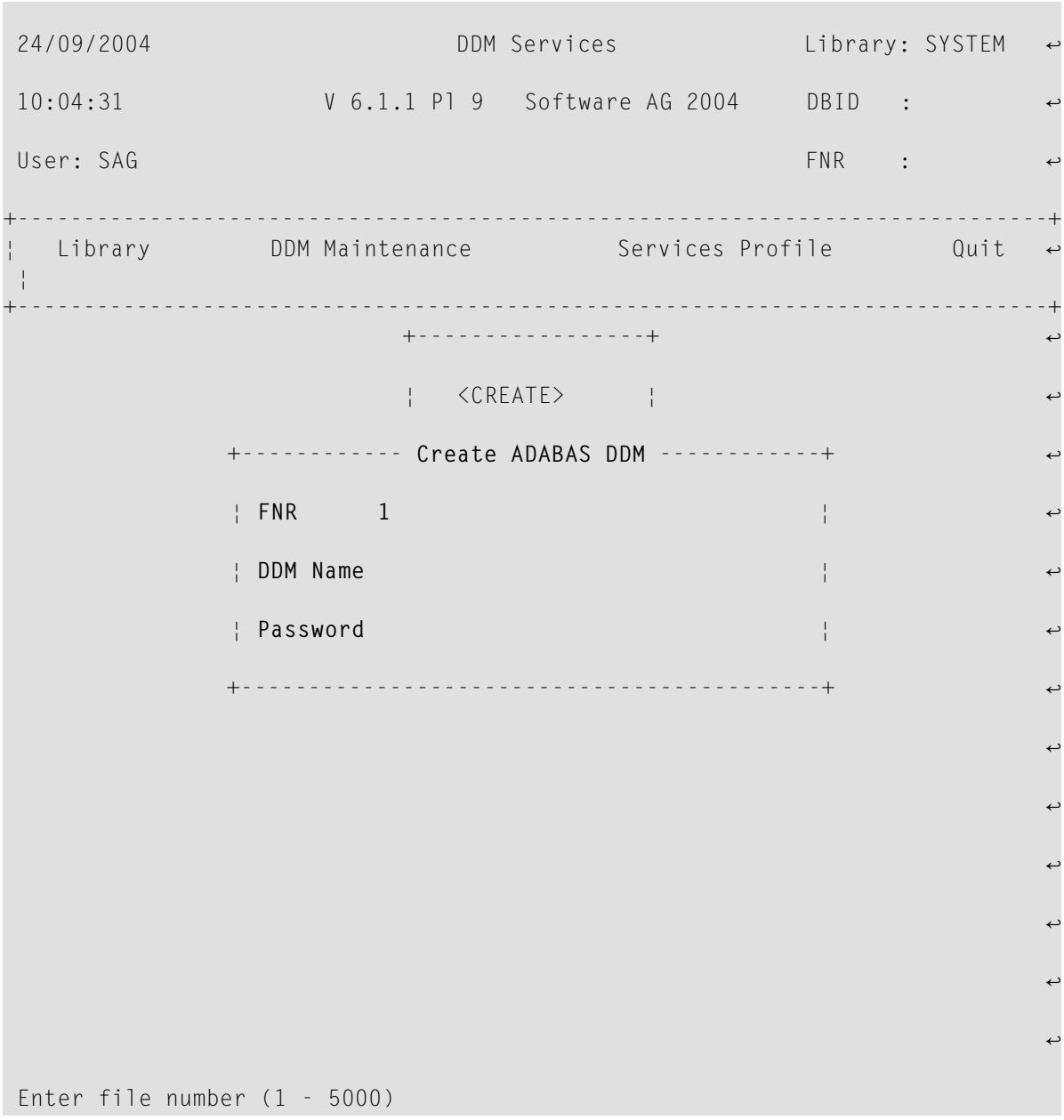
2 From the top of the DDM maintenance window, choose **<CREATE>**.

The **Select Database** window appears as shown in the example below:



3 Enter the database ID (**DBID**) of the Adabas file for which a DDM is to be created. Valid values are `0` - `65535`, except `255`. If you enter a `0` (zero), the database ID specified with the Natural profile parameter `UDB` (see the *Parameter Reference* documentation) of the NATPARM parameter file is used.

If the specified DBID identifies an Adabas database, the **Create ADABAS DDM** window appears:



4 Enter the file number (**FNR**) of the database file for which a DDM is to be created (valid values: 1 - 5000) and the name to be assigned to the DDM. If access to the Adabas file is protected, a password is required.

5 Press ENTER.

If the specified database and the file are available, the DDM editor is invoked and the fields contained in that database file are read into the editing area.

If the specified database is not active or cannot be accessed or if the file does not exist, a corresponding error message is issued. Nevertheless, if you press ENTER, you can still open an

empty DDM editor screen, enter new field attribute definitions and save the DDM source. However, in this case you cannot check any definitions against the database file description.

6 If required, edit the DDM: see the section *Using the DDM Editor*.

7 After editing, press ESC and, from the **MISC** menu choose **EXIT (with STOW)**. See also *Saving and Cataloging a DDM*.

The syntax of the DDM source is checked and the DDM is saved as a source and a cataloged object.

<CREATE> from SQL

This section describes how to create a DDM from an SQL database using the **<CREATE>** menu option.

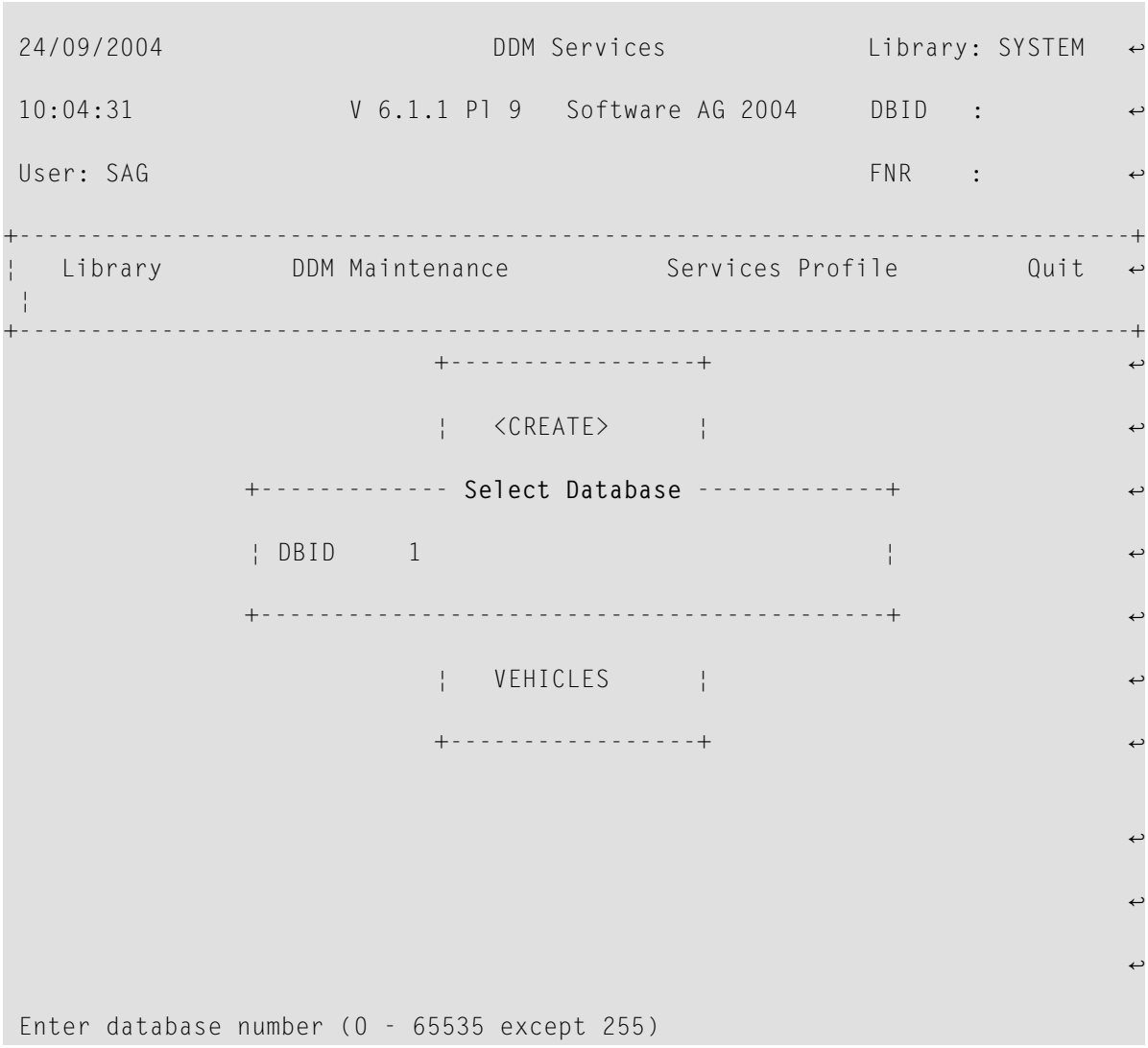
To create a DDM from an SQL database

1 On the **DDM Services** screen, choose **DDM Maintenance**.

The DDM maintenance window appears.

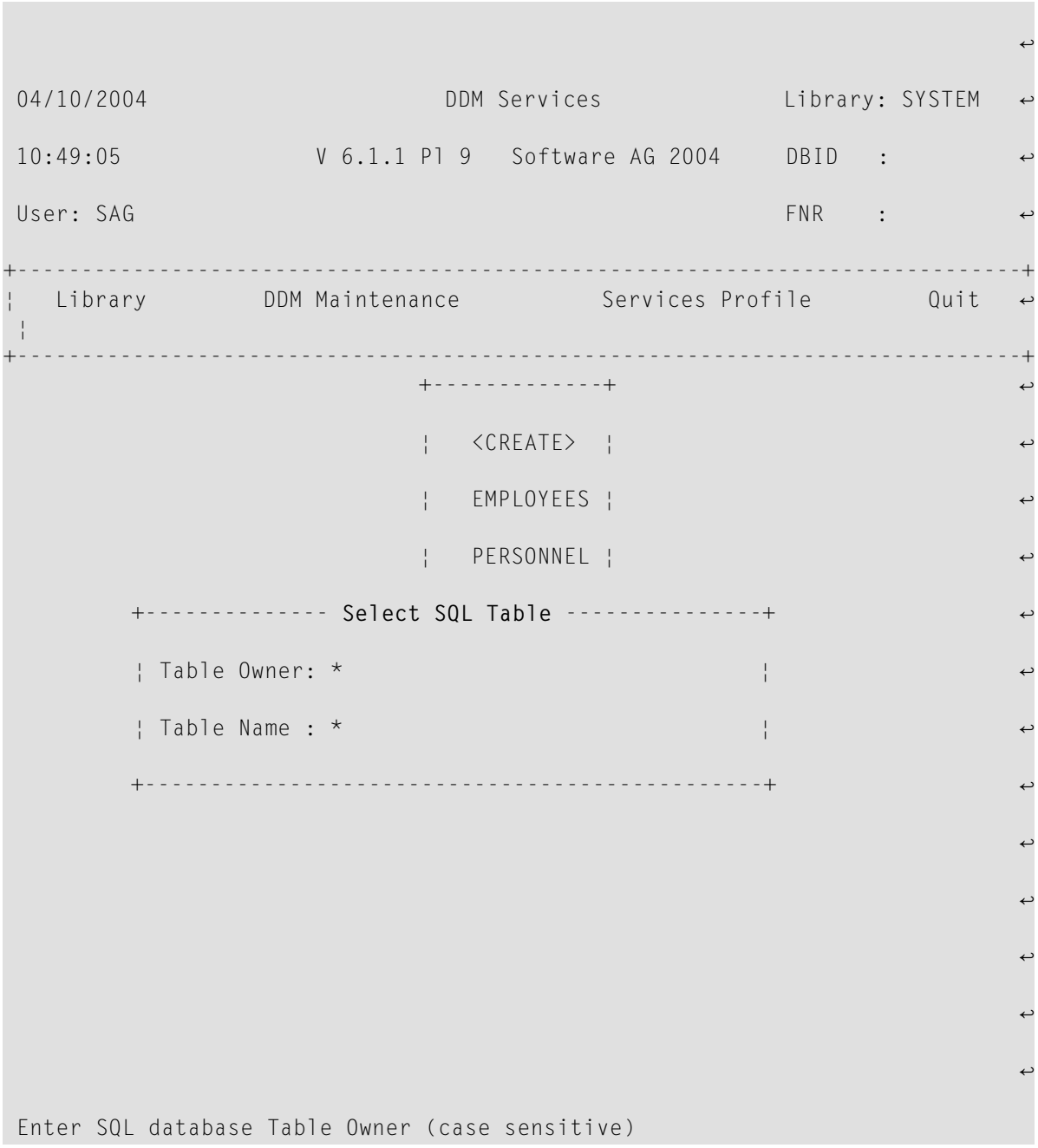
2 From the top of the DDM maintenance window, choose **<CREATE>**.

The **Select Database** window appears as shown in the example below:



3 Enter the database ID (**DBID**) of the SQL table for which a DDM is to be created. Valid values are `0` - `65535`, except `255`. If you enter a `0` (zero), the database ID specified with the Natural profile parameter `UDB` (see the *Parameter Reference* documentation) of the NATPARM parameter file is used.

If the specified DBID identifies an SQL database, the **Select SQL Table** window appears:



4 Enter the name of the table owner and the name of the table for which a DDM is to be created.

To specify a range: use asterisk (*) to list all tables (this is the default setting) or use asterisk (*) notation to list particular tables, for example, `AB*` selects all SQL tables with names that start with AB.

Depending on your SQL database settings, if you are accessing this SQL database for the first time in this session, the **Database Logon** window appears.

Enter the user ID and the password specified for the database and choose ENTER.

If the specified table exists, a DDM is created from this table.

If no such table exists or if you have specified a range of table owners and/or table names, the **Import SQL Table Contents** window appears from which you can select the required SQL table as shown in the example below.

```
 04/10/2004                       DDM Services             Library: SYSTEM   ↩
 11:18:15               V 6.1.1 Pl 9   Software AG 2004     DBID   :          ↩
 User: SAG                                                  FNR    :          ↩
+-------------------------------------------------------------------------------+
¦   Library         DDM Maintenance           Services Profile         Quit   ↩
 ¦
+-------------------------------------------------------------------------------+
    +--------------------- Import SQL Table Contents ----------------------+ ↩
     ¦ Table Owner                      Table Name                    Type  ¦ ↩
    ¦ ================================================================  ¦ ↩
     ¦ DEMO                             AUTOMOBILES                     T  ¦ ↩
     ¦ DEMO                             EMPLOYEES                       T  ¦ ↩
     ¦ DEMO                             SALARY                          T  ¦ ↩
     ¦ SAG                              YACHT                           T  ¦ ↩
     ¦ SAG                              ALLDATA                         T  ¦ ↩
    +-----------------------------------------------------------------------+ ↩
                                                                             ↩
 Select SQL table
```

Note: A table catalog is not provided for an SQL database that is accessed through an ODBC interface. In this case, you can only specify a table name but not a table owner.

5 Press ENTER.

The DDM editor is invoked and the DDM generated from the selected table is read into the editing area.

A name is generated automatically for the DDM. It is a combination of the table owner and the table name and cannot be changed. For example, if the table owner's name is `SAG` and the table name is `TEST`, the DDM name is `SAG-TEST`.

If the specified database is not active or cannot be accessed or if the file does not exist, a corresponding error message is issued. Nevertheless, if you press ENTER, you can still open an empty DDM editor screen, enter new field attribute definitions and save the DDM source. However, in this case you cannot check any definitions against the database file description.

6 If required, edit the DDM: see the section *Using the DDM Editor*.

7 After editing, press ESC and, from the **MISC** menu choose **EXIT (with STOW)**. See also *Saving and Cataloging a DDM*.

The syntax of the DDM source is checked and the DDM is saved as a source and a cataloged object.

Creating Multiple DDMs from SQL

The Natural program DDMGEN (supplied in the Natural system library SYSTEM) provides the option to generate multiple DDMs simultaneously from SQL tables without using the DDM editor.

Note: Before you start the creation process, log on to the SQL database.

To execute DDMGEN

1 Enter the following direct command:

```
DDMGEN
```

The **SQL DDM Generation** screen appears where you can fill the fields required to generate a DDM from an SQL table as demonstrated in the example below:

```
SQL DDM Generation
==================

DDM Library  : DDMTEST
DDM DBID     : 210
Table Owner  : QA*
Table Name   : *
Replace (Y/N): N
```

Enter the name of the library where you want to create the DDMs and enter the name of a table and/or specify a range as described in the previous section.

2 Press ENTER to execute the program.

Status messages appear at the bottom of the screen that indicate which DDM is generated from which SQL table. The DDMs generated are saved as source and cataloged objects in the specified library.

<CREATE> from Tamino

This section describes how to create a DDM from a Tamino database using the **<CREATE>** menu option.

To create a DDM from a Tamino database

1 On the **DDM Services** screen, choose **DDM Maintenance**.

The DDM maintenance window appears.

2 From the top of the DDM maintenance window, choose **<CREATE>**.

The **Select Database** window appears as shown in the example below:

```
01/10/2004                      DDM Services             Library: SYSTEM   ↩

15:50:58              V 6.1.1 Pl 9  Software AG 2004     DBID   :
User: SAG                                                FNR    :          ↩

+-------------------------------------------------------------------------+
¦   Library         DDM Maintenance           Services Profile         Quit   ↩
 ¦
+-------------------------------------------------------------------------+
                             +-----------------+                           ↩

                             ¦   <CREATE>      ¦                           ↩

                +------------ Select Database -------------+               ↩

                ¦ DBID     1                                ¦               ↩

                +-----------------------------------------+               ↩

                             ¦   EMPLOYEES_NEW ¦                           ↩

                             ¦   PERSONNEL     ¦                           ↩

                             ¦   VEHICLES      ¦                           ↩

                             +-----------------+                           ↩


                                                                           ↩

Enter database number (0 - 65535 except 255)
```

3 Enter the database ID (**DBID**) of the Tamino doctype for which a DDM is to be created.

Valid values are `0` - `65535`, except `255`. If you enter a `0` (zero), the database ID specified with the Natural profile parameter `UDB` (see the *Parameter Reference* documentation) of the NATPARM parameter file is used.

If the specified DBID identifies a Tamino database, the **Create XML DDM** window appears:

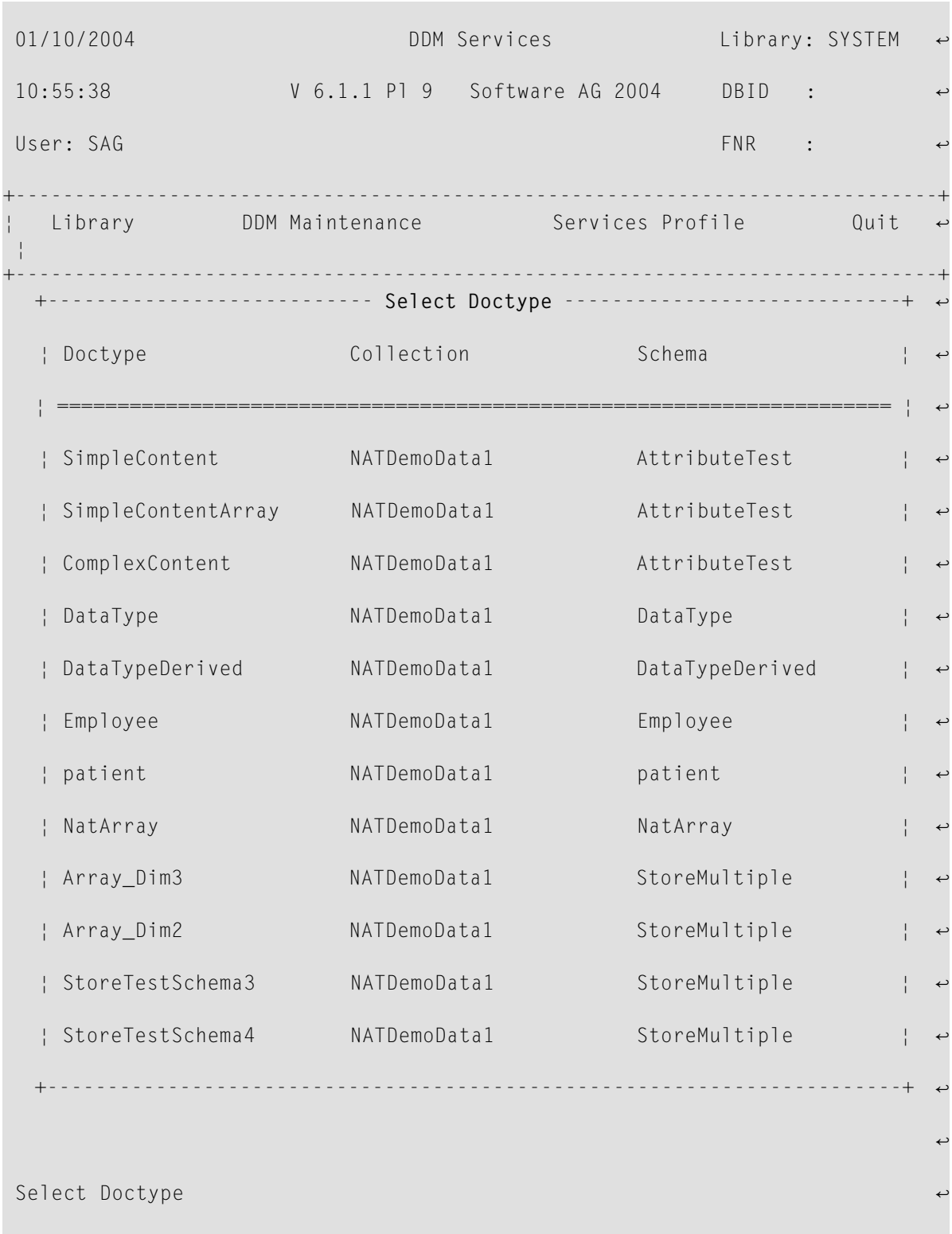
```
01/10/2004                      DDM Services             Library: SYSTEM   ↩
15:50:58              V 6.1.1 Pl 9   Software AG 2004     DBID   :          ↩
User: SAG                                                 FNR    :          ↩
+-------------------------------------------------------------------------+
¦   Library         DDM Maintenance           Services Profile         Quit   ↩
 ¦
+-------------------------------------------------------------------------+
                              +-----------------+                           ↩
                              ¦   <CREATE>      ¦                           ↩
                 +------------ Create XML DDM --------------+               ↩
                 ¦ FNR      1                                ¦               ↩
                 ¦ DDM Name                                  ¦               ↩
                 +-------------------------------------------+               ↩
                              ¦   PERSONNEL     ¦                           ↩
                              ¦   VEHICLES      ¦                           ↩
                              +-----------------+
```

4 Enter the name to be assigned to the DDM.

The file number (**FNR**) of the DDM is always `1`. The file number cannot be modified.

5 Press ENTER.

The **Select Doctype** window appears with a list of doctypes as shown in the example below:



6 From the list, select a doctype and press ENTER.

The DDM editor is invoked and the DDM generated from the selected doctype is read into the editing area.

If the specified database is not active or cannot be accessed or if the file does not exist, a corresponding error message is issued. Nevertheless, if you press ENTER, you can still open an empty DDM editor screen, enter new field attribute definitions and save the DDM source. However, in this case you cannot check any definitions against the database file description.

7 If required, edit the DDM source: see the section ***Using the DDM Editor***.

8 After editing, press ESC and, from the **MISC** menu choose **EXIT (with STOW)**.

The syntax of the DDM source is checked and the DDM is saved as a source and a cataloged object. See also ***Saving and Cataloging a DDM***.

23 Invoking and Terminating the DDM Editor

The DDM editor is used to edit the source of a DDM.

This section describes how to invoke and terminate the DDM editor for an existing DDM by using either the **DDM Maintenance** menu of DDM Services, or the Natural system command `EDIT`.

Invoking the Editor with DDM Maintenance

To invoke the DDM editor from DDM Maintenance

- In the DDM maintenance window, next to the DDM to be edited, enter the function code `E`.

  The source code of the specified DDM is read into the editing area of the DDM editor.

  If the Adabas or SQL database referenced by the specified DDM is available and a database file with the specified file number exists, a screen similar to the example below appears; if not, an empty screen is displayed instead.

```
17/09/2004                      DDM Services
12:44:43              V 6.1.1 Pl 9   Software AG 2004             Line: 1
DBID: 20    FNR: 14    DDM: EMPLOYEES                          DEF.SEQ.:
   C   T      L Name                              F  Length  S  D
              1 PERSONNEL-ID                      A     8       D
       *        CNNNNNNN
       G      1 FULL-NAME
              2 FIRST-NAME                        A    20    N
              2 MIDDLE-I                          A     1    N
              2 NAME                              A    20       D
              1 MIDDLE-NAME                       A    20    N
              1 MAR-STAT                          A     1    F
       *        M=MARRIED
              1 SEX                               A     1    F
              1 BIRTH                             N  06.0       D
              1 N@BIRTH                           I     2       D
       G      1 FULL-ADDRESS
       M      2 ADDRESS-LINE                      A    20    N
              2 CITY                              A    20    N  D
              2 ZIP                               A    10    N
              2 POST-CODE                         A    10    N
              2 COUNTRY                           A     3    N
F1  HELP  F2 CHOICE F3  STOW+EXIT      F10 STOW             F11 CHECK
F12 DB-SHORT-NAMES  F13 MODIFY HEADER  F14 SHOW EXT FIELD   F15
```

  If the Tamino database referenced by the specified DDM is available and a database file with the specified file number exists, a screen similar to the example below appears; if not, an empty screen is displayed instead:

```
01/10/2004                        DDM Services
11:05:45              V 6.1.1 Pl 9   Software AG 2004              Line: 1
DBID: 175   FNR: 1     DDM: DDM_TEST                           TYPE: XML
   C   T   L  Name                              F      Length  D
       G   1  EMPLOYEE
       G   2  GROUP$1
           3  PERSONNEL-ID                      A           8  D
       G   2  GROUP$2
       G   3  FULL-NAME
       G   4  GROUP$3
           5  FIRST-NAME                        A          20  D
           5  MIDDLE-NAME                       A          20  D
           5  MIDDLE-I                          A          20  D
           5  NAME                              A          20  D
           3  MAR-STAT                          A           1  D
           3  SEX                               A           1  D
           3  BIRTH                             A          10  D
       G   3  FULL-ADDRESS
       G   4  GROUP$4
           5  ADDRESS-LINE                      A          20
           5  CITY                              A          20  D
           5  ZIP                               A          20  D
F1  HELP  F2 CHOICE F3  STOW+EXIT      F10 STOW             F11 CHECK
F12 DOCTYPE INFO    F13 MODIFY HEADER  F14 SHOW EXT FIELD   F15
```

For information on the fields and commands provided in the DDM editor, see the section ***Using the DDM Editor***.

Invoking the Editor with EDIT

As an alternative to using DDM Services functions, you can invoke the DDM editor with the Natural system command `EDIT`.

To invoke the DDM editor with `EDIT`

- From the Natural main menu, choose **Direct** and, in the **Direct Command** window, enter the following:

  ```
  EDIT VIEW object-name
  ```

  where *object-name* denotes the name of the DDM to be edited.

  The DDM editor is invoked for the DDM specified and the DDM source is read into the editing area.

For information on all options available with `EDIT`, see the relevant section in the *System Commands* documentation.

Terminating the Editor

This section describes how to terminate an editor session and return to the DDM maintenance window.

To terminate the DDM editor

1 After editing, saving and cataloging the DDM source (see ***Using the DDM Editor*** and ***Saving and Cataloging a DDM***), on the DDM editor screen, press ESC.

The DDM editor menus **COMMANDS**, **MISC** and **QUIT** (see also the section *Commands*) appear at the top of the DDM editor screen as shown in the example below:

```
         COMMANDS                  MISC                        QUIT
14:22:37                V 6.1.1 Pl 7    Software AG 2004              Line: 1
DBID: 20    FNR: 14    DDM: EMPLOYEES                          DEF.SEQ.:
   C   T      L Name                                F  Length  S  D
              1 PERSONNEL-ID                        A     8       D
       *        CNNNNNNN
       G      1 FULL-NAME
              2 FIRST-NAME                          A    20    N
              2 MIDDLE-I                            A     1    N
              2 NAME                                A    20       D
              1 MIDDLE-NAME                         A    20    N
              1 MAR-STAT                            A     1    F
       *        M=MARRIED
              1 SEX                                 A     1    F
              1 BIRTH                               N  06.0       D
              1 N@BIRTH                             I     2       D
       G      1 FULL-ADDRESS
       M      2 ADDRESS-LINE                        A    20    N
              2 CITY                                A    20    N  D
              2 ZIP                                 A    10    N
              2 POST-CODE                           A    10    N
              2 COUNTRY                             A     3    N
F1  HELP  F2 CHOICE F3  STOW+EXIT      F10 STOW             F11 CHECK
Select a Command
```

2 Select the **QUIT** menu and choose **EXIT (with STOW)** or **QUIT (without STOW)** to leave the DDM source as described in ***QUIT Menu***.

The DDM maintenance window appears.

24 Using the DDM Editor

The DDM editor screen is organized in a table where the field definitions data is contained in rows and columns. All attributes that belong to a field defined for a DDM are contained in one row (that is, source-code line), separated by tabs.

This section describes the columns contained on the DDM Editor screen and the commands provided to create or modify a DDM field, navigate in the screen, or catalog a DDM source, for example.

DDM Header Information

This section describes the fields contained in the header at the top of the DDM editor screen and how to modify them. In addition, you can display read-only Tamino specific doctype information.

Explanation of DDM Header Fields

The fields contained in the DDM header are described in the following table. For the Tamino-specific doctype fields, see also *Introducing Tamino XML Schema Language* in the *Programming Guide*.

Header Field	Description
DBID	The database ID (DBID) as specified in the global configuration file. **DBID** contains the database file referenced by the DDM. Valid range: `0` to `65535` (except `255`) See also: *DBMS Assignment* and *Database Management* in the *Configuration Utility* documentation. If `0` (zero) is specified, the default DBID as specified with the `UDB` profile parameter in the NATPARM parameter file is used. To modify the field contents, see ***Modifying DDM Header Fields***.
FNR	The number of the file being referenced in the database The file number of a DDM from Tamino is always `1` and cannot be modified. Valid range: `1` to `5000` To modify the field contents, see ***Modifying DDM Header Fields***.
DDM	The name of the DDM currently contained in the work area of the DDM editor.
Line	The number of the source-code line where the cursor is currently positioned.

Header Field	Description
DEF. SEQ.	Not applicable to Tamino. The default sequence by which the file is read when it is accessed with a `READ LOGICAL` statement in a Natural program. See also the `READ` statement described in the *Statements* documentation. The default sequence is specified with the two-character field short name. The system validates the short name based on the selected file number. If the database is accessible, the short name is checked against the corresponding field in the database file. If such a field does not exist in the database, a selection list of valid short names is displayed. If the database cannot be accessed, no selection list is generated. To modify the field contents, see *Modifying DDM Header Fields*.
TYPE	Displays the type `XML` for a DDM created from a Tamino database.
Collection	Read-only Tamino-specific doctype information. The name of the collection which is used within the Tamino database. To view doctype information, see *Displaying Tamino Doctype Information*.
Schema	Read-only Tamino-specific doctype information. The name of the Tamino XML schema which is used within the Tamino database. To view doctype information, see *Displaying Tamino Doctype Information*.
Doctype	Read-only Tamino-specific doctype information. The name of the doctype within the collection. To view doctype information, see *Displaying Tamino Doctype Information*.
Namespace URI Prefix	Read-only Tamino-specific doctype information. The list of namespace URI/prefix pairs which corresponds to the doctype. To view doctype information, see *Displaying Tamino Doctype Information*.

Displaying Tamino Doctype Information

To switch Tamino-specific doctype information on or off

- Press F12 (toggle switch).

  Or:

  From the **MISC** menu, choose **SHOW DOCTYPE INFO ON/OFF**.

  If switched on (the default setting is off), the **Doctype Information** section is displayed in the bottom half of the DDM editor screen as shown in the example below:

```
01/10/2004                        DDM Services
11:05:45              V 6.1.1 Pl 9   Software AG 2004             Line: 1
DBID: 175   FNR: 1     DDM: DDM_TEST                           TYPE: XML
   C   T   L  Name                              F      Length  D
       G   1  EMPLOYEE
       G   2  GROUP$1
           3  PERSONNEL-ID                      A           8  D
       G   2  GROUP$2
       G   3  FULL-NAME
       G   4  GROUP$3
           5  FIRST-NAME                        A          20  D
           5  MIDDLE-NAME                       A          20  D
~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Doctype Information ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Collection: NATDemoData1
Schema    : Employee
Doctype   : Employee

Namespace URI                                                          Prefix
------------------------------------------------------------------------------
http://www.w3.org/2001/XMLSchema                                       xs


F1  HELP  F2 CHOICE F3  STOW+EXIT      F10 STOW             F11 CHECK
F12 DOCTYPE INFO    F13 MODIFY HEADER  F14 SHOW EXT FIELD   F15
```

Modifying DDM Header Fields

To modify the contents of DDM header fields (Tamino-specific information is read-only)

1 Press F13.

Or:

From the **MISC** menu, choose **MODIFY DDM-HEADER**.

The cursor is positioned in the first header field (**DBID**) that can be modified.

2 Press TAB to go to the next header field to be modified.

Columns of Field Attributes

This section describes the field attributes that can be defined in the rows and columns of the DDM editor screen.

Column Heading	Field Attribute	
T	The type of field:	
	blank	Elementary field. This type of field can hold data and does not contain any other fields. It can have only one value within a record.
	G	Group. A group is a number of fields defined under one common group name. This allows you to reference several fields collectively by using the group name instead of the names of all the individual fields. Such fields cannot hold any data, but are only containers for other fields. **Note:** Groups defined in a DDM need not necessarily be defined as groups in the Natural object(s) that reference this DDM.
	M	Not applicable to Tamino. Multiple-value field. This type of field can have more than one value within a record. See also *Multiple-Value Fields* in the *Programming Guide*.
	P	Not applicable to Tamino. Periodic group. A group of fields that can have more than one value within a record. See also *Periodic Groups* in the *Programming Guide*.
	*	Comment line.
L	The level number assigned to the field. Levels are used to indicate the structure and grouping of the field definitions. This is relevant with view definitions, redefinitions and field groups (see the relevant sections in the *Programming Guide*). Valid level numbers are 1 - 7. For Tamino: valid level numbers are 1 - 99. Level numbers must be specified in consecutive ascending order.	

Column Heading	Field Attribute
DB	Not applicable to Tamino. The display of the **DB** column is switched off by default. To switch the display on or off, press F12 (toggle switch). The **DB** column displays the two-character short name of the corresponding field in the database file (see also ***Example of a DB Column***). **Creating Fields:** If you create a new DDM field and the display of the **DB** column is switched off, the DDM editor assigns to the new field a short name that has not yet been used for another field. This means that for the new field there is no correlation between the database file and the DDM. To guarantee that the short name of a new field is checked against the database, create a field by using the **line command** I as described in the section *Commands for Editing and Function Execution*.
Name	The name of the field. It can be 3 - 32 characters long for Adabas fields and 1 - 32 characters for SQL columns and Tamino doctypes. The rules to create a name comply with the naming conventions for user-defined variables (see the *Using Natural* documentation), except that the first character of the name must always be a Latin capital letter (A - Z). In addition, the name must not start with L@ or N@. These prefixes identify **indicator fields** as explained in the following section. The field name is the name used in other Natural objects (for example, in a program) to reference the field. The field name is unique across the whole DDM. For Tamino, the field name is not necessarily the same name as **Tag Name** (see *Tamino-Specific Extended Field Attributes*).
F	The Natural data format of an elementary field, such as A (alphanumeric), P (packed numeric) or L (logical). For valid Natural data formats, refer to *Format and Length of User-Defined Variables* in the *Programming Guide*.
Length	The standard length of an elementary field. This length can be overridden by the user in a Natural program. For numeric fields (Natural data format N), the length is specified as *nn*.*m*, where *nn* is the number of digits before the decimal point an *m* is the number of digits after the decimal point. In the **Length** input field, you can specify either the field length as a numeric value or enter the keyword DYNAMIC to specify that the field length is variable.

<table>
<tr><th>Column Heading</th><th>Field Attribute</th></tr>
<tr><td></td><td>For further information, see DDM Generation and Editing for Varying Length Columns in the Programming Guide.</td></tr>
<tr><td>S</td><td>Not applicable to Tamino.

Null-value suppression option:
<table>
<tr><td></td><td></td></tr>
<tr><td>blank</td><td>Indicates that standard Adabas suppression is used; that is, trailing blanks in alphanumeric fields and leading zeros in numeric fields are suppressed.</td></tr>
<tr><td></td><td></td></tr>
<tr><td>F</td><td>Indicates that the field is defined with the Adabas fixed storage option; that is, no suppression is used and the field is stored without compression.</td></tr>
<tr><td></td><td></td></tr>
<tr><td>N</td><td>Indicates that the field is defined with the Adabas null-value suppression option. This means that null values for the field are not stored in the inverted list and are not returned when the field is used in the WITH clause of a FIND statement, or in a HISTOGRAM or READ LOGICAL statement.</td></tr>
<tr><td></td><td></td></tr>
<tr><td>M</td><td>Indicates that the field is defined with the SQL null-value option not null. The Remark field (see Specifying Extended Field Attributes) for this field contains NN NC (not null, not counted). Below this field, the corresponding null-indicator field is listed.</td></tr>
</table></td></tr>
<tr><td>D</td><td>The Adabas descriptor type of an elementary field that is not an array.

A descriptor can be used as the basis of a database search performed with the READ or the FIND statement. For example: a field from an Adabas database that has a D or an S in the D column can be used in the BY clause of the READ statement. Once a record has been read from the database using the READ statement, a DISPLAY statement can reference any field that has either a D or an S in this column.

For a Tamino XML schema, an element is marked as a descriptor in the DDM when it has an overall multiplicity of a maximum of 1, in other words, if the maxOccurs values of the element and all of its predecessors in the schema are never greater than 1.
Descriptors types are:
<table>
<tr><td></td><td></td></tr>
<tr><td>blank</td><td>No descriptor.
This field is not a descriptor.</td></tr>
<tr><td></td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td>D</td><td>Elementary descriptor.
Value lists are created and maintained for this field by Adabas, so that this field can be used as a search criterion in a FIND</td></tr>
</table></td></tr>
</table>

Column Heading	Field Attribute	
		statement, as a sort key in a FIND statement, or to control logical sequential reading in a READ statement.
	H	Not applicable to Tamino. Hyperdescriptor. A hyperdescriptor is a user exit in Adabas. For Natural, it provides the same functionality as a phonetic descriptor (see below).
	N	Not applicable to Tamino. Non-descriptor. A non-descriptor is not a descriptor, but can be used as a search field for a non-descriptor search.
	P	Not applicable to Tamino. Phonetic descriptor. A phonetic descriptor allows the user to perform a phonetic search on a field (for example, a person's name). A phonetic search results in the return of all values which sound similar to the search value.
	S	Not applicable to Tamino. Subdescriptor or superdescriptor. If a sub/superdescriptor contains a multiple-value field or a field from a periodic group (or part of such a field), the sub/superdescriptor is marked with an M or a P in the field type column; this enables Natural to create the correct search algorithms for this sub/superdescriptor.

Example of a DB Column

```
01/10/2004                        DDM Services
19:02:57                V 6.1.1 Pl 9   Software AG 2004              Line: 1
DBID: 20    FNR: 14    DDM: EMPLOYEES                            DEF.SEQ.:
   C   T  DB  L Name                                F  Length  S  D
          AA  1 PERSONNEL-ID                        A     8       D
       *        CNNNNNNN
       G  AB  1 FULL-NAME
          AC  2 FIRST-NAME                          A    20    N
          AD  2 MIDDLE-I                            A     1    N
          AE  2 NAME                                A    20       D
          AD  1 MIDDLE-NAME                         A    20    N
          AF  1 MAR-STAT                            A     1    F
       *        M=MARRIED
          AG  1 SEX                                 A     1    F
          AH  1 BIRTH                               N  06.0       D
          AH  1 N@BIRTH                             I     2       D
       G  A1  1 FULL-ADDRESS
       M  AI  2 ADDRESS-LINE                        A    20    N
```

Indicator Fields

An indicator field is used to retrieve the length of a variable length field or information about the data significance (`NULL` value indicator) of a database field. An indicator field does *not* provide the contents of a database field.

A database field name starting with `L@` or `N@` is interpreted as an indicator field, according to the indicator specified in the *NATCONV.INI* configuration file (see also `IDENTIFIER-VALIDATION` in *How to Use Different Character Sets* in the *Operations* documentation. Therefore, a database field name must not start with any of these character strings unless it represents an indicator field.

The following happens when a DDM is initially generated.

- An `L@`*xxxxx* field is automatically added for every variable length field, where *xxxxx* is the name of the related field.

  This applies to long alpha (LA) and large object (LB) fields in an Adabas file.

  If the length indicator relates to an LA, LB or LOB field, the Natural data format/length must be I4. For a VARCHAR field, the format/length must be I2.
- An `N@`*xxxxx* field is automatically added for a field that may contain a `NULL` value, where *xxxxx* is the name of the related field.

  This applies to Adabas fields defined with the SQL Null Value Option. The Natural data format/length of a `NULL` indicator field must be I2.

Help on Columns of Fields

The following section describes how to invoke the help function for the columns of fields provided on the DDM editor screen.

To display help information on field columns

- Position the cursor at a field and press F1 *once* for instructions on entering a valid input value in this field.

  Or:

  Position the cursor at a field and press F2 to select a valid input value for this field from a list.

Commands for Editing and Function Execution

This section provides information on the positioning commands, line commands, editor commands and Natural system commands provided with the DDM editor.

Positioning commands are used to navigate in the DDM editor screen and line commands manipulate one or more lines of DDM source code. Editor commands, for example, are used to change the display mode of the editor screen and system commands, for example, are used to save the source and the cataloged object of the DDM.

Positioning Commands

You can use the following keys to navigate in the DDM editor screen:

Key	Explanation
DOWN-ARROW	Scrolls down one line.
LEFT-ARROW	Moves left on the screen.
RIGHT-ARROW	Moves right on the screen.
UP-ARROW	Scrolls up one line.
TAB	Moves from one input field to the next.

Line Commands

The line commands available in the DDM editor are used to copy, delete, insert or move single or multiple DDM source-code lines. As an alternative to entering line commands in the DDM source as described below, you can use the equivalent function (F) keys described in *Function-Key Assignments*.

To execute a line command

- On the DDM editor screen, in the column **C**, position the cursor in the source-code line(s) to which the command applies, and enter any of the line commands listed below:

Line Command	Explanation
C	Copies a marked block of lines: Delimit the block of lines with an X and a Y and position the cursor in the line above which you want to copy the block of lines and enter a C.
D	Deletes the line in which the line command was entered or deletes a marked block of lines: Delimit the block of lines with an X and a Y and enter a D.
H	Removes the marks from a block of lines.

I	Inserts a blank line above the line in which the line command was entered.
	
	If the database is available, the **Select Database Field** window appears from which you can choose the field short name:
	
	`+------- Select Database Field -------+`
	`¦ AA AA-1                            ¦`
	`¦ AB AB-1                            ¦`
	`¦ AC AC-1                            ¦`
	`¦ AE AE-1                            ¦`
	`¦ AD AD-1                            ¦`
	`+--------------------------------------+`
	
	See also **DB** in the section *Columns of Field Attributes*.

M	Moves a marked block of lines: Delimit the block of lines with an X and a Y and position the cursor in the line above which you want to move the block of lines and enter an M.
X	Marks the first line of a block of lines to be copied, deleted or moved.
Y	Marks the last line of a block of lines to be copied, deleted or moved.

Editor and System Commands

The editor commands or Natural system commands available in the DDM editor are executed by choosing either a command from the DDM editor menu **COMMANDS**, **MISC** or **QUIT**, or a function (F) key.

To execute an editor or a system command from a menu

1 On the DDM editor screen, press ESC.

The DDM editor menus **COMMANDS**, **MISC** and **QUIT** appear at the top of the DDM editor screen.

(See also the **example screen** in *Terminating the DDM Editor*.)

2 Select a menu and press ENTER.

A list of valid commands appears. The commands and their equivalent F keys (if available) are explained later in this section.

3 Choose the command to be executed for the current DDM source and press ENTER.

The following section describes the editor or system commands provided with DDM editor menus and the standard F-key assignments.

- COMMANDS Menu
- MISC Menu
- QUIT Menu
- Function-Key Assignments

COMMANDS Menu

The commands available with the **COMMANDS** menu correspond to the Natural system commands with the same name. They are used to check or scan the DDM source currently contained in the editing area of the DDM editor, and save the source as a source and/or cataloged object in the current Natural library and/or system file.

The **COMMANDS** menu options and equivalent F keys (if available) are explained in the following table:

Menu Option	F Key	Explanation
CATALOG		Saves the DDM source as a cataloged object as described for the system command CATALOG in the *System Commands* documentation.
CHECK	F11	Checks the syntax of the DDM source as described for the system command CHECK in the *System Commands* documentation.
SAVE		Saves the DDM source as a source object as described for the system command SAVE in the *System Commands* documentation.

Menu Option	F Key	Explanation
SCAN		Searches for a string of characters within the DDM source, with the option to replace the string with another string as described for the system command `SCAN` in the *System Commands* documentation.
STOW	F10	Checks the syntax of the DDM source and saves both the source and the cataloged object as described for the system command `STOW` in the *System Commands* documentation.

MISC Menu

The **MISC** menu options and equivalent F keys (if available) are explained in the following table:

Menu Option	F Key	Function
DB-SHORT-NAMES (ON/OFF)	F12	Not applicable to Tamino. Switches the display of the attribute column **DB** on the DDM editor screen on or off. See also **DB** in the section *Columns of Field Attributes*.
SHOW DOCTYPE INFO (ON/OFF)	F12	Only applies to Tamino. Switches the display of the **Doctype Information** section on the DDM editor screen on or off: see ***Displaying Tamino Doctype Information***.
MODIFY DDM-HEADER	F13	Displays in edit mode the DDM editor header fields **DBID**, **FNR** and **DEF.SEQ.** See also the section ***DDM Header Information***.
SHOW EXTENDED FIELDS (ON/OFF)	F14	Switches the display of the **Extended Field Information** section on the DDM editor screen on or off: see ***Specifying Extended Field Attributes***.
EDIT EXTENDED FIELDS	F15	Displays in edit mode the fields contained in the **Extended Field Information** section on the DDM editor screen: see ***Specifying Extended Field Attributes***.
SHOW COUPLED FILES		Only applies to DDMs that refer to Adabas files. Specifies that a file is physically coupled to this DDM. Files are coupled by using Adabas descriptors. For further information on file coupling, refer to the *Adabas* documentation.

QUIT Menu

The **QUIT** menu options and equivalent F keys (if available) are explained in the following table:

Menu Option	F Key	Explanation
EXIT (with STOW)	F3	Executes the STOW command (see *COMMANDS Menu*) and leaves the DDM editor.
QUIT (without STOW)		Leaves the DDM editor without saving any modifications and without executing the STOW command (see *COMMANDS Menu*).

Function-Key Assignments

The commands available with F keys are used as an alternative to the commands provided with DDM editor menus or line commands. In addition, the F keys provide help information on the commands and fields available in the DDM editor.

To list all current F-key assignments

- On the DDM editor screen, press F1 *twice*.

The following table lists all standard F-key assignments and equivalent menu options or line commands:

Function Key	Explanation
F1	**HELP** If pressed *once*, displays help information for the field at which the cursor is positioned. If pressed *twice*, displays help information on the current F-key settings.
F2	**CHOICE** Displays a selection window (if relevant) for the field at which the cursor is positioned. From this window, you can choose a value. If no selection option is available, the help information window appears instead.
F3	**STOW + EXIT** Corresponds to **EXIT (with STOW)** in the **QUIT** menu.
F4 - F9	F4 - F9 are not displayed on the DDM editor screen because they can be reassigned to other keys. To display or change the current F-key settings, use the **Function Keys** option of the **Services Profile** menu described in the relevant section. F4 - F9 correspond to the following line commands:

F4	Corresponds to the line command D.
F5	Corresponds to the line command I.

Function Key	Explanation	
	F6	Corresponds to the line commands X and Y.
	F7	Corresponds to the line command H.
	F8	Corresponds to the line command C.
	F9	Corresponds to the line command M.
	See also *To copy or move a block of lines with F keys*.	
F10	**STOW** Corresponds to **STOW** in the **COMMANDS** menu.	
F11	**CHECK** Corresponds to **CHECK** in the **COMMANDS** menu.	
F12	**DB-SHORT-NAMES** or **DOCTYPE INFO** Toggle switch. Adabas: Corresponds to **DB-SHORT-NAMES (ON/OFF)** in the **MISC** menu. Tamino: Corresponds to **SHOW DOCTYPE INFO (ON/OFF)** in the **MISC** menu.	
F13	Corresponds to **MODIFY DDM-HEADER** in the **MISC** menu.	
F14	**SHOW EXT FIELD** or **HIDE EXT FIELD** Toggle switch. Corresponds to **SHOW EXTENDED FIELDS (ON/OFF)** in the **MISC** menu.	
F15	**EDIT EXT FIELD** Corresponds to **EDIT EXTENDED FIELDS** in the **MISC** menu.	

To copy or move a block of lines with F keys

1 Position the cursor in the first line of the block of lines to be copied, deleted or moved and press F6.

The line is marked.

2 Move down or up to the next or previous line by pressing DOWN-ARROW or UP-ARROW.

Each additional line is marked.

3 In the last line of the block of lines to be marked, press F6 to stop marking further lines.

4 Position the cursor in the line above which you want to copy or move the block of lines and press F8 or F9.

Specifying Extended Field Attributes

The extended field editing function provides the option to specify default field attributes for headers and edit masks as well as remarks to be applied when the field is used in another Natural object (for example, in a program).

The header attribute specifies the default column header to be displayed above the field when it is output, for example, with a `DISPLAY` statement. If no header is specified, the field name is used as column header.

The edit mask attribute specifies the default edit mask to be used when the field is output, for example, with a `DISPLAY` statement. The edit mask must conform with Natural syntax rules and be valid for the Natural data format and length of the field.

The remark attribute specifies a comment about the field.

For Tamino, the extended field editing function also provides additional Tamino-specific information.

Related Topics:

- `DISPLAY` and `INPUT` in the *Statements* documentation
- *EM - Edit Mask* in the *Parameter Reference* documentation

The section below covers the following topics:

Switching Extended Field Attributes On or Off

This section describes how to switch extended field attributes on or off.

To switch extended field attributes on or off

- Press F14.

  Or:

  From the **MISC** menu, choose **SHOW EXTENDED FIELDS (ON/OFF)**.

  If switched on (the default setting is off), the **Extended Field Information** section is displayed in the bottom half of the DDM editor screen as shown in the examples below.

Example of Extended Field Attributes from Adabas:

```
23/09/2004                      DDM Services                              ↩
15:16:36              V 6.1.1 Pl 9   Software AG 2004             Line: 8  ↩
DBID: 20    FNR: 14    DDM: EMPLOYEES                        DEF.SEQ.:      ↩
   C   T      L Name                              F  Length  S  D           ↩
             1 PERSONNEL-ID                       A     8       D           ↩
       *       CNNNNNNN                                                    ↩
       G     1 FULL-NAME                                                   ↩
             2 FIRST-NAME                         A    20    N            ↩
             2 MIDDLE-I                           A     1    N            ↩
             2 NAME                               A    20       D         ↩
             1 MIDDLE-NAME                        A    20    N            ↩
             1 MAR-STAT                           A     1    F            ↩
       *       M=MARRIED                                                   ↩
             1 SEX                                A     1    F            ↩
             1 BIRTH                              N  06.0       D         ↩
             1 N@BIRTH                            I     2       D         ↩
       G     1 FULL-ADDRESS                                                ↩
       M     2 ADDRESS-LINE                       A    20    N            ↩
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Extended Field Information ~~~~~~~~~~~~~~~~~~~~~~~~~~~
        Header   : MARITAL/STATUS                                          ↩
        Edit Mask:                                                         ↩
        Remark   : NC                                                      ↩
F1  HELP  F2 CHOICE F3  STOW+EXIT      F10 STOW            F11 CHECK       ↩
F12 DB-SHORT-NAMES  F13 MODIFY HEADER  F14 HIDE EXT FIELD   F15 EDIT EXT FIELD
```

Example of Extended Field Attributes from Tamino:

```
01/10/2004                      DDM Services
13:52:41              V 6.1.1 Pl 9   Software AG 2004             Line: 11
DBID: 102   FNR: 1     DDM: EMPLOYEES-TAMINO                 TYPE: XML
   C   T   L  Name                              F      Length  D
       G   4  GROUP$3
           5  FIRST-NAME                        A          20  D
           5  MIDDLE-NAME                       A          20  D
           5  MIDDLE-I                          A          20  D
           5  NAME                              A          20  D
           3  MAR-STAT                          A           1  D
           3  SEX                               A           1  D
           3  BIRTH                             A          10  D
~~~~~~~~~~~~~~~~~~~~~~~~~~~ Extended Field Information ~~~~~~~~~~~~~~~~~~~~~~~~~~~
Header       : Marital/Status
Edit Mask    :
Remark       : xs:string
Tag Name     : Mar-Stat
XPath        : /Employee/Mar-Stat
Occurrence   :
Flags        : MULT_OPTIONAL
Default Value:
Fixed Value  :
F1  HELP  F2 CHOICE F3  STOW+EXIT      F10 STOW             F11 CHECK
F12 DOCTYPE INFO    F13 MODIFY HEADER  F14 HIDE EXT FIELD   F15 EDIT EXT FIELD
```

The contents of the fields in the **Extended Field Information** section are triggered by the field at which the cursor is positioned. In the examples above, the cursor is positioned at `MAR-STAT`.

Editing Extended Field Attributes

The section below describes how to edit the field attributes contained in the **Extended Field Information** section of the DDM editor screen. Note that **Tamino-specific extended field attributes** (see the relevant section) cannot be edited.

To edit the fields in the Extended Field Information section

- Press F15.

  The cursor is positioned in the **Extended Field Information** section in the `Header` input field, which is now highlighted and can be modified.

  To move down to the next input field, press DOWN-ARROW or TAB.

  To move to up to the previous field, press UP-ARROW.

To terminate editing with or without field modification

- Press ENTER.

  The cursor is positioned outside the **Extended Field Information** section.

Tamino-Specific Extended Field Attributes

Tamino-specific extended field attributes are extracted from Tamino XML schema definitions.

In addition to the fields **Header**, **Edit Mask** and **Remark**, the following *read-only* Tamino-specific attributes are displayed in the **Extended Field Information** section:

Attribute	Function
Tag Name	The name of the field within a Tamino doctype. This name may be not unique within the whole XML document. Some group fields might not have a `Tag Name`.
XPath	The complete XPATH that references a field within a Tamino doctype. XPATH information is used during application runtime to uniquely identify a data element in a given XML document. Therefore, it is not possible to change the XPATH information. Some group fields might not have an XPATH.
Occurrence	The minimum and maximum numbers of occurrences. In Tamino, the multiplicity of the field as extracted from the Tamino XML schema. The multiplicity of a field is expressed with the `maxOccurs` facet in the Tamino XML schema.
Flags	The flags represent the hierarchical field structure within a Tamino group structure. They are used internally to help in correctly recognizing special group structures (that is, the attributes of an element tag) or multiple occurrences. Additionally, the user can identify DDM fields which are either mandatory or optional in XML documents. Combinations of the flags for one field are possible. The following flags can be displayed:

`ARRAY`	Field is an array; that is, `maxOccurs` is greater than 1.
`GROUP_ATTRIBUTES`	Field is a group that contains the attribute sub-fields of the predecessor field.
`GROUP_ALTERNATIVES`	Field is a group that represents the choice constructor; the choice elements are contained as sub-fields.

Attribute	Function	
	GROUP_SEQUENCE	Field is a group that represents the sequence constructor; the sequence elements are contained as sub-fields.
	GROUP_ALL	Field is a group that represents all constructors; all elements are contained as sub-fields.
	ATTR_REQUIRED	Field is an attribute marked as required.
	ATTR_OPTIONAL	Field is an attribute marked as optional.
	ATTR_PROHIBITED	Field is an attribute marked as prohibited.
	MULT_OPTIONAL	Field can occur in the XML document but does not need to.
	MULT_REQUIRED	Field must occur in the XML document.
	MULT_ONCE	Field must occur exactly once in the XML document.
	SIMPLE_CONTENT	Field was defined as `complexType` with `simpleContent`.
Default Value	The default value assigned to the field; this attribute is not yet used.	
Fixed Value	The fixed value assigned to the field; this attribute is not yet used.	

SQL-Specific Extended Field Attributes

In addition to the fields **Header**, **Edit Mask** and **Remark**, the following read-only SQL-specific attribute is displayed in the **Extended Field Information** section:

Attribute	Function
SQLTYPE	Information generated from the data types BLOB (Binary Large Object) or CLOB (Character Large Object) if contained in an Oracle database.

Setting Editor Preferences - Services Profile

The **Services Profile** menu of DDM Services is used to display or change the DDM editor profile settings for the function (F) keys F4 to F9, and the display mode of the **DB** column.

To invoke Services Profile

- On the **DDM Services** screen, choose **Services Profile**.

  The following menu options are displayed:

```
 22/09/2004                       DDM Services              Library: SYSTEM   ↩
 14:21:44                V 6.1.1 Pl 9   Software AG 2004     DBID   :          ↩
 User: SAG                                                  FNR    :          ↩
+-------------------------------------------------------------------------------+
¦   Library         DDM Maintenance           Services Profile         Quit   ↩
 ¦
+-----------------------------------------+-------------------+------------+
                                          ¦ Function Keys     ¦            ↩
                                          ¦ Other Definitions ¦            ↩
                                          +-------------------+            ↩
                                                                           ↩
                                                                           ↩
                                                                           ↩
                                                                           ↩
                                                                           ↩
 Show Other Definitions
```

This section contains information on the **Services Profile** menu options.

- Function Keys
- Other Definitions

Function Keys

With the **Function Keys** option, you can reassign the keys F4 to F9. These F keys are used to execute **line commands** (see the relevant section) in the DDM editor.

To reassign an F key

1 From the **Services Profile** menu, choose **Function Keys**.

The **Profile settings** window appears with the current F-key assignments:

```
+--- Profile settings ----+-
¦ Delete        F4        ¦
¦ Insert        F5        ¦
¦ Mark Block    F6        ¦
¦ Unmark Block  F7        ¦
¦ Copy          F8        ¦
¦ Move          F9        ¦
+-------------------------+
```

2 In the **Profile settings** window, position the cursor in the line of the function you want to re-assign and *press* the F key to which you want to assign this function. You can only assign an F key that is not already assigned to any other DDM Services function.

The new F key setting is displayed in the **Profile settings** window as shown in the example below:

```
+--- Profile settings ----+
¦ Delete        F21       ¦
¦ Insert        F5        ¦
¦ Mark Block    F6        ¦
¦ Unmark Block  F7        ¦
¦ Copy          F8        ¦
¦ Move          F9        ¦
+-------------------------+
```

Other Definitions

Not applicable to Tamino.

With the **Other Definitions** option, you can determine whether or not to display the **DB** column as described in *Columns of Field Attributes*.

To change the display mode of the DB column

1 From the **Services Profile** menu, choose **Other Definitions**.

The **Profile settings** window appears:

```
+------- Profile settings -------+
¦ Display Database short names N ¦
+---------------------------------+
```

2 Replace the current value by `Y` (Yes) to display the **DB** column permanently, and `N` (No) to hide the column permanently.

25 Saving and Cataloging a DDM

You can save a DDM as a source object and/or a cataloged object (generated program) in the current Natural library (if applicable) in the current Natural system file (see also *Storing DDMs - FDDM System File*).

For the naming conventions that apply to an object, refer to *Object Naming Conventions* in the *Using Natural* documentation.

To save and/or catalog a DDM

- On the DDM editor screen, from the **COMMANDS** menu, choose **SAVE**, **CATALOG** or **STOW** as described in *COMMANDS Menu*.

  Or:

  When leaving the DDM editor screen, from the **QUIT** menu, choose **SAVE**, **EXIT (with STOW)** as described in *QUIT Menu*.

26 Listing DDMs

This section provides instructions for displaying a list of DDMs by using either the **DDM Maintenance** menu of DDM Services or the Natural system command `LIST`.

Listing DDMs with DDM Maintenance

With the **DDM Maintenance** menu, you can list all DDMs available in the current Natural library and/or system file.

To list all DDMs

1 If you want to list all DDMs in the library where you are logged on, go to **Step 3**.

If you want to switch libraries, log on to the required library by performing the following:

On the **DDM Services** screen, choose the **Library** menu.

A selection window appears with a list of all libraries contained in the current Natural library and/or system file as shown in the example below:

```
2004-09-10                      DDM Services             Library: SYSTEM   ↩

17:05:48              V 6.1.1 Pl 9   Software AG 2004     DBID   :          ↩

User: SAG                                                 FNR    :          ↩

+-------------------------------------------------------------------------------+
¦   Library         DDM Maintenance           Services Profile         Quit   ↩
 ¦
+-------------------------------------------------------------------------------+
   +----------+                                                               ↩

   ¦ <LOGON>  ¦
   ¦ DEMO     ¦                                                               ↩

   ¦ DEMO-1   ¦                                                               ↩

   ¦ ORD-EXAM ¦                                                               ↩

   ¦ PRE      ¦                                                               ↩

   ¦ SYSEXPG  ¦                                                               ↩

   ¦ SYSEXSYN ¦
   ¦ SYSTEM   ¦                                                               ↩

   +----------+                                                               ↩


                                                                              ↩

 Logon to DDM library
```

2 From the window, select the required library or select **<LOGON>** and enter the name of a library in the **DDM Library** window.

3 On the **DDM Services** screen, choose **DDM Maintenance**.

A list of all DDMs available in the selected library is displayed in the DDM maintenance window as shown in the example below:

```
2004-09-15                      DDM Services              Library: SYSTEM   ↩
16:57:44               V 6.1.1 Pl 9   Software AG 2004     DBID   :          ↩
User: SAG                                                  FNR    :          ↩
+-------------------------------------------------------------------------------+
¦   Library         DDM Maintenance           Services Profile         Quit   ↩
 ¦
+-------------------------------------------------------------------------------+
                            +--------------------+                             ↩
                            ¦   <CREATE>         ¦                             ↩
                            ¦   EMPLOYEES        ¦                             ↩
                            ¦   PERSONNEL        ¦                             ↩
                            ¦   VEHICLES         ¦                             ↩
                            +--------------------+                             ↩
                                                                               ↩
Select DDM
```

From the DDM maintenance window, you can either select a DDM for further processing or you can add a new DDM by choosing the **<CREATE>** menu option. For details on the functions available in the DDM maintenance window, see the sections *Creating DDMs* and *Using DDM Maintenance Functions*.

Listing DDMs with LIST

The Natural system command `LIST` lists all DDMS available in the current Natural library, the steplibs, and the FNAT and FUSER system files.

To list all DDMs with `LIST`

- From the Natural main menu, choose **Direct** and, in the **Direct Command** window, enter the following:

  ```
  LIST VIEW *
  ```

  The **LIST V *** screen appears with a selection list of all DDMs available in the current Natural library, the steplibs and the FNAT and FUSER system files.

For information on all options available with `LIST`, see the relevant section in the *System Commands* documentation.

27 Maintaining DDMs in Different Environments

To transfer DDMs (for example, copy or move) between different libraries and system files, and to perform a DDM operation (for example, delete and find) in a different environment, you can use the Natural utility SYSMAIN (see the *Tools and Utilities* documentation).

To transfer DDMs between different hardware platforms (mainframes, Linux, and Windows), you can use the Object Handler (see the *Tools and Utilities* documentation).

28 Data Conversion for Adabas or RDBMS

The conversion tables shown in this section list the Natural data formats and their corresponding data types in an Adabas database or in a relational database management system (RDBMS).

The generation of DDMs from an RDBMS requires the conversion from RDBMS-specific data types to Natural data formats. For general information on data access and data conversion, see the list of related documentation in ***Related Topics***.

For information on using large and dynamic variables and/or fields, refer to the section *DDM Generation and Editing for Varying Length Columns* in the *Statements* documentation.

Adabas

Data Type	Adabas Data Format	Natural Data Format/Length
alphanumeric	A (*n*)	A*n*
binary	B (*n*)	B*n*
fixed	F (*n*) but: F8	I*n* I4
float	G (*n*)	F*n*
packed	P (*n*)	P (*2 * n - 1*)
unpacked	U (*n*)	N*n*
wide character (Unicode)	W (*n*)	U (*n/2 rounded down*)

Adabas D

RDBMS Data Type	Natural Data Format/Length
boolean	L
char (*n*)	A*n*
date	A10
fixed (*p*,*q*)	N*p*-*q*.*q*
float	F8
integer	I4
long	A (DYNAMIC)
long varchar	A (DYNAMIC)
smallint	I2
string	A*n*
time	A8

RDBMS Data Type	Natural Data Format/Length
timestamp	A26
varchar	A*n*

Db2

RDBMS Data Type	Natural Data Format/Length
date	A10
blob	B (DYNAMIC)
clob	A (DYNAMIC)
dbclob	U (DYNAMIC)
decimal(5)	N5
decimal(10,4)	N6.4
fixed character(5)	A5
float	F*n*
graphic(*n*)	U*n*
longvar	A (DYNAMIC)
longvarg	A (DYNAMIC)
large integer	I4
scientific notation	N10.6
small integer	I2
special data	A253
system date and time	A10
time	A8
timestamp	A26
varchar	A*n*
varg	2*A*n*
vargraphic(*n*)	U*n*

Notes:

1. Prerequisite for the use of the data types `blob`, `clob`, `dbclob`, `graphic` and `vargraphic` with Db2 is Entire Access Version 6.2.1 and above.
2. In Db2, the data types `graphic` and `vargraphic` are only available when the database has been generated with a statement like `CREATE DATABASE mydb USING CODESET UTF-8 TERRITORY US`. Refer to your local Db2 documentation for further information.

Oracle

RDBMS Data Type	Natural Data Format/Length
blob	B (DYNAMIC)
char (*n*)	A*n*
clob	A (DYNAMIC)
date	A10
decimal (*p*,*q*)	N*p*-*q*.*q*
double precision	F8
float	F4
integer	I4
long	A (DYNAMIC)
long raw	B (DYNAMIC)
nchar(*n*)	U*n*
nclob	U (DYNAMIC)
number	N*n*
nvarchar2(*n*)	U*n*
raw (*n*)	B*n*
real	F4
rowid	A*n*
smallint	I2
timestamp	A26
varchar	A*n*
varchar2 (*n*)	A*n*

Notes:

1. Do not confuse the data types `long` and `long raw`, and `clob` and `blob` in the same table.
2. Prerequisite for the use of data type `timestamp` is Entire Access Version 6.2.1 and above. The timestamp variants `timestamp with time zone` and `timestamp with local time zone` are not supported.
3. Prerequisite for the use of the data types `nchar`, `nvarchar2` and `nclob` with Oracle is Entire Access Version 6.2.1 and above.

Microsoft SQL Server

RDBMS Data Type	Natural Data Format/Length
binary (*n*)	B*n*
bit	N1
char (*n*)	A*n*
datetime	A26
float	F8
image	B (DYNAMIC)
int	I4
money	N15.4
nchar(2**n*)	U*n*
ntext	U (DYNAMIC)
nvarchar(2**n*)	U*n*
real	F4
smalldatetime	A26
smallint	I2
smallmoney	N6.4
text	A (DYNAMIC)
timestamp	B8
tinyint	I2
varbinary (*n*)	B*n*
varchar (*n*)	A*n*

Note: Prerequisite for the use of the data types `nchar`, `nvarchar` and `ntext` with Microsoft SQL Server is Entire Access Version 6.2.1 and above. Furthermore, these data types require the use of the MSSQLODBC driver of Entire Access Version 6.2.1 and above.

MySQL

RDBMS Data Type	Natural Data Format/Length
bigint	P19
binary (*n*)	B*n*
bit	N1
blob	B (DYNAMIC)
char (*n*)	A*n*
date	A10
datetime	A26
decimal	N10.0
decimal (*p*,*q*)	N*p*-*q*.*q*
double	F8
float	F4
int	I4
mediumblob	B (DYNAMIC)
mediumint	I4
mediumtext	A (DYNAMIC)
numeric	N10.0
numeric (*p*,*q*)	N*p*-*q*.*q*
longblob	B (DYNAMIC)
longtext	A (DYNAMIC)
text	A (DYNAMIC)
time	A26
timestamp	A26
tinyblob	B (DYNAMIC)
tinytext	A (DYNAMIC)
tinyint	I2
smallint	I2
varbinary (*n*)	B*n*
varchar (*n*)	A*n*
year	A4

PostgreSQL

RDBMS Data Type	Natural Data Format/Length
bigint	P19
bigserial	P19
boolean	L
bytea	B (DYNAMIC)
character (*n*)	A*n*
character varying (*n*)	A*n*
date	A10
decimal	N10.0
decimal (*p*,*q*)	N*p*-*q*.*q*
double precision	F8
integer	I4
interval	A20
numeric	N10.0
numeric (*p*,*q*)	N*p*-*q*.*q*
real	F4
serial	I4
smallint	I2
smallserial	I2
text	A (DYNAMIC)
time	A26
timestamp	A26

Related Topics

The following documentation sections relate to converting data from Adabas or RDBMSs:

- **Programming Guide:**
  Accessing Data in an Adabas Database

  Accessing Data in an SQL Database

29 Data Conversion for Tamino

The generation of Natural DDMs from Tamino is based on the Tamino XML schema language. The basic concepts of the Tamino XML schema language and how it interacts with Natural for Tamino is described in *Accessing Data in a Tamino Database* in the *Programming Guide*.

This section describes the mapping of Tamino data types to Natural data formats.

Built-In Tamino XML Schema Language Data Types

The Tamino XML schema language provides a large number of built-in data types that are mapped to the corresponding Natural data format, if possible. In some cases no adequate Natural data format is available. Then these Tamino data types are mapped to the most general Natural data format: `U (DYNAMIC)`. The data format `U (DYNAMIC)` can hold any Tamino XML schema built-in data type as the schema language is based on strings of unlimited length.

The following tables show the built-in Tamino primitive and derived data types supported by Natural for Tamino and the corresponding Natural data formats to which they are mapped.

Tamino Primitive Data Type

Tamino Primitive Data Type	Natural Data Format/Length
xs:string	U (DYNAMIC)
xs:boolean	L
xs:decimal	P22.7
xs:float	F4
xs:double	F8
xs:duration	U (DYNAMIC)
xs:dateTime	U (DYNAMIC)
xs:time	T
xs:date	D
xs:gYearMonth	U (DYNAMIC)
xs:gYear	U (DYNAMIC)
xs:gMonthDay	U (DYNAMIC)
xs:gDay	U (DYNAMIC)
xs:gMonth	U (DYNAMIC)
xs:hexBinary	U (DYNAMIC)
xs:base64Binary	U (DYNAMIC)

Tamino Primitive Data Type	Natural Data Format/Length
xs:anyURI	U (DYNAMIC)
xs:QName	U (DYNAMIC)
xs:NOTATION	U (DYNAMIC)

Tamino Derived Data Type

Tamino Derived Data Type	Natural Data Format/Length
xs:normalizedString	U (DYNAMIC)
xs:token	U (DYNAMIC)
xs:language	U (DYNAMIC)
xs:NMTOKEN	U (DYNAMIC)
xs:NMTOKENS	U (DYNAMIC)
xs:Name	U (DYNAMIC)
xs:NCName	U (DYNAMIC)
xs:ID	U (DYNAMIC)
xs:IDREF	U (DYNAMIC)
xs:IDREFS	U (DYNAMIC)
xs:ENTITY	U (DYNAMIC)
xs:ENTITIES	U (DYNAMIC)
xs:Integer	P29
xs:nonPositiveInteger	P29
xs:negativeInteger	P29
xs:long	P19
xs:int	I4
xs:short	I2
xs:byte	I2
xs:nonNegativeInteger	P29
xs:unsignedLong	P19
xs:unsignedShort	I2
xs:unsignedByte	I2
xs:unsignedInt	I4
xs:positiveInteger	P29

Tamino XML Schema Constructors

Tamino XML Schema constructors are used to define the structure of a document. Constructors can also be used to derive new data types from existing ones and to describe the nested structure of a document.

New Tamino XML Schema data types can be created by using a set of derivation methods from existing data types. If the derivation cannot be mapped to a Natural data format, Natural uses the most general data format `U (DYNAMIC)` instead.

The following table shows the Tamino XML schema constructors Natural supports and the attributes for each constructor. The Comment column describes the mapping, which is performed when a DDM is generated.

For restrictions concerning the use of XML schema constructors refer to the section *Accessing Data in a Tamino Database* in the *Programming Guide*.

Constructor	Attribute	Comment
xs:all	minOccurs maxOccurs	Is mapped to a Natural group structure.
xs:attribute	name ref type form use	Is mapped to a Natural group structure.
xs:choice	minOccurs maxOccurs	Is mapped to a Natural group structure.
xs:complexType	name mixed=false (only mixed=false supported)	Is a meta constructor and therefore does not result in a Natural data type immediately.
xs:element	name ref type form minOccurs maxOccurs	Mapped to a Natural data type or to a Natural group, depending on the `xs:element` sub-structures (simple or complex type definition).
xs:enumeration		Is mapped to type `U (DYNAMIC)`.
xs:extension	base	Is a meta constructor and therefore does not result in a Natural data type immediately.
xs:fractionDigits	value	Influences the precision of the Natural data type.

Constructor	Attribute	Comment
xs:length	value	Influences the length of a Natural data type; a length of `unbounded` is mapped to type `U (DYNAMIC)`.
xs:maxInclusive xs:maxExclusive xs:minInclusive xs:minExclusive xs:minLength xs:pattern	value	Does not influence the mapping (that is, the base type will not be restricted in any way).
xs:maxLength	value	Influences the length of a Natural data type; a length of `unbounded` is mapped to type `U (DYNAMIC)`.
xs:restriction	base	Is a meta constructor and therefore does not result in a Natural data type immediately.
xs:schema	attributeFormDefault elementFormDefault targetNamespace	Is a meta constructor and therefore does not result in a Natural data type immediately.
xs:sequence	minOccurs maxOccurs	Is mapped to a Natural group structure.
xs:simpleContent		Is a meta constructor and therefore does not result in a Natural data type immediately.
xs:simpleType	name	Is a meta constructor and therefore does not result in a Natural data type immediately.
xs:totalDigits	value	Influences the length of a numeric Natural data type.

Multiplicity in Tamino XML Schema Language

The multiplicity feature in the Tamino XML schema language is expressed by the attribute `maxOccurs` of the appropriate constructor. A `maxOccurs` value greater than `1` will result in an array definition in the Natural DDM from Tamino. Depending on the value of `maxOccurs`, a static array (if `maxOccurs` is set to a number) or an X-Array (if `maxOccurs` is set to `unbounded`) will be generated in the DDM. As usual, the array definition can be overwritten when defining a view from a DDM.