Data Area Editor

The 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.

The *Data Area Editor* documentation covers the following topics:

- Invoking the Data Area Editor
- Top Information Line
- Editor Command Line
- Bottom Information Line
- Using the Editing Area
- Columns in the Editing Area
- Extended Field Definition Editing
- Line Commands
- Editor Commands
- Editor Commands for Positioning
- Storing and Cataloging a Data Area
- User Exit for the Data Area Editor
- Exit Function

Related Topic:

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

Invoking the Data Area Editor

You invoke the data area editor with the 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 a local data area (indicated in the top left corner 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:

```
Local     LDA1     Library SAGTEST                            DBID  10 FNR  32
Command                                                                 > +
I T L  Name                            F Length     Miscellaneous
All -- --------------------------------- - ---------- ------------------------->
  *    LDA for new application
     1 INCOME                          A         20 (1:3,1:5) INIT ALL<'0'>
     1 PERSON
     2 SEX                             A          6
     2 AGE                             N          3
     1 NAME                            A         24
  R  1 NAME                                       /* REDEF. BEGIN : NAME
     2 FIRST-NAME                      A         10
     2 MIDDLE-INIT                     A          2
     2 LAST-NAME                       A         10
  C  1 DOLLAR                          A          5 CONST<'$US'>
  V  1 FINANCE-VIEW                                 FINANCE
     2 PERSONNEL-NUMBER                N        8.0
  P  2 MAJOR-CREDIT                                 (1:1) /* PERIODIC GROUP
     3 CREDIT-CARD                     A         18 (EM=XXX.XXX.XXX.XXX.XXX.XXX)
     3 CREDIT-LIMIT                    N        4.0
     3 CURRENT-BALANCE                 N        4.0
-- Current Source Size: 1969  Free: 78200 ----------------------- S 12   L 1
```

  The editor screen contains the following items (from top to bottom): the top information line, the editor command line, the editing area and the bottom information line. These items are explained in the following sections.

Top Information Line

The top information line of the editor screen can contain the following items (from left to right):

Data Area Type	Indicates the type of data area currently in the source work area: `Local`, `Global` or `Parameter`. The type can be changed by using the editor command `SET TYPE`.
Data Area Name	The name of the data area 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 with the `SAVE`, `CATALOG` or `STOW` command.
Modification Indicator: *	An asterisk (*) indicates whether the source code currently in the source work area contains unsaved modifications. The asterisk (*) also appears for new source code that has not yet been saved as a source object. The asterisk (*) is only visible if the editor profile option **Source Status Message** is set to `Y` (see *Editor Profile*). The asterisk (*) disappears when you execute a successful `SAVE` or `STOW` command on the source. See also *Exit Function*.
Lib	The library where you are currently logged on.
DBID	The database ID of the current system file.
FNR	The file number of the current system file.

Editor Command Line

The command line is indicated by the editor's **Command** prompt. 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 *Storing and Cataloging a Data Area*).

  For other system commands related to maintaining and using object sources, see *Editing and Storing Programming Objects* in the *System Commands* documentation.

- The name of a Natural program to be executed.

- One or more editor commands.

Note:
If you have changed a definition by typing in a modification or by using an editor command, a system command cannot be entered until you press ENTER.

Direction Indicator

The direction indicator entered next to the > (greater than) sign in the command line determines the operation direction of particular editor and line commands:

- +

  (plus sign)

  The command executes from the top line displayed on the screen (or from the line in which a line command is entered) towards the end of the source. This is the default setting.

- -

  (minus sign)

  The command executes from the top line displayed on the screen (or from the line in which a line command is entered) towards the beginning of the source.

More detailed information on the direction indicator can be found in the descriptions of the editor and line commands affected by the operation direction.

See also the editor profile option **Direction Indicator** described in *Editor Profile*.

Bottom Information Line

The bottom information line of the editor screen can contain the following:

- **Current Source Size**

  The size (number of characters) of the current source. This information is only displayed if the editor profile option **Source Size Information** is set to `Y` (see *Editor Profile*).

- **Free**

  The number of characters still available in the source work area. This information is only displayed if the editor profile option **Source Size Information** is set to `Y` (see *Editor Profile*).

- **S**

  The size (number of lines) of the source being edited.

- **L**

  The number of the source line currently displayed as the top line.

Using the 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.

In addition, if you use split-screen mode, the editing area displays fewer lines of source code. See also *Split-Screen Mode*.

To navigate in the editing area

- Use the editor commands described for the program editor in *Editor Commands for Positioning*.

  All positioning commands described for the program editor can be used with the data area editor as well.

To create or modify variables or fields

- Type in or modify all variable or field definitions in the columns of the relevant source line.

  You can specify whether the characters you type are automatically converted to upper case by using the editor profile options **Editing in Lower Case** and **Dynamic Conversion of Lower Case** (see *Editor Profile*).

  Or:
  Use one or more line commands as described in the relevant section.

  A line command, for example, is used to insert a line, copy variable or field definitions from another Natural object, or invoke the extended field definition editing function.

  Or:
  Use one or more editor commands as described in the relevant section.

  An editor command, for example, is used to delete a block of lines or specify prefixes for names.

Columns in the Editing Area

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

The editing area contains the following columns:

Column Heading	Explanation
I	The label indicator. An information field supplied by the editor. This column is not modifiable. Possible column entries are: `+` Indicates that more than one of the entries listed below exist for the variable or field. `E` Indicates that a definition error has been detected. `A` Indicates that array bounds have been defined by using the `.E` line command. `I` Indicates that an initial value has been defined by using the `.E` line command. `M` Indicates that an edit mask and/or a header has been defined by using the `.E` line command. `S` Indicates that both an initial value and an edit mask have been defined by using the `.E` line command. The following only applies to PDAs: *blank* Indicates the parameter specification call-by-reference (default). `V` Indicates the parameter specification call-by-value. `R` Indicates the parameter specification call-by-value-result. `O` Indicates an optional parameter that *can* be passed. For details, see function code **P** in *Extended Field Definition Editing*.

Column Heading	Explanation
T	The type of variable or field. Possible types are: `B` A data block within a GDA. `C` A user-defined constant (not applicable to PDAs) or a counter field (C* variable). A counter field is used to retrieve the number of occurrences of a multiple-value field or a period group in a view (DDM). See also *CONSTANT* in the *Statements* documentation and *Referencing the Internal Count for a Database Array (C* Notation)* in the *Programming Guide*. `G` A group within a view (DDM). `M` A multiple-value field within a view (DDM). `O` The handle of an object. `P` A periodic group within a view (DDM). `R` The redefinition of a variable or field. `U` The Globally Unique Identifier (GUID). `V` Not applicable to PDAs. A view definition created from a DDM. *blank* A user-defined variable or field, or a group structure (not within a view). `*` 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	The name of the variable or field, block or 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. Instead of specifying a variable name, the filler option *n*`X` can be used. With the filler option, *n* filler bytes can be denoted within the field or variable being redefined, where *n* can be up to 10 digits (smaller than 1 GB). The definition of trailing filler bytes is optional.
F	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).
Miscellaneous	This input field can be used to enter the definitions described in *Using the Miscellaneous Column*.

Using the Miscellaneous Column

The definitions that can be entered in the fields of the **Miscellaneous** column are described in this section.

As the **Miscellaneous** field may be too short to make all required specifications, the `.E` line command is provided for extended field definition editing.

A definition can be of up to 32 characters, whereby only 26 characters are displayed on the screen. You can scroll in the field by using the editor command `M +/-`. You can display all of the 32 characters or enter additional characters in an extra window, which opens when you enter a question mark (?) in the first position of the **Miscellaneous** field.

You can define the following:

Array

Enter the upper and lower bounds of an array. For detailed information on defining arrays, see *Arrays* in the *Programming Guide*.

Examples:

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

Initial Value

Not applicable to PDAs.

Enter an initial value according to the common Natural syntax definitions in a `DEFINE DATA` statement. For detailed information on defining initial values, see *Initial-Value Definition* and *Initial/Constant Values for an Array* in the *Statements* documentation.

Examples:

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

Edit Mask, Header and/or Print Mode

Edit masks and headers do not apply to PDAs.

Enter an edit mask or a header definition and/or the print mode according to the syntax rules that apply to the corresponding session parameter `EM`, `HD` or `PM` described in the *Parameter Reference* documentation.

Examples:

```
(EM=999.99)
(HD='TEXT' EM=XXX.XXX.XX PM=N)
```

Comment

A commentary text which must be preceded by a slash and an asterisk (/*).

Name of a DDM

For a view definition, you must enter the name of the DDM from which the view is derived.

You can modify the name of the DDM if all fields of the view are also contained in the DDM with the modified name.

Name of a Parent Block

For a block definition, you must enter the name of the corresponding parent block.

Extended Field Definition Editing

The extended field definition editing function can be used to define the following:

- Parameters and arrays within PDAs.
- Arrays, initial values, edit masks and headers within LDAs and GDAs. This is an alternative to using the **Miscellaneous** column.

To execute the extended field definition editing function

1. In the **T** column, next to the variable or field for which you want to define extended attributes, enter the following line command:

   ```
   .E
   ```

   An **Extended Field Definition Editing** menu similar to the example screen for a user-defined variable in an LDA is shown below:

   ```
   10:15:08                ******  EDIT FIELD  ******                    2008-09-22
                     - Extended Field Definition Editing -

   Local     LDA2*     Library SAGTEST                        DBID    10 FNR    32



            Code  Function                          Definition
           -----  ------------------------------  ----------
             S     Single Value Initialization    no
             F     Free Mode Initialization       no
             E     Edit Mask Definition           no
             A     Array Index Definition         no
             ?     Help
             .     Exit


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

      Code   ?  for Field: #USER-VARIABLE-1(A10)
   ```

   The functions provided on the **Extended Field Definition Editing** menu depend on the type of the data area, the type of variable and the contents of the **Miscellaneous** field. For example, if a variable has already been initialized in the **Miscellaneous** field, the functions **Single Value Initialization** and **Free Mode Initialization** are not available.

   Note:
   If `.E` is executed for a DDM field, the **Define Edit Mask / Header** screen (see the following step) is invoked immediately, because only edit masks and headers can be defined for DDM fields. It is not possible to define initial values for DDM fields.

2. Select the function required by entering the code that corresponds to the function required. For explanations of the functions available, see *Functions in the Extended Field Definition Editing Menu*.

   Depending on the function selected, either another menu or an extended field editing area similar to the example of a **Define Edit Mask / Header** screen below appears:

```
10:15:08                ******  EDIT FIELD  ******                    2008-09-22
                      - Define Edit Mask / Header -
Local     LDA2*     Library SAGTEST                        DBID    10 FNR    32
Command

#USER-VARIABLE-1(A10)
-------------------------------------------------------------------------
(EM=                                                                     )
-------------------------------------------------------------------------


#USER-VARIABLE-1(A10)
---------------------------------------------------------------------
(HD='                                                               ')
---------------------------------------------------------------------
```

3. Type a definition or enter a function code respectively.

   Note:
   A definition is *not* checked for syntax errors during editing. You can check a definition with the `CHECK` command after you terminated extended field definition editing.

4. When you are finished and return to the **Extended Field Definition Editing** menu, the **Definition** column reflects the changes as shown in the following example:

```
10:22:52                ******  EDIT FIELD  ******                    2008-09-22
                  - Extended Field Definition Editing -

Local     LDA2*     Library SAGTEST                        DBID    10 FNR    32



         Code  Function                       Definition
        -----  -----------------------------  ----------
          S     Single Value Initialization   no
          F     Free Mode Initialization      no
          E     Edit Mask Definition          yes
          A     Array Index Definition        no
          D     Delete all Definitions
          ?     Help
          .     Exit

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

   Code   ?  for Field: #USER-VARIABLE-1(A10)
```

   If any initial values, edit masks, headers or array index definitions have been defined, the corresponding status message in the **Definition** column changes from `no` to `yes`. If in a PDA any parameter type has been defined, an abbreviation of the parameter type (for example, `Val` for call-by-value) is displayed in the **Definition** column.

Any definitions made within the **Initial Values** and **Edit Mask / Header** subfunctions are immediately incorporated into the data area currently displayed in the data area editor but are not displayed in the **Miscellaneous** column of the editing area. A corresponding entry is only displayed in the **I** column (label indicator).

The functions available in the **Extended Field Definition Editing** menu and the commands available in an extended field editing area are described in the following section.

- Functions in the Extended Field Definition Editing Menu
- Commands in the Extended Field Editing Area

Functions in the Extended Field Definition Editing Menu

All functions that can be available in the **Extended Field Definition Editing** menu are described in the following table.

For an attribute control variable, only the functions codes **S**, **F**, **P**, **A** and **D** are allowed.

For a field that redefines another field, only the function codes **E**, **A** and **D** are allowed.

Function Code	Function
S	Defines an initial value for the specified variable or field in single-value mode. You only enter the required variable or field value; any further specifications necessary (including apostrophes for alphanumeric variables or fields, and value prefixes such as `H` for hexadecimal) are generated automatically. For example, from an initial value of `F1F2` for a binary variable (B2), the data editor will generate `INIT <H'F1F2'>`. If the variable or field is an array, an initial value can (but does not necessarily have to) be defined for each occurrence. With arrays, asterisk notation (*) can be entered in the command line to repeat the value in the last line of the previous page until the end of the current page. For attribute control variables, a screen is displayed where you can select attributes and colors as initial values. For details on attributes and colors, see the session parameters `AD` and `CD` in the *Parameter Reference* documentation. To define a constant value instead of an initial value, enter `Y` in the field **Define as CONSTANT (Y/N)**.
F	Defines an initial value for the specified field in free mode. A free-mode editor is provided where you can enter your initial value definitions according to the common Natural syntax definitions in a `DEFINE DATA` statement. For detailed information on defining initial values, see *Initial-Value Definition Initial/Constant Values for an Array* in the *Statements* documentation. See also *Examples* in *Initial Value*.

Function Code	Function
E	Defines an edit mask and/or header for the specified field according to the Natural rules for edit mask and header specifications. If both an edit mask and a header are specified, together they must not exceed 57 characters in length. However, if only an edit mask is specified, it can be up to 63 characters long; if only a header has been specified, it can be up to 58 characters long. If `.E` is entered for a DDM field, this function is invoked immediately, as only edit masks and headers can be defined for DDM fields. It is not possible to define initial values for DDM fields.
P	This function only applies to PDAs. It can be used to determine the way in which the value of a field specified as a parameter in a `CALLNAT` statement is passed from a program to an invoked object (for example, a subprogram). You can enter one of the following codes in the upper input field: `D` Call-by-reference (default). `V` Call-by-value. `R` Call-by-value-result. 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. Additionally, you can specify whether a parameter must be passed by entering one of the following values in the lower input field: `N` A parameter must be passed (default). `Y` An optional parameter that *can* be passed. 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.

Function Code	Function
A	Defines array bounds for the specified field. A free-mode editor is provided where you can enter your bound definitions in accordance with the common Natural syntax definitions. While you are editing, however, the specified values will not be checked (unless you enter the `CHECK` command).
D	Deletes all definitions made with the **S**, **F**, **E**, **P** and **A** function codes. An additional screen is provided, where you can specify the definitions to be deleted. By default, all definitions are marked with `Y`. If you do not want to delete a definition, remove the `Y` behind the definition or replace the `Y` by `N`.

Commands in the Extended Field Editing Area

The commands that can be entered in the command line of an extended field editing area are described in the following table:

Command	Function
`EDIT`	Returns to the editing area of the editor screen.
`.`	Returns to the previous screen to continue processing.
`--`	Returns to the beginning of the initial value specification(s).
`+`	Scrolls down one page. If the last page has been reached or if there is only one page available, returns to the editing area of the editor screen.
`*`	Copies the initial value of the last occurrence of the previous page to all empty fields of the current page. It is only available for arrays in single-value mode.

Line Commands

You enter a line command in the **T** column of a source line. You are recommended to enter a blank at the end of each line command. This prevents the editor from attempting to interpret any information existing on the line as part of the line command.

The default escape character which must precede each line command is a period (.). You can change the default character by using the editor profile option **Escape Character for Line Command** (see *Editor Profile*).

The line commands provided by the program editor are described in the following section. The notation `(`*n*`)`, `(`*nnn*`)` or `(`*nnnn*`)` indicates a repetition factor. The default repetition value is 1 (with the exception of the `.I` line command). For explanations of the syntax symbols used in this section, refer to *System Command Syntax* in the *System Commands* documentation.

Command	Function
`.C[(`*nnnn*`)]`	Copies the line in which the command was entered. See also *Notes for Line Commands*.

Command	Function
.CX[(*nnnn*)] or .CY[(*nnnn*)]	Copies the X-marked or the Y-marked line. See also the line commands .X and .Y and *Notes for Line Commands*.
.CX-Y[(*nnnn*)]	Copies the block of lines delimited by the X and Y markers. See also the line commands .X and .Y and *Notes for Line Commands*.
.D	Deletes one or more lines beginning with the line in which you enter the command towards the end of the source (regardless of any direction indicator setting). When entered for an individual field, only that field definition is deleted. When entered for a part of a hierarchical structure (view, group, redefinition), all subsequent definitions on subordinate levels are also deleted. For example, if you enter .D for a group defined at level 2, everything belonging to that group and with a level number greater than 2 is also deleted up to (but not including) the next level 2 definition. Comment lines (which usually are not assigned a level) are also considered to be at a subordinate level. To avoid the undesired deletion of a comment, assign an appropriate level to it.
.D(*nnnn*)	Deletes *nnnn* lines beginning with the line in which you enter the command towards the end of the source (regardless of any direction indicator setting). Unlike .D (see above), .D(*nnnn*) affects only the number of lines specified, regardless of any hierarchical structure.
.E	Invokes the **Extended Field Definition Editing** screen which is used to define array bounds, initial values, edit masks, headers and parameter attributes. For more information, see the section *Extended Field Definition Editing*.
.F(*file-name*)	This command includes a Predict file (applicable to the file types Conceptual, Standard, Sequential and Other).
.I[(*n*)]	This command adds *n* empty lines, where *n* can be in the range from 1 to 9. If *n* is not (or not correctly) specified, 10 lines (5 lines in split-screen mode) are added by default. Lines that are left blank are eliminated from the source, depending on the setting of the editor profile option **Empty Line Suppression** described in *Editor Profile*. **Note:** Only one .I can be performed at a time. See also *Notes for Line Commands*.

Command	Function
`.I(obj)`	Copies variable or parameter definitions from another Natural object of one of the following types: Data area Program Subprogram Subroutine Helproutine Map If the object specified as *obj* is not a data area, it must be available as a cataloged object. A window appears in the data area editor screen where you can select one of the following data definitions to be incorporated into your current data area: ● All local variables and parameters contained in the specified object (including those incorporated from LDAs and/or PDAs). ● All local variables contained in the specified object (including those incorporated from LDAs). ● Only those local variables defined within the specified object. ● All parameters contained in the specified object (including those incorporated from PDAs). ● Only those parameters defined within the specified object. If you incorporate variable definitions from objects without a `DEFINE DATA` definition (that is, from objects coded in reporting mode), variable redefinitions (see the `REDEFINE` statement in the *Statements* documentation) might be placed at the wrong position; that is, after the wrong variable. So, before compiling your new data area, check all variable definitions and redefinitions for correct positioning. If a variable redefinition results in more than one variable, each variable is incorporated as one individual redefinition by using filler bytes where appropriate. If the specified object has been cataloged with the Natural Optimizer Compiler, initial values and constants cannot be incorporated. If the object you want to insert has features the data area editor does not support, an appropriate message appears and the relevant line is marked as a comment line. See also *Notes for Line Commands*.

Command	Function
.I(*obj*,*ssss*,*nnnn*)	Includes a GDA, an LDA or a PDA. This feature is only supported for data areas which do not contain initial values or edit masks. The *ssss* entry can be used to indicate at which line the insertion is to begin. For example, when setting *ssss* to 20, the insertion begins with the 20th line of the data area. The *nnnn* entry can be used to indicate the number of lines to be inserted. If *ssss* and/or *nnnn* is specified for an object other than a data area (see the .I(*obj*) command), the specified value(s) are ignored. See also *Notes for Line Commands*.
.L	Undoes all modifications that have been made to the line since the last ENTER.
.MX or .MY	Moves the X-marked or the Y-marked line. See also the line commands .X and .Y and *Notes for Line Commands*.
.MX-Y	Moves the block of lines delimited by the X and Y markers. See also the line commands .X and .Y, and *Notes for Line Commands*.
.N	Marks (invisibly) a line to be positioned at the beginning of the source work area by the editor command POINT described in *Editor Commands for Positioning*. The mark is automatically deleted when an error with a line command or editor command occurs, or when the RESET command is executed.
.P	Positions the line marked with this command to the top of the screen.
.R	Redefines a variable or field as a single variable or a group of variables. With the filler option (*n*X), *n* filler bytes can be denoted within the variable or field being redefined. The definition of trailing filler bytes is optional. See also *Notes for Line Commands*.

Command	Function
.V[(*ddm-name*[,NOFL])]	Not applicable to PDAs. Defines a view from a DDM. Specify the DDM (*ddm-name*) from which you want to define a view. The fields of this DDM are then displayed in the editing area. Mark the fields to be incorporated into the view by entering any character in the **I** column next to the field(s) required. When you press ENTER, these fields are copied as a view definition into the current data area with the name of the view (default is the name of the DDM) assigned at level 1. In split-screen mode, the DDM currently in the split screen is displayed in the editing area when you enter .V without *ddm-name*. If .V(*ddm-name*) is specified within a view of the same name as specified for *ddm-name*, the selected fields are included in this view and no new view is defined. If NOFL is specified, the selected fields are included without format and length specification. When a periodic group or multiple-value field defined - in a DDM generated with Predict - as PC or MC respectively is included in a data area, a counter field (C* variable) for the group or field is automatically generated and placed before the group or field. The index for such a periodic group or multiple-value field is defined with the number of occurrences defined in Predict. If the number of occurrences has not been defined in Predict, the value 191 is used. If Predict is active, Predict redefinitions and comments are incorporated too. With VSAM views, the actual number of occurrences is always displayed. In addition, VSAM views contain information on subdescriptors and superdescriptors. For further information, see the *Natural for VSAM* documentation.
.X	Not applicable to periodic groups, multiple-value fields or view definitions. Marks a line with an X. See also *Notes for Line Commands*.
.Y	Not applicable to views, periodic groups or redefinitions. Marks a line with a Y. See also *Notes for Line Commands*.
.*	Generates a counter field (C* variable) for multiple-value fields or fields within a periodic group. See also *Notes for Line Commands*.

Command	Function
number [(*nnn*[,*m*])]	This command is available in split-screen mode and with a DDM in the split-screen area only. To obtain fields and groups from the split-screen area, the line number of the field or group from the split-screen area must be specified in the first column, without a period (.). Fields and groups from the split-screen area can be included as fields of a view (if *number* is entered inside a view) or as user-defined variables. If the selected field has the same name as the field for which the command was entered, it is substituted instead of inserted. Multiple lines can be obtained from the split screen by using the *nnn* notation where *nnn* is the number of lines to be included. The *m* notation can be used to specify a level number to be assigned to the field or group to be inserted. The level number in the data area can be modified. See also *Notes for Line Commands*.

Notes for Line Commands:

- The commands .I(*obj*), .R and .* are available in full-screen mode only, not in split-screen mode.
- If both the commands .X and .Y are applied to one line, it is treated as being marked with an X and with a Y; the line marker actually shown to reflect this status is a Z.
- If the direction indicator is set to + (plus sign), the copied, inserted or moved lines are placed *after* the line in which the corresponding command was entered; if the direction indicator is set to - (minus sign), the copied, inserted or moved lines are placed *before* the line in which the command was entered.

Editor Commands

The editor commands that can be entered in the command line of the data area editor are described in the following section. For explanations of the syntax symbols used in this section, refer to *System Command Syntax* in the *System Commands* documentation.

Command	Function
ADD[(*n*)]	Adds *n* blank lines. If the direction indicator is set to + (plus sign), the lines are added after the last line of the object being edited; if the direction indicator is set to - (minus sign), the lines are added before the first line of the object. The value for *n* can be in the range from 1 to 9. If *n* is not (or not correctly) specified, 9 lines (4 in split-screen mode) are added by default. With the next ENTER, lines that are still left blank are eliminated.
CANCEL or . (a period)	Leaves the editor. Any modifications made since the last time the SAVE command was entered are *not* saved.
CATALOG [*object-name*]	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. See also *Storing and Cataloging a Data Area*.
CHANGE ['*scan-value*'*replace-value*']	Scans the data area for a character string (*scan-value*) and replaces each such *scan-value* found with the character string entered as *replace-value*. Any special character which is not valid within a Natural variable name can be used as the delimiter character. Each line in which a character string is replaced is marked with an R to the left of the line. For information on how the scan operation is performed, see the SCAN command.

Command	Function
CHECK	Executes the system command CHECK which checks the syntax of the current data area definition. If an error is found, the erroneous line is marked with an E and an appropriate error message appears in the message line. If no errors are found, a message appears indicating successful completion of the check. CHECK also orders the entries in the **Miscellaneous** column in the following sequence: Array definition Initial value Edit mask, header and/or print mode Name of a DDM or parent block Comment
CLEAR	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.
DX or DY	Deletes the X-marked or the Y-marked line. See also the line commands .X and .Y.
DX-Y	Deletes the block of lines delimited by the X and Y markers. See also the line commands .X and .Y.
EX or EY	Deletes lines from the top of the editing area to, but not including, the X-marked line; or from the line following the Y-marked line to the bottom of the editing area. See also the line commands .X and .Y.
EX-Y	Deletes all lines in the source work area excluding the block delimited by X and Y. See also the line commands .X and .Y.
EXIT	Leaves the editor. Any modifications to the source are saved depending on the setting of the editor profile described in *Exit Function*.

Command	Function
GENERATE [*object-name*]	Generates a Natural object of the type copycode from the data area definition currently in the source work area. The program editor opens with the generated copycode source in the editing area including a DEFINE DATA LOCAL and corresponding END-DEFINE statement. If an *object-name* is specified, the generated copycode is saved under this name in the current Natural library in the current system file.
M +|-	Scrolls the **Miscellaneous** column. + Scrolls to the right. - Scrolls to the left.
PROFILE [*name*]	Invokes the **Editor Profile** screen where you can view or change your current editor profile settings. For details, see the section *Editor Profile*.
READ *object-name*	Executes the system command READ which reads an existing data area definition into the source work area. For all syntax rules that apply to the command, see READ in the *System Commands* documentation.
RESET	Deletes the current X and Y line markers and any marker previously set with the line command .N. See also the line commands .X and .Y.
SAVE [*object-name*]	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. See also *Storing and Cataloging a Data Area*.

Command	Function
SCAN *scan-value*	Scans the data area for a character string (*scan-value*) in the **Name** (default) and/or the **Miscellaneous** column of the editor screen, depending on whether the SET SCAN command was executed earlier. Each line in which the *scan-value* is found is marked with an S to the left of the line. The first line which contains the *scan-value* is positioned to the top line or the bottom line, depending on the current setting of the direction indicator. **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. If the direction indicator is set to + (plus sign), the scan is performed from the first line shown on the screen to the last line of the source work area. If the direction indicator is set to - (minus sign), the scan is performed from the last line shown on the screen to the first line of the source work area.
SCAN =[+|-]	Scans for the next occurrence of the *scan-value* specified with the SCAN command. The direction for a given scan command can be explicitly specified by entering SCAN =+ or SCAN =-. The setting of the direction indicator is then ignored. **Note:** The equal sign (=) used with the SCAN command is the default input assign character. If another character has been specified as input assign character (see session parameter IA described in the *Parameter Reference* documentation), that other character must be used instead.

Command	Function
`SET ABS [ON|OFF]`	Determines whether the `SCAN` command operates in absolute or non-absolute mode. `ON` The `SCAN` command operates in absolute mode, which means that the value to be scanned does not have to be delimited by blanks or special characters. `OFF` The `SCAN` command operates in non-absolute mode, which means that the value to be scanned must be delimited by blanks or special characters. The default is `OFF`. The `SET ABS` command corresponds to the editor profile option **Absolute Mode for SCAN/CHANGE** described in *Editor Profile*.
`SET PREFIX` *`prefix`*`|OFF`	Specifies a prefix for variable or field names. This prefix is then automatically placed before the value entered in the **Name** column for each line that is entered or modified, unless the name already begins with this prefix. If the concatenated variable or field is longer than 32 bytes, an appropriate message appears and the value in the **Name** column can be shortened. If this is not done, the prefix will not be inserted.

`SET SAVEFORMAT V31|V41`
 	
or
 	
`SET SF V31|V41`
	Specifies the default source format of data areas.
	
	If set to `V31`, data area sources are stored in compatible Natural Version 3.1 format by default.
	
	If is set to `V41`, data area sources are stored in extended source format.
	
	See also *Source Format for Data Area Storage*.

Command	Function
SET SCAN COMMENT|NAME|ALL	Determines the column(s) in which the SCAN command searches for a *scan-value*: COMMENT — The **Miscellaneous** column is scanned. NAME — The **Name** column is scanned. ALL — The **Name** and **Miscellaneous** columns are scanned. The default is NAME.
SET SIZE ON|OFF	If SET SIZE is set to ON, the size of the data area is displayed in the bottom information line of the editor screen. The SET SIZE command corresponds to the editor profile option **Source Size Information** described in *Editor Profile*.
SET STAY ON|OFF	If STAY is set to ON, the current screen will stay when ENTER is pressed. Forward and backward positioning can be done by positioning commands only. If STAY is set to OFF, pressing ENTER positions to the next screen if no changes were applied to the current screen. The SET STAY command corresponds to the editor profile option **Stay on Current Screen** described in *Editor Profile*.
SET TYPE G|L|A	Changes the type of the current data area: G — Global data area L — Local data area A — Parameter data area

Command	Function
SPLIT *parameter*	Splits the editor screen and displays the source of another Natural object in one half of the screen as described in *Split-Screen Mode*. *parameter* represents a parameter that must be specified with the command as described in *Split-Screen Commands*.
STOW [*object-name*]	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 definition or if you want to copy the current data area. Otherwise, an appropriate message appears. See also *Storing and Cataloging a Data Area*.

Editor Commands for Positioning

The editor commands that can be used for navigating through the current data area are described in the following section. You enter an editor command in the command line of the data area editor.

Command	Function
ENTER or +P or +	Positions forwards one page.
-P or -	Positions backwards one page.
+H	Positions forwards half a page.
-H	Positions backwards half a page.
T or --	Positions to top of source.
B or ++	Positions to bottom of source.
+*nnnn*	Positions forwards *nnnn* lines (maximum 4 digits).
-*nnnn*	Positions backwards *nnnn* lines (maximum 4 digits).
X	Positions to the line marked with an X.
Y	Positions to the line marked with a Y.
POINT	Positions to the line in which the line command .N was entered. See also the line command .P.

Storing and Cataloging a Data Area

Before a data area can be used in a Natural program (or another object), it must be saved and cataloged as a source object and/or a cataloged object that is stored in a Natural library in the current system file.

To save and/or catalog the current data area

- Use the system command SAVE, CATALOG or STOW as described in *Saving and Cataloging Objects* in the *Using Natural* documentation.

Note:
When you leave the data area editor with the `EXIT` editor command, the current source code is saved automatically if the appropriate editor profile option is set accordingly as described in *Exit Function*.

To keep a copy of the current source

- Use the editor options **Source Save into** and **Auto Save Numbers** as described in *Editor Profile*.

  A copy of the source edited last with any of the Natural editors is then automatically saved as a source object in the current Natural environment.

Source Format for Data Area Storage

The data area editor uses an internal source format to store the sources of data areas in the FUSER system file. New features and definitions that are available from Natural Version 4.1 onwards require that the data area source is stored in the FUSER system file using an extended source format.

Data areas that are stored using the extended source format cannot be used or edited with Natural Version 3.1 where a different source format was used. The data area editor of Natural Version 4.1 and above supports the Natural Version 3.1 format *and* the extended source format. The editor can read both formats and converts the Natural Version 3.1 format to the extended source format. As long as no Natural Version 4.1 (and above) features or definitions are used, data areas are stored in the Natural Version 3.1 format by default. This format guarantees compatibility between data area sources stored in a Natural Version 3.1 and a Natural Version 4.1 (and above) environment.

The source format to be used as a default for storing data areas can be specified with the user exit routine GDA-EX01 (see *User Exit for the Data Area Editor*) or, during an editor session, with the following editor command: `SET SAVEFORMAT V31` or `SET SAVEFORMAT V41`.

User Exit for the Data Area Editor

The data area editor provides a user exit routine for specifying default settings. The source of the user exit routine is provided in the library SYSEXT and named GDA-ES01. To activate this exit, `CATALOG` or `STOW` the source object as GDA-EX01 and copy GDA-EX01 to the library SYSLIB. For a detailed description, see the source object of GDA-ES01 in the library SYSEXT.

Exit Function

The effect of the `EXIT` editor command depends on the setting of the editor profile option **Prompt Window for Exit Function**:

- If set to `N`, the `EXIT` command leaves the editor and saves all modifications made to the current source; no prompt window is displayed.

- If set to `Y`, the `EXIT` command invokes the **EXIT Function** window whenever you execute the command on a source that contains unsaved modifications (see also Modification Indicator). If no modifications were made to the source, the window does *not* appear and the editor closes without saving the source.

The **EXIT Function** window provides the following options:

Option	Explanation
Save and Exit	Leaves the editor and saves 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.