Natural

Editors

Version 9.2.1

October 2022

ADABAS & NATURAL

This document applies to Natural Version 9.2.1 and all subsequent releases.

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

Copyright © 1992-2022 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA, Inc. and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at http://softwareag.com/licenses.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at http://softwareag.com/licenses/ and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third-Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software AG Products / Copyright and Trademark Notices of Software AG Products". These documents are part of the product documentation, located at http://softwareag.com/licenses and/or in the root installation directory of the licensed product(s).

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with Software AG.

Document ID: NATWIN-NNATEDITORS-921-20221212

Table of Contents

Preface xi
1 About this Documentation 1
Document Conventions 2
Online Information and Support 2
Data Protection 3
I NaturalONE as the Default Development Environment 5
2 NaturalONE as the Default Development Environment 7
II Program Editor 9
3 Invoking the Program Editor 11
4 Creating and Modifying Source Code 15
Entering and Inserting Text 16
Selecting Text 17
Copying, Cutting and Pasting Text 18
Dragging and Dropping Text 19
Undoing or Redoing Modifications 20
Deleting Text 21
Adding and Removing Comments 21
Renumbering Lines 23
Translating to Upper or Lower Case 23
Using Automatic Line Breaks 24
Converting to and from Hexadecimal Format 26
Recording, Replaying and Saving Keystrokes 28
Protecting Source-Code Lines 31
5 Finding Text 35
6 Replacing Text 39
7 Setting Bookmarks 43
8 Going to a Line Number 45
9 Toggling Breakpoints 47
10 Importing Data Fields 49
11 Editing or Listing Referenced Objects 51
12 Setting Display Modes 53
Formatting and Coloring Source Code 54
Showing and Hiding Source Code 55
Splitting the Editor Window 57
13 Saving Source Code 59
14 Using Context-Sensitive Help 61
III Data Area Editor 63
15 Data Area Editor 65
Invoking the Data Area Editor 66
Rows and Columns in the Editor Window 68
Editing Data Areas 73
Rearranging Columns 98
Showing or Hiding Fields 101

Navigating between Field Levels 103
Saving and Cataloging Data Areas 103
Generating Copycode from Data Areas 104
IV Map Editor 105
16 Map Editor 107
Inserting Map Fields and Menus 108
Character Encoding of Literal Strings 110
Modifying Map Contents 110
Defining Fields 116
Defining Field Attributes 131
Defining Arrays 132
Modifying Field Colors and Representation 134
Using Field Rules 135
Defining Data Areas for Maps 140
Testing Maps 142
Previewing Maps 142
Reversing Maps 143
Modifying the Map Profile 144
Saving and Cataloging Maps 148
V DDM Editor 149
17 Principles of Operation 151
Storing DDMs - FDDM System File 152
Restrictions of Use 153
18 Creating DDMs 155
Copying DDMs 156
Creating DDMs from Adabas 156
Creating DDMs from SQL 158
Creating Multiple DDMs from SQL 161
Creating DDMs from Tamino 162
Creating DDMs for VSAM 164
19 Invoking the DDM Editor 165
20 Using the DDM Editor 167
DDM Header Information 168
Using File Coupling 171
Columns of Field Attributes 172
Selecting Fields or Field Attributes 178
Inserting and Modifying Fields 180
Copying, Cutting and Pasting Fields 182
Finding and Replacing Field Names 183
Deleting Fields 186
Rearranging Columns 186
Showing or Hiding Fields 190
Specifying Extended Field Attributes 192
Displaying Descriptor Information 198
21 Saving and Cataloging a DDM 201

Additional Options for VSAM Files ... 202
22 Listing DDMs ... 205
Listing DDMs in a Workspace ... 206
Listing DDMs with LIST ... 206
23 Maintaining DDMs in Different Environments ... 209
24 Data Conversion for Adabas or RDBMS ... 211
Adabas ... 212
Adabas D ... 212
Db2 ... 213
Oracle ... 214
Microsoft SQL Server ... 215
MySQL ... 216
PostgreSQL ... 217
Related Topics ... 217
25 Data Conversion for Tamino ... 219
Built-In Tamino XML Schema Language Data Types ... 220
Tamino XML Schema Constructors ... 222
Multiplicity in Tamino XML Schema Language ... 223
VI Dialog Editor ... 225
26 General Information ... 227
27 Dialog Editor Window ... 229
Changing the Initial Position of the Dialog ... 230
Changing the Initial Size of the Dialog ... 231
Selecting/Deselecting Dialog Elements ... 231
Aborting Mouse Operations ... 232
Creation Mode in Map Editor and Dialog Editor ... 232
Changing the Position of a Dialog Element ... 232
Changing the Size of a Dialog Element ... 233
Moving the Pointer ... 233
Simulating the Mouse with the Spacebar ... 234
Scrolling in a Dialog ... 234
Using the Clipboard ... 235
28 Editing Dialogs ... 237
Editing a Dialog's Source Code ... 238
Editing a Dialog's Attributes ... 239
Editing a Dialog's Event Handlers ... 239
Defining a Dialog's Menu Bar ... 240
Defining a Dialog's Toolbar ... 240
Creating and Maintaining Timers for a Dialog ... 241
Creating and Maintaining Signals for a Dialog ... 241
Creating and Maintaining Context Menus ... 242
Creating and Maintaining Wallpapers ... 243
Adding a Comment Section to a Dialog ... 243
Defining a Parameter or Local Data Area for a Dialog ... 244
Selecting a Global Data Area for a Dialog ... 244

Defining an Inline Subroutine for a Dialog 245
Defining the Control Sequence in a Dialog 245
29 Dialog Wizard 247
Dialog Types (Standards) 249
How to create a template 252
30 Creating Dialog Elements 253
31 Importing Data Fields 255
32 Editing Dialog Elements 257
Cutting a Dialog Element 258
Copying a Dialog Element 258
Pasting a Dialog Element from the Clipboard 259
Deleting a Dialog Element 259
Selecting all Dialog Elements with the same Parent in a Dialog 260
Editing a Dialog Element's Attributes 260
Editing a Dialog Element's Event Handlers 261
Unifying the Size of Several Dialog Elements 261
Aligning the Position of Several Dialog Elements 262
Unifying the Spacing Between Several Dialog Elements 262
Stretching a Dialog Element 262
33 Attributes Windows for Dialogs and Dialog Elements 265
34 ActiveX Control Attributes Window 267
Entries 268
35 ActiveX Control Property Pages 271
36 Bitmap Control Attributes Window 273
Entries 274
37 Canvas Control Attributes Window 277
Entries 278
38 Control Box Control Attributes Window 281
Entries 282
39 Date/Time Picker Control Attributes Window 285
Entries 286
40 Dialog Attributes Window 289
Entries 290
41 Dialog Bar Control Attributes Window 295
Entries 296
42 Dialog Context Menus Window 299
Entries 300
43 Dialog Image Lists Window 301
Entries 302
44 Edit Area Control Attributes Window 305
Entries 306
45 Group Frame Control Attributes Window 309
Entries 310
46 Image List Base Images Subwindow 313
Entries 314

47 Image List Overlay Images Subwindow 317
Entries 318
48 Input Field Control Attributes Window 321
Entries 322
49 List Box Control Attributes Window 325
Entries 326
50 List View Control Attributes Subwindow 329
Entries 330
51 List View Control Attributes Window 333
Entries 334
52 List View Items Subwindow 337
Entries 338
53 Menu Editor Window 341
Entries 342
54 OLE Container Control Attributes Window 345
Entries 346
55 Progress Bar Control Attributes Window 349
Entries 350
56 Push Button Control Attributes Window 353
Entries 354
57 Radio Button Control Attributes Window 357
Entries 358
58 Scrollbar Control Attributes Window 361
Entries 362
59 Selecting an OLE Server or Document 365
Differences Between OLE Server, New OLE Object and Existing OLE Object 366
OLE Server 366
60 Selection Box Control Attributes Window 371
Entries 372
61 Signal Attributes Window 375
Entries 376
62 Slider Control Attributes Window 379
Entries 380
63 Spin Control Attributes Window 383
Entries 384
64 Status Bar Control Attributes Window 387
Entries 388
65 Status Bar Control Attributes Subwindow 391
Entries 392
66 Table Attributes Window 395
Entries 396
67 Table Attributes Subwindow 399
Entries 400
68 Tab Control Attributes Window 403

Entries 404
69 Tab Control Attributes Subwindow 407
Entries 408
70 Text Constant Control Attributes Window 411
Entries 412
71 Timer Attributes Window 415
Entries 416
72 Toggle Button Control Attributes Window 417
Entries 418
73 Toolbar Attributes Window 421
Entries 422
74 Tool Bar Control Attributes Window 425
Entries 426
75 Tool Bar Control Attributes Subwindow 429
Entries 430
76 Tree View Control Attributes Window 433
Entries 434
77 Tree View Control Attributes Subwindow 437
Entries 438
78 Wallpaper Attributes Window 441
Entries 442
79 Dialog Boxes 445
80 Array 447
Purpose 448
Entries 448
81 Data Area - Local, Parameter 449
Purpose 450
82 Data Area - Global 451
Purpose 452
83 Dialog Compile Error 453
84 Events 455
Purpose 456
Entries 456
85 Import Data Field 459
Purpose 460
Entries 460
86 Font 461
Purpose 462
87 Source 463
Purpose 464
88 Subroutines 465
Purpose 466
Entries 466
89 Enhanced Source Code Format 467
Syntax Conventions 468

How Natural Dialogs Work 468
Syntax 469
VII Class Builder 483
90 Class Builder 485
What is the Class Builder? 486
Class Builder Interface 488
Class Builder Nodes 499
Node Properties 509
Adding Class Components 517
Renaming Class Components 522
Removing Class Components 522
Editing Class Components 523
Using Interfaces from Several Classes 524
Locking Concept 526
Tutorial 527
Glossary 530
VIII Editor Features With SPoD 533
91 Editor Features With SPoD 535
Map Editor: GUI Controls and Field-Sensitive Maps 536
Data Area Editor: Source Format 536
DDM Editor: Database Features 536

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/or modify Natural programs, subprograms, subroutines, classes, adapters, 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 Editor	Describes the DDM editor which is used to create, maintain and delete Natural data definition modules (DDMs).
Dialog Editor	Describes the dialog editor which is used to create and modify applications with the following basic components: dialogs, dialog elements, attributes, event handlers, data areas (local and parameter; global data areas can be referenced) and inline subroutines.
Class Builder	Describes the class builder which is used to display Natural classes in structured hierarchical orders and to manage the classes and their components.
Editor Features With SPoD	Indicates server-specific editor features available with Natural's Single Point of Development.

1 About this Documentation

- Document Conventions 2
- Online Information and Support 2
- Data Protection 3

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 NaturalONE as the Default Development Environment

2 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.

II Program Editor

The Natural program editor is used to create and modify the source code of a Natural object of the type program, subprogram, function, subroutine, copycode, helproutine or text. Additionally, you can edit but not create an object of the type class or adapter. For detailed descriptions of these object types, refer to the section *Objects for Natural Application Management* in the *Programming Guide*.

You can open multiple editor sessions, making it possible to copy or move source code from one object to another.

Invoking the Program Editor
Creating and Modifying Source Code
Finding Text
Replacing Text
Setting Bookmarks
Going to a Line Number
Toggling Breakpoints
Importing Data Fields
Editing or Listing Referenced Objects
Setting Display Modes
Saving Source Code
Using Context-Sensitive Help

Related Topics:

- *Editors in the SPoD Environment* in the *Unicode and Code Page Support* documentation
- *Setting the Options* and *Program Editor Options* in the *Using Natural Studio* documentation
- *Toolbars* in the *Using Natural Studio* documentation
- *Shortcut Keys* and *Program Editor Shortcut Keys* in the *Using Natural Studio* documentation

3 Invoking the Program Editor

You invoke the program editor when you create a new object or open an existing object of type program, class, subprogram, function, subroutine, capacity, helproutine or text.

This section provides instructions for invoking the program editor.

To invoke the editor for a new object

- From the **Object** menu, choose **New** and the type of object (for example, program) you want to create.

  Or:

  In the library workspace, select the required library node, invoke the context menu and choose **New Source** and the type of object you want to create.

  Or:

  For an object of type program, press CTRL+N (shortcut keys are not provided for other object types).

  Or:

  Choose the toolbar button that corresponds to the type of object you want to create (toolbar buttons are not available for all object types).

  Or:

  In the command line, enter the following system command:

  ```
  EDIT object-type
  ```

  where *object-type* is the type of object to be created.

For example, to create an object of type program, enter the following:

```
EDIT PROGRAM
```

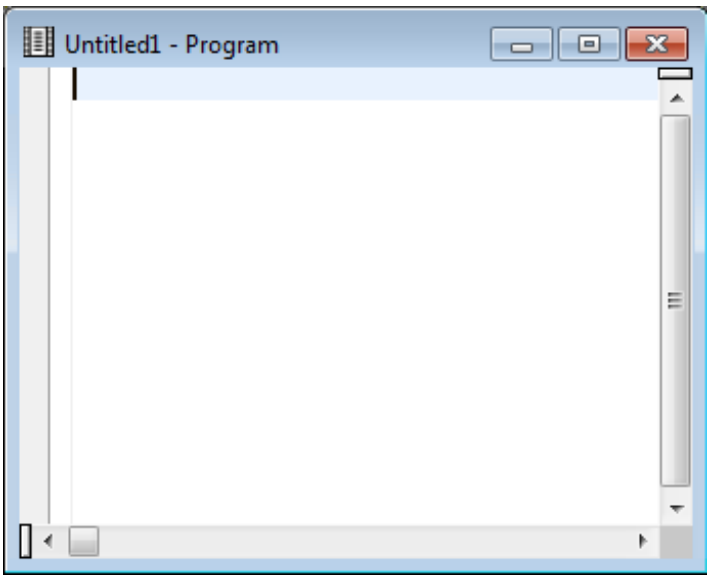
Or:

Use the type code (`P`) that corresponds to a program:

```
EDIT P ↩
```

For detailed information on the command syntax, see `EDIT` and *`object-type`* in the *System Commands* documentation.

The program editor is invoked and an empty editor window similar to the example below appears for the selected object type:



The title bar of the editor window displays the name and the type of the object (here: **Program**). If the object is new and has not yet been saved as a source object, the object is named **Untitled1** where **1** denotes that this is the first editor window that has been opened for a new object during the current Natural session. Each additional editor window for a new object increments the number assigned to **Untitled** by one.

To invoke the editor for an existing object

- From the library workspace or application workspace, select the object you want to open, invoke the context menu and choose **Open**.

  Or:

  Double-click on an object.

Or:

Select an object and choose the following toolbar button:

Or:

In the command line, enter the following system command:

```
EDIT object-name
```

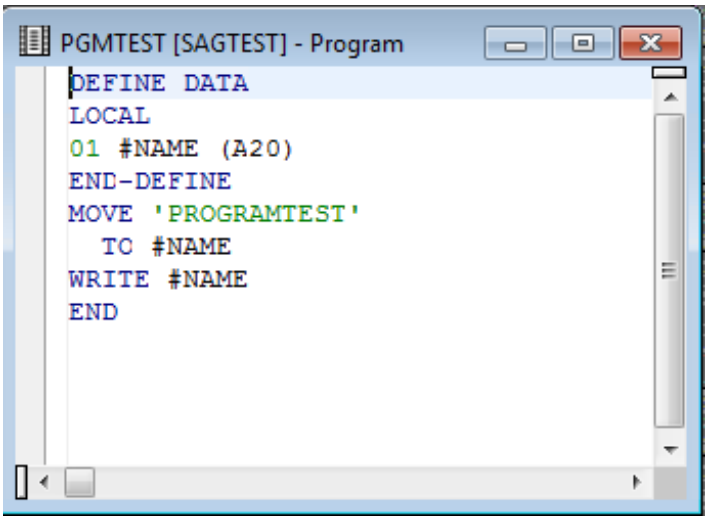
where *object-name* is the name of an existing source object.

For example, to modify an object with the name PGMTEST, enter the following:

```
EDIT PGMTEST
```

For detailed information on the command syntax and *object-name*, see EDIT in the *System Commands* documentation.

If the specified object exists in the current library and system file, the program editor is invoked as shown in the following example:



The source code of the specified object is contained in the editor window where the current line is highlighted (default). You can switch highlighting on or off or change the highlight color by setting the corresponding editor option as described in *Program Editor Options* in the *Using Natural Studio* documentation.

4 Creating and Modifying Source Code

- Entering and Inserting Text 16
- Selecting Text 17
- Copying, Cutting and Pasting Text 18
- Dragging and Dropping Text 19
- Undoing or Redoing Modifications 20
- Deleting Text 21
- Adding and Removing Comments 21
- Renumbering Lines 23
- Translating to Upper or Lower Case 23
- Using Automatic Line Breaks 24
- Converting to and from Hexadecimal Format 26
- Recording, Replaying and Saving Keystrokes 28
- Protecting Source-Code Lines 31

The program editor supports bi-directional languages where text is written from right-to-left (for example, Arabic and Hebrew), and multi-byte character set languages such as Japanese and Chinese.

We recommend that you create source code in structured mode rather than reporting mode. Otherwise, you cannot use the expand/collapse editor feature (see ***Showing and Hiding Source Code***).

For the differences between structured and reporting mode, see the section *Natural Programming Modes* in the *Programming Guide*. If you want to check whether structured mode has been activated or switch structured mode on, proceed as described in *Programming Modes* in the *First Steps* documentation.

For programming advice, see ***Using Context-Sensitive Help*** and the documentation in the Language section of the main overview page of Natural for Windows.

Entering and Inserting Text

You start entering text in the first line (numbered with `0010`) of an empty editor window.

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 numbers the inserted lines in increments of one. Upon saving the source code, Natural automatically renumbers all lines contained in the source code in increments of ten beginning with `0010` at the first line.

Apart from that, you can renumber all source-code lines any time by using the renumber command as described in ***Renumbering Lines***.

To insert text

- Place the cursor at the position in the source code where you want to insert text (the current line is highlighted by default) and type in new text or paste text from the clipboard (see ***Copying, Cutting and Pasting Text***). See also ***Importing Data Fields***.

  Or:

  To insert a blank line, place the cursor at the beginning of the source-code line above which you want the new line to be inserted and press ENTER.

  The new text is inserted in the source code.

Selecting Text

You can select one of the following:

- A whole word.
- A series of consecutive characters within a source-code line.
- A series of consecutive characters that spans one or more lines.

To select a whole word

- Double-click on the word to be selected.

  Or:

  Position the cursor on the first character to be selected by using either the left mouse button or the arrow keys. Drag the cursor to the last character to be selected by pressing and holding down SHIFT and using the arrow keys.

  The selected word is highlighted.

To select any portion of text

1. Position the cursor on the first character to be selected by using either the left mouse button or the arrow keys.
2. Drag the cursor to the last character to be selected by holding down the left mouse button or by pressing and holding down SHIFT and using the arrow keys.

   The selected text is highlighted.
3. Release the mouse button or the keys.

To select the entire text

- From the **Edit** menu, choose **Select all**.

  Or:

  Choose the following toolbar button:

To deselect text

- Click anywhere in the source code.

Or:

Press an arrow key.

The text is no longer highlighted.

Copying, Cutting and Pasting Text

The copy/cut and paste functions of the program editor are used to copy, move or delete text within a single object source or between multiple object sources (see also *Dragging and Dropping Text*).

It is possible to revoke a cut or paste operation after it has been performed. See *Undoing or Redoing Modifications*.

If you want to create data fields, as an alternative to using copy and paste, you can import fields from another source object as described in *Importing Data Fields*.

To copy or cut and paste text

1 Select the text you want to copy or cut as described in *Selecting Text*.

2 From the **Edit** menu, choose **Copy** or **Cut**.

Or:

Choose either of the following toolbar buttons:

(**Copy**) or (**Cut**)

Or:

Press CTRL+C (to copy) or CTRL+X (to cut).

The selected text is placed on the clipboard and can be pasted in the source code contained in the active editor window.

3 If the text is to be pasted in another object source, open the appropriate source.

4 Position the cursor on the source-code line above which the copied or cut text is to be pasted.

5 From the **Edit** or context menu, choose **Paste**.

Or:

Choose the following toolbar button:

Or:

Press CTRL+V.

The text is pasted at the specified position in the source code contained in the active editor window.

6 If you want to paste the same text again, repeat Steps 4 and 5.

Dragging and Dropping Text

Drag-and-drop operations can be used to move or copy text. The drag source and drop target may be Natural Studio or a different application that supports drag-and-drop.

It is possible to revoke a drag and drop operation after it has been performed. See ***Undoing or Redoing Modifications***.

To move or copy text using drag-and-drop functionality

1 Select the text you want to move or copy (see also ***Selecting Text***) and position the mouse pointer anywhere in the selected (highlighted) text.

2 Press and hold down the left mouse button and move the mouse. In addition, hold down CTRL if you want to copy the text instead of moving it which is the default.

The mouse pointer changes to an arrow pointer inside a rectangle indicating that drag-and-drop editing is enabled. For a copy operation, a plus sign (+) is attached to the rectangle. A text insertion caret next to the arrow pointer indicates the position where the selected text can be dropped.

If the mouse pointer changes to a “no drop” pointer (slashed circle), the current target location does not accept a drop. This happens when you attempt to insert text into protected source code (see also ***Protecting Source-Code Lines***) or when the drop target has no drag-and-drop capabilities. If you execute a move operation on protected source code, the text will be copied rather than moved to the target location.

You can press ESC if you want to cancel drag-and-drop editing.

3 Release the mouse button where you want to drop the text.

The selected text is dropped at the current position of the caret.

If the mouse button is released when the “no drop” pointer is visible, the drag-and-drop operation is canceled.

Undoing or Redoing Modifications

The editing actions you perform in the editor window can be revoked by using the **Undo** command or redone (after being undone) by using the **Redo** command. The **Undo** or **Redo** command applies to any source-code modifications that are *not* made via the command line (for example, `RENUMBER` or `STRUCT`).

The number of actions which can be undone or redone is determined by the value specified in the preferences for the program editor and is limited by the memory allocated: see the editor options **Max undo mem (MB)** and **Max no of undo** described in *Program Editor Options* in the *Using Natural Studio* documentation.

The undo/redo buffer is cleared when you close the editor window.

Undo operations restore line numbering to its status before the operation. Redo operations restore line numbering to its status before the previous undo operation.

To undo an action

- From the **Edit** or context menu, choose **Undo**.

  Or:

  Press CTRL+Z.

  Or:

  Choose the following toolbar button:

  The source code is restored to its condition before the previous editing action or redo operation.

To redo an action that has been undone

- From the **Edit** menu, choose **Redo**.

  Or:

  Press CTRL+Y.

  Or:

  Choose the following toolbar button:

The source code is restored to its condition before the previous undo operation.

Deleting Text

When you delete text, it is cut from the object but is *not* placed on the clipboard. The only way to recover deleted text is by undoing the delete operation. See *Undoing or Redoing Modifications*.

Note that when text is deleted from a source, no warning is provided.

To delete text

1 Select text as described in *Selecting Text*.

2 From the **Edit** or context menu, choose **Delete**.

Or:

Choose the following toolbar button:

Or:

Press DEL.

The text is deleted from the source code contained in the active editor window.

Adding and Removing Comments

You can mark or unmark an entire source-code line as a comment line or place a commentary text within a line.

To add a comment line

- At the beginning of a source-code line, enter an asterisk (*) followed by at least one blank character. For example: `* This is a comment line.`

  Or:

  At the beginning of a source-code line, enter two asterisks (**), with or without subsequent blanks. For example: `**This is a comment line.`

Or:

At the beginning of a source-code line, enter the character string slash-asterisk (/*), with or without subsequent blanks. For example: `/*This is a comment line.`

To append a comment to a statement

- In a source-code line, at the end of a statement, enter the character string blank-slash-asterisk (/*).

  For example: `DEFINE DATA /*This is a comment.`

To mark lines as comment lines

1. Place the cursor anywhere in the source-code line you want to mark.

   Or:

   Select one or more source-code lines as described in *Selecting Text*.
2. From the **Edit** or context menu, choose **Advanced** > **Add Comment Mark(s)**.

   Or:

   Press CTRL+M.

   A comment mark (*) followed by a blank character is added to the beginning of the specified source-code line(s).

To unmark lines as comment lines

1. Place the cursor anywhere in the comment line you want to unmark.

   Or:

   Select one or more source-code lines as described in *Selecting Text*.
2. From the **Edit** or context menu, choose **Advanced** > **Remove Comment Mark(s)**.

   Or:

   Press CTRL+SHIFT+M.

   A comment mark (`*` , `**` or `/*`) is removed from the beginning of the specified source-code line(s).

Renumbering Lines

To renumber source-code lines

- From the **Edit** menu, choose **Renumber**.

  Or:

  Choose the following toolbar button:

  Or:

  In the command line, enter the following system command:

  ```
  RENUMBER
  ```

  See also RENUMBER in the *System Commands* documentation.

  The source-code lines of the object in the active window are renumbered in increments of ten.

Related Topic:

- *Renumbering of Source-Code Line Number References - Programming Guide*

Translating to Upper or Lower Case

You can translate source code from upper to lower case or from lower to upper case.

To translate text from upper to lower case or vice versa

1. Select the text you want to translate as described in *Selecting Text*.
2. From the **Edit** or context menu, choose either **Advanced** > **Upper Case** or **Advanced** > **Lower Case**.

   Or:

   Press CTRL+SHIFT+U for upper case or CTRL+SHIFT+L for lower case.

You can switch automatic upper-case translation on by setting the corresponding editor option as described in *Program Editor Options* in the *Using Natural Studio* documentation. With automatic upper-case translation, the entire source code (except commentary text) is translated to upper case

whenever you save, check or catalog the source code. Automatic upper-case translation is useful in a remote environment on a mainframe computer where the LOWSRCE (Allow Lower-Case Source) compilation option is set to OFF. For details, see LOWSRCE in COMPOPT in the *System Commands* documentation for mainframes.

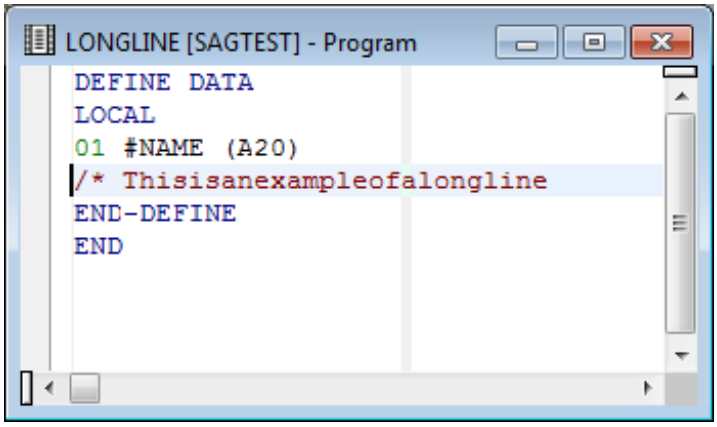
Using Automatic Line Breaks

The program editor provides the option of inserting automatic line breaks at a specified column while editing text or after selecting existing text.

The following rules apply when this option is active:

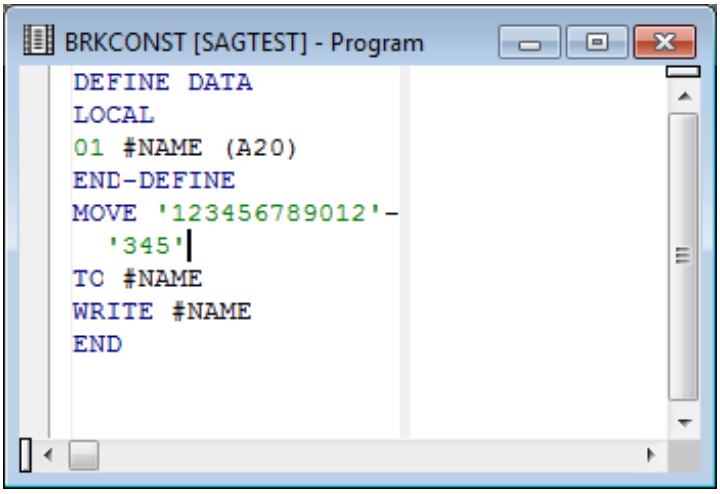
- A line break is inserted whenever the cursor crosses the ruler while entering new text or modifying existing text. (For information on setting a ruler position, see *Program Editor Options* in the *Using Natural Studio* documentation.)
- A line break is inserted at the blank character located next to the ruler (blank characters at the beginning of a line are ignored).

  If there is no blank, the editor generates a long line which extends beyond the ruler as shown in the example below:

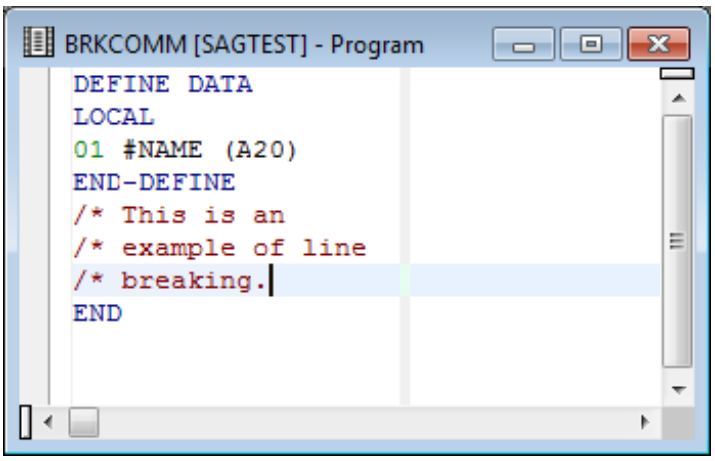


- If you insert text or replace text strings that generate a long line, no automatic line break is performed. A line break will only be inserted if you modify this text and move over the ruler.
- A text constant is continued on the next line(s).

  An apostrophe (') or a quotation mark (") followed by a minus sign (-) is inserted at the line break where the cursor crosses the ruler or, if the constant contains one or more blank characters, at the blank located next to the ruler. Additionally, an apostrophe (') or a quotation mark (") is added to the beginning of the new line(s). For example:



- A comment is continued on the next line(s) if it contains a blank character or if it is preceded by a blank character. The mark that indicates a comment (*, ** or /*) is automatically added to the beginning of the new line(s). For example:



- If BACKSPACE is used at the end of a split line, this line will be concatenated with the next one. Any marks resulting from a previous line break will be retained in the new line and must be removed manually.

To activate automatic line breaking

1. In the **Options** dialog box, set the position of the ruler by using the **Ruler position** spin box and select the **Automatic line breaks** check box as described in *Program Editor Options* in the *Using Natural Studio* documentation.

   A line break is inserted automatically whenever you move the cursor over the ruler while typing in text.

To insert automatic line breaks for existing text

1 Select the desired text section.

2 From the **Edit** or context menu, choose **Advanced** > **Break at Ruler position**.

Automatic line breaks are inserted into the text according to the rules described earlier.

Converting to and from Hexadecimal Format

The program editor provides options to display the hexadecimal Unicode format of characters contained in a text constant and to convert these characters to their equivalent hexadecimal format. You can convert characters either to hexadecimal code page format or hexadecimal Unicode format, or vice versa. For conversion to hexadecimal code page format, the code page defined for the source is used. If no code page is defined for the source, the code page defined for the current Natural Studio session is used.

To display hexadecimal Unicode format

- Hover the mouse pointer over a character contained in a text constant to display the hexadecimal value of the character.

  Or:

  Select multiple characters from a text constant and hover the mouse pointer over the selected characters to display the hexadecimal value of the characters.

  A tool tip shows the hexadecimal Unicode representation of the character(s).

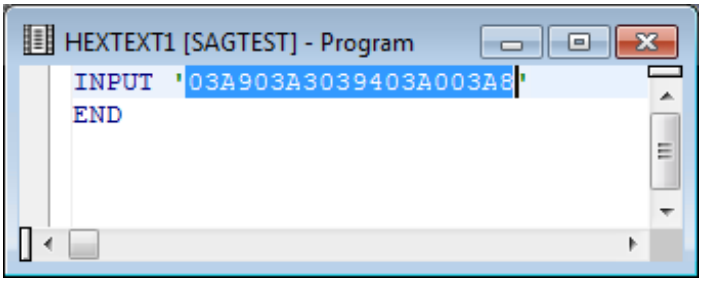
To convert characters to hexadecimal format

1 For hexadecimal code page format, select single or multiple characters from a text constant and press CRTL+ALT+X.

  Or:

  For hexadecimal Unicode format, select single or multiple characters from a text constant and press CRTL+ALT+U.

2 The selected character(s) are replaced by the equivalent hexadecimal format as shown in the Unicode example with the Greek letters ΩΣΔΠΨ below:

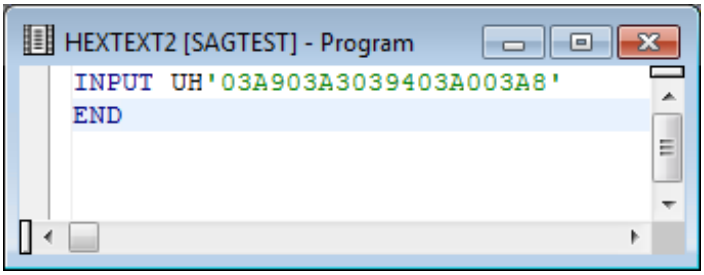


For hexadecimal code page format: a character that cannot be converted will be replaced by the substitution character of the defined code page.

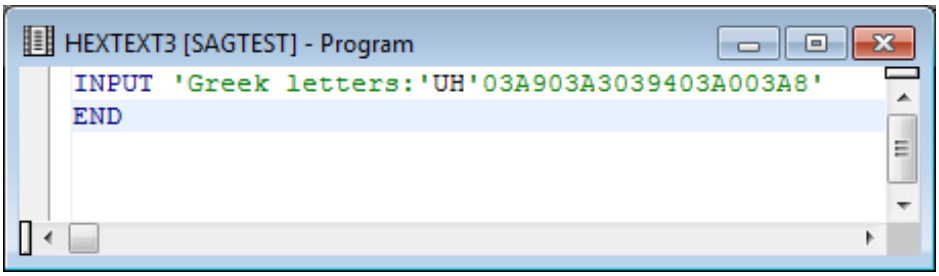
3 Place either of the following prefixes before the apostrophe that precedes the constant:

- For hexadecimal Unicode format: `UH`
- For hexadecimal code page format: `H`

The constant must start with the hexadecimal value as shown in the Unicode example below:



Therefore, if you only convert part of a text constant, you may have to split the constant into multiple constants to avoid syntactical errors as shown in the Unicode example below:



To convert hexadecimal to character format

- Select the hexadecimal string you want to convert and press CRTL+ALT+C. For example:

  All digits selected in `'41424344'` are converted to `'ABCD'`.

If the hexadecimal string selected is prefixed with `UH` or `H`, it is interpreted as a Unicode character.

For example:

All digits selected in `U'0041004200430044'` or `UH'0041004200430044'` are converted to `U'ABCD'` or `UH'ABCD'` respectively.

A hexadecimal value that cannot be interpreted by the defined code page is substituted by non-printable characters.

For example:

All digits selected in `'0041004200430044'` are converted to `'[NUL]A[NUL]B[NUL]C[NUL]D'`.

Recording, Replaying and Saving Keystrokes

The keystroke recorder records and replays only input from the keyboard; mouse movements and operations are ignored. The keystroke sequence recorded last is retained for the duration of the Natural Studio session. The keystroke sequence can be saved as a macro for permanent use during any subsequent sessions.

This section contains the following topics:

- Recording and Replaying Keystrokes
- Handling Macros

Recording and Replaying Keystrokes

To record a keystroke sequence

1 Place the cursor at the position where you want to begin recording.

2 From the **Tools** menu, select **Macros** > **Start Recording**.

Or:

Choose the following toolbar button:

Or:

Press CTRL+SHIFT+R.

The mouse pointer changes to an arrow pointer with a symbol illustrating a cassette tape. This indicates that recording is in progress.

3 Press the keys in the sequence you want to record.

To stop recording

- From the **Tools** menu, select **Macros** > **Stop Recording**.

  Or:

  Choose the following toolbar button:

  Or:

  Press CTRL+SHIFT+S.

  Recording ends. The recording is retained for the duration of the current session and is overwritten by any subsequent recordings.

To replay a recorded keystroke sequence

1 Place the cursor at the position where you want to begin replaying the recording.

2 From the **Tools** menu, select **Macros** > **Replay Recording**.

  Or:

  Choose the following toolbar button:

  Or:

  Press CTRL+SHIFT+P.

  The keystroke sequence recorded last is replayed.

Handling Macros

You can save a recording as a macro, which can be used permanently, in any Natural Studio session. You can add a most frequently used macro or a macro command to the toolbar or menu bar or create shortcuts. You can overwrite macro code or delete a macro that is no longer required.

To save a recording as a macro

1 When you have stopped recording, from the **Tools** menu, choose **Macros** > **Save Macro**.

The **Save Macro** dialog box appears.

2 In the **Name** box, type a name of up to 32 characters and choose **OK** to save the macro.

If a macro with the specified name already exists, an appropriate message appears asking whether to overwrite the contents of the existing macro or save the macro under a different name: choose **Yes** to overwrite or **No** to rename the macro.

The macro is saved under the specified name, added to your user profile and listed in the **Macros** submenu of **Tools**.

You can save a maximum of 16 macros.

To run a macro

1 From the **Tools** menu, choose **Macros**.

All macros saved in your user profile are listed in the submenu.

2 From the macro list, choose the required macro.

The macro is executed and replays the recorded keystroke sequence.

To delete a macro

1 From the **Tools** menu, choose **Macros** > **Delete Macro**.

The **Delete Macros** window appears with a list of all macros saved in your user profile.

2 Select the macro you want to delete and choose **Delete**.

The macro is deleted from the list.

3 Choose **OK** to confirm the action and remove the macro from your user profile.

Or:

Choose **Cancel** to exit the dialog box without any action.

To add a macro or a macro command to the toolbar or menu bar

1 Invoke the **Customize** dialog box and open the **Commands** page as described in *Customizing Natural Studio* in the *Using Natural Studio* documentation.

2 From the **Categories** drop-down list box, select **Tools.Macros**.

All macro commands and available macros are listed in the **Commands** list box.

3 Select a command or a macro and drag it to the menu bar. For detailed instructions, see *Customizing Natural Studio*.

To assign a shortcut to a macro or a macro command

1 Invoke the **Customize** dialog box and open the **Keyboard** page as described in *Customizing Natural Studio*.

2 From the **Categories** drop-down list box, select **Tools.Macros**.

All macro commands and available macros are listed in the **Commands** list box.

3 Select a command or a macro and press the key combination you want to assign as a shortcut to the selected command or macro. For detailed instructions, see *Keyboard* in *Customizing Natural Studio*.

The keyboard shortcut is now assigned to the macro or macro command and appears next to the associated macro command or macro name listed in the **Macros** submenu of **Tools**.

Protecting Source-Code Lines

You can protect a single source-code line or a block of source-code lines against unauthorized modification by using read-only tags.

Protected source-code lines cannot be edited. In addition, you cannot add lines to the beginning or end of source code where a protected block of lines starts in the first line (`0010`) and ends in the last (here: `0090`) as shown in ***Example of Line Protection***. You can only add lines if the protected block is preceded or followed by an unprotected line.

The following editor functions cannot be performed on protected source-code lines:

- Cut and paste
- Delete
- Replace
- Add/remove comment mark(s)
- Upper/lower case translation

- Insert/add line at top/bottom of protected block that starts in the first line of the source and ends in the last

Caution: Read-only tags cannot be modified or deleted once the corresponding source object has been reloaded into the source buffer.

Read-only tags are stored in the source object in the Natural system file. They are removed from the source buffer and are therefore hidden when the corresponding source object is reloaded into the source buffer. A source object is reloaded into the source buffer, for example, when you execute a system command such as `STRUCT` or `RENUMBER` on the source, or when you open the source object in the editor window.

You can exclude protected source-code lines from structural indentation as described for the `STRUCT` command under *Indentation of Source Code Lines* in the *System Commands* documentation.

In a reloaded source object where read-only tags are hidden, protected source-code lines are grayed out as shown in ***Example of Line Protection***. (You can specify a different background color by changing the color definition as described in *Program Editor Options* in the *Using Natural Studio* documentation.)

To protect a single line

- At the end of the line to be protected, enter a blank character and append the following tag:

```
/*<RO>
```

To protect a code block

1. At the end of the line where the block to be protected begins, enter a blank character and append the following tag:

```
/*<RO>>
```

2. At the end of the line where the block to be protected ends, enter a blank character and append the following tag:

```
/*<<RO>
```

Read-only tags are reinserted into the source buffer whenever you save the source object. You can prevent read-only tags from being reinserted by following the instructions below.

To stop reinsertion of read-only tags

- At the end of the first source-code line, enter a blank character and append the following tag:

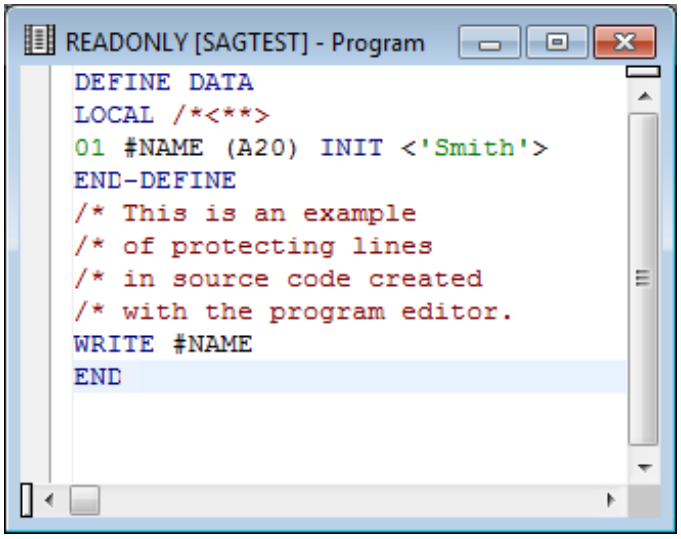
```
/*<**>
```

If the first source-code line contains a read-only tag, place this tag immediately (not separated by a blank character) in front of the read-only tag.

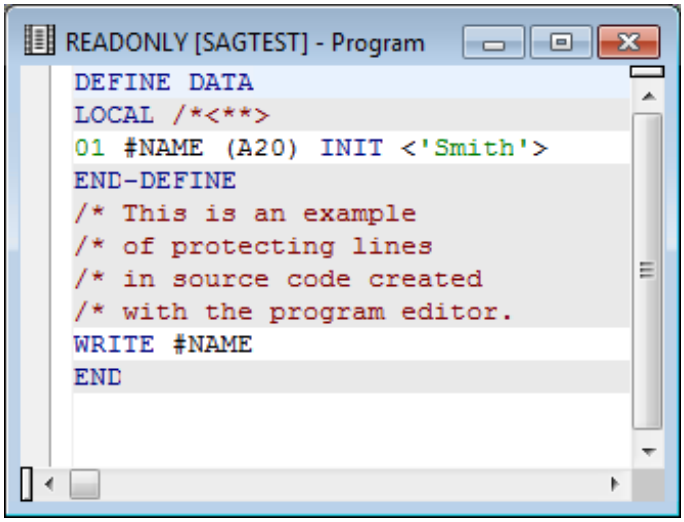
See also *Example of Line Protection* below.

Example of Line Protection

In the example of a source object below, the lines of the `DEFINE DATA` statement and the consecutive comment lines are protected with read-only tags. The tag `/*<**>` guarantees that the read-only tags are not reinserted into the source buffer.



Read-only tags are hidden when the source object is reloaded into the source buffer as shown in the example below. Protected lines are grayed out.



5 Finding Text

You can search for any string of characters contained in the source code of an active editor window by using one of the following methods:

- Mark a search string in the source to highlight all matches.
- Type text in the source to immediately find the next match by using the **Incremental Find** function.
- Specify additional search criteria to find matches in single or multiple program editor windows by using the **Find** function.

By default, **Incremental Find** and **Find** restart a search from the beginning when the end of the source code is reached. If you want to terminate the find operation when the end of the source is reached, set the **Stop find at end** editor option described in *Program Editor Options* in the *Using Natural Studio* documentation.

To find a string by marking text

1 Select a whole word by positioning the cursor anywhere in the word you want to find. See also ***To select a whole word*** in the section *Creating and Modifying Source Code*.

Or:

Select any series of characters. See also ***To select any portion of text*** in the section *Creating and Modifying Source Code*.

Note that the character string may *not* span one or more lines.

2 Press the CTRL+SPACEBAR toggle.

All instances of the search string found are highlighted.

3 If you want to switch highlighting off, press ESC or position the cursor anywhere within a highlighted word or in the white space areas of the source code and press CTRL+SPACEBAR.

To find a string by using Incremental Find

1 From the **Edit** or context menu, choose **Incremental Find**.

Or:

Press CTRL+J.

The **Incremental Find** mode is set as indicated by the corresponding message in the status bar: `Incremental find started.`

2 Type any text.

As you type text, the first match for this text is highlighted. The search string is updated for each character typed. The current search string is indicated in the status bar.

3 You can press DOWN-ARROW or `UP-ARROW` to go to the next or previous match respectively.

4 You can terminate the find operation by pressing ESC or ENTER.

The **Incremental Find** mode is deactivated as indicated by the corresponding message in the status bar: `Incremental find canceled.`

To find a string by using Find

1 From the **Edit** or context menu, choose **Find**.

Or:

Choose the following toolbar button:

Or:

Press CTRL+F.

The **Find** dialog box appears.

2 In the **Find** dialog box, you can specify the following:

- In the **Find** combo box, enter the character string to be found or select a character string from the drop-down list box. All character strings used in previous searches are retained in the list box for the duration of the current Natural session.
- Select the **Case sensitive** check box (not selected by default) if you want to search for strings that exactly match the entry in the **Find** text box. Otherwise, any combination of upper and lower-case letters will be found.
- Select the **Whole words only** check box (not selected by default) if you want to restrict the search to whole words only. Otherwise, all occurrences of the string will be found.

- Select the **Exclude collapsed blocks** check box to exclude collapsed blocks from the search; do *not* select the box (this is the default setting) if you want to scan the entire (expanded) source code. **Exclude collapsed blocks** is only available if structured mode was used for the source(s) to be scanned and if the **Expand/Collapse** editor option is selected (see *Program Editor Options* in the *Using Natural Studio* documentation).

  For more information on expanded and collapsed code blocks, see ***Showing and Hiding Source Code***.
- Select the **Search up** check box to perform the search from the current position of the text insertion caret to the beginning of the source (up). If this check box is not selected (this is the default setting), the search is performed from the current caret position to the end of the source (down).
- Select the **Highlight occurrences** check box if you want to immediately view all instances of the search string found. Otherwise, the first instance found is selected. To switch highlighting off, see **Step 3** of *To find a string by marking text*.

  For a different highlight color, use the **Colors** editor option and change the color definition of **Highlighted text** as described in *Program Editor Options* in the *Using Natural Studio* documentation.
- Select the **Current editor window** radio button (selected by default), if you want to scan only the source code contained in the current window of the program editor.
- Select the **All program editor windows** radio button, if you want to scan the source code contained in all open windows of the program editor.
- You can choose the **Help** button to invoke the online documentation for help information on finding text.

3 Choose the **Find Next** button to execute the find function.

Depending on the setting of the **Search up** check box, the find goes down or up the source code from the current caret position:

- If no instance of the search string is found, a corresponding message is displayed.
- If an instance of the search string is found, it is selected.
- If you selected **Highlight occurrences**, the first instance found is selected and all other instances are highlighted.

4 If the first instance of a search string is selected, you can go to the next instance by choosing one of the following methods:

In the **Find** dialog box, choose the **Find Next** button.

Or:

From the **Edit** or context menu, choose **Find Next**.

Or:

Choose the following toolbar button:

Or:

Press F3.

5 You can choose the **Close** button to close the **Find** dialog box. However, you can also edit the current source when the **Find** dialog box is open.

When the **Find** dialog box is closed, you can continue the search function by using the alternative methods described earlier in Step 4.

6 Replacing Text

You can replace any string of characters contained in the source code of an active editor window.

By default, **Replace** restarts a search from the beginning when the end of the source code is reached. However, you can change the default setting and instruct **Replace** to terminate when the end of the source is reached by setting the **Stop find at end** editor option described in *Program Editor Options* in the *Using Natural Studio* documentation.

To replace a character string

1 From the **Edit** or context menu, choose **Replace**.

Or:

Choose the following toolbar button:

Or:

Press CTRL+H.

The **Replace** dialog box appears.

2 In the **Replace** dialog box, you can specify the following:

- In the **Find** combo box, enter the character string to be found or select a string from the drop-down list box. All character strings used in previous searches are retained in the list box for the duration of the current Natural session.
- In the **Replace with** combo box, enter a replacement string or select a string from the drop-down list box. All character strings used for previous replacements are retained in the list for the duration of the current Natural session.

- Select the **Case sensitive** check box (not selected by default) to find and replace only characters strings that exactly match the entry in the **Find** text box. Otherwise, any combination of upper and lower-case letters will be found and replaced.
- Select the **Whole words only** check box (not selected by default) to find and replace whole words only. Otherwise, all occurrences of the string will be found and replaced.
- Select the **Exclude collapsed blocks** check box to exclude collapsed blocks from the search and replace functions; do *not* select the box (this is the default setting) if you want to search and replace in the entire (expanded) source code. **Exclude collapsed blocks** is only available if structured mode was used for the source(s) to be scanned and if the **Expand/Collapse** editor option is selected (see *Program Editor Options* in the *Using Natural Studio* documentation).

  For more information on expanded and collapsed code blocks, see ***Showing and Hiding Source Code***.
- Select the **Search up** check box to perform the search and replace functions from the current position of the text insertion caret to the beginning of the source (up). If this check box is not selected (this is the default setting), the search and replace functions are performed from the current caret position to the end of the source (down).
- Select the **Current editor window** radio button (selected by default), if you want to execute the search and replace functions only in the source code contained in the current window of the program editor.
- Select the **All program editor windows** radio button, if you want to execute the search and replace functions in the source code contained in all open windows of the program editor.
- Select the **Selection** radio button, if you want to execute the **Replace All** function for a selected portion of source code. This button is only enabled, if the text selected in the current editor window spans one or more lines. The buttons **Find Next** and **Replace** are then disabled.
- You can choose the **Help** button to invoke the online documentation for help information on replacing text.

3 Choose the button **Find Next**, **Replace** or **Replace All** to execute the find and/or replace function.

Depending on the setting of the **Search up** check box, the find and/or replace goes down or up the source code from the current caret position. If no instance of the search string is found, a corresponding message is displayed. If an instance of the search string is found, the following applies:

- If **Find Next** was selected, the search string is selected.
- If **Replace** was selected, the previously selected search string is replaced by the replacement string.
- If **Replace All** was selected, all search strings found are replaced by the replacement string.

4 If the first instance of a search string is selected, you can go to the next instance by choosing one of the following methods:

From the **Edit** or context menu, choose **Find Next** or **Replace Next**.

Or:

Choose one of the following toolbar buttons:

for **Find Next**

for **Replace Next**

Or:

Press F3 for **Find Next** or CTRL+F3 for **Replace Next**.

5 You can choose the **Close** button to close the **Replace** dialog box. However, you can also edit the current source when the **Replace** dialog box is open.

When the **Replace** dialog box is closed, you can continue the find and/or replace functions by using the alternative methods described earlier in Step 4.

7 Setting Bookmarks

You can use bookmarks to mark single or multiple source-code lines you want to quickly identify and access.

This section provides instructions for setting and clearing bookmarks and navigating between bookmarks.

To set bookmarks

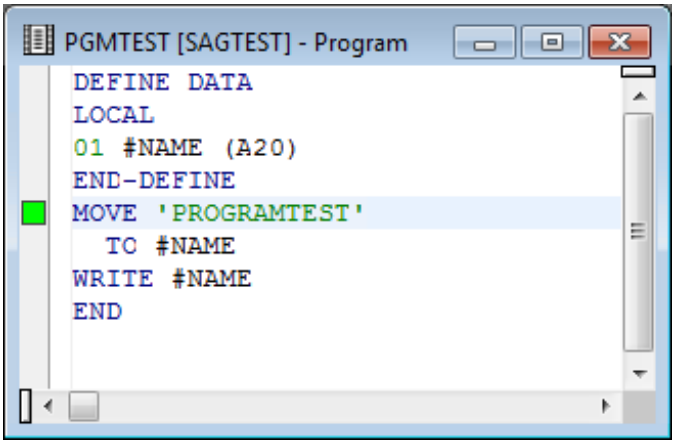
1 Position the cursor anywhere in the source-code line you want to mark.

2 From the **Edit** or context menu, choose **Advanced** > **Toggle Bookmark**.

Or:

Press the CTRL+ALT+B toggle.

A square appears to the left of the specified source-code which indicates the bookmark set as shown in the following example for line 0050:



3 Repeat Steps 1 and 2 if you want to mark additional lines.

To clear a single bookmark

1 Position the cursor anywhere in the source-code line you want to unmark.

2 From the **Edit** or context menu, choose **Advanced** > **Toggle Bookmark**.

Or:

Press the CTRL+ALT+B toggle.

The bookmark indicator disappears for the specified line.

To clear all bookmarks

- From the **Edit** or context menu, choose **Advanced** > **Clear Bookmarks**.

Or:

Press CTRL+ALT+L.

All bookmark indicators disappear.

To go to the next bookmark

- From the **Edit** or context menu, choose **Advanced** > **Next Bookmark**.

Or:

Press CTRL+ALT+N.

To go to the previous bookmark

- From the **Edit** or context menu, choose **Advanced** > **Previous Bookmark**.

Or:

Press CTRL+ALT+P.

8 Going to a Line Number

If you know the number of the source-code line you want to find, you can use the **Go To** command to position the cursor on a specific line. Depending on whether the line number option is switched on or off (this is the default setting), you can specify to search for either a numbered line or a physical line. See also *Program Editor Options* in the *Using Natural Studio* documentation.

If you use the **Go To** command in source code where the specified line is within a collapsed code block, the block is expanded to show the required line. For more information on expanded and collapsed code blocks, see ***Showing and Hiding Source Code***.

To go to a specific line

1 From the **Edit** or context menu, choose **Go To**.

Or:

Choose the following toolbar button:

Or:

Press CTRL+G.

The **Go To** dialog box appears.

2 If the **Line numbers** option is switched on (source-code line numbers are displayed), select the **Numbered Line** option button. If the **Line numbers** option is switched off (source-code line numbers are not displayed), select **Physical Line**. The physical line number is the default setting.

3 In the **Line Number** text box, enter the line number for which to search.

4 Choose **Go To**.

The editor scrolls to the specified line and the cursor is placed at the beginning of the line.

9 Toggling Breakpoints

You can use the **Toggle Breakpoint** function of the context menu to set or remove a breakpoint. For detailed instructions, see *Adding and Removing a Breakpoint* in the section *Setting Breakpoints and Watchpoints* in the *Debugger* documentation.

10 Importing Data Fields

You can import data field definitions from another source into your current source code.

To import a data field

1 Position the cursor in the source code where you want to place the imported data field(s). This is usually the first position of a line within the `DEFINE DATA` block.

2 From the **Program** menu, choose **Import**.

The **Import Data Field** dialog box appears.

3 From the **Library** drop-down list box, select the library that contains the source object from which you want to import data fields.

The list contains all libraries that reside in the current FNAT and FUSER system file, which are displayed as nodes in the Natural Studio tree view (the display can be limited by using the Display Filter function of Natural Studio). In addition, the list contains all libraries from inactive system files as specified in the steplib table.

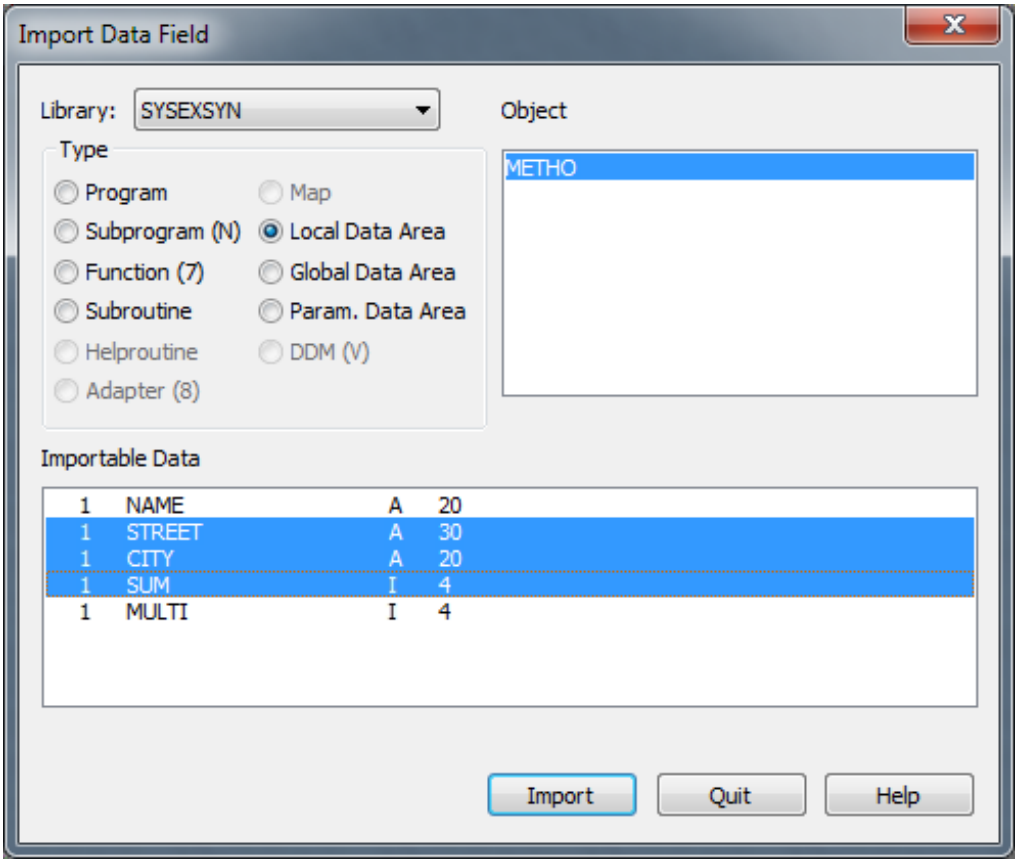
4 In the **Type** group frame, select the object type of the required source object.

All source objects of the selected object type that are contained in the specified library are shown in the **Object List** list box.

The list contains all objects, which are displayed in the library nodes of the Natural Studio tree view (the display can be limited by using the Display Filter function of Natural Studio).

5 From the list of source objects, select the required source object.

All data fields that can be imported from the selected source object are listed as shown in the example below:



6 Select the field(s) you want to implement in the source code and choose the **Import** button as shown in the example above.

The data field(s) are placed at the specified position in the current source code.

11 Editing or Listing Referenced Objects

From the source code contained in the active editor window, you can open or list other objects which are referenced in this source, assuming these objects exist. For example, if you are editing a program that calls a subprogram, you can open the subprogram, adapt it to the program, and return to the program.

To edit a referenced object

1 Place the cursor within the name of an object referenced in the source code.

2 From the **Program** menu, choose **Open**.

Or:

Choose the following toolbar button:

Or:

Press CTRL+O.

The source code of the referenced object is displayed in edit mode in the editor window of the appropriate editor (for example, the map editor for an object of type map).

To list a referenced object

1 Click or double-click on the object.

2 From the **Program** menu, choose **List**.

Or:

Choose the following toolbar button:

The source code of the referenced object is displayed in read-only mode.

12 Setting Display Modes

- Formatting and Coloring Source Code 54
- Showing and Hiding Source Code 55
- Splitting the Editor Window 57

You can set preferences for the display of source code in the editor window.

Formatting and Coloring Source Code

You can format source code by indenting source-code lines. Indentation is performed differently for sources created in reporting mode than for sources created in structured mode. In addition, you can color syntax elements for better readability. Syntax coloring is set on by default. However, you can change the default color definitions or switch syntax coloring off.

To format source code

1 Place the cursor anywhere in the source code.

2 From the **Edit** menu choose **Format Source**.

Or:

Choose the following toolbar button:

Or:

In the command line, enter the following system command:

```
STRUCT
```

See also `STRUCT` in the *System Commands* documentation.

The lines in the source code are indented.

To change syntax coloring

- Change the default color settings by using the **Colors** editor option, or switch **Syntax coloring** off as described in *Program Editor Options* in the *Using Natural Studio* documentation.

Showing and Hiding Source Code

You can show (expand) or hide (collapse) logical blocks of source code to improve readability and maintainability of objects with complex code structures. When a block of source code is collapsed, all of the lines of source code between the block begin statement and block end statement are hidden, including any other nested blocks if they are part of the chosen block. Hidden blocks retain their collapsed or expanded status.

Expanding and collapsing code blocks only applies to object sources that were created in structured mode.

Code blocks that can be expanded or collapsed are, for example, `DEFINE DATA` blocks, `REPEAT` blocks, `IF THEN ELSE` blocks, `READ` blocks, and blocks of two or more consecutive comment lines.

When scanning source code, collapsed blocks can be excluded from the search. For more information, see ***Finding Text***.

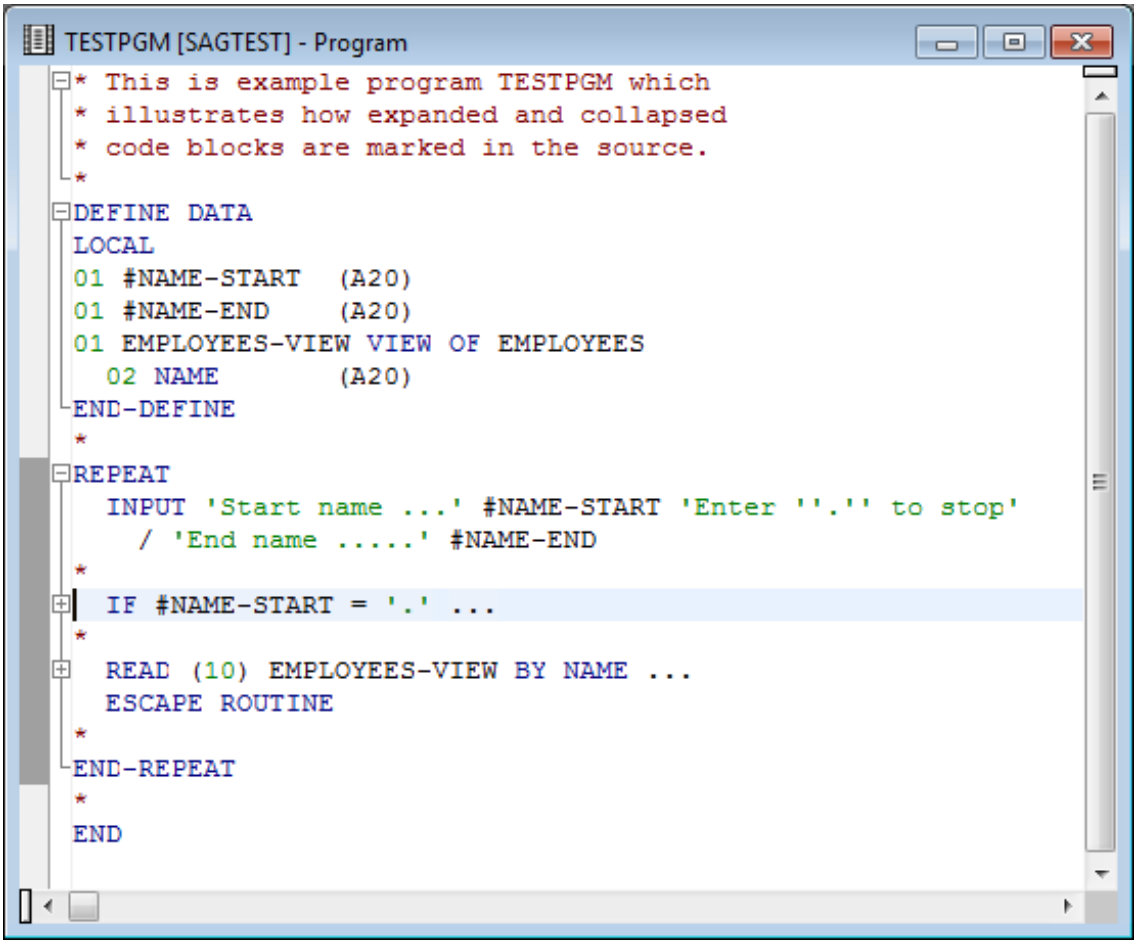
When you are searching for a line number in source code that is part of a collapsed block, the block is expanded to display the line. For more information, see ***Going to a Line Number***.

If you want to expand and collapse code blocks, you need to set the appropriate editor option referenced in the instructions below.

To expand and collapse single code blocks

1 Set the **Expand/Collapse** editor option as described in *Program Editor Options* in the *Using Natural Studio* documentation.

When the **Expand/Collapse** option is set, an expand or a collapse toggle (⊞ or ⊟) appears as shown in the example below:



The ⊟ toggle indicates the first line of an expanded block.

Each block is marked with a vertical line that goes from the begin statement (here, for example: `DEFINE DATA`) to the end statement (here: `END-DEFINE`).

The gray vertical bar in the column next to the source-line numbers indicates the current block (here: `REPEAT`) which depends on the logical relationship of the statement where the cursor is positioned. In the example above, the cursor is positioned in the `INPUT` statement line which belongs to the `REPEAT` loop. (`IF` and the `READ` are separate nested loops.)

The ⊞ toggle indicates the first line of a collapsed block.

2 Click on the ⊞ toggle to expand the block or click on the ⊟ toggle to collapse the block.

Or:

To expand a block and insert a line:

Place the cursor in the first position of a ⊞ toggle line and press SHIFT+ENTER to expand the block and insert a line *above* the expanded block.

Place the cursor in the last position of a ⊞ toggle line and press SHIFT+ENTER to expand the block and insert a line *below* the expanded block.

If you place the cursor in the first or last position of a ⊞ toggle line and choose ENTER, a line will be inserted above or below the collapsed block but the block will *not* expand.

Or:

Position the cursor within a line that contains the toggle ⊞ or ⊟ and, from the **View** menu, choose **Expand/Collapse** or choose the following toolbar button:

To expand or collapse all code blocks

- From the **View** menu, choose **Expand All** or **Collapse All**.

  Or:

  Choose either of the following toolbar buttons:

  (**Expand All**) or (**Collapse All**).

Splitting the Editor Window

You can split the program editor window vertically or horizontally to view and modify two different parts of the object simultaneously. This feature saves you the trouble of printing an object source to view two different sections simultaneously. Changes made in one section are made simultaneously in the other section.

You can jump between split windows.

To split the editor window into two sections

1 From the **View** menu, choose **Vertical Split** or **Horizontal Split**.

  Or:

  Choose either of the following toolbar buttons:

  (**Split Vertical**) or (**Split Horizontal**)

  The mouse pointer changes to a split pointer.

2 Drag the line to the left/right or up/down to the position you want by either moving the mouse or using the arrow keys.

3 Freeze panes by either clicking the left mouse button or pressing the ENTER key.

The screen is split into two sections, each displaying the same information. You can now scroll each section individually and edit both sections as if they were part of the same object (as indeed they are).

To exit split-editor mode and return to a single screen

- From the **View** menu, choose **Unsplit**.

  Or:

  Choose the following toolbar button:

  Two editor sections are transformed into one.

To jump between split screens

- Place the mouse pointer in the required screen section and click.

  Or:

  Press F6.

  The cursor moves from one screen section to the other.

13 Saving Source Code

You can save source code as either a source object only or as a source object and, additionally, a cataloged object (generated program) in the current Natural library in the current Natural system file.

For the naming conventions that apply to an object, refer to *Object Naming Conventions* in the *Using Natural Studio* documentation.

To save source code as a source object

- Proceed as described in *Saving Objects* in the *Using Natural Studio* documentation.

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

- Proceed as described in *Stowing Objects* in the *Using Natural Studio* documentation.

14 Using Context-Sensitive Help

The program editor provides context-sensitive help for the following source-code elements:

- Statements (for example, `WRITE`)
- System variables (for example, `*USER`)
- System functions (for example, `COUNT`)
- Parameters (for example, `AD`)

The help information includes syntax explanations and programming advice.

To invoke the help function for source-code elements

- Position the cursor on the keyword within a syntax element for which you require help and press F1.

  The **Help for Natural** window appears which contains the help information about the specified keyword.

  If the specified keyword refers to several help topics, a dialog box appears where you can select a topic from a list.

  If the specified keyword forms part of a complete statement (for example, the keyword `TOP` in the statement `AT TOP OF PAGE`), the help function tries to identify the statement that contains the keyword:

  - If the result is unambiguous (that is, a single complete statement is found), help information is displayed for that particular statement.
  - If a keyword is found which is used in several statements (for example, the keyword `IF`), a dialog box appears where you can select a statement from a list of all statements that begin with or contain the specified keyword.

- If no complete statement can be found (for example, if you have only typed in the word `DEFINE` and pressed F1 before typing in the full statement) or if the keyword is used in several statements, a dialog box appears where you can select a statement from a list of all statements that begin with or contain the specified keyword.

III Data Area Editor

15 Data Area Editor

- Invoking the Data Area Editor 66
- Rows and Columns in the Editor Window 68
- Editing Data Areas 73
- Rearranging Columns 98
- Showing or Hiding Fields 101
- Navigating between Field Levels 103
- Saving and Cataloging Data Areas 103
- Generating Copycode from Data Areas 104

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 from a data definition module (DDM), which are used by one or more Natural objects. You can also generate Natural objects of the type copycode from a data area. Note that data views from a DDM cannot be defined in PDAs.

Note: 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.

A data element in a data area is referred to as a field.

Related Topics:

- ***Data Area Editor: Source Format*** in *Editor Features With SPoD*
- *Editors in the SPoD Environment* in the *Unicode and Code Page Support* documentation
- *Setting the Options* and *Data Area Editor Options* in the *Using Natural Studio* documentation
- *Toolbars* in the *Using Natural Studio* documentation
- *Shortcut Keys* and *Data Area Editor Shortcut Keys* in the *Using Natural Studio* documentation

Invoking the Data Area Editor

You can invoke the data area editor by either using a menu function or the system command `EDIT`.

To invoke the editor for a new data area

- From the **Object** menu, select one of the following: **New** > **Local Data Area**, or **New** > **Global Data Area**, or **New** > **Parameter Data Area**.

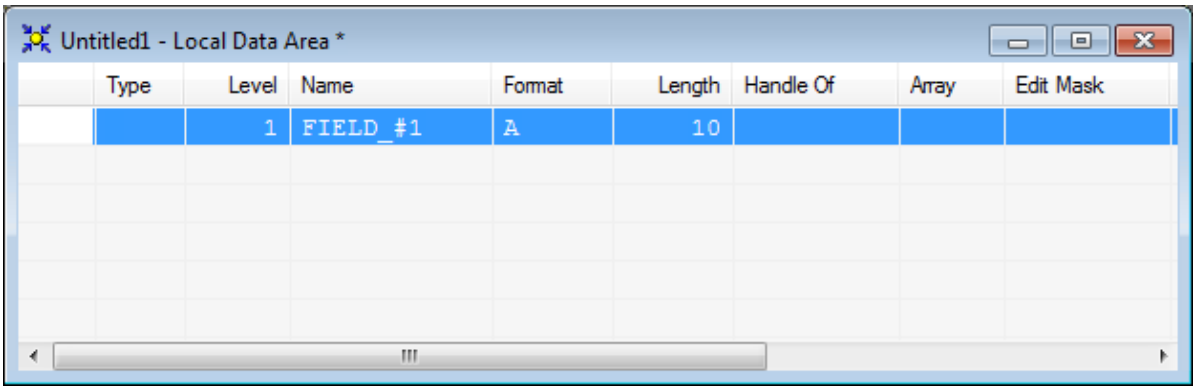
  Or:

  In the command line, enter the following:

```
EDIT object-type
```

where *object-type* is the one-letter code that denotes the type of data area: `L` for local data area, `G` for global data area, or `A` for parameter data area.

An editor window similar to the example below appears:



The title bar displays **Untitled** and the type of data area: local data area, global data area or parameter data area (in the example above, **Local Data Area**). The first row is preset to default values in the cells **Level**, **Name**, **Format** and **Length**.

In addition, a status bar can appear below the title bar depending on whether the corresponding editor option is set: see *Data Area Editor Options* in *Using Natural Studio*.

To invoke the editor for an existing data area

- In the Logical View, expand a **Local Data Areas** subnode to display the data areas available.

  Select a data area and choose **Open** from the context menu.

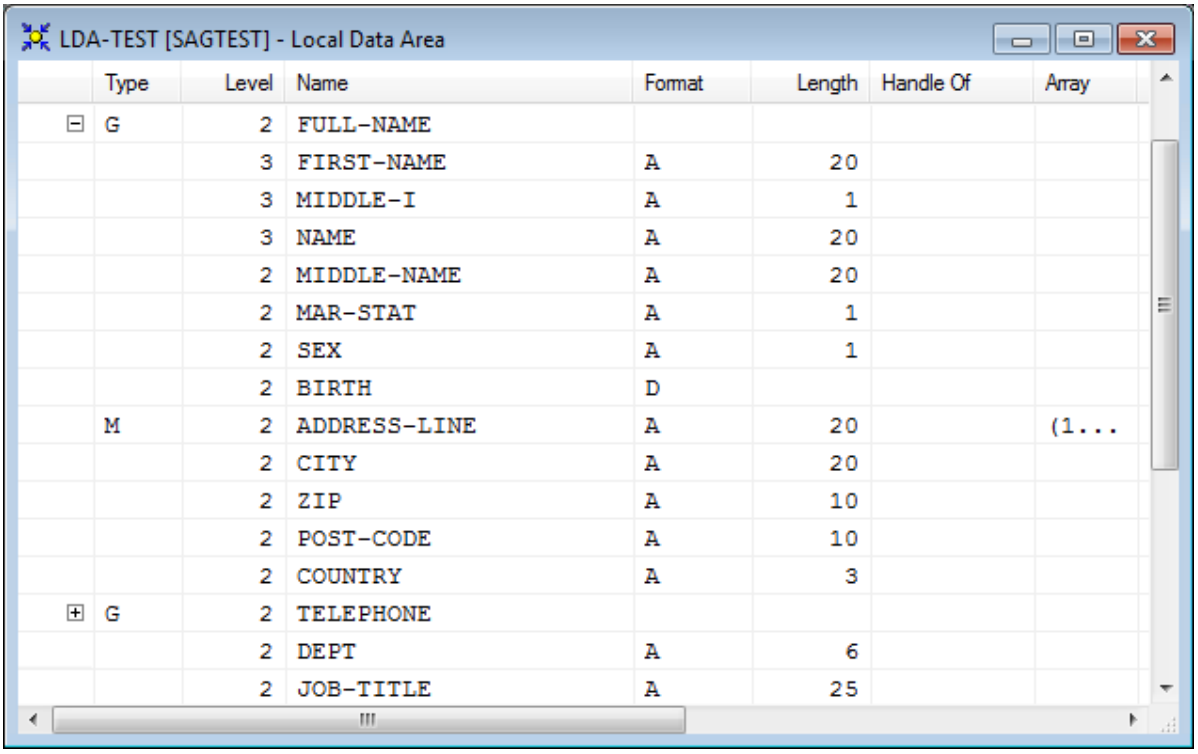
  Or:

  In the command line, enter the following:

  ```
  EDIT object-name
  ```

  where *object-name* is the name of the data area you want to edit.

  An editor window similar to the example below appears:



	Type	Level	Name	Format	Length	Handle Of	Array
⊟	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			
	M	2	ADDRESS-LINE	A	20		(1...
		2	CITY	A	20		
		2	ZIP	A	10		
		2	POST-CODE	A	10		
		2	COUNTRY	A	3		
⊞	G	2	TELEPHONE				
		2	DEPT	A	6		
		2	JOB-TITLE	A	25		

The fields contained in the specified data area are read into the editing area of the editor window. The title bar displays the name of the data area (in the example above, **LDA-TEST**) and the type of data area: local data area, global data area or parameter data area (in the example above, **Local Data Area**).

For further information on `EDIT`, see the relevant section in the *System Commands* documentation.

Rows and Columns in the Editor Window

The editing area of the editor window is organized in a table where the field definition data is contained in rows and columns. The editor provides a separate row for each field defined for a data area. All attribute definitions that belong to a field are contained in the cells of this row.

The columns and corresponding headings contained in the editor window are described in the following section. The display of the columns depends on whether a column is relevant for the type of data area being edited. For example, the **Parent** column is only displayed for global data areas. For explanations of the field attributes mentioned in this section, see also *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 for the fields contained in a data area.

You can resize, move and hide columns as described in *Rearranging Columns*.

Column Heading	Explanation
None	The indicator column is displayed in the leftmost section of the editor window. It can contain the following signs which appear next to the appropriate row:
	An error sign which indicates incorrect syntax. You are then required to enter a valid value. A tool tip provides error information.
	An information sign which warns you of potential problems caused by the value entered. A tool tip helps evaluate and eliminate the problem.
	⊟ or ⊞ A toggle key which indicates an expanded or a collapsed block of fields (see also ***Showing or Hiding Fields***).
Type	The type of field. Possible types are:
	blank Elementary field. This type of field can hold data and does not contain any other nested fields.
	`B` Data block. A data block is a collection of variables. Data blocks only apply to global data areas. For details, see *Data Blocks* in the *Programming Guide* and *Defining Global Data* in the *Statements* documentation.
	`C` A variable that is defined as a named constant (see also `CONSTANT` in *Variable Definition* in the *Statements* documentation) 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 from an Adabas database. See also *Referencing the Internal Count for a Database Array C* Notation)* in the *Programming Guide.*
	`G` Group. A group is a number of fields defined under one common group name within a view. 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.
	`H` Handle of dialog element. A handle identifies a dialog element and is stored in a handle variable. See also `DEFINE DATA` in the *Statements* documentation and *Defining Object Handles* in the *Programming Guide*.

Column Heading	Explanation	
	M	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*.
	O	Handle of object.
	P	Periodic group. A group of fields that can have more than one value within a record. See also *Periodic Groups* in the *Programming Guide*.
	R	Redefinition. For information on redefining fields, see *Redefining Fields* in the *Statements* documentation.
	S	Data Structure. A structure is a number of fields defined under one common name. This allows you to reference several fields collectively by using the structure name instead of the names of all the individual fields. Such fields cannot hold any data, but are only containers for other fields. See also *Qualifying Data Structures* in the *Programming Guide*.
	U	Globally Unique Identifier (GUID). A GUID is a constant that is guaranteed to be unique worldwide in the COM/DCOM model. See also *Globally Unique Identifiers - GUIDs* in the *Programming Guide*.
	V	Not applicable to PDAs. View from a DDM. See also the *View Definition* in the *Statements* documentation.
	*	A commentary field. For instructions on commenting out fields by using **Type**, see *Specifying Comments*.
Level	The level of the field. Levels are used to indicate the structure and grouping of fields. This is relevant to fields of the type view, group, structure and redefinition. Valid level numbers are 1 - 99. Level numbers must be specified in consecutive ascending order. A level number can only be one level higher than the previous level.	

Column Heading	Explanation
	See also the following sections in the *Programming Guide*: *Level Numbers in View Definitions*, *Level Numbers in Redefinitions* and *Level Numbers in Group Fields*.
Name	The name of the field. This name corresponds to the field name used in another Natural object (for example, a program) that references this field. For valid names, see the naming conventions for *User-Defined Variables* and *User-Defined Constants* in the *Programming Guide*. Redefine Function: 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 a field or variable being redefined, where *n* can be up to 10 digits (1 GB). The definition of trailing filler bytes is optional. **Note:** In a remote mainframe environment, you can preserve mixed-case field names by setting the **Support mixed-case field names on mainframes** editor option described in *Data Area Editor Options* in the *Using Natural Studio* documentation.
Format	The Natural data format of an elementary field such as A (alphanumeric), P (packed numeric) or L (logical). For valid Natural data 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). To modify the format of a field, see also the explanations in ***Modifying Fields***.
Length	The length of the field. For valid Natural length specifications, see *Format and Length of User-Defined Variables* and *Special Formats* in the *Programming Guide*. Depending on the Natural data format selected from the **Format** drop-down list box, **Length** is preset to one of the following default values: 10 — for formats A, B and U 4 — for formats F and I 7 — for formats N and P No length is permitted for the Natural data formats C, D, L and T. For alphanumeric and binary fields, you can define dynamic variables by specifying DYNAMIC in the **Length** column or setting the **Dynamic** option in the dialog box.

Column Heading	Explanation
	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).
Handle Of	The type of handle such as a list box.
Array	The array indices. As an alternative to entering values in the table cell, you can define or modify an array in the **Array Definition** dialog box as described in *Defining Arrays*. For further information on how to define an array, see *Arrays* and *Database Arrays* in the *Programming Guide*, and *Array Dimension Definition* in the *Statements* documentation.
Edit Mask	Not applicable to parameter data areas. The edit mask to be used when the field is displayed with an I/O statement. The syntax that applies to specifying an edit mask corresponds to the syntax of the session parameter `EM` or `EMU` (for a Unicode edit mask) described in the *Parameter Reference* documentation.
Header	Not applicable to parameter data areas. The header to be produced for each field specified in a `DISPLAY` statement. The syntax that applies to specifying a header corresponds to the syntax of the session parameter `HD` described in the *Parameter Reference* documentation.
Init	Not applicable to parameter data areas. The initial value assigned to a field. For detailed instructions on how to assign initial values, see *Defining Initial Values*. For basic information on how to assign initial values, see the sections *Initial Values (and the RESET Statement)* and *Initial Values for Arrays* in the *Programming Guide*.
Comment	A comment which applies to the field. See also *Specifying Comments*.
Parent	Not applicable to parameter data areas. In a global data area: The name of the parent (master) block. If you use a parent block, it must be defined in the current data area. Otherwise, a syntax error occurs. In a local data area: The name of the DDM from which the field derives.
Properties	Only applies to parameter data areas.

Column Heading	Explanation
	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). Possible entries are: By Reference (default) By Reference Optional By Value By Value Optional By Value Result By Value Result Optional For detailed information, see the corresponding options BY VALUE, BY VALUE RESULT and OPTIONAL described for the DEFINE DATA statement in *Parameter Data Definition*, and *operand2* described for the CALLNAT statement in the *Statements* documentation.
Print Mode	Not applicable to parameter data areas. The print mode to be used for the field. You can select I (inverse print direction) or N (no hardcopy). For details, see the PM session parameter, which corresponds to this field. The print mode is not selected by default indicating that the standard character set is used for printing.

Editing Data Areas

You can add a new field to a data area, insert a field, or modify the attributes of a field by using one of the following methods:

- Enter each attribute definition into the respective cell of a field row or replace existing definitions.
- Enter or replace all attributes in a **Definition** dialog box by using the appropriate insert function.
- Copy fields within a data area, from another data area or from a different type of Natural object by using copy and paste functionality or the import function.

This section below covers the following topics:

- Selecting Fields or Field Attributes
- Inserting Fields

- Modifying Fields
- Copying, Cutting and Pasting Fields
- Inserting Data Fields
- Inserting Data Blocks
- Inserting Constants
- Inserting Handles
- Inserting Data Structures
- Inserting Globally Unique Identifiers
- Defining Arrays
- Defining Initial Values
- Defining Counter Fields
- Importing Fields
- Specifying Comments
- Finding and Replacing Text
- Redefining Fields
- Deleting Fields

Selecting Fields or Field Attributes

Before you perform an editor function, you select (highlight) the row or row cell where you want to create, modify or delete a field.

- Selecting Single Fields
- Selecting Multiple Fields
- Selecting Field Attributes

Selecting Single Fields

To select a single field if a cell is selected

- Press SHIFT+SPACEBAR.

  The field row of the cell is selected.

  Or:

  Click on the leftmost column of the field row you want to select.

  The specified field row is selected.

To select a single field if a row is selected

- Click on the field row you want to select.

  Or:

Navigate to the field row you want to select by pressing UP-ARROW, DOWN-ARROW, HOME or END.

The specified field row is selected.

Selecting Multiple Fields

To select a range of fields if a cell is selected

1 Press SHIFT+SPACEBAR.

The field row of the cell is selected.

Or:

Click on the leftmost column of the first field row in the range.

The specified field row is selected.

2 Hold down SHIFT while you select the row of the last field in the range.

The rows of the specified field range are selected.

To select a range of fields if a row is selected

1 Click on the leftmost column of the first field row in the range.

Or:

Navigate to the first field row in the range by pressing UP-ARROW, DOWN-ARROW, HOME or END.

The first field row in the range is selected.

2 Hold down SHIFT while you select the row of the last field in the range.

The rows of the specified field range are selected.

To select all fields

■ From the **Edit** menu, choose **Select All**.

Or:

Choose the **Select All** toolbar button.

Or:

Press CTRL+A.

All field rows contained in the current DDM source are selected.

Selecting Field Attributes

To select a field attribute if a cell is selected

- Click on the row cell where you want to add or modify an attribute.

  Or:

  Navigate to the row cell where you want to add or modify an attribute by pressing TAB, SHIFT+TAB, UP-ARROW, DOWN-ARROW, LEFT-ARROW, RIGHT-ARROW, HOME or END.

  The specified row cell is selected.

To select a field attribute if a row is selected

- Press F2.

  The leftmost cell of the field row is selected.

  Or:

  First, click on the row that contains the cell you want to select and then click on the cell where you want to add or modify an attribute.

  The specified row cell is selected.

Inserting Fields

This section provides instructions for inserting fields into a data area.

Note that you cannot insert a field within a view definition.

To insert a field

1. Select a single row (multiple rows are not allowed) where you want to place the new field.

   The insert position (before or after the selected field) depends on the current setting of:

   - the **Insert After On/Off** toolbar button.
   - the **Insert before/Insert after** editor option which is described in *Data Area Editor Options* in the *Using Natural Studio* documentation.

2 Invoke the **Definition** dialog box by using the insert function that corresponds to the type of field you want to define. The type of field is indicated in the label of the dialog box (for example, **Periodic Group Definition**).

For explanations of the values to be entered in the **Definition** dialog box, see *Rows and Columns in the Editor Window*.

Or:

Copy single or multiple fields into the data area by using copy and paste functionality (see *Copying, Cutting and Pasting Fields*) or the import function (see *Importing Fields*).

Or:

Select a row and press INS, or select a cell in the *last* row and press DOWN-ARROW.

A data field with default values for name, data format, length and level is then added.

When you insert a field of the type redefinition, group, periodic group or structure, the level of each subsequent field is automatically incremented properly.

Modifying Fields

This section provides instructions for modifying single fields (ranges of fields are not allowed) within a data area. You can only modify single fields

Caution: When changing the field type, all field attribute definitions may be reset to their default values. This happens, for example, when you convert a data field to a data structure. You can keep original attribute definitions by commenting out a field as described in *Specifying Comments*.

To modify a field

- Select the row cell that contains the field attribute definition you want to change and either overwrite the existing value or choose a value from a selection box.

  Or:

  Open the **Definition** dialog box choosing one of the following methods:

  - Double-click on the row that contains the field attribute definition you want to change.
  - Select the field row that contains the attribute you want to change and choose the **Modify** toolbar button.
  - Select the field row that contains the attribute you want to change and choose **Modify** from the **Field** or context menu.

In the **Definition** dialog box, edit the text boxes and/or select values from the drop-down list boxes as described for the insert function that corresponds to the type of field you want to modify.

For explanations of the values to be entered in a row cell or in the **Definition** dialog box, see *Rows and Columns in the Editor Window*.

When you modify the level of a field of the type redefinition, group, periodic group or structure, the level of each subsequent field is automatically incremented or decremented properly, depending on the new level value.

When you modify the Natural data format of a field, the current length is kept if it is also valid for the new data format. Otherwise, the current length specification is automatically replaced by a valid default length (see also the description of the **Length** column).

When you modify the length of a field that belongs to a redefinition, consider the following: If the total length of all fields that belong to the redefinition exceeds the length of the redefined field, the information sign or an appropriate warning message appears.

Copying, Cutting and Pasting Fields

The copy/cut and paste functions of the data area editor are used to copy, move or delete fields within a single data area or between multiple data areas. In addition, you can copy field definitions from a data area into a Natural object that is handled by the program editor (for example, a program). If you want to copy field definitions from another Natural object such as a map and a DDM, use the **Import** function described in *Importing Fields*.

To copy or cut and paste fields

1 In a data area, select the field(s) you want to copy.

2 From the **Edit** or context menu, choose **Copy** or **Cut**.

Or:

Choose the **Copy** or **Cut** toolbar button.

Or:

Press CTRL+C (to copy) or CTRL+X (to cut).

The definitions of the selected fields are placed on the clipboard and can now be pasted into the data area contained in the active editor window.

3 If you want to paste the fields into another data area, open the appropriate object.

4 Select the field before or after which (see also **insert position**) you want to paste the fields.

5 From the **Edit** or context menu, choose **Paste**.

Or:

Choose the **Paste** toolbar button.

Or:

Press CTRL+V.

The copied or cut fields are pasted at the specified position in the data area contained in the active editor window.

6 To paste the same fields again, repeat Steps 3 through 5.

When you cut or paste a field of the type redefinition, group, periodic group or structure, the level of each subsequent field is automatically adjusted properly.

Inserting Data Fields

This function is used to insert elementary fields that contain scalable definitions.

For explanations of the values to be entered in the dialog box described in the following instructions, see *Rows and Columns in the Editor Window*.

To insert a data field

1 Select the row where you want to insert the field (see also **insert position**).

2 Select a row and press INS, or select a cell in the *last* row and press DOWN-ARROW.

A data field with default values for name, data format, length and level is then added.

Or:

From the **Insert** menu, choose **Data Field** or press SHIFT+D, or choose the **Insert Data Field** toolbar button.

The **Data Field Definition** dialog box appears.

3 In the **Data Field Definition** dialog box, specify the following:

In the **Level** text box, enter a valid level number.

In the **Name** text box, enter a valid field name.

From the **Format** drop-down list box, select the required Natural data format.

Select the **Dynamic** check box if you want the field length to be set dynamically. In this case, the length text box will be deactivated.

In the **Length** text box, enter the field length.

In the **Edit mask** text box, specify an edit mask if you want to use one. This definition does not apply to parameter data areas.

In the **Header** text box, enter a header if you want to specify one. This definition does not apply to parameter data areas.

In the **Comment** text box, enter a commentary text if you want to document the field: see *Specifying Comments*.

For parameter data areas:

From the **Value clause** drop-down list box, select any of the following input/output characteristics for the field: `By Reference` (this is the default setting), `By Value` or `By Value Result`.

Select the **Optional parameter** check box if you want to specify the data field as `Optional`.

For further information, see **Properties** in *Rows and Columns in the Editor Window*.

From the **Print Mode** text box, select the required print mode. This definition does not apply to parameter data areas.

Choose **Array Definition** if you want to invoke the **Array Definition** dialog box where you can define an array: see *Defining Arrays*.

Choose **Initialize** if you want to invoke the **Field Initialization** dialog box where you can define an initial value for the field: see *Defining Initial Values*. This definition does not apply to parameter data areas.

4 Choose **Add**.

The field is inserted into the specified position of the data area. The **Data Field Definition** dialog box is cleared and remains open.

5 Choose either of the following options:

Repeat Steps 3 and 4 if you want to define additional fields and insert them into the data area.

Or:

Choose **Quit** when you are finished.

The **Data Field Definition** dialog box is closed.

Inserting Data Blocks

This function only applies to global data areas.

For explanations of the values to be entered in the dialog box described in the following instructions, see *Rows and Columns in the Editor Window*.

To insert a data block

1 Select the row where you want to insert the data block (see also **insert position**).

2 From the **Insert** menu, choose **Block** or press SHIFT+B.

Or:

Choose the **Insert Block** toolbar button.

The **Block Definition** dialog box appears.

3 In the **Block Definition** dialog box, specify the following:

In the **Name** text box, enter a valid name for the data block.

In the **Parent** text box, enter the name of the parent (master) block. If you use a parent block, it must be defined in the current data area. Otherwise, a syntax error occurs.

In the **Comment** text box, you can enter a comment that documents the data block: see *Specifying Comments*.

4 Choose **OK**.

The parent block is inserted into the specified position of the data area where the **Type** column indicates B (for Block), and the **Data Field Definition** dialog box appears.

5 Define the subordinate block(s) that belong to the parent block as described in *Inserting Data Fields*.

Inserting Constants

This functions does not apply to parameter data areas.

For explanations of the values to be entered in the dialog box described in the following instructions, see *Rows and Columns in the Editor Window*.

To insert a constant

1 Select the row where you want to insert the field (see also **insert position**).

2 From the **Insert** menu, choose **Constant** or press SHIFT+C.

Or:

Choose the **Insert Constant** toolbar button.

The **Constant Definition** dialog box appears.

3 In the **Constant Definition** dialog box, specify the following:

In the **Name** text box, enter a valid field name.

From the **Format** drop-down list box, select the required Natural data format.

In the **Length** text box, enter a field length.

In the **Edit mask** text box, specify an edit mask if you want to use one. This definition does not apply to parameter data areas.

In the **Header** text box, enter a header if you want to specify one. This definition does not apply to parameter data areas.

In the **Comment** text box, enter a commentary text if you want to document the field: see *Specifying Comments*.

From the **Print Mode** text box, select the required print mode. This definition does not apply to parameter data areas.

Choose **Array Definition** if you want to define an array: see *Defining Arrays*.

Choose **Initialize** to invoke the **Field Initialization** dialog box where you can define an initial value for the field: see *Defining Initial Values*. This definition does not apply to parameter data areas.

4 Choose **Add**.

The field is inserted into the specified position of the data area where the **Type** column indicates `C` (for Constant) and the **Property** column indicates `I` (for Initialize). The **Constant Definition** dialog box is cleared and remains open.

5 Choose either of the following options:

Repeat Steps 3 and 4 if you want to define additional fields and insert them into the data area.

Or:

Choose **Quit** when you are finished.

The **Constant Definition** dialog box is closed.

Inserting Handles

For a handle, you can define the type dialog element or object.

For explanations of the values to be entered in the dialog box described in the following instructions, see *Rows and Columns in the Editor Window*.

To insert a handle

1 Select the row where you want to insert the field (see also **insert position**).

2 From the **Insert** menu, choose **Handle** or press SHIFT+H.

Or:

Choose the **Insert Handle** toolbar button.

The **Handle Definition** dialog box appears.

3 In the **Handle Definition** dialog box, specify the following:

In the **Level** text box, enter a valid level number.

In the **Name** text box, enter a valid field name.

In the **Type** field, choose either of the following options:

- Select the **Dialog Element** option button for a handle of the type dialog element.

  Then from the drop-down list box, select the required dialog element.
- Or:

  Select the **Object** option button for a handle of the type object.

  The drop-down list box then displays `OBJECT`.

In the **Comment** text box, enter a commentary text if you want to document the field: see *Specifying Comments*.

For parameter data areas:

From the **Value clause** drop-down list box, select any of the following input/output characteristics for the field: `By Reference` (this is the default setting), `By Value` or `By Value Result`.

Select the **Optional parameter** check box if you want to specify the data field as `Optional`.

For further information, see **Properties** in *Rows and Columns in the Editor Window*.

Choose **Array Definition** if you want to invoke the **Array Definition** dialog box where you can define an array: see *Defining Arrays*.

Choose **Initialize** if you want to invoke the **Field Initialization** dialog box where you can define an initial value for the field: see *Defining Initial Values*. This definition does not apply to parameter data areas.

4 Choose **Add**.

The field is inserted into the specified position of the data area where the **Type** column indicates H (for Handle). The **Data Field Definition** dialog box is cleared and remains open.

5 Choose either of the following options:

Repeat Steps 3 and 4 if you want to define additional fields and insert them into the data area.

Or:

Choose **Quit** when you are finished.

The **Data Field Definition** dialog box is closed.

Inserting Data Structures

A data structure consists of fields and nested structures.

For explanations of the values to be entered in the dialog box described in the following instructions, see *Rows and Columns in the Editor Window*.

To insert a data structure

1 Select the row where you want to insert the data structure (see also **insert position**).

2 From the **Insert** menu, choose **Structure** or press SHIFT+S.

Or:

Choose the **Insert Structure** toolbar button.

The **Structure Definition** dialog box appears.

3 In the **Structure Definition** dialog box, specify the following:

In the **Level** text box, enter a valid level number.

In the **Name** text box, enter a valid name for the structure.

In the **Comment** text box, enter a commentary text if you want to document the data structure: see *Specifying Comments*.

Choose **Array Definition** if you want to invoke the **Array Definition** dialog box where you can define an array: see *Defining Arrays*.

4 Choose **OK**.

The data structure is inserted into the specified position of the data area where the **Type** column indicates `S` (for Structure), and the **Data Field Definition** dialog box appears.

5 Define the subordinate field(s) that belong to the structure as described in *Inserting Data Fields*.

Inserting Globally Unique Identifiers

This function only applies to local data areas and global data areas.

For explanations of the values to be entered in the dialog box described in the following instructions, see *Rows and Columns in the Editor Window*.

To insert a Globally Unique Identifier

1 Select the row where you want to insert the field (see also **insert position**).

2 From the **Insert** menu, choose **Globally Unique ID** or press SHIFT+U.

Or:

Choose the **Insert GUID** toolbar button.

The **Globally Unique ID Definition** dialog box appears.

3 In the **Globally Unique ID Definition** dialog box, specify the following:

In the **Level** text box, enter a level number.

In the **Name** text box, enter a valid field name.

In the **Comment** text box, enter a commentary text if you want to document the field: see *Specifying Comments*.

4 Choose **Add**.

The field is inserted into the specified position of the data area as a Natural constant with length A36. The **Type** column for the field row indicates `U` (for Globally Unique Identifier) and the **Init** column displays the contents of the constant (for example, `CONST <'2AEB9D1A-EAC2-4E5E-8983-0AF0CCB12098'>`). The **Globally Unique ID Definition** dialog box is cleared and remains open.

5 Choose either of the following options:

Repeat Steps 3 and 4 if you want to define additional fields and insert them into the data area.

Or:

Choose **Quit** when you are finished.

The **Global Unique ID Definition** dialog box is closed.

Defining Arrays

The **Array Definition** dialog box can be used to define multi-dimensional tables for the field name and field type indicated in the box.

For detailed information on how to define an array, see *Arrays* and *Database Arrays* in the *Programming Guide*, and *Array Dimension Definition* in the *Statements* documentation.

To define an array in the Array Definition dialog box

1 From the **Definition** dialog box, choose **Array Definition**.

The **Array Definition** dialog box appears for the specified field name and type.

2 In the **Array Definition** dialog box, specify the following:

From the **Dimensions** drop-down list box, select the number of dimensions for the array: `1`, `2`, or `3`. To delete an array definition, select `0` (zero).

In the **Lower bound** text box, enter the lower bound for each dimension.

In the **Upper bound** text box, enter the upper bound for each dimension.

3 Choose **OK**.

The definitions are saved, the **Array Definition** dialog box is closed and the **Data Field Definition** dialog box appears.

Defining X-Arrays

An X-array (eXtensible array) can be defined by specifying an asterisk (`*`) for at least one bound of at least one dimension of the array. The asterisk (`*`) in the bound definition indicates that the corresponding bound is extensible. Only one bound - either upper or lower - may be extensible, but not both. If the lower bound is extensible, the **Upper bound** text box contains the upper bounds of the X-array.

For more information on defining an X-array, see *X-Arrays* in the *Programming Guide* and *Array Dimension Definition* in the *Statements* documentation.

Defining Initial Values

This definition does not apply to parameter data areas.

The **Field Initialization** dialog box is used to assign an initial value to a field. For further information on how to assign initial values, see the sections *Initial Values (and the RESET Statement)* and *Initial Values for Arrays* in the *Programming Guide*.

In the **Field Initialization** dialog box, you can enter the value(s) for a data field in two different ways: single-value mode or free-form mode.

In single-value mode, you enter the values in a structured way. Parentheses, apostrophes or value prefixes (for example, `H` for Hex, `D` for Date, or `T` for Time) are not required.

In free-form mode, you enter the values just as you would in a `DEFINE DATA` statement; see also *Initial-Value Definition* and *Initial/Constant Values for an Array* in the *Statements* documentation.

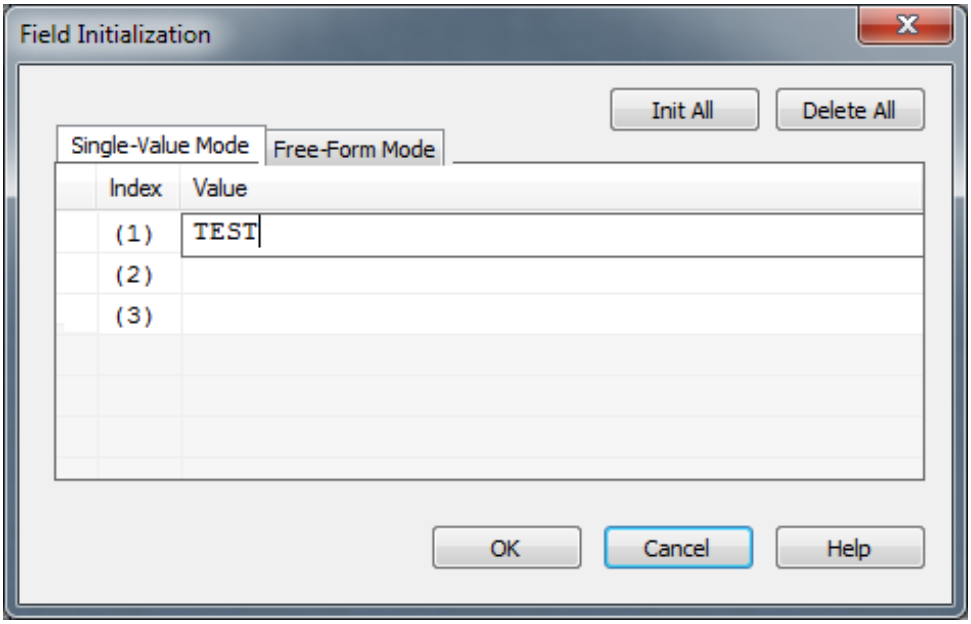
To define an initial value in single-value mode

1 From the **Definition** dialog box, choose **Initialize**.

Or:

Select the row cell of the **Init** column which contains the initial value required and choose the following button: This button is only available for a field that has been defined as an array.

The **Field Initialization** dialog box appears with the tabbed pages **Single-Value Mode** and **Free-Form Mode**. The **Single-Value Mode** page is opened by default. It contains a table similar to the example shown below:



The indicator column in the leftmost section contains error information if an incorrect value is entered in the **Value** column.

For a field that has been defined as an array, the **Index** column is displayed, which lists all occurrences of the array as shown in the example above for an array defined as `(1:2,1:3)`.

The **Value** column contains the initial value if assigned to the field (scalar) or array occurrence.

You can use TAB, CR or DOWN ARROW to go down one row in the table, and SHIFT+TAB or UP ARROW to go up one row. When you resize a column or select a row, press F2 to deselect the row and go to the **Value** column.

2 For a field that has been defined as a scalar, enter an initial value or replace the existing entry in the **Value** column, skip the following step and proceed with **Step 4**.

For a field that has been defined as an array, proceed with the following step.

3 In the **Value** column next to the **Index** row cell that contains the required array occurrence, add an initial value or replace the existing entry.

Or:

To assign an initial value to *all* occurrences, enter the required initial value for one of these occurrences, keep the cursor inside this row cell, and choose the **Init All** button. The initial value entered is then assigned to all occurrences.

Or:

To remove the initial values assigned to *all* occurrences, choose the **Delete All** button. The initial values entered are then removed from all occurrences.

The initial value is checked when you leave the row cell and continue editing another field definition, or when you choose **OK** in the **Field Initialization** dialog box.

4 Choose **OK**.

The definitions on the **Single-Value Mode** page are checked and saved, and the **Field Initialization** dialog box closes.

To define an initial value in free-form mode

1 From the **Definition** dialog box, choose **Initialize**.

Or:

Select the row cell of the **Init** column which contains the initial value required and choose the following button: . This button is only available for a field that been defined as an array, or for a value definition that spans multiple lines.

The **Field Initialization** dialog box appears with the tabbed pages **Single-Value Mode** and **Free-Form Mode**.

2 Open the **Free-Form Mode** page.

An edit box appears. If no initial value yet exists, the box is preset to a value such as `INIT (1) <...>`.

3 Enter the initial value definitions according to the common syntax definitions in a `DEFINE DATA` statement (see the *Statements* documentation).

4 Choose **OK**.

The definitions on the **Free-Form Mode** page are saved and the **Field Initialization** dialog box closes. The definitions are validated when you check or stow the data area with the appropriate menu function or system command.

Defining Counter Fields

For explanations of the values to be entered in the dialog box described in the following instructions, see *Rows and Columns in the Editor Window*.

To define a counter field

1 Select the multiple field or periodic field you want to define as a counter field (`C*` variable).

2 From the **Field** menu, choose **Counter** or press SHIFT+C.

Or:

Choose the **Counter Field** toolbar button.

A **'C*' Counter Definition** dialog box appears for the specified field.

3 In the **'C*' Counter Definition** dialog box, specify the following:

In the **Level** text box, change the field level if required.

From the **Format** drop-down list box, select the required format.

In the **Length** text box, enter a valid field length.

In the **Comment** text box, enter a commentary text if you want to document the counter field: see *Specifying Comments*.

4 Choose **OK**.

The **'C*' Counter Definition** dialog box is closed and the **Type** column now indicates `C` (for counter).

Importing Fields

You can import single or multiple fields into a data area from different Natural object types in any library or from a Predict server.

To import fields from another type of Natural object

1 Select the row where you want to insert the fields (see also **insert position**).

2 From the **Insert** menu, choose **Import** or press SHIFT+O.

Or:

Choose the **Import Data Field** toolbar button.

The **Import Data Field** dialog box appears. The name of the current library is displayed in the **Library** list box.

3 In the **Import Data Field** dialog box, specify the following:

From the **Library** list box, select another library if the object that contains the fields you want to import is located in a different library.

The list contains all libraries that reside in the current FNAT and FUSER system file, which are displayed as nodes in the Natural Studio tree view (the display can be limited by using the Display Filter function of Natural Studio). In addition, the list contains all libraries from inactive system files as specified in the steplib table.

From the **Type** group frame, select the option button that corresponds to the type of Natural object from which you want to import fields.

Or:

Depending on the type of Natural object required, use one of the following shortcut keys:

ALT+P for program	ALT+M for map
ALT+N for subprogram	ALT+L for local data area
ALT+7 for function	ALT+G for global data area
ALT+S for subroutine	ALT+A for parameter data area
ALT+H for helproutine	ALT+V for DDM

The **Object List** list box appears with a list of all Natural objects of the specified object type available in the selected library.

The list contains all objects, which are displayed in the library nodes of the Natural Studio tree view (the display can be limited by using the Display Filter function of Natural Studio).

For DDMs and data areas, the list only contains objects for which both a source object and a cataloged object exist.

From the **Object List** list box, select the object that contains the fields you want to import.

The fields contained in the selected object appear in the **Importable Data Fields** list box.

From the **Importable Data Fields** list box, select one or more fields that you want to import.

4 Choose **Import** or double-click on a field.

- For fields from a DDM, the **View Definition** dialog box appears where the name of the DDM is displayed in the **Name of DDM** text box.

  In the **Name of View** text box, enter the name to be used for the view in the data area.

  In the **Comment** text box, enter a commentary text if you want to document the view: see *Specifying Comments*.

  Choose **OK** when you are finished.
- If you import a multiple-value field from a DDM, the **Array Definition** dialog box appears.

  In the **Lower bound** and **Upper bound** text boxes change the values if required.

  The **Array Definition** dialog box does not appear if you import a multiple-value field from a view definition in a local or a global data area. The number of occurrences is then automatically copied from the selected field into the array definition.

  Choose **OK** when you are finished.
- If you import one or more fields that belong to a periodic group, the **Periodic Group Definition** dialog box appears.

Select either of the following option buttons: **PE group** or **Each field in PE group** (this is the default setting).

If you select **PE group**, the **Lower bound** and **Upper bound** text boxes appear where you can change the values if required.

The **Periodic Group Definition** dialog does not appear if you import fields of a periodic group from a view definition in a local or a global data area. The number of occurrences is then automatically copied from the selected fields into the array definition.

Choose **OK** when you are finished.

The fields are copied into the current data area and the **Import Data Field** dialog box remains open.

5 Repeat Steps 3 and 4 if you want to import additional fields.

6 Choose **Quit** when you are finished.

The **Import Data Field** dialog box is closed.

Specifying Comments

You can exclude fields from the syntax check by marking or unmarking the corresponding rows. In addition, you can insert commentary rows or add commentary text to an existing field. You can also add a comment and insert a commentary row with the **Redefine** function described in *Redefining Fields*.

To convert single or multiple fields to commentary rows

1 Select the required field row or the range of field rows.

2 Choose one of the following methods:

- From the **Edit** menu, choose either **Advanced** > **Add Comment Mark(s)** or **Advanced** > **Remove Comment Mark(s)**.
- Or:

  Press CTRL+M to comment out the field or CTRL+SHIFT+M to remove the comment mark.
- Or:

  From the drop-down list box in the **Type** cell of the required field row, select ***** to comment out the field, or select the appropriate field type value if you want to remove the comment mark, and choose ENTER.

A comment mark (*) in the **Type** cell of the selected field row indicates that the field has been converted to a commentary row. All field attribute definitions in the row cells are retained.

An empty **Type** cell or another field type value other than * indicates that the comment mark has been removed from the selected field row.

To convert a field to an empty commentary row

- From the drop-down list box in the **Type** cell of the field row to be marked, select **/*** and choose ENTER.

  A comment mark (*) appears in the **Type** cell of the field row and all row cells are cleared.

To insert a commentary row

1. Select the field row above or below which (see also **insert position**) you want to insert the commentary row.
2. From the **Insert** menu, choose **Comment** or press SHIFT+M.

   Or:

   Choose the **Insert Comment** toolbar button.

   The **Comment Line Definition** dialog box appears.
3. In the **Comment** text box, enter any text.
4. Choose **Add**.

   The commentary row is inserted into the specified position of the data area and the **Comment** text box is cleared and remains open.
5. Choose either of the following options:

   Repeat Steps 3 and 4 if you want to insert additional commentary rows into the data area.

   Or:

   Choose **Quit** when you are finished.

   The **Comment Line Definition** dialog box is closed.

To add a comment to a field

- In the **Comment** column of the field you want to document, enter any text.

  Or:

  Open the **Definition** dialog box and enter any text in the **Comment** text box.

Finding and Replacing Text

You can search for field names and comments contained in the current data area by using the find function. If it should be necessary to replace a frequently occurring text string, you can use the combined find and replace function.

The find function is performed on all data definitions including collapsed blocks of fields (see also *Showing or Hiding Fields*).

Caution: There is no undo function available to restore original names.

To search for a text string

1 From the **Edit** menu, choose **Find**.

Or:

Choose the **Find** toolbar button.

Or:

Press CTRL+F.

The **Find** dialog box appears.

2 In the **Find** text box, enter a search string.

Select the **Name** check box if you want to restrict the search to the field names contained in the **Name** column.

Select the **Comment** check box if you want to restrict the search to the commentary text contained in the **Comment** column.

Select the **Case sensitive** check box to search for strings that exactly match the entry in the **Find** text box. Otherwise, any combination of upper and lower-case letters will be found. This option only applies to commentary text contained in the **Name** or **Comment** column.

Select the **Whole words only** check box to restrict the search to whole words only. Otherwise, all occurrences of the search string will be found.

In the **Direction** section, select the option button **Up** or **Down** to specify whether the search is to be performed from the cursor position to the end of the data area or from the cursor position to the beginning of the data area. The default setting is **Down**.

3 Choose **Find Next**.

If no instance of the search string is found, an appropriate message is displayed.

If an instance of the search string is found, it will be selected.

4 To search for additional instances of the search string: from the **Edit** menu, choose **Find Next**.

Or:

Choose the **Find Next** toolbar button.

Or:

Press F3.

To replace a text string

1 From the **Edit** menu, choose **Replace**.

Or:

Choose the **Replace** toolbar button.

Or:

Press CTRL+H.

The **Replace** dialog box appears.

2 In the **Find** text box, enter a search string.

In the **Replace with** text box, enter a replacement string.

Select the **Name** check box if you want to restrict the search to the field names contained in the **Name** column.

Select the **Comment** check box if you want to restrict the search to the commentary text contained in the **Comment** column.

Select the **Case sensitive** check box to search for text strings that exactly match the entry in the **Find** text box. Otherwise, any combination of upper and lower-case letters will be found. This option only applies to commentary text contained in the **Name** or **Comment** column.

Select the **Whole words only** check box to restrict the search to whole words. Otherwise, all occurrences of the search string will be found.

In the **Direction** section, select the option button **Up** or **Down** to specify whether the search is to be performed from the cursor position to the end of the data area or from the cursor position to the beginning of the data area. The default setting is **Down**.

3 Choose **Replace** to replace the next hit found in the source.

Choose **Find Next** and **Replace** to find the next hit and replace it.

Or:

Choose the **Find Next** and **Replace** toolbar buttons.

Or:

Choose **Replace Next** to replace the next hit found without selecting the hit first.

Or:

Choose **Replace All** to replace all search strings found.

If no instance of the search string is found, an appropriate message is displayed.

4 Choose **Close** to exit the dialog box.

Redefining Fields

This function does not apply to a field in a view defined in a parameter data area.

When redefining a field, you can convert the Natural data format of a field or divide a single field into data segments. For details, see the redefinition option of the `DEFINE DATA` statement in the *Statements* documentation.

To redefine a field definition from one type to another

1 Select the field you want to redefine.

2 From the **Field** menu or context menu, choose **Redefine** or press SHIFT+E.

Or:

Choose the **Redefine** toolbar button.

A new row is inserted into the data area which contains the same name and level as the selected field and a `BEGIN REDEFINE` comment. In addition, the **Insert Redefine** dialog box appears.

3 In the **Insert Redefine** dialog box, select any of the following option buttons:

Structure to define a structure.

Or:

Data field to define an elementary field.

Or:

Comment to add a commentary line that documents the redefinition (see also ***Specifying Comments***).

4 Choose **OK**.

Depending on the option set, a corresponding dialog box appears for the specified field: **Structure Definition, Redefine** or **Comment Line Definition**.

5 Enter the required values as described in *Inserting Data Fields*, *Inserting Data Structures* or *Specifying Comments* respectively.

Note: In the **Name** text box of the **Redefine** dialog box, you can enter *nX* to specify filler bytes.

6 Choose **Add**.

The field is inserted into the data area and the **Insert Redefine** dialog box appears again.

7 Repeat Steps 3 through 6 until no more space is available or until the redefinition is complete.

If the total length of all fields that belong to the redefinition exceeds the length of the redefined field, the information sign or an appropriate warning message appears.

Deleting Fields

This section provides instructions for deleting fields in a data area.

To delete fields in the data area editor

1 Select the field(s) to be deleted.

2 From the **Edit** or context menu, choose **Delete**.

Or:

Press DEL.

Or:

Choose the **Cut** toolbar button.

Or:

Press CTRL+X.

If delete messages are active, you are requested to confirm the deletion. Otherwise, the fields are deleted without prior warning.

When you delete a field of the type redefinition, group, periodic group or structure, the level of each subsequent field is automatically decremented properly.

Rearranging Columns

In the editor window, you can adjust the display of the data area to your needs by resizing, moving or hiding columns that are not required for an editing operation in the current data area.

- Resizing Columns
- Moving Columns
- Hiding or Displaying Columns

Resizing Columns

You can automatically adjust a single column or all columns to the best size, or change the width of a single column to a specific size.

To resize all columns to best fit

- Choose one of the following methods:
  - Select a field as described in *Selecting Single Fields*.
  - From the **View** menu, choose **Customize Columns**.

    Or:
    In any column heading, click the right mouse button and choose **Customize Columns** from the context menu.

    The **Customize Columns** dialog box appears.
  - Select the **Best Fit** check box. This option is not selected by default.

  All columns in the active editor window are automatically resized to the size that best fits into the editor window whereby the column headings always remain visible.

  Or:

  Press CTRL+PLUS.

  Or:

  If you want to apply **Best Fit** to all active editor windows, set the corresponding editor option described in *Data Area Editor Options* in the *Using Natural Studio* documentation.

To resize all columns to best fit while typing in text

1. Open the **Customize Columns** dialog box as described in *To resize all columns to best fit*.
2. Select the **Best Fit** check box and, additionally, select the **Auto Fit** check box.

Each column in the active editor window is then automatically adjusted to fit the text you type in a row cell or a **Definition** dialog box when you leave the column or dialog box respectively.

Or:

If you want to apply **Best Fit** and **Auto Fit** to all active editor windows, set the corresponding editor options described in *Data Area Editor Options* in the *Using Natural Studio* documentation.

To resize a single column to fit the contents

- In the heading of the column you want to change, place the pointer over the right border. When the pointer changes to a divider, double-click on the border between the column headings. Note that you cannot resize the leftmost column.

  The column is automatically adjusted to fit its contents.

To resize a single column to a specific size

- In the heading of the column you want to change, place the pointer over the right border. When the pointer changes to a divider, drag the divider to the width you require. Note that you cannot resize the leftmost column.

  The width of the column has changed to the specified size.

To save a resized table layout

- Open the **Customize Columns** dialog box as described in *To resize all columns to best fit* and choose one of the following buttons.

  - **OK** saves the new table layout for the current editor session.
  - **Save Layout** saves the new layout in your user profile and retains it for future editor sessions.
  - **Restore Layout** overwrites the current layout with the layout previously saved in the user profile. Choose **OK** to save this layout.
  - **Restore Defaults** followed by **OK** overwrites the layout saved in the user profile with the default layout initially provided by the editor. Choose **OK** to save this layout.

  Or:

  In the editing area of the editor window, press CTRL+ALT+L.

  The new layout is saved in your user profile and retained for future editor sessions.

Moving Columns

You can change the table layout by moving single or multiple columns.

To move a column

1 Choose one of the following methods:

- Select a field as described in *Selecting Single Fields*.
- Open the **Customize Columns** dialog box as described in *To resize all columns to best fit*.
- From the **Displayed Columns** list box, select the columns you want to move and choose **Move Up** or **Move Down** (if required repeatedly) until the columns have reached the target position.

  The top-to-bottom order of the list box corresponds to the left-to-right of the table in the editor window, that is, the top list column corresponds to the leftmost table column.

Or:

- Drag the column heading you want to move and drop it in the position required. Note that you cannot move the leftmost column.
- Open the **Customize Columns** dialog box as described in *To resize all columns to best fit*.

2 To keep the new table layout, proceed as described in *To save a resized table layout*.

Hiding or Displaying Columns

You can change the table layout by hiding or displaying columns.

To hide a column by rearranging the display order

1 Select a field as described in *Selecting Single Fields*.

2 Open the **Customize Columns** dialog box as described in *To resize all columns to best fit*.

3 From the **Displayed Columns** list box, select the columns you want to hide.

The top-to-bottom order of the list box corresponds to the left-to-right of the table in the editor window, that is, the top list column corresponds to the leftmost table column.

Note: You cannot select **Type**, **Level**, **Name**, **Format** and **Length** which are mandatory for the table layout.

4 Choose **Remove**.

The selected columns are removed from **Displayed Columns** and appear in the **Hidden Columns** list box.

5 To keep the new table layout, proceed as described in *To save a resized table layout*.

To hide a column by moving column borders

1 In the heading of the column you want to hide, place the pointer over the right border. When the pointer changes to a divider, drag the divider to the left border until the column heading is completely invisible (right and left border lines must coincide).

Note: You cannot hide the columns **Type**, **Level**, **Name**, **Format** and **Length** which are mandatory for the table layout.

The hidden column then appears in the **Hidden Columns** list box of the **Customize Columns** dialog box.

2 To keep the new table layout, proceed as described in *To save a resized table layout*.

To display a hidden column

1 Select a field as described in *Selecting Single Fields*.

2 Open the **Customize Columns** dialog box as described in *To resize all columns to best fit*.

3 From the **Hidden Columns** list box, select the columns you want to display in the editor window.

4 Choose **Add**.

The selected columns are removed from **Hidden Columns** and appear in the **Displayed Columns** list box.

5 To keep the new table layout, proceed as described in *To save a resized table layout*.

Showing or Hiding Fields

You can show (expand) or hide (collapse) blocks of fields to improve readability and maintainability of data areas with complex data structures. When a block of fields is collapsed, all fields contained in this block are hidden, including any other nested blocks if they are part of the chosen block. Hidden blocks retain their collapsed or expanded state.

Blocks that can be expanded or collapsed are blocks of fields that are defined for the same field level (1 to 99). Blocks are expanded or collapsed by the hierarchy of levels, from highest level 1 to lowest level 99. A block that contains fields from a lower-ranking level is contained in a block from a higher level.

When scanning fields (see also *Finding and Replacing Field Names*), collapsed blocks are also scanned.

If you want to expand and collapse blocks of fields, you need to set the appropriate editor option referenced in the instructions below.

To expand and collapse single blocks

1 Set the **Expand/Collapse** editor option as described in *Data Area Editor Options* in the *Using Natural Studio* documentation.

When the **Expand/Collapse** option is set, an expand/collapse toggle (⊟ or ⊞) appears as shown in the example of an **editor window** shown earlier.

The toggle ⊟ indicates the first row of an expanded block.

The toggle ⊞ indicates the first row of a collapsed block.

2 Click on the toggle ⊞ to expand a block or click on the toggle ⊟ to collapse a block.

Or:

Position the cursor in a row that contains the toggle ⊞ or ⊟ and, from the **View** menu, choose **Expand/Collapse** or choose the **Expand/Collapse** toolbar button.

Or:

Use any of the shortcut keys listed in *Data Area Editor Shortcut Keys* in the *Using Natural Studio* documentation.

To expand or collapse all blocks

- From the **View** menu, choose **Expand All** or **Collapse All**.

Or:

Choose the **Expand All** or the **Collapse All** toolbar button.

Navigating between Field Levels

You can navigate through the level hierarchy of fields contained in a data area.

To navigate to the next lower field level

- From the **View** menu, choose **Next Level**.

  Or:

  Press CTRL+SHIFT+I.

  Or:

  Choose the **Next Level** toolbar button.

  The first field with a lower level is selected.

To navigate to the next higher field level

- From the **View** menu, choose **Previous Level**.

  Or:

  Press CTRL+SHIFT+J.

  Or:

  Choose the **Previous Level** toolbar button.

  The first field with a higher level is selected.

Saving and Cataloging Data Areas

You can save source code of a data area as a source object and/or a cataloged object (generated program) in the current Natural library in the current Natural system file.

For the naming conventions that apply to an object, refer to *Object Naming Conventions* in the *Using Natural Studio* documentation.

To save source code as a source object

- Proceed as described in *Saving Objects* in the *Using Natural Studio* documentation.

To save source code as a source object and/or a cataloged object

- Proceed as described in either *Stowing Objects* or *Cataloging Objects* in the *Using Natural Studio* documentation.

Generating Copycode from Data Areas

This function generates a Natural object of the type copycode from a data area. The `DEFINE DATA` statement in the copycode then contains the data definitions from the current data area. You can then edit the generated copycode with the program editor.

To generate copycode

1. Open the data area from which you want to generate copycode.
2. From the **Object** menu, choose **Generate** or press SHIFT+G.

   An **Untitled - Copycode** window appears that contains the source code of the data area.
3. Save the copycode as a source object as described in *Saving Objects* in the *Using Natural Studio* documentation.

IV Map Editor

16 Map Editor

- Inserting Map Fields and Menus 108
- Character Encoding of Literal Strings 110
- Modifying Map Contents 110
- Defining Fields 116
- Defining Field Attributes 131
- Defining Arrays 132
- Modifying Field Colors and Representation 134
- Using Field Rules 135
- Defining Data Areas for Maps 140
- Testing Maps 142
- Previewing Maps 142
- Reversing Maps 143
- Modifying the Map Profile 144
- Saving and Cataloging Maps 148

The Natural map editor is used to create and modify 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.

Note: 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.

Related Topics:

- ***Map Editor: GUI Controls and Field-Sensitive Maps*** in *Editor Features With SPoD*
- *Editors in the SPoD Environment* in the *Unicode and Code Page Support* documentation
- *Setting the Options* and *Map Editor Options* in the *Using Natural Studio* documentation
- *Toolbars* in the *Using Natural Studio* documentation
- *Shortcut Keys* in the *Using Natural Studio* documentation

Inserting Map Fields and Menus

You can create the following types of fields:

- text constants
- data fields
- menus
- push buttons
- bitmaps
- toggle buttons
- radio buttons
- selection boxes

Text constants and data fields correspond to the fields used in character-oriented Natural platforms (Mainframe Natural, for example). All other field types apply only to applications with graphical user interfaces.

Note: Map editor field types are not the same as the dialog elements with the same name used in the dialog editor. Whereas the dialog editor's dialog elements are identified by a handle definition in a data area, the map editor's fields are not defined in a data area. Each is therefore addressed differently in Natural code: the map fields are addressed by an `INPUT USING MAP` statement, whereas the dialog elements are addressed by event-driven programming features.

The procedures in this section for inserting map fields assume the use of a mouse. The keyboard equivalents are provided in *Keyboard Equivalents*.

To insert a map field

1 From the **Insert** menu, choose a field type.

Or:

Click the toolbar button for the required field type.

2 Move the mouse pointer into the editor.

The cross-hair pointer is displayed with a symbol for the selected field.

3 Position the mouse pointer at the place in the map where you want to put the field, hold down the left mouse button and drag the mouse to size the field (up, down, right, or left, depending on the type of field).

A box is displayed to indicate graphically the size of the field you are creating. Its actual length is displayed in the **Len** field in the editor status line.

4 Release the mouse button.

The map field appears in the map editor. It is selected and its properties are displayed in the status line.

To create a map menu field using the mouse

- From the **Insert** menu, choose **Menu**.

Or:

Click the **Create Menu** toolbar button.

If no menu has been created yet for the map, a menu bar appears at the top of the map editor with a menu field. The menu field is selected.

If a menu has already been created for the map, then a menu field is added to the menu bar. The menu field is selected.

For information on manipulating map fields once they have been created, see *Modifying Map Contents*.

For information on importing variables from other Natural objects, see *Importing Fields*.

Character Encoding of Literal Strings

If a field contains a literal string (for example, a label) with one or more characters that cannot be retained using the current code page of the map source, a dialog box appears prompting you to encode the characters in UTF-8 (Universal Transformation Format, 8-bit form):

- Choose **Yes** (default) to encode the characters in UTF-8.

  The characters are encoded in UTF-8 as can be seen when saving or compiling the source or when viewing the **Properties** dialog box.

  UTF-8 encoding prevents any character from being replaced by a substitution character.
- Choose **No** if you do not want to use UTF-8 encoding.

  Each character that cannot be retained with the current code page of the map source is replaced by the substitution character specified with the `SUBCHAR` profile parameter (see the *Parameter Reference* documentation).
- Choose **Cancel** to exit the dialog box without any action.

  (Any current command processing is terminated.)

Modifying Map Contents

This section covers the following topics:

- Keyboard Equivalents
- Selecting Fields
- Deselecting Fields
- Copying Fields
- Cutting Fields
- Pasting Fields
- Deleting Fields
- Moving Fields
- Resizing Fields
- Aligning Fields
- Importing Fields

- Importing System Variables

Keyboard Equivalents

Most of the actions described in this section can be performed using the keyboard instead of the mouse. The table below provides key sequences for each action.

This action	Is performed with this keyboard action
Move the mouse pointer	Press arrow keys.
Select map field	Place mouse pointer on field and press SPACEBAR.
Select map fields	Place mouse pointer outside field, press and hold down SPACEBAR, press arrow keys to encircle map fields to be selected, release SPACEBAR.
Deselect map field(s)	Move mouse pointer outside field(s) and press SPACEBAR.
Move map field(s)	Select map field(s), press and hold down SPACEBAR and press arrow keys.
Copy map field	Select map field and press CTRL+C.
Cut map field	Select map field and press CTRL+X.
Paste map field	Select map field and press CTRL+V.
Delete map field	Select map field and press DEL.
Resize map field	Select map field, move mouse pointer to a field handle, press and hold down SPACEBAR and press arrow keys.
Open dialog box to define local data	Press ALT+M+L.
Open dialog box to define parameter data	Press ALT+M+D.

Selecting Fields

To select a single field

- Point to the field you want to select and click the left mouse button.

  The field handles appear indicating that the field is selected.

To select more than one field

- Use either of the following methods:

  1. Point to a spot outside the range of fields to be selected.
  2. Drag the mouse across the map, drawing a rubber band box that surrounds the fields.
  3. Release the mouse button and the field handles appear.

  Or:

1. Select the fields requested by pressing the SHIFT key and the left mouse button together.
2. To move the fields drag the selected area to where needed, by keeping the left mouse button pressed.
3. Release the mouse button to place the fields.

Deselecting Fields

To deselect a field

- Move the pointer away from the field you want to deselect and click the left mouse button.

  The field handles disappear.

Copying Fields

Fields can be copied within the same map or between two different maps.

To copy a field

1 Select the field(s) to be copied using the instructions provided in *Selecting Fields*.

2 From the **Edit** menu, choose **Copy**.

Or:

Click the **Copy** toolbar button.

Or:

Press CTRL+C.

The field is copied to the clipboard and can now be pasted within the same map or another map. For instructions on pasting fields, see *Pasting Fields*.

Cutting Fields

The cut function can be used to delete fields from a map or to move fields within/between maps.

To cut a field

1 Select the field(s) to be cut using the instructions provided in *Selecting Fields*.

2 From the **Edit** menu, choose **Cut**.

Or:

Click the **Cut** toolbar button.

Or:

Press CTRL+X.

The field is cut to the clipboard and can now be pasted within the map or to another map. For instructions on pasting fields, see *Pasting Fields*.

Pasting Fields

The paste function is used to place a field at a specific position within an editor after it has been copied or cut to the clipboard from another position within the same map or another map. A field, which has been copied or cut to the clipboard, can be pasted repeatedly without recopying it.

To paste a field

1 Copy or cut the field to be pasted as described in *Copying Fields* or *Cutting Fields*.

2 If the field is to be pasted in another map, select the map.

3 From the **Edit** menu, choose **Paste**.

Or:

Click the **Paste** toolbar button.

Or:

Press CTRL+V.

The field is pasted to the map.

4 To paste the same field again, repeat Steps 2 and 3.

Deleting Fields

When a field is deleted, it is cut from the map but is *not* placed on the clipboard.

To delete a field or a range of fields

1 Select the field(s) to be deleted using the instructions provided in *Selecting Fields*.

2 Select the field or range of fields.

3 From the **Edit** menu, choose **Delete**.

Or:

Click the **Delete** toolbar button.

Or:

Press DEL.

The field is deleted from the map.

Moving Fields

To move a field or a range of fields to a different location on the map

1 Select the field(s) to be moved using the instructions provided in *Selecting Fields*.

2 Place the pointer within the field handles and drag the field or range of fields to the new location.

3 Release the mouse button.

Resizing Fields

To resize a field

1 Select the field(s) to be resized using the instructions provided in *Selecting Fields*.

2 Point to any of the field handles surrounding the field.

The pointer changes to a double-sided arrow.

3 Drag the mouse until the field(s) reach the required length.

4 Release the mouse button.

Aligning Fields

As an alternative to arranging fields within a map individually using the move function, you can arrange fields more accurately with respect to each other or with respect to the map by using the align function. You can align fields in a map in the following ways:

- Justify selected fields to the left, right, top, or bottom of their field handles.
- Center selected fields vertically or horizontally with respect to each other.
- Center selected fields horizontally with respect to the map editor window.

To align fields

1 Select the field(s) to be aligned using the instructions provided in *Selecting Fields*.

2 From the **Field** menu, choose **Alignment** and, from the cascading menu, one of the entries.

Or:

Click one of the alignment toolbar buttons.

The selected fields are aligned.

Importing Fields

You can import data fields, system variables, toggle buttons, selection boxes, and radio buttons into the active map. Fields can be imported from any object, including DDMs (data definition modules), in any library. Imported fields are placed on the system clipboard. You can paste them into as many map editor windows as required.

To import one or more fields from another object into the map editor window

1 From the **Insert** menu, choose **Import** and one of the following types of field: **Data Field**, **Toggle Button**, **Radio Button**, **Selection Box** or **System Variable**.

(For system variables, see *Importing System Variables*.)

The **Import** dialog box appears. The name of the current library is displayed in the **Library** list box.

2 If the object containing the fields you want to import is located in a different library, open the **Library** list box and select the library.

The list contains all libraries that reside in the current FNAT and FUSER system file, which are displayed as nodes in the Natural Studio tree view (the display can be limited by using the **Display Filter** function of Natural Studio). In addition, the list contains all libraries from inactive system files as specified in the steplib table.

3 From the **Type** group box, select the type of Natural object from which you want to import fields.

A list of all Natural objects in the current library of the type you selected appears in the **Object** list box.

The list contains all objects, which are displayed in the library nodes of the Natural Studio tree view (the display can be limited by using the **Display Filter** function of Natural Studio).

4 Select the object that contains the fields you want to import.

The fields in the selected object appear in the **Importable Data** list box.

5 Select the fields that you want to import.

6 Choose **Import**.

7 Choose **Quit**.

The dialog box closes and the fields appear in the upper left-hand corner of the map editor window. You can move the fields around within the map.

Note: If you import a multiple-value field or a periodic group from a view definition in a local or a global data area, the number of occurrences is automatically copied from the selected field into the array definition box.

Importing System Variables

To import one or more system variables into the map editor window

1 From the **Insert** menu, choose **Import** > **System Variable**.

The **Import System Variable** dialog box appears.

2 Select the system variables that you want to import.

3 Choose **Import**.

4 Choose **Quit**.

The dialog box closes and the system variables appear in the upper left-hand corner of the map editor window. You can move the system variables around within the map editor window.

Note: If you are working with Natural Single Point of Development (SPoD), only the system variables available for the environment you are using (that is, your operating system and Natural version) will be displayed.

Defining Fields

When a map field is inserted, it is given a default field definition and/or default field attributes which can be modified at any time.

To modify a definition for a field

1 Select the field to be modified.

2 From the **Field** menu, choose **Definition**.

The **Field Definition** dialog appears.

This section covers the following topics:

- Defining Text Constants
- Defining Data Fields
- Defining Selection Boxes
- Defining Radio Buttons
- Defining Toggle Buttons

- Defining Menu Items
- Defining Push Buttons
- Defining Bitmaps

Defining Text Constants

This field is always alphanumeric. You can modify the text in a text constant at any time.

To modify a text constant

1 Double-click on the text field to be modified.

Or:

Select the field and choose **Definition** from the **Field** menu.

The background in the text field is highlighted.

2 Enter the required text directly in the text field. For text field modification, you can also use the context menu.

If the text is long, it could be necessary to increase the size of the text field. See *Resizing Fields*.

3 Deselect the text field.

Defining Data Fields

To modify a data field definition

1 Double-click on the field to be modified.

Or:

Select the field and choose **Definition** from the **Field** menu.

The **Field Definition** dialog box appears.

2 The **Field** text box contains the current name of the field. You can change it by typing in a new name.

3 From the **Format** list box, choose a Natural data format for the field. The default format is `A` (alphanumeric).

For valid input values, see *Format and Length of User-Defined Variables* in the *Programming Guide*.

Note: Because the information required to define a field depends on the Natural data format, text or list boxes in the **Field Definition** dialog box may appear/disappear when the format changes.

4 In the **Length** text box, enter the internal length of the field which is used by a program that references this field.

This field is only displayed for Natural data formats F, A, B, I, N, P and U.

For valid input values, see *Format and Length of User-Defined Variables* in the *Programming Guide*.

5 You can specify the output length to be used when displaying the field by using the following boxes:

AL	For Natural data formats A (alphanumeric) and U (Unicode). For valid input values, see the possible settings of the corresponding AL session parameter described in the *Parameter Reference* documentation.
NL	For Natural data formats B (binary), I (integer), N (numeric) and P (packed numeric). For valid input values, see the possible settings of the corresponding NL session parameter described in the *Parameter Reference* documentation.
FL	For Natural data format F (floating point). For valid input values, see the possible settings of the corresponding FL described in the *Parameter Reference* documentation.
DF	For Natural data format D (date). For valid input values, see the possible settings of the corresponding DF session parameter described in the *Parameter Reference* documentation.
DL	For Natural data formats A (alphanumeric) and U (Unicode). This box can be used to specify the output length for Unicode strings which can require extra space. For further information on using this box and valid input values, see the corresponding session parameter DL described in the following documentation: ■ *DL - Display Length for Output* in *Parameter Reference* ■ *Display Length for Output - DL Parameter* in the *Programming Guide* ■ *DL versus AL* in *Session Parameters* in *Unicode and Code Page Support*

For descriptions of all profile and session parameters that can be used to control the output format of fields, see *Parameters to Influence the Output of Fields* in the *Programming Guide*.

6 Select the **SG** toggle button to specify whether a sign position is to be allowed for the field.

This field is only displayed for Natural data formats F, I, N and P.

7 The **Rules** field displays the number of processing rules that are defined for the data field.

8 The **Mode** field displays the current mode of the data field. The following table describes the possible modes.

`Data`	The field was created by selecting a field from a `DEFINE DATA` definition.
`Sys`	The field is a system variable.
`Undef`	The field was created directly on the screen and has a dummy name.
`User`	The name of the field was created by changing the field name
`View`	The field was created by selecting a field from a view (DDM).

9 Select the **Array** toggle button to define an array for the data field.

The **Array** button is enabled. For more information, see *Defining Arrays*.

10 The **AD** field displays the current attribute definition for the data field. For instructions on modifying attribute definitions, see *Defining Field Attributes*.

11 From the **PM** list box, choose a print mode for the field:

blank	The standard character set is used.
C	An alternative character set is used.
I	Inverse print direction.
N	No hardcopy can be made.

For further information, see the session parameter `PM` (Print Mode) in the *Parameter Reference* documentation.

12 From the **CD** list box, choose a color definition for the field content.

Note: You can also define a color for a field by choosing **Color** from the **Field** menu.

13 In the **CV** text box, you can enter a dynamic field attribute control variable.

This is the control variable that will contain the attributes to be used for the data field. The variable must be defined with Natural data format `C` (for attribute control) in the program that references the map.

The control variable also contains a `MODIFIED` data tag, which indicates if 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 will be set if any of the fields referencing the control variable have been modified.

See the *Parameter Reference* documentation for more information on the `CV` parameter.

14 From the **Dim** list box, you can select the number of dimensions for an array of a control variable. The default is none.

You must mark the array box in order to specify the control variable as an array.

15 In the **DY** text box, enter the dynamic string attributes.

The dynamic string parameter is used to assign attributes for dynamic attribute field display. See the *Parameter Reference* documentation for more information on the DY parameter.

16 Select the **ZP** toggle button to specify zero printing for the field. If this button is selected, zero values are printed as one zero. If this button is not selected, zero values are suppressed.

This field is only displayed for Natural data formats F, I, N and P.

17 In the **EM** text box, enter the edit mask to be used for the data field.

See the *Parameter Reference* documentation for more information on the EM and the EMU parameters.

Note: The edit mask overrides the display length.

18 In the **Helproutine** text box, enter the name of a helproutine or help map to be assigned to the map field. The helproutine or help map is then invoked for the map field at execution time when a help request is made for this map field.

In the **Parameters** text box, enter the name of the parameter(s) that are to be passed to the helproutine or help map specified in the **Helproutine** text box. Removing a parameter from the **Parameters** text box 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 **Start From** value (see *Defining Arrays*).

The syntax that applies to specifying names and parameters in the **Helproutine** and **Parameters** text boxes, corresponds to the syntax of the HE session parameter described in *HE Parameter Syntax* (*Parameter Reference* documentation). 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 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:

- 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.

19 Select the **Array** toggle button and choose the **Array** button if you want to define a parameter as an array.

For additional information, see *Defining Arrays*.

Defining Selection Boxes

To define a selection box, use the same procedure as described under *Defining Data Fields*. Note that all variables you define for a selection box must have the same Natural data format.

To define attributes for a selection box, see *Defining Field Attributes*.

To define items for a selection box

1 Access the **Field Definition** dialog box for the selection box.

2 Choose **Items**.

The **Selection Box Definition** dialog box appears. The field name is displayed in the **Selection Box** field. The existing items are displayed in the **Items** list box. A default item `Item` is displayed.

The functions you can execute from the **Selection Box Definition** dialog box, are described in the following section:

- Defining Constant Selection Box Items
- Defining Variable Selection Box Items
- Importing Selection Box Items
- Modifying Selection Box Items
- Removing Selection Box Items
- Moving Selection Box Items

Defining Constant Selection Box Items

To add an item constant

1 Choose **Add Constant**.

The **Selection Box Item - Constant** dialog box appears.

2 In the **Constant** text field, enter the name of the new item.

The length of the constant cannot be longer than the value you specified in the **Length** list box of the **Field Definition** dialog box.

3 Choose **OK** to define the selection box item.

Defining Variable Selection Box Items

To add an item variable

1 Choose **Add Variable**.

The **Selection Box Item - Variable** dialog box appears.

2 In the **Variable** text box, enter the name of the variable to serve as an item.

The variable must be a valid Natural identifier. The length of the variable value is fixed at the value set for **Length** in the **Field Definition** dialog box.

3 If the variable is an array, you can define the array by choosing **Define Array**.

4 Choose **OK** to define the selection box item.

Importing Selection Box Items

To import an item from another object

1 Choose **Import Item**.

The **Import Selection Box Item** dialog box appears. The name of the current library is displayed in the **Library** list box.

2 If the object containing the fields you want to import is located in a different library, open the **Library** list box and select the library.

3 From the **Type** group box, select the type of Natural object from which you want to import fields.

A list of all Natural objects in the current library of the type you selected appears in the **Object** list box.

4 Select the object that contains the fields you want to import.

The fields in the selected object appear in the **Importable Data** list box.

5 Select the fields that you want to import.

6 Choose **Import**.

The dialog box closes and the fields appear at the bottom of the **Items** box.

Modifying Selection Box Items

To modify an item

- From the **Items** list box, select the item to be modified and choose **Modify Item**.

  If the item is a constant, the **Selection Box Item Constant** dialog box appears. See *Defining Constant Selection Box Items*.

  If the item is a variable, the **Selection Box Item Variable** dialog box appears. See *Defining Variable Selection Box Items*.

Removing Selection Box Items

To delete an item

- From the **Items** list box, select the item to be deleted and choose **Remove Item**.

  The item is removed from the **Items** list box.

Moving Selection Box Items

When you add an item to a selection box, it is placed at the bottom of the **Items** list box. In most cases, you will want to reorder these items in logical groupings.

To move an item to another position in the list box

1. In the **Items** list box, select the item to be moved and drag the cursor to the new position.

   A dashed line between items indicates the position to which the item will be moved.
2. Drop the item in the new position.

   The item is inserted at the new position.

Defining Radio Buttons

To define a radio button, use the same procedure as described under *Defining Data Fields*. Note that all variables you define for a radio button must have the same Natural data format.

To define attributes for a radio button, see *Defining Field Attributes*.

Defining Radio Button Contents

To define the contents of a radio button

1 Open the **Field Definition** dialog box for the required radio button.

2 Choose **Contents**.

The **Edit Constant or Variable** dialog box appears.

3 In the **Type** group box of the **Edit Constant or Variable** dialog box, specify the following:

Depending on the type of radio button you want to define, select either the **Constant** or the **Variable** radio button.

In the **Name** text box, enter the name of the constant or variable. The length of the name must not exceed the length specified in the **Length** text box.

For information on naming conventions for constants and variables, see *Field Definitions* in the *Programming Guide*.

If **Variable** is selected, you can choose the **Import** command button to import an alphanumeric field from another Natural object.

For information on how to import variables from other Natural objects, see *Importing Fields*.

If **Variable** is selected and you want to define an array, you can select the **Array** toggle button and choose the **Define** command button. For information on how to define an array, see *Defining Arrays*.

4 Choose **OK** when you have finished typing in all items to be defined.

The **Edit Constant or Variable** dialog box is closed.

Defining Toggle Buttons

To define a toggle button, use the same procedure as described under *Defining Data Fields*.

To define attributes for a toggle, see *Defining Field Attributes*.

Defining a Toggle Button Label

The Natural data format is always L (Logical).

To define the label of a toggle button

1 Access the **Field Definition** dialog box for the required toggle button.

2 Choose **Label**.

The **Edit Constant or Variable** dialog box appears.

3 In the **Type** group box of the **Edit Constant or Variable** dialog box, specify the following:

Depending on the type of toggle button you want to define, select either the **Constant** or the **Variable** radio button.

In the **Name** text box, enter the name of the constant or variable. The length of the name must not exceed the length specified in the **Length** text box.

For information on naming conventions for constants and variables, see *Field Definitions* in the *Programming Guide*.

If **Variable** is selected, you can choose the **Import** command button to import an alphanumeric field from another Natural object.

For information on how to import variables from other Natural objects, see ***Importing Fields***.

4 If **Variable** is selected and you want to define an array, you can select the **Array** toggle button and choose the **Define** command button. For information on how to define an array, see ***Defining Arrays***.

5 Choose **OK** when you have finished typing in all items to be defined.

The **Edit Constant or Variable** dialog box is closed.

Defining Menu Items

To define items for a menu

- In the menu bar, double-click on any menu item.

The **Edit Menu** dialog box appears.

The functions you can execute from the **Edit Menu** dialog box, are described in the following section:

- Editing a Menu Name
- Adding Menu Items

- Adding a Submenu
- Adding Menu Separators
- Modifying Menu Items
- Moving a Menu Item
- Removing Menu Items

Editing a Menu Name

To edit a menu name

1 From the **Edit Menu** dialog box, choose **Menu Name**.

The **Edit Constant or Variable** dialog box appears.

2 In the **Type** group box, specify whether the menu name is to be a constant or a variable.

3 In the **Name** text box, modify the name of the constant or variable. If you want to import a variable from another Natural object, select **Variable** and choose **Import**.

For information on how to import variables from other Natural objects, see *Importing Fields*.

4 If the menu is an array, select the **Array** toggle button and choose **Define**.

For information on how to define an array, see *Defining Arrays*.

5 Choose **OK**.

Adding Menu Items

To add items to a menu

1 From the **Edit Menu** dialog box, choose **Add Item**.

The **Define Menu Item** dialog box appears.

2 Choose **Item Name**.

The **Edit Constant or Variable** dialog box appears.

3 In the **Type** group box, specify whether the menu item is to be a constant or a variable.

4 In the **Name** text box, modify the name of the constant or variable. If you want to import a variable from another Natural object, select **Variable** and choose **Import**.

For information on how to import variables from other Natural objects, see *Importing Fields*.

5 If the item is an array, select the **Array** toggle button and choose **Define**.

For information on how to define an array, see *Defining Arrays*.

6 Choose **OK**.

The **Define Menu Item** dialog box appears.

7 From the **Key name** list box, choose a PF-key name for the item.

PF keys that have already been allocated do not appear in the list box.

8 Select the **Enabled** toggle button to permit menu item selection. Otherwise, the menu item cannot be selected.

9 Choose **OK**.

The **Edit Menu** dialog box appears with the new menu item.

Adding a Submenu

To define a submenu for a menu item

1 From the **Edit Menu** dialog box, choose **Add Submenu**.

The **Edit Submenu** dialog box appears.

2 From the **Edit Submenu** dialog box, choose **Submenu Name**.

The **Edit Constant or Variable** dialog box appears.

3 In the **Type** group box, specify whether the submenu name is to be a constant or a variable.

4 In the **Name** text box, modify the name of the constant or variable. If you want to import a variable from another Natural object, select **Variable** and choose **Import**.

For information on how to import fields from other Natural objects, see *Importing Fields*.

5 If the submenu is an array, select the **Array** toggle button and choose **Define**.

For information on how to define an array, see *Defining Arrays*.

6 Choose **OK**.

The **Edit Submenu** dialog box appears.

7 To add items to the submenu, choose **Add Item** and follow the instructions under *Adding Menu Items*.

8 To add a separator to the submenu, choose **Add Separator** and follow the instructions under *Adding Menu Separators*.

9 To modify a submenu item, choose **Modify Item** and follow the instructions under *Modifying Menu Items*.

10 To remove a submenu item, choose **Remove Item** and follow the instructions under *Removing Menu Items*.

11 Choose **OK** to complete the submenu definition/modification.

Adding Menu Separators

You can designate logical groupings in your menus or submenus by separating these groupings with horizontal lines.

To add a separator to a menu or submenu

1 From the **Edit Menu** or **Edit Submenu** dialog box, choose **Add Separator**.

A selected separator is placed behind the last menu item.

2 Drag the separator to the new position.

A dashed line between items indicates the position to which the separator will be moved.

3 Drop the separator in the new position.

The separator is inserted at the new position.

Modifying Menu Items

To modify a menu item

1 In the **Menu Items** list box of the **Edit Menu** dialog box, select the item to be modified and choose **Modify Item**.

The **Define Menu Item** dialog box appears.

2 Follow the instructions as described in *Adding Menu Items*.

Moving a Menu Item

When you add an item to a menu or submenu, it is placed at the bottom of the **Menu Items** list box. In most cases, you will want to reorder these items in logical groupings.

To move an item to another position in the list box

1 In the **Menu Items** list box of the **Edit Menu** dialog box, select the menu item to be moved and drag the cursor to the new position.

A dashed line between items indicates the position to which the item will be moved.

2 Drop the item in the new position.

The item is inserted at the new position.

Removing Menu Items

To delete a menu item

- In the **Menu Items** list box, select the menu item to be deleted and choose **Remove Item**.

  The menu item is removed from the **Menu Items** list box.

Defining Push Buttons

To define a push button

1. Double-click on the push button to be defined.

   The **Define Push Button** dialog box appears.
2. Choose **Edit Label**.

   The **Edit Constant or Variable** dialog box appears.
3. In the **Type** group box of the **Edit Constant or Variable** dialog box, specify the following:

   Depending on the type of push button you want to define, select either the **Constant** or the **Variable** radio button.

   In the **Name** text box, enter the name of the constant or variable. The length of the name must not exceed the length specified in the **Length** text box.

   For information on naming conventions for constants and variables, see *Field Definitions* in the *Programming Guide*.

   If **Variable** is selected, you can choose the **Import** command button to import an alphanumeric field from another Natural object.

   For information on how to import variables from other Natural objects, see *Importing Fields*.
4. If **Variable** is selected and you want to define an array, you can select the **Array** toggle button and choose the **Define** command button. For information on how to define an array, see *Defining Arrays*.
5. Choose **OK**.

   The **Define Push Button** dialog box appears.
6. From the **Key name** list box, choose a PF-key name for the push button.

   PF keys that have already been allocated do not appear in the list box.
7. Select the **Enabled** toggle button to permit push button selection. Otherwise, the push button cannot be selected.

8 Choose **OK**.

The push button is displayed with the new name.

Defining Bitmaps

To define a bitmap

1 Double-click on the bitmap to be defined.

The **Define Bitmap** dialog box appears.

2 Choose **Edit Label**.

The **Edit Constant or Variable** dialog box appears.

3 In the **Type** group box, specify whether the bitmap name is to be a constant or a variable.

4 In the **Name** text box, modify the name of the constant or variable. If you want to import a variable from another Natural object, select **Variable** and choose **Import**.

For information on how to import variables from other Natural objects, see *Importing Fields*.

5 If the bitmap is an array, select the **Array** toggle button and choose **Define**.

For information on how to define an array, see *Defining Arrays*.

6 Choose **OK**.

The **Define Bitmap** dialog box appears.

7 Choose **Edit File Name**.

The **Edit Constant or Variable** dialog box appears.

8 Repeat Steps 3 to 6.

9 From the **Key name** list box, choose a PF-key name for the bitmap.

PF keys that have already been allocated do not appear in the list box.

10 Select the **Enabled** toggle button to permit bitmap selection. Otherwise, the bitmap cannot be selected.

11 Select the **Scale bitmap** toggle button to cause the bitmap to be displayed with the size defined by the map. Otherwise, the bitmap is displayed in its original size.

12 Choose **OK**.

The bitmap is displayed with the new name.

Defining Field Attributes

Field attributes can be defined for data fields, toggle buttons, radio buttons and selection boxes.

To define the attributes for a field

1 In the **Field Definition** dialog box of the required field, choose **Attributes**.

The **Attribute Definitions** dialog box appears.

2 To change the definition for an attribute, open the list box for the attribute and select a different value. A description of the options for each attribute is provided in the following table.

3 To change the filler character, in the **Filler character** text box, enter a different character.

Empty input fields or modifiable output fields are filled with this character. When you edit an input field, which is padded with blanks to its maximum length, it may appear as if the text cannot be edited. In such cases, the blanks must be explicitly deleted from the end of the field.

4 Choose **OK**.

Attribute	Code	Explanation
Field Representation		
Blinking	B	Value is displayed blinking.
Italic	C	Value is displayed italic/cursive.
Default intensity	D	Value is displayed with normal intensity.
Intensified	I	Value is displayed intensified.
Non-display	N	Value entered in field will not be displayed.
Underlined	U	Value is displayed underlined.
Reverse video	V	Value is displayed in reverse video.
Dynamic attributes	Y	Attributes are controlled via a control variable.
Field Alignment		
Left-justified	L	Value is displayed left-justified.
Right-justified	R	Value is displayed right-justified.
Leading zeros	Z	Numeric values are displayed with leading zeros, right-justified.
Field Input/Output Characteristics		
Input field, non-protected	A	Field is an input field and non-protected.
Output field, modifiable	M	Field is an output field and can be modified.
Output field, protected	O	Field is an output field and write-protected.
Mandatory Input Characteristics		

Attribute	Code	Explanation
Input mandatory	E	Value must be entered in field. Only relevant for input-only fields (AD=A).
Input optional	F	Value can, but need not, be entered in the field.
Length of Input Value Characteristics		
Fixed input length	G	Value entered in field must have same length as field. Only relevant for input-only fields.
Variable input length	H	Value entered in field can be shorter than field.
Field Upper/Lower Case Characteristics		
Translate to upper	T	Value entered is translated to upper case.
Accept lower case	W	Lower-case values are to be accepted.

Defining Arrays

You can define an array of up to three dimensions for a data field. The order in which the dimensions of the array are mapped to the map layout is determined by the values you enter.

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

The **Array** command button is inactive unless the **Array** toggle button is selected.

To define an array and its dimensions for a data field

1 Choose **Array**.

The **Define Map Array** dialog box appears.

2 From the **Dimensions** list box, select the number of dimensions for the array.

The default number of dimensions is 1. Text boxes appear for the number of dimensions specified.

3 In the **Line spacing** text box, enter the number of blank lines to be inserted horizontally between each dimension occurrence in the array.

4 In the **Column spacing** text box, enter the number of blank columns to be inserted vertically between each dimension occurrence in the array.

5 For a field defined as a three-dimensional array the **Cls/Ls** text box is displayed. Enter the number of blank lines to be inserted vertically or columns to be inserted horizontally between dimensions 1, 2, or 3, depending on the layout definition for dimension 3.

The values specified in the **Layout** list boxes determine in which direction each of the dimensions are painted in the edit area of the map. (V = vertical, H = horizontal, VD/HD = vertical/horizontal, position of third dimension)

6 In the **Start from** text box, enter the starting index value for each dimension.

You can enter a number or the name of a variable; the actual value is supplied in the program that invokes the map definition. Unless defined otherwise as a field in the map, the variable is assumed to be defined as Natural data format/length N7.

Note: Removing a **Start from** value from an array implies that the variable is removed from the map, too, unless it is a map field or it is associated to any other map field as a **Start from** value or help parameter.

7 In the **Upper bounds** text box, enter a value for each dimension.

This number is the highest occurrence of the first, second and third dimension. A field defined in a program (a user-defined variable or a database field) can be imported to define the map array. In this case, the upper bounds of the field, as defined in the program, are used. All values can be overwritten for imported data fields. However, when you choose **OK** in the **Field Definition** dialog box, you are asked if you really want to change an imported field's definition.

If the array is initially defined in the map, the number of occurrences defined for the vertical, horizontal and fixed indexes sets the upper bounds.

8 In the **Occurrences** text box, enter a value for each dimension.

This is the number of occurrences that will be displayed on the map. This does not apply to the third dimension of a 3-dimensional array because only two ranges of occurrences can be displayed on the screen. One-dimensional arrays can be displayed as multi-line/multi-column fields. For these arrays, the second number of occurrences and the layout for the second dimension can be defined.

9 From the **Layout** list box, select a layout value for each dimension.

Layout values determine the axis assumed by each dimension of the array. This determines how the array is represented in the map layout. For arrays of one and two dimensions, the only possible values are H (horizontal axis) and V (vertical axis). For arrays with three dimensions, you must decide how the third dimension will be represented. The options are HD (horizontally detached) or VD (vertically detached). The dimension specified as HD or VD is represented by members, which are grouped in a horizontal or vertical orientation.

10 Choose **OK**.

The array is defined and you are returned to the **Field Definition** dialog box.

When defining multi-dimensional arrays, note that the defaults for the first and second dimensions have been reversed, which means that horizontal will be vertical and vice versa; in addition, the

first and second dimensions can now also be specified as “fixed”; the third dimension is still “fixed” by default.

Changing the Number of Displayed Occurrences in an Array

Once you have defined an array, you can use the mouse to graphically modify the number of displayed occurrences and, indirectly, the upper bounds without accessing the **Define Map Array** dialog box.

To modify the number of displayed occurrences and/or upper bounds for an array

1 Place the mouse pointer on one of the four corner field handles.

2 Drag the mouse to increase/decrease the number of elements in the array.

During resizing, a dotted outline indicates the intended modification.

3 When you have reached the required size for the array, release the mouse button.

If you increase the number of displayed occurrences for a dimension beyond the upper bound, then the upper bound is automatically increased as well. If you decrease the number of displayed occurrences for a dimension, the upper bound remains at its previous value.

Modifying Field Colors and Representation

For any text constant, data field, toggle button, radio button or selection box you can specify a color and a display style for its name.

To define the color and/or representation for a field

1 In the map editor, select the required field.

2 From the **Field** menu, choose **Color**.

Or:

Click the **Field Color** toolbar button.

The **Field Color and Representation** dialog box appears.

3 From the **Color Selection** group, select a color.

4 From the **Field Representation** group, select an option.

5 Choose **OK**.

Using Field Rules

Field rules can be defined for any data field, toggle button, radio button or selection box. A field can have up to 100 processing rules (ranked `0 - 99`). At map execution time, the processing rules are executed in ascending order, first by rank, then by screen position of the field.

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 rules can be defined as either inline processing rules or Predict free rules.

Inline processing rules are rules that are defined within a map source and do not have a name assigned.

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.

You can access Predict free rules that are stored in either a local or a remote Predict environment located on a Linux, Windows, or mainframe computer.

Predict free rules that are stored in a remote environment on a Linux, Windows, or mainframe host can also 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* documentation) and *Dictionary Server Assignments* in the section *Overview of Configuration File* (*Configuration Utility* documentation).

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.

This section covers the following topics:

- Creating Field Rules
- Copying Field Rules
- Editing Field Rules
- Changing Field Rule Ranks
- Unlinking or Deleting Field Rules
- Linking Predict Free Rules
- Converting Predict Free Rules to Inline Rules
- Defining Key Rules

Creating Field Rules

To create a field rule

1 Select the map field for which you want to create a field rule.

2 From the **Field** menu, choose **Rules**.

The **Field Rule** dialog box appears.

3 From the **Field Rule** dialog box, choose **Create**.

The program editor appears.

4 Enter the rule.

5 When you are finished, save the rule by choosing **Save** from the **Object** menu.

The **Rule Selection** dialog box appears.

6 From the **Rank** list box, select a rank number.

7 Choose **OK**.

8 Close the program editor.

Copying Field Rules

To copy a field rule

1 Select the map field containing the field rule to be copied.

2 From the **Field** menu, choose **Rules**.

The **Field Rules** dialog box appears.

3 From the **Ranks** list box, select the number of the rule to be copied.

4 Choose **Copy**.

The **Rule Selection** dialog box appears.

5 From the **Rank** list box, select a rank for the new rule.

6 Choose **OK**.

If you chose the rank number of an existing rule, then you are asked whether you want to overwrite the rule. Otherwise, the rule is copied and the **Field Rules** dialog box is redisplayed. The rule is displayed in the **Rule Text Fragment** box.

Editing Field Rules

To edit a field rule

1 Select the map field containing the field rule to be edited.

2 From the **Field** menu, choose **Rules**.

The **Field Rules** dialog box appears.

3 From the **Ranks** list box, select the rank number of the rule to be edited.

4 Choose **Edit**.

The program editor appears.

5 Edit the rule.

6 When you are finished, save the rule by choosing **Save** from the **Object** menu.

7 Close the program editor.

Changing Field Rule Ranks

To change the rank of a field rule

1 Select the map field containing the field rule to be modified.
2 From the **Field** menu, choose **Rules**.

The **Field Rule** dialog box appears.

3 In the **Ranks** list box, select the rank number of the rule to be reranked.
4 Choose **Move**.

The **Rule Selection** dialog box appears.

5 In the **Rank** list box, select another rank for the rule.
6 Choose **OK**.

Unlinking or Deleting Field Rules

Unlinking is the same as deleting a field rule.

To delete a field rule

1 Select the map field containing the field rule to be deleted.
2 From the **Field** menu, choose **Rules**.

The **Field Rule** dialog box appears.

3 In the **Ranks** list box, select the rank number of the rule to be deleted.
4 Choose **Unlink**.

You are asked if you really want to unlink (delete) the field rule.

5 Choose **Yes** to delete the field rule.

Linking Predict Free Rules

To link a free rule to a selected field

1 Select the map field to which you want to assign a Predict free rule.
2 From the **Field** menu, choose **Rules**.

The **Field Rule** dialog box appears.

3 Choose **Free Rule**.

Since free rules cannot be created but only selected, the **Create** command button changes to **Select**.

4 Choose **Select**.

The **Free Rule Selection** dialog box appears.

5 Enter the rule name, the Predict owner and up to five keywords under which the rule is stored in Predict.

Asterisk notation can be used for the rule name, the keywords can be combined with the Boolean operators `AND` and `OR`, and a `BUT NOT` keyword can be specified, too.

6 Choose **OK**.

A list of free rules is displayed, from which you can select the rule(s) you want to link to the field.

Choose **Read Source** to read the source code into the **Free Rule Text Fragment** information box of a selected rule.

7 Choose **OK**.

With each selected rule, you are asked to assign a rank.

8 Specify a rule rank.

The rule is now linked to the field. At this point, you can convert a linked free rule into an inline rule as described in *Converting Predict Free Rules to Inline Rules*.

Converting Predict Free Rules to Inline Rules

To convert a linked Predict free rule to an inline rule

1 From the **Field** menu, choose **Rules**.

The **Field Rule** dialog box appears.

2 Select the rank of the free rule you want to convert.

The **Free rule** toggle button is selected and the free rule appears in the **Rule Text Fragment** information box.

3 Deselect the **Free rule** toggle button.

A message box appears asking you whether you want to change the rule type to inline.

4 Choose **OK**.

The free rule becomes an inline rule and its source code is displayed in the **Rule Text Fragment** information box.

Defining Key Rules

The **PF-Key Rules** function allows you to create, edit, move, copy and unlink (delete) function-key related processing rules for the active map. Key rules also can be defined as either inline processing rules or Predict free rules.

Key rules can be used to assign activities to program sensitive function keys during map processing. For function keys that 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.

To create, copy, and edit function key rules

- From the **Map** menu, choose **PF-Key Rules**.

  The **PF-Key Rules** dialog box appears.

  Key rules are defined exactly like field rules. For detailed instructions, see *Using Field Rules*.

Defining Data Areas for Maps

You can define a local data area for a map to define variables to be used for map processing rules. You can define a parameter data area for a map to define the parameters that can be received from programs.

The procedures for defining, modifying, and deleting local and parameter data elements are virtually identical and are therefore explained generically in the following section.

- Defining a Data Element
- Modifying a Data Element

- Removing a Data Element

Defining a Data Element

To define a data element for a map

1 From the **Map** menu, choose **Local Data** or **Parameter Data**.

The **Define Local/Parameter Data** dialog box appears.

2 Choose **Add** to add a data definition.

The **Data Definition** dialog box appears.

3 In the **Name** text box, enter the name of the data element to be defined.

4 From the **Format** drop-down list box, select a Natural data format.

5 If you selected Natural data format `A`, `B`, `F`, `I`, `N` or `U`, in the **Length** box, enter the field length.

6 If the data element is an array, in the **Dimensions** text box, select the number of dimensions (`1` - `3`).

The **Lower bounds** and **Upper bounds** text boxes appear for each dimension.

7 Enter the lower bound and upper bound for each dimension in the array.

8 Choose **OK**.

The **Define Local/Parameter Data** dialog box appears. The data element is displayed in the information box.

Modifying a Data Element

To modify a data element for a map

1 From the **Map** menu, choose **Local Data** or **Parameter Data**.

The **Define Local/Parameter Data** dialog box appears.

2 In the information box, select the data element to be modified.

3 Choose **Modify** to modify the data definition.

The **Data Definition** dialog box appears.

4 Modify the data definition as required.

5 Choose **OK**.

The **Define Local/Parameter Data** dialog box appears. The modified data element is displayed in the information box.

Removing a Data Element

To delete a data element for a map

1 From the **Map** menu, choose **Local Data** or **Parameter Data**.

The **Define Local/Parameter Data** dialog box appears.

2 In the information box, select the data element to be deleted.

3 Choose **Remove** to delete the data definition.

The data definition is removed from the information box.

4 Choose **OK**.

Testing Maps

Once you have created and successfully saved a map, you can test it so see how it will appear in an application.

To test a map

1 Open the map to be tested.

2 From the **Object** menu, choose **Test**.

The Natural output window is opened and the map appears as it would in the application.

3 Double-click on the output window or press ENTER or ESC to return to the map editor.

Previewing Maps

Maps that are larger than the area shown in the map editor window can be scrolled using the scroll bars on the right, and at the bottom of the map editor window. Extremely large maps can be displayed in the map editor window by using the **Preview Mode** function.

When preview mode is active, the display size of the map fields is reduced so that the entire map fits into the map editor window. All map editor functions work normally when preview mode is in effect.

To display a map in preview mode

- From the **Window** menu, choose **Preview Mode**.

The map is displayed in preview mode.

Reversing Maps

You can toggle the screen direction of the active map from left-to-right to right-to-left, or vice versa. This is only a visual change; the physical field position defined in the map source is retained.

See also *Bidirectional Language Support* in the *Unicode and Code Page Support* documentation.

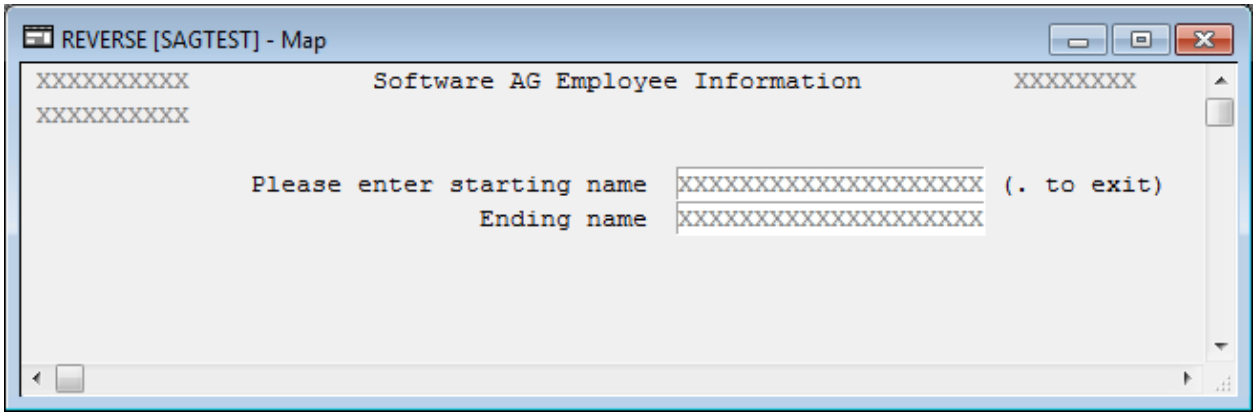
To reverse the screen direction of a map

- From the **Map** menu, choose **Reverse Map**.

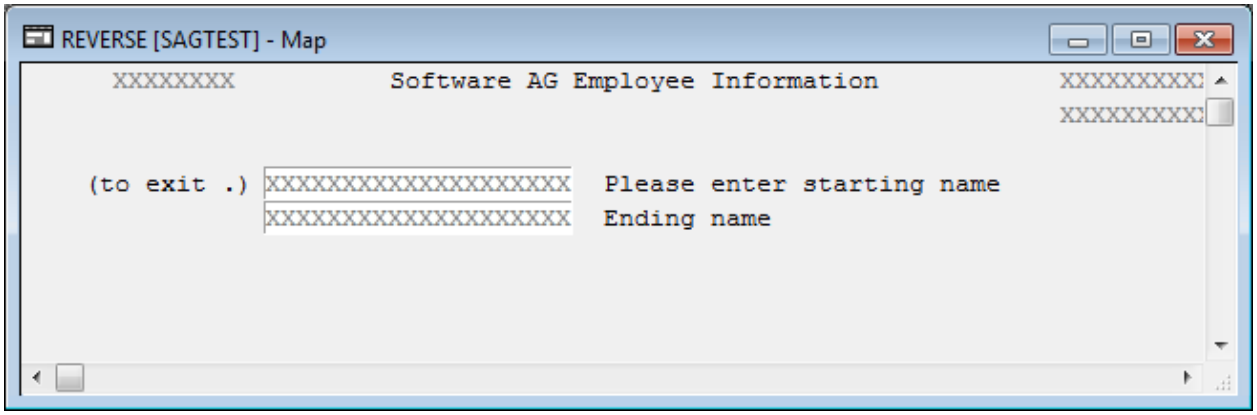
  The screen direction of the map is reversed.

Example:

The default screen (left-to-right) of a map looks as follows:



The reversed screen (right-to-left) of this map looks as follows:



Modifying the Map Profile

With the map profile, you set the profile parameters for the active map. These parameters are saved with the map.

To set profile parameters

1 From the **Map** menu, choose **Map Profile**.

The **SYSPROF - Map Editor Profile** dialog box appears.

2 Set profile parameters. Each parameter is described in the table below.

3 Choose **OK** to save the profile for the active map.

Map editor profile option settings are described in the following table.

Note: The settings for **Zero printing** and **Upper case** are copied into the field definition when a new field is created. These settings can be modified for each new field.

Option	Explanation
Format group box:	
Page size	The number of map lines to be edited (1 - 250). If **Standard keys** is selected, the number of lines is restricted to the range 3 - 250. For a map that is output with a WRITE statement, specify the number of lines of the logical page output, not the map size. The map can then be output several times on one page.
Line size	The number of map columns (5 - 249).
Column shift	Only available in a mapped mainframe environment. Column shift (0 or 1) to be applied to the map. This feature can be used to address all 80 columns on a 80-column screen (**Column shift** = 1, **Line size** = 80).
Layout	The name of the map that serves as standard layout for the current map. You can use this option to simplify the creation of many similar maps by creating one map as the

<table>
<tr><th>Option</th><th>Explanation</th></tr>
<tr><td></td><td>basic layout map with a set of fields to be used by the other fields. In all similar maps, you specify the name of the standard layout map and you only add the fields that are specific to the new map. See also the Dynamic layout option below.</td></tr>
<tr><td>Decimal char.</td><td>The character to be used as the decimal notation character; the default character is a period (.). You can change the default decimal character with either of the following methods:

■ Use Tools > Session Parameter as described in Using Session Parameters (Using Natural Studio documentation).
■ Use the GLOBALS system command as described in the System Commands documentation.</td></tr>
<tr><td>Print mode</td><td>The default print mode for variables. This value is copied into the field definition when a new field is created.

Possible settings are:

<table><tr><td>blank</td><td>The standard character set is used (this is the default setting).</td></tr><tr><td>C</td><td>An alternative character set is to be used.</td></tr><tr><td>I</td><td>Inverse print direction.</td></tr><tr><td>N</td><td>No hardcopy of the display can be made.</td></tr></table></td></tr>
<tr><td>Control var.</td><td>The name of an attribute control variable, the content of which determines the attribute characteristics of fields and texts that have the attribute definition AD=Y. The attribute control variable referenced in the map must be defined in the program using that map.

Removing an attribute control variable from the map editor profile implies that the attribute control variable is removed from the map, too, unless it is associated to any other map field.</td></tr>
<tr><td>Standard keys</td><td>When you select this check box, the last two lines of the map remain empty so that function key specifications can be entered at execution time.

When you clear this check box, all lines are used for the map.</td></tr>
<tr><td>Upper case</td><td>When you select this check box, input is converted to upper case during map execution.

When you clear this check box, input is not converted to upper case during map execution.</td></tr>
</table>

Option	Explanation
Field sensitive	When you select this check box, the consistency check for a map field is made as soon as the field is filled by the user. When you clear this check box, field checking is performed when the map is filled completely.
Dynamic layout	If you have specified the name of a standard layout map in the **Layout** text box, you select this check box to determine that fields are imported dynamically into the layout at runtime. This means that you can alter your standard layout and this change will automatically be reflected in all similar maps using this layout. When you clear this check box, the standard layout is not used dynamically.
Zero printing	When you select this check box, numeric fields that contain all zeroes (print one zero, right-justified) are printed. When you clear this check box, the printing of numeric fields that contain all zeroes is suppressed.
Right justified	When you select this check box, numeric and alphanumeric fields copied from data definitions in other Natural objects are right-justified. When you clear this check box, numeric and alphanumeric fields copied from data definitions in other Natural objects are not right justified.
Manual skip	When you select this check box, the cursor is *not* moved automatically to the next field in the map at execution time, even if the current field is completely filled. When you clear this check box, the cursor is moved automatically to the next field in the map at execution time.
Context group box:	
Device check	A device name can be viewed in this field.
WRITE statement	When you select this check box, the result of the map definition process is a `WRITE` statement and the resulting (output) map can be invoked from a 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. When you clear this check box, the result of the map definition process is an `INPUT` statement and the resulting map can be invoked from a program using an `INPUT` statement. **Note:** This check box is disabled for `INPUT` maps containing GUI controls.
Helproutine	The name of the helproutine or help map that is called at execution time when the help function is invoked for this map (global help for the map). The syntax that applies to entering values in the **Helproutine** text box corresponds to the syntax of the `HE` session parameter described in *HE Parameter Syntax* (*Parameter Reference* documentation).

Option	Explanation
Parameters	Parameters can only be defined if a helproutine or a help map is specified in the **Helproutine** text box. Enter the name of the parameter(s) that will be called at execution time when the help function is invoked. The syntax that applies to entering values in the **Parameters** text box corresponds to the syntax of the HE session parameter described in *HE Parameter Syntax* (*Parameter Reference* documentation). **Note:** If the combined length of the name entered in **Helproutine** and **Parameters** exceeds 19 characters, the map editor truncates it.
As field default	Only applies if the **Helproutine** text box is filled. When you select this check box, the helproutine or help map specified for the map applies as the default to each individual field on the map, that is, the name of each field is passed individually to the helproutine or help map. This option can only be chosen if a helproutine or a help map is specified in the **Helproutine** text box. When you clear this check box, the name of the map is passed to the helproutine or help map specified in the **Helproutine** text box.
Position line	The position (line, vertical) where the specified help map is being output.
Column	The position (column, horizontal) where the specified help map is being output.
Help text	When you select this check box, the map is marked as help text. When you clear this check box, the map is not marked as help text.
AutoRuleRank	The rank (priority) assigned to Predict automatic rules when they are linked to the map during field definition. Default is 1.
Filler characters group box:	
Optional, partial	Indicates that a field is optional and can be partially filled.
Optional, complete	Indicates that a field is optional but must be filled completely if it is used.
Required, partial	Indicates that a field is required but can be partially filled.
Required, complete	Indicates that a field is required and must be filled completely. **Note:** If you enter a field using the mouse, the cursor will be placed to the right of the filler character. The filler character is blank by default, and the field appears to be not modifiable. Press BACKSPACE to delete the blank, and enter a new filler character.

Saving and Cataloging Maps

You can save a map definition as a source object and/or a cataloged object (generated program) in the current Natural library in the current Natural system file.

For the naming conventions that apply to an object, refer to *Object Naming Conventions* in the *Using Natural Studio* documentation.

To save a map as a source object

- Proceed as described in *Saving Objects* in the *Using Natural Studio* documentation.

To save a map as a source object and/or a cataloged object

- Proceed as described in either *Stowing Objects* or *Cataloging Objects* in the *Using Natural Studio* documentation.

V DDM Editor

The DDM editor is used to create, maintain and delete Natural data definition modules (DDMs).

Tip: You can generate, print and store data reports from DDMs by using the *data browser*, which is described in the *Tools and Utilities* documentation.

Principles of Operation
Creating DDMs
Invoking the DDM Editor
Using the DDM Editor
Saving and Cataloging a DDM
Listing DDMs
Maintaining DDMs in Different Environments
Data Conversion for Adabas or RDBMS
Data Conversion for Tamino

Related Topics:

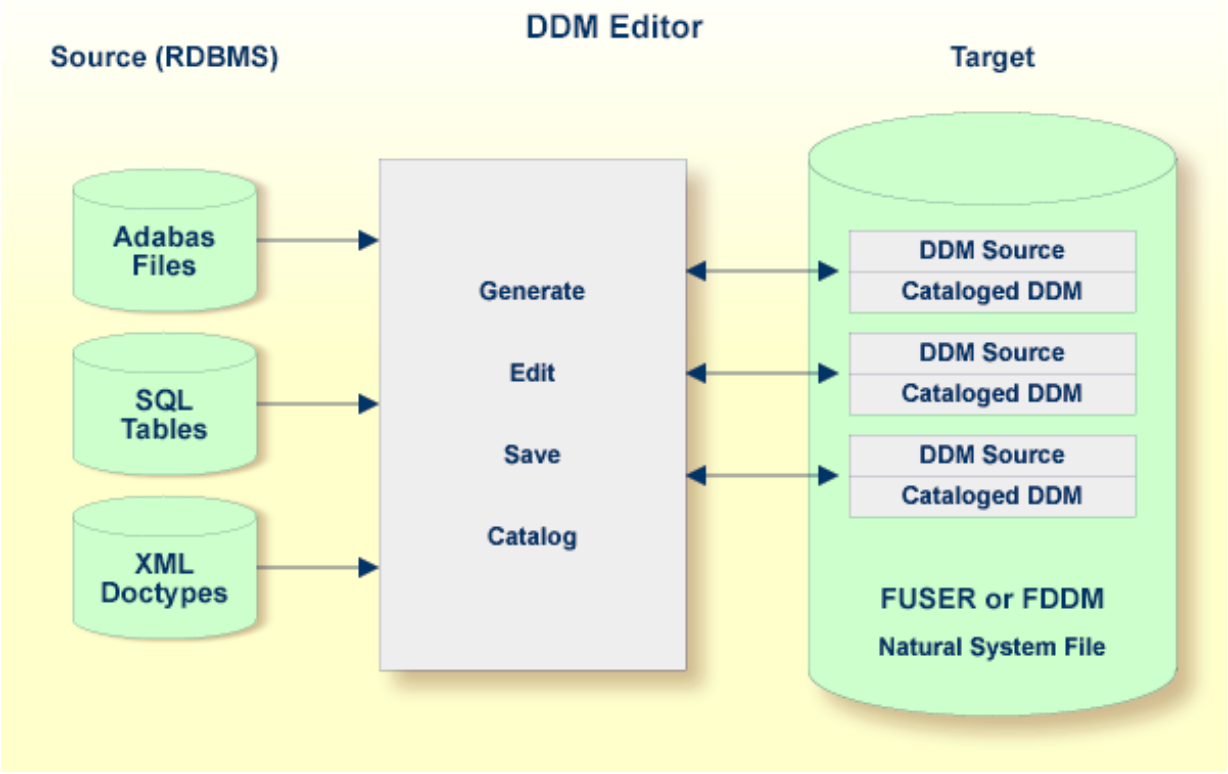
- *DDM Editor: Database Features* in *Editor Features With SPoD*
- *Editors in the SPoD Environment* in the *Unicode and Code Page Support* documentation
- *Setting the Options* and *DDM Editor Options* in the *Using Natural Studio* documentation
- *Toolbars* in the *Using Natural Studio* documentation
- *Shortcut Keys* and *DDM Editor Shortcut Keys* in the *Using Natural Studio* documentation

17 Principles of Operation

- Storing DDMs - FDDM System File 152
- Restrictions of Use 153

The DDM editor is 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 the DDM editor:



Storing DDMs - FDDM System File

DDMs reside either in individual Natural user libraries or system libraries contained in the system file FUSER or FNAT, FDIC (remote environments on a mainframe platform) 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.

In Natural Studio, DDMs are contained in the DDMs node of the tree view. Depending on the FDDM parameter setting, the DDMs node is either a subnode of a user library (FUSER) or a system library (FNAT), or represents the FDDM system file with the DDMs node located on the same level as the node User Libraries or System Libraries. In a remote environment located on a mainframe platform, the DDMs node represents the FDIC system file.

You cannot link DDMs contained in user or system libraries to an application. Therefore, in the Application Workspace, you can only use either DDMs from the FDIC system file (remote on mainframes) or the FDDM system file (remote on Linux).

DDMs are not restricted to files stored in an Adabas database. Some options described in the *DDM Editor* 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 editor functions under:

- Predict
- Natural Security

Predict

To guarantee data integrity for DDMs defined in Predict, the Predict administrator can restrict the use of DDM editor 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.

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.

18 Creating DDMs

- Copying DDMs 156
- Creating DDMs from Adabas 156
- Creating DDMs from SQL 158
- Creating Multiple DDMs from SQL 161
- Creating DDMs from Tamino 162
- Creating DDMs for VSAM 164

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.

To copy a DDM

1 From a DDMs node or subnode, choose a DDM and open the object.

The DDM editor is invoked with the DDM source displayed in the editing area.

2 From the **Object** menu, choose **Save As**.

The **Save As** dialog box appears.

3 In the **Name** text box, enter a new DDM name and, in the **Libraries** combo box, select a library if required.

The new DDM is saved as a source object and is added to the corresponding DDMs node or subnode.

4 From the **Object** menu, choose **Stow** to save the new DDM as a cataloged object. See also *Saving and Cataloging a DDM*.

You can also copy single or multiple objects by using the copy-and-paste or drag-and-drop feature of Natural Studio as described in *Copying Objects* in Natural Studio documentation.

If you want to copy DDMs between different libraries, database files, and/or hardware platforms, see also *Maintaining DDMs in Different Environments*.

Creating DDMs from Adabas

To create a DDM from an Adabas database

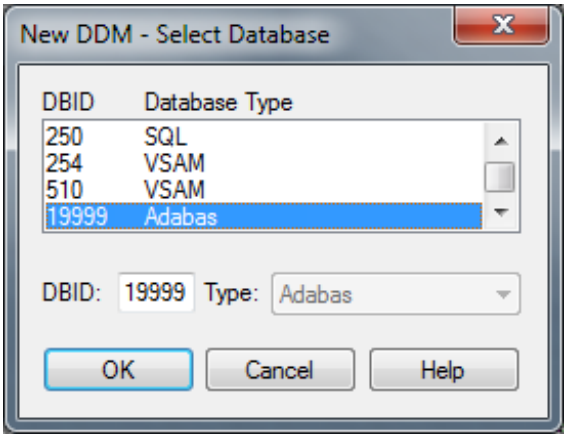
1 Select the DDMs node or subnode or log on to the library where you want to create the DDM.

2 From the **Object** menu, choose **New > DDM**.

Or:

Choose the **New DDM** toolbar button.

The **New DDM - Select Database** dialog box appears as shown in the example below:



3 Select an Adabas database. The DBID (database ID) and database type are then shown in the corresponding **DBID** and **Type** boxes.

Or:

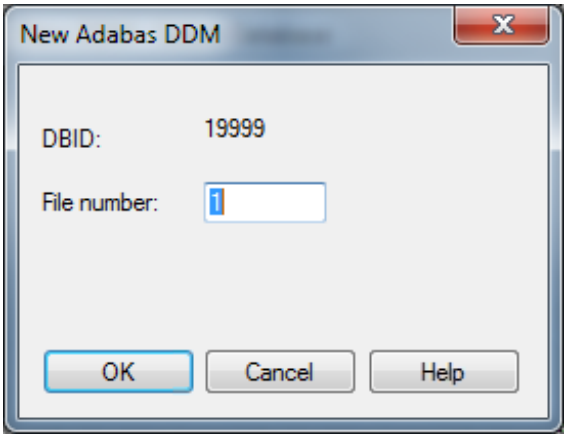
In the **DBID** text box, enter a numeric value in the range from `0` to `65535` (except `255`).

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 you enter a DBID that is not yet contained in the list box, from the **Type** combo box, select Adabas if required.

4 Choose **OK**.

If the specified DBID identifies an Adabas database, the **New Adabas DDM** dialog box appears as shown in the example below:



5 In the **File number** text box, enter a numeric value in the range from `1` to `5000` and choose **OK**.

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 source: see the section *Using the DDM Editor*.

7 After editing, from the **Object** menu, choose **Stow** and, in the **Stow As** dialog box, enter the name of the new DDM.

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*.

Creating DDMs from SQL

To create a DDM from an SQL database

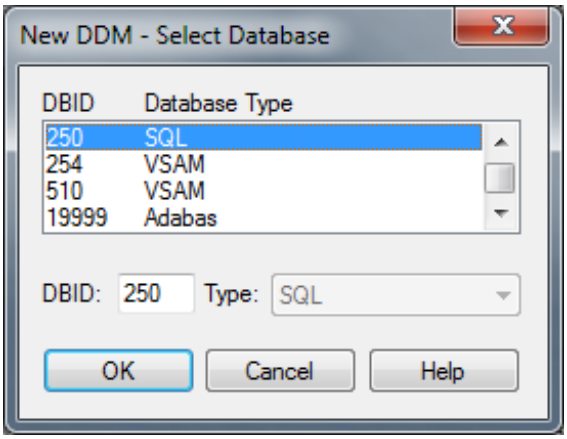
1 Select the DDMs node or subnode or log on to the library where you want to create the DDM.

2 From the **Object** menu, choose **New > DDM**.

Or:

Choose the **New DDM** toolbar button.

The **New DDM - Select Database** dialog box similar to the example below appears:



3 Select an SQL database. The DBID (database ID) and database type are then shown in the corresponding **DBID** and **Type** boxes.

Or:

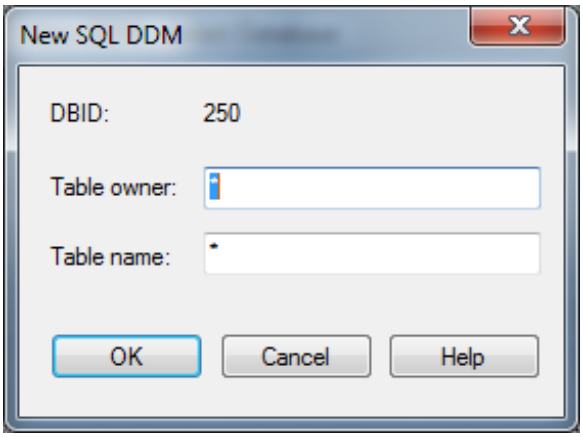
In the **DBID** text box, enter a numeric value in the range from `0` to `65535` (except `255`).

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 you enter a DBID that is not yet contained in the list box, from the **Type** combo box, select `SQL` if required.

4 Choose **OK**.

If the specified DBID identifies an SQL database, the **New SQL DDM** dialog box appears as shown in the example below:



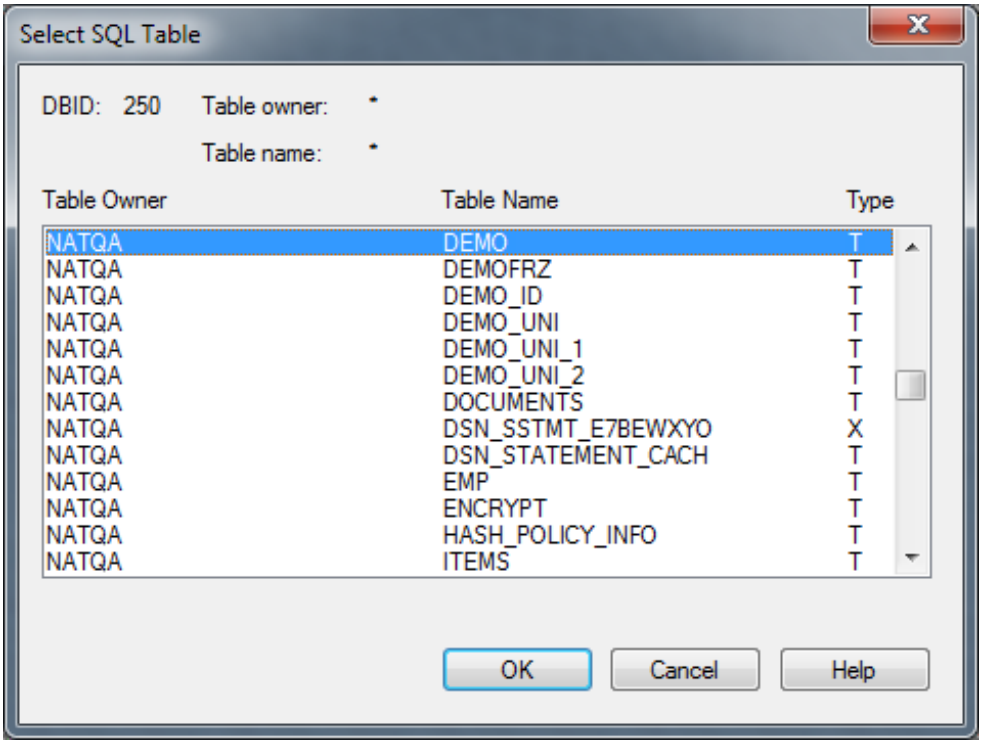
Select the SQL table for which a DDM is to be created:

- To list all SQL tables for selection, use the asterisks (*) in the text boxes **Table owner** and **Table name**. The asterisks (*) are entered by default.
- To list a particular range of SQL tables, use asterisk (*) notation, for example, `AB*` selects all SQL tables with names that start with AB.

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 Choose **OK**.

The **Select SQL Table** list box appears with the selection list of SQL tables specified in Step 4 as shown in the example below:



6 From the list box, select the required SQL table and choose **OK**.

Depending on your SQL database settings, the **Database Logon** window appears if you access this SQL database for the first time in this session. Enter the user ID and the password specified for the database and choose **OK**.

If the specified database and the file are available, the DDM editor is invoked and the SQL table selected 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.

7 If required, edit the DDM source: see the section *Using the DDM Editor*.

8 After editing, from the **Object** menu, choose **Stow** and, in the **Stow As** dialog box, enter the name of the new DDM.

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*.

Creating Multiple DDMs from SQL

The following section only applies to a local environment.

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 In the command line of Natural Studio, enter the following:

```
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.

Creating DDMs from Tamino

Only applies to a local Windows environment, or a remote environment on a Windows or a Linux platform.

To create a new DDM from a Tamino database

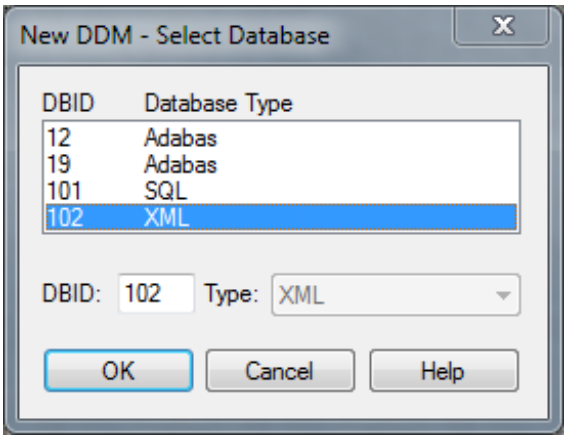
1 Select the DDMs node or subnode or log on to the library where you want to create the DDM.

2 From the **Object** menu, choose **New > DDM**.

Or:

Choose the **New DDM** toolbar button.

The **New DDM - Select Database** dialog box appears as shown in the example below:



3 Select an XML database. The DBID (database ID) and database type are then shown in the corresponding **DBID** and **Type** boxes.

The file number (**FNR**) of the DDM is always `1`. The file number cannot be modified.

Or:

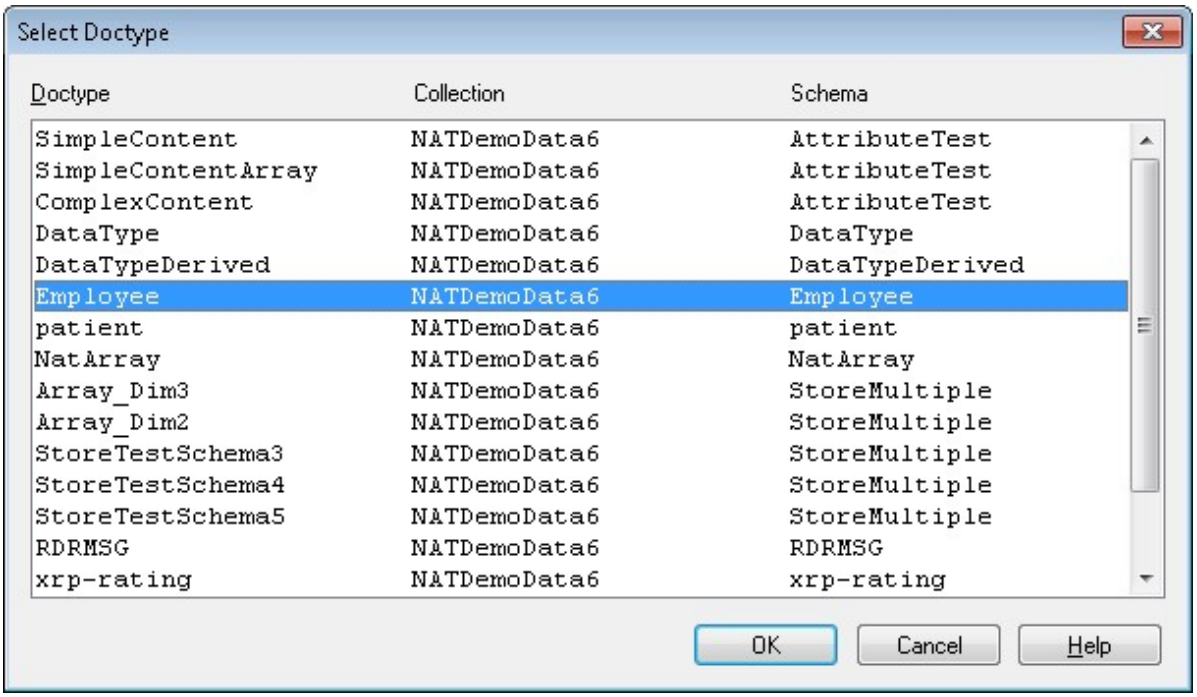
In the **DBID** text box, enter a numeric value in the range from 0 to 65535 (except 255).

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 you enter a DBID that is not yet contained in the list box, from the **Type** combo box, select `XML` if required.

4 Choose **OK**.

If the specified DBID identifies a Tamino database, the **Select Doctype** window appears with a list of available doctypes as shown in the example below:



5 From the list, select a doctype and choose **OK**.

If you are accessing this Tamino database for the first time in this session, the **Database Logon** window appears. Enter the user ID and password for the database and choose **OK**.

The DDM editor is invoked and the DDM generated from the selected doctype is read into the editing area.

6 If required, edit the DDM source: see the section *Using the DDM Editor*.

7 After editing, from the **Object** menu, choose **Stow** and, in the **Stow As** dialog box, enter the name of the new DDM.

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*.

Creating DDMs for VSAM

Only applies to a remote environment on a mainframe platform.

DDMs for VSAM files are not generated by Natural. Therefore, instead of selecting VSAM file definitions from a list, you enter DDM field attribute definitions directly in the editing area by using the **New** option of the **Object** menu.

19 Invoking the DDM Editor

The DDM editor is used to edit the source of a DDM.

This section describes how to invoke the DDM editor for an existing DDM source. If you want to create a new DDM, follow the instructions provided in *Creating DDMs*.

To invoke the DDM editor for an existing DDM

- In the **Logical View**, expand a DDMs node or subnode, and select and open the required DDM.

  Or:

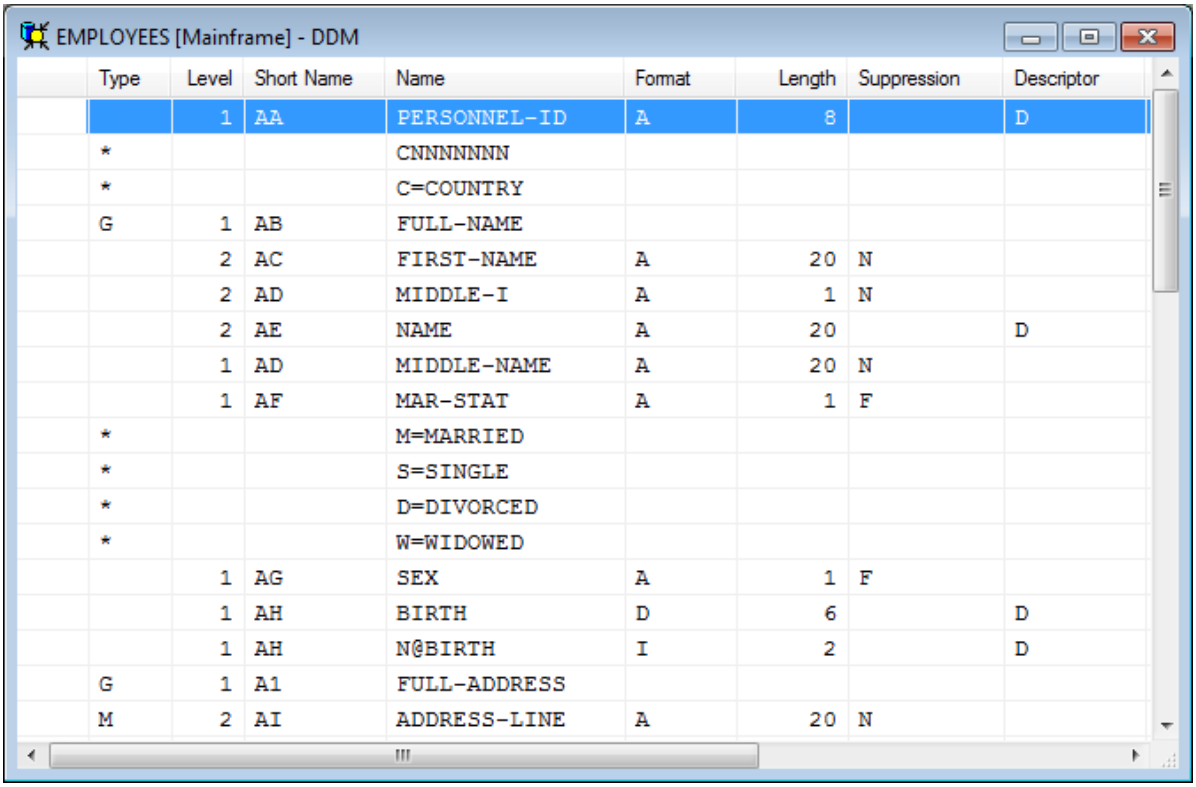
  Enter the following system command:

  ```
  EDIT VIEW object-name
  ```

  where *object-name* denotes the name of the DDM to be edited.

  For information on all options available with `EDIT`, see the relevant section in the *System Commands* documentation.

  A DDM editor window similar to the example below appears:



Type	Level	Short Name	Name	Format	Length	Suppression	Descriptor
	1	AA	PERSONNEL-ID	A	8		D
*			CNNNNNNN				
*			C=COUNTRY				
G	1	AB	FULL-NAME				
	2	AC	FIRST-NAME	A	20	N	
	2	AD	MIDDLE-I	A	1	N	
	2	AE	NAME	A	20		D
	1	AD	MIDDLE-NAME	A	20	N	
	1	AF	MAR-STAT	A	1	F	
*			M=MARRIED				
*			S=SINGLE				
*			D=DIVORCED				
*			W=WIDOWED				
	1	AG	SEX	A	1	F	
	1	AH	BIRTH	D	6		D
	1	AH	N@BIRTH	I	2		D
G	1	A1	FULL-ADDRESS				
M	2	AI	ADDRESS-LINE	A	20	N	

For detailed information on editing a DDM source, see *Using the DDM Editor*.

For information on saving and cataloging a DDM source, see *Saving and Cataloging a DDM*.

20 Using the DDM Editor

- DDM Header Information 168
- Using File Coupling 171
- Columns of Field Attributes 172
- Selecting Fields or Field Attributes 178
- Inserting and Modifying Fields 180
- Copying, Cutting and Pasting Fields 182
- Finding and Replacing Field Names 183
- Deleting Fields 186
- Rearranging Columns 186
- Showing or Hiding Fields 190
- Specifying Extended Field Attributes 192
- Displaying Descriptor Information 198

The DDM editor is organized in a table where the field definition data is contained in rows and columns. The editor provides a separate row for each field defined for a DDM. All attribute definitions that belong to a field are contained in the cells of this row.

You can add a new field to a DDM source or modify the attributes of an existing field by using one of the following methods:

- Enter each attribute definition in the respective cell of a field row or replace existing definitions.
- Copy fields from the current database file if available.
- Copy fields within a DDM or from another DDM by using copy and paste functionality.

This section describes the columns contained in the DDM editor window and the functions provided to create or modify a DDM field, change the display of the DDM editor window, rearrange columns, or hide fields that are not required for a current editing operation.

Related Topics in the Using Natural Studio Documentation:

- *Setting the Options* and *DDM Editor Options*
- *Shortcut Keys* and *DDM Editor Shortcut Keys*

DDM Header Information

This section describes the fields contained in the **DDM Header** dialog box that are used to display and change the correlation between the database and the DDM.

The information contained in the **DDM Header** dialog box is similar to the information displayed in the status bar. For details on the status bar, refer to *Status Bar* in the *Using Natural Studio* documentation. You can switch the status bar on or off by setting the corresponding DDM editor option as described in the *Using Natural Studio* documentation.

- Displaying and Modifying DDM Header Fields
- Explanation of DDM Header Fields

Displaying and Modifying DDM Header Fields

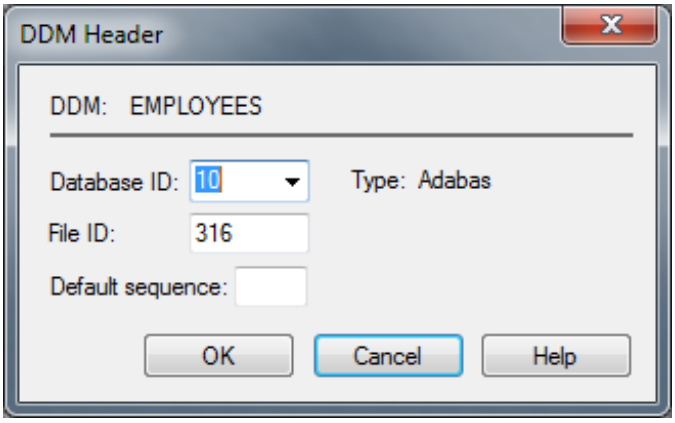
To display and modify DDM header fields

1 In the active DDM editor window, choose **DDM Header**. from the **DDM** menu.

Or:

Choose the **DDM Header** toolbar button.

The **DDM Header** dialog box appears with the name of the current DDM displayed at the top of the box:

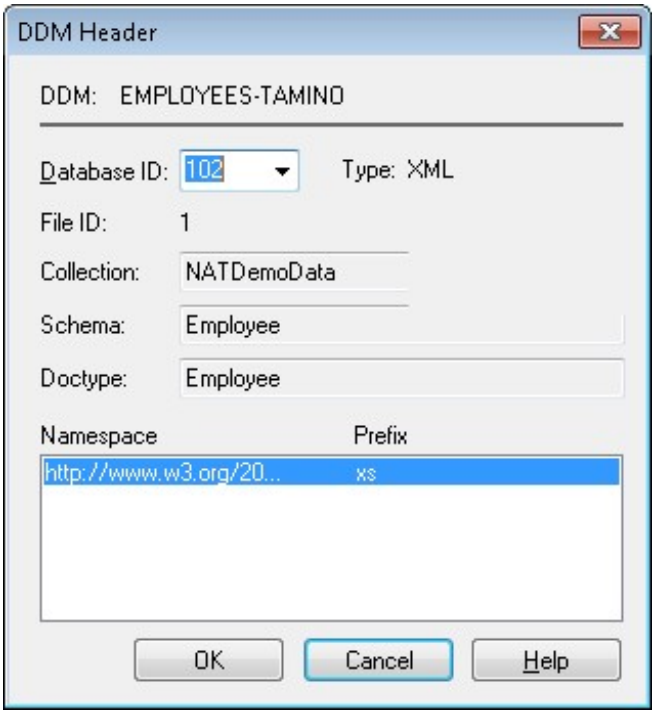


2 From the **Database ID** combo box, choose an ID, or enter an ID in the box: see also **Database ID** in *Explanation of DDM Header Fields*.

In the **File ID** text box, enter a new value: see also **File ID** in *Explanation of DDM Header Fields*.

In the **Default sequence** text box (not applicable to Tamino), enter a short name as default sequence: see also **Default sequence** in *Explanation of DDM Header Fields*.

For Tamino, the **DDM Header** dialog box also displays read-only doctype information as shown in the example below:



3 Choose **OK** to save the new values.

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
Database ID	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.
File ID	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`
DDM	The name of the DDM currently contained in the work area of the DDM editor.
Default sequence	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.
Type	The type of database.
Collection	Read-only Tamino-specific doctype information. The name of the collection which is used within the Tamino database.
Schema	Read-only Tamino-specific doctype information. The name of the Tamino XML schema which is used within the Tamino database.
Doctype	Read-only Tamino-specific doctype information. The name of the doctype within the collection.
Namespace URI Prefix	Read-only Tamino-specific doctype information. The list of namespace URI/prefix pairs which corresponds to the doctype.

Using File Coupling

Only applies to Adabas.

You can use the **File Coupling** option to list or specify Adabas files that are physically coupled to a DDM.

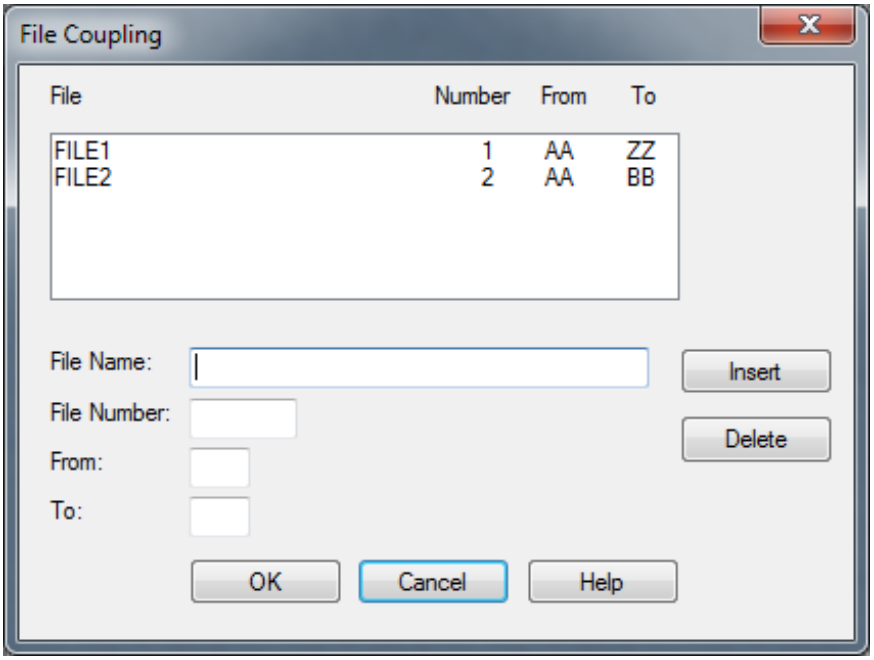
To list or specify a coupled file

1 In the active DDM editor window, from the **DDM** menu, choose **File Coupling**.

Or:

Choose the **File Coupling** toolbar button.

The **File Coupling** dialog box appears as shown in the example below:



All files coupled to the current DDM are listed together with the short names of the descriptors used for coupling.

2 In the **File Name** text box, enter the name of the file to be coupled to the DDM.

In the **File Number** text box, enter the number of the file to be coupled to the DDM.

In the **From** text box, enter the field short name, at which the file coupling begins.

In the **To** text box, enter the field short name, at which the file coupling ends.

3 Choose **Insert** to add the defined entry to the list box.

Or:

Choose **Delete** to remove the selected entry from the list box.

For further information on physical file coupling, refer to the *Adabas* documentation.

Columns of Field Attributes

This section describes the field attributes that can be defined in the rows and columns of the DDM editor window.

The display of the columns depends on whether a column is relevant for the DDM being edited. For example, Tamino-specific information is not displayed for a DDM created from an Adabas database.

Column Heading	Field Attribute	
None	The indicator column is displayed in the leftmost section of the editor window. It can contain the following signs which appear next to the appropriate row:	
		An error sign which indicates incorrect syntax. You are then required to enter a valid value. A tool tip provides error information.
		An information sign which warns you of potential problems caused by the value entered. A tool tip helps evaluate and eliminate the problem.
	⊟ or ⊞	A toggle key which indicates an expanded or a collapsed block of fields (see also *Showing or Hiding Fields*).
Type	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.

Column Heading	Field Attribute	
	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.
Level	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.	
Short Name	Not applicable to Tamino. The **Short Name** column displays the two-character short name of the corresponding field in the database file. In a remote environment: The two-character code used for segment types in DL/I. **Creating Fields:** If you create a new DDM field and the display of the **Short Name** column is switched off, the DDM editor assigns to the new field a new 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 **Insert** function as described in ***Inserting and Modifying Fields***.	

<table>
<tr><th>Column Heading</th><th>Field Attribute</th></tr>
<tr><td>Name</td><td>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.

In a remote environment on a mainframe platform, the name can be 1 - 19 characters long for DL/I fields.

The rules to create a name comply with the naming conventions for user-defined variables (see the Using Natural Studio 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).</td></tr>
<tr><td>Format</td><td>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.

To modify the format of a field, see also the explanations in Inserting and Modifying Fields.</td></tr>
<tr><td>Length</td><td>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.

Depending on the Natural data format selected from the Format drop-down list, the Length column is preset to one of the following values:
<table>
<tr><td></td><td></td></tr>
<tr><td>10</td><td>for formats A, B and U</td></tr>
<tr><td>4</td><td>for formats F and I</td></tr>
<tr><td>7</td><td>for formats N and P</td></tr>
<tr><td>6</td><td>for format D (standard value)</td></tr>
<tr><td>1</td><td>for format L (standard value)</td></tr>
</table>
</td></tr>
</table>

Column Heading	Field Attribute	
	12	for format T (standard value)
	For further information, see *DDM Generation and Editing for Varying Length Columns* in the *Programming Guide*.	
Suppression	Not applicable to Tamino. Null-value suppression option:	
	blank	Indicates that standard Adabas suppression is used; that is, trailing blanks in alphanumeric fields and leading zeros in numeric fields are suppressed.
	F	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.
	N	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.
	M	Indicates that the field is defined with the SQL null-value option not null. The **Remarks** text box (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.
Descriptor	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 **Descriptor** 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:	
	blank	No descriptor. This field is not a descriptor.
	A	Only applies to a remote mainframe environment. Indicates that the field is an alternate index for a VSAM file.

Column Heading	Field Attribute	
	D	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 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). See also *Displaying Descriptor Information*.
	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. See also *Displaying Descriptor Information*.
	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. See also *Displaying Descriptor Information*.
	X	Only applies to a remote mainframe environment. Alternate subdescriptor or superdescriptor; that is, an alternate index for a VSAM file.
Header	The header to be produced for each field specified in a DISPLAY statement: see *Specifying Extended Field Attributes*.	

Column Heading	Field Attribute
Edit Mask	The edit mask to be used: see *Specifying Extended Field Attributes*.
Remarks	A comment which applies to a field and/or the DDM.
Tag Name	Tamino-specific information as described in *Tamino-Specific Extended Field Attributes*.
XPath	
Occurrence	
Flags	
Default Value	
Fixed Value	
SQLTYPE	SQL-specific information as described in *SQL-Specific Extended Field Attributes*.

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.

Selecting Fields or Field Attributes

Before you perform an editor function, you select (highlight) the row or row cell where you want to create, modify or delete a field.

- Selecting Single Fields
- Selecting Multiple Fields
- Selecting Field Attributes

Selecting Single Fields

To select a single field if a cell is selected

- Press SHIFT+SPACEBAR.

  The field row of the cell is selected.

  Or:

  Click on the leftmost column of the field row you want to select.

  The specified field row is selected.

To select a single field if a row is selected

- Click on the field row you want to select.

  Or:

  Navigate to the field row you want to select by pressing UP-ARROW, DOWN-ARROW, HOME or END.

  The specified field row is selected.

Selecting Multiple Fields

To select a range of fields if a cell is selected

1 Press SHIFT+SPACEBAR.

  The field row of the cell is selected.

  Or:

  Click on the leftmost column of the first field row in the range.

The specified field row is selected.

2 Hold down SHIFT while you select the row of the last field in the range.

The rows of the specified field range are selected.

To select a range of fields if a row is selected

1 Click on the leftmost column of the first field row in the range.

Or:

Navigate to the first field row in the range by pressing UP-ARROW, DOWN-ARROW, HOME or END.

The first field row in the range is selected.

2 Hold down SHIFT while you select the row of the last field in the range.

The rows of the specified field range are selected.

To select all fields

- From the **Edit** menu, choose **Select All**.

Or:

Choose the **Select All** toolbar button.

Or:

Press CTRL+A.

All field rows contained in the current DDM source are selected.

Selecting Field Attributes

To select a field attribute if a cell is selected

- Click on the row cell where you want to add or modify an attribute.

Or:

Navigate to the row cell where you want to add or modify an attribute by pressing TAB, SHIFT+TAB, UP-ARROW, DOWN-ARROW, LEFT-ARROW, RIGHT-ARROW, HOME or END.

The specified row cell is selected.

To select a field attribute if a row is selected

- Press F2.

  The leftmost cell of the field row is selected.

  Or:

  First, click on the row that contains the cell you want to select and then click on the cell where you want to add or modify an attribute.

  The specified row cell is selected.

Inserting and Modifying Fields

This section provides instructions for adding fields into a DDM source or modifying fields within a DDM source.

To insert a field from the current database file

1 Select the row where you want to place the new field.

  The insert position (before or after the selected field) of the new field depends on the current setting of:

  - The **Insert After On/Off** toolbar button.
  - The **Insert before/Insert after** editor option (see: *DDM Editor Options* in the *Using Natural Studio* documentation).

2 From the **Field** menu, choose **Insert**.

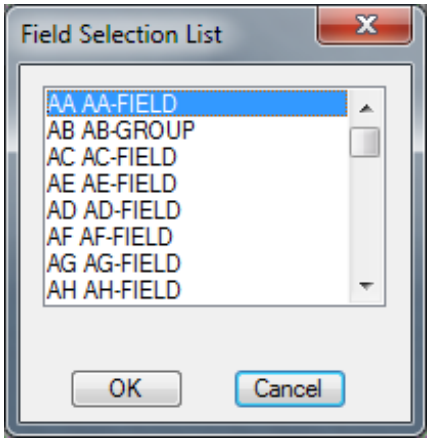
  Or:

  Choose the **Insert Field** toolbar button.

  Or:

  Select a row and press INS, or select a cell in the *last* row and press DOWN-ARROW.

  - If you are editing a DDM that references fields in the database file, the **Field Selection List** dialog box appears as shown in the example below:



From the **Field Selection List** dialog box, select a field.

The new field is pasted into the DDM source either before or after the field selected in Step 1.

- If you are editing a DDM that cannot reference the corresponding fields in the database file, a *blank* row is inserted before or after the selected field.

Note: You cannot validate new field attributes if the database is not available. See also the ***Creating Fields*** note for column **Short Name** in the section *Field Attribute Definitions*.

3 In any blank row cell, you can enter a value for the field attribute to be added or choose a value from a selection box depending on the type of field you want to define.

When you insert a field of the type group or periodic group, the level of each subsequent field is automatically incremented properly.

To modify a field

- Select the row cell that contains the field attribute definition you want to change and either overwrite the existing value or choose a value from a selection box.

  When you modify the level of a field of the type group or periodic group, the level of each subsequent field is automatically incremented or decremented properly, depending on the new level value.

  When you modify the Natural data format of a field, the current length is kept if it is also valid for the new data format. Otherwise, the current length specification is automatically replaced by a valid default length (see also the description of the **Length** column).

Copying, Cutting and Pasting Fields

The copy/cut and paste functions of the DDM editor are used to copy, move or delete one or more fields within the current DDM source or between different DDM sources.

To copy or cut and paste fields

1 Select the fields to be copied or cut.

2 From the **Edit** menu, choose **Copy** or **Cut**.

Or:

Choose the **Copy** or **Cut** toolbar button.

Or:

Press CTRL+C or CTRL+X.

The fields are placed on the clipboard and can be pasted into the current DDM source or in another DDM source.

3 If the fields are to be pasted into another DDM, open the appropriate DDM source.

4 Select the field before or after which the copied or cut fields are to be pasted (see also the **insert position in *To insert a field***).

5 From the **Edit** menu, choose **Paste**.

Or:

Choose the **Paste** toolbar button.

Or:

Press CTRL+V.

The copied or cut fields are pasted into the current DDM source at the specified position.

6 To paste the same fields again, repeat Steps 3 through 5.

When you cut or paste a field of the type group or periodic group, the level of each subsequent field is automatically adjusted properly.

Finding and Replacing Field Names

The search function is used to search for field names and replace field names in the current DDM source.

The find function is performed on all data definitions including collapsed blocks of fields (see also *Showing or Hiding Fields*).

Caution: There is no undo function available to restore original names.

To search for a DDM field name

1 From the **Edit** menu, choose **Find**.

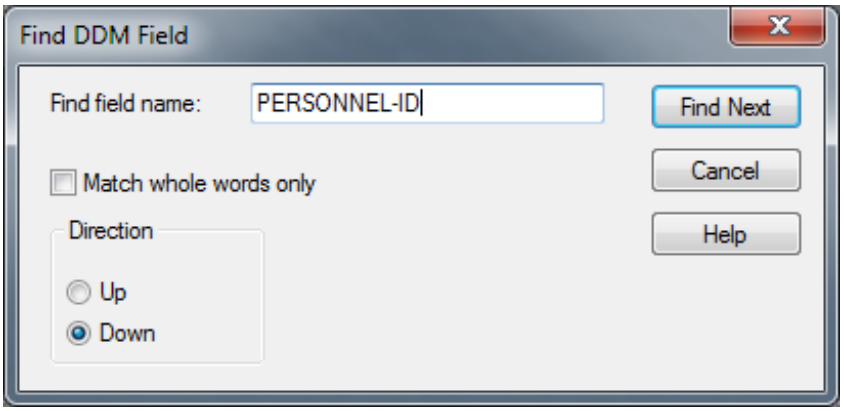
Or:

Choose the **Find** toolbar button.

Or:

Press CTRL+F.

The **Find DDM Field** dialog box appears as shown in the example below:



2 In the **Find field name** text box, enter the long name of the field for which to search (in the example above: `PERSONNEL-ID`).

Set the **Match whole words only** check box if you want to find whole field names only and not parts of field names. If the box is not set, all instances of the search string will be found.

In the **Direction** section, set the option button **Up** or **Down** to specify whether the search is to be performed from the cursor position to the end of the DDM source or from the cursor position to the beginning of the DDM source. The default setting is **Down**.

3 Choose **Find Next**.

If no instance of the search string is found, an appropriate message is displayed.

If an instance of the search string is found, it will be selected.

4 To search for additional instances of the search string:

from the **Edit** menu, choose **Find Next**.

Or:

Press F3.

Or:

Choose the **Find Next** toolbar button.

To replace a field name

1 From the **Edit** menu, choose **Replace**.

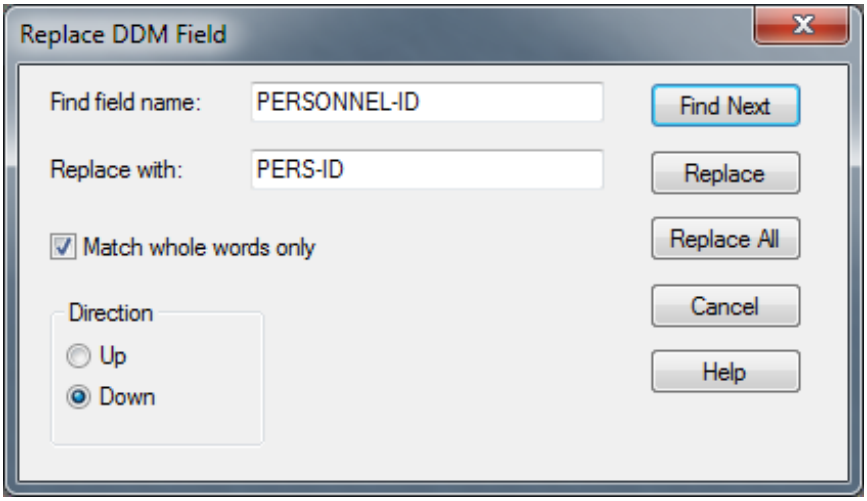
Or:

Choose the **Replace** toolbar button.

Or:

Press CTRL+H.

The **Replace DDM Field** dialog box appears as shown in the example below:



2 In the **Find field name** text box, enter a search string.

In the **Replace with** text box, enter a replacement string.

Set the **Match whole words only** check box if you want to find whole field names only and not parts of field names. If the box is not set, all instances of the search string will be found.

In the **Direction** section, set the option button **Up** or **Down** to specify whether the search is to be performed from the cursor position to the end of the DDM source or from the cursor position to the beginning of the DDM source. The default setting is **Down**.

3 Choose **Replace** to replace the next hit found in the source.

Choose **Find Next** and **Replace** to find the next hit and replace it.

Or:

Choose the **Find Next** and **Replace** toolbar buttons.

Or:

Choose **Replace Next** to replace the next hit found without selecting the hit first.

Or:

Choose **Replace All** to replace all search strings found.

If no instance of the search string is found, an appropriate message is displayed.

4 Choose **Close** to exit the dialog box.

Deleting Fields

When a field is deleted, it is cut from the DDM source but is *not* placed on the clipboard. Once deleted, the field can no longer be recovered.

To delete fields from the DDM editor

1 Select the field(s) you want to delete.

2 From the **Edit** menu, choose **Delete**.

Or:

Choose the **Delete** toolbar button.

Or:

Press DEL.

The fields are deleted from the DDM source and cannot be recovered.

When you delete a field of the type group or periodic group, the level of each subsequent field is automatically decremented properly.

Rearranging Columns

In the editor window, you can adjust the display of the DDM to your needs by resizing, moving or hiding columns that are not required for an editing operation in the current DDM.

- Resizing Columns
- Moving Columns
- Hiding or Displaying Columns

Resizing Columns

You can automatically adjust a single column or all columns to the best size, or change the width of a single column to a specific size.

To resize all columns to best fit

- Choose one of the following methods:
  - Select a field as described in *Selecting Single Fields*.

- From the **View** menu, choose **Customize Columns**.

  Or:
  In any column heading, click the right mouse button and choose **Customize Columns** from the context menu.

  The **Customize Columns** dialog box appears.
- Select the **Best Fit** check box. This option is not selected by default.

All columns in the active editor window are automatically resized to the size that best fits into the editor window whereby the column headings always remain visible.

Or:

Press CTRL+PLUS.

Or:

If you want to apply **Best Fit** to all active editor windows, set the corresponding editor option described in *DDM Editor Options* in the *Using Natural Studio* documentation.

To resize all columns to best fit while typing in text

1 Open the **Customize Columns** dialog box as described in *To resize all columns to best fit*.

2 Select the **Best Fit** check box and, additionally, select the **Auto Fit** check box.

Each column in the active editor window is then automatically adjusted to fit the text you type in a row cell or a **Definition** dialog box when you leave the column or dialog box respectively.

Or:

If you want to apply **Best Fit** and **Auto Fit** to all active editor windows, set the corresponding editor options described in *DDM Editor Options* in the *Using Natural Studio* documentation.

To resize a single column to fit the contents

- In the heading of the column you want to change, place the pointer over the right border. When the pointer changes to a divider, double-click on the border between the column headings. Note that you cannot resize the leftmost column.

  The column is automatically adjusted to fit its contents.

To resize a single column to a specific size

- In the heading of the column you want to change, place the pointer over the right border. When the pointer changes to a divider, drag the divider to the width you require. Note that you cannot resize the leftmost column.

  The width of the column has changed to the specified size.

To save a resized table layout

- Open the **Customize Columns** dialog box as described in *To resize all columns to best fit* and choose one of the following buttons.

  - **OK** saves the new table layout for the current editor session.
  - **Save Layout** saves the new layout in your user profile and retains it for future editor sessions.
  - **Restore Layout** overwrites the current layout with the layout previously saved in the user profile. Choose **OK** to save this layout.
  - **Restore Defaults** followed by **OK** overwrites the layout saved in the user profile with the default layout initially provided by the editor. Choose **OK** to save this layout.

  Or:

  In the editing area of the editor window, press CTRL+ALT+L.

  The new layout is saved in your user profile and retained for future editor sessions.

Moving Columns

You can change the table layout by moving single or multiple columns.

To move a column

1 Choose one of the following methods:

  - Select a field as described in ***Selecting Single Fields***.
  - Open the **Customize Columns** dialog as described in *To resize all columns to best fit*.
  - From the **Displayed Columns** list box, select the columns you want to move and choose **Move Up** or **Move Down** (if required repeatedly) until the columns have reached the target position.

    The top-to-bottom order of the list box corresponds to the left-to-right of the table in the editor window, that is, the top list column corresponds to the leftmost table column.

  Or:

- Drag the column heading you want to move and drop it in the position required. Note that you cannot move the leftmost column.
- Open the **Customize Columns** dialog box as described in *To resize all columns to best fit*.

2 To keep the new table layout, proceed as described in ***To save a resized table layout***.

Hiding or Displaying Columns

You can change the table layout by hiding or displaying columns.

To hide a column by rearranging the display order

1 Select a field as described in ***Selecting Single Fields***.

2 Open the **Customize Columns** dialog box as described in *To resize all columns to best fit*.

3 From the **Displayed Columns** list box, select the columns you want to hide.

The top-to-bottom order of the list box corresponds to the left-to-right of the table in the editor window, that is, the top list column corresponds to the leftmost table column.

Note: You cannot select **Type**, **Level**, **Name**, **Format** and **Length** which are mandatory for the table layout.

4 Choose **Remove**.

The selected columns are removed from **Displayed Columns** and appear in the **Hidden Columns** list box.

5 To keep the new table layout, proceed as described in ***To save a resized table layout***.

To hide a column by moving column borders

1 In the heading of the column you want to hide, place the pointer over the right border. When the pointer changes to a divider, drag the divider to the left border until the column heading is completely invisible (right and left border lines must coincide).

Note: You cannot hide the columns **Type**, **Level**, **Name**, **Format** and **Length** which are mandatory for the table layout.

The hidden column then appears in the **Hidden Columns** list box of the **Customize Columns** dialog box.

2 To keep the new table layout, proceed as described in ***To save a resized table layout***.

To display a hidden column

1 Select a field as described in *Selecting Single Fields*.

2 Open the **Customize Columns** dialog box as described in *To resize all columns to best fit*.

3 From the **Hidden Columns** list box, select the columns you want to display in the editor window.

4 Choose **Add**.

The selected columns are removed from **Hidden Columns** and appear in the **Displayed Columns** list box.

5 To keep the new table layout, proceed as described in *To save a resized table layout*.

Showing or Hiding Fields

You can show (expand) or hide (collapse) blocks of fields to improve readability and maintainability of DDMs with complex data structures. When a block of fields is collapsed, all fields contained in this block are hidden, including any other nested blocks if they are part of the chosen block. Hidden blocks retain their collapsed or expanded state.

Blocks that can be expanded or collapsed are blocks of fields that are defined for the same field level (1 to 99). Blocks are expanded or collapsed by the hierarchy of levels, from highest level 1 to lowest level 99. A block that contains fields from a lower-ranking level is contained in a block from a higher level.

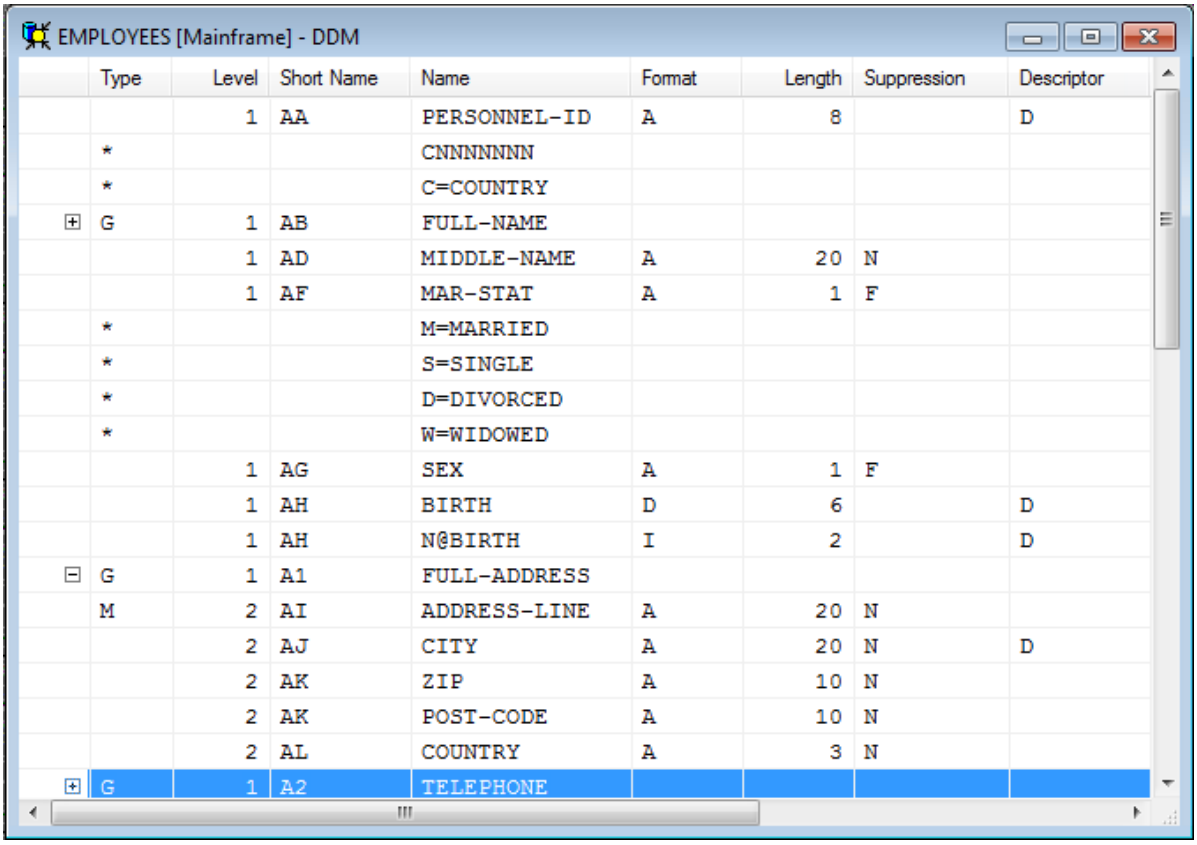
When scanning field definitions (see also *Finding and Replacing Fields*), collapsed blocks are also scanned.

If you want to expand and collapse blocks of fields, you need to set the appropriate editor option referenced in the instructions below.

To expand and collapse single blocks

1 Set the **Expand/Collapse** editor option as described in *DDM Editor Options* in the *Using Natural Studio* documentation.

When the **Expand/Collapse** option is set, an expand or a collapse toggle (⊞ or ⊟) appears as shown in the example below:



	Type	Level	Short Name	Name	Format	Length	Suppression	Descriptor
		1	AA	PERSONNEL-ID	A	8		D
	*			CNNNNNNN				
	*			C=COUNTRY				
⊞	G	1	AB	FULL-NAME				
		1	AD	MIDDLE-NAME	A	20	N	
		1	AF	MAR-STAT	A	1	F	
	*			M=MARRIED				
	*			S=SINGLE				
	*			D=DIVORCED				
	*			W=WIDOWED				
		1	AG	SEX	A	1	F	
		1	AH	BIRTH	D	6		D
		1	AH	N@BIRTH	I	2		D
⊟	G	1	A1	FULL-ADDRESS				
	M	2	AI	ADDRESS-LINE	A	20	N	
		2	AJ	CITY	A	20	N	D
		2	AK	ZIP	A	10	N	
		2	AK	POST-CODE	A	10	N	
		2	AL	COUNTRY	A	3	N	
⊞	G	1	A2	TELEPHONE				

The ⊞ toggle indicates the first row of a collapsed block.

The ⊟ toggle indicates the first row of an expanded block.

2 Click on a ⊞ toggle to expand a block or click on a ⊟ toggle to collapse a block.

Or:

Position the cursor in a row that contains a ⊞ or ⊟ toggle and, from the **View** menu, choose **Expand/Collapse** or choose the **Expand/Collapse** toolbar button.

Or:

Use any of the shortcut keys listed in *Shortcut Keys* in the *Using Natural Studio* documentation.

To expand or collapse all blocks

- From the **View** menu, choose **Expand All** or **Collapse All**.

  Or:

  Choose the **Expand All** or the **Collapse All** toolbar button.

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
- *EMU - Unicode Edit Mask* in the *Parameter Reference* documentation

The section below covers the following topics:

- Extended Field Attributes
- Tamino-Specific Extended Field Attributes
- SQL-Specific Extended Field Attributes
- Extended Field Attributes in a Remote Mainframe Environment

Extended Field Attributes

This section describes how to display and edit the extended attributes of the fields contained in the current DDM.

To display and edit extended field attributes

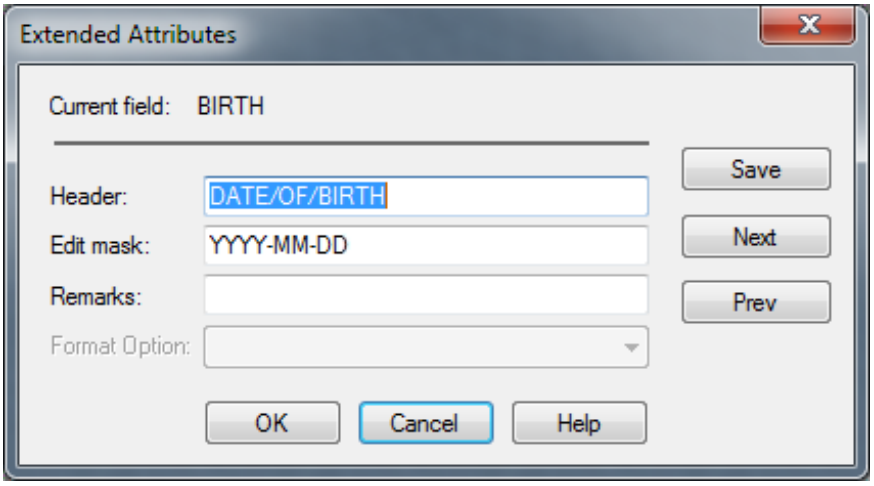
1 In the active DDM editor window, select a field.

2 From the **Field** or context menu, choose **Extended Attributes**.

Or:

Choose the **Extended Attributes** toolbar button.

The **Extended Attributes** dialog box appears with the name of the selected field as shown in the example below:

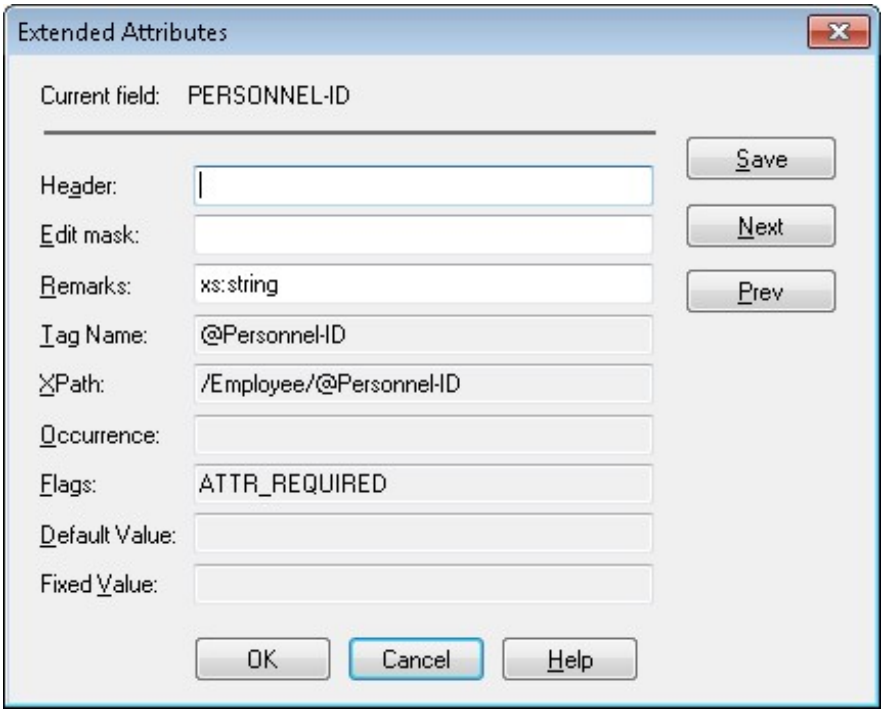


If a header exists for the selected field, it is displayed in the **Header** text box. You can edit the header, or add a header if none exists.

If an edit mask exists for the selected field, it appears in the **Edit mask** text box. You can modify the edit mask, or add an edit mask if none exists.

If a remark exists for the selected field, it appears in the **Remarks** text box. You can edit the remark, or add a remark if none exists.

For Tamino, **Tamino-specific extended field attributes** are displayed as shown in the example below:



3 Choose **Save** to save and validate any changes you have made for the current field.

Choose **Next** to view and edit extended attributes for the next field in the DDM source.

Or:
Choose **Prev** to view and edit extended attributes for the previous field in the DDM source.

(Commentary fields identified with an asterisk (*) are skipped.)

4 Choose **OK** to save and validate all field modifications.

The DDM editor window appears.

Tamino-Specific Extended Field Attributes

Tamino-specific extended field attributes are extracted from Tamino XML schema definitions.

In addition to the text boxes **Header**, **Edit Mask** and **Remarks**, the following *read-only* Tamino-specific attributes are displayed in the **Extended Attributes** dialog box:

<table>
<tr><th>Attribute</th><th>Function</th></tr>
<tr><td>Tag Name</td><td>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 <code>Tag Name</code>.</td></tr>
<tr><td>XPath</td><td>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.</td></tr>
<tr><td>Occurrence</td><td>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 <code>maxOccurs</code> facet in the Tamino XML schema.</td></tr>
<tr><td>Flags</td><td>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:
<table>
<tr><td></td><td></td></tr>
<tr><td><code>ARRAY</code></td><td>Field is an array; that is, <code>maxOccurs</code> is greater than 1.</td></tr>
<tr><td></td><td></td></tr>
<tr><td><code>GROUP_ATTRIBUTES</code></td><td>Field is a group that contains the attribute sub-fields of the predecessor field.</td></tr>
<tr><td></td><td></td></tr>
<tr><td><code>GROUP_ALTERNATIVES</code></td><td>Field is a group that represents the choice constructor; the choice elements are contained as sub-fields.</td></tr>
<tr><td></td><td></td></tr>
<tr><td><code>GROUP_SEQUENCE</code></td><td>Field is a group that represents the sequence constructor; the sequence elements are contained as sub-fields.</td></tr>
<tr><td></td><td></td></tr>
<tr><td><code>GROUP_ALL</code></td><td>Field is a group that represents all constructors; all elements are contained as sub-fields.</td></tr>
<tr><td></td><td></td></tr>
<tr><td><code>ATTR_REQUIRED</code></td><td>Field is an attribute marked as required.</td></tr>
<tr><td></td><td></td></tr>
<tr><td><code>ATTR_OPTIONAL</code></td><td>Field is an attribute marked as optional.</td></tr>
<tr><td></td><td></td></tr>
<tr><td><code>ATTR_PROHIBITED</code></td><td>Field is an attribute marked as prohibited.</td></tr>
</table></td></tr>
</table>

Attribute	Function	
	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 text boxes **Header**, **Edit Mask** and **Remarks**, the following read-only SQL-specific attribute is displayed in the **Extended Attributes** dialog box:

Attribute	Function
SQLTYPE	Information generated from the data types BLOB (Binary Large Object) or CLOB (Character Large Object) if contained in an Oracle database.

Extended Field Attributes in a Remote Mainframe Environment

Only applies to DDMs generated from VSAM files.

This section describes how to display and edit the extended field attributes of a DDM generated from a VSAM file.

Related Topic:

Extended Editing at Field Level - Natural for VSAM documentation.

To display and edit extended field attributes

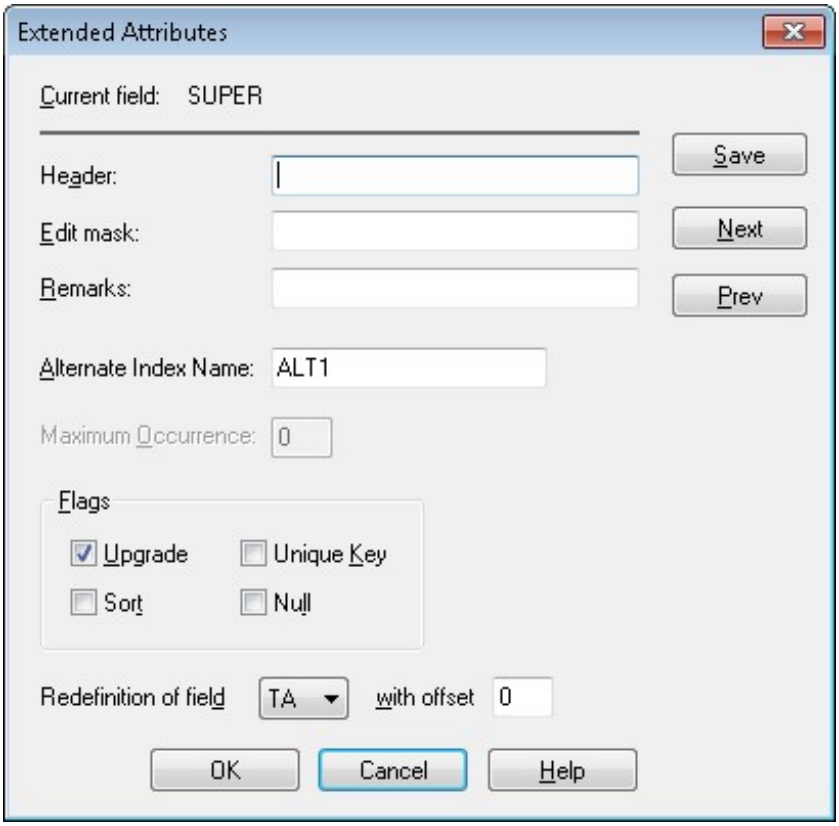
1 In the active DDM editor window, select a field.

2 From the **Field** menu, choose **Extended Attributes**.

Or:

Choose the **Extended Attributes** toolbar button.

The **Extended Attributes** dialog box appears with the name of the selected (current) field indicated at the top of the window as shown in the example below:



3 If a header exists for the selected field, it is displayed in the **Header** text box. You can modify the header, or add a header if none exists.

If an edit mask exists for the selected field, it appears in the **Edit mask** text box. You can modify the edit mask, or add an edit mask if none exists.

If a remark exists for the selected field, it appears in the **Remarks** text box. You can modify the remark, or add a remark if none exists.

If an alternate descriptor (Type A) or superdescriptor (Type X) is defined for the field, you can enter an alternative index name.

If the field is a multiple or periodic group field, you can specify the number of occurrences in the **Maximum Occurrence** text box.

If an alternate descriptor (Type A) or superdescriptor (Type X) is defined for the field, you can set the flags **Upgrade**, **Unique Key**, **Sort** and **Null**.

If the field has a primary or secondary key descriptor (Type A) or superdescriptor (Type X), you can select the field short name from the **Redefinition of field** combo box.

4 Choose **Save** to save and validate any changes you have made for the current field.

Choose **Next** to view and edit extended attributes for the next field in the DDM source.

Or:
Choose **Prev** to view and edit extended attributes for the previous field in the DDM source.

(Commentary fields identified with an asterisk (*) are skipped.)

5 Choose **OK** to save and validate all field modifications.

The DDM editor window appears.

Displaying Descriptor Information

Only applies to Adabas.

This section describes how to display the definition of a descriptor of type `H` (hyperdescriptor), `P` (phonetic descriptor) or `S` (subdescriptor or superdescriptor). For further information on descriptors, see **Descriptor** in the section *Columns of Field Attributes*.

To display descriptor definition

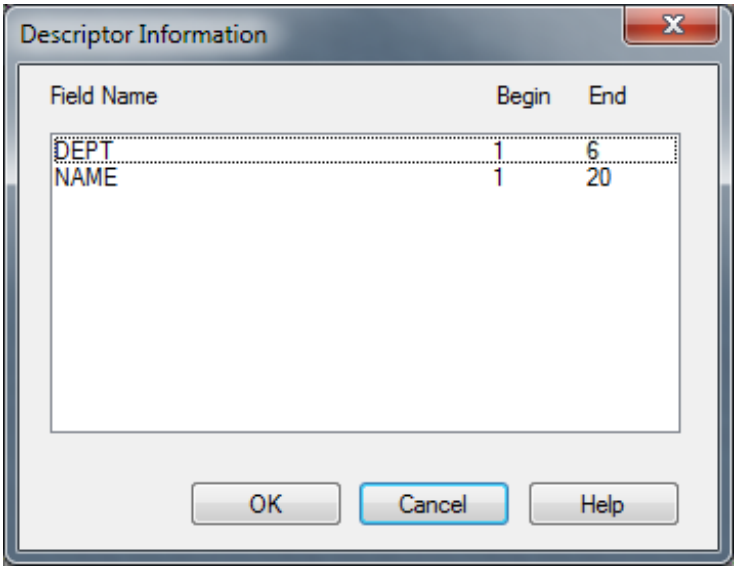
1 In the DDM editor window, select the row or row cell that contains `S`, `P` or `H`.

2 From the **Field** or context menu, choose **Descriptor Information**.

Or:

Choose the **Descriptor Information** toolbar button.

The **Descriptor Information** dialog box appears as shown in the example below:



The definition of the descriptor is displayed (in the example above: `DEPT` and `NAME`) next to **Field Name** and the field offsets are displayed next to **Begin** and **End**.

3 Choose **OK** to exit the **Descriptor Information** dialog box.

21 Saving and Cataloging a DDM

- Additional Options for VSAM Files 202

You can save a DDM as a source object and/or a cataloged object (generated program) in the specified Natural library (if applicable) in the current Natural system file (see also *Storing DDMs - FDDM System File*).

In a remote environment on a mainframe platform, you can only save a DDM as a cataloged object.

For the naming conventions that apply to an object, refer to *Object Naming Conventions* in the *Using Natural Studio* documentation.

To save a DDM

- Proceed as described in *Saving Objects* in the *Using Natural Studio* documentation.

To save and catalog a DDM

- Proceed as described in either *Stowing Objects* or *Cataloging Objects* in the *Using Natural Studio* documentation.

The additional information you can specify for a DDM generated from a VSAM file are described in *Additional Options for VSAM Files*.

Additional Options for VSAM Files

Only applies to a remote environment on a mainframe platform.

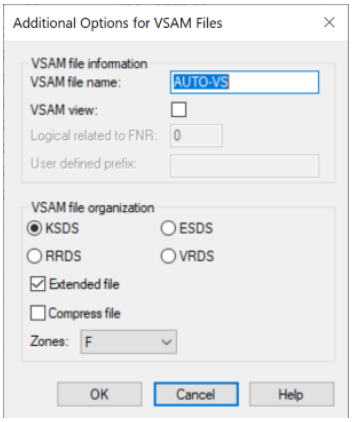
The DDM editor provides the option to specify additional information for DDMs generated from VSAM files.

The additional options for VSAM files can be specified when saving a DDM source as a cataloged object. The additional options consist of two parts: VSAM file information and VSAM file organization.

To specify additional options for DDMs from VSAM files

1 Catalog a DDM source generated from a VSAM file.

The **Additional Options for VSAM Files** dialog box appears as shown in the example below:



2 In the **VSAM file name** text box, enter the DDNAME/FCT entry as defined to the TP monitor or when using batch mode.

3 If the **VSAM view** check box is set, this DDM represents a logical DDM. If the check box is not set, it represents a physical DDM.

If the **VSAM view** check box is set:

- In the **Logical related to FNR** text box, enter the file number of the physical DDM from which the logical file or DDM is derived.
- In the **User defined prefix** text box, enter the prefix value to be assigned to the logical file.

4 In the **VSAM file organization** section, set the type of the VSAM file by selecting one of the option buttons.

5 Set the **Extended file** check box if the extended VSAM file (8 byte ISN) is to be used.

6 Set the **Compress file** check box if the file is to be compressed.

7 From the **Zones** combo box, select the zone for the VSAM file. `F` indicates that all packed data is written to the VSAM file with the zone `X'0F'`. `C` indicates that all packed values are written to the VSAM file with the zone `X'0C'`.

For details on the options displayed in the **Additional Options for VSAM Files** dialog box, refer to *Natural File Access* in the *Natural for VSAM* documentation.

22 Listing DDMs

- Listing DDMs in a Workspace 206
- Listing DDMs with LIST 206

This section provides instructions for displaying a list of DDMs either in the Library Workspace or Application Workspace of Natural Studio or by using the Natural system command `LIST`.

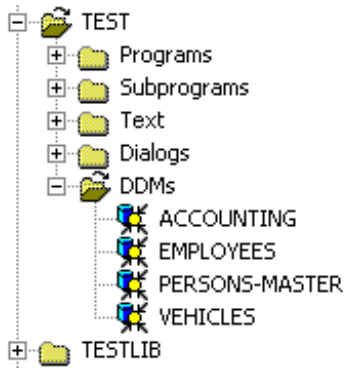
Listing DDMs in a Workspace

The section below describes how to display a list of DDMs in the Library Workspace or Application Workspace.

To list all DDMs in a workspace

- In the Logical View, from the node User Libraries or System Libraries, expand the DDMs subnode or node.

  A list of all DDMs available is displayed as shown in the example of the library TEST below:



Listing DDMs with LIST

The Natural system command `LIST` is used to list all DDMs available in the current Natural library, the steplibs and the FNAT and FUSER system files.

To list all DDMs with `LIST`

- In the command line of Natural Studio, enter the following:

```
LIST VIEW *
```

  A window 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.

23 Maintaining DDMs in Different Environments

This section briefly describes the Natural utilities that can be used, in addition to the clipboard functions provided with Natural Studio.

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).

24 Data Conversion for Adabas or RDBMS

- Adabas 212
- Adabas D 212
- Db2 213
- Oracle 214
- Microsoft SQL Server 215
- MySQL 216
- PostgreSQL 217
- Related Topics 217

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

25 Data Conversion for Tamino

- Built-In Tamino XML Schema Language Data Types 220
- Tamino XML Schema Constructors 222
- Multiplicity in Tamino XML Schema Language 223

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 Derived Data Type

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.

VI Dialog Editor

This part covers the following topics:

General Information
Dialog Editor Window
Editing Dialogs
Dialog Wizard
Creating Dialog Elements
Importing Data Fields
Editing Dialog Elements
Attributes Windows for Dialogs and Dialog Elements
Dialog Boxes
Enhanced Source Code Format

See also the following sections in the *Using Natural Studio* documentation:

- *Setting the Options* and *Dialog Editor Options*
- *Shortcut Keys* and *Dialog Editor Shortcut Keys*

26 General Information

A single dialog is not only an isolated Natural object like a map or a program but can also represent an entire event-driven application. The dialog editor can be used to create an application with the following basic components:

- Dialog(s)
- Dialog elements
- Attributes
- Event handlers
- Data areas (local and parameter); global data areas can be referenced
- Inline subroutines

For a reference description of dialogs, dialog elements, attributes and event handlers, see the *Dialog Component Reference*.

For an overview of dialog editor terminology, see *Introduction to Event-Driven Programming* in the *Programming Guide*.

You can open a new dialog editor window from the Natural base window by choosing "Object > New > Dialog". Alternatively, you can edit an existing dialog by selecting it from the "Library Workspace" window.

Menus, toolbar buttons, and commands available with the dialog editor can be used to create the components of an event-driven application and edit them in various editor windows. You can create or edit another dialog, or invoke a different editor and create or edit a different type of object (for example, program, DDM or data area).

You can set editor preferences by using the editor options described in *Dialog Editor Options* and *Setting the Options* in the *Using Natural Studio* documentation.

You can open windows and dialog boxes and perform editor functions by using dialog editor shortcut keys and generally available shortcut keys described in the section *Shortcut Keys* in the *Using Natural Studio* documentation.

27 Dialog Editor Window

- Changing the Initial Position of the Dialog 230
- Changing the Initial Size of the Dialog 231
- Selecting/Deselecting Dialog Elements 231
- Aborting Mouse Operations 232
- Creation Mode in Map Editor and Dialog Editor 232
- Changing the Position of a Dialog Element 232
- Changing the Size of a Dialog Element 233
- Moving the Pointer 233
- Simulating the Mouse with the Spacebar 234
- Scrolling in a Dialog 234
- Using the Clipboard 235

The dialog editor window includes a title bar, an information bar below the title bar, and a status line.

The title bar includes the name of the dialog (or "Untitled" if the dialog and the library have not been named). For example:

```
MYDIALOG [MYLIB] - Dialog
```

The information bar below the title bar contains the following information:

Item	Explanation
Status	Indicates whether the dialog has been modified since it was saved.
Selected (handle)	Indicates the handle name of the currently selected dialog element; the selection box displays the handle names of all dialog elements in the dialog together with their level number in the dialog element hierarchy. You can select another dialog element in the selection box.
x	X axis position of the currently selected dialog element relative to the upper left corner of the client area of the parent dialog element (or dialog, for top-level dialog elements). Equivalent to the current value of the `RECTANGLE-X` attribute.
y	Y axis position of the currently selected dialog element relative to the upper left corner of the client area of the parent dialog element (or dialog, for top-level dialog elements). Equivalent to the current value of the `RECTANGLE-Y` attribute.
w	Width of the currently selected dialog element. Equivalent to the current value of the `RECTANGLE-W` attribute.
h	Height of the currently selected dialog element. Equivalent to the current value of the `RECTANGLE-H` attribute.

Dialogs that are larger than the area shown in the dialog editor window can be scrolled using the scroll bars on the right and at the bottom of the dialog editor window.

The following topics are covered below:

Changing the Initial Position of the Dialog

To change the initial position of the dialog

- Select its title bar and drag it to the desired location.

  Or:

  Open its attributes window and type in the new coordinates (in pixels) in the "X" and "Y" fields.

To open the attributes window

- From the **Dialog** menu or from the dialog's context menu, choose **Attributes**.

  Or:

  Select the dialog and press ENTER.

Changing the Initial Size of the Dialog

To change the initial size of the dialog

- Use the sizing border of the dialog.

  Or:

  Open its attributes window and type in the new size (in pixels) in the "W" and "H" fields.

Selecting/Deselecting Dialog Elements

It is possible to select multiple dialog elements, but only one can be active at any time. The active selection is delineated by black selection marks using which the control can be resized. The inactive selection is delineated by grey selection marks.

Dialog Editor commands which are based on a single dialog element use the active selection, whereas other (such as **Delete**) use both the active and inactive selection. Selecting a dialog element which is part of the inactive selection makes it the active selection without deselecting any other dialog element.

To select a dialog element

- Select an unselected dialog element, which becomes selected, while all other dialog elements become deselected. To select an additional dialog element, hold down SHIFT and select the dialog element. The dialog element selected last becomes the active selection, the ones selected before are the inactive selection. To deselect the dialog element(s), select the blank space in the dialog window.

  Or:

  Point to the background in the dialog window and then drag the pointer to enclose or partially enclose the elements you want to select. To deselect or select additional elements, do the same as above while pressing SHIFT.

Or:

Press TAB to select the next dialog element in the control sequence.

Or:

Press SHIFT+TAB to select the previous dialog element in the control sequence.

Or:

Select a dialog element from the drop-down list in the status bar. You can do this by using the mouse or by pressing F6 to switch to the drop-down list box and then using the arrow keys to select the dialog element. To drop down the list and view the dialog elements to choose from, press F4. To deselect the drop-down list box, press ESC or ENTER.

Note: If one or more dialog elements are already selected, you can only *additionally* select other controls with the same parent.

Aborting Mouse Operations

Any operation that is completed by releasing the left mouse button may be aborted by pressing the ESC key before releasing the left mouse button.

Creation Mode in Map Editor and Dialog Editor

If you create a dialog element by selecting "Insert" plus the dialog element type, the dialog editor is in "creation mode". After creation, the dialog editor is no longer in creation mode; that is, you do not have to switch off creation mode by selecting the dialog element as you would in the map editor.

Changing the Position of a Dialog Element

To change the position of one or more dialog elements, select the dialog element(s). The name of the dialog element selected first will be displayed in the status bar, together with its current position.

To change position, you can use one of the following options:

1 Drag the dialog element to its new location using the mouse.

2 For each selected dialog element, open its attributes window and type in the new coordinates (in pixels) in the "X" and "Y" fields.

3 Hold down SHIFT and press any arrow to move the selected dialog element(s) the number of pixels specified with the grid settings in the **Dialog Editor** dialog box described in *Dialog Editor Options* and *Setting the Options* in the *Using Natural Studio* documentation.

4 Hold down SHIFT+CTRL and press any arrow to move the selected dialog element(s) by one pixel.

Changing the Size of a Dialog Element

To change the size of one or more dialog elements, select the dialog element(s). You then have the following options:

1 Point to one of the eight small black squares (the selection mark of the last selected dialog element). The mouse pointer now indicates the direction into which you can resize the dialog element. Hold down the left mouse button and drag (one of) the dialog element(s) to the desired size. If more than one dialog element is selected, the other dialog elements selected are resized proportionally.

2 Open the dialog element's attributes window and type in the new size (in pixels) in the "W" and "H" fields.

3 Choose "Control > Stretch", then the direction into which you can resize the dialog element. Then use the mouse or the keyboard to continue the operation.

Moving the Pointer

To move the pointer

- Move the mouse.

  Or:

  Press any arrow key to move the pointer by the number of pixels specified with the grid settings in the **Dialog Editor** dialog box described in *Dialog Editor Options* and *Setting the Options* in the *Using Natural Studio* documentation.

  Or:

  Hold down CTRL and press any arrow key to move the pointer by one pixel.

Simulating the Mouse with the Spacebar

You can simulate mouse operations with the spacebar as described in the following table. Note that the pointer must lie on the element to be manipulated.

Mouse Operation	Keyboard Operation
Press left mouse button	Press and hold down the spacebar.
Release left mouse button	Release the spacebar.
Mouse click	Press and release the spacebar.
Mouse, double-click	Press and release the spacebar twice.
Move dialog element	Move pointer to element, press and hold down the spacebar, press the appropriate arrow key(s).
Select several dialog elements	Move pointer to background, press and hold down the spacebar, press the appropriate arrow key(s).
Resize dialog element	Move pointer to any black square of selected element, press and hold down the spacebar, press the appropriate arrow key(s), release the spacebar.

Simulating a mouse double-click with the spacebar opens the attributes window for the dialog element on which the pointer is positioned; if the pointer is not positioned on any dialog element, the dialog attributes window is opened.

Scrolling in a Dialog

You can scroll in a dialog window if at least one dialog element is outside its scroll range. For you to be able to scroll in a dialog, the dialog scroll bars must be active. To activate the dialog scroll bars, open the dialog attributes window either by pressing ENTER or by double-clicking in the dialog. Then select either the "Horizontal Scrollbar" or "Vertical Scrollbar" entry.

To scroll with the mouse

- Point to the scroll-bar slider and drag the slider in the desired direction.

  Or:

  Point to the scroll-bar shaft and select.

  Or:

  Point to one of the scroll-bar arrow buttons and hold down the left mouse button.

To scroll with the keyboard, you do not need a scroll bar. You have four options:

1. To simulate clicking into a vertical scroll bar, press the PAGE-UP or PAGE-DOWN keys; or
2. To simulate clicking into a horizontal scroll bar, press SHIFT+PAGE-UP or SHIFT+PAGE-DOWN; or
3. To simulate clicking on the corresponding vertical arrow button, press CTRL+PAGE-UP or CTRL+PAGE-DOWN; or
4. To simulate clicking on the corresponding horizontal arrow button, press CTRL+SHIFT+PAGE-UP or CTRL+SHIFT+PAGE-DOWN.

Using the Clipboard

Key (Combination)	Function
DEL	Delete the selected dialog element
CTRL+C	Copy
CTRL+V	Paste

28 Editing Dialogs

- Editing a Dialog's Source Code 238
- Editing a Dialog's Attributes 239
- Editing a Dialog's Event Handlers 239
- Defining a Dialog's Menu Bar 240
- Defining a Dialog's Toolbar 240
- Creating and Maintaining Timers for a Dialog 241
- Creating and Maintaining Signals for a Dialog 241
- Creating and Maintaining Context Menus 242
- Creating and Maintaining Wallpapers 243
- Adding a Comment Section to a Dialog 243
- Defining a Parameter or Local Data Area for a Dialog 244
- Selecting a Global Data Area for a Dialog 244
- Defining an Inline Subroutine for a Dialog 245
- Defining the Control Sequence in a Dialog 245

The following topics are covered below:

Editing a Dialog's Source Code

To edit a dialog's source code

1 Load the dialog into the editor.

2 From the **Dialog** menu, choose **Source Code**.

Or:

Press CTRL+ALT+C.

The dialog's source code window appears and the program editor is loaded. This editor enables you to scan for text strings, replace them, and so on.

You can switch between the dialog editor and the program editor by selecting the source code window or the dialog window. If you edit in either window, you need to synchronize your updates: (graphically) modifying the dialog locks the source code window and you may not make changes there. Correspondingly, if you change the source code, you may not make changes in the dialog window, which is locked. If your editor is locked, its status bar displays "Locked".

If a source code window is open, but not active, you can activate it by choosing **Source Code** from the **Dialog** menu.

When you issue a command from the program editor window that affects the source code, such as **Save** or **Run**, the dialog editor updates itself automatically by scanning the source code, displaying the modified dialog, and then regenerating the source code. When you issue a command from the dialog editor window after you have modified the code in the source code window, you are prompted whether you want to update the source code or not.

3 To stow any modified source code: from the program editor's **Object** menu, choose **Stow**.

Whenever you want to save a dialog under a new name, select **Save as** from the **Object** menu, where for the **Save dialog as** dialog box appears.

Editing a Dialog's Attributes

To edit a dialog's attributes

1 Load the dialog into the editor.

2 From the **Dialog** menu or from the dialog's context menu, choose **Attributes**.

Or:

Double-click on the dialog.

Or:

Press ENTER.

The dialog's attributes window appears. To find out what the entries in the attributes window mean, choose "Help". For context-sensitive help, select the attribute entry and press F1.

3 Enter the desired attribute values.

4 Choose **OK** to confirm your changes.

Editing a Dialog's Event Handlers

To edit a dialog's event handlers

1 Load the dialog into the editor.

2 From the **Dialog** menu or from the dialog's context menu, choose **Event handlers**.

Or:

Press CTRL+ALT+E or SHIFT+ENTER.

The dialog's event handler section appears.

3 Select the type of event (such as BEFORE-OPEN or ERROR).

Or:

Choose "New" to enter a user-defined event.

Or:

Choose "Rename" to save a user-defined event with a new name.

4 Enter the desired event code in free form either in the edit window in the **Dialog Event Handler** window itself, or using the Program Editor. To use the Program Editor, select the **Editor** push button, then close the **Dialog Event Handler** window using the **OK** push button. This code will be executed when the event occurs for the dialog. Note that if you have specified code in the before-any and after-any event sections, this will be triggered before and after the code entered here. So if you need common event code, you only have to enter it once in your dialog's before-any and after-any event section.

5 Choose **OK** to save your code, if using the **Dialog Event Handler** window. If using the Program Editor, the code can be saved by choosing **Save** from the **Object** menu, or by closing the Program Editor and selecting to save the changes when prompted.

Defining a Dialog's Menu Bar

To define a dialog's menu bar

1 Load the dialog into the editor.

2 From the **Dialog** menu, choose **Menu bar**.

Or:

Press CTRL+ALT+M.

A dialog box appears asking you whether you want to turn the dialog's menu bar setting on.

3 Choose **Yes**.

A blank default menu bar is added to the dialog and the menu bar's attributes window appears. For more information on the attributes window, see the section *Menu Editor Window*.

4 Choose **OK** to confirm your changes.

Defining a Dialog's Toolbar

To define a dialog's toolbar

1 Load the dialog into the editor.

2 From the **Dialog** menu, choose **Toolbar**.

Or:

Press CTRL+ALT+T.

A dialog box appears asking you whether you want to turn the dialog's toolbar setting on.

3 Choose **Yes**.

A blank default toolbar is added to the dialog and the toolbar's attributes window appears. For more information on the attributes window, see the section *Toolbar Control Attributes Window*. In this section you will also find information on new toolbar control features.

4 Choose **OK** to confirm your changes.

Creating and Maintaining Timers for a Dialog

You use timers to trigger a dialog event periodically.

To create and maintain a timer for a dialog

1 Load the dialog into the editor.

2 From the **Dialog** menu, choose **Timers**.

Or:

Press CTRL+ALT+I.

The timer's attributes window appears. For more information on the attributes window, see the section *Timer Attributes Window*.

3 Choose **OK** to confirm your changes.

Creating and Maintaining Signals for a Dialog

You use signals to efficiently implement user commands, allowing their re-use by multiple user interface elements (such as menu or toolbar items).

To create and maintain signals for a dialog

1 Load the dialog into the editor.

2 From the **Dialog** menu, choose **Signals**.

Or:

Press CTRL+ALT+N.

The signal attributes window appears. For more information on the attributes window, see the section *Signal Attributes Window*.

3 Choose **OK** to confirm your changes.

4 To associate a menu or toolbar item with the signal, select the signal from the list presented in the **Same as** field for the item in the menu editor or toolbar attributes window. The menu or toolbar item will then inherit the attributes from the signal. Furthermore, when the menu or toolbar item is clicked on running the dialog, the signal's `CLICK` event is invoked.

Creating and Maintaining Context Menus

You use context menus to define context-specific menus that can be associated with the dialog and/or any number of dialog elements within it. For more information, see the section *Defining and Using Context Menus*.

To create and maintain context menus

1 Load the dialog into the editor.

2 From the **Dialog** menu, choose **Context Menus**.

Or:

Press CTRL+ALT+X.

The context menus window appears, which displays a list of context menus currently defined for the dialog. For more information on this window, see the section *Dialog Context Menus Window*. To edit the menu items for a context menu listed in this window, select the respective context menu in the list, then select the **Edit** pushbutton to invoke the **Menu Editor Window**.

3 Choose **OK** to confirm your changes.

4 To associate the context menu with the dialog or a dialog element within the dialog, open the corresponding attributes window (e.g. by double-clicking on the dialog or dialog element) and select the context menu from the list presented in the **Context Menu** field. Note, however, that not all dialog element types support context menus.

Creating and Maintaining Wallpapers

You use wallpapers to define background images that can be associated with the dialog and/or any number of dialog elements within it.

To create and maintain wallpapers

1 Load the dialog into the editor.

2 From the **Dialog** menu, choose **Dialog Wallpapers**.

Or:

Press CTRL+ALT+W.

The wallpaper attributes window appears. For more information on the attributes window, see the section *Wallpaper Attributes Window*.

3 Choose **OK** to confirm your changes.

4 To associate the wallpaper with the dialog or a dialog element within the dialog, open the corresponding attributes window (e.g. by double-clicking on the dialog or dialog element) and select the wallpaper from the list presented in the **Wallpaper** field. Note, however, that not all dialog element types support wallpapers.

Adding a Comment Section to a Dialog

To add a comment section to a dialog

1 From the **Dialog** menu, choose **Dialog comment**.

Or:

Press CTRL+ALT+O.

The dialog comment section appears where you can enter your comments in free form. Please note that you do not have to use the "/*" notation when entering comments in the text area. If you are listing your dialog code, you will find your comment at the beginning.

2 Choose **OK** to save your comment.

Defining a Parameter or Local Data Area for a Dialog

To define a local data area for a dialog

1 From the **Dialog** menu or from the dialog's context menu, choose **Local Data Area**.

Or:

Press CTRL+ALT+L.

The definition section for the local data area appears. In a local data area, you must include all the user-defined variables or other variables that you want to use in an event handler code section or a subroutine of the current dialog. Note that the dialog editor automatically generates the data definitions for the dialog elements.

The **Using** button opens a dialog box that enables you to include existing inline data definitions.

2 Choose **OK** to save your data definition.

To define a parameter data area for a dialog

1 From the **Dialog** menu, choose **Parameter Data Area**.

Or:

Press CTRL+ALT+P.

The definition section for the parameter data area appears. In a parameter data area, you must include all the parameters that you want to be passed on to the current dialog in an `OPEN DIALOG` or `SEND EVENT` statement.

The **Using** button opens a dialog box that enables you to include existing inline data definitions.

2 Choose **OK** to save your data definition.

Selecting a Global Data Area for a Dialog

To select a global data area for a dialog

1 From the **Dialog** menu, choose **Global Data Area**.

Or:

Press CTRL+ALT+G.

A dialog box appears where you can select a global data area for the dialog.

2 Select an entry in the **Available Global Data Areas** list box.

3 Choose **OK**.

Defining an Inline Subroutine for a Dialog

To define an inline subroutine for a dialog

1 Load the dialog into the editor.

2 From the **Dialog** menu or from the dialog's context menu, choose **Inline Subroutines**.

Or:

Press CTRL+ALT+S.

The "Dialog inline subroutines" code section appears.

3 Choose "New" to enter a new subroutine.

Or:

Select the name of an existing subroutine you want to edit.

If you have chosen "New", a dialog box prompts you for a name.

4 Enter the name of the new subroutine.

5 Choose **OK**.

6 Enter the desired subroutine code in free form, either directly in the window itself or using the Program Editor by selecting the **Editor** push button, then close the window using the **OK** push button.

7 Choose **OK** to save your code.

Defining the Control Sequence in a Dialog

The control sequence is the keyboard navigation sequence in which the end user will go through the dialog elements.

To define the control sequence of a dialog

1 From the **Dialog** menu, choose **Control sequence**.

Or:

Press CTRL+ALT+Q.

The control sequence is displayed as a number at the top left corner of each dialog element. The editor is now in navigation sequence definition mode.

2 Use the mouse to select the dialog elements in the desired sequence.

If you *do not* select a dialog element before enabling navigation sequence definition mode, the next dialog element that you select will be the first in the navigation sequence. Its number is greyed and you can select the next dialog element in the sequence, and so on.

If you *do* select a dialog element, you can redefine the sequence from this element onwards. You can also select a dialog element when in control sequence editing mode *without* resequencing it by holding down the SHIFT key whilst making the selection. This is especially useful if the selected dialog element is one of the last elements in the sequence - you do not have to redefine the sequence of all preceding dialog elements. Note that instead of selecting each dialog element with the mouse, you can also select them from the selection box in the status bar of the dialog editor. This selection box always shows the dialog elements in their control sequence.

You can exit control sequence editing mode implicitly, by selecting another command (e.g. 'Insert Push Button') or explicitly by selecting the 'control sequence' menu of this again, or by pressing ESC.

Note: The control sequence also decides the order in which the dialog elements overlap.

29 Dialog Wizard

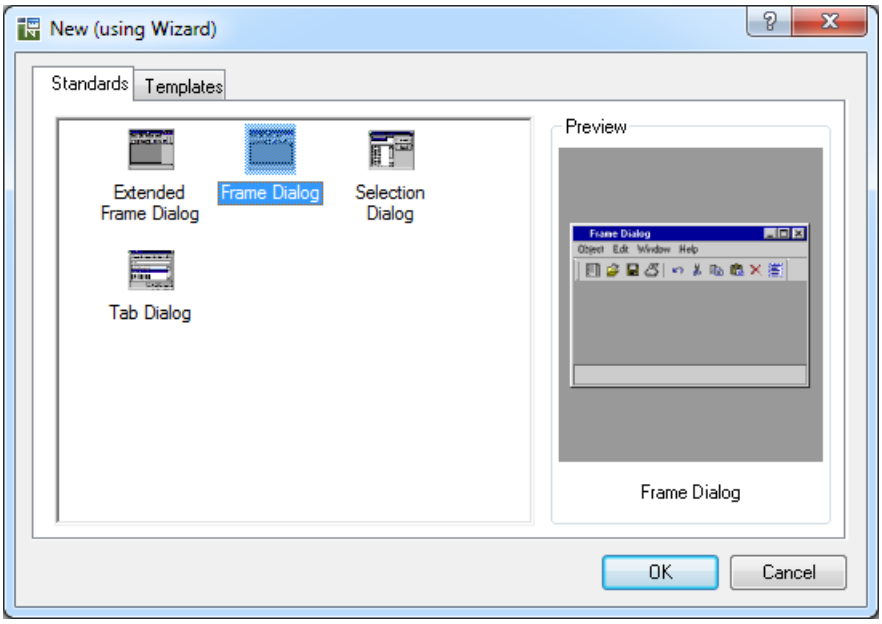
- Dialog Types (Standards) 249
- How to create a template 252

The Dialog Wizard is a tool for creating dialogs for specific purposes. The defined dialogs can have several layouts that adapt to desired requirements.

The generated dialog can be modified with the dialog editor. In the dialog there are User Code Sections with sample coding for data retrieval and result flagging. These sections should be replaced by user-specific requirements. If not suppressed by the user, the generated dialogs contain action events showing the intended functionality and destination of the dialog. Run the dialog from the dialog editor and test the various actions.

To activate the Wizard

- From the **Object** menu, choose **New > Dialog Wizard**.



There are two areas to select a desired dialog type. The standards tab offers the four standard types of dialogs with a basic setting. The templates tab offers several specifically predefined dialog definitions. For each item a context menu and a properties window can be called for different views (see below).

This chapter covers the following topics:

Dialog Types (Standards)

Frame Dialog

In the Frame Dialog Wizard a new dialog can be created in a Frame Layout. The structure of the Frame Dialog for example is applicable in an application frame.

A default Frame Dialog is generated if you define nothing in the wizard.

Extended Frame Dialog

In the Extended Frame Dialog Wizard a new dialog can be created in a Frame Layout with additionally up to four dialog bar controls. The structure of the Frame Dialog for example is applicable in an application frame. The dialog bars may contain controls for a Notepad or for item selection. A default Extended Frame Dialog is generated if you define nothing in the wizard.

Selection Dialog

In the Selection Dialog Wizard a new dialog can be created in a Selection Layout. The structure of a Selection Dialog for example is applicable for reading, saving or opening objects.

A default Selection Dialog is generated if you define nothing in the wizard.

Tab Dialog

In the Tab Dialog Wizard a new dialog can be created in a Tab Layout. The structure of a Tab Dialog for example is applicable in a help dialog or for option settings.

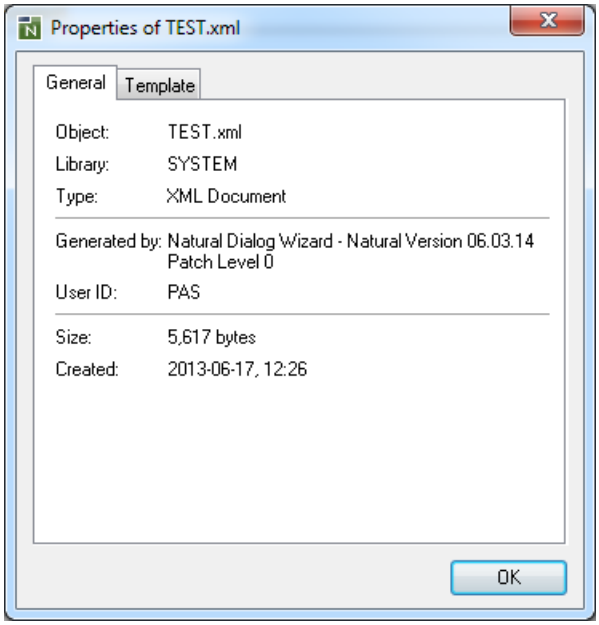
A default Tab Dialog is generated if you define nothing in the wizard.

Dialog Templates

A template can contain any type of dialog (as documented above). Using templates enables the customer to apply his own standards and base layouts for the generated dialogs. The templates are created as a result of a previous Dialog Wizard session, either by yourself or by another one. The templates tab lists all templates of the current library resource folder and the appropriate steplib resource folders.

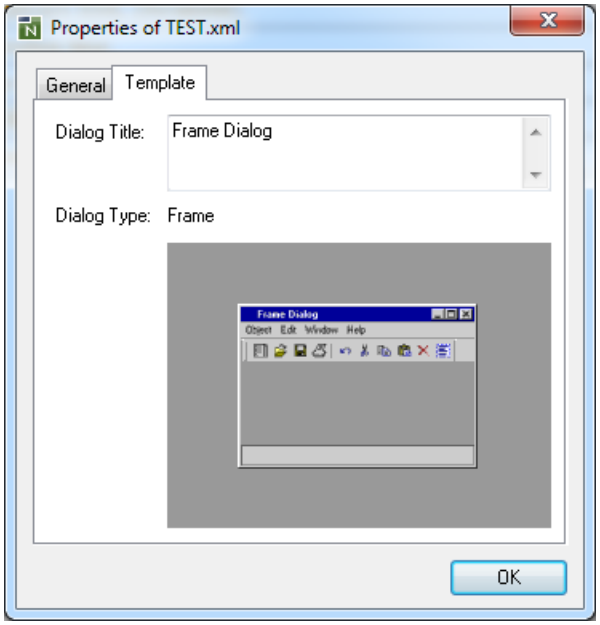
Dialog Item Properties

The properties of the selected dialog item (standard/template) is displayed as far as applicable.



The tab with the general properties shows:

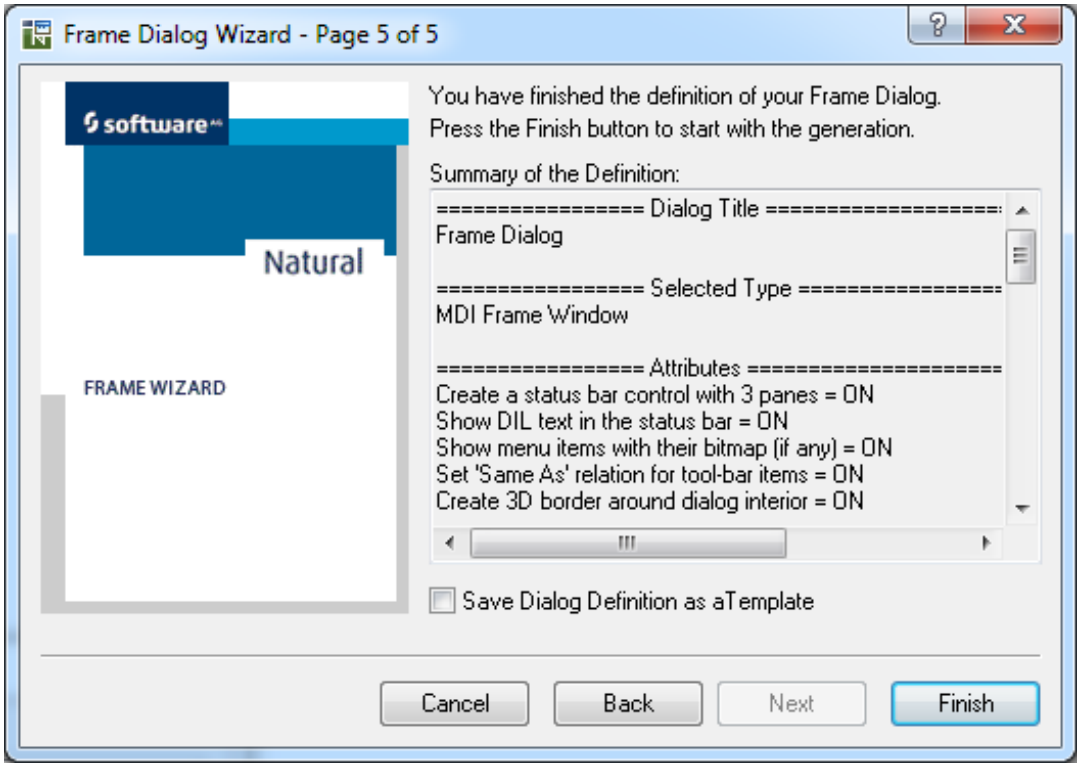
Object	File name of the template.
Library	Library where the resource folder can be found.
Type	Shows the file extension (always XML Document).
Generated by	Natural version used at creation time.
User ID	ID of the user creating the template.
Size	Byte size of the template file.
Created	Date and time when the template was created.



The tab with the template properties shows:

Dialog Title	Shows the title of the dialog as displayed in the title line.
Dialog Type	Dialog type as documented in the Standards.

How to create a template



On the last page of the dialog wizard a check box can be checked to force the saving of the dialog in a template file after you have completed the definition of the dialog. A standard windows save dialog is opened where a file name can be defined. Independently of the created template, the dialog is generated and displayed in the dialog editor.

30 Creating Dialog Elements

To create a dialog element

1 From the **Insert** menu, choose one of the following entries, depending on which dialog element type you wish to create:

ActiveX control
Bitmap
Canvas
Control Box
Edit area
Group frame
Input field
List box
OLE container
Push button
Radio button
Scroll bar
Selection box
Table
Text constant
Toggle button

After one of these items has been selected, you are in creation mode. If you move the mouse within the dialog window, the cursor shape is a cross with a minimized graphical representation of the dialog element to be created.

2 Move the cursor to the desired upper left position of the dialog element.

3 Either hold down the left mouse button, drag the cursor until you have created the desired outline of the new dialog element and release the mouse button.

Or:

Select or press ENTER.

This creates a dialog element with a default size.

The control sequence is the keyboard navigation sequence in which the end user will go through the dialog elements. It is decided by the order in which you create the dialog elements. When you create a new dialog element, it is inserted after the active selection and any of its successive direct and indirect children if the active selection shares the same parent as the newly-inserted control. If not, the insertion point is based on the last dialog element with the same parent which *precedes* the active selection in the control sequence. If there is no such control, or if no controls are selected, the new control is inserted immediately before the first control with the same parent, or immediately after its container if no such control exists. You can modify this default sequence by choosing "Dialog > Control Sequence". For more information, see the section ***Defining the Control Sequence in a Dialog***.

Note: The same rules apply to dialog elements created by pasting them from the clipboard.

If you insert a new dialog element dynamically by using the `PROCESS GUI` statement action `ADD`, you decide its position in the navigation sequence by creating the dialog element and setting the `SUCCESSOR` attribute to the handle value of its successor.

31 Importing Data Fields

You can import a data field from an object in your Natural environment into an input field control or a selection box control which you create at the same time.

To import data fields

1 From the **Insert** menu, choose either **Import > Input field** or **Import > Selection box**.

The **Import Data Field** dialog box appears with the **Library** list box.

The list contains all libraries that reside in the current FNAT and FUSER system file, which are displayed as nodes in the Natural Studio tree view (the display can be limited by using the Display Filter function of Natural Studio). In addition, the list contains all libraries from inactive system files as specified in the steplib table.

2 Choose a library and a Natural object type.

A list of objects appears.

The list contains all objects, which are displayed in the library nodes of the Natural Studio tree view (the display can be limited by using the Display Filter function of Natural Studio).

3 Choose an object.

A list of data fields appears.

4 Choose the data field(s) you want to use for creating a dialog element.

5 Choose **Import**.

The selection box control or the input field control is created with the selected data field(s). Note that the fields themselves are not imported.

32 Editing Dialog Elements

- Cutting a Dialog Element 258
- Copying a Dialog Element 258
- Pasting a Dialog Element from the Clipboard 259
- Deleting a Dialog Element 259
- Selecting all Dialog Elements with the same Parent in a Dialog 260
- Editing a Dialog Element's Attributes 260
- Editing a Dialog Element's Event Handlers 261
- Unifying the Size of Several Dialog Elements 261
- Aligning the Position of Several Dialog Elements 262
- Unifying the Spacing Between Several Dialog Elements 262
- Stretching a Dialog Element 262

To edit one or several dialog elements as a whole, you can use the entries provided in the **Edit** menu. These entries enable you to reuse dialog elements for similar contexts instead of creating them from scratch.

The following topics are covered below:

Cutting a Dialog Element

To cut a dialog element

1 Select the dialog element.

2 From the **Edit** menu or from the dialog element's context menu, choose **Cut**.

Or:

Select the "Cut" toolbar button.

Or:

Press SHIFT+DEL or CTRL+X.

The selected dialog element and any of its child dialog elements is cut to the clipboard for pasting elsewhere, for example into other dialogs.

Note: You can also select several or all dialog elements in a dialog and cut them all at once.

Copying a Dialog Element

To copy a dialog element

1 Select the dialog element.

2 From the **Edit** menu or from the dialog element's context menu, choose **Copy**.

Or:

Select the "Copy" toolbar button.

Or:

Press CTRL+INS or CTRL+C.

The selected dialog element is copied to the clipboard for pasting elsewhere. If the selected dialog element has child dialog elements you will be prompted as to whether these should also be copied or not.

Note: You can also select several or all dialog elements in a dialog and copy them all at once.

Pasting a Dialog Element from the Clipboard

To paste a dialog element from the clipboard

- From the **Edit** menu or from the dialog element's context menu, choose **Paste**.

  Or:

  Select the "Paste" toolbar button.

  Or:

  Press SHIFT+INS or CTRL+V.

  The dialog element in the clipboard is pasted into the current container in the current dialog. The current container is the lowest level dialog element containing the selected dialog elements which is not the selected dialog element itself. If you are pasting a dialog element back into the same container from which it was copied, the original dialog element is overlaid by the copy. You then have to move the pasted dialog element to its new location. (The pasted dialog element is preselected by default.) Note that if it is desired to paste dialog elements into an *empty* container, a dummy child dialog element must be created and selected first.

Note: You can paste several or all dialog elements in a dialog if you have cut or copied them to the clipboard at once.

Deleting a Dialog Element

To delete a dialog element

1 Select the dialog element.

2 From the **Edit** menu or from the dialog element's context menu, choose **Delete**.

  Or:

  Select the "Delete" toolbar button.

Or:

Press DEL.

A dialog box appears asking you to confirm the deletion.

3 Choose **Yes**.

The selected dialog element is deleted, together with any of its child dialog elements.

Note: You can also select several or all dialog elements in a dialog and delete them all at once.

Selecting all Dialog Elements with the same Parent in a Dialog

To select all dialog elements in a dialog

- From the **Edit** menu or from the dialog element's context menu, choose **Select all**.

  If a dialog element is selected, all unselected dialog elements with the same parent become selected. If no dialog element is selected, all top-level dialog elements become selected.

Editing a Dialog Element's Attributes

To edit a dialog element's attributes

1 Select the dialog element.

2 From the **Control** menu or from the dialog element's context menu, choose **Attributes**.

Or:

Double-click on the dialog element.

Or:

Press ENTER.

The dialog element's attributes window appears. To find out what the entries in the attributes window mean, choose "Help". For context-sensitive help, select the attribute entry and press F1.

3 Enter the desired attribute values.

4 Choose **OK** to confirm your changes.

Editing a Dialog Element's Event Handlers

To edit a dialog element's event handlers

1 Select the dialog element.

2 From the **Control** menu or from the dialog element's context menu, choose **Event handlers**.

Or:

Press SHIFT+ENTER.

The dialog element's event handler section appears.

3 Select the event (such as click or double-click).

4 To view the event parameters of an ActiveX control, select the **Event Info** button.

5 Enter the desired event code in free form, either directly in the window itself or by using the Program Editor by selecting the **Editor** push button, then closing the window with the **OK** push button.

Or:

Choose "Use" to enter a user-defined event.

This code will be executed when the event occurs for the dialog element. Note that if you have specified code in the before-any and after-any event sections, this will be triggered before and after the code entered in Step 4. So if you need common event code, you only have to enter it once in your dialog's before-any and after-any event section.

6 Choose **OK** to save your code.

Unifying the Size of Several Dialog Elements

To unify the size of several dialog elements

1 Select all dialog elements whose size is to be unified.

2 Select the dialog element that has the reference width or height.

3 From the **Control** menu, choose either **Unify size > Width** or **Unify size > Height**.

The dialog elements are aligned to the width or height of the reference dialog element.

Aligning the Position of Several Dialog Elements

To align the position of several dialog elements

1 Select all dialog elements whose position is to be aligned.

2 Select the dialog element that has the reference position.

3 From the **Control > Align position** menu, choose one of the following commands: .

Left
Center (horizontal)
Right
Top
Center (vertical)
Bottom

The dialog elements are aligned to the position of the reference dialog element.

Unifying the Spacing Between Several Dialog Elements

To unify the spacing between several dialog elements

1 Select all dialog elements between which you want to unify spacing.

2 From the **Control** menu, choose either **Unify spacing > Horizontal** or **Unify spacing > Vertical**.

The spaces between the dialog elements are now distributed evenly.

Stretching a Dialog Element

To stretch a dialog element in a particular direction

1 Select the dialog element to be stretched.

2 From the **Control** menu, choose **Stretch >** *direction*, where *direction* is, for example, **North west**.

A cursor appears indicating that you can stretch the dialog element.

3 Drag the cursor with the mouse until your dialog element has the desired size.

4 Use the left mouse button to fix the dialog element's size.

The dialog element is now resized; as it is still selected, you can edit it further.

Note: You can also select several dialog elements and stretch them all at the same time.

33 Attributes Windows for Dialogs and Dialog Elements

This part explains the editing options in the attributes windows for dialogs and dialog elements.

- ActiveX Control Attributes Window
- ActiveX Control Property Pages
- Bitmap Control Attributes Window
- Canvas Control Attributes Window
- Control Box Control Attributes Window
- Date/Time Picker Control Attributes Window
- Dialog Attributes Window
- Dialog Bar Control Attributes Window
- Dialog Context Menus Window
- Dialog Image Lists Window
- Edit Area Control Attributes Window
- Group Frame Control Attributes Window
- Image List Base Images Subwindow
- Image List Overlay Images Subwindow
- Input Field Control Attributes Window
- List Box Control Attributes Window
- List View Control Attributes Window
- List View Control Attributes Subwindow
- Menu Editor Window
- OLE Container Control Attributes Window
- Progress Bar Control Attributes Window

- Push Button Control Attributes Window
- Radio Button Control Attributes Window
- Scrollbar Control Attributes Window
- Selecting an OLE Server or Document
- Selection Box Control Attributes Window
- Signal Attributes Window
- Slider Control Attributes Window
- Spin Control Attributes Window
- Status Bar Control Attributes Window
- Status Bar Control Attributes Subwindow
- Table Attributes Window
- Table Attributes Subwindow
- Tab Control Attributes Window
- Tab Control Attributes Subwindow
- Text Constant Control Attributes Window
- Timer Attributes Window
- Toggle Button Control Attributes Window
- Toolbar Attributes Window
- Toolbar Control Attributes Window
- Toolbar Control Attributes Subwindow
- Tree View Control Attributes Window
- Tree View Control Attributes Subwindow
- Wallpaper Attributes Window

34 ActiveX Control Attributes Window

- Entries 268

Accessible Using

1 Double-click on the ActiveX control; or

2 if selected: "Control > Attributes" or by selecting 'Attributes...' from the control's context menu or

3 if selected: ENTER.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Name	Handle name of the ActiveX control (may be overwritten with another name).
Control	Name of the ActiveX control.
DIL Text	DIL-TEXT attribute value (string).
. . .	Dialog box for determining sources of DIL-TEXT attribute values.
Help ID	HELP-ID attribute value.
Context Menu	CONTEXT-MENU attribute value. Specifies the context menu (if any) associated with the control.
Drag Mode	DRAG-MODE attribute value. Indicates whether the control can act as the source in a drag-drop operation and, if so, which types of drag-drop operation it supports.
Drop Mode	DROP-MODE attribute value. Indicates whether the control can act as the target in a drag-drop operation and, if so, which types of drag-drop operation it supports.
Style:	
OK Button	STYLE attribute value: if the end user presses ENTER, this button is pushed. This attribute is only available for ActiveX controls that behave like buttons. These ActiveX controls are marked with the style OLEMISC_ACTSLIKEBUTTON in the system registry.
Cancel Button	STYLE attribute value: if the end user presses ESC, this button is pushed. This attribute is only available for ActiveX controls that behave like buttons. These ActiveX controls are marked with the style OLEMISC_ACTSLIKEBUTTON in the system registry.
State:	
Visible	VISIBLE attribute value.
Enabled	ENABLED attribute value.

Entry in Attributes Window	Represents
Rectangle:	The following four attributes decide the ActiveX control's x and y axis position, its height and its width on the screen. **X** - RECTANGLE-X attribute value. **Y** - RECTANGLE-Y attribute value. **W** - RECTANGLE-W attribute value. **H** - RECTANGLE-H attribute value.
Properties...	Displays a dialog box for editing the properties provided with the selected ActiveX control. To enable editing a property, select it from the "Properties" list box. Only simple properties are displayed in this list box. Other properties (for example parameterized properties) can be configured using the ActiveX control's property pages. See *ActiveX Control Property Pages*. The current value of the selected property is displayed in the "Value" field. If the ActiveX control does not allow reading of the current value, the field is captioned "Value (write-only)" and the value is not displayed. The "Value" field appears as a text box or a combo box, depending on the type of property. There are three ways to edit a property: 1. If "Value" appears as a text box, type in the value and use the "Apply" button to indicate that you have finished editing. 2. If "Value" appears as a combo box, you must pull down the combo box and select an entry. 3. If an additional dialog box is provided to select a value for the property, the "Select..." button is enabled. Choose the "Select..." button. In the dialog box that appears, select a value. Return to the "Properties" dialog box. To confirm, choose the "Apply" button. To reset a property to its initial value, use the "Reset" button. **Note:** The initial value is not displayed as long as the "Properties" dialog box is still open. It is valid, though, after the dialog box has been closed. If you can edit the value of the property directly, the default value is displayed in the "Value" field. To select a value other than the default value, overwrite it. For help on the selected property, select the "Help" button. (The help file for the ActiveX control must have been installed). To confirm the property settings, choose "Close".
About...	Dialog box with information on the ActiveX control.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

35 ActiveX Control Property Pages

Accessible Using

1 if selected: "Control > Property Pages..."; or

2 "Property Pages..." from the context menu.

Property pages are available with most ActiveX controls. They are used to configure the attributes of the ActiveX control in an individual way. After an ActiveX control in a dialog has been configured using its property pages, Natural stores the result of the configuration in binary form in the private resource file associated with the dialog. For more information on resources, see *Resource* in the *Programming Guide*.

36 Bitmap Control Attributes Window

- Entries 274

Accessible Using

1 Double-click on the bitmap control; or

2 if selected: "Control > Attributes" or by selecting 'Attributes...' from the control's context menu or

3 if selected: ENTER.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Name	Handle name of the bitmap control (may be overwritten with another name).
Array...	"Array" dialog box for defining an array of bitmap controls.
Bitmap	BITMAP-FILE-NAME attribute value. If you pull down the selection box, you can choose from the existing set of .bmp files.
. . .	"Source" dialog box for determining sources of BITMAP-FILE-NAME attribute values. Also provides a list of all available bitmaps to be used.
DIL Text	DIL-TEXT attribute value (string).
. . .	"Source" dialog box for determining sources of DIL-TEXT attribute values.
Accelerator	ACCELERATOR attribute value.
. . .	Dialog box for determining sources of ACCELERATOR attribute values.
State:	
Visible	VISIBLE attribute value.
Draggable	DRAGGABLE attribute value. If you check this item, the end user may drag the bitmap control and drop it onto another bitmap control.
Enabled	ENABLED attribute value.
Help ID	HELP-ID attribute value.
Command ID	CLIENT-KEY attribute value (used in this context for associating a command ID).
Context Menu	CONTEXT-MENU attribute value. Specifies the context menu (if any) associated with the control.
Drag Mode	DRAG-MODE attribute value. Indicates whether the control can act as the source in a drag-drop operation and, if so, which types of drag-drop operation it supports.
Drop Mode	DROP-MODE attribute value. Indicates whether the control can act as the target in a drag-drop operation and, if so, which types of drag-drop operation it supports.
Background Color:	

Entry in Attributes Window	Represents
Selection box	BACKGROUND-COLOUR-NAME attribute value. If 'default' is specified, the color of the first (top-left) pixel in the bitmap determines the background color.
. . .	Dialog box for editing BACKGROUND-COLOUR-VALUE attribute value.
Rectangle:	The following four attributes decide the bitmap control's x and y axis position, its height and its width on the screen. **X** - RECTANGLE-X attribute value. **Y** - RECTANGLE-Y attribute value. **W** - RECTANGLE-W attribute value. **H** - RECTANGLE-H attribute value.
Style:	
Vertical Justification:	
Top / Center /	Mutually exclusive STYLE attribute values: align to the
Bottom	bottom, the vertical center, the top.
Horizontal Justification:	
Left / Center /	Mutually exclusive STYLE attribute values: align the bitmap
Right	to the left (of the rectangle), the horizontal center, the right.
Framed	STYLE attribute value: three-dimensional frame.
Scaled	STYLE attribute value: scale the bitmap to fit into the underlying bitmap control's rectangle.
Transparent	STYLE attribute value: bitmap pixels in the background color do not change the state of the screen.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

37 Canvas Control Attributes Window

- Entries 278

Accessible Using

1 Double-click on the canvas control; or

2 if selected: "Control > Attributes" or by selecting 'Attributes...' from the control's context menu or

3 if selected: ENTER.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Name	Handle name of the canvas control (may be overwritten with another name).
Array...	"Array" dialog box for defining an array of canvas controls.
Font	Output field where the font currently selected is displayed.
. . .	Dialog box for selecting fonts.
DIL Text	DIL-TEXT attribute value (string).
. . .	"Source" dialog box for determining sources of DIL-TEXT attribute values.
Help ID	HELP-ID attribute value.
Context Menu	CONTEXT-MENU attribute value. Specifies the context menu (if any) associated with the control.
Style:	
Frame	STYLE attribute value: creates a frame around the canvas control.
State:	
Visible	VISIBLE attribute value.
Enabled	ENABLED attribute value.
Rectangle:	The following four attributes decide the canvas control's x and y axis position, its height and its width on the screen. **X** - RECTANGLE-X attribute value. **Y** - RECTANGLE-Y attribute value. **W** - RECTANGLE-W attribute value. **H** - RECTANGLE-H attribute value.
Foreground Color:	
Selection box	FOREGROUND-COLOUR-NAME attribute value.
. . .	Dialog box for editing FOREGROUND-COLOUR-VALUE attribute value.
OK	Save settings and exit the window.

Entry in Attributes Window	Represents
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

38 Control Box Control Attributes Window

- Entries 282

Accessible Using

1 Double-click on the control box control; or

2 if selected: "Control > Attributes" or by selecting 'Attributes...' from the control's context menu or

3 if selected: ENTER.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Name	Handle name of the control box control.
State	
Visible	VISIBLE attribute value.
Enabled	ENABLED attribute value.
Style:	
Framed	STYLE attribute value; creates a simple frame around the control box control.
Lowered	STYLE attribute value; creates a 3-D border with a sunken appearance.
Exclusive	STYLE attribute value; marks the control box as exclusive. Amongst any set of sibling controls (i.e., controls with the same parent), only one control box marked as exclusive can be visible at any one time. This applies both in the Dialog Editor and at run-time.
Transparent	STYLE attribute value; creates a transparent control box. Allows the control box itself to be invisible without making the child controls it contains invisible.
Size to parent	STYLE attribute value; control boxes with this style are resized to fill the entire client area of their parent whenever the parent control is resized, or when this style is initially set in the Dialog Editor.
Rectangle:	The following four attributes decide the control box control's x and y axis position, its height and its width on the screen. **X** - RECTANGLE-X attribute value. **Y** - RECTANGLE-Y attribute value. **W** - RECTANGLE-W attribute value. **H** - RECTANGLE-H attribute value.
Background color:	
Selection box	BACKGROUND-COLOUR-NAME attribute value.
. . .	Invokes dialog box for editing BACKGROUND-COLOUR-VALUE attribute value.

Entry in Attributes Window	Represents
Wallpaper	WALLPAPER attribute value. Specifies the wallpaper (if any) associated with the control.
Drop Mode	DROP-MODE attribute value. Indicates whether the control can act as the target in a drag-drop operation and, if so, which types of drag-drop operation it supports.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

39 Date/Time Picker Control Attributes Window

- Entries 286

Accessible Using

1 Double-click on the date/time picker control; or

2 if selected: "Control > Attributes" or by selecting 'Attributes...' from the control's context menu or

3 if selected: ENTER.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Name	Handle name of the date/time picker control (may be overwritten with another name).
Font	Output field where the font currently selected is displayed.
. . .	Dialog box for selecting fonts.
DIL text	DIL-TEXT attribute value.
. . .	Dialog box for determining sources of DIL-TEXT attribute values.
Context Menu	CONTEXT-MENU attribute value. Specifies the context menu (if any) associated with the control.
Format string	EDIT-MASK attribute value. Specifies custom format definition (if any). **Note:** The format string used by this control does not use the normal Natural edit mask format specifiers. For a complete list of format specifiers, please refer to the article *Working with Date and Time Picker (DTP) Controls*.
Background:	
Selection box	BACKGROUND-COLOUR-NAME attribute value.
. . .	Invokes dialog box for editing BACKGROUND-COLOUR-VALUE attribute value.
Help ID	HELP-ID attribute value.
Format:	
Short date	STYLE attribute value. System-defined short date format is used. **Note:** The control does not have a sunken appearance if Windows XP styles are active.
Century date	STYLE attribute value. System-defined long date format is used, extended with century information (if not already present). **Note:** Depending on the active regional settings, this format may be identical to the short date format.

Entry in Attributes Window	Represents
Long date	STYLE attribute value. System-defined long date format is used.
Time	STYLE attribute value. Control displays (and stores) only times, not date information.
Style:	
Up-down	STYLE attribute value. Control uses Up and Down buttons to scroll through the values, rather than a drop-down month calendar. **Note:** This option is implicitly set if the control's "time" style is set.
Calendar on right	STYLE attribute value. Drop down calendar is right-aligned with the control, rather than left-aligned.
Allow "no value"	STYLE attribute value. Control displays a check box indicating the presence or absence of a value.
Calendar styles:	These styles relate to the drop-down month calendar (if any).
Week numbers	STYLE attribute value. Week numbers are displayed.
No today	STYLE attribute value. Today's date (click-sensitive) is not displayed at the bottom of the calendar.
No today circle	STYLE attribute value. Today's date is not highlighted (e.g., circled).
State:	
Visible	VISIBLE attribute value. Control will be shown.
Enabled	ENABLED attribute value. Control can accept user input.
Rectangle:	The following four attributes decide the date/time picker control's x and y axis position, its height and its width on the screen. **X** - RECTANGLE-X attribute value. **Y** - RECTANGLE-Y attribute value. **W** - RECTANGLE-W attribute value. **H** - RECTANGLE-H attribute value.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

40 Dialog Attributes Window

- Entries 290

Accessible Using

1 Double-click on the dialog background; or

2 "Dialog > Attributes"; or by selecting 'Attributes...' from the dialog's context menu or

3 ENTER.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Name	Handle name of the dialog window (may be overwritten with another name).
Type	TYPE attribute value. Allows you to decide whether the dialog provides a Multiple Document Interface (MDI frame or MDI child) or not (standard).
String	STRING attribute value (title of the **Dialog** window).
. . .	"Source" dialog box for determining sources of STRING attribute values. For more information on the "Source" dialog box, see **Source**.
Font	Value of the FONT-STRING attribute. Decides the font for all dialog elements in this dialog except for the system-supplied window decorations and the dialog elements for which a font has been chosen explicitly.
. . .	"Font" dialog box for determining sources of FONT-STRING attribute values. For more information on the "Font" dialog box, see **Source**.
Context Menu	CONTEXT-MENU attribute value. Specifies the context menu (if any) associated with the dialog itself.
Icon	BITMAP-FILE-NAME attribute value. Also provides a list of all available icons to be used.
. . .	"Source" dialog box for determining sources of BITMAP-FILE-NAME attribute values. For more information on the "Source" dialog box, see **Source**.
Wallpaper	WALLPAPER attribute value. Specifies the wallpaper (if any) associated with the dialog.
Drop Mode	DROP-MODE attribute value. Indicates whether the control can act as the target in a drag-drop operation and, if so, which types of drag-drop operation it supports.
Help file	HELP-FILENAME attribute value. Decides a dialog's help file name (without extension).
Default button	DEFAULT-BUTTON attribute value: type in or select the handle name of the push-button control for which you want to assign this attribute.
Help ID	HELP-ID attribute value.

Entry in Attributes Window	Represents
Docking	DOCKING attribute value. Determines the sides of the dialog (if any) on which dialog bars or tool bars are allowed to dock.
Compatibility	COMPATIBILITY attribute value. Determines whether the dialog should behave compatibly to an earlier Natural version.
Background Color:	
Selection box	BACKGROUND-COLOUR-NAME attribute value. Choose a predefined color.
. . .	"Custom" dialog box for editing BACKGROUND-COLOUR-VALUE attribute value.
Style:	
Modeless (Popup) /Modal /Dialog box	Mutually exclusive values for the STYLE attribute.
Relative position	STYLE attribute value. The dialog position is interpreted as being relative to its owner window.
Centered position	STYLE attribute value. The dialog will be centered on screen.
Default position	STYLE attribute value. If set, the initial position (but not size) of the dialog is determined by the windowing system. The setting will be ignored if "Dialog box" is set. This option is especially useful for MDI child dialogs.
Default rectangle	Value of the STYLE attribute. If set, the initial position and size of the dialog are decided by the windowing system. The setting will be ignored if "Dialog box" is set.
Control clipping	Value of the STYLE attribute. If set, dialog elements are not allowed to overpaint other dialog elements with the same parent which occur later in the control sequence.
3-D client window	STYLE attribute value. If set, the dialog interior is drawn with a sunken 3-D appearance.
Property Sheet	STYLE attribute value. This option is only enabled if the dialog contains at least one tab control. If set, the Ctrl-Tab and Ctrl-Shift-Tab keys are used for browsing forwards and backwards (respectively) between the tab control's tabs.
State:	
Visible	VISIBLE attribute value. If you check this entry, the dialog is visible.
Enabled	ENABLED attribute value. If you check this entry, the end user may interact with the dialog.
Maximized	MAXIMIZED attribute value. If you check this entry, the dialog is maximized to fill the entire screen.
Minimized	MINIMIZED attribute value. If you check this entry, the dialog is minimized to icon size. The end user then will have to double-click on the icon to restore the dialog to its default size.

Entry in Attributes Window	Represents
Save layout	If checked, the dialog's size and position, together with the layout of any dialog bars, tool bar controls and status bar controls, will be automatically saved and restored on a per-user basis between sessions at run-time. **Note:** This option is not available for MDI child dialogs, or if the dialog is currently untitled.
Popup help	POPUP-HELP attribute value. Help for this dialog or any of its controls will be displayed in a popup window.
Auto-adjust	AUTOADJUST attribute value. If you check this entry, the dialog will be scaled at run time according to the current system font size (i.e. "large fonts"/"small fonts" setting).
Event queueing	EVENT-QUEUEING attribute value. If you check this entry, messages for this dialog are queued instead of being processed immediately.
Background color:	
Selection box	BACKGROUND-COLOUR-NAME attribute value. Choose a predefined color.
. . .	"Custom" dialog box for editing
	BACKGROUND-COLOUR-VALUE attribute value.
Components:	
Menu bar	MENU-HANDLE attribute value: if checked, the dialog editor will assign the handle value specified in the menu bar attributes window.
Toolbar	HAS-TOOLBAR attribute value: if checked, the dialog editor will assign the handle value specified in the toolbar attributes window and set HAS-TOOLBAR to TRUE.
Status bar	HAS-STATUS-BAR attribute value. If you check this entry, the dialog has a status bar.
Dynamic info line	HAS-DIL attribute value. If you check this entry, the dialog has a dynamic information line.
System button	HAS-SYSTEM-BUTTON attribute value. If you check this entry, the dialog has a system button.
Size modifiable	SIZE-MODIFIABLE attribute value. If you check this entry, the dialog's size may be modified.
Maximizable	MAXIMIZABLE attributes value. If you check this entry, the dialog may be maximized.
Minimizable	MINIMIZABLE attribute value. If you check this entry, the dialog may be minimized.
Horizontal scroll bar	HORIZ-SCROLLABLE attribute value. If you check this entry, the dialog has a horizontal scroll bar.
Vertical scroll bar	VERT-SCROLLABLE attribute value. If you check this entry, the dialog has a vertical scroll bar.

Entry in Attributes Window	Represents
Help button	HAS-HELP-BUTTON attribute value. If you check this entry, the dialog title bar contains a help button. **Note:** Windows does not display the help button if minimize and maximize buttons are present.
Rectangle:	The following four attributes decide the dialog's x and y axis position, its height and its width on the screen. **X** - RECTANGLE-X attribute value. **Y** - RECTANGLE-Y attribute value. **W** - RECTANGLE-W attribute value. **H** - RECTANGLE-H attribute value.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Subroutines	Dialog box for editing subroutines.
Help	Provides online help on the attributes window.

41 Dialog Bar Control Attributes Window

- Entries 296

Accessible Using

1 Double-click on the edit area control; or

2 if selected: "Control > Attributes" or by selecting 'Attributes...' from the control's context menu or

3 if selected: ENTER.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Name	Handle name of the dialog bar control.
String	STRING attribute value. This is the text displayed in the window caption when a dockable dialog bar control is floated.
Control ID	CLIENT-KEY attribute value (used in this context for associating a user-defined ID).
Context Menu	CONTEXT-MENU attribute value. Specifies the context menu (if any) associated with the control.
Wallpaper	WALLPAPER attribute value. Specifies the wallpaper (if any) associated with the dialog bar control.
Docking	DOCKING attribute value. Determines the sides of the dialog (if any) on which this dialog bar control is allowed to dock (if dockable). **Note:** The dialog itself must also support docking on the specified side(s).
Location	LOCATION attribute value. Determines the side of the dialog on which the dialog bar control is initially positioned, or whether the dialog bar control is floated in a separate window (if dockable).
Help ID	HELP-ID attribute value.
Background color:	
Selection box	BACKGROUND-COLOUR-NAME attribute value.
. . .	Invokes dialog box for editing BACKGROUND-COLOUR-VALUE attribute value.
Internal Metrics:	
Margin-X	MARGIN-X attribute value. Specifies the margin (in pixels) on the left and right of the dialog bar control.
Margin-Y	MARGIN-Y attribute value. Specifies the margin (in pixels) at the top and bottom of the dialog bar control.
Borders:	

Entry in Attributes Window	Represents
Left	STYLE attribute value. Specifies whether a border should be displayed on the left side of the control. This option is not available for dockable dialog bars.
Top	STYLE attribute value. Specifies whether a border should be displayed at the top of the control. This option is not available for dockable dialog bars.
Right	STYLE attribute value. Specifies whether a border should be displayed on the right side of the control. This option is not available for dockable dialog bars.
Bottom	STYLE attribute value. Specifies whether a border should be displayed at the bottom of the control. This option is not available for dockable dialog bars.
Style:	
Gripper	STYLE attribute value. Determines whether a gripper bar is displayed within the dialog bar control. **Note:** The gripper bar does not appear if the dialog bar control is floated.
Dynamic	STYLE attribute value. Indicates that the dialog bar control can be resized when floated or docked.
3-D border	STYLE attribute value. If set, the dialog bar control's border (if any) is drawn with a 3-D appearance.
Raised	STYLE attribute value. If set, the dialog bar control is drawn with a raised interior.
Close button	STYLE attribute value. If set, the dialog bar control is drawn with a close button (which actually hides the dialog bar, rather than closing it).
Zoom button	STYLE attribute value. If set, the dialog bar control can only be dragged via the gripper bar (if any), and not also via the dialog bar control's background.
UI Transparent	STYLE attribute value. If set, the dialog bar control is drawn with a close button (which actually hides the dialog bar, rather than closing it).
State:	
Visible	VISIBLE attribute value.
Enabled	ENABLED attribute value.
Dockable	DRAGGABLE attribute value. If set, the dialog bar control may be docked and/or floated in its own separate window.
Maximized	MAXIMIZED attribute value. If you check this entry, the dialog bar control is expanded to fill the maximum amount of space available on the row. If you check this entry, this and the other dialog bar controls on the row are restored to their previous size. This option is only available if two or more dynamic dialog bar controls are on the same row.
Rectangle:	The following four attributes decide the dialog bar control's x and y axis position, its height and its width on the screen. **X** - RECTANGLE-X attribute value. **Y** - RECTANGLE-Y attribute value. **W** - RECTANGLE-W attribute value. **H** - RECTANGLE-H attribute value.
OK	Save settings and exit the window.

Entry in Attributes Window	Represents
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

42 Dialog Context Menus Window

- Entries 300

Accessible Using

1 "Dialog > Context Menus"; or

2 CTRL+ALT+X.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Available Context Menus	Shows the context menus currently defined for this dialog. One or more context menus can be selected from this list.
New	Creates a new, empty, context menu with a default name. The new context menu is inserted into the list immediately after the selected context menu(s).
Cut	Cuts the selected context menu(s) to the clipboard.
Paste	Pastes context menu(s) from the clipboard, which are inserted into the list immediately after the selected context menu(s).
Selected Context Menu	Shows information relating to the currently selected context menu. If multiple context menus are selected, this section is disabled.
Name	Displays the name of the currently selected context menu, which can be modified here or in the Menu Editor Window itself.
Enabled	The initial ENABLED attribute state for the context menu. A disabled context menu is suppressed at run-time.
Edit	Invokes the menu editor for the selected context menu, where the menu items themselves can be defined.
Events	Invokes the event handler window for editing the events for the context menu itself. (The events for the menu items and any submenus are accessed using the Menu Editor Window).
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

43 Dialog Image Lists Window

- Entries 302

Accessible Using

1 "Dialog > Image Lists"; or

2 CTRL+ALT+J.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Available image lists:	Displays the handle names of the image lists already created. If you select an item in this list, its attributes are displayed for editing. You can also select several items for cutting and pasting.
New	Creates a new image list.
Cut	Cuts a selected image list and copies it to the clipboard. You can also cut and paste several image lists at once.
Copy	Copies the selected image lists(s) to the clipboard.
Paste	Pastes one or more items from the clipboard. **Note:** The "New" and "Paste" entries insert image lists behind the currently selected item, or, if no items are selected, at the top of the list. You deselect items by holding down CTRL while selecting the selected items.
Selected image list:	In this group frame, you assign attribute values to the item selected in the "Nodes" list box on the left.
Name	Handle name of the image list (may be overwritten with another name).
Base Images...	Subordinate window for defining the image list control's base images. For more information, see *Image List Base Images Subwindow*.
Overlay Images...	Subordinate window for defining the image list control's overlay images. For more information, see *Image List Overlay Images Subwindow*.
Style:	
Large images	STYLE attribute value. Image list control contains images corresponding to the system-defined large icon size.
Small images	STYLE attribute value. Image list control contains images corresponding to the system-defined small icon size. **Note:** An image list control may contain both large and small images.
Internal Metrics:	
Image width	ITEM-H attribute value. Specifies the width of the images in the image file. If zero, the system small icon width is used if the "small images" style is specified, or the

Entry in Attributes Window	Represents
	system large icon width if the "large images" style is specified, or the system small icon width otherwise.
Image height	ITEM-W attribute value. Specifies the height of the images in the image file. If zero, the system small icon height is used if the "small images" style is specified, or the system large icon height if the "large images" style is specified, or the system small icon height otherwise.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

44 Edit Area Control Attributes Window

- Entries 306

Accessible Using

1 Double-click on the edit area control; or

2 if selected: "Control > Attributes" or by selecting 'Attributes...' from the control's context menu or

3 if selected: ENTER.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Name	Handle name of the edit area control (may be overwritten with another name).
Array...	"Array" dialog box for defining an array of edit area controls.
String	STRING attribute value.
. . .	"Source" dialog box for determining sources of STRING attribute values.
Font	Output field where the font currently selected is displayed.
. . .	Dialog box for selecting fonts.
DIL Text	DIL-TEXT attribute value (string).
. . .	"Source" dialog box for determining sources of DIL-TEXT attribute values.
Context Menu	CONTEXT-MENU attribute value. Specifies the context menu (if any) associated with the control.
Drop Mode	DROP-MODE attribute value. Indicates whether the control can act as the target in a drag-drop operation and, if so, which types of drag-drop operation it supports.
State:	
Visible	VISIBLE attribute value.
Enabled	ENABLED attribute value.
Modifiable	MODIFIABLE attribute value. If this entry is checked, the end user may edit the text.
RTL	RTL attribute value. If this entry is checked, the dialog element direction is right-to-left.
Horizontal scroll bar	HORIZ-SCROLLABLE attribute value.
Vertical scroll bar	VERT-SCROLLABLE attribute value.
Help ID	HELP-ID attribute value.
Length	LENGTH attribute value. Specifies the maximum number of characters which can be entered into the edit area control.

Entry in Attributes Window	Represents
	Each line break consumes two characters (carriage return / line feed). This applies regardless of whether the line break was explicitly entered by the user or implicitly inserted due to word wrapping.
Style:	
Framed	STYLE attribute value: creates a frame around the edit area control.
Wordwrapped	STYLE attribute value: when text exceeds the width of the edit area control, it is automatically wrapped to the next line. **Note:** When you set the STYLE attribute value to "WORDWRAP", you cannot set the HORIZ-SCROLLABLE attribute value to "TRUE" and vice versa.
Autoscroll	STYLE attribute value: Text is vertically scrollable and is automatically scrolled upwards when the ENTER key is pressed on the last displayed line. **Note:** This option only has an effect if the edit area control does not have a vertical scroll bar. Otherwise, the text is implicitly autoscrollable.
Rectangle:	The following four attributes decide the edit area control's x and y axis position, its height and its width on the screen. **X** - RECTANGLE-X attribute value. **Y** - RECTANGLE-Y attribute value. **W** - RECTANGLE-W attribute value. **H** - RECTANGLE-H attribute value.
Foreground color:	
Selection box	FOREGROUND-COLOUR-NAME attribute value.
. . .	Dialog box for editing FOREGROUND-COLOUR-VALUE attribute value.
Background color:	
Selection box	BACKGROUND-COLOUR-NAME attribute value.
. . .	Dialog box for editing BACKGROUND-COLOUR-VALUE attribute value.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

45 Group Frame Control Attributes Window

- Entries 310

Accessible Using

1 Double-click on the group frame control; or

2 if selected: "Control > Attributes" or by selecting 'Attributes...' from the control's context menu or

3 if selected: ENTER.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Name	Handle name of the group frame control (may be overwritten with another name).
Array...	Dialog box for defining an array of group frame controls.
String	STRING attribute value.
. . .	"Source" dialog box for determining sources of STRING attribute values.
Font	Output field where the font currently selected is displayed.
. . .	Dialog box for selecting fonts.
Style:	
Container	STYLE attribute value. If checked, all existing controls within the group frame, and any controls created within it, become children of the group frame.
State:	
Visible	VISIBLE attribute value.
Enabled	ENABLED attribute value.
Foreground color:	
Selection box	FOREGROUND-COLOUR-NAME attribute value.
. . .	"Custom" dialog box for editing FOREGROUND-COLOUR-VALUE attribute value.
Background color:	
Selection box	BACKGROUND-COLOUR-NAME attribute value.
. . .	"Custom" dialog box for editing BACKGROUND-COLOUR-VALUE attribute value.
Rectangle:	The following four attributes decide the group frame control's x and y axis position, its height and its width on the screen. **X** - RECTANGLE-X attribute value. **Y** - RECTANGLE-Y attribute value.

Entry in Attributes Window	Represents
	W - RECTANGLE-W attribute value. **H** - RECTANGLE-H attribute value.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

46 Image List Base Images Subwindow

- Entries 314

Accessible Using

- Select the "Base Images..." button in the dialog image lists window.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Images:	Displays the handle names of the image controls already created. If you select an item in this list, its attributes are displayed for editing. You can also select several items for cutting and pasting.
New	Creates a new image control.
Cut	Cuts a selected image control and copies it to the clipboard. You can also cut and paste several image controls at once.
Copy	Copies the selected image control(s) to the clipboard.
Paste	Pastes one or more items from the clipboard. **Note:** The "New" and "Paste" entries insert image controls behind the currently selected item, or, if no items are selected, at the top of the list. You deselect items by holding down CTRL while selecting the selected items.
Selected image:	In this group frame, you assign attribute values to the item selected in the "Images" list box on the left.
Name	Handle name of the image control (may be overwritten with another name).
Bitmap	BITMAP-FILE-NAME attribute value. **Note:** This can be the name of a bitmap (*.bmp) or icon (*.ico) file.
. . .	"Source" dialog box for determining sources of BITMAP-FILE-NAME attribute values.
Background	BACKGROUND-COLOUR-NAME attribute value. If 'default' is specified, the color of the first (top-left) pixel in the bitmap determines the background color. In the case of icons, the icon's bitmap is used.
. . .	Dialog box for editing BACKGROUND-COLOUR-VALUE attribute value.
Style:	
Scaled	STYLE attribute value. Images are scaled to the requested size, rather than being either truncated or extended with the background color.
Transparent	STYLE attribute value. Image is rendered transparently. Generally, if this style is set, pixels in the background color are not drawn. However, if no explicit background color is specified, unscaled icons are instead rendered transparently using their built-in mask.

Entry in Attributes Window	Represents
Composite image	STYLE attribute value. Image file is considered to consist of multiple images joined together horizontally. For example, if the requested image size is 16 by 16 pixels, and the bitmap is 80 by 16 pixels, it will be assumed to comprised of 5 images. If this style were not set, the image would be treated as a single 80 by 16 pixel image, which would then be either truncated or scaled to the requested image size (depending on whether the "scaled" style is set).
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

47 Image List Overlay Images Subwindow

- Entries 318

Accessible Using

- Select the "Overlay Images..." button in the dialog image lists window.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Images:	Displays the handle names of the overlay image controls already created. If you select an item in this list, its attributes are displayed for editing. You can also select several items for cutting and pasting.
New	Creates a new overlay image control.
Cut	Cuts a selected overlay image control and copies it to the clipboard. You can also cut and paste several image controls at once.
Copy	Copies the selected overlay image control(s) to the clipboard.
Paste	Pastes one or more items from the clipboard. **Note:** The "New" and "Paste" entries insert image controls behind the currently selected item, or, if no items are selected, at the top of the list. You deselect items by holding down CTRL while selecting the selected items.
Selected image:	In this group frame, you assign attribute values to the item selected in the "Images" list box on the left.
Name	Handle name of the overlay image control (may be overwritten with another name).
Bitmap	BITMAP-FILE-NAME attribute value. **Note:** This can be the name of a bitmap (*.bmp) or icon (*.ico) file.
. . .	"Source" dialog box for determining sources of BITMAP-FILE-NAME attribute values.
Background	BACKGROUND-COLOUR-NAME attribute value. If 'default' is specified, the color of the first (top-left) pixel in the bitmap determines the background color. In the case of icons, the icon's bitmap is used.
. . .	Dialog box for editing BACKGROUND-COLOUR-VALUE attribute value.
Style:	
Scaled	STYLE attribute value. Images are scaled to the requested size, rather than being either truncated or extended with the background color.
Composite image	STYLE attribute value. Image file is considered to consist of multiple images joined together horizontally. For example, if the requested image size is 16 by 16 pixels, and the bitmap is 80 by 16 pixels, it will be assumed to comprised of 5 images. If this style were not set, the image would be treated as a single 80 by 16 pixel image, which

Entry in Attributes Window	Represents
	would then be either truncated or scaled to the requested image size (depending on whether the "scaled" style is set).
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

48 Input Field Control Attributes Window

- Entries 322

Accessible Using

1 Double-click on the input field control; or

2 if selected: "Control > Attributes" or by selecting 'Attributes...' from the control's context menu.

3 if selected: ENTER.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Name	Handle name of the input field control (may be overwritten with another name).
Array...	Dialog box for defining an array of input field controls.
String	STRING attribute value.
. . .	"Source" dialog box for determining sources of STRING attribute values.
Font	Output field where the font currently selected is displayed.
. . .	Dialog box for selecting fonts.
DIL text	DIL-TEXT attribute value (string).
. . .	Dialog box for determining sources of DIL-TEXT attribute values.
Context Menu	CONTEXT-MENU attribute value. Specifies the context menu (if any) associated with the control.
State:	
Visible	VISIBLE attribute value.
Enabled	ENABLED attribute value.
Modifiable	MODIFIABLE attribute value. If this entry is checked, the end user may edit the text.
RTL	RTL attribute value. If this entry is checked, the dialog element direction is right-to-left.
Help ID	HELP-ID attribute value.
Edit mask	EDIT-MASK attribute value (only enabled if STRING source is a linked variable).
Length	LENGTH attribute value.
Left / Center / Right	Mutually exclusive STYLE attribute values: align input to the left, the center, the right. When you create an input field control, and you assign it a STYLE value of "Center" or "Right", the input field control must be higher than the font. Otherwise, the STRING will not be displayed.
Mandatory	STYLE attribute value: input is mandatory.
Upper case	STYLE attribute value: input will be converted to UPPERCASE letters.

Entry in Attributes Window	Represents
Lower case	STYLE attribute value: input will be converted to lower-case letters.
Nondisplay	STYLE attribute value: input is displayed as a series of asterisks (for example, for passwords).
Digits only	STYLE attribute value: only the digits 0 thru 9 may be entered.
Foreground color:	
Selection box	FOREGROUND-COLOUR-NAME attribute value.
. . .	Dialog box for editing FOREGROUND-COLOUR-VALUE attribute value.
Background color:	
Selection box	BACKGROUND-COLOUR-NAME attribute value.
. . .	Dialog box for editing BACKGROUND-COLOUR-VALUE attribute value.
Rectangle:	The following four attributes decide the input field control's x and y axis position, its height and its width on the screen. **X** - RECTANGLE-X attribute value. **Y** - RECTANGLE-Y attribute value. **W** - RECTANGLE-W attribute value. **H** - RECTANGLE-H attribute value.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

49 List Box Control Attributes Window

- Entries 326

Accessible Using

1 Double-click on the list box control; or

2 if selected: "Control > Attributes" or by selecting 'Attributes...' from the control's context menu or

3 if selected: ENTER.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Name	Handle name of the list box control (may be overwritten with another name).
Items	Input field where you can specify the number of list box items in the list box control. When you enter a number here, the dialog editor generates the corresponding list box items and the "Source" dialog box becomes enabled.
. . .	Dialog box for determining sources of the list box items' STRING attribute values.
Font	Output field where the font currently selected is displayed.
. . .	Dialog box for selecting fonts.
DIL text	DIL-TEXT attribute value (string).
. . .	Dialog box for determining sources of DIL-TEXT attribute values.
Accelerator	ACCELERATOR attribute value.
. . .	Dialog box for determining sources of ACCELERATOR attribute values.
Context Menu	CONTEXT-MENU attribute value. Specifies the context menu (if any) associated with the control.
Help ID	HELP-ID attribute value.
Drag Mode	DRAG-MODE attribute value. Indicates whether the control can act as the source in a drag-drop operation and, if so, which types of drag-drop operation it supports.
Drop Mode	DROP-MODE attribute value. Indicates whether the control can act as the target in a drag-drop operation and, if so, which types of drag-drop operation it supports.
State:	
Visible	VISIBLE attribute value.
Enabled	ENABLED attribute value.
Multiple selection	MULTI-SELECTION attribute value. If you check this entry, the end user may select several list box items at a time.
Sorted	SORTED attribute value. If you check this entry, the items are sorted and you cannot modify them.

Entry in Attributes Window	Represents
Autoselect	AUTOSELECT attribute value. If you check this entry, Natural automatically updates the selection in response to a right mouse button click before displaying a context menu.
Style:	
3-D Border	STYLE attribute value: list box has sunken appearance.
Integral height	STYLE attribute value: partial rows are not displayed.
Insertion Mark	STYLE attribute value: insertion mark will be displayed if the control is a target in a drag-drop operation and a drop is allowed.
Rectangle:	The following four attributes decide the list box control's x and y axis position, its height and its width on the screen. **X** - RECTANGLE-X attribute value. **Y** - RECTANGLE-Y attribute value. **W** - RECTANGLE-W attribute value. **H** - RECTANGLE-H attribute value.
Foreground color:	
Selection box	FOREGROUND-COLOUR-NAME attribute value.
. . .	"Custom" dialog box for editing FOREGROUND-COLOUR-VALUE attribute value.
Background color:	
Selection box	BACKGROUND-COLOUR-NAME attribute value.
. . .	"Custom" dialog box for editing BACKGROUND-COLOUR-VALUE attribute value.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

50 List View Control Attributes Subwindow

- Entries 330

Accessible Using

- Select the "Attributes..." button in the list view control attributes window.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Font	Output field where the font currently selected is displayed.
. . .	Dialog box for selecting fonts.
DIL text	DIL-TEXT attribute value.
. . .	Dialog box for determining sources of DIL-TEXT attribute values.
Accelerator	Accelerator attribute value.
. . .	Dialog box for determining sources of Accelerator attribute values.
Context Menu	CONTEXT-MENU attribute value. Specifies the context menu (if any) associated with the control.
Wallpaper	WALLPAPER attribute value. Specifies the wallpaper (if any) displayed within the control.
Image List	IMAGE-LIST attribute value. Specifies the image list (if any) used to provide the images used by the control's items.
Background:	
Selection box	BACKGROUND-COLOUR-NAME attribute value.
. . .	Invokes dialog box for editing BACKGROUND-COLOUR-VALUE attribute value.
Foreground:	
Selection box	FOREGROUND-COLOUR-NAME attribute value.
. . .	Invokes dialog box for editing FOREGROUND-COLOUR-VALUE attribute value.
Help ID	HELP-ID attribute value.
View Mode	VIEW-MODE attribute value. Specifies the control's display mode.
Drag Mode	DRAG-MODE attribute value. Indicates whether the control can act as the source in a drag-drop operation and, if so, which types of drag-drop operation it supports.
Drop Mode	DROP-MODE attribute value. Indicates whether the control can act as the target in a drag-drop operation and, if so, which types of drag-drop operation it supports.
Spacing:	
X	SPACING-X attribute value. Specifies the width of the logical (in pixels) grid used in the icon view modes.

Entry in Attributes Window	Represents
Y	SPACING-Y attribute value. Specifies the height of the logical (in pixels) grid used in the icon view modes.
W	SPACING attribute value. Specifies the width of the columns in list view mode.
Style:	
3-D border	STYLE attribute value. Control has sunken appearance. **Note:** The control does not have a sunken appearance if Windows XP styles are active.
Align vertically	STYLE attribute value. Items are arranged vertically in icon view modes.
Auto-arrange	STYLEattribute value. Items are maintained arranged on logical grid.
Snap to grid	STYLEattribute value. (Re-)positioned items are snapped to their nearest logical grid position.
No scroll	STYLE attribute value. Scroll bars not displayed in icon view modes.
No header	STYLE attribute value. Column header not displayed in report view mode.
No sort header	STYLE attribute value. Column headers are not click-sensitive.
Check boxes	STYLE attribute value. Displays check boxes next to the items.
Full row select	STYLE attribute value. Selection emphasis extends across full row in report view mode.
Grid lines	STYLE attribute value. Show grid lines in report view mode.
Header drag	STYLE attribute value. Allows re-ordering of columns by dragging their headers.
Label tip	STYLE attribute value. Show tooltip with label text when mouse cursor hovers over partially hidden labels.
Wrap icon labels	STYLE attribute value. Allows item labels to wrap across multiple lines.
Underline hot	STYLE attribute value. Underlines item under mouse cursor and enables single click item activation.
Underline cold	STYLE attribute value. Underlines all items and enables single click item activation.
No hide selection	STYLE attribute value. Keeps selection visible even when control no longer has the focus.
Border select	STYLE attribute value. Emphasizes the large icons of selected items via use of thick border.
Hot-track select	STYLE attribute value. Automatically selects an item when the mouse cursor hovers over it for a short time..
Marquee select	STYLE attribute value. Allows selection of multiple controls in multiple-selection list view controls via "rubber banding" (drag rectangle).
Size to parent	STYLE attribute value. Control size automatically maintained to fill the client area of its parent.
State:	
Visible	VISIBLE attribute value. Control will be shown.
Enabled	ENABLED attribute value. Control can accept user input.

Entry in Attributes Window	Represents
Modifiable	MODIFIABLE attribute value. Item labels may be edited.
RTL	RTL attribute value. If this entry is checked, the dialog element direction is right-to-left.
Multiple selection	MULTI-SELECTION attribute value. Multiple items may be selected.
Sorted	SORTED attribute value. Items are initially sorted in alphabetic sequence.
Descending	DESCENDING attribute value. Items are initially sorted in descending alphabetic sequence.
Tooltips	HAS-TOOLTIP attribute value. If not set, display of the tool tip text (if any) for the items will be suppressed.
Rectangle:	The following four attributes decide the input field control's x and y axis position, its height and its width on the screen. **X** - RECTANGLE-X attribute value. **Y** - RECTANGLE-Y attribute value. **W** - RECTANGLE-W attribute value. **H** - RECTANGLE-H attribute value.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

51 List View Control Attributes Window

- Entries 334

Accessible Using

1 Double-click on the list view control; or

2 if selected: "Control > Attributes" or by selecting "Attributes..." from the control's context menu or from the control's context menu.

3 if selected: ENTER.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Name	Handle name of the list view control (may be overwritten with another name).
Attributes...	Subordinate window for editing the list view control's attribute values. For more information, see *List View Control Attributes Subwindow*.
Columns:	Displays the titles of the list view columns already created. If you select an item in this list, its attributes are displayed for editing. You can also select several items for cutting and pasting.
New	Creates a new column.
Cut	Cuts a selected item and copies it to the clipboard. You can also cut and paste several items at once.
Paste	Pastes one or more columns from the clipboard. **Note:** The "New" and "Paste" entries insert columns behind the currently selected item, or, if no items are selected, at the top of the list. You deselect items by holding down CTRL while selecting the selected items.
Selected column	In this group frame, you assign attribute values to the column selected in the "Columns" list box on the left.
Name	Handle name of the column (may be overwritten with another name).
String	STRING attribute value. Specifies the column's title.
. . .	Dialog box for determining sources of STRING attribute values.
Format	FORMAT attribute value. Specifies the column's data format type and length.
Edit Mask	EDIT-MASKattribute value. Specifies the edit mask (if any) used to display values within the column.
Width	RECTANGLE-W attribute value. Specifies the column width (in pixels).
Alignment:	
Left	STYLE attribute value. Column title is left aligned.

Entry in Attributes Window	Represents
Center	STYLE attribute value. Column title is centered. **Note:** This option has no effect on the primary column, which is always left aligned.
Right	STYLE attribute value. Column title is right aligned. **Note:** This option has no effect on the primary column, which is always left aligned.
Sorting:	
Case insensitive	STYLE attribute value. Alphanumeric columns are sorted case-insensitively.
Word compare	STYLE attribute value. Alphanumeric columns are sorted using a lexical compare that treats hyphens and apostrophes differently such that words such as "coordinate" and "co-ordinate" stay together in a sorted list.
State:	
Visible	VISIBLE attribute value. If checked, the column will be shown.
Events	Dialog box for editing the column's event handlers.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

52 List View Items Subwindow

- Entries 338

Accessible Using

- Select the "Items..." button in the list view control attributes window.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Items:	Displays the handle names of the list view items already created. If you select an item in this list, its attributes are displayed for editing. You can also select several items for cutting and pasting.
New	Creates a new item.
Cut	Cuts a selected item and copies it to the clipboard. You can also cut and paste several items at once.
Copy	Copies the selected item(s) to the clipboard.
Paste	Pastes one or more items from the clipboard. **Note:** The "New" and "Paste" entries insert items behind the currently selected item, or, if no items are selected, at the top of the list. You deselect items by holding down CTRL while selecting the selected items.
Selected item:	In this group frame, you assign attribute values to the items selected in the "Items" list box on the left.
Name	Handle name of the item (may be overwritten with another name).
String	STRING attribute value. Specifies the item's label.
. . .	Dialog box for determining sources of the STRING attribute value.
Tooltip	TOOLTIP attribute value. Specifies the tooltip text associated with the item.
. . .	Dialog box for determining sources of the TOOLTIP attribute value.
Image	IMAGE attribute value. Specifies the image control (if any) containing the image used by the item.
Image index	IMAGE-INDEX attribute value. Specifies the zero-based image number used from the specified image control, if any, or the one-based image number from the control's image list otherwise.
Length	LENGTH attribute value. Specifies the maximum number of label characters that may be entered by the user.
Position:	

Entry in Attributes Window	Represents
X	The following attributes determine the item's position, relative to the control's interior (client) area: X - RECTANGLE-X attribute value. Y - RECTANGLE-Y attribute value. **Note:** These attributes have no effect if the control is not in one of the icon view modes, and may be subsequently implicitly modified if the control's "auto-arrange" or "snap to grid" styles are specified.
Style:	
Upper case	STYLE attribute value. Label characters entered by the user are automatically converted to upper case.
State:	
Checked	CHECKED attribute value. Item is checked. **Note:** Check boxes are only displayed if the control's "check boxes" style is specified.
Modifiable	MODIFIABLE attribute value. Item label is modifiable by the user. **Note:** The control itself must also be MODIFIABLE.
RTL	RTL attribute value. If this entry is checked, the dialog element direction is right-to-left.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

53 Menu Editor Window

- Entries 342

Accessible Using

1 First check the "Menu Bar" field in the "Dialog Attributes" window, then double-click on the dummy menu bar in the dialog; or

2 "Dialog > Menu Bar" or by selecting 'Menu Bar...' from the menu bar's context menu or

3 CTRL+ALT+M.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Menu:	
Name	Handle name of the menu (may be overwritten with another name).
Submenus	Lists the menu's handle name and all of the menu items of MENU-ITEM-TYPE attribute "Submenu" defined so far. This list is indented, that is, the menu structure becomes visible. If you select an entry, its *children* menu items or submenu controls appear in the "Selected Submenu" group frame.
Selected submenu:	Displays the STRING attribute values of the menu items or submenu controls which have been created as child of the "Submenus". You can edit the attributes of the currently "Selected Submenu" in the "Selected Menu Item" group frame. The entries marked ">" are submenus. (You can also select several menu items for cutting and pasting.)
Cool menu	Enables the display of bitmaps within a menu. If not checked, no menu bitmaps will be displayed, even if the menu items themselves have a bitmap assigned to them.
Image width	The width of images to be displayed alongside the menu items. Menu item bitmaps with a different width will be scaled, or truncated or extended (in the background color) to fit, depending on the value of the menu item's 'scaled' attribute.
Image height	The height of images to be displayed alongside the menu items. Menu item bitmaps with a different height will be scaled, or truncated or extended (in the background color) to fit, depending on the value of the menu item's 'scaled' attribute. The specified image height may determine the menu item height if larger than the standard menu item height will allow.
Menu items:	Displays the STRING attribute values of the menu items or submenu controls which have been created as child of the "Submenus". You can edit the attributes of the currently "Selected Submenu" in the "Selected Menu Item" group frame. The entries marked ">" are submenus. (You can also select several menu items for cutting and pasting.)

Entry in Attributes Window	Represents
<< Parent menu Submenu >>	When you are creating a menu hierarchy, these two push buttons enable you to navigate to the next higher level (<< Parent Menu) or the next lower level (Submenu >>) of the existing branches.
Selected menu item:	Displays the attribute values of the selected submenu for editing. For editing, it is necessary that one menu item be selected.
Name	Handle name of the menu item or submenu control (may be overwritten with another name starting with the # sign).
Type	MENU-ITEM-TYPE attribute value for the selected menu item. If the type is "Submenu" or "Window submenu", this item is automatically changed into a submenu control.
Same as	SAME-AS attribute value (only available for MENU-ITEM-TYPE attribute "Normal"); the selection box displays the signals available. If this field is filled, the fields for the attributes which are inherited from the referenced signal are disabled, and can only be re-enabled if the link is broken again by deleting the "Same as" field contents.
OLE	MENU-ITEM-OLE attribute value. If a dialog has a menu bar and an OLE container control is being edited in-place, this attribute decides whether a top-level menu item or a submenu control is not an OLE menu item, or whether it is an item that represents the OLE Container or File or Window group.
String	STRING attribute value.
. . .	Dialog box for determining sources of STRING attribute values.
Bitmap	BITMAP-FILE-NAME attribute value.
. . .	Dialog box for determining sources of BITMAP-FILE-NAME attribute values. Also provides a list of all available bitmaps to be used.
DIL text	DIL-TEXT attribute value (string).
. . .	Dialog box for determining sources of DIL-TEXT attribute values.
Accelerator	ACCELERATOR attribute value.
. . .	Dialog box for determining sources of ACCELERATOR attribute values.
Command ID	CLIENT-KEY attribute value (used in this context for associating a command ID).
Background Color:	
Selection box	BACKGROUND-COLOUR-NAME attribute value to be used for display of the menu item's bitmap (if any). If 'default' is specified, the color of the first (top-left) pixel in the bitmap determines the background color.
. . .	Dialog box for editing BACKGROUND-COLOUR-VALUE attribute value.
State:	
Enabled	ENABLED attribute value.
Shared	SHARED attribute value. CLICK events for this menu item will be forwarded to the active MDI child dialog (if any). This attribute is ignored for non-MDI dialogs.
Checked	CHECKED attribute value (not applicable to submenu controls).
Style:	

Entry in Attributes Window	Represents
Scaled	STYLE attribute value: scale the menu item's bitmap to fit the image height and width specified for the submenu.
Transparent	STYLE attribute value: menu item bitmap pixels in the background color do not change the state of the screen.
Default	STYLE attribute value: menu item text is drawn using a bold font. **Note:** This style is only available for menu items within context menus and submenus. Furthermore, selecting this style for a menu item implicitly deselects the style for all other menu items in the same menu.
Events	Dialog box for editing event handlers; may only be used with the appropriate "Type" field.
New	Creates a new submenu control or menu item. If you change the type in the "Type" field of the "Selected Menu Item" group frame, it creates a menu item with a corresponding MENU-ITEM-TYPE attribute value. Within a submenu, it creates a menu item.
Cut	Cuts the selected menu item(s) and copies it (them) to the clipboard.
Copy	Copies the selected menu item(s) to the clipboard.
Paste	Pastes menu item(s) from the clipboard. **Note:** The "New" and "Paste" entries insert menu items behind the currently selected item, or, if no items are selected, at the top of the list. You deselect items by holding down CTRL while selecting the selected items.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

54 OLE Container Control Attributes Window

- Entries 346

Accessible Using

1 Double-click on the OLE container control; or

2 if selected: "Control > Attributes"; or

3 if selected: ENTER.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Name	Handle name of the OLE container control (may be overwritten with another name).
DIL text	DIL-TEXT attribute value (string).
. . .	Dialog box for determining sources of DIL-TEXT attribute values.
Accelerator	ACCELERATOR attribute value.
. . .	Dialog box for determining sources of ACCELERATOR attribute values.
Help ID	HELP-ID attribute value.
Object Information:	
mf-screen	In this group box, you decide the OLE object's type whose name is then displayed.
Type	Decides whether the OLE container control contains an OLE server, a new OLE object, an existing OLE object, or none of all. For more information on these three types, see *Selecting an OLE Server or Document*. **Note:** This is not a value of the TYPE attribute.
. . .	Dialog box for selecting a particular OLEserver, a new OLE object, or an existing Natural embedded OLE object. EMBEDDED-OBJECT, SERVER-OBJECT and SERVER-PROGID attribute values.
Name	Displays the name of the selected item. You cannot edit this entry.
Framed	STYLE attribute value: draw a frame around the OLE container control.
Zoom (%)	ZOOM-FACTOR attribute value: magnify or reduce the default representation of an OLE server application that has become visible in an OLE container control.
Status:	
Visible	VISIBLE attribute value.
Enabled	ENABLED attribute value.
Modifiable	MODIFIABLE attribute value. If this entry is checked, the end user may modify the OLE object in-place.

Entry in Attributes Window	Represents
RTL	RTL attribute value. If this entry is checked, the dialog element direction is right-to-left.
Rectangle:	The following four attributes decide the OLE container control's x and y axis position, its height and its width on the screen. **X** - RECTANGLE-X attribute value. **Y** - RECTANGLE-Y attribute value. **W** - RECTANGLE-W attribute value. **H** - RECTANGLE-H attribute value.
Background color:	
Selection box	BACKGROUND-COLOUR-NAME attribute value.
. . .	Dialog box for editing BACKGROUND-COLOUR-VALUE attribute value.
OK & Start Server	Save settings, start the OLE server and exit the window.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

55 Progress Bar Control Attributes Window

- Entries 350

Accessible Using

1 Double-click on the progress bar control; or

2 if selected: "Control > Attributes" or by selecting 'Attributes...' from the control's context menu.

3 if selected: ENTER.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Name	Handle name of the progress bar control (may be overwritten with another name).
Context Menu	CONTEXT-MENU attribute value. Specifies the context menu (if any) associated with the control.
Background color:	Defines the color of the progress bar background. **Note:** The specified color is ignored if Windows XP styles are active.
Selection box	BACKGROUND-COLOUR-NAME attribute value.
. . .	Invokes dialog box for editing BACKGROUND-COLOUR-VALUE attribute value.
Foreground color:	Defines the color of the progress bar itself. **Note:** The specified color is ignored if Windows XP styles are active.
Selection box	FOREGROUND-COLOUR-NAME attribute value.
. . .	Invokes dialog box for editing FOREGROUND-COLOUR-VALUE attribute value.
Range:	
Min	MIN attribute value. Lower limit of control's range.
Max	MAX attribute value. Upper limit of control's range.
Pos.	POSITION attribute value. Current progress bar position within control's numeric range.
Style:	
Smooth	STYLE attribute value. Progress bar is smooth rather than segmented. **Note:** The progress bar is always segmented if Windows XP styles are active.
Vertical	STYLE attribute value. The control is oriented vertically, rather than horizontally.
State:	
Visible	VISIBLE attribute value. Control will be shown.

Entry in Attributes Window	Represents
Rectangle:	The following four attributes decide the input field control's x and y axis position, its height and its width on the screen. **X** - RECTANGLE-X attribute value. **Y** - RECTANGLE-Y attribute value. **W** - RECTANGLE-W attribute value. **H** - RECTANGLE-H attribute value.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

56 Push Button Control Attributes Window

- Entries 354

Accessible Using

1 Double-click on the push-button control; or

2 if selected: "Control > Attributes" or by selecting 'Attributes...' from the control's context menu or

3 if selected: ENTER.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Name	Handle name of the push-button control (may be overwritten with another name).
Array...	Dialog box for defining an array of push-button controls.
String	STRING attribute value.
. . .	Dialog box for determining sources of STRING attribute values.
Font	Output field where the font currently selected is displayed.
. . .	Dialog box for selecting fonts.
DIL text	DIL-TEXT attribute value (string).
. . .	Dialog box for determining sources of DIL-TEXT attribute values.
Accelerator	ACCELERATOR attribute value.
. . .	Dialog box for determining sources of ACCELERATOR attribute values.
Context Menu	CONTEXT-MENU attribute value. Specifies the context menu (if any) associated with the control.
State:	
Visible	VISIBLE attribute value.
Enabled	ENABLED attribute value.
Help ID	HELP-ID attribute value.
Style:	
OK Button	STYLE attribute value: if the end user presses ENTER, this button is pushed.
Cancel Button	STYLE attribute value: if the end user presses ESC, this button is pushed.
Command ID	CLIENT-KEY attribute value (used in this context for associating a command ID).
Rectangle:	The following four attributes decide the push button control's x and y axis position, its height and its width on the screen. **X** - RECTANGLE-X attribute value. **Y** - RECTANGLE-Y attribute value.

Entry in Attributes Window	Represents
	W - RECTANGLE-W attribute value. **H** - RECTANGLE-H attribute value.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

57 Radio Button Control Attributes Window

- Entries 358

Accessible Using

1 Double-click on the radio-button control; or

2 if selected: "Control > Attributes" or by selecting 'Attributes...' from the control's context menu or

3 if selected: ENTER.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Name	Handle name of the radio-button control (may be overwritten with another name).
Array...	Dialog box for defining an array of radio-button controls.
String	STRING attribute value.
. . .	Dialog box for determining sources of STRING attribute values.
Font	Output field where the font currently selected is displayed.
. . .	Dialog box for selecting fonts.
DIL text	DIL-TEXT attribute value (string).
. . .	Dialog box for determining sources of DIL-TEXT attribute values.
Accelerator	ACCELERATOR attribute value.
. . .	Dialog box for determining sources of ACCELERATOR attribute values.
Context Menu	CONTEXT-MENU attribute value. Specifies the context menu (if any) associated with the control.
State:	
Visible	VISIBLE attribute value.
Enabled	ENABLED attribute value.
Checked	CHECKED attribute value.
Help ID	HELP-ID attribute value.
Group ID	GROUP-ID attribute value (means this radio-button control belongs to the group of radio buttons with this ID).
Rectangle:	The following four attributes decide the radio-button control's x and y axis position, its height and its width on the screen. **X** - RECTANGLE-X attribute value. **Y** - RECTANGLE-Y attribute value. **W** - RECTANGLE-W attribute value.

Entry in Attributes Window	Represents
	H - RECTANGLE-H attribute value.
Foreground color:	
Selection box	FOREGROUND-COLOUR-NAME attribute value.
. . .	Dialog box for editing FOREGROUND-COLOUR-VALUE attribute value.
Background color:	
Selection box	BACKGROUND-COLOUR-NAME attribute value.
. . .	Dialog box for editing BACKGROUND-COLOUR-VALUE attribute value.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

58 Scrollbar Control Attributes Window

- Entries 362

Accessible Using

1 Double-click on the scroll-bar control; or

2 if selected: "Control > Attributes" or by selecting 'Attributes...' from the control's context menu or

3 if selected: ENTER

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Name	Handle name of the scroll-bar control (may be overwritten with another name starting with the # sign).
Array...	Dialog box for defining an array of scroll-bar controls.
DIL text	DIL-TEXT attribute value (string).
. . .	Dialog box for determining sources of DIL-TEXT attribute values.
Help ID	HELP-ID attribute value.
Context Menu	CONTEXT-MENU attribute value. Specifies the context menu (if any) associated with the control.
State:	
Visible	VISIBLE attribute value.
Enabled	ENABLED attribute value.
Values:	
Minimum	MIN attribute value (minimum numerical value on the scale).
Maximum	MAX attribute value (maximum numerical value on the scale).
Line	LINE attribute value (number of logical units by which the slider moves if the end user presses the up and down arrow buttons).
Page	PAGE attribute value (number of logical units by which the slider moves if the end user clicks on the scroll-bar control's shaft).
Slider	SLIDER attribute value (position of the slider in between the MIN and MAX values).
Rectangle:	The following four attributes decide the scroll-bar control's x and y axis position, its height and its width on the screen. **X** - RECTANGLE-X attribute value. **Y** - RECTANGLE-Y attribute value. **W** - RECTANGLE-W attribute value.

Entry in Attributes Window	Represents
	H - RECTANGLE-H attribute value.
Horizontal / Vertical	Mutually exclusive STYLE attribute values: slider will scroll horizontally or vertically. **Note:** When you edit the STYLE attribute value in the scroll-bar control attributes window, setting "h" instead of "v" and vice versa, the RECTANGLE-H and RECTANGLE-W attribute values are exchanged. The dialog editor thus ensures that the scroll-bar control will not provide for vertical scrolling in a horizontal shape and vice versa.
Foreground color:	
Selection box	FOREGROUND-COLOUR-NAME attribute value.
. . .	Dialog box for editing FOREGROUND-COLOUR-VALUE attribute value.
Background color:	
Selection box	BACKGROUND-COLOUR-NAME attribute value.
. . .	Dialog box for editing BACKGROUND-COLOUR-VALUE attribute value.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

59 Selecting an OLE Server or Document

- Differences Between OLE Server, New OLE Object and Existing OLE Object 366
- OLE Server 366

If you select an OLE server or document, the three options "OLE server", "OLE object" and "Existing OLE object" imply a number of restrictions when using Natural and when using the server application.

This chapter covers the following topics:

Differences Between OLE Server, New OLE Object and Existing OLE Object

Before you select an entry, decide if this is what you need.

Type	Characteristics
OLE Server	Creates an OLE object in its native form. Either server with no content ("Create New") or server with existing file as content ("Create from File").
New OLE Object	Creates a new OLE embedded object to be stored within the Natural environment (default file extension ".neo"). Only "Create New" allowed.
Existing OLE Object	Creates an existing OLE embedded object that has been stored within the Natural environment (default file extension ".neo"). Only "Create from File" allowed.

OLE Server

If you have selected the "OLE server" entry

1. Select the "..." button to the right of the drop-down combo box.

   The "Select OLE Server or Document" dialog box appears. Here you have two options:

   - The "Create New" radio button enables you to select a server application to be started when the end user activates the OLE container control at runtime. The server application is started as such, with no file loaded into it.
   - The "Create from File" radio button enables you to insert the contents of a file as an OLE object. You can browse for the file. When the end user activates the OLE container control at runtime, the application used to create the file is started as a server application, with the content being the selected file.
2. Either select "Create New".

   Or select "Create from File".

   If you have selected "Create New", proceed with 3a.

   If you have selected "Create from File", proceed with 3b.

3. 3a. - From the "Object Type" list box, select an application, for example "Microsoft Word 6.0 Document".

   3b. - In the file text box, enter the path of the file you want to select.

   Or, if you are not sure where the file is, choose the "Browse" button to search in your environment.
4. Select the "Display as Icon" check box or not. This lets you decide whether you want to display your application or file as an application icon inside the OLE container control or whether you want your application or file to appear as a text string called <<*applicationname*>> or <<*path-name*>>. Both act as a placeholder for the server application.

   If you choose to display the application or file as an icon, you can customize the icon by selecting the "Change Icon..." button.
5. To save your settings and quit the dialog box, select OK.

   Or select "Cancel" to quit without saving.

To edit an OLE object inside an OLE container control at runtime

Prerequisite: you have selected "OLE server".

1 Select and hold down the right mouse button inside the OLE container control's rectangle.

   The pop-up menu specific to your server application appears, saying for example:

   Edit *object-type*Object; or

   object-type Object with the submenus "Edit" and "Open".

   "Open" activates the server application in a separate window and enables you to edit and save the object. You can then quit the server application and return to Natural. "Edit" lets you activate the server application inside your Natural dialog.

2 Make your selection in the pop-up menu.

3 Edit (and save) your object using the menu entries provided by the OLE server application.

To quit the OLE server application at runtime

If you have chosen "Edit":

- Select outside the OLE container control's rectangle.

  The OLE server application is deactivated in the Natural dialog, but the object is still displayed inside the OLE container control.

To quit the OLE server application at runtime

If you have chosen "Open":

- From the OLE server application's menu, select "File", then "Close and Return to *container-application-name*".

  The object is unloaded from the OLE server application in the separate window, but the object is still displayed inside the OLE container control.

New OLE Object

If you have selected the "New OLE object" entry, do the following:

1. Select the "..." button to the right of the drop-down combo box.

   The "Select OLE Server or Document" dialog box appears. Important: Here you may only select "Create New", even though the other option is not disabled.
2. Select "Create New".
3. From the "Object Type" list box, select an application, for example "Microsoft Word 6.0 Document".
4. Select the "Display as Icon" check box or not. This lets you decide whether you want to display your object as an application icon inside the OLE container control or whether you want your object to appear as a text string called <<*applicationname*>>.

   If you choose to display the application or file as an icon, you can customize the icon by selecting the "Change Icon..." button.
5. To save your settings and quit the dialog box, select OK.

   Or select "Cancel" to quit without saving.

   You have returned to the attributes window. Note that your server's name now appears in the "Name" text box, prepended by an "@". This is the current value of the SERVER-PROGID attribute. The OK button is disabled. Instead, the "OK & Start Server" button is enabled.
6. Ensure you have made all choices in the attributes window.
7. Choose "OK & Start Server".

   Your attributes window settings are saved and the server application is started.
8. Create your OLE object.
9. Quit the server application. The server application usually provides the menu entries "File", then "Close and Return to *container-application-name*".

   A file list box called "Save As" appears.

10 Save the file as a Natural embedded object with the default file extension ".neo".

If your end user has edited your new OLE object at runtime

1 To edit it in-place, the server application provides additional entries to the Natural application's menu bar.

2 To quit the server application, the server application usually provides the menu entries "File", then "Close and Return to *container-application-name*".

> **Note:** Depending on the server application, you might have to set the focus back to Natural as the server applications usually remain active.

A file list box called "Save As" appears.

3 The end user must save the file as a Natural embedded object with the default file extension ".neo".

Existing OLE Object

If you have selected the "Existing OLE object" entry:

1 Select the "..." button to the right of the drop-down combo box.

The "Select existing Natural Embedded Object" dialog box appears. It displays all Natural embedded objects with the default file extension ".neo" in the default directory.

2 Select a file.

Both the OK and the "OK & Start Server" buttons are enabled. You now have two options:

- If you quit the attributes window by selecting OK, the embedded object will be shown in the container, but cannot be modified (read-only).
- If you quit the attributes window by selecting "OK & Start Server", the corresponding server application is started and the chosen object can be modified (read-write).

3 Choose "OK & Start Server".

Or choose OK.

Your attributes window settings are saved and the server application is started.

4 Modify your OLE object (if you have chosen "OK & Start Server" and the object is read-write).

Or look at your OLE object (if you have chosen OK and the object is read-only).

5 Quit the server application. The server application usually provides the menu entries "File", then "Close and Return to *container-application-name*".

A file list box called "Save As" appears.

6 If you confirm the default, the file is automatically saved as a Natural embedded object with the default file extension ".neo".

60 Selection Box Control Attributes Window

- Entries 372

Accessible Using

1 Double-click on the selection box control; or

2 if selected: "Control > Attributes" or by selecting 'Attributes...' from the control's context menu or

3 if selected: ENTER.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Name	Handle name of the selection box control (may be overwritten with another name).
String	STRING attribute value.
. . .	Dialog box for determining sources of STRING attribute values.
Items	Input field where you can specify the number of selection box items in the selection box control. When you enter a number here, the dialog editor generates the corresponding selection box items and the corresponding "Source" dialog box becomes enabled.
. . .	Dialog box for determining sources of the selection box items' STRING attribute values.
Font	Output field where the font currently selected is displayed.
. . .	Dialog box for selecting fonts.
DIL text	DIL-TEXT attribute value (string).
. . .	Dialog box for determining sources of DIL-TEXT attribute values.
Context Menu	CONTEXT-MENU attribute value. Specifies the context menu (if any) associated with the control.
State:	
Visible	VISIBLE attribute value.
Enabled	ENABLED attribute value.
Modifiable	MODIFIABLE attribute value.
RTL	RTL attribute value. If this entry is checked, the dialog element direction is right-to-left.
Sorted	SORTED attribute value. If you check this entry, the items are sorted and you cannot modify them.
Help ID	HELP-ID attribute value.
Edit mask	EDIT-MASK attribute value.

Entry in Attributes Window	Represents
Length	LENGTH attribute value.
Rectangle:	The following four attributes decide the selection box control's x and y axis position, its height and its width on the screen. **X** - RECTANGLE-X attribute value. **Y** - RECTANGLE-Y attribute value. **W** - RECTANGLE-W attribute value. **H** - RECTANGLE-H attribute value.
Foreground color:	
Selection box	FOREGROUND-COLOUR-NAME attribute value.
. . .	Dialog box for editing FOREGROUND-COLOUR-VALUE attribute value.
Background color:	
Selection box	BACKGROUND-COLOUR-NAME attribute value.
. . .	Dialog box for editing BACKGROUND-COLOUR-VALUE attribute value.
Mandatory	STYLE attribute value: input in the selection box control's input field is mandatory.
Upper case	STYLE attribute value: input will be converted to UPPERCASE letters.
Box dropped down	STYLE attribute value: the box stays dropped down all the time.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

61 Signal Attributes Window

- Entries 376

Accessible Using

1 "Dialog > Signals"; or

2 CTRL+ALT+N.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Signals:	Displays the handle name of the signals already created. If you select a signal in the list, its attributes are displayed for editing. You can also select several signals for cutting and pasting.
New	Creates a new signal.
Cut	Cuts the selected signal and copies it to the clipboard. You can also cut and paste several signals at once.
Copy	Copies the selected signal(s) to the clipboard.
Paste	Pastes a signal from the clipboard. **Note:** The "New" and "Paste" entries insert signals behind the currently selected signal, or, if no signals are selected, at the top of the list. You deselect items by holding down CTRL while selecting the selected items.
Selected signal:	In this group frame, you assign attribute values to the signals selected in the "Signals" list box on the left.
Name	Handle name of the signal (may be overwritten with another name).
Type	MENU-ITEM-TYPE attribute value for the selected signal.
Bitmap	BITMAP-FILE-NAME attribute value.
. . .	Dialog box for determining sources of BITMAP-FILE-NAME attribute values. Also provides a list of all available bitmaps to be used.
DIL text	DIL-TEXT attribute value (string).
. . .	Dialog box for determining sources of DIL-TEXT attribute values.
Accelerator	ACCELERATOR attribute value.
. . .	Dialog box for determining sources of ACCELERATOR attribute values.
Tooltip	TOOLTIP attribute value.
...	Dialog box for determining sources of TOOLTIP attribute values.
Command ID	CLIENT-KEY attribute value (used in this context for associating a command ID).
Background Color:	

Entry in Attributes Window	Represents
Selection box	BACKGROUND-COLOUR-NAME attribute value to be used for display of the signal's bitmap (if any). If 'default' is specified, the color of the first (top-left) pixel in the bitmap determines the background color.
...	Dialog box for editing BACKGROUND-COLOUR-VALUE attribute value.
State:	
Visible	VISIBLE attribute value.
Enabled	ENABLED attribute value.
Checked	CHECKED attribute value.
Shared	SHARED attribute value. If checked, CLICK events for this signal will be forwarded to the active MDI child dialog (if any). This attribute is ignored for non-MDI dialogs.
Events	Dialog box for editing event handlers; may only be used with the appropriate "Type" entry.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

62 Slider Control Attributes Window

- Entries 380

Accessible Using

1 Double-click on the slider control; or

2 if selected: "Control > Attributes" or by selecting 'Attributes...' from the control's context menu.

3 if selected: ENTER.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Name	Handle name of the slider control (may be overwritten with another name).
DIL text	DIL-TEXT attribute value.
. . .	Dialog box for determining sources of DIL-TEXT attribute values.
Context Menu	CONTEXT-MENU attribute value. Specifies the context menu (if any) associated with the control.
Help ID	HELP-ID attribute value.
Spacing	SPACING attribute value. tick mark interval. If zero, the default value of one is used. **Note:** This option is only effective if the "auto ticks" style is set.
Range:	
Min	MIN attribute value. Lower limit of control's slider range.
Max	MAX attribute value. Upper limit of control's slider range.
Line	LINE attribute value. Absolute value change caused by pressing the corresponding arrow keys on the keyboard whilst the control has the focus.
Page	PAGE attribute value. Absolute value change caused by pressing the "Page Up" and "Page Down" keys on the keyboard whilst the control has the focus, or by clicking on the slider's shaft.
Slider	SLIDER attribute value. Current thumb position.
Style:	
Auto ticks	STYLE attribute value. Tick marks are automatically displayed at intervals determined by the specified SPACING. **Note:** The tick marks at the lower and upper end of the slider's range are permanent and are always displayed.
Side 1 ticks	STYLE attribute value. Tick marks are displayed on the top or left side of the slider, depending on its orientation.

Entry in Attributes Window	Represents
Side 2 ticks	STYLE attribute value. Tick marks are displayed on the bottom or right side of the slider, depending on its orientation.
Vertical	STYLE attribute value. The control is oriented vertically, rather than horizontally.
Position tip	STYLE attribute value. A tooltip window displays the current position when the thumb is dragged.
No thumb	STYLE attribute value. Control does not display a thumb.
State:	
Visible	VISIBLE attribute value. Control will be shown.
Enabled	ENABLED attribute value. Control can accept user input.
Rectangle:	The following four attributes decide the input field control's x and y axis position, its height and its width on the screen. **X** - RECTANGLE-X attribute value. **Y** - RECTANGLE-Y attribute value. **W** - RECTANGLE-W attribute value. **H** - RECTANGLE-H attribute value.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

63 Spin Control Attributes Window

- Entries 384

Accessible Using

1 Double-click on the spin control (not on its buddy, if any); or

2 if selected: "Control > Attributes" or by selecting 'Attributes...' from the control's context menu.

3 if selected: ENTER.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Name	Handle name of the spin control (may be overwritten with another name).
Context Menu	CONTEXT-MENU attribute value. Specifies the context menu (if any) associated with the control.
Array...	Dialog box for defining an array of input field controls.
String	STRING attribute value.
. . .	"Source" dialog box for determining sources of STRING attribute values.
Font	Output field where the font currently selected is displayed.
. . .	Dialog box for selecting fonts.
DIL text	DIL-TEXT attribute value (string).
. . .	Dialog box for determining sources of DIL-TEXT attribute values.
Context Menu	CONTEXT-MENU attribute value. Specifies the context menu (if any) associated with the control.
Help ID	HELP-ID attribute value.
Range:	
Min	MIN attribute value. Lower limit of control's numeric range.
Max	MAX attribute value. Upper limit of control's numeric range.
Pos.	POSITION attribute value. Current position within control's numeric range.
Style:	
Horizontal	STYLE attribute value. Control is oriented horizontally, rather than vertically.
Left align	STYLE attribute value. The control has an implicit buddy input field. The spin control's buttons are aligned to the left of the buddy.
Right align	STYLE attribute value. The control has an implicit buddy input field. The spin control's buttons are aligned to the right of the buddy.
Set buddy	STYLE attribute value. Buddy control (if any) is automatically set to indicate the spin control's position when the latter changes.

Entry in Attributes Window	Represents
Wrap	STYLE attribute value. Control value wraps from maximum to minimum value, and vice versa, if scrolled past the corresponding limit.
Arrow keys	STYLE attribute value. The control's value can be scrolled via use of the arrow keys.
No thousands	STYLE attribute value. No thousands separators are displayed in the buddy control (if any). For example, "1234" instead of "1,234".
Hot tracking	STYLE attribute value. Buttons are highlighted when traversed by the mouse cursor. **Note:** This option is implicitly set if Windows XP styles are in use.
Hexadecimal	STYLE attribute value. Values are displayed in buddy control (if any) in hexadecimal format (e.g. "0x0032" instead of "50"). **Note:** This option is only effective if the "set buddy" style is also set.
State:	
Visible	VISIBLE attribute value. Control will be shown.
Enabled	ENABLED attribute value. Control can accept user input.
Rectangle:	The following four attributes decide the input field control's x and y axis position, its height and its width on the screen. **X** - RECTANGLE-X attribute value. **Y** - RECTANGLE-Y attribute value. **W** - RECTANGLE-W attribute value. **H** - RECTANGLE-H attribute value.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

64 Status Bar Control Attributes Window

- Entries 388

Accessible Using

1 Double-click on the status bar control; or

2 if selected: "Control >Attributes " or by selecting 'Attributes...' from the control's context menu or

3 if selected: ENTER.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Name	Handle name of the status bar control (may be overwritten with another name)
Attributes...	Subordinate window for editing the status bar control's attribute values. For more information, see *Status Bar Control Attributes Subwindow*. (Normally, all attributes of a dialog element can be edited in the attributes window. Instead, the attributes of each pane in the status bar control can be edited here. For reasons of space, the status bar control's attributes are edited in a separate subwindow).
Status-bar panes:	Displays the handle name of the panes already created. If you select a pane in this list, its attributes are displayed for editing. You can also select several panes for cutting and pasting.
New	Creates a new pane.
Cut	Cuts the selected pane and copies it to the clipboard. You can also cut and paste several panes at once.
Paste	Pastes a pane from the clipboard. **Note:** The "New" and "Paste" entries insert panes behind the currently selected item, or, if no items are selected, at the top of the list. You deselect items by holding down CTRL while selecting the selected items.
Selected status bar pane:	In this group frame, you assign attribute values to the panes selected in the "Status bar panes" list box on the left.
Name	Handle name of the pane (may be overwritten with another name).
Width	ITEM-W attribute value. Specifies the width of the pane in pixels. **Note:** If 0, the pane does not have a fixed width, but is instead automatically sized to fill the space available ("stretchy pane").
String	STRING attribute value. Specifies the initial pane text.
. . .	Dialog box for determining sources of STRING attribute values.

Entry in Attributes Window	Represents
Icon	BITMAP-FILE-NAME attribute value. Species the icon (if any) to be displayed alongside the pane text. Natural attempts to extract the small (16 x 16 pixel) icon (if any) from the specified icon file. If only a large (32 x 32 pixel) icon is present, Windows will automatically synthesize a small icon from it, which may lead to undesirable scaling effects.
. . .	Dialog box for determining sources of BITMAP-FILE-NAME attribute values. Also provides a list of all available icons to be used.
Tooltip	TOOLTIP attribute value.
...	Dialog box for determining sources of TOOLTIP attribute values.
Command ID	CLIENT-KEY attribute value (used in this context for associating a command ID).
State:	
Visible	VISIBLE attribute value.
Enabled	ENABLED attribute value.
Shared	SHARED attribute value. If checked, CLICK events for this pane will be forwarded to the active MDI child dialog (if any). This attribute is ignored for non-MDI dialogs.
Style:	
Centered	STYLE attribute value. If set, text will be horizontally centered within the pane.
Hide disabled	STYLE attribute value. If set, the pane text and icon (if any) will be hidden (instead of being grayed out) when the pane is disabled.
Raised	STYLE attribute value. If set, the pane appears to "pop out".
No borders	STYLE attribute value. If set, the pane borders are not drawn. This style is typically applied to stretchy panes.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

65 Status Bar Control Attributes Subwindow

- Entries 392

Accessible Using

- Select the "Attributes..." button in the status bar control attributes window

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attrib. Subwindow	Represents
Control ID	CLIENT-KEY attribute value (used in this context for associating a user-defined ID).
Context Menu	CONTEXT-MENU attribute value. Specifies the context menu (if any) associated with the control.
Location	LOCATION attribute value. Determines the side of the dialog on which the status bar control is initially positioned.
Internal Metrics:	
Minimum height	ITEM-H attribute value. Specifies the *minimum* height of the status bar panes (in pixels). This is particularly useful for status bar controls which display icons. By default, the minimum height of a status bar control depends on the font used to draw the text.
Margin-X	MARGIN-X attribute value. Specifies the margin (in pixels) to the left and right of the status bar panes.
Margin-Y	MARGIN-Y attribute value. Specifies the margin (in pixels) above and below the status bar panes.
Borders:	
Top	STYLE attribute value. Species whether a border should be displayed at the top of the control.
Bottom	STYLE attribute value. Species whether a border should be displayed at the bottom of the control.
3-D	STYLE attribute value. If set, the status bar control borders (if any) are drawn with a 3-D appearance.
Style:	
Gripper	STYLE attribute value. Determines whether a sizing gripper is displayed within the status bar control.
State:	
Visible	VISIBLE attribute value.
Enabled	ENABLED attribute value.
Tooltips	HAS-TOOLTIP attribute value. If not set, display of the tool tip text (if any) for the status bar panes will be suppressed.

Entry in Attrib. Subwindow	Represents
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

66 Table Attributes Window

- Entries 396

Accessible Using

1 Double-click on the table; or

2 if selected: "Control > Attributes" or by selecting 'Attributes...' from the control's context menu or

3 if selected: ENTER.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Table:	
Name	Handle name of the table (may be overwritten with another name)
Attributes...	Subordinate window for editing the table's attribute values. For more information, see *Table Attributes Subwindow*. (Normally, all attributes of a dialog element can be edited in the attributes window. Instead, the attributes of each column specification control in the table can be edited here. For reasons of space, the table's attributes are edited in a separate subwindow).
Columns:	Displays the handle name, the COLUMN-TYPE and the STRING attribute values of the column specification controls already created. If you select a column specification control, its attributes are displayed for editing. You can also select several column specifications for cutting and pasting.
Selected Column Specification:	In this group frame, you assign attribute values to the column specification controls selected in the "Columns" list box on the left.
Name	Handle name of the column specification control (may be overwritten with another name).
Type	COLUMN-TYPE attribute value for the selected column specification control. If the type is "Selection Box", the "Items" entry is enabled and enables you to define the number of selection box items and the source of their values.
String	STRING attribute value.
. . .	Dialog box for determining sources of STRING attribute values.
DIL text	DIL-TEXT attribute value (string).
. . .	Dialog box for determining sources of DIL-TEXT attribute values.
Items	If the "Type" entry is set to "Selection Box", this entry is enabled and allows you to enter the number of selection box items.
. . .	Dialog box for determining sources of selection box item values.

Entry in Attributes Window	Represents
Help ID	HELP-ID attribute value.
Width	RECTANGLE-W attribute value.
Length	LENGTH attribute value.
State:	
Modifiable	MODIFIABLE attribute value.
RTL	RTL attribute value. If this entry is checked, the dialog element direction is right-to-left.
New	Creates a new column specification control. If you change the type in the "Type" field of the "Selected Column Specification" group frame, it creates a column specification control with a corresponding COLUMN-TYPE attribute value.
Cut	Cuts the selected column specification control(s) and copies it (them) to the clipboard.
Paste	Pastes the selected column specification control(s) from the clipboard. **Note:** The "New" and "Paste" entries insert column specification controls behind the currently selected control, or, if no controls are selected, at the top of the list. You deselect controls by holding down CTRL while selecting the selected controls.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

67 Table Attributes Subwindow

- Entries 400

Accessible Using

- Select the "Attributes..." button in the table attributes window.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attrib. Subwindow	Represents
Name	Handle name of the table (may be overwritten with another name).
Font	Output field where the font currently selected is displayed.
. . .	Dialog box for selecting fonts.
Header font	Output field where the font currently selected for the table header is displayed.
. . .	Dialog box for selecting fonts.
DIL text	DIL-TEXT attribute value (string).
. . .	Dialog box for determining sources of DIL-TEXT attribute values.
Context Menu	CONTEXT-MENU attribute value. Specifies the context menu (if any) associated with the control.
Rectangle:	The following four attributes decide the table's x and y axis position, its height and its width on the screen. **X** - RECTANGLE-X attribute value. **Y** - RECTANGLE-Y attribute value. **W** - RECTANGLE-W attribute value. **H** - RECTANGLE-H attribute value.
Row count	ROW-COUNT attribute value.
Row height	ROW-HEIGHT attribute value.
Header height	HEADER-HEIGHT attribute value.
Width 1st col.	FIRST-COLUMN-WIDTH attribute value.
Frozen cols.	FIRST-COLUMN-WIDTH attribute value.
Help ID	HELP-ID attribute value.
First visible col.	FIRST-VISIBLE-COLUMN attribute value.
First visible row	FIRST-VISIBLE-ROW attribute value.
Columns header	STYLE attribute value: buttons with field names are displayed at the top of each column.
Extendable	STYLE attribute value: end users can delete and insert rows using DEL and INS.
No lines	STYLE attribute value: the table control is displayed without the lines that normally separate the cells.

Entry in Attrib. Subwindow	Represents
Resize columns	STYLE attribute value: end users may resize the columns horizontally.
Single cell selection	STYLE attribute value: if set, end users may only select single cells. If not set, end users may select ranges of cells.
Resize rows	STYLE attribute value: end users may resize the rows vertically.
Whole row selection	STYLE attribute value: selecting an individual cell sets the selection to the entire row.
Draggable columns	STYLE attribute value: if set, end users may drag the columns.
Integral height	STYLE attribute value: partial rows are not displayed.
Foreground color:	
Selection box	FOREGROUND-COLOUR-NAME attribute value.
. . .	Dialog box for editing FOREGROUND-COLOUR-VALUE attribute value.
Background color:	
Selection box	BACKGROUND-COLOUR-NAME attribute value.
. . .	Dialog box for editing BACKGROUND-COLOUR-VALUE attribute value.
State:	
Visible	VISIBLE attribute value.
Enabled	ENABLED attribute value.
Modifiable	MODIFIABLE attribute value.
RTL	RTL attribute value. If this entry is checked, the dialog element direction is right-to-left.
Has first column	HAS-FIRST-COLUMN attribute value.
Horizontal scroll bar	HORIZ-SCROLLABLE attribute value.
Vertical scroll bar	VERT-SCROLLABLE attribute value.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

68 Tab Control Attributes Window

- Entries 404

Accessible Using

1 Double-click on the tab control; or

2 if selected: "Control > Attributes" or by selecting 'Attributes...' from the control's context menu or

3 if selected: ENTER.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attrib. Subwindow	Represents
Name	Handle name of the tab control (may be overwritten with another name).
Attributes...	Subordinate window for editing the tab control's attribute values. For more information, see *Tab Control Attributes Subwindow*. (Normally, all attributes of a dialog element can be edited in the attributes window. Instead, the attributes of each tab in the tab control can be edited here. For reasons of space, the tab control's attributes are edited in a separate subwindow).
Tabs:	Displays the labels of the tabs already created. If you select an item in this list, its attributes are displayed for editing. You can also select several items for cutting and pasting.
New	Creates a new tab.
Cut	Cuts a selected item and copies it to the clipboard. You can also cut and paste several items at once.
Paste	Pastes one or more tabs from the clipboard. **Note:** The "New" and "Paste" entries insert tabs behind the currently selected item, or, if no items are selected, at the top of the list. You deselect items by holding down CTRL while selecting the selected items.
Selected tab	In this group frame, you assign attribute values to the tabs selected in the "Tabs" list box on the left.
Name	Handle name of the tab (may be overwritten with another name).
String	STRING attribute value. Specifies the tab's label.
Icon	BITMAP-FILE-NAME attribute value. Specifies the tab's icon.
Tooltip	TOOLTIP attribute value. Specifies the tab's tool tip text.
State:	
Visible	VISIBLE attribute value. If checked, the tab will be shown.

Entry in Attrib. Subwindow	Represents
Selected	SELECTED attribute value. If checked, the tab will be selected. **Note:** Only one tab may be selected at any one time.
Events	Dialog box for editing the tab's event handlers.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

69 Tab Control Attributes Subwindow

- Entries 408

Accessible Using

- Select the "Attributes..." button in the tab control attributes window.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attrib. Subwindow	Represents
Font	Output field where the font currently selected is displayed.
. . .	Dialog box for selecting fonts.
Context Menu	CONTEXT-MENU attribute value. Specifies the context menu (if any) associated with the control.
Wallpaper	WALLPAPER attribute value. Specifies the wallpaper (if any) associated with the tab control's interior area.
Background:	
Selection box	BACKGROUND-COLOUR-NAME attribute value.
. . .	Invokes dialog box for editing BACKGROUND-COLOUR-VALUE attribute value.
Help ID	HELP-ID attribute value.
Tab Metrics:	
Width	ITEM-W attribute value. Specifies the width of all tabs in the tab control if the "fixed-width tab" style is set, otherwise specifies the minimum tab width.
Height	ITEM-H attribute value (specifies the height of all tabs in the tab control).
Margin-X	MARGIN-X attribute value. Specifies the margin (in pixels) on the left and right of each tab in the tab control.
Margin-Y	MARGIN-Y attribute value. Specifies the margin (in pixels) at the top and bottom of each tab in the tab control.
Style:	
Bottom tabs	STYLE attribute value. If set, tabs are displayed at the bottom of the tab control instead of at the top.
Fixed-width tabs	STYLE attribute value. If set, all tabs have the same width.
Multi-row	STYLE attribute value. If set, the tab control will display multiple rows of tabs (instead of one scrollable row) if not enough room is available to display them simultaneously on one row.
Ragged	STYLE attribute value. If set, tabs in multi-row tab controls are only aligned on one side of the control.
Left icon	STYLE attribute value. If set, tab icons are left-aligned.

Entry in Attrib. Subwindow	Represents
Left label	STYLE attribute value. If set, tab labels are left-aligned. **Note:** This style also implicitly implies the "left icon" style.
Browsable	STYLE attribute value. If set, the tabs can receive the focus, whereupon browsing between the tabs is possible with the arrow keys.
State:	
Visible	VISIBLE attribute value. If checked, the tab control will be shown.
Enabled	ENABLED attribute value. If checked, the tab control can accept user input.
Modifiable	MODIFIABLE attribute value. If checked, the user is able to switch between tabs.
RTL	RTL attribute value. If this entry is checked, the dialog element direction is right-to-left.
Tooltips	HAS-TOOLTIP attribute value. If not set, display of the tool tip text (if any) for the tabs will be suppressed.
Rectangle:	The following four attributes decide the tab control's x and y axis position, its height and its width on the screen. **X** - RECTANGLE-X attribute value. **Y** - RECTANGLE-Y attribute value. **W** - RECTANGLE-W attribute value. **H** - RECTANGLE-H attribute value. **Note:** The positions are relative to the dialog window.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

70 Text Constant Control Attributes Window

- Entries 412

Accessible Using

1 Double-click on the text constant control; or

2 if selected: "Control > Attributes" or by selecting 'Attributes...' from the control's context menu or

3 if selected: ENTER.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Name	Handle name of the text constant control (may be overwritten with another name).
Array...	Dialog box for defining an array of text constant controls.
String	STRING attribute value.
. . .	Dialog box for determining sources of STRING attribute values.
Font	Output field where the font currently selected is displayed.
. . .	Dialog box for selecting fonts.
Style:	
Left / Centered / Right	Mutually exclusive STYLE attribute values: align output to the left, the center, the right.
Framed	STYLE attribute value: draw a frame around the text constant control.
State:	
Visible	VISIBLE attribute value.
Enabled	ENABLED attribute value.
Rectangle:	The following four attributes decide the text constant control's x and y axis position, its height and its width on the screen. **X** - RECTANGLE-X attribute value. **Y** - RECTANGLE-Y attribute value. **W** - RECTANGLE-W attribute value. **H** - RECTANGLE-H attribute value.
Foreground color:	
Selection box	FOREGROUND-COLOUR-NAME attribute value.
. . .	Dialog box for editing FOREGROUND-COLOUR-VALUE attribute value.
Background color:	
Selection box	BACKGROUND-COLOUR-NAME attribute value.

Entry in Attributes Window	Represents
. . .	Dialog box for editing BACKGROUND-COLOUR-VALUE attribute value.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

71 Timer Attributes Window

- Entries 416

Accessible Using

1 "Dialog > Timers"; or

2 CTRL+ALT+I.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Timer:	Displays the handle names and the TIMER-INTERVAL attribute values of the timers already created. (You may create up to 16 timers per dialog).
Selected Timer:	In this group frame, you assign attribute values to the timer selected in the "Timers" list box on the left.
Name	Handle name of the timer (may be overwritten with another name starting with the # sign).
Interval	TIMER-INTERVAL attribute value.
Events	Dialog box for editing event handlers.
New	Creates a new timer.
Cut	Cuts the selected timer and copies it to the clipboard. (You can cut and paste one or several timers).
Paste	Pastes timer(s) from the clipboard. **Note:** The "New" and "Paste" entries insert timers behind the currently selected timer, or, if none is selected, at the top of the list. You deselect timers by holding down CTRL while selecting the selected timers.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

72 Toggle Button Control Attributes Window

- Entries 418

Accessible Using

1 Double-click on the toggle-button control; or

2 if selected: "Control > Attributes" or by selecting 'Attributes...' from the control's context menu or

3 if selected: ENTER.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Name	Handle name of the toggle-button control (may be overwritten with another name).
Array...	Dialog box for defining an array of toggle-button controls.
String	STRING attribute value.
. . .	Dialog box for determining sources of STRING attribute values.
Font	Output field where the font currently selected is displayed.
. . .	Dialog box for selecting fonts.
DIL text	DIL-TEXT attribute value (string).
. . .	Dialog box for determining sources of DIL-TEXT attribute values.
Accelerator	ACCELERATOR attribute value.
. . .	Dialog box for determining sources of ACCELERATOR attribute values.
Context Menu	CONTEXT-MENU attribute value. Specifies the context menu (if any) associated with the control.
Help ID	HELP-ID attribute value.
State:	
Visible	VISIBLE attribute value.
Enabled	ENABLED attribute value.
Checked	CHECKED attribute value.
Foreground color:	
Selection box	FOREGROUND-COLOUR-NAME attribute value.
. . .	Dialog box for editing FOREGROUND-COLOUR-VALUE attribute value.
Background color:	
Selection box	BACKGROUND-COLOUR-NAME attribute value.
. . .	Dialog box for editing BACKGROUND-COLOUR-VALUE attribute value.

Entry in Attributes Window	Represents
Rectangle:	The following four attributes decide the toggle-button control's x and y axis position, its height and its width on the screen. **X** - RECTANGLE-X attribute value. **Y** - RECTANGLE-Y attribute value. **W** - RECTANGLE-W attribute value. **H** - RECTANGLE-H attribute value.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

73 Toolbar Attributes Window

- Entries 422

Accessible Using

1 "Dialog > Toolbar"; or

2 first check the "Toolbar" entry in the dialog attributes window, then double-click on the dummy toolbar in the dialog; or select 'Toolbar...' from the toolbar's context menu.

3 CTRL+ALT+T.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Name	Handle name of the toolbar (may be overwritten with another name).
Position	TOOLBAR-POS attribute values.
Wrapped	STYLE attribute value: if set and there are more toolbar items than can be displayed on the top of the dialog, the toolbar wraps around to a new line. (The default: the toolbar can be scrolled with the two small arrow push buttons on the left of the toolbar.)
Margin-X	MARGIN-X attribute value (specifies which margin to the left, to the right and above the bitmaps is displayed in the toolbar area. This attribute only applies if TOOLBAR-POS is set to TB-LEFT or TB-RIGHT.
Margin-Y	MARGIN-Y attribute value (specifies which margin to the left, above and below the bitmaps is displayed in the toolbar area. This attribute only applies if TOOLBAR-POS is set to TB-TOP or TB-BOTTOM.
Item width	ITEM-W attribute value (specifies the width of all items in the toolbar).
Item height	ITEM-H attribute value (specifies the height of all items in the toolbar).
Toolbar	Displays the handle name and the BITMAP-FILE-NAME of the existing toolbar items.
Selected toolbar item:	In this group frame, you assign attribute values to the toolbar item selected in the "Toolbar items" group frame on the left.
Name	Handle name of the toolbar item (may be overwritten with another name).
Type	MENU-ITEM-TYPE attribute value for the selected toolbar item.
Same as	SAME-AS attribute value (not for MENU-ITEM-TYPE attribute "Separator"); the selection box displays the menu items available.
Bitmap	BITMAP-FILE-NAME attribute value.
. . .	Dialog box for determining sources of BITMAP-FILE-NAME attribute values. Also provides a list of all available bitmaps to be used.
DIL text	DIL-TEXT attribute value (string).

Entry in Attributes Window	Represents
. . .	Dialog box for determining sources of DIL-TEXT attribute values (not for MENU-ITEM-TYPE attribute "Separator").
Accelerator	ACCELERATOR attribute value.
. . .	Dialog box for determining sources of ACCELERATOR attribute values.
Command ID	CLIENT-KEY attribute value (used in this context for associating a command ID).
Background Color:	
Selection box	BACKGROUND-COLOUR-NAME attribute value to be used for display of the item's bitmap (if any). If 'default' is specified, the color of the first (top-left) pixel in the bitmap determines the background color.
...	Dialog box for editing BACKGROUND-COLOUR-VALUE attribute value.
State:	
Visible	VISIBLE attribute value.
Enabled	ENABLED attribute value (not for MENU-ITEM-TYPE attribute "Separator").
Checked	CHECKED attribute value (not for MENU-ITEM-TYPE attribute "Separator").
Shared	SHARED attribute value. CLICK events for this item will be forwarded to the active MDI child dialog (if any). This attribute is ignored for non-MDI dialogs.
Style:	
Scaled	STYLE attribute value: allows for stretched bitmaps to be displayed on the toolbar items.
Transparent	STYLE attribute value: bitmap pixels in the background color do not change the state of the screen.
Events	Dialog box for editing event handlers; may only be used with the appropriate "Type" entry; may not be used if the toolbar item is associated with a menu item using the SAME-AS attribute.
New	Creates a new toolbar item.
Cut	Cuts a selected toolbar item and copies it to the clipboard. You can also cut and paste several toolbar items at once.
Paste	Pastes a toolbar item from the clipboard. **Note:** The "New" and "Paste" entries insert toolbar items behind the currently selected item, or, if no items are selected, at the top of the list. You deselect items by holding down CTRL while selecting the selected items.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

74 Tool Bar Control Attributes Window

- Entries 426

Accessible Using

1 Double-click on the tool bar control; or

2 if selected: "Control > Attributes" or by selecting 'Attributes...' from the control's context menu or

3 if selected: ENTER.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Name	Handle name of the tool bar control (may be overwritten with another name).
Attributes...	Subordinate window for editing the tool bar control's attribute values. For more information, see Tool Bar Control Attributes Subwindow. (Normally, all attributes of a dialog element can be edited in the attributes window. Instead, the attributes of each tool bar item in the tool bar control can be edited here. For reasons of space, the tool bar control's attributes are edited in a separate subwindow).
Tool bar items:	Displays the handle name and the BITMAP-FILE-NAME attribute values of the tool bar items already created. If you select a tool bar item, its attributes are displayed for editing. You can also select several tool bar items for cutting and pasting.
New	Creates a new tool bar item.
Cut	Cuts a selected tool bar item and copies it to the clipboard. You can also cut and paste several tool bar items at once.
Paste	Pastes a tool bar item from the clipboard. **Note:** The "New" and "Paste" entries insert tool bar items behind the currently selected item, or, if no items are selected, at the top of the list. You deselect items by holding down CTRL while selecting the selected items.
Selected tool bar item:	In this group frame, you assign attribute values to the tool bar items selected in the "Tool bar items" list box on the left.
Name	Handle name of the tool bar item (may be overwritten with another name).
Type	MENU-ITEM-TYPE attribute value for the selected tool bar item.
Width	RECTANGLE-W attribute value. This is only available for MENU-ITEM-TYPE attribute "Separator" and specifies the separator width (0 = default separator width).
Same as	SAME-AS attribute value (only available for MENU-ITEM-TYPE attribute "Normal"); the selection box displays the signals and menu items available.
Bitmap	BITMAP-FILE-NAME attribute value.

Entry in Attributes Window	Represents
. . .	Dialog box for determining sources of BITMAP-FILE-NAME attribute values. Also provides a list of all available bitmaps to be used.
DIL text	DIL-TEXT attribute value (string).
. . .	Dialog box for determining sources of DIL-TEXT attribute values (not for MENU-ITEM-TYPE attribute "Separator").
Accelerator	ACCELERATOR attribute value.
. . .	Dialog box for determining sources of ACCELERATOR attribute values.
Tooltip	TOOLTIP attribute value.
. . .	Dialog box for determining sources of TOOLTIP attribute values.
Command ID	CLIENT-KEY attribute value (used in this context for associating a command ID).
Background Color:	
Selection box	BACKGROUND-COLOUR-NAME attribute value to be used for display of the item's bitmap (if any). If 'default' is specified, the color of the first (top-left) pixel in the bitmap determines the background color.
...	Dialog box for editing BACKGROUND-COLOUR-VALUE attribute value.
State:	
Visible	VISIBLE attribute value.
Enabled	ENABLED attribute value (not for MENU-ITEM-TYPE attribute "Separator").
Checked	CHECKED attribute value (not for MENU-ITEM-TYPE attribute "Separator").
Shared	SHARED attribute value. CLICK events for this item will be forwarded to the active MDI child dialog (if any). This attribute is ignored for non-MDI dialogs.
Style:	
Scaled	STYLE attribute value: allows for stretched bitmaps to be displayed on the toolbar items.
Wrapped	STYLE attribute value: if set, tool bar item is started on a new row.
Transparent	STYLE attribute value: bitmap pixels in the background color do not change the state of the screen.
Events	Dialog box for editing event handlers; may only be used with the appropriate "Type" entry; may not be used if the toolbar item is associated with a menu item using the SAME-AS attribute.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

75 Tool Bar Control Attributes Subwindow

- Entries 430

Accessible Using

- mf-screen· Select the "Attributes..." button in the tool bar control attributes window.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Subwindow	Represents
String	STRING attribute value. This is the text displayed in the window caption when a dockable tool bar control is floated.
...	Dialog box for determining sources of STRING attribute values.
Control ID	CLIENT-KEY attribute value (used in this context for associating a user-defined ID).
Context Menu	CONTEXT-MENU attribute value. Specifies the context menu (if any) associated with the control.
Docking	DOCKING attribute value. Determines the sides of the dialog (if any) on which this tool bar is allowed to dock (if dockable). **Note:** The dialog itself must also support docking on the specified side(s).
Location	LOCATION attribute value. Determines the side of the dialog on which the tool bar control is initially positioned, or whether the tool bar control is floated in a separate window (if dockable).
Internal Metrics:	
Item width	ITEM-W attribute value (specifies the width of all items in the toolbar).
Item height	ITEM-H attribute value (specifies the height of all items in the toolbar).
Margin-X	MARGIN-X attribute value. Specifies the margin (in pixels) to the left and right of the tool bar items (for horizontal tool bars) or above and below the tool bar items (for vertical tool bars).
Margin-Y	MARGIN-Y attribute value. Specifies the margin (in pixels) above and below the tool bar items (for horizontal tool bars) or to the left and right of the tool bar items (for vertical tool bars).
Borders:	
Left	STYLE attribute value. Species whether a border should be displayed on the left side of the control. This option is not available for dockable tool bars.
Top	STYLE attribute value. Species whether a border should be displayed at the top of the control. This option is not available for dockable tool bars.

Entry in Attributes Subwindow	Represents
Right	STYLE attribute value. Species whether a border should be displayed on the right side of the control. This option is not available for dockable tool bars.
Bottom	STYLE attribute value. Species whether a border should be displayed at the bottom of the control. This option is not available for dockable tool bars.
Rectangle:	The following four attributes decide the tool bar control's x and y axis position, its height and its width on the screen. **X** - RECTANGLE-X attribute value. **Y** - RECTANGLE-Y attribute value. **W** - RECTANGLE-W attribute value. **H** - RECTANGLE-H attribute value. **Note:** the positions are relative to the dialog window.
Style:	
Gripper	STYLE attribute value. Determines whether a gripper bar is displayed within the tool bar control. **Note:** The gripper bar does not appear if the tool bar is floated.
Flat	STYLE attribute value. Indicates that the tool bar items should be displayed with a flat appearance.
Dynamic	STYLE attribute value. Indicates that the tool bar control can be resized when floated. **Note:** Dynamic tool bars cannot contain any child controls.
3-D border	STYLE attribute value. If set, the tool bar control's border (if any) is drawn with a 3-D appearance.
State:	
Visible	VISIBLE attribute value.
Enabled	ENABLED attribute value (not for MENU-ITEM-TYPE attribute "Separator").
Dockable	DRAGGABLE attribute value. If set, the tool bar control may be docked and/or floated in its own separate window.
Tooltips	HAS-TOOLTIP attribute value. If not set, display of the tool tip text (if any) for the tool bar items will be suppressed.
Flyby text	HAS-DIL attribute value. If not set, display of the DIL text (if any) for the tool bar items will be suppressed.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

76 Tree View Control Attributes Window

- Entries 434

Accessible Using

1 Double-click on the tree view control; or

2 if selected: "Control > Attributes" or by selecting 'Attributes...' from the control's context menu.

3 if selected: ENTER.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Name	Handle name of the tree view control (may be overwritten with another name).
Attributes...	Subordinate window for editing the tree view control's attribute values. For more information, see *Tree View Control Attributes Subwindow*.
Nodes:	Displays the labels of the tree view items already created. If you select an item in this list, its attributes are displayed for editing. You can also select several items for cutting and pasting.
<<Parent node Subnodes>>	These two push buttons enable you to navigate to the next higher level (<< Parent node) or the next lower level (Subnodes >>) relative to the currently displayed branch.
New	Creates a new item.
Cut	Cuts a selected item and copies it to the clipboard. You can also cut and paste several items at once.
Paste	Pastes one or more items from the clipboard. **Note:** The "New" and "Paste" entries insert items behind the currently selected item, or, if no items are selected, at the top of the list. You deselect items by holding down CTRL while selecting the selected items.
Selected node:	In this group frame, you assign attribute values to the item selected in the "Nodes" list box on the left.
Name	Handle name of the item (may be overwritten with another name).
String	STRING attribute value. Specifies the item's label.
. . .	Dialog box for determining sources of STRING attribute values.
Format	FORMAT attribute value. Specifies the item's data format type and length.
Edit mask	EDIT-MASK attribute value. Specifies the edit mask (if any) used to display the item's label.
Tooltip	TOOLTIP attribute value. Specifies the tooltip text associated with the item.

Entry in Attributes Window	Represents
Image	IMAGE attribute value. Specifies the image control (if any) containing the image used by the item.
Image index	IMAGE-INDEX attribute value. Specifies the zero-based image number used from the specified image control, if any, or the one-based image number from the control's image list otherwise.
Length	LENGTH attribute value. Specifies the maximum number of label characters that may be entered by the user.
Style:	
Upper case	STYLE attribute value. Label characters entered by the user are automatically converted to upper case.
No check box	STYLE attribute value. Check box is not displayed for this item. **Note:** This option only has an effect if the control's "check boxes" style is specified.
State:	
Checked	CHECKED attribute value. Item is checked. **Note:** Check boxes are only displayed if the control's "check boxes" style is specified.
Modifiable	MODIFIABLE attribute value. Item label is modifiable by the user. **Note:** The control itself must also be MODIFIABLE.
RTL	RTL attribute value. If this entry is checked, the dialog element direction is right-to-left.
Expanded	EXPANDED attribute value. Item's node is expanded.
Sorted	SORTED attribute value. Items are inserted in sorted sequence. **Note:** This option is implicitly set if the control itself is SORTED. The existing items on the same branch must already be in sorted sequence.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

77 Tree View Control Attributes Subwindow

- Entries 438

Accessible Using

- Select the "Attributes..." button in the tree view control attributes window.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Font	Output field where the font currently selected is displayed.
. . .	Dialog box for selecting fonts.
DIL text	DIL-TEXT attribute value.
. . .	Dialog box for determining sources of DIL-TEXT attribute values.
Accelerator	ACCELERATOR attribute value.
Context Menu	CONTEXT-MENU attribute value. Specifies the context menu (if any) associated with the control.
Image List	IMAGE-LIST attribute value. Specifies the image list (if any) used to provide the images used by the control's items.
Background:	
Selection box	BACKGROUND-COLOUR-NAME attribute value.
. . .	Dialog box for editing BACKGROUND-COLOUR-VALUE attribute value.
Foreground:	
Selection box	FOREGROUND-COLOUR-NAME attribute value.
. . .	Dialog box for editing FOREGROUND-COLOUR-VALUE attribute value.
Help ID	HELP-ID attribute value.
Item height	ITEM-H attribute value. Specifies the control's item (row) height.
Indentation	SPACING attribute value. Specifies the indentation of child item nodes relative to their parent.
Drag mode	DRAG-MODE attribute value. Indicates whether the control can act as the source in a drag-drop operation and, if so, which types of drag-drop operation it supports.
Drop mode	DROP-MODE attribute value. Indicates whether the control can act as the target in a drag-drop operation and, if so, which types of drag-drop operation it supports.
Style:	
3-D border	STYLE attribute value. Control has sunken appearance. **Note:** The control does not have a sunken appearance if Windows XP styles are active.

Entry in Attributes Window	Represents
+/- buttons	STYLE attribute value. Displays plus (+) and minus (-) buttons next to parent items. **Note:** These buttons are only shown for root items if the "Lines at root" style is also specified.
Lines	STYLE attribute value. Lines are used to show the item hierarchy.
Lines at root	STYLE attribute value. Lines are used to link items at the root of the item hierarchy.
No scroll	STYLE attribute value. Scroll bars are not displayed.
Single expand	STYLE attribute value. Column header not displayed in report view mode.
Dbl. click expand	STYLE attribute value. Items are automatically expanded when selected, and automatically collapsed when deselected.
Check boxes	STYLE attribute value. Displays check boxes next to the items.
Full row select	STYLE attribute value. Selection emphasis extends across the entire control.
No hide selection	STYLE attribute value. Keeps selection visible even when control no longer has the focus.
Hot-track select	STYLE attribute value. Automatically selects an item when the mouse cursor hovers over it for a short time.
Size to parent	STYLE attribute value. Control size automatically maintained to fill the client area of its parent.
State:	
Visible	VISIBLE attribute value. Control will be shown.
Enabled	ENABLED attribute value. Control can accept user input.
Modifiable	MODIFIABLE attribute value. Item labels may be edited.
RTL	RTL attribute value. If this entry is checked, the dialog element direction is right-to-left.
Sorted	SORTED attribute value. Items are initially sorted in alphabetic sequence.
Tooltips	HAS-TOOLTIP attribute value. If not set, display of the tool tip text (if any) for the items will be suppressed.
Rectangle:	The following four attributes decide the input field control's x and y axis position, its height and its width on the screen. **X** - RECTANGLE-X attribute value. **Y** - RECTANGLE-Y attribute value. **W** - RECTANGLE-W attribute value. **H** - RECTANGLE-H attribute value.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

78 Wallpaper Attributes Window

- Entries 442

Accessible Using

1 "Dialog > Dialog Wallpapers"; or

2 CTRL+ALT+W.

Entries

Note: For context-sensitive help on attribute entries, select the entry so it has the focus, and press F1.

Entry in Attributes Window	Represents
Wallpapers:	Displays the handle name of the wallpapers already created. If you select a wallpaper in the list, its attributes are displayed for editing. You can also select several wallpapers for cutting and pasting.
New	Creates a new wallpaper.
Cut	Cuts the selected wallpaper and copies it to the clipboard. You can also cut and paste several wallpapers at once.
Copy	Copies the selected wallpaper(s) to the clipboard.
Paste	Pastes a wallpaper from the clipboard. **Note:** The "New" and "Paste" entries insert wallpapers behind the currently selected wallpaper, or, if no wallpapers are selected, at the top of the list. You deselect items by holding down CTRL while selecting the selected items.
Selected wallpaper:	In this group frame, you assign attribute values to the wallpapers selected in the "Wallpapers" list box on the left.
Name	Handle name of the wallpaper (may be overwritten with another name).
Type	MENU-ITEM-TYPE attribute value for the selected signal.
Bitmap	BITMAP-FILE-NAME attribute value. Specifies the wallpaper image.
. . .	Dialog box for determining sources of BITMAP-FILE-NAME attribute values. Also provides a list of all available bitmaps to be used.
Background:	
Selection box	BACKGROUND-COLOUR-NAME attribute value. If 'default' is specified, the color of the first (top-left) pixel in the bitmap determines the background color. Pixels matching the background color are not drawn if the "transparent" style is set.
...	Dialog box for editing BACKGROUND-COLOUR-VALUE attribute value.
Blend factor	BLEND attribute value: Alpha-blending factor in percent (0 = opaque wallpaper; 100 = fully transparent wallpaper).
State:	
Visible	VISIBLE attribute value. If checked, the wallpaper is shown.

Entry in Attributes Window	Represents
Style:	
Pattern	STYLE attribute value; the wallpaper image is tiled.
Transparent	STYLE attribute value; pixels in the background color appear transparent.
Vertical Position:	STYLE attribute values specifying the vertical alignment of the wallpaper image in the host window. These styles are not used if the "pattern" style is set.
Top	Top alignment.
Center	Vertical center alignment.
Bottom	Bottom alignment.
Horizontal Position:	STYLE attribute values specifying the horizontal alignment of the wallpaper image in the host window. These styles are not used if the "pattern" style is set.
Left	Left alignment.
Center	Horizontal center alignment.
Right	Right alignment.
OK	Save settings and exit the window.
Cancel	Exit the window without saving the settings.
Help	Provides online help on the attributes window.

79 Dialog Boxes

- Array
- Data Area - Local, Parameter
- Data Area - Global
- Dialog Compile Error
- Events
- Import Data Field
- Font
- Source
- Subroutines

80 Array

- Purpose 448
- Entries 448

Accessible Using

1 First open the attributes window of a dialog or dialog element by double-clicking on it or by pressing ENTER or by selecting 'Attributes...' from the dialog or dialog element's context menu.

2 Then select the "Array..." push button.

Purpose

Define an array of dialog elements of the same type. This is especially useful for quickly creating a layout for end user input. An array of dialog elements will be treated as an entity by the dialog editor, that is, you can edit the entire array (move, resize, etc.). For example, you can create a column of evenly spaced input field controls plus a column of corresponding text constant controls.

Entries

Entry	Function
Dimensions	None means there will be no array, one means there will be a row or a column, two means there will be an array with both an x and a y axis.
Bounds	Dialog elements on the first and second axis from occurrence to occurrence.
Spacing	Number of pixels between occurrences aligned on the x and y axis.
Arrangement	Mutually exclusive options of how to arrange the dialog elements; the last axis is either the horizontal or the vertical one.

81 Data Area - Local, Parameter

- Purpose 450

Accessible Using

1 "Dialog > Parameter Data Area/Local Data Area"; or

2 CTRL+ALT+P/L, or (for LDAs) by selecting 'Local Data Area...' from the dialog's context menu.

Purpose

Enter inline data definitions for a dialog. In a parameter data area, you must include all the parameters that you want to be passed on to the current dialog in an `OPEN DIALOG` or `SEND EVENT` statement. In a local data area, you must include all the user-defined variables or other variables that you want to use in an event handler code section or a subroutine of the current dialog. Note that the dialog editor automatically generates the data definitions for the dialog elements.

The "Using" button opens a dialog box that allows you to include existing inline data definitions.

82 Data Area - Global

- Purpose 452

Accessible Using

1 "Dialog > Global Data Area"; or

2 CTRL+ALT+G.

Purpose

Select an existing global data area from a list of available global data areas. To select, click on the entry in the list box. The data area is then displayed in the input field. To select, you can also enter the name of a global data area in the input field.

To create a new global data area, you use the data area editor.

83 Dialog Compile Error

Appears When

You check/run/stow a dialog, the compiler finds an error in the dialog's generated code, and you select "Edit" in the "Error" dialog box.

Purpose

Describes the error and lists the line of generated code together with the line number. If you press the OK push button, the section of the dialog appears where the compiler has located the error.

If you have saved your dialog sources in non-enhanced format and the enhanced listing option is enabled, any Natural error message will contain an incorrect line number. To ensure that you get the correct line number, disable the enhanced listing option. As under Natural for Windows and Linux, dialog sources are always saved in enhanced format, this line number inconsistency does not exist.

To disable enhanced dialog list mode

- From the "Options" menu, choose "Enhanced dialog list mode".

  The menu item no longer has a check mark. This indicates the option is disabled.

84 Events

- Purpose 456
- Entries 456

Accessible Using

1 Select the "Events..." push button in an attributes window; or

2 "Dialog > Event Handlers" for dialog events; or

3 CTRL+ALT+E or SHIFT+ENTER for dialog events; or

4 "Control > Event Handlers" for a selected dialog element; or

5 CTRL+SHIFT+E or SHIFT+ENTER for a selected dialog element.

Purpose

Enter Natural event handler code for those events that are provided for the dialog or dialog element; also allows you to enter event handler code for user-defined SEND EVENTs.

Entries

Entry	Function
Event Name	This selection box lists the names of the system-provided events, such as the CLICK EVENT; it also lists the names of the user-written events that can be triggered by specifying SEND EVENT *user-written-event-name*. Please note that *user-written-event-names* are limited to 32 characters and that the option only applies to dialog events.
Editor	Invokes the program editor for the currently displayed event. Before using the program editor the dialog box must be closed using the 'OK' push button.
Rename	(Only applies to dialog events). This push button opens a dialog box where you can rename a user-written event.
New	(Only applies to dialog events). This push button opens a dialog box where you can enter the name of a new, user-written event.
Clear	(Only applies to dialog events). This push button opens a message box where you can specify whether you want to delete the code of a system-provided event or the code and the name of a user-written event.
Use	This push button opens a dialog box where you can select a subprogram or a subroutine by choosing an item from a list of objects or by entering the object name in the input field. Depending on whether it is a subprogram or a subroutine, you get a display of whether the CALLNAT or the PERFORM statement will be used. After having selected the subprogram or subroutine, you leave the dialog box by choosing OK. The subprogram or subroutine will be used by your current event handler code section. At the position where you left the event handler section, you will find the CALLNAT or PERFORM statement with the name of the object.
Suppress	Suppresses an event for which a corresponding SUPPRESS-*eventname*-EVENT attribute exists. The event is also suppressed if you leave the event handler section empty.

Entry	Function
Event Info...	(Only applies to ActiveX control events): Provides information on the parameters of each event.
(Edit area)	Here you enter your Natural code that you want to be triggered when the event occurs.
OK	Saves the code (and name) of the event handler section and exits the dialog box.
Cancel	Exits the dialog box without saving the settings.
Help	Provides online help.

85 Import Data Field

- Purpose 460
- Entries 460

Accessible Using

- "Insert > Import > Input Field/Selection Box".

Purpose

Create an input field control or a selection box control based on a data field from another Natural object in another Natural library. The dialog element is created with a linked variable as the source of its STRING attribute value. You must declare this linked variable in a data area of the dialog.

Entries

Entry	Function
Library	Selection box where you can select the library containing the Natural object with the data field of your choice.
Type	Mutually exclusive options for Natural object types.
Object List	Once you have chosen a library and an object type, all objects with these criteria will be displayed here.
Data fields	Once you have chosen an object from the object list, all data fields defined in this object will be displayed.
Import	Once you have chosen one or several data field(s), you push this button and its content will be imported into your input field control or selection box control.
Cancel	Exits the dialog box without saving the settings.
Help	Provides online help.

86 Font

- Purpose 462

Accessible Using

1 First open the attributes window of a dialog or dialog element by double-clicking on it or by pressing ENTER.

2 Then select the "..." push button next to the font selection box.

Purpose

Select a font type, such as "Times New Roman", a font face, such as "bold", and a font size, such as "10". After selecting your font, a sample is displayed. When you choose OK, a font control is generated and assigned to the FONT-HANDLE attribute of the dialog element you are currently editing.

87 Source

- Purpose 464

Accessible Using

1 First open the attributes window of a dialog or dialog element by double-clicking on it or by pressing ENTER.

2 Then select the "..." push button to the left of an attribute entry.

Purpose

Define the source of attribute values, for example for the STRING attribute.

Entry	Function
Value	Current attribute value; the name of this entry varies depending on the attribute source.
. . .	Opens the standard file selection dialog. This button is only available if "Attribute Source" is set to "Constant".
Attribute Source:	
Constant	Text string.
Message file	Number of the string in the message file. If you have specified an array of dialog elements, the number of the first string appears here. The number of the string for each occurrence in the array is generated in ascending order from the first string number onwards.
Variable	When a dialog is opened with an OPEN DIALOG statement, the content of this variable will be assigned to the attribute. You can dynamically change the content of this variable in the before-open event handler.
	For more information, see *Message Files and Variables as Sources of Attribute Values*.
Linked variable	Only applicable to input field controls and selection box controls. The input of an end user will automatically be moved to this variable when the dialog element is left. When you have changed the content of a linked variable dynamically (during processing of an event handler section), you can use the PROCESS GUI statement action REFRESH-LINKS and the refreshed variable content will be displayed in the input field control or selection box control.
Array Values:	Attribute values if there is an array of dialog elements.
Individual values	Each occurrence in the array will have its individual attribute value.
Repeated single value	All occurrences in the array will have the same attribute value.
OK	Saves the settings and exits the dialog box.
Cancel	Exits the dialog box without saving the settings.
Help	Provides online help.

88 Subroutines

- Purpose 466
- Entries 466

Accessible Using

1 "Dialog > Inline subroutines"; or

2 CTRL+ALT+S.

Purpose

Enter standard sections of Natural code to be used in several event handler sections.

Entries

Entry	Function
Subroutine name	This selection box lists the names of the existing subroutines for the dialog.
Editor	Invokes the program editor for the currently displayed subroutine. Before you use the program editor, the dialog box must be closed using the 'OK' push button.
Rename	This push button opens a dialog box where you can rename a subroutine.
New	This push button opens a dialog box where you can enter the name of a new subroutine. Subroutine names specified in the dialog editor are limited to 120 characters. The first 32 characters must be unique.
Delete	This push button opens a message box where you can specify whether you want to delete the code and the name of a subroutine.
Action:	Here you enter your Natural code in free form, that is, without having to specify the DEFINE SUBROUTINE and END-SUBROUTINE statements.
OK	Saves the code and name of the subroutine and exits the dialog box.
Cancel	Exits the dialog box without saving the settings.
Help	Provides online help.

89 Enhanced Source Code Format

- Syntax Conventions 468
- How Natural Dialogs Work 468
- Syntax 469

This section describes the syntax conventions for entering code in the source code window provided by the program editor.

Syntax Conventions

Syntax is described using the following meta-notation:

```
syntax_element_name :=: description
```

Syntax Element	Represents
syntax_element_name :=:	Identifies the construct whose structure is defined by *description*.
description	Program code displayed in UPPER-CASE letters.

The Syntax Symbols used in this section are explained in the *Statements* documentation. In addition, the following symbols are used within the diagrams:

Symbol	Represents
{*element1* | *element2*}	Elements contained within braces and separated by vertical bars indicate that exactly one of the elements must be specified.
< >	Pairs of angle brackets indicate that code is separated into multiple lines. Each pair of brackets represents the end of a Natural source line.
/*[and **/*]**	These bracket pairs form a collapsible block in list mode, even in user code sections.

Line-number references are explicitly *not* supported within dialogs, as the dialog editor will generate line numbers for dialog sources without consideration of any line-number references.

Line length and source size are subject to the general limits imposed by Natural.

Note: Where syntax elements are shown separated by blanks, blanks are required, but the exact number of blanks is not significant.

How Natural Dialogs Work

This section gives you some background information so you can better understand what the various elements of the dialog source do.

A Natural dialog is an executable module. You invoke it either directly from the Natural environment or from another executing module with an `OPEN DIALOG` statement. Invoking the dialog causes the dialog's executable to be instantiated and executed with the system variables `*CONTROL` equal to `NULL-HANDLE` and `*EVENT` equal to `OPEN`. This `OPEN` event is processed by calling the inline

subroutine *DLG$SUBR$CREATE$WINDOW which creates the dialog window. Window creation invokes the dialog with *CONTROL equal to NULL-HANDLE and *EVENT equal to AFTER-OPEN, to which the dialog responds by calling the inline subroutine *DLG$SUBR$CREATE$CONTROLS. This subroutine creates any dialog elements defined for the dialog, and then calls PROCESS GUI ACTION AFTER-CREATION to notify Natural that window creation is complete.

Depending on further application processing and user input, the dialog will repeatedly be executed with *CONTROL and *EVENT set appropriately. The dialog remains instantiated until it is explicitly unloaded as a result of the window being destroyed. This is the default reaction to the CLOSE event which in turn occurs as a result of either end-user interaction or the CLOSE DIALOG statement.

The same dialog module may be instantiated more than once, resulting in more than one dialog and its window and dialog elements being active. Each dialog has a unique numeric ID which is available as *DIALOG-ID during execution of that dialog, and as *window_handle*.CLIENT-DATA wherever that dialog's window handle is accessible (the window must be created with that attribute setting).

Syntax

- General Syntax
- Subsections of the General Syntax
- Subordinate Syntax Sections

General Syntax

This is the complete source of a dialog.

```
dialog_source_22D
 :=:
```

```
/** DIALOG SOURCE 22D dialog_info_section_22D
[dialog_options_section_22D]
dialog_data_section_22D [user_subroutines_section_22D]
window_definition_22D control_definitions_22D
default_handler_section_22D
error_handler_section_22D before_any_section_22D
event_handlers_22D after_any_section_22D
END /** END-DIALOG-SOURCE
```

Subsections of the General Syntax

dialog_info_section_22D

```
dialog_info_section_22D
 :=:
```

The dialog's "info" section, consisting of a banner line, frame gallery information,

and an optional comment.

```
/*[ DEFINE DIALOG INFO
```

The following line is always generated by the dialog editor, but ignored on input.

```
/*D* Natural Dialog Description version_string / date
 time
```

One or more following lines are present in dialogs generated using the frame gallery.

If present, they are preserved by (but cannot be changed in) the dialog editor.

```
[/*DF frame_gallery_info]...
```

The empty comment line is generated by the dialog editor if no user comment is

present, but ignored on input.

```
[dialog_comment :=: {/** EMPTY DIALOG COMMENT | ↩
[user_code_line_protected_by_/**_prefix]...}]
 /*] END-DIALOG-INFO
```

dialog_options_section_22D

```
dialog_options_section_22D
 :=:
```

These option settings are only generated by the dialog editor if the "Save settings

with dialog" option is on (see *Dialog Editor Options* and *Setting the Options* in the *Using Natural Studio* documentation). When a dialog is being loaded and this section is present,

the option is turned on.

```
[/*[ DEFINE OPTION SETTINGS {/** SET option_name option_value}...
 /*] END-OPTION-SETTINGS]
```

dialog_data_section_22D

```
dialog_data_section_22D
 :=:
```

The dialog's data section, consisting of a global, parameter, and local data area.

The sequence is fixed.

```
 DEFINE DATA gda_section_22D
 pda_section_22D lda_section_22D
 END-DEFINE
```

Subsections of the dialog_data_section_22D

gda_section_22D

```
gda_section_22D
 :=:
```

The optional global data area specification. Note that it may reference blocks.

```
/*[ DEFINE GLOBAL DATA [GLOBAL USING gda_specification] /*] END-GLOBAL-DATA
```

pda_section_22D

```
pda_section_22D
 :=:
```

The dialog's parameter data area must always contain the parent handle as the

first field.

```
/*[ DEFINE DIALOG PARAMETERS PARAMETER 01 #DLG$PARENT HANDLE OF
 GUI BY VALUE user_data_section_22D
 /*[ DEFINE USING [PARAMETER USING pda_name]... /*] END-USING /*] ↩
END-DIALOG-PARAMETERS
```

lda_section_22D

```
lda_section_22D
 :=:
```

The dialog's local data area. The handle declarations for all dialog elements are

necessary if the dialog is to be compiled, but are ignored by the dialog editor

on input and re-generated from the actual dialog element definitions below.

```
/*[ DEFINE LOCAL DATA LOCAL /*[ DEFINE HANDLES {01 control_name
 HANDLE OF control_class[<>]}... /*] END-HANDLES user_data_section_22D
 /*[ DEFINE USING [LOCAL USING lda_name]... /*] END-USING /*] END-LOCAL-DATA
```

user_subroutines_section_22D

```
user_subroutines_section_22D
 :=:
```

The list of user-defined subroutines. The enclosing pseudo-syntax is generated

only if there actually are subroutines.

```
/*[ DEFINE SUBROUTINES [DEFINE SUBROUTINE subroutine_name
 user_code_section_22D END-SUBROUTINE]...
 /*] END-SUBROUTINES
```

window_definition_22D

```
window_definition_22D
 :=:
```

The window is defined within a standard subroutine (this must always be present).

```
DEFINE SUBROUTINE #DLG$ SUBR$CREATE$WINDOW /** DEFINE CONTROL window_name
 non_array_control_definition_22D
  END-SUBROUTINE /** END-CONTROL
```

control_definitions_22D

```
control_definitions_22D
 :=:
```

All dialog elements are defined within one standard subroutine (this must always be present). The sequence generated by the dialog editor is: menu bar, tool bar, font controls, timers, any other dialog elements.

```
DEFINE SUBROUTINE #DLG$SUBR$CREATE$CONTROLS /** DEFINE DIALOG
 ELEMENTS [control_definition :=:
```

Each dialog element definition is enclosed in pseudo-comments.

```
/*[
 DEFINE CONTROL control_name[array_bounds]
```

The following optional comment appears in the source code only.

```
[control_comment
 :=: {user_code_line_protected_by_/**_prefix}...]
```

The following code creates the dialog element.

```
{non_array_control_definition_22D
 | array_control_definition_22D}
```

List box controls and selection-box controls may have items lists defined as one

array of dialog elements. These follow the dialog element creation code.

```
[control_items :=: /* DEFINE ITEMS control_name_array_bounds
 array_control_definition_22D
 /*] END-ITEMS ] /*] END-CONTROL
```

```
]...
 END-SUBROUTINE /** END-DIALOG-ELEMENTS
```

default_handler_section_22D

```
default_handler_section_22D
 :=:
```

The DEFAULT event handler, specified as subroutine. For compilation, it must be

present, as this subroutine is called from various places.

```
DEFINE SUBROUTINE
 #DLG$HANDLER$DEFAULT /** DEFINE EVENT DEFAULT user_code_section_22D
  END-SUBROUTINE /** END-EVENT
```

error_handler_section_22D

```
error_handler_section_22D
 :=:
```

The ERROR event handler, specified as ON ERROR section. Optional.

```
ON ERROR /** DEFINE
 EVENT ERROR user_code_section_22D
 END-ERROR /** END-EVENT
```

before_any_section_22D

```
before_any_section_22D
 :=:
```

The BEFORE-ANY event handler, that is, the code which precedes the DECIDE statements which evaluate *CONTROL and *EVENT.

```
/*[ DEFINE EVENT BEFORE-ANY user_code_section_22D
 /*] END-EVENT
```

event_handlers_22D

```
event_handlers_22D
 :=:
```

The DECIDE statements which evaluate first *CONTROL, then *EVENT, activating the appropriate event handlers.

```
DECIDE
 ON FIRST *CONTROL /** DEFINE ALL EVENTS {dialog_events :=: /*[ DEFINE
 EVENTS FOR DIALOG VALUE NULL-HANDLE
```

*CONTROL = NULL-HANDLE indicates an event associated either with the window or with the dialog itself.

```
DECIDE ON FIRST *EVENT /*[
 DEFINE EVENT OPEN VALUE 'OPEN'
```

The OPEN event handler starts with any BEFORE-OPEN user code and ends with the window creation call.

```
user_code_section_22D
 PERFORM #DLG$SUBR$CREATE$WINDOW /*] END-EVENT /*[ DEFINE EVENT AFTER-OPEN VALUE
 'AFTER-OPEN'
```

The AFTER-OPEN event occurs while the window creation call is being processed, that is, it is a nested call of the dialog. It includes the creation of all dialog elements and the assignment of those window attributes which use dialog element handles (for example, the handle of the menu bar). User code may follow.

```
PERFORM #DLG$SUBR$CREATE$CONTROLS [extra_window_attributes_22D]
 PROCESS GUI ACTION AFTER-CREATION WITH window_name PROCESS GUI ACTION
 RESET-ATTRIBUTES
```

The user's after-open code follows.

```
user_code_section_22D
 /*] END-EVENT /*[ DEFINE EVENT CLOSE
```

The CLOSE event occurs either because the end user directly closes the window, or because the parent window is closed, or as the result of a CLOSE DIALOG statement.

```
VALUE 'CLOSE'
 {dialog_close_handler_22D :=: user_code_section_22D}
 ↩
```

The following call "destroys" the window and unloads the dialog.

```
PROCESS GUI ACTION DELETE-WINDOW WITH window_name
```

The following statement leaves the dialog, bypassing the AFTER-ANY handler.

```
ESCAPE ROUTINE IMMEDIATE /*] END-EVENT
```

The following dialog event handlers are optional and include any user-defined

events.

```
[event_handler_section_22D...]
 NONE PERFORM #DLG$HANDLER$DEFAULT END-DECIDE /*] END-EVENTS
```

```
}
```

All event handlers for dialog elements defined in

*DLG$SUBR$CREATE$CONTROLS are listed below.

```
[control_events_section_22D...]
 NONE PERFORM #DLG$HANDLER$DEFAULT END-DECIDE /** END-ALL-EVENTS
```

All events that are not processed and have not been suppressed are handled in the

DEFAULT event handler. Any dialog elements created outside

DLG$SUBR$CREATE$CONTROLS are handled there.

after_any_section_22D

```
after_any_section_22D
 :=:
```

The AFTER-ANY event handler, that is, the code which follows the DECIDE

statements evaluating *CONTROL and *EVENT.

```
/*[ DEFINE EVENT AFTER-ANY user_code_section_22D
 /*] END-EVENT
```

Subordinate Syntax Sections

user_data_section_22D

```
user_data_section_22D
 :=:
```

A section with data declarations. The dialog editor generates a comment if no user code is present. The user code layout is preserved, except that the whole section is indented appropriately.

```
[frame_code_section_22D]
 {/** EMPTY USER CODE SECTION | user_code_line_22D...}
 [frame_code_section_22D]
```

user_code_section_22D

```
user_code_section_22D
 :=:
```

A section with executable statements. The dialog editor generates a comment if no user code is present. The user code layout is preserved, except that the whole section is indented appropriately.

```
[frame_code_section_22D]
 {;/** EMPTY USER CODE SECTION | user_code_line_22D...}
 [frame_code_section_22D]
```

frame_code_section_22D

```
frame_code_section_22D
 :=:
```

This protected section is code in a frame gallery dialog. Do not change this code.

```
/*[ DEFINE FRAME CODE [user_code_line_22D...]
 /*] END-FRAME-CODE
```

user_code_line_22D

```
user_code_line_22D
 :=:
```

A code line within an event section or subroutine. Indentation is preserved by the dialog editor. However, a minimum indentation is enforced.

```
indented_code_line
```

event_handler_section_22D

```
event_handler_section_22D
 :=:
```

A single event handler section, that is, the VALUE clause for the event in a DECIDE statement for the relevant dialog element.

```
/*[ DEFINE EVENT event_name
 VALUE 'event_name' user_code_section_22D
  /*] END-EVENT
```

control_events_section_22D

```
control_events_section_22D
 :=:
```

The collection of all event handlers for one dialog element, that is, a VALUE clause for the dialog element's handle containing a DECIDE statement for *EVENT. If th dialog element is an array, all elements are handled in this section.

```
/*[
 DEFINE EVENTS FOR control_name VALUE control_name[(*[,*)] DECIDE
 ON FIRST *EVENT event_handler_section_22D...
 NONE PERFORM #DLG$HANDLER$DEFAULT END-DECIDE /*] END-EVENTS
```

non_array_control_definition_22D

```
non_array_control_definition_22D
 :=:
```

A non-array dialog element definition corresponds to the PROCESS GUI statement action that creates the dialog element. As the WITH PARAMETERS ... END-PARAMETERS clause is used, all attributes not mentioned have their default values. The dialog editor does not generate such attributes.

The first and the second attributes must be HANDLE-VARIABLE and TYPE.

The dialog editor uses the predefined attribute value names defined in the standard local data area NGULKEY1 if this data area has been included in the dialog. If it is not included, the dialog cannot be compiled. Note that variable references are allowed for only a subset of the attributes.

```
PROCESS GUI ACTION ADD WITH
 PARAMETERS {attribute_name = {constant | variable_reference}<>}...
 END-PARAMETERS [frame_code_line_22D...] /*] END-CONTROL
```

array_control_definition_22D

```
array_control_definition_22D
 :=:
```

An array definition of a dialog element consists of one PROCESS GUI statement for each array element. Only a subset of dialog elements may be defined as arrays (for example, not the dialog window or list box controls). Instead of the WITH PARAMETERS ... END-PARAMETERS clause, explicit attribute assignments and

the simple WITH clause are used, as array elements will share many equal but non-default attribute values.

The dialog editor is quite restrictive with respect to dialog element arrays: a maximum of two dimensions is allowed; elements cannot be placed individually; only a subset of the attributes may vary between elements.

The dialog editor will, when scanning dialog element array definitions, take those attribute values which may not vary from the first element's definition. Specifically, all coordinate definitions but the first will be derived from the pseudo-attributes H-SPACING, V-SPACING, and ARRANGE-IN-COLUMNS. (Note, however, that not all dialog elements have coordinates.) All attributes not mentioned explicitly have their default values; this is enforced by the initial PROCESS GUI ACTION RESET-ALL.

```
PROCESS GUI ACTION RESET-ATTRIBUTES [spacing_info:=:
 /** H-SPACING = horiz_spacing /** V-SPACING = vert_spacing /**
 ARRANGE-IN-COLUMNS = {TRUE|FALSE}<>] ] {
```

Creation of one array element. Numeric references to a message file string must be handled in an explicit PROCESS GUI statement action for that string, as direct assignment to a number would simply convert that number to a string.

```
[attribute_assignment:=:{
  control_name_and_index.attribute_name := attribute_value|
 PROCESS GUI ACTION GET-MESSAGE-TEXT WITH number indexed_control_name
 }<>]... PROCESS GUI ACTION ADD WITH parent_name type_name control_name_and_index
```

```
}...
```

extra_control_attributes_22D

```
extra_control_attributes_22D
 :=:
```

Some attributes cannot be defined meaningfully in the PROCESS GUI ACTION ADD that creates the dialog element; an example is a non-modifiable selection-box control's STRING, which must be one of the items' STRING values. This attribute is therefore defined after all items have been added.

```
control_name_and_index.attribute_name
 := attribute_value
```

extra_window_attributes_22D

```
extra_window_attributes_22D
 :=:
```

Some attributes cannot be defined meaningfully in the PROCESS GUI ACTION ADD that creates the dialog, for example, the default button can only be defined after that button has been created.

```
window_name.attribute_name
 := attribute_value
```

VII Class Builder

90 Class Builder

- What is the Class Builder? 486
- Class Builder Interface 488
- Class Builder Nodes 499
- Node Properties 509
- Adding Class Components 517
- Renaming Class Components 522
- Removing Class Components 522
- Editing Class Components 523
- Using Interfaces from Several Classes 524
- Locking Concept 526
- Tutorial 527
- Glossary 530

The Class Builder is a tool which can be used to display a Natural class in a structured hierarchical order, and also to manage the class and its components efficiently.

A Natural class can be composed of various components: “real” Natural objects (for example, an object data area) or objects which exist only in the class source (for example, interface components).

The Class Builder represents each component of the class in the form of a node. By selecting these nodes, the class and its components can be managed in a context-sensitive manner.

The *Class Builder* documentation explains how to create and modify a Natural class with the Class Builder. Please refer to *Defining Classes* in *Introduction to NaturalX* (*Programming Guide*) to become acquainted with the general usage of Natural classes.

What is the Class Builder?

The Class Builder provides the following features:

- It is fully integrated in the general Natural user interface.
- The components of a class are displayed as nodes in the same way as Natural modules. Every type of node has a special icon assigned which provides detailed information for that component.
- Natural objects which are used by a class (for example, ODA), can be managed (edit, stow, ...) by the Class Builder.
- Class and interface GUIDs (Global Unique IDs) are generated and hidden.
- Class comments (one comment for every class component) can be created and changed by the Class Builder.
- The class source is generated automatically.

Which Classes can be handled by the Class Builder?

The Class Builder can manage any syntactically correct class. Even if it is possible to change the class source with the program editor as well, the Class Builder is the recommended editor for changing classes. Please note that a class, which has been changed with the program editor and saved with syntax errors, can no longer be opened using the Class Builder.

The class syntax is highly “flexible”, i.e., it is possible to obtain the same runtime behavior with different syntax constructs. This was important for earlier Natural versions, because the user had to type all class code himself. With the Class Builder, this is no longer necessary; the Class Builder will generate the class code and create Natural objects, which are used by the class. The Class Builder will generate only the most reasonable code.

For this reason, the following features are not supported by the Class Builder:

- **create a new GUID LDA:**
  The Class Builder generates a GUID for the class and the interfaces of the class. If you want to define the GUID yourself, you must create a LDA outside of the Class Builder and then link it to the class.
- **create new inline data definitions:**
  The Class Builder only provides for the creation of new data areas. This is because data definitions are usually used in several places (for example, method parameter in class and method subprogram) and it is fault-prone if the same inline data definitions have to be used more than once.
- **use data from inline data definitions for assignments in the Class Builder:**
  If data definitions have to be assigned to class components in the case of unique IDs and property implementations, the Class Builder offers a list of all data definitions from the corresponding data areas. Data from inline data definitions will not be included in these lists. This means, for example, that the object data variable which is defined inline cannot be used as property implementation.

Although the Class Builder does not permit the creation of all class syntax constructs, it can nonetheless read existing classes with these constructs and can be used to modify these constructs.

If the Class Builder cannot read a class because it is syntactically incorrect, it displays an error message and activates the program editor. The syntax error must be corrected in the program editor. After the class has been saved, it can be opened with the Class Builder.

Note: If you save a class with the Class Builder, the class source will be generated. This means that any special source formats, such as indentation, will be lost.

When is a Class saved?

When a class is opened in the Class Builder, the contents are read from the class source and stored in an internal structure. If you then change the class, these changes are performed only on the internal structures. The changes are visible in all views of Natural. So, for example, when a new interface is added in the library workspace, a node for this interface will also be created in the "Interfaces" list view of the class. If you want to save your changes, you must execute **Save**, **Save As** or **Stow** for the class.

If you create a new class, this does not automatically create a new class module. This is only done when **Save**, **Save As** or **Stow** is executed for the class. For this reason, a “new” class will not be visible in the File View of the library workspace until it is saved the first time.

If you want to remove the changes which you applied to a class, you can use the **Restore** command. This command will restore the class as it is contained in the class module, i.e., the last saved state.

If Natural is ended and unsaved classes exits, the user will be asked if the classes should be saved.

Class Comments

The Class Builder tries to assign every comment found in the class source to one component of the class. A comment is usually assigned to the following class component. For example a comment which is found before the definition of an interface is taken as comment for this interface.

The comments can be changed and created via the **Properties** menu item, which is available for all class component nodes. For more information, see *Node Properties*.

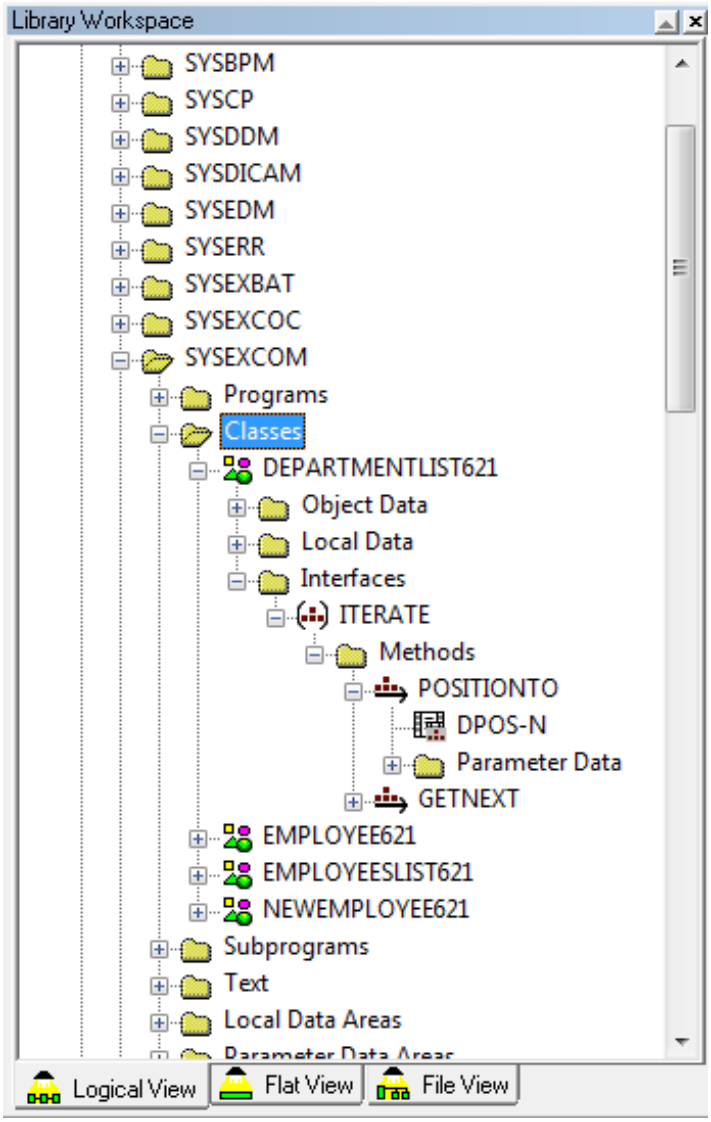
Note: If a class is read by the Class Builder for the first time, it is possible that the Class Builder assigns the comments to a component other than the one the user expects. No comment will be lost when the class is saved, but the user should check if the comments are assigned to the correct components.

When a class is saved by the Class Builder for the first time, all comments will be marked with a special tag. This ensures that the comment assignment is correct when this class is read later by the Class Builder.

Class Builder Interface

The Class Builder is available in the logical and flat view of Natural. It is fully integrated in the general Natural user interface which shows the Natural objects as nodes of a tree or list view.

In a tree view of the Library Workspace or Application Workspace, a class can be "opened" by expanding the class node. The class nodes are grouped hierarchically. For example, the interface is a child of the class node and the method is a child of the interface node. Every class node provides the same features as all other nodes, for example, a context menu which allows node-specific actions. Most of the class nodes that have child nodes can be opened as a list view which displays all children of this node. The List View shows more information about the nodes (for example, the library in which an object data area is located). The list view nodes offer the same context menu as the corresponding tree view nodes.



The following topics are covered below:

- Logical View

- Flat View

Logical View

The class nodes of the logical view are inscribed with the class name, i.e., the name that is used when an object of this class is created with the `CREATE OBJECT` statement.

In the logical view the nodes are, as a basic principle, grouped by their type. This is also valid for the class nodes. Class nodes of the same type are collected under a group node which describes the type with its contents. Therefore, all object data nodes are children of the object data group node named "Object Data".

The following topics are covered below:

- Tree Views
- List Views
- Class List View
- Object Data Group List View
- Local Data Group List View
- Interface Modules Group List View
- Interface Module List View
- Interfaces Group List View
- Interface List View
- Properties Group List View
- Methods Group List View
- Method Parameter Data Group List View

Tree Views

You can expand and collapse nodes of a class. Expand displays all child nodes and Collapse hides all child nodes of the selected class node.

The logical view provides you with a structured view of the class. You can then expand those class nodes on which you want to work. For more information, refer to the section *Library Workspace*.

List Views

Most of the parent nodes of a class have an assigned list view which can be opened with the **Open** command from the context menu. This section describes the information which is shown in the list views of the logical view. For more information, refer to the section *List View*.

Class List View

The class list view consists of group nodes. The list view for a group node can be opened with the **Open** command.

The following group nodes exist:

- **"Object Data" group:**
  is displayed if the class uses a ODA
- **"Local Data" group:**
  is displayed if the class uses a LDA for class or interface GUIDs
- **"Interface Modules" group:**
  is displayed if the class uses an Interface Module (see *Using Interfaces from several Classes*).
- **"Interfaces" group:**
  is displayed if the class has defined interfaces (internal or external)

The class list view has the following columns:

- **Type:**
  type of the node (e.g. Object Data)
- **Count:**
  number of components of the specified type

Object Data Group List View

The "Object Data" group list view consists of object data nodes. Choosing the **Open** command for a node will open the data area editor for data areas and a special Class Builder dialog for inline definitions.

The "Object Data" group list view has the following columns:

- **Name:**
  name of the object data module or "Inline" in the case of an inline data definition
- **Library:**
  library where the object data module is located (is empty for inline data definitions or if the data area has not yet been created)
- **Type:**
  Natural type of the object data module ("Local Data Area", "Parameter Data Area" or "Inline Definition")

Local Data Group List View

The "Local Data" group list view consists of local data nodes. Choosing the **Open** command for a node will open the data area editor for data areas and a special Class Builder dialog for inline definitions.

The "Local Data" group list view has the following columns:

- **Name:**
  name of the local data module or "Inline" for an inline data definition.
- **Library:**
  library where the local data module is located (empty for inline data definitions or if the data area has not yet been created).
- **Type:**
  Natural type of the local data module ("Local Data Area", "Parameter Data Area" or "Inline Definition").

Interface Modules Group List View

The "Interface Modules" group list view consists of interface module nodes (see *Interface Module List View*). Choosing the **Open** command for a node will open the list view (see *Using Interfaces from several Classes*) for this particular interface module.

The "Interface Modules" group list view has the following columns:

- **Name:**
  name of the interface module (copycode name)
- **Library:**
  library where interface module is located.

Interface Module List View

The interface module list view consists of interface nodes. Choosing the **Open** command for a node will open the list view (see *Interface List View*) for this particular interface.

The interface module list view has the following columns:

- **Name:**
  name of the interface.

Interfaces Group List View

The "Interfaces" group list view consists of interface nodes. Choosing the **Open** command for a node will open the list view (see *Interface List View*) for this particular interface.

The "Interfaces" group list view has the following columns:

- **Name:**
  name of the interface.
- **Component Type:**
  "Internal Interface" for interfaces which are defined in the class and "External Interface" for interfaces which are defined in an interface module included in this class.
- **Defined In:**
  interface module name for externally defined interfaces (empty for internal interfaces).

Interface List View

The interface list view consists of group nodes. Choosing the **Open** command for a node will open a list view for this particular group.

The following group nodes exist:

- **"Properties" group:**
  is displayed if the interface contains property definitions
- **"Methods" group:**
  is displayed if the interface contains method definitions.

The interface list view has the following columns:

- **Type:**
  type of the node (e.g. Properties).
- **Count:**
  number of components of the specified type.

Properties Group List View

The "Properties" group list view consists of property nodes. The "Property" group list view has the following columns:

- **Name:**
  name of the property.

- **Format:**
  format of property.
- **Length:**
  length of property.
- **Dimension:**
  dimension of property.
- **Read-only:**
  shows whether property is read-only or not.
- **ODA Variable:**
  name of assigned ODA variable

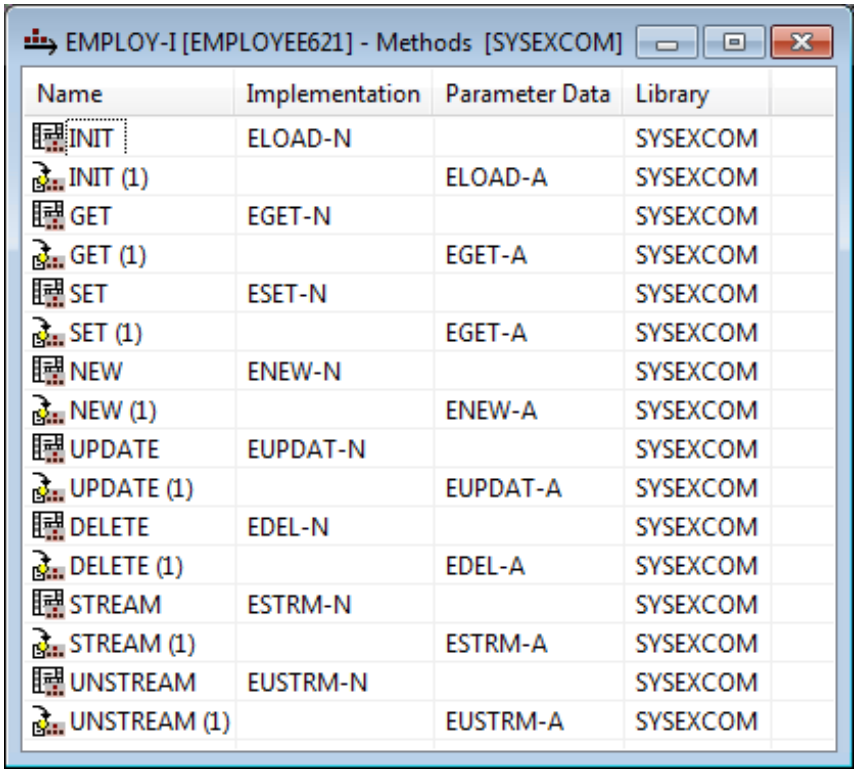
Methods Group List View

The "Methods" group list view consists of method implementation and parameter data nodes. For every method of the interface, it contains one method implementation (subprogram) node and one node for every parameter data definition of the method.

Choosing the **Open** command for a node of this list view will open the editor for the particular node type (for example, program editor for method implementation node).

The "Methods" group list view has the following columns:

- **Name:**
  name of the method. The parameter data nodes are numbered from 1 to n (for example, INIT (2) for the second parameter data node of method INIT).
- **Implementation:**
  only for method implementation node: the name of the subprogram which implements the method
- **Parameter Data:**
  only for method parameter data node: the name of the parameter data module or "Inline" for an inline data definition
- **Library:**
  depending on the node type, library where implementation or parameter data module is located (empty for inline data definitions or if the Natural module has not yet been created).



Name	Implementation	Parameter Data	Library
INIT	ELOAD-N		SYSEXCOM
INIT (1)		ELOAD-A	SYSEXCOM
GET	EGET-N		SYSEXCOM
GET (1)		EGET-A	SYSEXCOM
SET	ESET-N		SYSEXCOM
SET (1)		EGET-A	SYSEXCOM
NEW	ENEW-N		SYSEXCOM
NEW (1)		ENEW-A	SYSEXCOM
UPDATE	EUPDAT-N		SYSEXCOM
UPDATE (1)		EUPDAT-A	SYSEXCOM
DELETE	EDEL-N		SYSEXCOM
DELETE (1)		EDEL-A	SYSEXCOM
STREAM	ESTRM-N		SYSEXCOM
STREAM (1)		ESTRM-A	SYSEXCOM
UNSTREAM	EUSTRM-N		SYSEXCOM
UNSTREAM (1)		EUSTRM-A	SYSEXCOM

Method Parameter Data Group List View

The "Parameter Data" group list view consists of parameter data nodes. Choosing the **Open** command for a node will open the data area editor for data areas and a special Class Builder dialog for inline definitions.

The "Parameter Data" group list view has the following columns:

- **Name:**
  name of the parameter data module or "Inline" for an inline data definition.
- **Library:**
  library where parameter data module is located (empty for inline data definitions or if the data area has not yet been created)
- **Type:**
  Natural type of parameter data module ("Parameter Data Area" or "Inline Definition")

Flat View

The class nodes of the flat view show the class module name.

Unlike the logical view, the flat view does not contain any group nodes. The flat view has the advantage that the level where a specific class component is displayed is lower compared to the logical view, and thereby provides you with a better class overview.

The following topics are covered below:

- Tree Views
- List Views
- Class List View
- Interface Module List View
- Interface List View

Tree Views

You can expand and collapse nodes of a class. Expand displays all child nodes and Collapse hides all child nodes of the selected class node. The flat view provides you with a general overview of the class. It lists all sub-components of a class component on the same level. For example, if an interface node is expanded, all properties and methods of the interface will be displayed as child nodes of the interface node. For more information, see *Library Workspace*.

List Views

The flat view supports only a few list views because of the low node nesting level. The list views can be opened with the **Open** command from the context menu. This section describes the information which is shown in the list views of the flat view. For more information, refer to the section *List View*.

Class List View

The class list view contains a node for every child component.

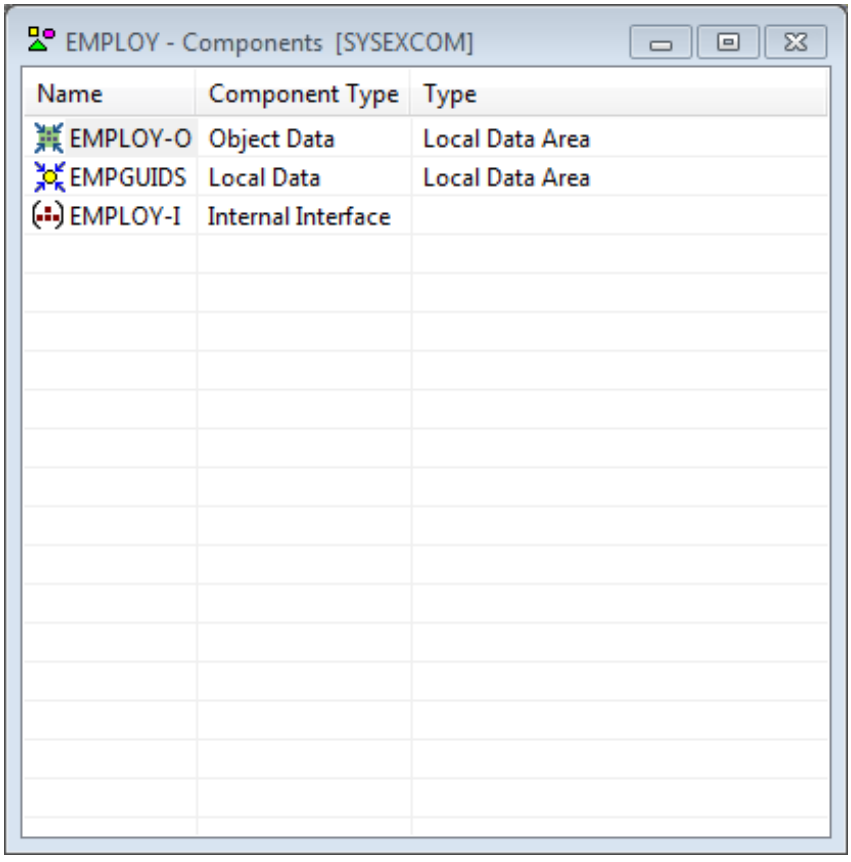
The following nodes exist:

- **Object Data node**
  for every ODA of the class. Choosing the **Open** command of the node opens the data area editor for data areas and a special Class Builder dialog for inline definitions
- **Local Data node**
  for every GUID LDA of the class. Choosing the **Open** command of the node opens the data area editor for data areas and a special Class Builder dialog for inline definitions.

- **Interface Module node**
  for every interface module which is used by the class. Choosing the **Open** command of the node will open the interface module list view.
- **Interface node**
  for every interface of the class (external and internal). Choosing the **Open** command of the node will open the interface list view.

The class list view has the following columns:

- **Name:**
  name of the component.
- **Component Type:**
  indicates the type of the component ("Object Data", "Local Data", "Interface Module", "External Interface" or "Internal Interface").
- **Type:**
  only for component type "Object Data" and "Local Data": Natural type of data module ("Local Data Area", "Parameter Data Area" or "Inline Definition")



Interface Module List View

The interface module list view consists of interface nodes. Choosing the **Open** command of a node will open the list view (see *List Views*) for this particular interface.

The interface module list view has the following columns:

- **Name:**
  name of the interface.

Interface List View

The interface list view contains all nodes for the properties and methods of the interface.

The following nodes exist:

- **Property node**
  for every property of the interface.
- **Method implementation node**
  for every method of the interface. Choosing the **Open** command for the node will open the program editor with the specified implementation (subprogram).
- **Method parameter data node**
  for every parameter data component of every method of the interface. Choosing the **Open** command for the node will open the data area editor for data areas and a special Class Builder dialog for inline definitions.

The interface list view has the following columns:

- **Name:**
  name of the property or method; the parameter data nodes for methods are numbered from 1 to *n* (for example, INIT (2) for the second parameter data node of method INIT).
- **Implementation:**
  only for properties and method implementation node: the name of the assigned ODA variable for properties and the name of the subprogram which implements the method for methods.
- **Parameter Data:**
  only for method parameter data node: the name of the parameter data module or "Inline" for an inline data definition.
- **Library:**
  only for methods: depending on the node type, library where implementation or parameter data module is located (empty for inline data definitions or if the Natural module has not yet been created).

- **Format:**
  only for properties: format of property.
- **Length:**
  only for properties: length of property.
- **Dimension:**
  only for properties: dimension of property.
- **Read-only:**
  only for properties: shows whether property is read-only or not.

Class Builder Nodes

Related to the user interface, every component of a class is represented by a node. Nodes are displayed both in tree views and in list views.

Every node has an icon and textual information about the component which can be the name of the component (in the library workspace) or the name of the component and additional information (in the list views).

The following table lists all available Class Builder nodes with their icons and a short description:

Type	Icon	Description
new class		new class which has not yet been saved
class (src)		class which is only available as source
class (gp)		class which is only available as generated program
class (src & gp)		class which is available as source and generated program
ODA		object data defined in a data area module
inline ODA		object data defined with an inline data definition
LDA		local data (for GUIDs) defined in a data area module
inline LDA		local data (for GUIDs) defined with an inline data definition
Interface Module		interface module, i.e., copycode which defines interfaces
internal interface		interface which is defined in the class
external interface		interface which is defined in an interface module that is used by the class
internal property		property which is defined in an internal interface
external property		property which is defined in an external interface

Type	Icon	Description
internal method		method which is defined in an internal interface
external method		method which is defined in an external interface
method implementation		subprogram which implements a method
method PDA		method parameter data defined in a data area module
inline method PDA		method parameter data defined with an inline data definition

In the following section, the Class Builder nodes are described in more detail. The commands of a specific node can be invoked from the context menu of the node or the "Classes" toolbar.

The following topics are covered below:

- Class Nodes
- Object Data Nodes
- GUID Local Data Nodes
- Interface Nodes
- Property Nodes
- Method Nodes
- Method Implementation Nodes
- Method Parameter Data Nodes

Class Nodes

The class node represents the class itself. The name displayed in the class node is either the class name (logical view) or the class module name (flat view).

Types

New Class

If a new class is created, it is displayed with the new class icon until it is saved the first time. Therefore, new class means that the class is only “transient” in the current Natural session and is not available in source format. For this reason, the new class will not be shown in the File View which shows the source and gp files of the Natural objects. In addition, it is not possible to execute all class node commands on a new class.

Source-Only

The source-only class icon is displayed if the class is only available in source format but has not yet been cataloged.

GP-only

The GP-only class icon is displayed if the class is only available in GP format. Classes of this type cannot be handled with the Class Builder and the context menu of these classes is the same as for all other Natural objects which are only available in GP format.

Source-and-GP

The Source-and-GP class icon is displayed if the class is available in source and GP format.

Commands

Command	available for	Description
Open	new source-only source-and-GP	Opens the class list view. For more information, see *List Views*
List	new source-only source-and-GP	Opens the program editor in read-only state with the internal source format of the current class structure.
Cat	source-only source-and-GP	Catalogs the current class.
Save	new source-only source-and-GP	Saves the current class structure in the given class module.
Save As	new source-only source-and-GP	Saves the current class structure in a new Natural module or with a different encoding.
Stow	new source-only source-and-GP	Stows the current class structure in the given class module.
New ODA	new source-only source-and-GP	Creates a new object data area for the class.
New Interface	new source-only source-and-GP	Creates a new interface for the class.
New Interface Module	new source-only source-and-GP	Creates a new interface module. This interface module is linked to the class.
Link LDA	new source-only source-and-GP	Uses an existing data area as GUID LDA for the class. See *Link*.
Link ODA	new source-only source-and-GP	Uses an existing data area as ODA for the class. See *Link*.

Command	available for	Description
Link Interface Module	new source-only source-and-GP	Uses an existing copycode as interface module for the class. All interfaces defined in the Interface Module will be included in the class. See ***Link***.
Register	source-and-GP	Registers the class in the system registry. For more information, see *NaturalX* in the *Programming Guide*.
Unregister	source-and-GP	Unregisters the class from the system registry. For more information, see *NaturalX* in the *Programming Guide*.
Rename	new source-only source-and-GP	Changes either the class name or the class module name depending on the current view of the library workspace. For more information, see ***Renaming Class Members***.
Delete	new source-only source-and-GP	Deletes the Natural module of the class (for source-only and source-and-GP) or only the internal structure of the class (new).
Restore	source-only source-and-GP	Removes all changes of the class which have not yet been saved. This command will close all list views of the class and collapse the class node in the library workspace.
Cut	source-only source-and-GP	Cuts the class module.
Copy	source-only source-and-GP	Copies the class module.
Paste	source-only source-and-GP	Pastes the class module.
Print	new source-only source-and-GP	Prints the source format of the current class structure.
Properties	new source-only source-and-GP	Opens the Properties dialog which shows class-specific information. For more information, see ***Node Properties***.

Object Data Nodes

An object data node represents an object data area module or an inline object data definition. A class can have several object data nodes. If more than one object data node exists, you must take care to follow the correct object data sequence when you use these nodes in method implementations.

Types

Data Area

This type indicates that the object data is defined in a separate Natural module of type local data area or parameter data area. The name which is displayed in the node is the name of the Natural data area module.

Inline Data Definition

This type indicates that the object data is defined direct in the class source with a `DEFINE DATA OBJECT` statement. In this case, the object data has to be defined again in every method implementation which uses the object data. A node of this type is always named "Inline".

Commands

Command	available for	Description
Open	data area	Opens the data area module with the data area editor.
Edit	inline data definition	Opens a dialog which shows the contents of the inline data definition for editing.
List	data area	Lists the data area module.
Cat	data area	Catalogs the data area module.
Stow	data area	Stows the data area module.
Unlink	data area	Unlinks the data area module from the class, i.e. it is no longer used as Object Data Area for the class.
Rename	data area	Renames the Object Data Area link, i.e. uses another data area module as Object Data Area for the class. For more information, see *Renaming Class Members*.
Delete	inline data definition	Deletes the inline data definition from the class.
Print	data area	Prints the data area module.
Properties	data area inline data definition	Opens the Properties dialog which shows object data-specific information. For more information, see *Node Properties*.

GUID Local Data Nodes

An GUID Local Data node represents a local data area module or an inline local data definition which contains GUID definitions. A class can have several local data nodes.

Types

Data Area

This type indicates that the GUID local data is defined in a separate Natural module of type local data area or parameter data area. The name which is displayed in the node is the name of the Natural data area module.

Inline Data Definition

This type indicates that the GUID local data is defined direct in the class source with a `DEFINE DATA LOCAL` statement. A node of this type is always named "Inline".

Commands

Command	available for	Description
Open	data area	Opens the data area module with the data area editor.
Edit	inline data definition	Opens a dialog which shows the contents of the inline data definition for editing.
List	data area	Lists the data area module.
Cat	data area	Catalogs the data area module.
Stow	data area	Stows the data area module.
Unlink	data area	Unlinks the data area module from the class, i.e. the data area module is no longer used as GUID Local Data Area for the class.
Rename	data area	Renames the GUID Local Data Area link, i.e. uses another data area module as GUID Local Data Area for the class. For more information, see *Renaming Class Members*.
Delete	inline data definition	Deletes the inline data definition from the class.
Print	data area	Prints the data area module.
Properties	data area inline data definition	Opens the Properties dialog which shows local data-specific information. For more information, see *Node Properties*.

Interface Module Nodes

An Interface Module node represents an interface module. The interface module is a Natural module of type copycode which defines interfaces that can be included in several classes. For more information about interface modules and their usage, see *Using Interfaces from several Classes*.

Commands

Command	Description
Open	Opens the interface module list view. For more information, see *List Views*.
List	Opens the program editor in read-only state with the source format of the current Interface Module structure.
Save	Saves the current Interface Module structure in the given Natural copycode module.
New Interface	Creates a new interface in the Interface Module.
Unlink	Unlinks the Interface Module from the class, i.e. the interfaces defined in the Interface Module are no longer available in the class.
Print	Prints the source format of the current Interface Module structure.
Properties	Opens the Properties dialog which shows Interface Module-specific information. For more information, see *Node Properties*.

Interface Nodes

An interface node represents an interface of an interface module or a class. For more information about internal and external interfaces, see *Using Interfaces from several Classes*.

Types

Internal

The parent of an internal interface is either an interface module or a class. If its parent is an interface module, this means that the interface is defined in the interface module which is used by the class. In this case, the interface will be displayed a second time as an external interface of the class (For more information, see *Using Interfaces from several Classes*). If the internal interface is a child of the class itself, this means that the interface is defined direct in the class.

External

An external interface can appear only as subnode of a class, which uses an interface module which defines this interface. The commands which can be executed on an external interface node are only a subset of the commands available for an internal interface. Basically you can only change the implementation of such an interface. For more information, see *Using Interfaces from several Classes*.

Commands

Command	available for	Description
Open	internal external	Opens the interface list view. For more information, see *List Views*.
New Method	internal	Creates a new method for the interface.
New Property	internal	Creates a new property for the interface.
Rename	internal	Renames the interface. For more information, see *Renaming Class Members*.
Delete	internal	Deletes the interface and all its dependent components.
Properties	internal external	Opens the Properties dialog which shows interface-specific information. For more information, see *Node Properties*.

Property Nodes

A property node represents a property of an internal or external interface.

Types

Internal

If a property appears as subnode of an internal interface, it will be displayed as internal property. An internal property node always has a dedicated external property node.

External

If a property appears as subnode of an external interface, it will be displayed as external property. The commands which can be executed on an external property are only a subset of the commands which are available for internal properties.

Commands

Command	available for	Description
Rename	internal	Renames the property. For more information, see *Renaming Class Members*.
Delete	internal	Deletes the property.
Properties	internal external	Opens the Properties dialog which shows property-specific information. For more information, see *Node Properties*.

Method Nodes

A method node represents a method of an internal or external interface.

Types

Internal

If a method appears as subnode of an internal interface, it will be displayed as an internal method. An internal method node always has a dedicated external method node.

External

If a method appears as subnode of an external interface, it will be displayed as external method. The commands which can be executed on an external method are only a subset of the commands which are available for internal methods.

Commands

Command	available for	Description
New PDA	internal	Creates a new method parameter data area for the method.
Link PDA	internal	Uses an existing parameter data area as method PDA. See *Link*.
Link implementation	internal external	Uses an existing subprogram as method implementation. See *Link*.
Rename	internal	Renames the method. For more information, see *Renaming Class Members*.
Delete	internal	Deletes the method and all its dependent components.
Properties	internal external	Opens the Properties dialog which shows method-specific information. For more information, see *Node Properties*.

Method Implementation Nodes

A method implementation node represents the Natural subprogram which is executed when the method is called.

Commands

Command	Description
Open	Opens the subprogram of the method implementation in the program editor.
List	Lists the subprogram of the method implementation in read-only mode in the program editor.
Cat	Catalogs the subprogram of the method implementation.
Stow	Stows the subprogram of the method implementation.
Rename	Renames the method implementation, i.e. uses another subprogram for the method implementation. For more information, see *Renaming Class Members*.
Print	Prints the subprogram of the method implementation.
Properties	Opens the Properties dialog which shows method implementation-specific information. For more information, see *Node Properties*.

Method Parameter Data Nodes

A method parameter data node represents a parameter data area module or an inline parameter data definition. A method can have several method parameter data nodes, which define the parameter used by the method implementation. If more than one method parameter data node exists, you must ensure that the correct parameter data sequence is used in method implementations.

Types

Data Area

This type indicates that the method parameter data is defined in a separate Natural module of type parameter data area. The name which is displayed in the node is the name of the Natural parameter data area module.

Inline Data Definition

This type indicates that the method parameter data is defined direct in the class source (or interface module source) with a `DEFINE DATA PARAMETER` statement. In this case, the parameter data must be defined again in every method subprogram. A node of this type is always named "Inline".

Commands

Command	available for	Description
Open	data area	Opens the data area module with the data area editor.
Edit	inline data definition	Opens a dialog which shows the contents of the inline data definition for editing..
List	data area	Shows the listing of the data area module.
Cat	data area	Catalogs the data area module.

Command	available for	Description
Stow	data area	Stows the data area module.
Unlink	data area	Unlinks the data area module from the method, i.e. the data area module is no longer used as parameter data area for the method.
Rename	data area	Renames the method parameter data area link, i.e. uses another data area module as parameter data area for the method. For more information, see *Renaming Class Members*.
Delete	inline data definition	Deletes the inline data definition.
Print	data area	Prints the data area module.
Properties	data area inline data definition	Opens the Properties dialog which shows method parameter data-specific information. For more information, see *Node Properties*.

Node Properties

The Class Builder provides node-specific information on Natural classes and their elements if context-menu entry **Properties** is chosen. This context-menu entry is available if an object is selected in the library workspace or in a list view. The property sheet provides no information on group nodes.

The information itself is presented in a property sheet. The actual number of property pages shown depends on the type of the selected object.

- **OK**: Accept modifications.
- **Cancel**: Skip modifications.

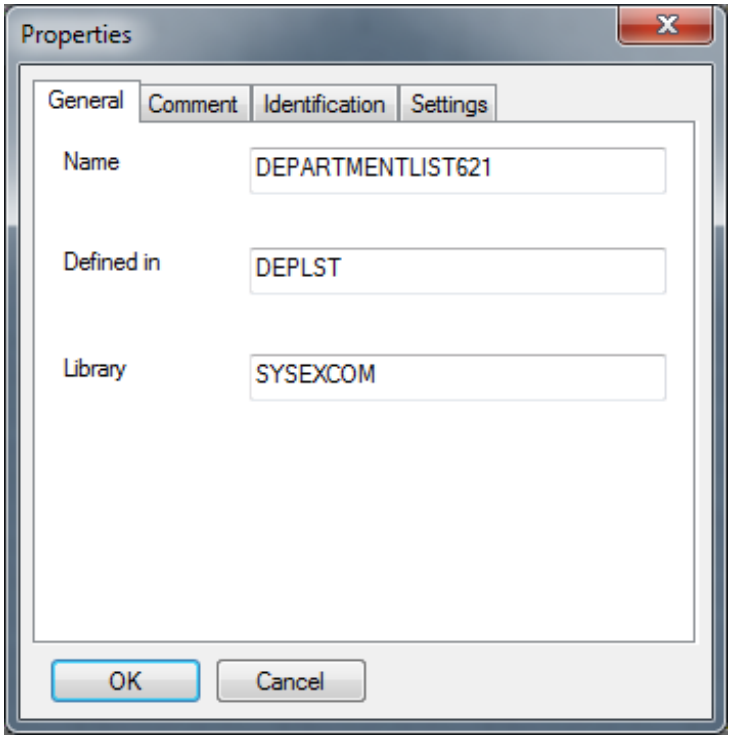
For all class elements, property pages **General** and **Comment** are available. The other property pages depend on the selected node type.

The following topics are covered below:

- General
- Comments
- Identification
- Settings

- Definition

General



This property page shows general information on the selected object. Its contents vary with the corresponding type of node and are described in the following sections.

Class

Name	Class Name
Defined in	Class Module
Library	Library

Object and Local Data Area

Name	Name of Object or Local Data Area
Used in	Class Name
Library	Library

Inline Data Definition

Name	"Inline Definition"
Defined in	Class Name

Interface Module

Name	Name of Interface Module
Used in	Class Name
Library	Library

Interface

Name	Name of Interface
Defined in	Class Name
Interface Module	If the interface is defined in an interface module this field shows the corresponding name.

Method

Name	Name of Method
Defined in	Name of the interface that offers this method.
Interface Module	If the method is defined in an interface module this field shows the corresponding name.

Implementation

Name	Name of Subprogram
Used in	Name of the method that is implemented by this subprogram.
Library	Library

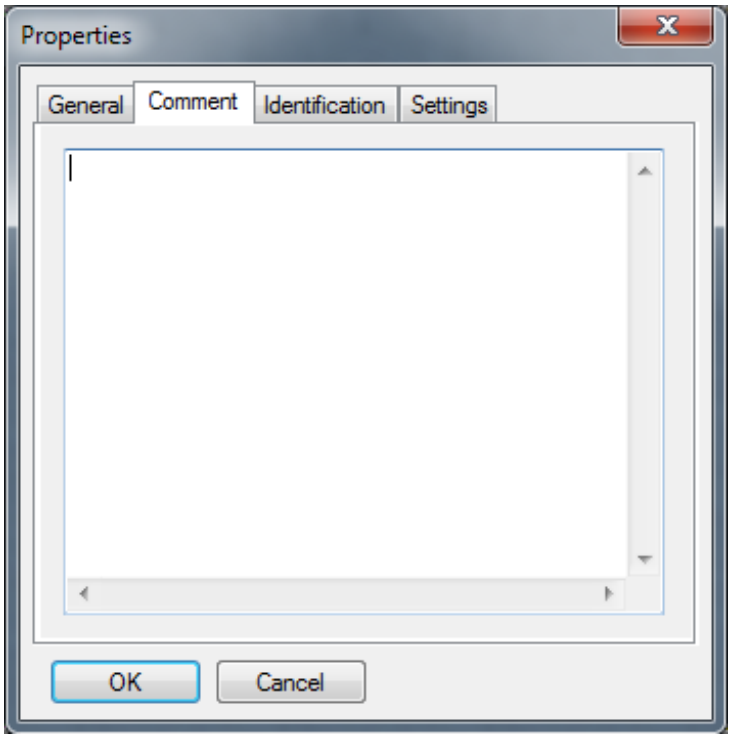
Parameter Data Area

Name	Name of Parameter Area
Used in	Name of Method
Library	Library

Property

Name	Name of Property
Defined in	Name of the interface that offers this property.
Interface Module	If the property is defined in an interface module this field shows the corresponding name.

Comments

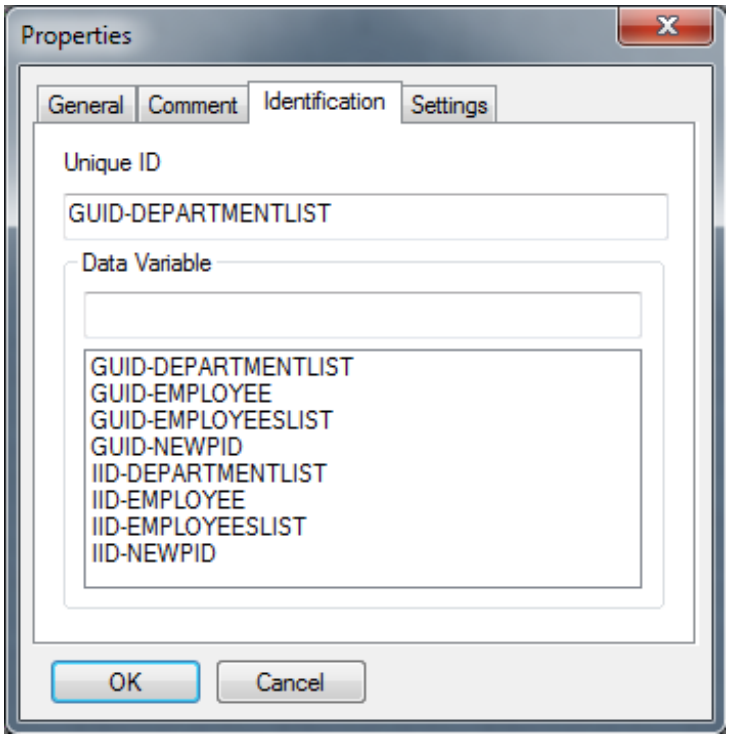


Each component has its own comment.

This property page shows the comment and allows adding new or modifying existing comments. They are entered and listed without any special syntactic notation.

The comment is changed if the property sheet is left by pressing **OK**. Pressing **Cancel** leaves the comment unchanged.

Identification



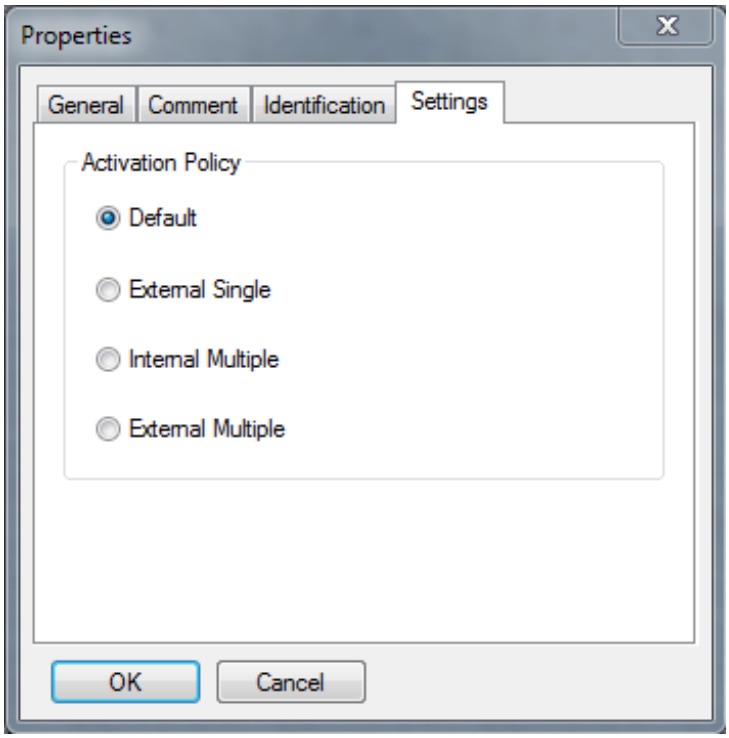
This property page is available for class and interface nodes. For interfaces, the list box below is only enabled if the interface is defined direct as part of the class. The list box is not visible if the interface is defined in an interface module.

The upper control **Unique ID** shows the current **G**lobal **U**nique **ID** of a class or an interface as read-only information.

This list box offers all data variables contained in local data areas that are linked to the class. These variables can be used as unique identifiers. Inline definitions of variables are not supported.

To exchange the current **G**lobal **U**nique **ID** that is displayed in the upper control with another value, select a variable from the list. The name control is then updated with the newly selected variable name. The **G**lobal **U**nique **ID** is exchanged if a variable has been selected and the property sheet is left by pressing **OK**. Pressing **Cancel** leaves the identification unchanged. There is no check whether a selected variable represents a valid **G**lobal **U**nique **ID**.

Settings



This property page is available for class nodes only. It allows setting the class's activation policy within the Class Builder.

An activation policy for a class can be:

- External Single
- Internal Multiple
- External Multiple

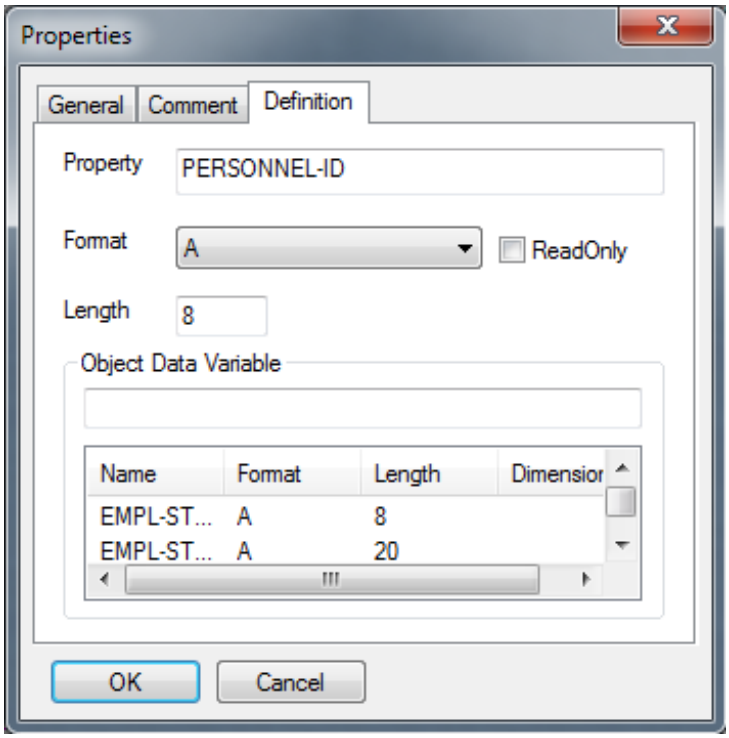
Or it is set to default.

More information on the meaning of these values can be found in *NaturalX* in the *Programming Guide*.

To change the current activation policy select the required value.

The value is changed if the property sheet is left by pressing **OK**. Pressing **Cancel** leaves the identification unchanged.

Definition



This property page is available for properties of interfaces only. It allows modifying the definition of an existing property.

The property's name cannot be changed. The following changes are possible:

- An Object Data Variable can be assigned to the property.

  The available Object Data Variables are listed in the page's list box together with their format definition and dimension.

  They are taken from the Object Data Areas that are linked to the current class. Inline definitions of variables are not supported.
- Existing assignments of Object Data Variables to properties can be changed. The corresponding control is then updated with the newly selected variable's name.
- The property's format definition can be added or changed if it is different from the Object Data Variable's definition.

  Otherwise format and length definition are taken from the assigned Object Data Variable.
- It can be defined whether this property is used read only.

The definition of the property is changed if the property sheet is left by pressing **OK**. Pressing **Cancel** leaves the definition unchanged.

Adding Class Components

To make the development of a class more comfortable the Class Builder offers two ways to add components to a class.

The following topics are covered below:

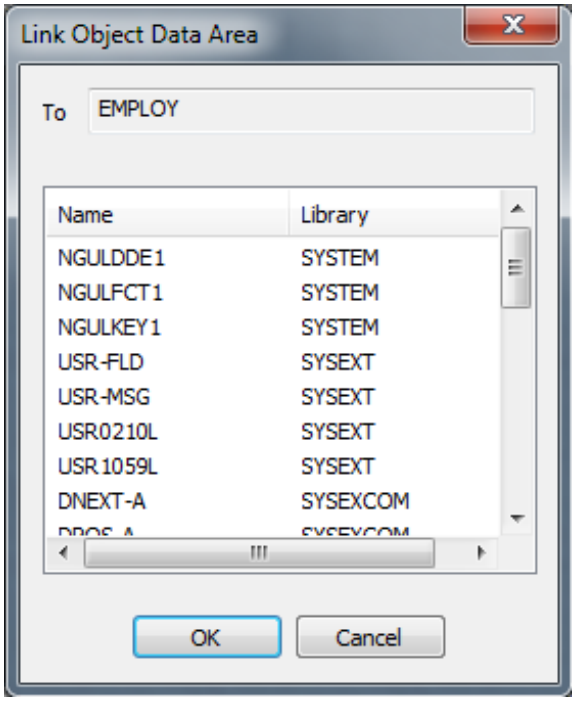
- Link
- New
- New Class
- New Object Data Area
- New Interface Module
- New Interface
- New Method
- New Property

Link

Existing Natural objects can be linked to a class component.

If context menu item **Link** is activated for an object node a dialog is opened. It lists all objects of the required type that can be found in the current library or its steplibs.

If an object has been selected and the dialog is left by pressing **OK**, a reference to the selected object is added to the class structure. **Cancel** leaves the class structure unchanged.



Link to Class

A GUID Local Data Area, an Object Data Area or an Interface Module can be linked to a class. The dialog shows object name and library.

Link to Method

Each method requires a method implementation. The existing implementation can be exchanged by linking another subprogram to a selected method. Moreover, one or more Parameter Data Areas can be linked to a method. The dialog shows object name and library.

New

New class components are created with context menu item **New**.

In the library workspace, class components are created using in-place editing. List views use dialogs to query the necessary data and create new objects. This applies to all nodes apart from class properties: They are always created using a dialog.

The following sections describe how the different class components are created.

New Class

A new class is first created as an internal class structure. At this time the class name is defined. The class module name, i.e. the name of the actual Natural object, is assigned when the class is saved the first time.

Library Workspace

A new class name, for example NEWCLS, is generated. The corresponding tree node is selected and made available for in-place editing. The name can be changed to any valid class name.

List View

A dialog is opened that asks for the name of the new class.

New Object Data Area

Creating a new object data area adds a reference to a new component to the class structure. The corresponding Natural object is not yet created. It is created if you confirm such when you open it.

Library Workspace

A new object data area, for example NEWODA, is generated. The corresponding node is selected and is made available for *in-place editing*. The name can be changed to any valid data area name.

List View

A dialog is opened that asks for the name of the new object data area.

New Interface Module

Creating a new interface module adds a reference to a new component to the class structure. The corresponding Natural object is not yet created. It is created if it contains interfaces at the time the class is saved.

Library Workspace

A new interface module, for example NEWEIF, is generated. The corresponding node is selected and is made available for in-place editing. The name can be changed to any valid copycode name.

List View

A dialog is opened that asks for the name of the interface module.

New Interface

Library Workspace

A new interface, for example NEWIIF, is generated. The corresponding node is selected and is made available for in-place editing. The name can be changed to any valid interface name.

List View

A dialog is opened that asks for the name of the interface.

New Method

Library Workspace

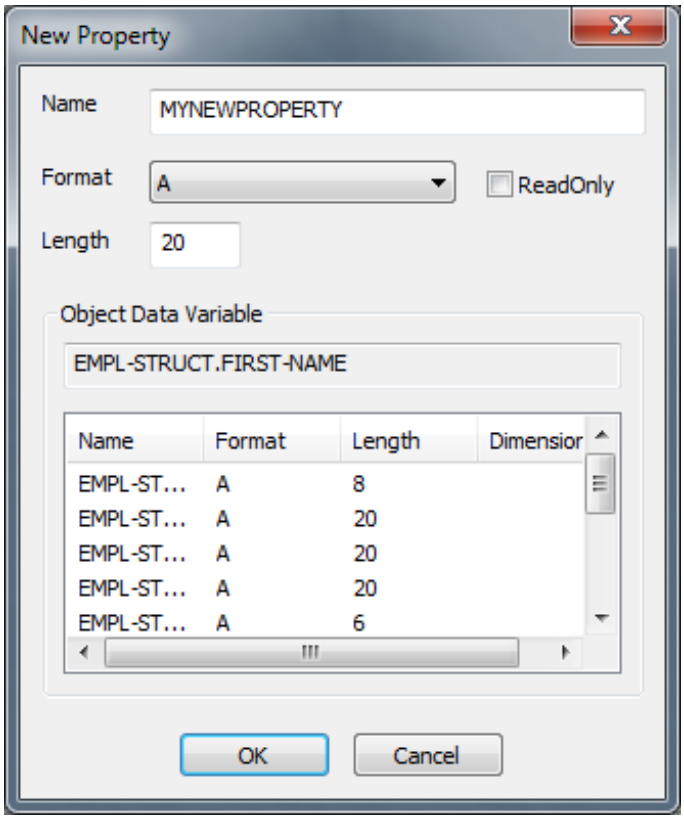
A new method, for example NEWMET, is generated. The corresponding node is selected and is made available for in-place editing. The name can be changed to any valid method name. The new method name is also taken as the name of the method implementation. Both are added to the class structure. If the method name is longer than a valid Natural subprogram name, only the first characters are used to guarantee a valid implementation name.

List View

A dialog is opened that asks for the name of the method. The new method name is also taken as the name of the method implementation. Both are added to the class structure. If the method name is longer than a valid Natural subprogram name the first characters are used to guarantee a valid implementation name.

New Property

New properties are always created using a dialog.



This dialog retrieves the following information:

Property name:	A valid property name. This is either a new name or the name of the selected ODA variable. For fully qualified ODA variable names the dot is replaced by an underscore.
ODA variable:	The list box lists all variables that are defined in the linked ODAs. If property name, format and length are not changed, these values are taken from the selected ODA variable.
Format:	Format and length can be changed if they must be different from the ODA variable's definitions.
Read-Only:	The property can be marked as read-only.

Renaming Class Components

Like any other Natural object that can be modified in the Natural Studio, the components of a class are renamed by editing their identifier in place. This is done using the mouse or by pressing F2 or by choosing context menu entry **Rename** which is enabled for every class component.

During the edit process the new name is checked for syntactical correctness. If it is not a valid Natural name the edit mode cannot be left. Pressing ESC cancels the edit mode and resets the old identifier.

If class components refer to Natural objects such as Object Data Areas, Parameter Data Areas or Interface Modules, only the references within the class are changed. The corresponding Natural objects are not renamed. They have to be changed explicitly if required.

Removing Class Components

Unlink

Context menu entry **Unlink** is available for class components that refer to Natural objects like Data Areas or Interface Modules. If these modules have been linked to a class previously they can be removed using **Unlink**.

This action only removes the reference to selected components from the class. It does not delete an existing Natural object.

Delete

Context menu entry **Delete** is available for classes and those of their components that do not refer to Natural objects.

If this context menu item is selected, the Class Builder's **Delete** dialog will be displayed.

You are asked whether you want to delete the selected component(s). A list of references that shows the dependent Natural objects is displayed for each component (if you do not choose **Yes to All**). These Natural objects are identified by name, library and Natural object type if required. The list serves for information purposes only. The dependent Natural sources are not affected.

If the selected component is the class itself, the internal structure is deleted and the corresponding Natural source and cataloged modules are removed from the library.

Yes	Deletes the selected component. If several components are selected, the list of references is shown for the next component.
Yes to All	Closes the dialog and deletes all selected components.
No	Does not delete the selected component. If other components are selected, the delete procedure continues by displaying the delete dialog for the next component.
Cancel	Closes the dialog without deleting anything.

Editing Class Components

Classes

At the time a new class is created, the corresponding new class module is not yet created. This occurs only if **Save**, **Save As** or **Stow** is called for the class.

Save

Save called for an existing class writes the class source to the class module.

If **Save** is called for a new class that does not yet have a corresponding class module, then **Save** is treated like **Save As**. The encoding is initialized with the default code page currently used. If such a class module does not yet exist in the current library, the class module is created and the source is written to this object.

Save As

If **Save As** is called, a dialog is opened that prompts for the class module and the encoding of the class. The input length is restricted to guarantee a valid Natural class module name and the input is checked for validity. If such a class module does already exist or if the name is invalid, an error message is issued.

Cat

If the command **Cat** is called, the class source is cataloged and a corresponding class GP is generated. This does not apply to new classes.

Stow

As for other Natural objects **Stow** internally saves and catalogs a class. If a new class is to be stowed, you are prompted for the class module as described for **Save As**.

Natural Objects

Natural objects that can act as class components can also be modified in the context of the class structure. References to Object Data Areas, Parameter Data Areas and Interface Modules can be created by **New**. Existing objects can be edited, saved and stowed.

Local Data Areas and method implementations cannot be created in the class's context. Here only existing objects can be linked to the class. But they can be edited, saved and stowed.

Other Class Components

Other class components such as interfaces, methods and properties cannot be saved, cataloged or stowed independently. They can only be modified in the context of a class.

Using Interfaces from Several Classes

For some applications, it is useful to implement the same interface in several classes. For this purpose, it is possible to define the interface in a Natural copycode module and include this copycode module in the class which wants to implement the interface. The implementation-specific settings, like method implementations, can be defined in the copycode as a default setting, and they can be overwritten in the class, to use class specific implementations.

Natural copycode modules which define interfaces are called Interface Modules in the Class Builder environment. Interface Modules are fully integrated in the Class Builder, so that interfaces defined in an Interface Module can be handled in the same way as interfaces of a class. However, an Interface Module can only be changed with the Class Builder when it is included from a class.

Interfaces which are defined in an Interface Module are always visible in two places of a class: they are shown as an internal interface under the Interface Module node and they are shown as an external interface under the class node. The commands available for an external interface can be used to change the implementation of the interface.

You can save a changed Interface Module without saving the whole class. If an Interface Module is changed and the class which is the parent of the Interface Module node is saved, the Class Builder asks the user if he wants to save the Interface Module as well.

The locking principles for Interface Modules are described in *Locking Concept*.

Note: If you change an Interface Module, you should always be aware that this Interface Module can also be used by other classes. After saving the changes other classes can possibly no longer be stowed without errors. The Class Builder cannot check if your Interface Module is used by other classes!

Creating a new Interface Module

The class command **New Interface Module** (see *Class Builder Nodes*) creates a new Interface Module.

An Interface Module node is added in the tree and list views and you can then create new interfaces for the Interface Module, methods and properties for the interfaces and so on. If a new component is created for the Interface Module, the corresponding external node will be added for the class. For example, if a new interface INT1 as added to the Interface Module, an external interface node named INT1 will be created as subnode of the class. The new Interface Module is saved just as an existing Interface Module. As soon as the Interface Module exists as Natural module, it can be linked from other classes.

Linking an existing Interface Module

The class command **Link Interface Module** (see *Class Builder Nodes*) uses an existing Interface Module for the class. A dialog is shown which lists all Natural copycode modules of the current step libraries.

Note: The dialog will list all copycode modules and not only the Interface Modules.

If you select a copycode module from this list which defines class interfaces, these interfaces are added to the current class interfaces. An error will be generated if you select a copycode module which does not define interfaces or if the selected copycode module contains an interface which is already defined in the class. In this case, the Interface Module is not linked to the class.

If the Interface Module was linked successfully to the class, a node for it will be added to the class tree. Opening the Interface Module node will show the interfaces of the Interface Module. Furthermore all interfaces of the Interface Module are added as external interfaces nodes to the class itself.

Unlinking an Interface Module

If the **Unlink** command (see *Interface Module Nodes*) is executed for an Interface Module, the interfaces of this Interface Module are no longer used by the class.

This has the effect that the Interface Module node itself and all external interface nodes from this Interface Module are removed from the class.

Note: If you unlink an Interface Module from a class, all class-specific settings contained in the class source module, such as method implementations for the interfaces of this Interface Module, will be deleted as well.

Interface Nodes

If an Interface Module is used by a class, every interface defined in the Interface Module is represented by two nodes: an internal interface node which is a subnode of the Interface Module and an external interface node which is a subnode of the class. These two interface node types can be distinguished by their icon (see *Interface Nodes*). The same is of course valid for the property and method nodes: if they are children of an internal interface, they are represented by an internal node and if they are children of an external interface, they are represented by an external node (see *Property Nodes* and *Method Nodes*).

Furthermore the commands which can be executed on external interfaces, properties and methods are only a subset of the commands available on internal interfaces, properties and methods. For example, the name of an interface can only be changed for an internal interface. External interfaces allow only the redefinition of the implementation of the interface, i.e. changing the method implementation and the ODA variable which is assigned to a property.

Locking Concept

Natural must ensure that a Natural module cannot be changed at the same time from different places. Therefore, related to the Class Builder, this means that a Natural user must be prevented from changing a Natural module with the program editor which has already been changed with the Class Builder and vice versa.

The Class Builder can be used to change Natural classes and Interface Modules which are special copycode modules (see *Using Interfaces from several Classes*).

Because of the different requirements, the locking concept for classes differs from the Interface Module locking concept. In the following sections both concepts are described.

Locking of Classes

The locking of classes is done very flexibly. The Class Builder does not lock a class until it is changed. This means that a class which is opened with the Class Builder can be opened in the program editor as well.

If a class is opened in the program editor, the class nodes can be viewed in the Class Builder, but it is not possible to apply any changes. Before changing the class, the program editor session has to be closed first.

If a class is visible in the Class Builder and the user changed the class in the program editor, the changes will also be shown in the Class Builder when the class is saved. If a class has been changed with the Class Builder it is no longer possible to open this class with the program editor.

Locking of Interface Modules

The locking of Interface Modules is a bit more restrictive than the locking of classes. A two stage locking exists for the Interface Modules. For the first time the Class Builder must ensure that the Interface Module cannot be changed with the Class Builder and the program editor at the same time: if a class which uses an Interface Module is opened in the Class Builder, the Interface Module is locked. This means on the one hand, that an Interface Module can no longer be opened with the program editor, when a class which uses it is opened in the Class Builder. On the other hand, a class cannot be opened with the Class Builder when it uses an Interface Module which is already open in the program editor.

Moreover, an Interface Module can be opened several times in the Class Builder if it is included from several classes. The Class Builder must ensure that an Interface Module is opened only once, when the user wants to change it, because the other Interface Module instances are then no longer up-to-date: it will try to close all other instances, to make sure that only the current instance of the Interface Module remains visible. The Class Builder will display a confirmation dialog for this purpose which allows the user to stop the process.

If one of the classes was already changed, the user will be asked, if the changes are to be saved . After saving a changed Interface Module, it is again possible to open other classes which use the Interface Module.

Tutorial

This section provides a short introduction on the usage of the Class Builder.

The example shows how class EMPLOYEE in library SYSEXCOM can be built using the Class Builder.

The following topics are covered:

- New class
- Linking Object Data
- Creating an Interface
- Creating Methods
- Creating Properties
- Using an Interface Module
- Linking a GUID Local Data Area
- Activation Policy
- Save and Stow Class

- Register

New class

Activate the logical view in the library workspace and create a new library MYEXCOM that contains the local data areas EMPGUIDS and EMPLOY-O. These are just copies of the objects in SYSEXCOM.

EMPGUIDS contains GUID definitions and EMPLOY-O contains object data definitions. To create a new class MYEMPLOYEE select the library node and then select context menu item **New Source > Class**. A new tree node labeled "NEWCLS" is presented for in-place editing. Just change its name to "MYEMPLOYEE".

Linking Object Data

The object data for MYEMPLOYEE have to be defined in an object data area. This object data area can either be created by selecting context menu item **New** of node "MYEMPLOYEE" or by linking an existing object data area via context menu item **Link > Object Data Area**.

A dialog pops up and shows a list of all local and parameter data areas in MYEXCOM and its steplibs. These objects can be used as object data areas. Select EMPLOY-O.

Creating an Interface

To create the first interface select context menu item **New > Interface** of node "MYEMPLOYEE". A new tree node labeled "NEWIIF" is presented for in-place editing. Just change its name to "EMPLOY-I". Further interfaces can be created accordingly or by selecting **New** in the context menu for "Interfaces" (group node).

Creating Methods

To create the first method select context menu item **New > Method** of interface node "EMPLOY-I". A new tree node labeled "NEWMET" is presented for in-place editing. Rename this node to "INIT". A method implementation node with the same name is created automatically.

To use subprogram ELOAD-N (copied from SYSEXCOM) to implement this method, select the method's context-menu item **Link > Implementation** and change the method implementation.

Parameter Data Area ELOAD-A (copied from SYSEXCOM) can be linked using **Link > Parameter Data Area** and then selecting the appropriate module. Further methods can be created accordingly or by selecting **New** in the "Methods" (group node) context menu.

Creating Properties

To create the first property, select context-menu item **New > Property** of interface node "EMPLOYI". The dialog lists all object data variables that are defined in linked object data areas and can be assigned to a property. They are shown together with their format and length definition and dimension. If one of these variables is selected without entering any information in the other control, this variable name is taken as property name and format and length definition are generated accordingly.

But the Class Builder allows assigning the property another name and format and length can be adapted as long as the new format is data-transfer compatible (see *NaturalX* in the *Programming Guide*). The new property can be marked as read only.

Using an Interface Module

So far class MYEMPLOYEE only defines interfaces internally. But there might be interfaces defined in modules that were adequate to incorporate.

For this purpose an interface module can be linked using the Class's context-menu item **Link > Interface Module**. The interfaces that are defined in this module are then inserted under the corresponding interface module in group "Interface Modules" and at the same time under the group node "Interfaces". To implement their methods, select the corresponding node that can be found under "Interfaces".

Linking a GUID Local Data Area

The Class Builder generates Global Unique IDs for classes and interfaces automatically. But if variables are to be used instead of the generated identifiers, a local data area with the corresponding definition can be linked to MYEMPLOYEE.

The existing Global Unique ID of MYEMPLOYEE can then be changed. Select context menu item **Properties** and activate page **Identifiers**. This page is available for classes and interfaces.

The generated GUID is displayed in the upper control. Local variables that are defined in EMPGUIDS are listed in the lower box. Select EMPGUID and leave the property sheet with **OK**.

Activation Policy

The Class Builder allows setting a class's activation policy explicitly. The current activation policy of MYEMPLOYEE can be viewed under **Settings** if context menu item **Properties** is selected. This option is available for classes only. Select **External Multiple** and leave the property sheet with **OK**.

Save and Stow Class

Up to now the new class MYEMPLOYEE has only existed as an internal class structure. To save all changes the class can be saved and stowed in the class module. This change of state is indicated by the changed icon.

Register

And finally register MYEMPLOYEE by selecting context menu item **Register** on the class node.

Glossary

External Interface

An external interface is an interface which is defined in an interface module, that is included by the class.

Interface Module

An Interface Module is a Natural copycode module which defines interfaces. The Interface Module can be used in a class to define the contained interfaces. The class can overwrite the method and property implementations, but all other settings of the interface are used as defined in the Interface Module.

Internal Interface

An internal interface is an interface which is defined direct in the class, or an interface of an Interface Module, which is defined in the Interface Module.

Method Implementation

A method implementation is a Natural subprogram which is assigned to the method and executed when this method is called for a class object.

Property Implementation

A property implementation is the object data variable that is assigned to a property.

VIII Editor Features With SPoD

91 Editor Features With SPoD

- Map Editor: GUI Controls and Field-Sensitive Maps 536
- Data Area Editor: Source Format 536
- DDM Editor: Database Features 536

The editor features provided with Natural's Single Point of Development (SPoD) depend on the server you use with Natural Studio. This section indicates all server-specific editor features.

For details about the features mentioned in this section, refer to the appropriate sections in the *Editors* documentation.

Map Editor: GUI Controls and Field-Sensitive Maps

Field-sensitive maps
Field-sensitive maps are only available on a Linux or a Windows server.

GUI controls
GUI controls are only available on a Windows server and are not available for output maps (i.e., maps that are based on a `WRITE` statement).

Data Area Editor: Source Format

Depending on the features used, the data area editor uses an internal source format to store the sources of data areas in the FUSER system file.

For details about the source format, see *Source Format for Data Area Storage* in the section *Storing and Cataloging a Data Area* in the *Editors* documentation for mainframes, Linux.

DDM Editor: Database Features

Adabas command procedures
Adabas command procedures are only available on a mainframe server.

DL/I
DL/I-specific features are only available on a mainframe server.

VSAM
VSAM-specific features only available on a mainframe server.

XML
XML-specific features are only available on a Linux or a Windows server.

User databases
User-specific databases can only be defined on a mainframe server.