

Natural

Tools and Utilities

Version 6.3.13 for Windows

October 2012

This document applies to Natural Version 6.3.13 for Windows.

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

Copyright © 1992-2012 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, United States of America, and/or their licensors.

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

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://documentation.softwareag.com/legal/> 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 and license terms, please refer to "License Texts, Copyright Notices and Disclaimers of Third-Party Products". This document is part of the product documentation, located at <http://documentation.softwareag.com/legal/> and/or in the root installation directory of the licensed product(s).

Document ID: NATWIN-NNATUTILITIES-6313-20121005

Table of Contents

Preface	xi
I Utility Activation	1
1 Utility Activation	3
II Component Browser	5
2 Component Browser	7
Tree View	8
Data View	9
Interaction Tree View and Data View	10
Menu Bar	16
Application Development Support	16
III Data Browser	27
3 Data Browser	29
Navigation	30
File Selection List	30
Field Selection	33
Output Report Fields	34
Filter Criteria	36
Report Options	37
Summary	39
Field Properties	40
Results Window	45
IV FTOUCH Utility	53
4 FTOUCH Utility	55
Using the Utility FTOUCH	56
Syntax of ftouch	57
Examples of ftouch	60
V INPL Utility	63
5 INPL Utility	65
Introducing the INPL Utility	66
Load Libraries Only	71
Load DDMs Only	72
Load Error Messages Only	73
Load All Objects	73
Scan INPL File	74
Natural Security Recover	74
User Exit Routines	75
VI Installer	77
6 Installer	79
VII Object Handler	81
7 General Information on the Object Handler	83
Principles of Object Transfer	84
Invoking the Object Handler	86
Batch or Direct Command Calls	87

Issuing Object Handler Commands from a Natural Program	88
Text Members for Reports, Restarts and Traces	88
Natural Security	88
Using FDDM System Files	89
8 Functions	91
9 Wizards	93
10 General Information on Wizards	95
Invoking Wizards	96
Navigation and Command Execution	96
11 Unload Wizard	97
Unload Objects into Natural Work File(s)	98
Find Objects	101
Start Object Handler Command Procedure	102
12 Load Wizard	103
Load/Scan Objects from/in Work Files	104
Start Object Handler Command Procedure	107
Load SYSPAU Application	108
13 Advanced User	109
Activating Advanced User	110
Advanced User Unload	111
Advanced User Load	112
14 Restart Load	113
Invoking Restart Load	114
Specifying Permanent Members	115
15 View	117
Invoking View	118
Terminating View	119
Navigating	119
Saving Object Selections	120
Sorting Objects	120
Listing Objects Separately	120
Deleting Objects	121
16 Find	123
Find in Advanced User Mode	124
17 Scan	127
Scan in Advanced User Mode	128
18 Administration	131
Administration Wizard	132
Advanced User Administration	134
Change Workplan Library	136
19 Object Specification	139
20 Object Specification - All Objects	141
21 Object Specification - Natural Library Objects	143
Natural Library Objects	144
Natural Library Object Details	145

Natural Library Object Properties	147
Natural Library Object Exceptions	149
Natural Library Object Exception Properties	150
22 Object Specification - Natural System Error Messages	151
Natural System Error Messages	152
Natural System Error Message Details	153
Natural System Error Message Exceptions	153
23 Object Specification - Natural Command Processors	155
Natural Command Processor Sources	156
Natural Command Processor Source Exceptions	157
24 Object Specification - Natural DDMs	159
Natural DDMs	160
Natural DDM Details	161
Natural DDM Exceptions	162
25 Object Specification - Natural-Related Objects	163
Natural-Related Objects - Windows, UNIX and OpenVMS	164
Natural-Related Objects - Mainframes	167
26 Object Specification - External Files	171
External Files	172
External File Details	173
External File Exceptions	174
27 Object Specification - FDTs	175
28 Use Selection or List	177
29 Settings	179
30 Settings - Options	181
Set Additional Options	183
31 Settings - Parameters	191
Set Global Parameters	192
32 Workplans	197
Creating, Selecting and Modifying Workplans	198
Contents of Workplans	198
Examples of Workplans	199
Referencing Workplans	200
33 Name, Date and Time Specification	203
Name	204
Date	205
Time	206
34 Work Files	207
Work File Assignment	208
Work File Format	209
35 Direct Commands	213
36 Basic Command Syntax	215
37 select-clause	219
Syntax of select-clause	220
SELECTION or LIST Workplan	220

Natural Library Object and DDM Selection	221
Natural-Related Object Selection	227
Natural-Related Debug Environment Selection	229
Natural-Related Profile Selection	231
Natural-Related DL/I Subfile Selection	232
Natural System Error Message Selection	234
Natural Command Processor Selection	236
External File Selection	237
FDT Selection	239
Application Selection	240
Object Selection for Delete Instructions	242
38 Object List - LIST Workplan	245
Syntax of object-type-and-location	246
Syntax of object-name-description	247
Example of an Object List	249
39 parameter-setting	251
Syntax of parameter-clause	252
Keyword Explanation of parameter-clause	253
40 option-setting	257
Syntax of option-setting	258
Keyword Explanation of option-setting	260
41 Examples of Using Direct Commands	267
Unloading Objects for the Same Platform	268
Unloading Objects for Different Platforms	269
Loading Objects in Internal Format	269
Loading Objects in Transfer Format	270
Batch Processing in a Remote Environment	270
42 Batch Condition Codes and User Exit Routines	273
Condition Codes Returned in Batch	274
Applying User Exit Routines	274
User Exit Routines Available	275
43 Tools	277
Status	278
Last Result	278
Traces	278
Reports	279
Transfer Work File	279
44 Options	281
Settings	282
Profile	282
Advanced User	282
Free Format Editing	282
Details	283
Single Objects	283
Display Command	283

45 Profile Settings	285
46 Migration from SYSTRANS to the Object Handler	287
Converting Individual Commands	288
Processing SYSTRANS Commands with OBJHAPI	289
Unsupported SYSTRANS Options	289
VIII SYSAPI Utility - APIs of Natural Add-On Products	291
47 SYSAPI Utility - APIs of Natural Add-On Products	293
Prerequisites	294
Invoking and Terminating SYSAPI	294
SYSAPI Tree View Items	295
Performing SYSAPI Utility Functions	296
IX SYSCP Utility - Code Page Information	299
48 SYSCP Utility - Code Page Information	301
Invoking and Terminating SYSCP	302
All Code Pages	304
Unicode Properties	308
General Information	309
X SYSERR Utility	311
49 General Information on Messages	313
Message Types	315
Message Languages	315
Issuing Messages	316
Retrieving Natural System Short Messages	317
Retrieving User-Defined Short Messages	317
Obtaining Message Information	318
50 Invoking SYSERR	319
Invoking SYSERR for User-Defined Messages	320
Invoking SYSERR for Natural System Messages	322
51 SYSERR Utility Window and Functions	323
List Box	325
Fields	325
Command Buttons	326
File Menu	327
Edit Menu	329
View Menu	331
Options Menu	335
Toolbar Buttons	338
Status Bar	339
Online Help	340
52 Converting Natural System Short Messages	341
53 Generating Message and Text Files	343
Storing a Message File	344
Creating a Text File	344
Generating a Message File	345
Recreating a Text File	346

54 Managing Messages in Different Libraries	347
55 Application Programming Interface USR0020P	349
XI SYSEXT Utility - Natural Application Programming Interfaces	351
56 SYSEXT Utility - Natural Application Programming Interfaces	353
Prerequisite	354
Introduction to SYSEXT	354
Invoking and Terminating SYSEXT	356
SYSEXT Tree View Items	357
Performing SYSEXT Utility Functions	358
Interface Versions	361
Reserved Keywords	361
Using a Natural API	361
List of Natural APIs	362
XII SYSEXV Utility	369
57 SYSEXV Utility	371
Executing Example Programs of Current Versions	372
Executing Example Programs of Non-current Versions	372
Terminating the SYSEXV Utility	373
XIII SYSMAN Utility - Object Maintenance	375
58 General Information on SYSMAN	377
59 Invoking and Terminating SYSMAN	379
Invoking SYSMAN	380
Terminating SYSMAN	382
60 Listing Objects	383
61 Finding Objects	387
62 Copying Objects	391
63 Moving Objects	397
64 Deleting Objects	403
65 Renaming Objects	407
66 Importing Objects	411
67 Using SYSMAN with Subprogram	415
Invoking and Executing MAINUSER	416
Using Commands	416
LIST and FIND Command Syntax	417
COPY and MOVE Command Syntax	418
DELETE Command Syntax	418
RENAME Command Syntax	419
IMPORT Command Syntax	419
where-clause	420
with-clause	420
Keywords and Variables in Commands	420
Specifying a Range of Names	425
68 XRef Considerations	427
Processing of XRef Data	428
XRef Processing Errors	429

69 Security Considerations for Administrators	431
File Security in Remote Environments	432
Natural Security	434
XIV SYSNCP Utility	437
70 SYSNCP Utility	439
Prerequisites for Windows	440
Introducing the SYSNCP Utility	440
Invoking SYSNCP	448
Processor Selection	449
Header Records	450
Keyword Maintenance	459
Function Maintenance	464
Runtime Actions	469
Processor Cataloging	475
Administrator Services	476
Session Profile	483
XV SYSRPC Utility	487
71 Basic SYSRPC Functions	489
Invoking SYSRPC	490
Terminating SYSRPC	491
Service Directory Tree	492
Menu Bar	492
Toolbar	495
Context Menu	496
72 Service Directory Maintenance	497
Service Directory Concept	498
Tree Nodes	500
Logon Option	504
Transport Protocol	504
73 Generating Interface Objects	505
74 Generating Single Interface Objects with Parameter Specification	507
Using the Interface Object Generation Function	508
Specifying Parameters	510
Examples of Interface Object Generation	512
75 Generating Multiple Interface Objects	515
Using the Function Interface Object Mass Generation	516
Using the SYSRPC SGMASS Command	520
Name Specification and Compression	520
76 Generating Interface Objects or PDAs from IDL Files	523
Building a Selection List	526
77 Calculating Size Requirements	529
Using the Function Interface Object Mass Calculation	530
Using the SYSRPC CSMASS Command	534
Name Specification and Compression	534
78 Server Command Execution	537

Message Display Window	538
Pinging a Server	539
Terminating a Server	540
XVI Tamino Server Extensions	543
79 Tamino Server Extensions	545
Overview	546
Developing a Tamino Server Extension	547
Using Callbacks	553
Deploying a Tamino Server Extension	553
Installing a Tamino Server Extension	554
Tamino Server Extension Example	554

Preface

The *Tools and Utilities* documentation explains how Natural invokes a utility and describes the utilities available in Natural.

Utility Activation	Describes how Natural invokes a utility.
Component Browser	This utility views ActiveX components that are available for developing NaturalX applications.
Data Browser	This utility is used to generate, print and store data reports from Natural DDMs (data definition modules).
FTOUCH	This utility makes a downloaded object executable by Natural.
INPL	This utility loads or scans Natural objects and shared resources supplied by Software AG.
Installer	This utility helps install Natural add-on products.
Object Handler	This utility processes Natural and non-Natural objects for distribution in Natural environments. This is done by unloading the objects in the source environment into work files and loading them from work files into the target environment.
SYSAPI	This utility locates Application Programming Interfaces (APIs) provided by Natural add-on products.
SYSCP	This utility provides code page information.
SYSERR	This utility creates application-specific messages. In addition, it can be used to modify the texts of the existing Natural system messages (not recommended).
SYSEXT	This utility locates Natural Application Programming Interfaces (APIs).
SYSEXV	This utility contains examples of the new features of the current Natural versions.
SYSMAIN	This utility performs object operations in Natural such as copy, move, delete and import.
SYSNCP	This utility defines command-driven navigation systems for Natural applications.
SYSRPC	This utility establishes and maintains Natural Remote Procedure Call environments.
Tamino Server Extensions	This utility extends the query and mapping Tamino Server functionality by adding user-defined logic.

I Utility Activation

1 Utility Activation

Natural invokes a Natural utility without performing a logon to the corresponding utility library in the FNAT system file. As a result, Natural preserves the global data area (GDA) and/or application-independent variables (AIV). The current user library and the settings are maintained. (To reset the GDA and/or the AIVs, see the profile parameter `FREEGDA` in the *Parameter Reference*.)

To preserve the settings of your application environment, do *not* log on to a utility library. Instead, invoke a utility by using the Natural system command that corresponds to the utility.

After terminating a utility, you will be returned to the library from which you invoked the utility. However, if you explicitly log on to a utility library before invoking the utility, you will stay in this (utility) library after utility termination.

Exception:

The utilities SYSEXT and SYSEXV still perform an implicit logon to the corresponding utility library since object sources can only be edited within an active library.

For information on how to control the use of Natural utilities with Natural Security, see the section *Protecting Utilities* in the *Natural Security* documentation.

II

Component Browser

2 Component Browser

- Tree View 8
- Data View 9
- Interaction Tree View and Data View 10
- Menu Bar 16
- Application Development Support 16

The Component Browser can be used to view ActiveX components which are available for developing NaturalX applications. It presents the information in a way that is especially useful to Natural application developers.

The Component Browser comprises the following features:

- Available ActiveX components and their dispatch and dual interfaces are listed.
- Data types are mapped to Natural data formats.
- The external components' help files are directly accessible.
- Natural programming examples are automatically generated.
- Many programming errors can be prevented.

The Component Browser uses a split window. The left pane contains a tree view that represents the available external components, and the right pane contains the data view that provides information on a selected node item.

Tree View

At startup the Component Browser's tree view consists of four nodes that group the available external components:

- **All ActiveX Components**

This group lists ActiveX controls and Automation Objects.

- **ActiveX Controls**

This group lists only the ActiveX controls.

- **Automation Objects**





This group lists the Automation Objects.

- **Interfaces**

This group lists all dual and dispatch interfaces that can be addressed in a Natural application. In this context, their relation to an ActiveX component is not taken into account.

In general, a node in the tree view represents either an ActiveX component or an interface. It provides textual information on the node and has a specific icon assigned that represents additional information.

The following table lists all available nodes with their icons and gives a short description:

Type	Icon	Description
Group		Group node.
ActiveX component		ActiveX component.
Interface		Interface of the current ActiveX component.
Default interface		Default interface of the current ActiveX component.

By default, the ActiveX component nodes are inserted in alphabetical order of their external names. Interface nodes are always inserted in alphabetical order.

► **To sort ActiveX component nodes according to their ProgID**

- From the **View** menu, choose **Show by ProgID**.

Data View

The data view uses a property sheet to display the specific information for a selected node. This sheet consists of four tabbed pages:

- **General**

Components and interface specific information such as name, globally unique identifier (GUID) and help file name. It is always the active page if a new node is selected.

- **Properties**

Specific information on the properties offered by an interface. These are, for example, the property's name and type.

- **Events**

Specific information on a component's event interface. These are, for example, the event's name and parameters.

- **Methods**

Specific information on the methods and complex properties (that is, properties with parameters) offered by an interface. Displayed are, for example, the method's name, return type and parameters.

These pages are discussed in more detail in the context of interaction between tree and data view.

Interaction Tree View and Data View

The contents of the data view depend on the node selected in the tree view.


If any of the group nodes **All ActiveX Components**, **ActiveX Controls**, **Automation Objects** or **Interfaces** is selected, the data view remains empty.

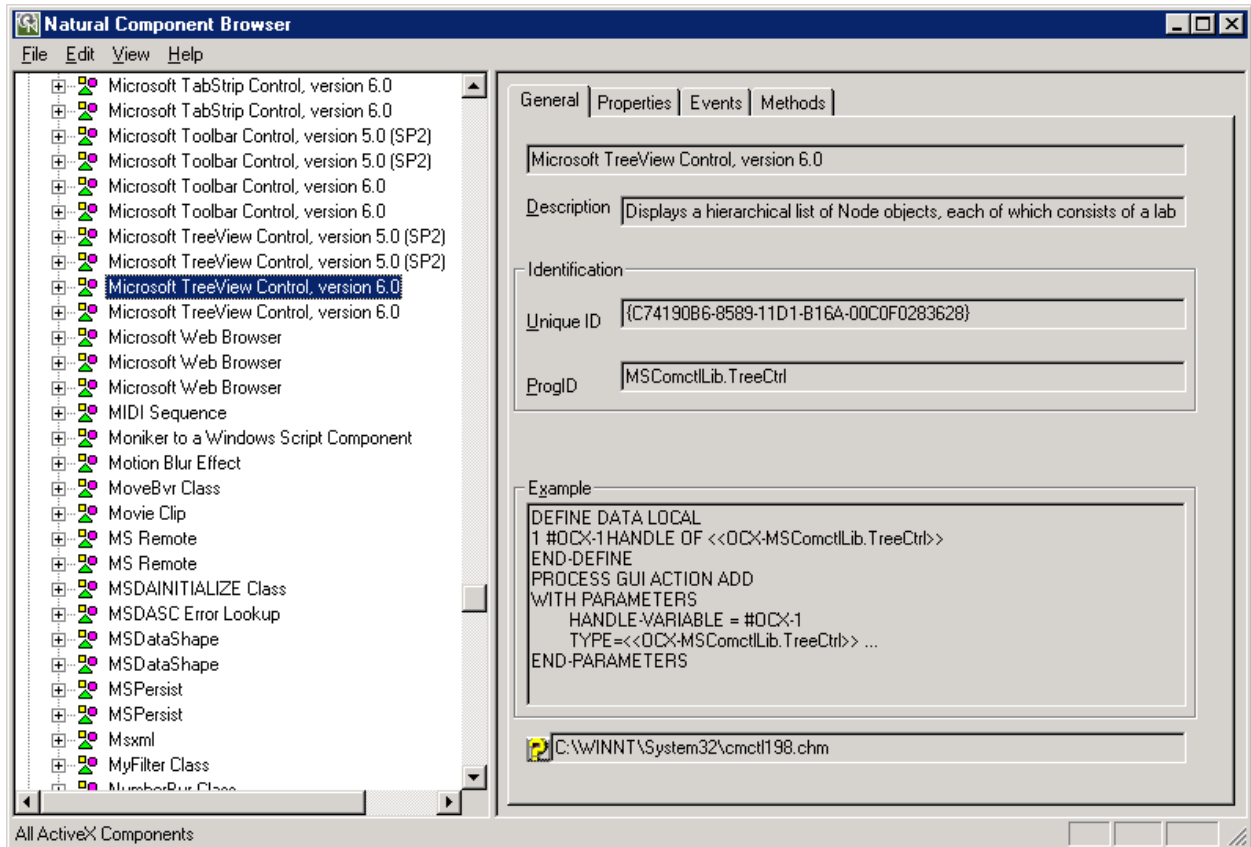
This section describes the tabbed pages that may be displayed in the data view.

- [ActiveX Component](#)
- [Interface](#)

ActiveX Component

If a component node is selected, all four tabbed pages are available. The page **General** provides the following information:

Name	Component name
Description	Short textual description.
Unique ID	GUID.
ProgID	ProgID.
Example	Example Natural source code. See also Application Development Support .
Help	Help file name. This file can be opened by choosing 



If a component is selected and the page **Properties**, **Events** or **Methods** is activated, the information displayed on this page refers to the default interface.

Interface

If an interface node is selected, the number of available tabbed pages depends on the type of the interface.

- If an **interface** is browsed in the context of a component and if it is an **event interface**, then only the pages **General** and **Events** are set.
- Otherwise, the pages **General**, **Properties** and **Methods** are set.


Each page is described in the following section.

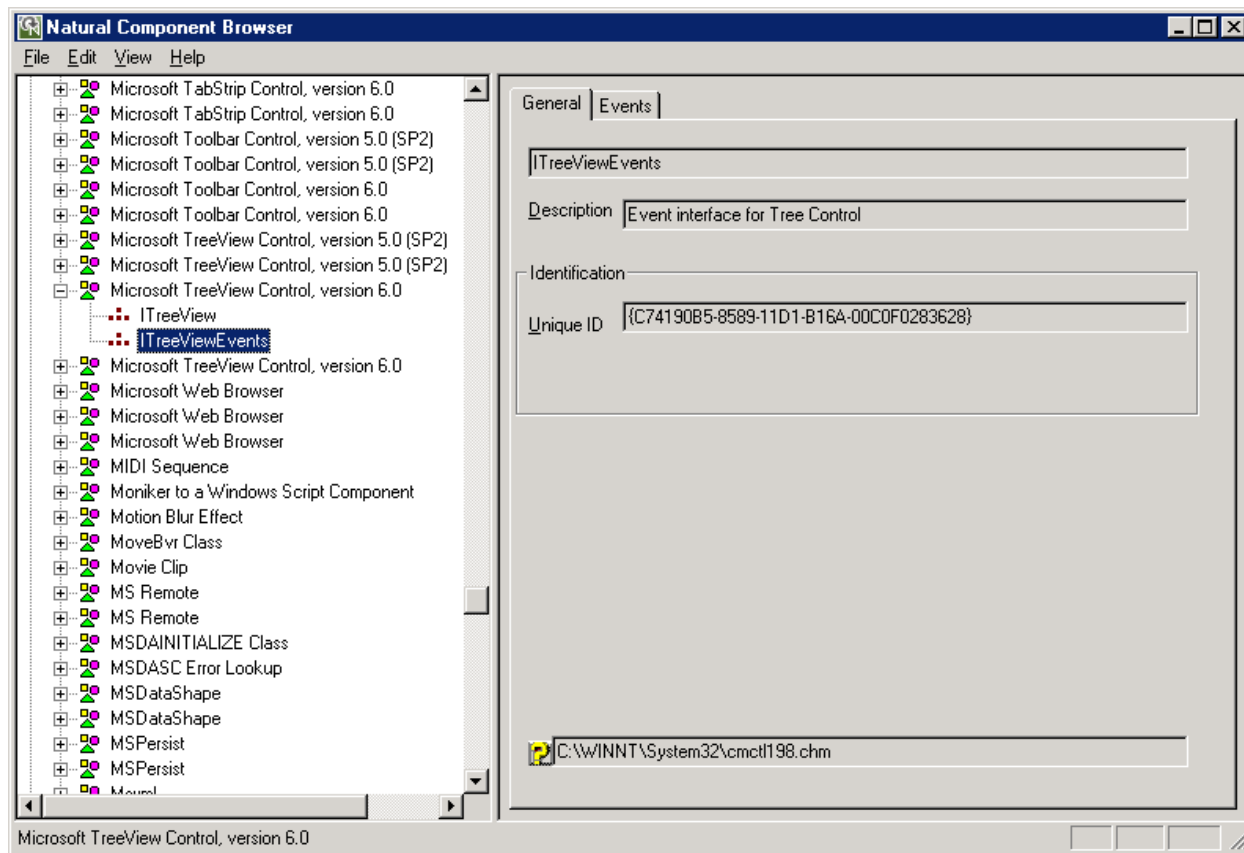
- [General](#)
- [Properties](#)
- [Methods](#)

- Events

General


The page **General** provides the following information:

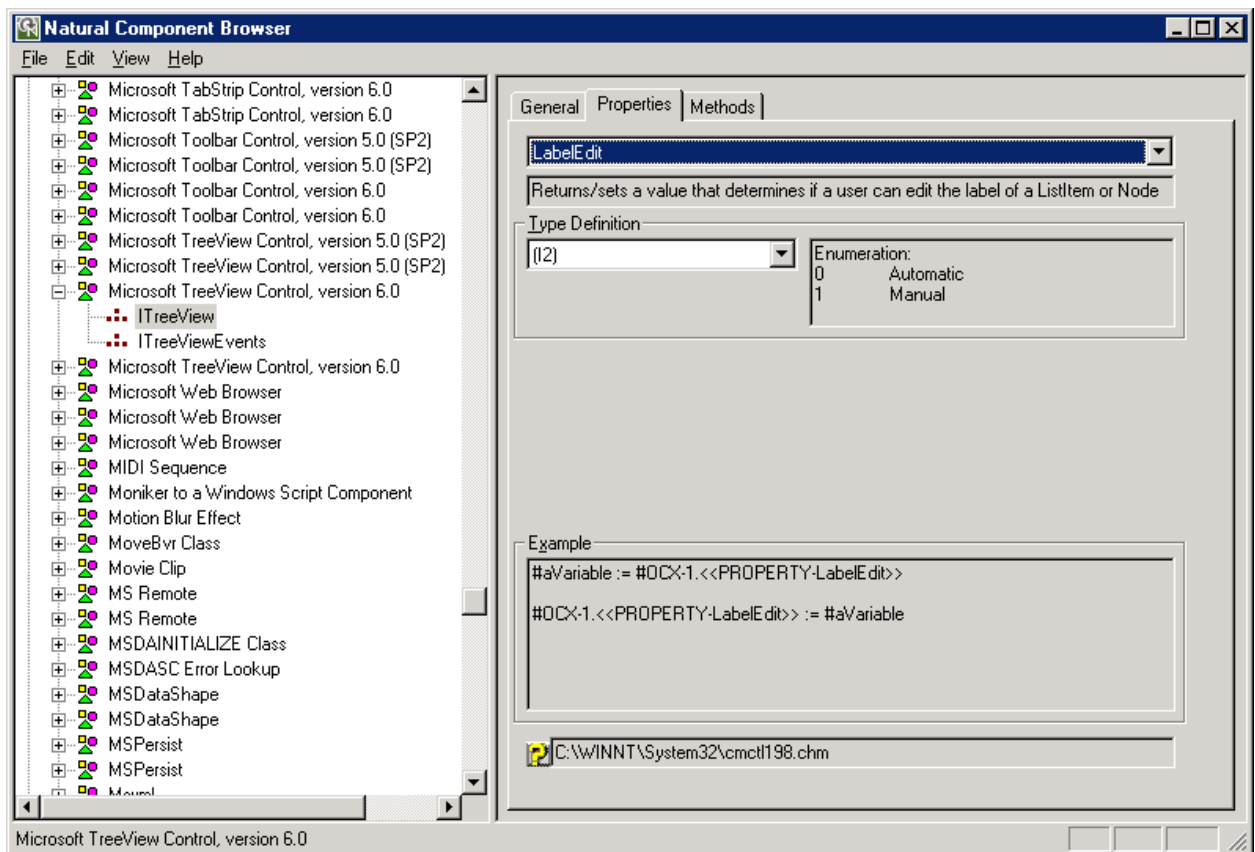
Name	Interface name
Description	Short textual description.
Unique ID	GUID.
Help	Help file name. This file can be opened by choosing 



Properties


The page **Properties** provides the list of properties that belong to the selected interface. For a specific property, the following information is displayed:

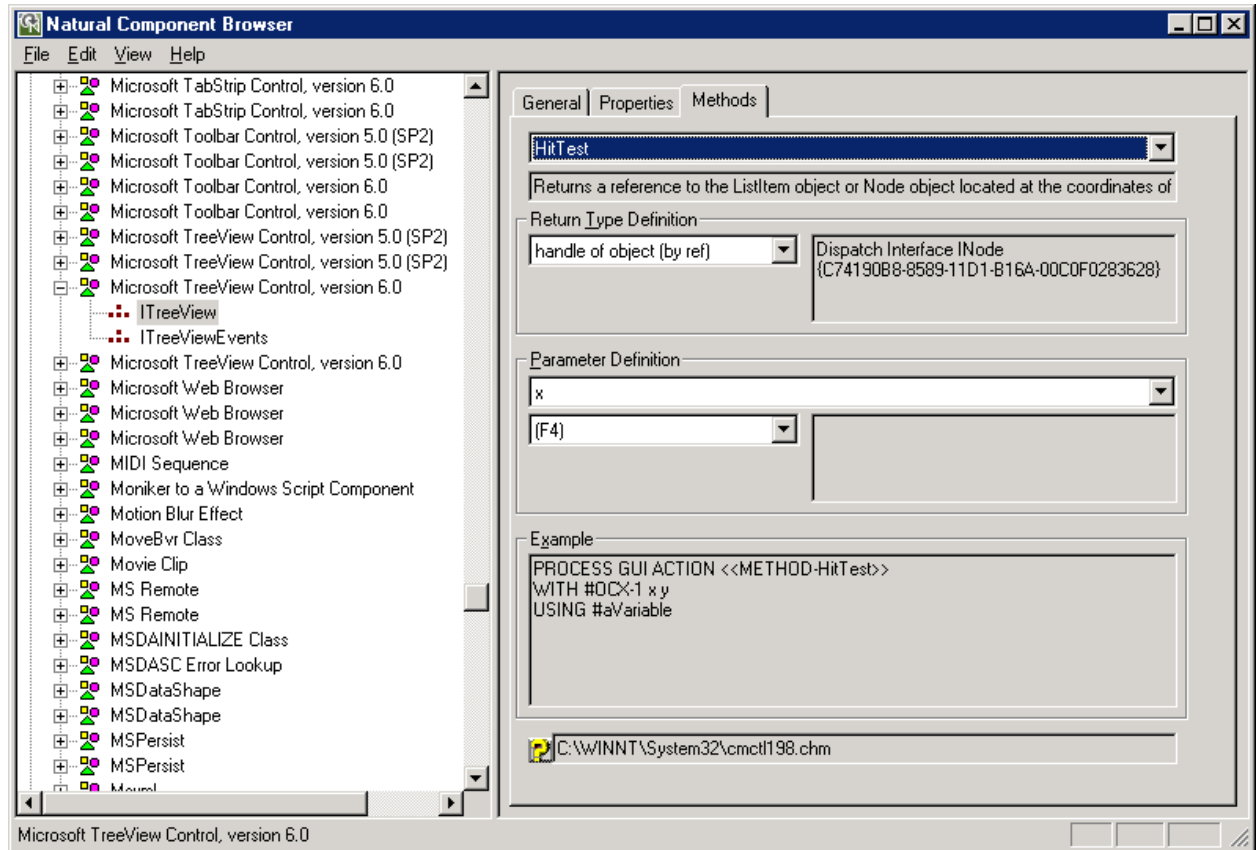
Description	Short textual description for the selected property.
Type Definition	List of valid Natural data formats for the property's type. Additional type info on the selected Natural data format, e.g. valid values for enumeration types.
Example	Example Natural source code. See also Application Development Support .
Help	Help file name. This file can be opened by choosing 



Methods


The page **Methods** provides the list of methods that belong to the selected interface. This list includes properties with parameters. For a specific method or complex property, the following information is displayed:

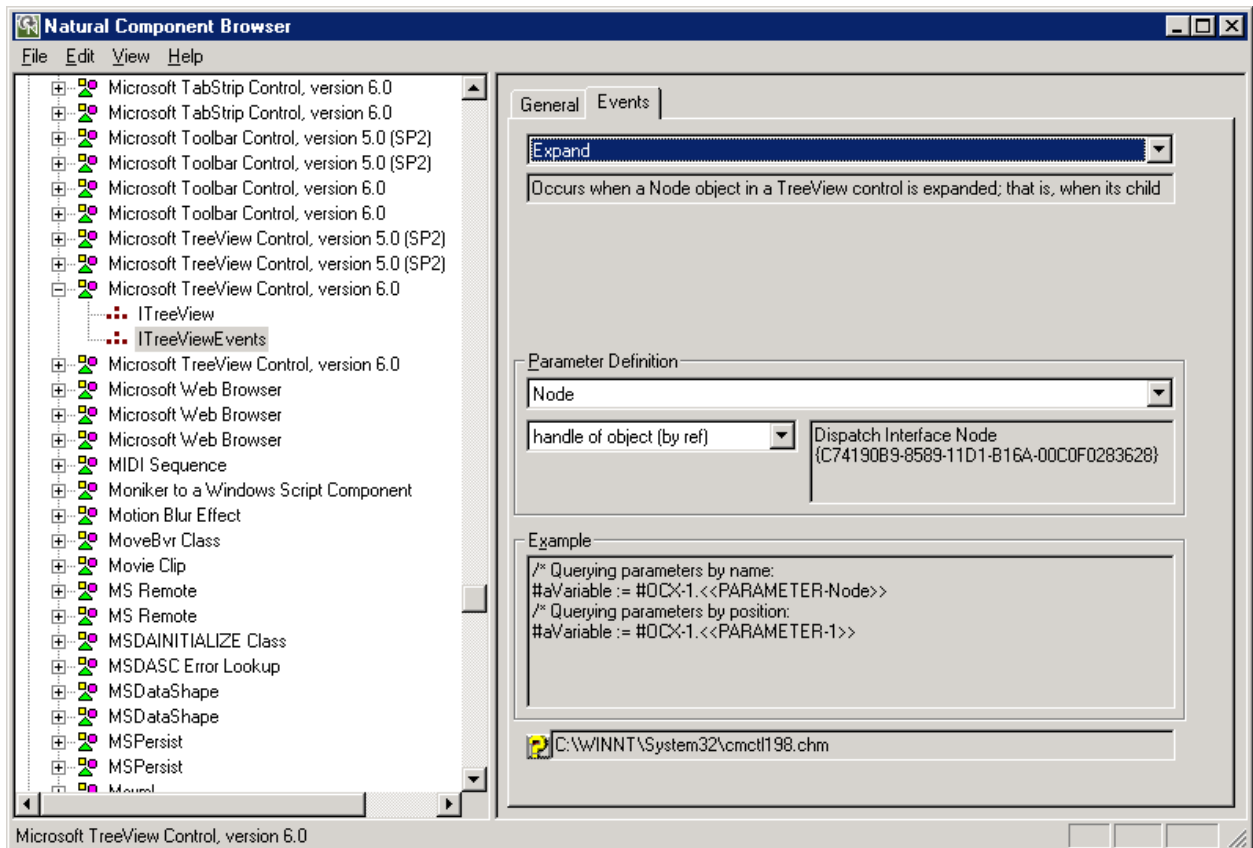
Description	Short textual description for the selected method.
Return Type Definition	List of valid Natural data formats for the method's type. Additional type information on a selected Natural data format, e.g. Name and GUID of the interface if a handle of object corresponds to a dispatch interface. This interface can then be found in group Interfaces .
Parameter Definition	List of parameters that are required by the method. List of Natural data formats for each parameter together with additional info on the call mode (by ref). Additional type info on a selected Natural data format, e.g. Name and GUID of the interface if a handle of object corresponds to a dispatch interface. This interface can then be found in group Interfaces .
Example	Example Natural source code. See also Application Development Support .
Help	Help file name. This file can be opened by choosing 



Events

The page **Events** provides the list of events that belong to the selected event interface. For a specific event, the following information is displayed:

Description	Short textual description for the selected event.
Parameter Definition	List of parameters that are required by the event. List of valid data Natural data formats for each parameter together with additional info on the call mode (by ref). Additional type info on selected Natural data format, e.g. Name and GUID of the interface if a handle of object corresponds to a dispatch interface. This interface can then be found in group Interfaces .
Example	Example Natural source code. See also Application Development Support .
Help	Help file name. This file can be opened by choosing 



Menu Bar

The following menus are available in the Component Browser window:

Menu	Item	Description
File	EXIT	Leaves the Component Browser.
Edit	Copy	This item is enabled if the data view has the focus. Copies the selected text to the clipboard.
	Copy CLSID to Clipboard	This item is enabled if either the tree view or the data view has the focus. Copies a component's CLSID to the clipboard.
	Copy ProgID to Clipboard	This item is enabled if either the tree view or the data view has the focus. Copies a component's ProgID to the clipboard.
	Find Unique ID	This item is enabled if the tree view has the focus. Searches for a Unique ID in the selected group of components.
View	Show by ProgID	This item is enabled if the tree view has the focus. By default, the tree view is sorted according to the component's external names. If this option is checked, it is sorted according to the component's ProgIDs. The currently displayed information is updated.
	Show Current Version	This item is enabled if the tree view has the focus. By default, the tree view shows all versions of a component. If this option is checked, only the current version of a component is shown. The currently displayed information is updated.
	Status Bar	Shows or hides the status bar.
	Refresh	This item is enabled if the tree view has the focus. Refreshes the tree view, that is, the information currently displayed is updated.
Help	About Component Browser	This item is enabled if either the tree view or the data view has the focus. Displays general information on the Component Browser such as Copyright and Version.

Application Development Support

The **Example** boxes in the pages of the data view provide detailed examples of Natural source code. These examples show how to use a selected object in a Natural application. Which statements are generated depends on the object's type.

The example source code or just parts of it can be selected, copied to the clipboard and used directly in an application. Only variable names might have to be adapted to meet the application's requirements.

▶ **To copy frequently used identifiers to the clipboard**

- From the **Edit** menu, choose **Copy CLSID to Clipboard**.

Or:

From the **Edit** menu, choose **Copy ProgID to Clipboard**.

This section covers the following topics:

- [ActiveX Controls](#)
- [Automation Objects](#)
- [Interfaces](#)

ActiveX Controls

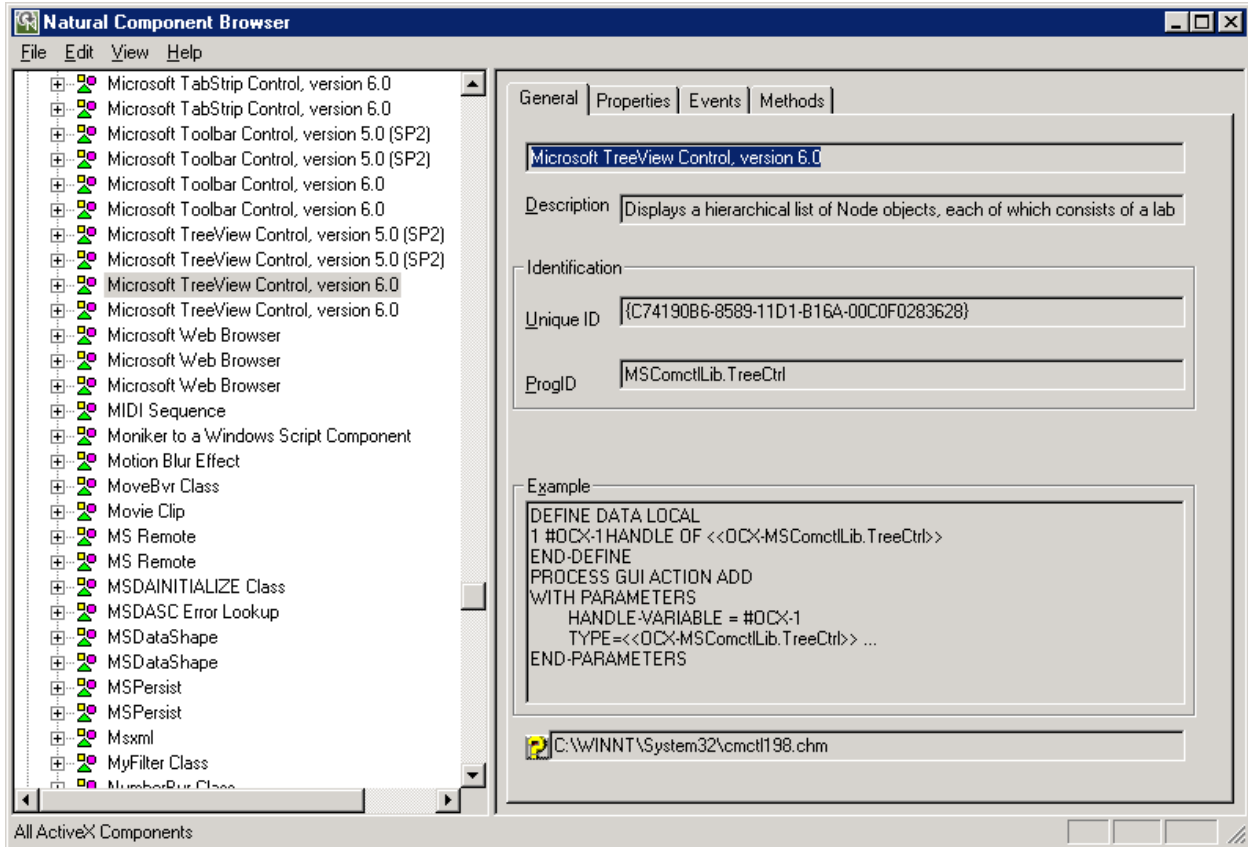
For ActiveX controls, the appropriate `PROCESS GUI` statement is generated that shows how to use these components in Natural applications.

This section describes example source code shown on the pages provided for items contained in the **ActiveX Controls** node.

- [General](#)
- [Properties](#)
- [Events](#)
- [Methods](#)

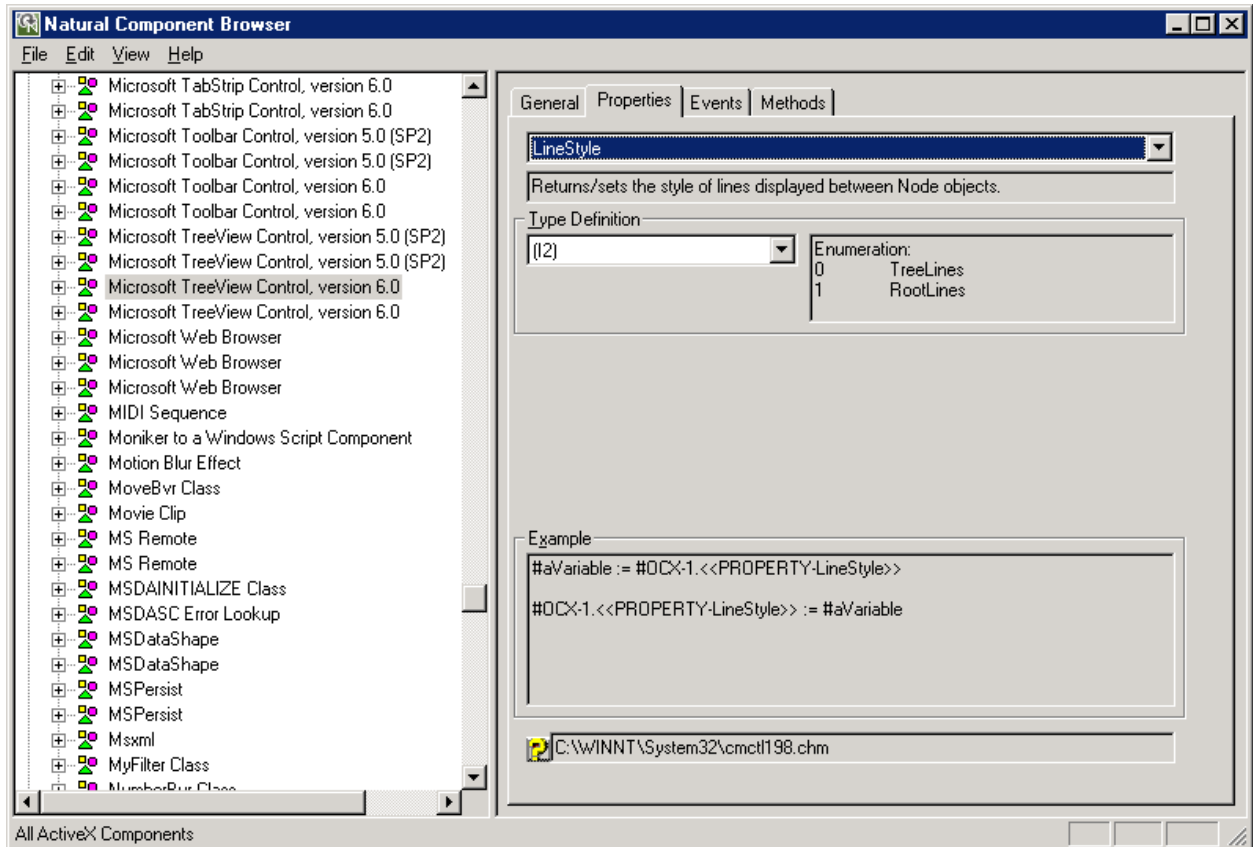
General

The example on the page **General** shows how an object of this type is instantiated. Here `#OCX - 1` denotes a variable that can be adapted to the current application.



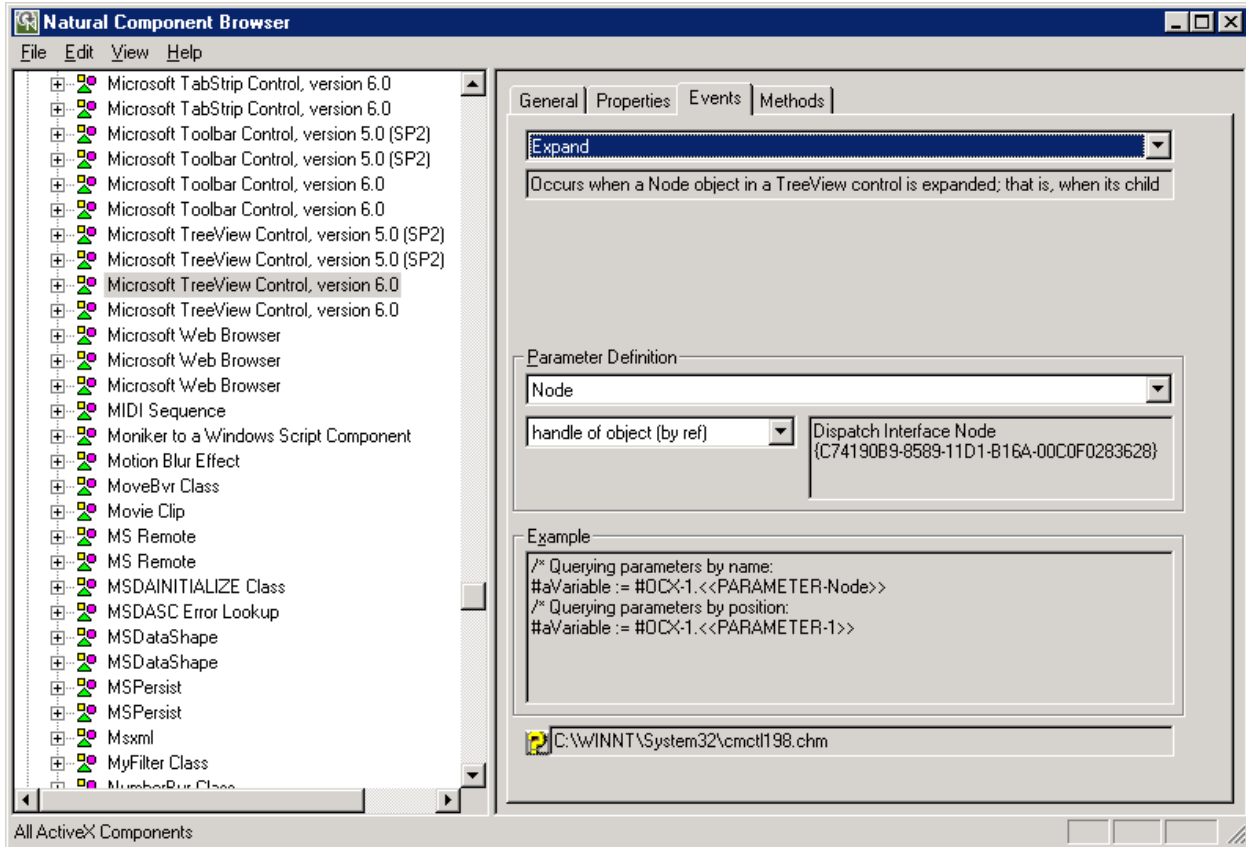
Properties

The example on the page **Properties** shows how to assign a property to a variable and how to assign a variable to a property. Here #OCX-1 is the defined object handle and #aVariable refers to an application-specific variable. Both names can be adapted if required.



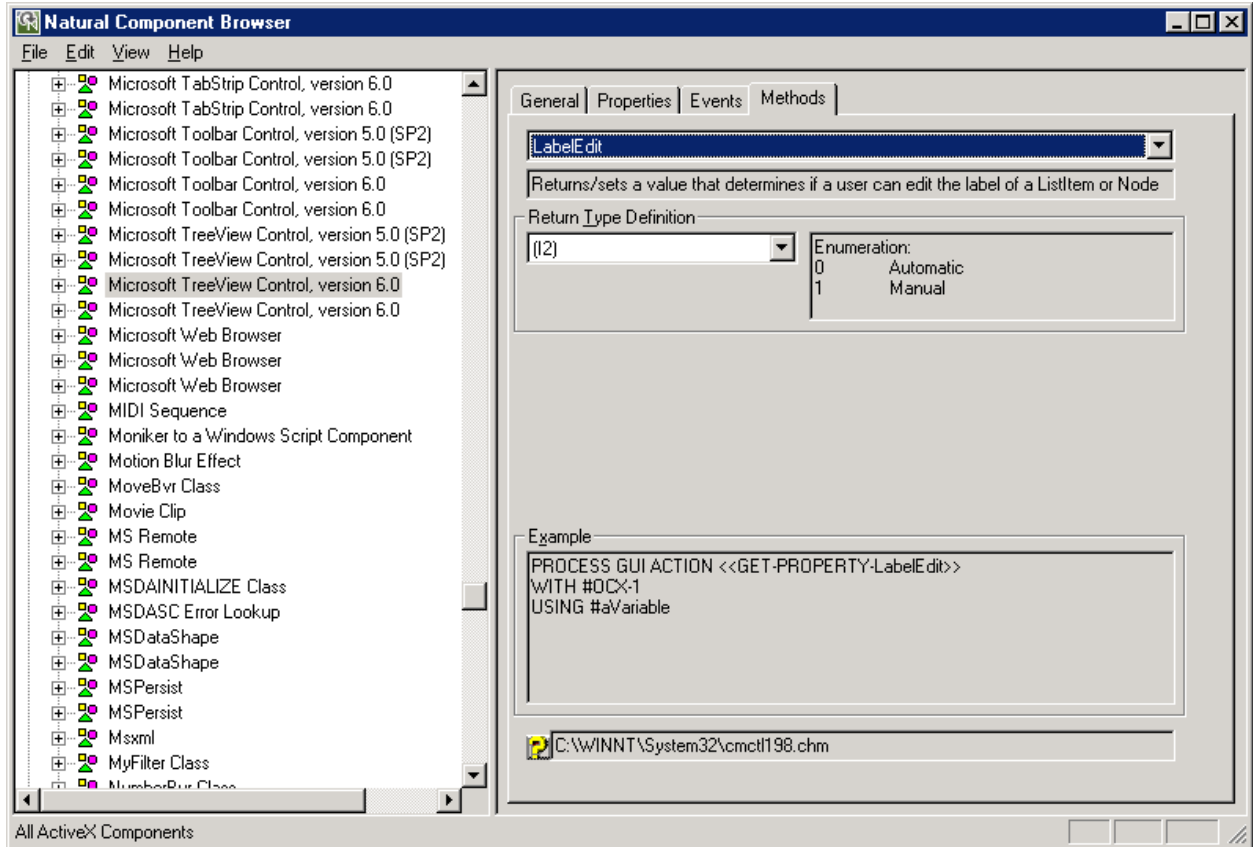
Events

The example on the page **Events** shows how to query event parameters by name or by position. Here `#OCX-1` is the defined object handle and `#aVariable` refers to an application-specific variable. Both names can be adapted if required.



Methods

The example on the page **Methods** shows how to use methods and properties with parameters. Here #OCX-1 is the defined object handle and #aVariable refers to an application-specific variable. Both names can be adapted if required. The actual parameter names are already inserted into the statement if they are available. Otherwise, default parameter names such as P0 and P1 are used as placeholders. These names can be replaced by application-specific variables.



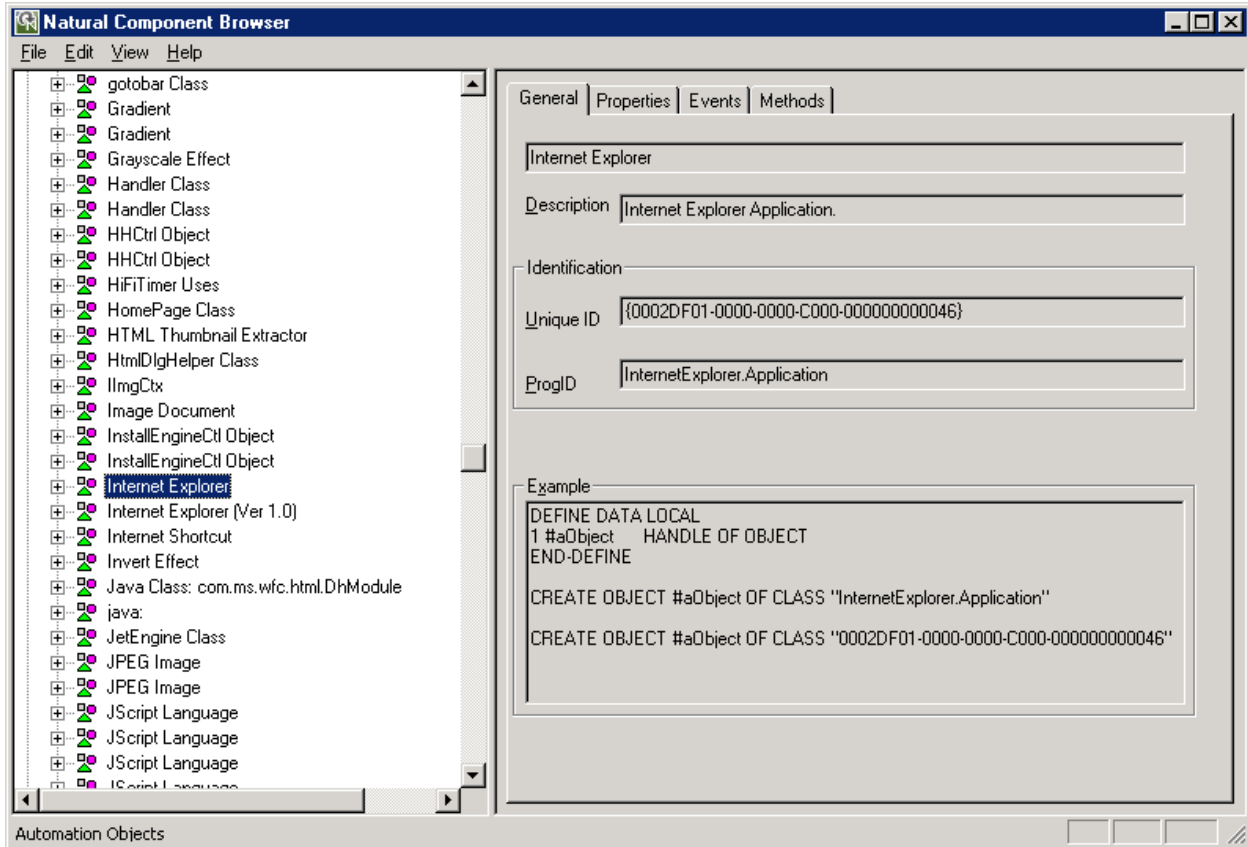
Automation Objects

This section describes example source code shown on the pages provided for items contained in the **Automation Objects** node.

- [General](#)
- [Properties](#)
- [Methods](#)

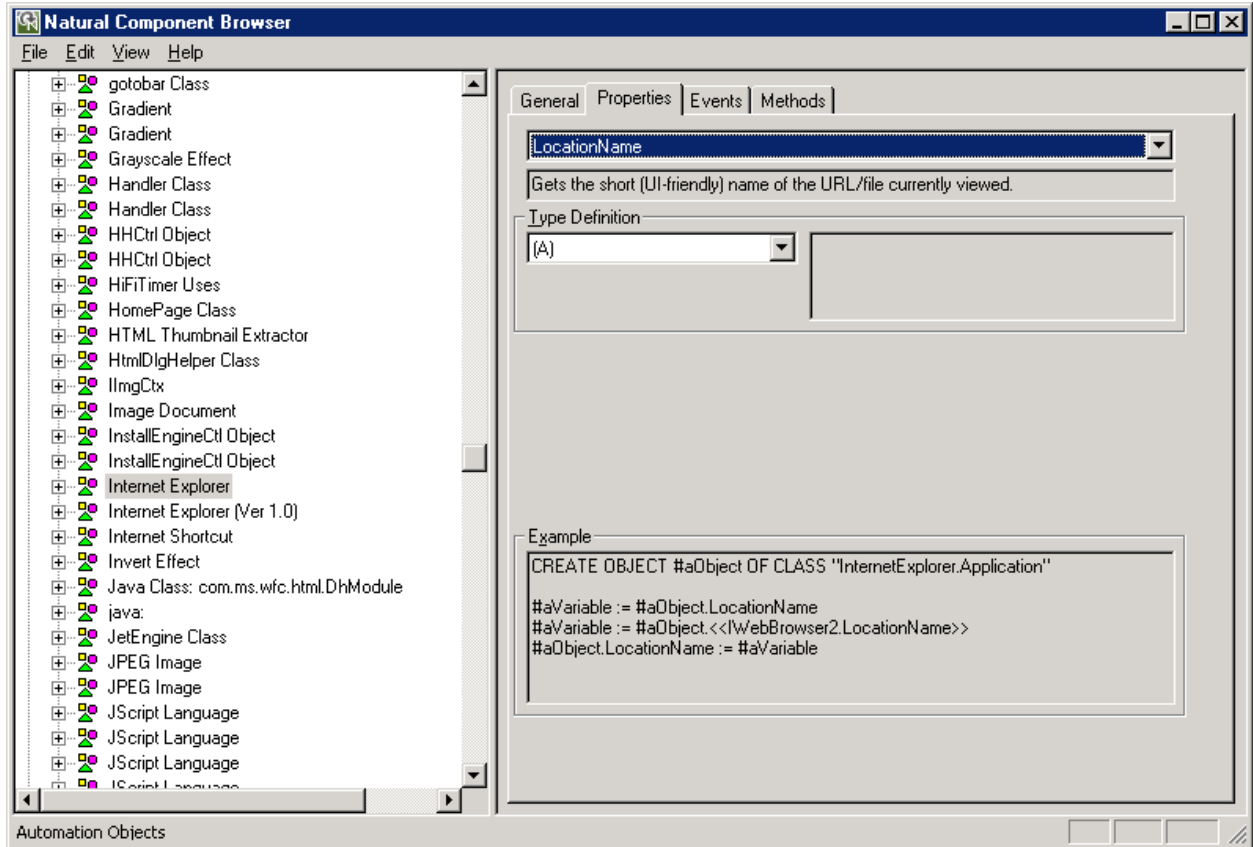
General

The example on the page **General** shows how an object of this type is instantiated. Here `#aObject` denotes a variable that can be adapted to the current application.



Properties

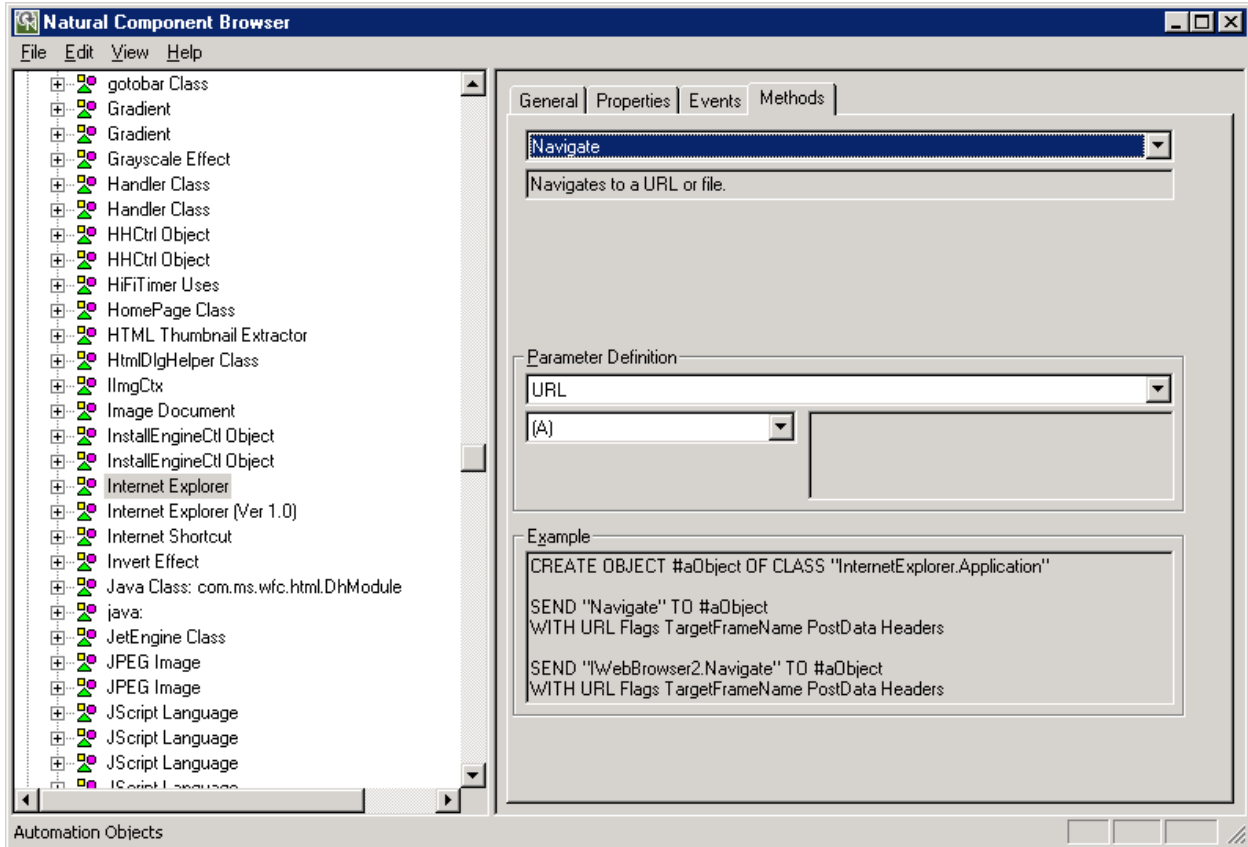
The example on the page **Properties** shows how to assign a property to a variable and how to assign a variable to a property. Here `#aObject` is the defined object handle and `#aVariable` refers to an application-specific variable. Both names can be adapted if required.



Methods

The example on the page **Methods** shows how to use methods. Here `#aObject` is the defined object handle and `#aVariable` refers to an application-specific variable. Both names can be adapted if required.

The actual parameter names are already inserted into the statement if they are available. Otherwise, default parameter names such as `P0` and `P1` are used as placeholders. These names can be replaced by application-specific variables.



Interfaces

For interfaces that belong to group **Interfaces** and that are not considered in the context of a class, the examples are generated as for Automation Objects. Only the `CREATE OBJECT` statement is left aside.

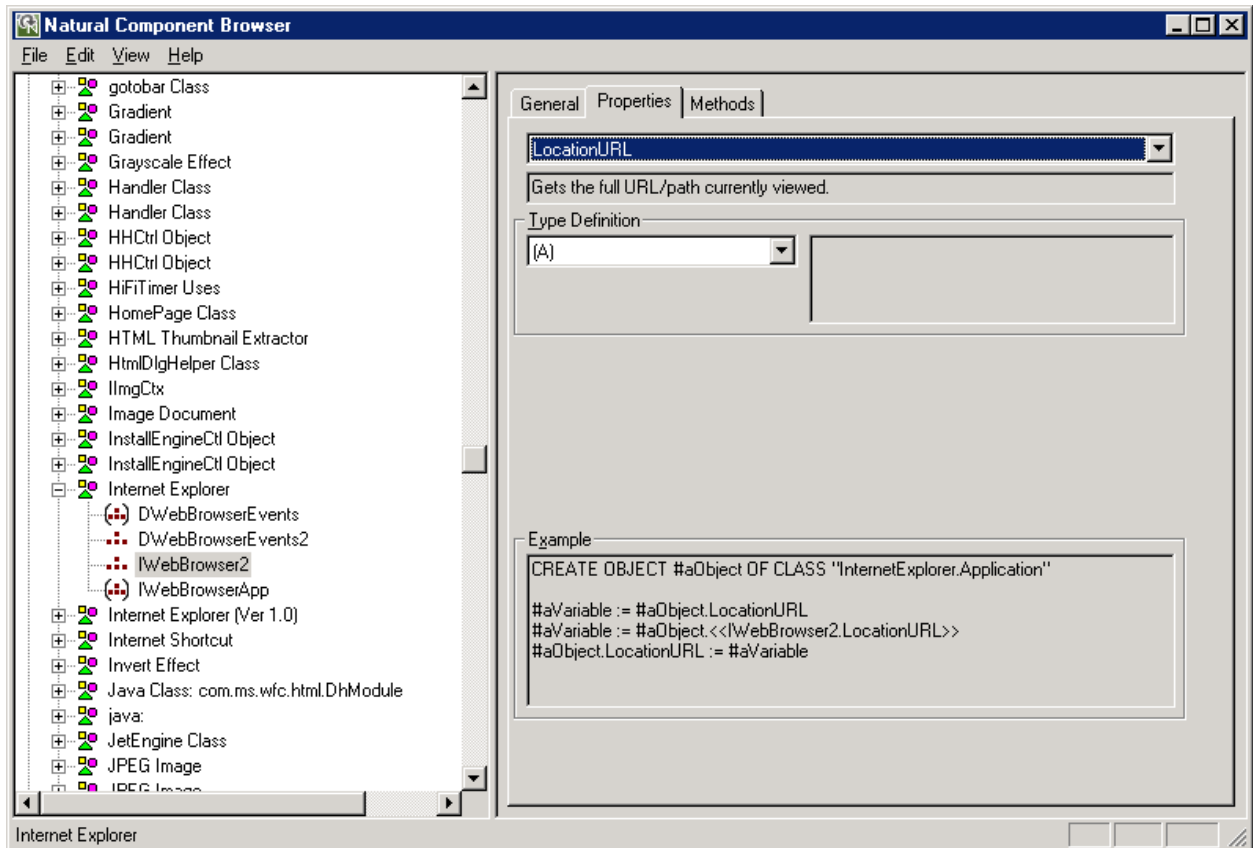
This section describes example source code shown on the pages provided for items contained in the **Interfaces** node.

- [Properties](#)

- **Methods**

Properties

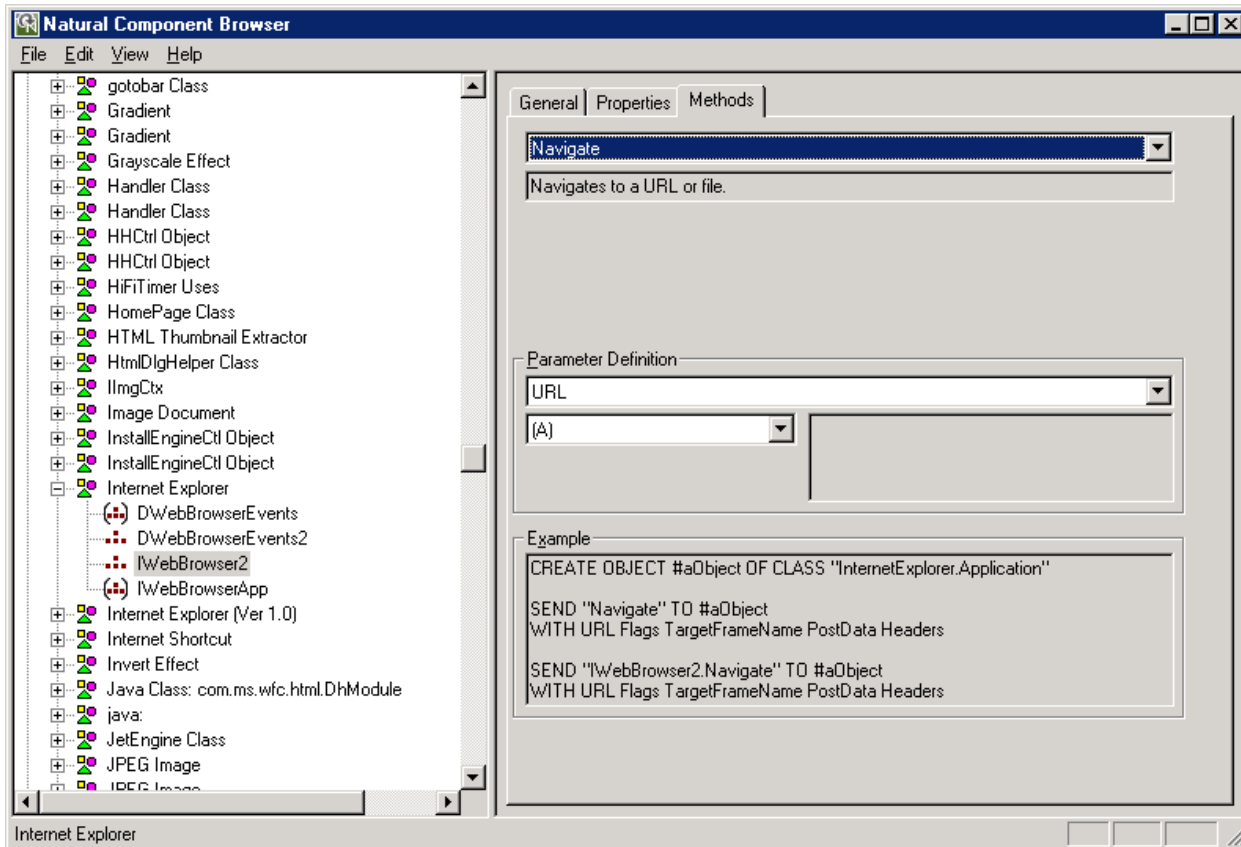
The example on the page **Properties** shows how to assign a property to a variable and how to assign a variable to a property. Here `#aObject` is the defined object handle and `#aVariable` refers to an application-specific variable. Both names can be adapted if required.



Methods

The example on the page **Methods** shows how to use methods. Here `#aObject` is the defined object handle and `#aVariable` refers to an application-specific variable. Both names can be adapted if required.

The actual parameter names are already inserted into the statement if they are available. Otherwise, default parameter names such as `P0` and `P1` are used as placeholders. These names can be replaced by application-specific variables.



III

Data Browser

3 Data Browser

- Navigation 30
- File Selection List 30
- Field Selection 33
- Output Report Fields 34
- Filter Criteria 36
- Report Options 37
- Summary 39
- Field Properties 40
- Results Window 45

The data browser is used to generate data reports from Natural DDMs (data definition module) created from an Adabas, XML or SQL database available in your current Natural environment. You can select the DDM fields to be included in the report and specify filter criteria.

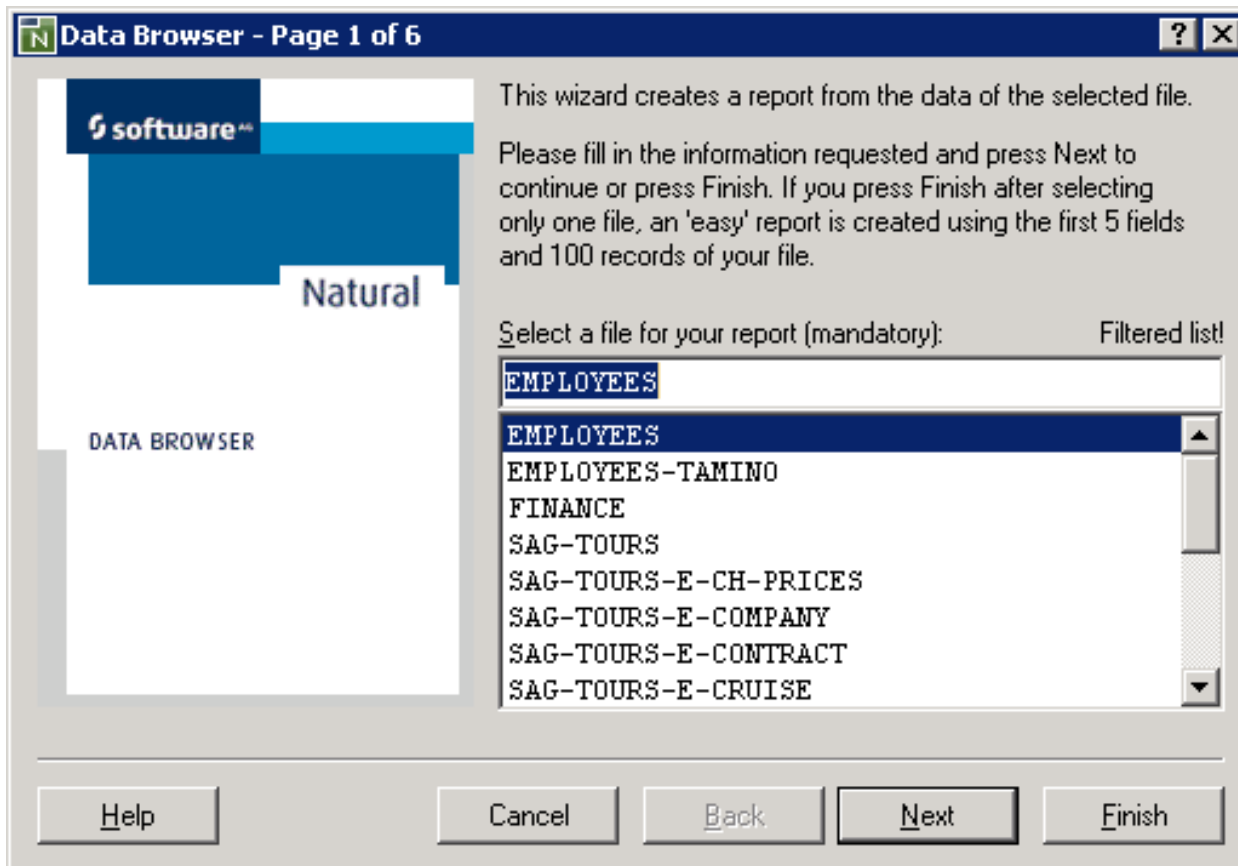
The data reports generated can be displayed in either the **Results** window of Natural Studio or in a text file downloaded to the PC. From the **Results** window, you can select records for printing or for saving as text files. You can generate one or more reports, where each report comprises a separate tabbed page of the **Results** window.

Navigation

Help	Activates the Natural Help system and leads to this chapter.
Cancel	Closes the dialog box without making any changes.
Back	Displays the previous page in the data browser.
Next	Displays the next page in the data browser.
Finish	Ends the report definition, generates the database request and displays the report layout in the Results window.

File Selection List

The file selection list displays all cataloged DDMs created from an Adabas file, an SQL table or an XML doctype which are available in the current Natural environment:



In a local Windows environment or in a remote Windows, UNIX or OpenVMS environment, the DDMs are contained in the current Natural library and concatenated steplib (if defined) in the current FNAT or FUSER system file.

If the FDDM option is set in a local Windows environment or in a remote Windows, UNIX or OpenVMS environment, the DDMs are contained in the FDDM system file.

In a remote mainframe environment, the DDMs are contained in the current FDIC system file.

This section provides information on the following:

- [Natural Studio Filter](#)
- [Context Menu](#)

- [DDM Selection](#)

Natural Studio Filter

A filter that has been defined and activated with the corresponding Natural Studio function (see also *Filtering Libraries and Objects* in the *Natural Studio* documentation), by default, also applies to the data browser. Therefore, if a filter is used for the current library, concatenated steplib or the system file, the selection list contains a reduced number of DDMs.

Exceptions

The data browser ignores a filter that contains any of the following definitions:

- In a remote Windows, mainframe, UNIX or OpenVMS environment, a filter that contains a name range (-).
- In a local Windows environment, a filter that contains a name range (-) combined with a wildcard (* or ?).

An active filter is indicated by the message `Filtered list!`, which appears above the selection list as shown in the example screen above. The data browser ignores an active filter if you select **Show Unfiltered List** from the context menu. If selected, the message `Unfiltered list!` appears above the selection list and the selection list contains all DDMs available in the current Natural environment.

Context Menu

The context menu for the file selection list on **Page 1** provides the following functions:

Show Unfiltered List	Deactivates or activates the Natural Studio filter as described above.
List DDMs with Database Types	Indicates the type of database (Adabas, XML or SQL) from which the DDMs were created.
List DDMs with Database IDs and File Numbers	Indicates the database IDs (DBID) and file numbers (FNR) where the DDMs are stored.

DDM Selection

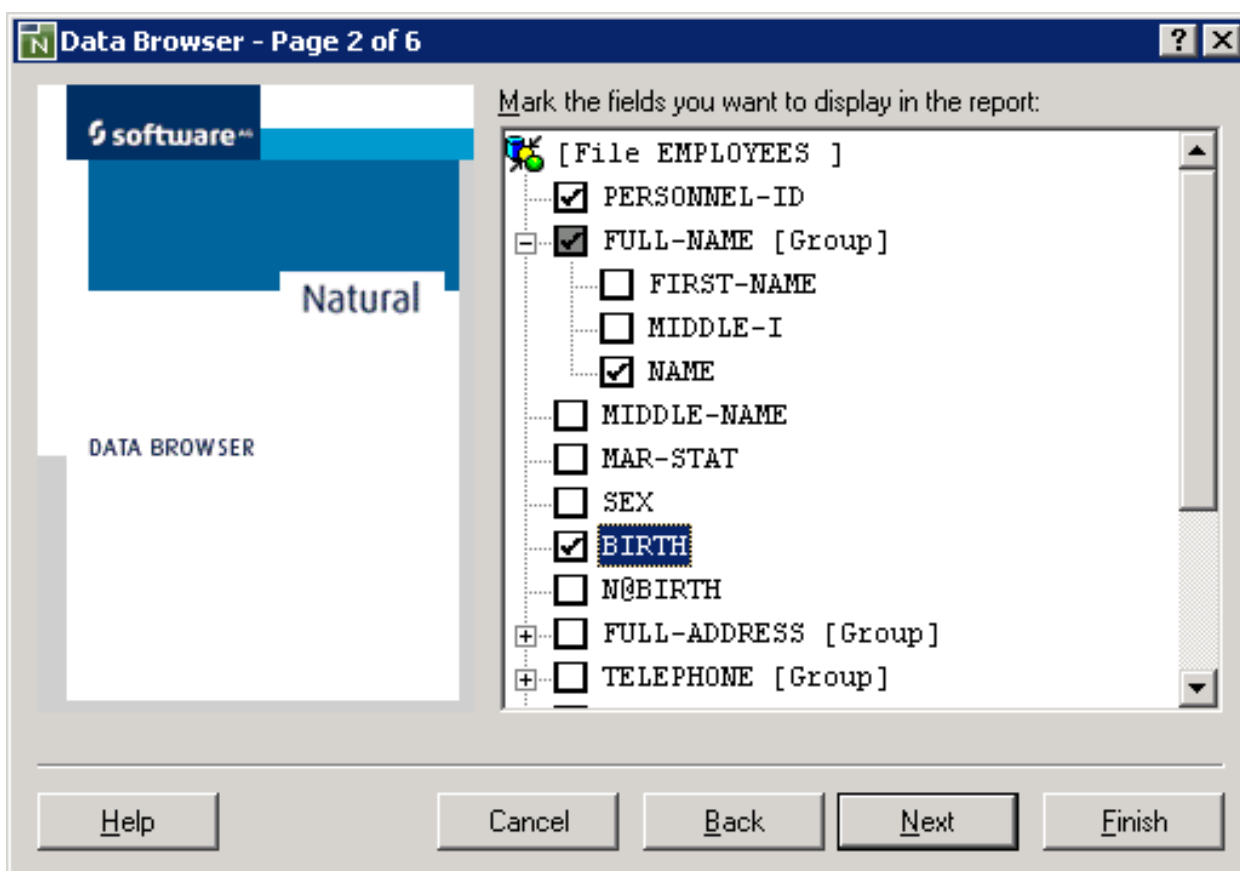
The selection of a DDM is mandatory. After selecting a DDM from the selection list, a check is performed whether the appropriate database is available.

If this is not the case, the DDM check will report the reason for the error, usually a database response code. Based on this information, the user can take measures to correct the error and restart the database.

After DDM selection, you can choose **Next** to go to the next data browser page where you can specify field selection criteria, or you can choose **Finish** to immediately generate the data report. If you choose **Finish**, the data browser creates an “easy” report that outputs the first 100 records of the selected DDM with the first 5 fields of the record. As soon as you specify any field selection or filter criteria on one of the following pages of the data browser, the “easy” report is replaced by an individual report.

Field Selection

All fields of the selected DDM are displayed in a tree view similar to the example shown below:



Groups are collapsed and can be expanded by clicking on them. Every item in the tree view has a check box. The fields that are selected here are taken over into the list of output fields. If groups are selected, all individual fields and subgroups that belong to them are automatically selected too. If in a group, because of manual intervention, not all fields are selected, the check box with the check mark is grayed out for the corresponding group name (in the example above, FULL-NAME). A cleared check box causes the field or fields of a group to be removed from the output list.

This section provides information on the following:

- Context Menu

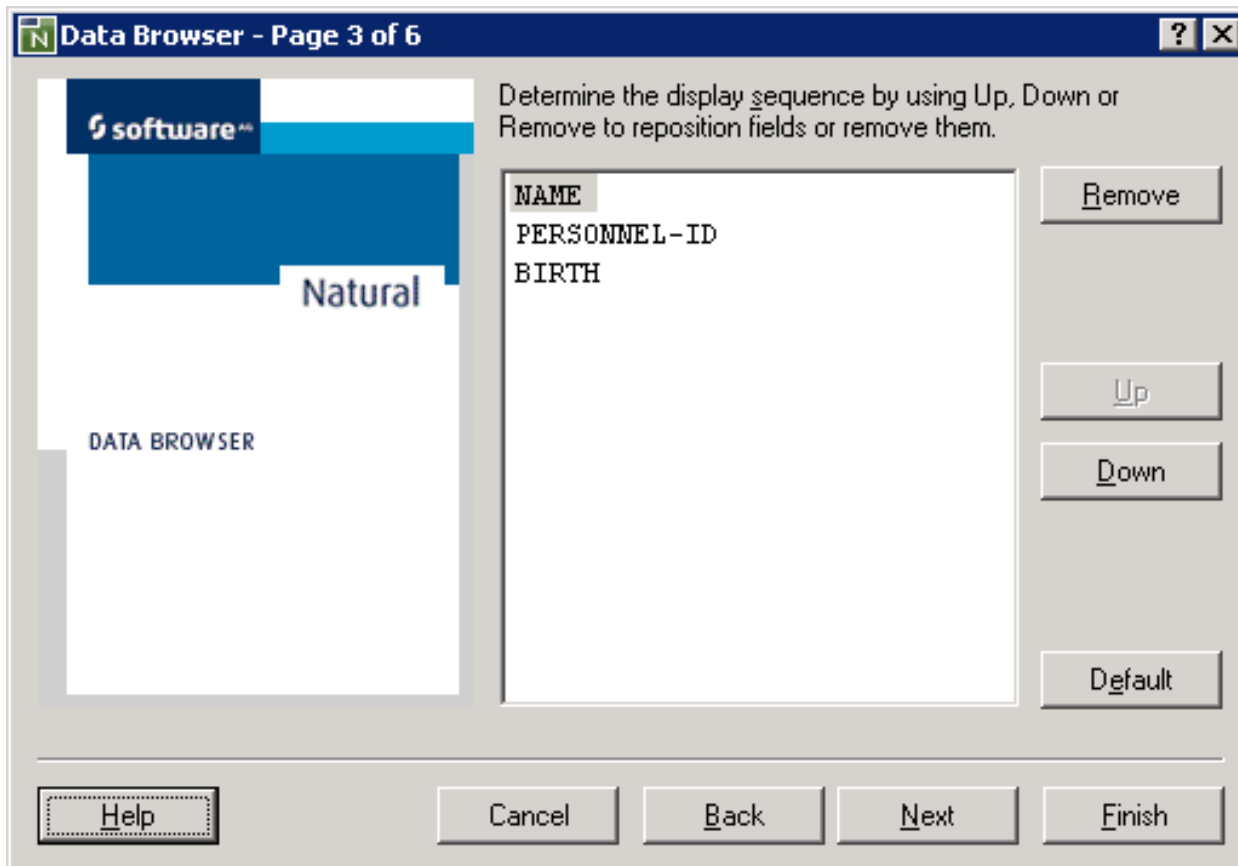
Context Menu

The context menu of the tree view on **Page 2** provides the following functions:

Expand All	Expands all group fields of the DDM.
Collapse All	Collapses all group fields of the DDM.
Select All	Selects all fields of the DDM.
Deselect All	Deselects all fields of the DDM.
Properties	Opens a pop-up window containing properties of the selected field. (For more information, refer to Field Properties). Or, if the DDM name is selected, the file properties show the DBID (database ID), the file number and the database type (Adabas, SQL or XML).

Output Report Fields

All individual fields that were selected in the tree view are listed on **Page 3** as shown in the following example:



For a field that has been defined as an array, the list indicates the dimension(s) of this array.

The sequence in which the fields appear in the list corresponds to the sequence in the report (from left to right).

The buttons **Up** and **Down** are provided for modifying the output sequence. They move the corresponding selected field one position up or down.

Using **Remove**, the selected field can be deleted from the report list.

With the **Default** button, the output sequence corresponding to the DDM structure is recreated.

If no fields were selected in the tree view, this page is skipped and per default the first 5 fields of the DDM are used for the report output.

This section provides information on the following:

- Context Menu

Context Menu

The context menu of the output sequence on **Page 3** provides the following functions:

Position Top	Positions the selected field at the top of the field list.
Position Bottom	Positions the selected field at the bottom of the field list.
Properties	Opens a pop-up window containing properties of the selected field. (For more information, refer to Field Properties).

Filter Criteria

Page 4 of the data browser provides the option to specify filter criteria similar to the example shown below:

The screenshot shows a dialog box titled "Data Browser - Page 4 of 6". On the left, there is a preview of a report page with a blue header containing "software" and "Natural", and the text "DATA BROWSER" below it. On the right, there is a text area with the instruction: "You can limit the number of data records by specifying filter criteria." Below this, there are three input fields: a dropdown menu labeled "Select a filter criterion:" with "NAME" selected; a text box labeled "Enter a start value in format A20:" containing the letter "P"; and another text box labeled "Enter an end value in format A20:" containing the letter "R". At the bottom right, there is a field labeled "Filter by number of records:" with the value "10" and the text "(0 is 'No Limit')". At the bottom of the dialog, there are five buttons: "Help", "Cancel", "Back", "Next", and "Finish".


A report can be generated unfiltered in that the records are read as they stand physically in the file (physical read). This is the default.

The range of values of the descriptor field can be defined by a start value and, additionally, by an end value. An end value can be applied only if a start value exists and the start value must be less than the end value. The appropriate value format is shown on the label of the start/end value box.

For a date field, you can enter a start/end date in the Natural format **D** (see the *Programming Guide*) according to the example date indicated on the box label where **YYYY** represents the year, **MM** the month and **DD** the day. Note that the format in which the date has to be entered also depends on the setting of the profile parameter **DTFORM** (see the *Parameter Reference* documentation).

For a time field, you can enter a start/end time in the Natural format **T** or in the extended format **T** according to the example date/time indicated on the box label where **HH:II:SS** represents the time (**HH** = hour, **II** = minutes, **SS** = seconds), **YYYY** the year, **MM** the month and **DD** the day. Note that the format in which the date has to be entered also depends on the setting of the profile parameter **DTFORM** (see the *Parameter Reference* documentation). In addition, consider the format used when storing the fields in your database system.

All filter criteria can be further limited by entering an absolute record limit (the default setting is 100 records). Independent of any filter criteria, only the corresponding number of records is written in the result list (report). By entering 0 (zero), the record limit is switched off.

 **Caution:** A large number of records can result in a long response time for data retrieval from the database. Therefore, if you switch the record limit off (value set to 0) or if you specify a value greater than 1000, a message will warn you of a possible delay in generating the result list.

This section provides information on the following:

- [Context Menu](#)

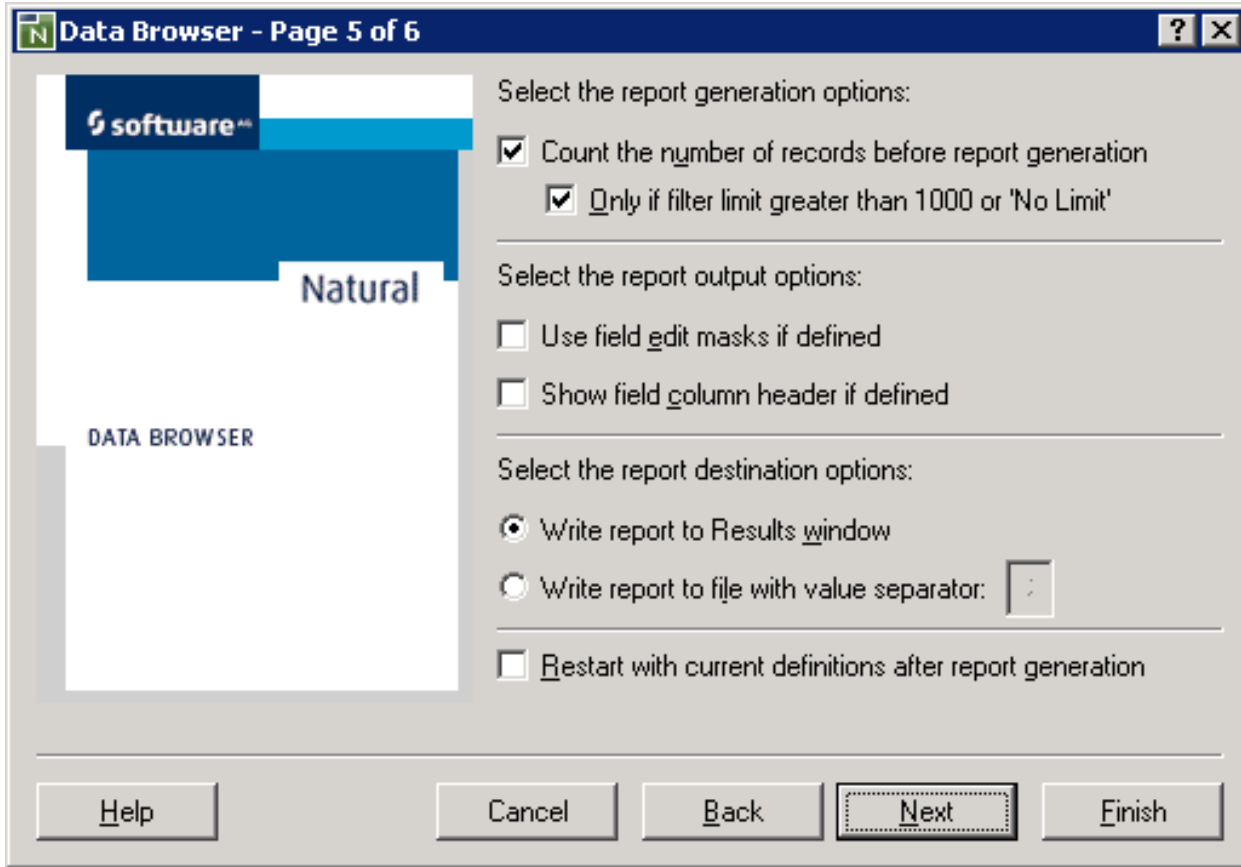
Context Menu

The context menu of the filter criteria on **Page 4** provides the following function:

Properties	Opens a pop-up window containing properties of the selected field. (For more information, refer to Field Properties).
-------------------	--

Report Options

The report options of the data browser are provided on **Page 5**:

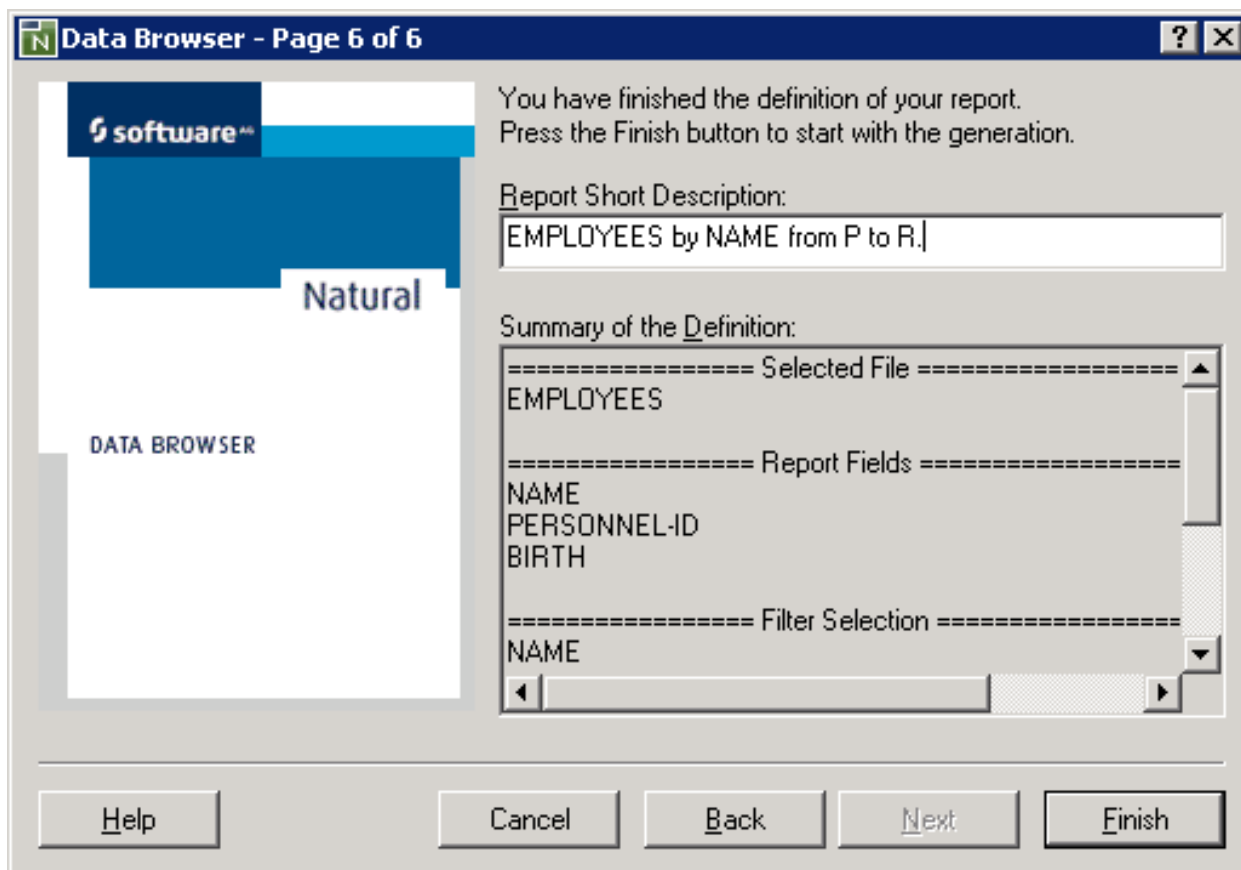


Count the number of records before report generation	<p>Displays the number of records that meet the selection and filter criteria specified on the previous pages of the data browser. The number is displayed before the report is generated.</p> <p>This option is selected by default.</p>
Only if filter limit greater than 1000 or 'No Limit'	<p>Displays the number of records only if a number greater than 1000 is entered as filter limit or if no filter limit is set on Page 4 (see <i>Filter Criteria</i>).</p> <p>This option is selected by default.</p>
Use field edit masks if defined	<p>If defined for the selected fields, uses edit masks for displaying the field values.</p>
Show field column headers if defined	<p>If defined for the selected fields, displays the column headers instead of the field names.</p>
Write report to Results window	<p>Displays the report generated in the Results window.</p> <p>This option is selected by default.</p>

Write report to file with value separator	Writes the report generated to a text file, which can be used for spreadsheet manipulation. This file contains character-separated values that delimit rows and columns. You can change the value separator for columns by replacing the default value separator semicolon (;) with another special character. A blank character is not allowed.
Restart with current definitions after report generation	Restarts the data browser on Page 2 , with the definitions used for the previous report generation.

Summary

When you have finished specifying report generation options, all report definitions are summarized on the last page of the data browser:



In the **Report Short Description** text box, you can replace the default description (Report of *ddm-name*) by your own text of up to 253 characters as shown in the example above. The first 30 characters of this description are then shown as a tab in the **Results** window. The full text is dis-

played on the tabbed page **Description** of the report properties (see also *Properties of Report*). The full text is also contained in the header section of a printout (see also *Print*).



Note: When you choose **Finish**, a warning appears if more than 200 columns are to be created for the report, which can cause significant performance problems. A large number of columns can result from a large number of fields or a wide range of array occurrences specified for the report.

Field Properties

You can use the context menu to view the properties of each field that is listed on a page (Pages 2 to 4) or in the **Results** window.

The property sheet of each field always contains the tabbed pages **General** and **Details**. An additional page is provided for the following fields:

- For fields with header or edit-mask entries, an **Extended** page with layout information.
- For fields with array definitions, an **Array Range** page with the option to specify a range of occurrences.

The tabbed page **General** shows the following:

Properties of field LANG

General | Details | Extended | Array Range

Name: LANG

Format: A Alphanumeric

Length: 3

Type: M Multiple-value field

Level: 1

Descriptor: D Descriptor

Remark:

File Name: EMPLOYEES

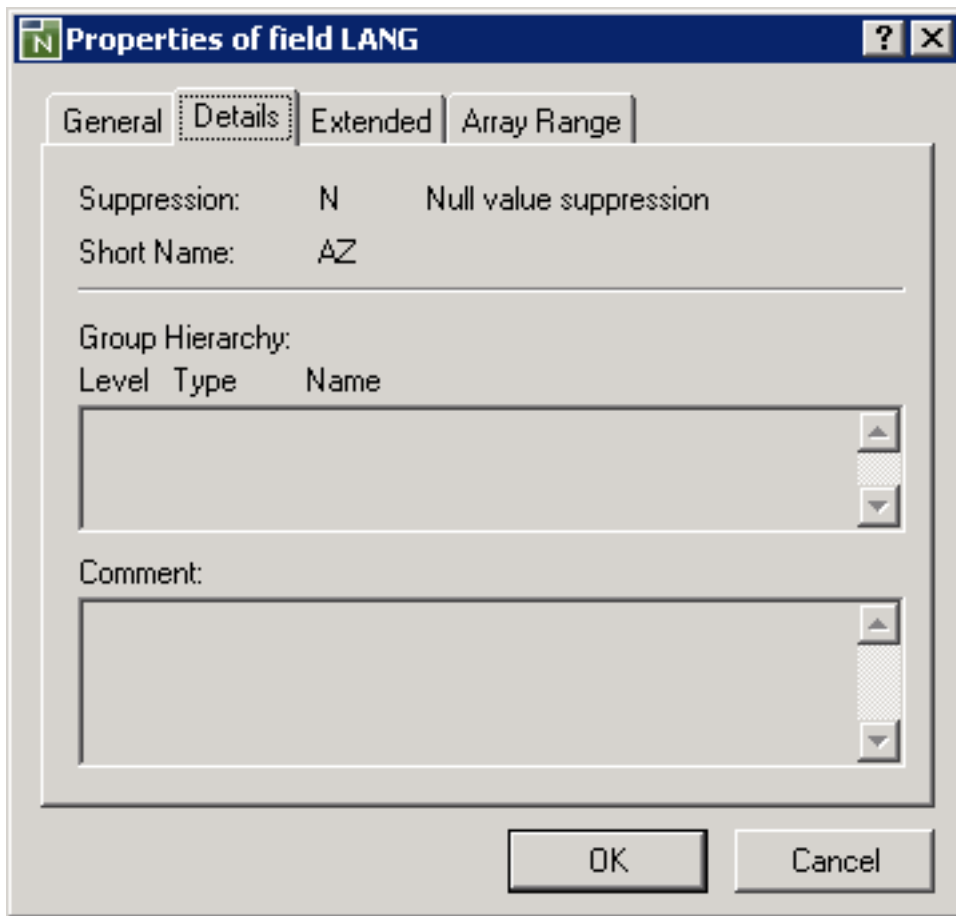
File Type: ADABAS

OK Cancel

Name	Name as of the DDM.	
Format	Format as of the DDM.	
Length	Length as of the DDM.	
Type	Field type: group, periodic group, multiple-value field or elementary field.	
Level	Level as of the DDM.	
Descriptor	Descriptor type:	
	<i>blank</i>	No descriptor.
	D	Descriptor
	S	Subdescriptor or superdescriptor (not applicable to an XML database)
	H	Hyperdescriptor (not applicable to an XML database)

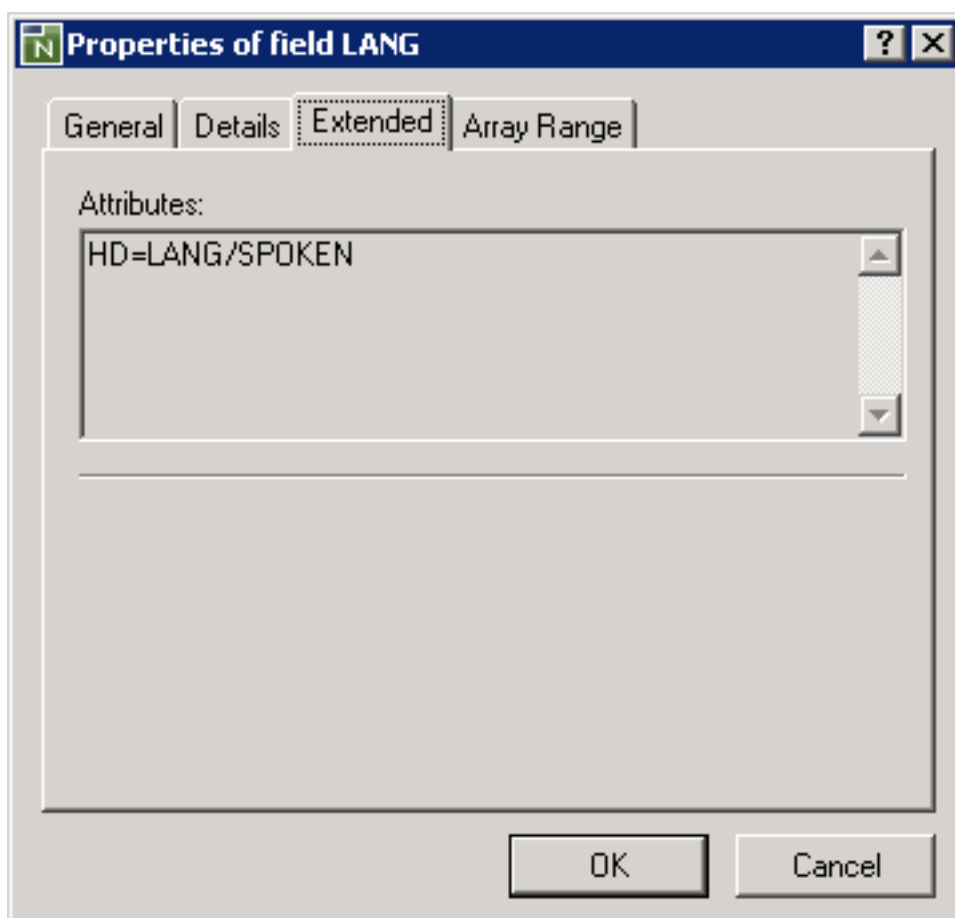
	N	Non-descriptor (not applicable to an XML database)
	See further information, see also <i>Columns of Field Attributes</i> in the <i>DDM Editor</i> documentation.	
Remark	Remark as of the DDM.	
File Name	File name as of the DDM.	
File Type	Database type.	

The tabbed page **Details** shows the following:



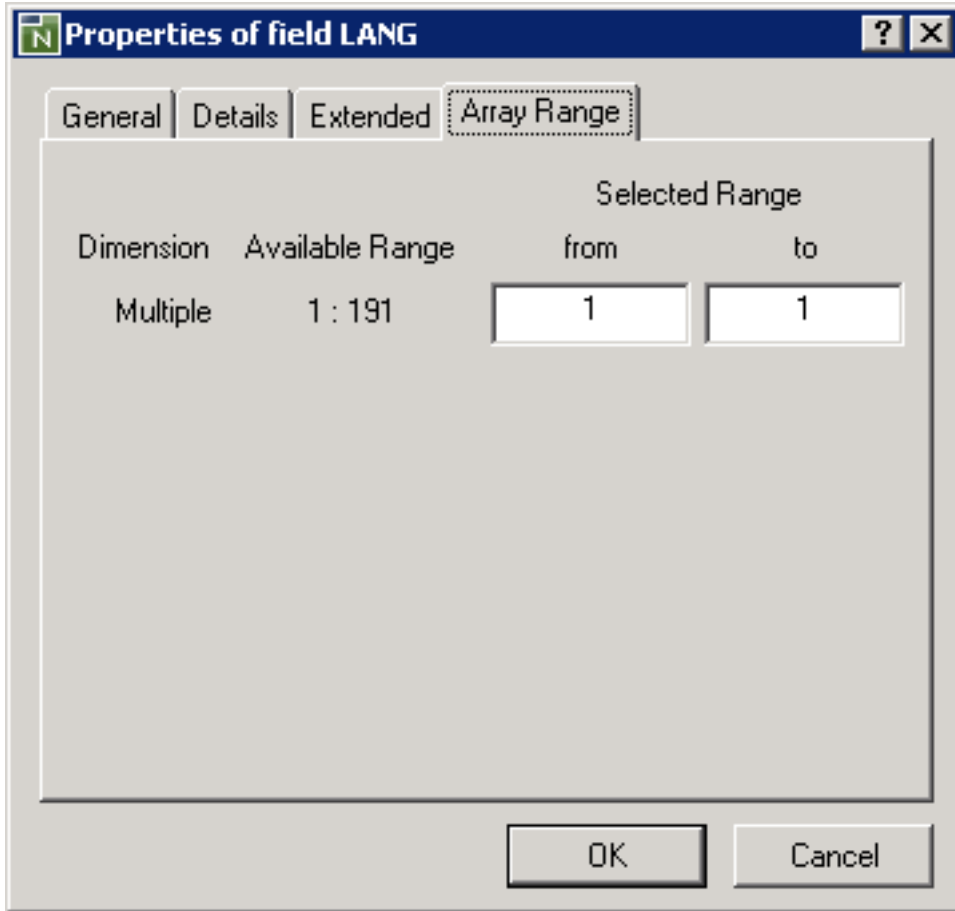
Suppression	Null value suppression, fixed storage, not null.
Short Name	Adabas short name of the field.
Group Hierarchy	All groups and group levels to which a field belongs.
Comment	Comment as of the DDM.

The (optional) tabbed page **Extended** shows the following:



Attributes	Edit mask (EM=) or header (HD=) as of the DDM.
-------------------	--

The (optional) tabbed page **Array Range** shows the following:

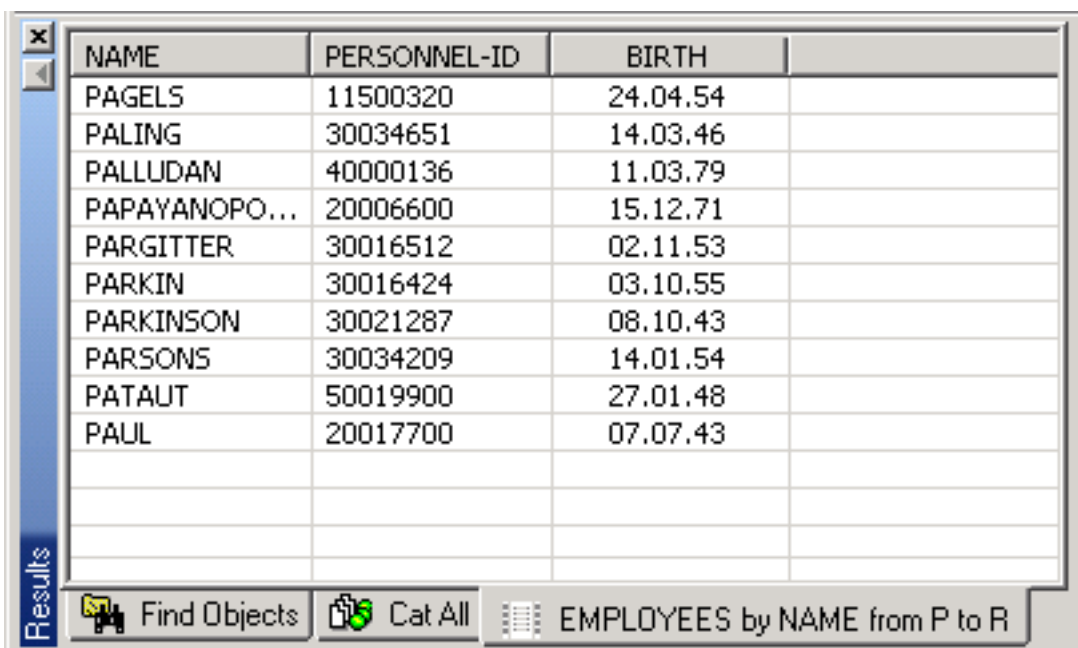


Dimension	Array dimension(s) of the field:	
	Periodic and/or Multiple	For a periodic group or a multiple-value field from an Adabas database.
	1	For the dimension of a field from an SQL database.
	1, 2, 3	For one, two or three dimensions of a field from an XML database.
Available Range	Shows the range of valid values that can be entered in the from and to fields:	
	1 - 191	A number range from 1 to 191 for a field created from an Adabas or SQL database.
	1 - <i>n</i>	A number range from 1 to <i>n</i> for a field created from an XML database where <i>n</i> denotes the maximum number of occurrences as defined for this field in the DDM.

Selected Range	<p>For each dimension of an array, you can specify the range of occurrences to be displayed in the report:</p> <p>In the from field, you enter the first occurrence to be displayed and in the to you can enter the last occurrence. The value entered in from must be greater than the value entered in to. See also Available Range below.</p> <p>The default setting for both fields is 1 for the first occurrence of each dimension.</p> <p>The selected range(s) of occurrences are indicated on Page 3 (see Output Report Fields) and Page 6 (see Summary).</p> <p>In the report, each occurrence is listed in a separate column, in a left-to-right order, from the first occurrence of the first dimension to the last occurrence of the last dimension.</p> <p>Caution: A wide range of occurrences can result in a large number of columns, which can cause significant performance problems.</p>
-----------------------	---

Results Window

The result list(s) of the data browser are displayed in the **Results** window of Natural Studio similar to the example shown below:



The screenshot shows a window titled 'Results' containing a table with the following data:

NAME	PERSONNEL-ID	BIRTH	
PAGELS	11500320	24.04.54	
PALING	30034651	14.03.46	
PALLUDAN	40000136	11.03.79	
PAPAYANOPO...	20006600	15.12.71	
PARGITTER	30016512	02.11.53	
PARKIN	30016424	03.10.55	
PARKINSON	30021287	08.10.43	
PARSONS	30034209	14.01.54	
PATAUT	50019900	27.01.48	
PAUL	20017700	07.07.43	

At the bottom of the window, there are three buttons: 'Find Objects', 'Cat All', and a report preview button showing 'EMPLOYEES by NAME from P to R'.

You can have as many reports as you want in parallel. With a click on the column header, the lines are sorted by the column value ascending or descending. Modification of the column sequence is possible by shifting the column headers. One or more lines of the result list can be selected for further processing by using the context menu.



Note: A field header definition that spans multiple lines in a `DISPLAY` statement, is displayed in a single line, where each line break is denoted by a slash (/).

This section provides information on the following:

- [Context Menu](#)
- [Properties of Report](#)

Context Menu

The context menu of the results list provides the following functions:

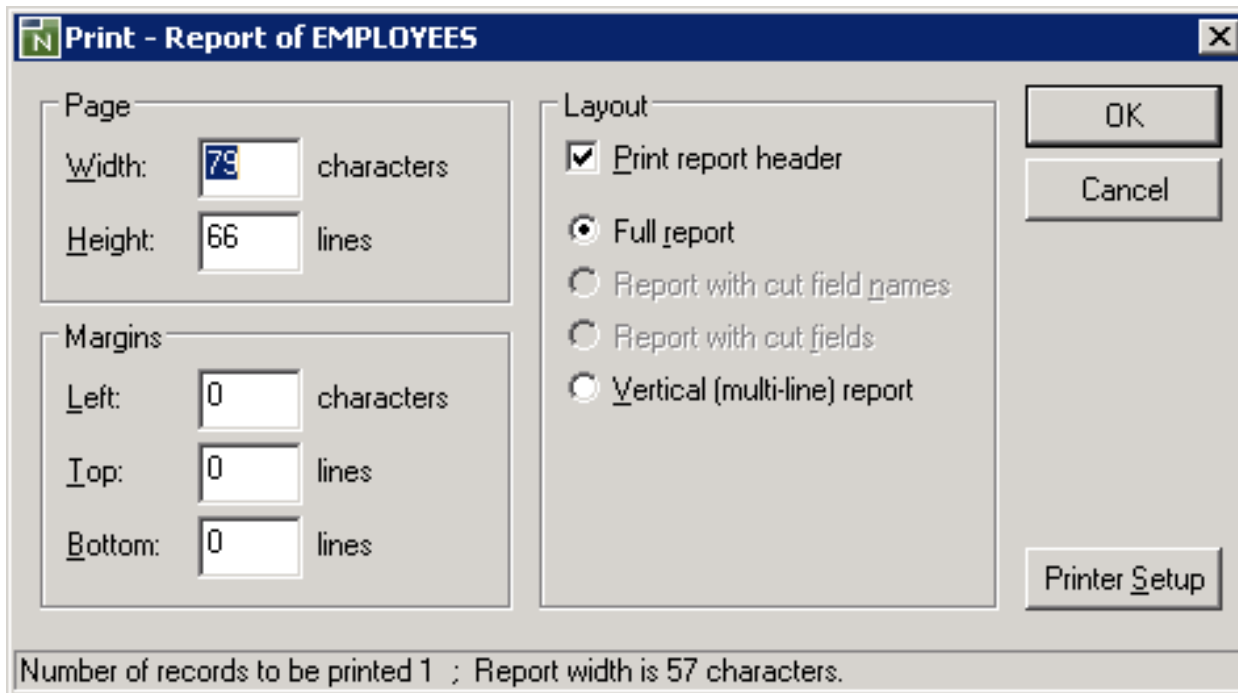
Select All	All lines of the result list are selected.	
Expand All Columns	Expands all columns in the result list.	
Collapse All Columns	Collapses all columns in the result list.	
Output	Print Selection	Prints the selected lines of the result list. For more information, refer to Print .
	Export Selection to Textfile	Writes the selected lines of the result list in a text file (.txt). For more information, refer to Save Report Data .
Delete Tab	Deletes the current tab.	
Delete Tab and Hide	Deletes the current tab and closes the Results window.	
Properties	Additional information for the current report is displayed. (For more information, refer to Properties of Report).	

This section provides information on the following:

- [Print](#)
- [Save Report Data](#)

Print

The **Print Selection** function invokes a window similar to the example below:



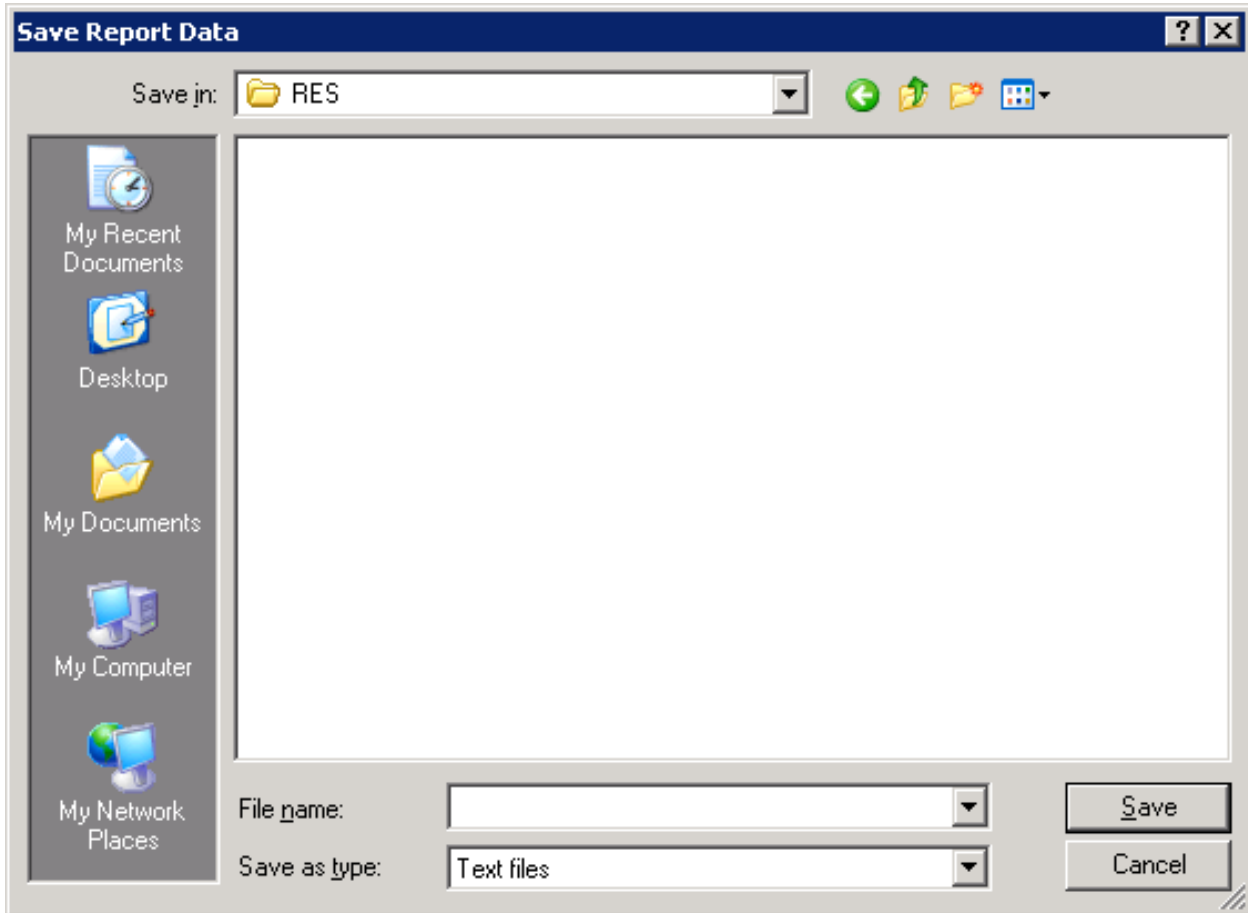
The **Print** window provides the following options:

Page	Define the report width and height as matching to printer page size.
Margins	You can change the margin settings to adjust the distance between the report data and the left, top or bottom edge of the printed page.
Layout	<p>For some criteria you can define the printed report layout.</p> <p>A report header can be specified which contains the report creation date, the selection criteria, the number of records printed, and the full text of the report description as created by the data browser wizard on Page 6.</p> <p>A record format can be selected, as far as depending on the report size offered, to see a full report list, report list with cut header or fields or a report list with field headers and values in vertical sequence.</p>

The status line informs about the selected number of records and the original width of the report in characters. With the **Printer Setup** button, you can open the PC printer setup dialog to select one of the printers available in your system.

Save Report Data

The **Export Selection to Textile** function invokes a **Save Report Data** window similar to the example below:



The standard save dialog enables you to enter a file name completed with the file type `.txt`.

The text file contains the selected lines of the result list with the delimiters: the input delimiter character as specified with the `ID` session parameter (default is a comma) separates field values and carriage return/line feed separates records.

Properties of Report

The **Properties** function invokes a property sheet similar to the example below:

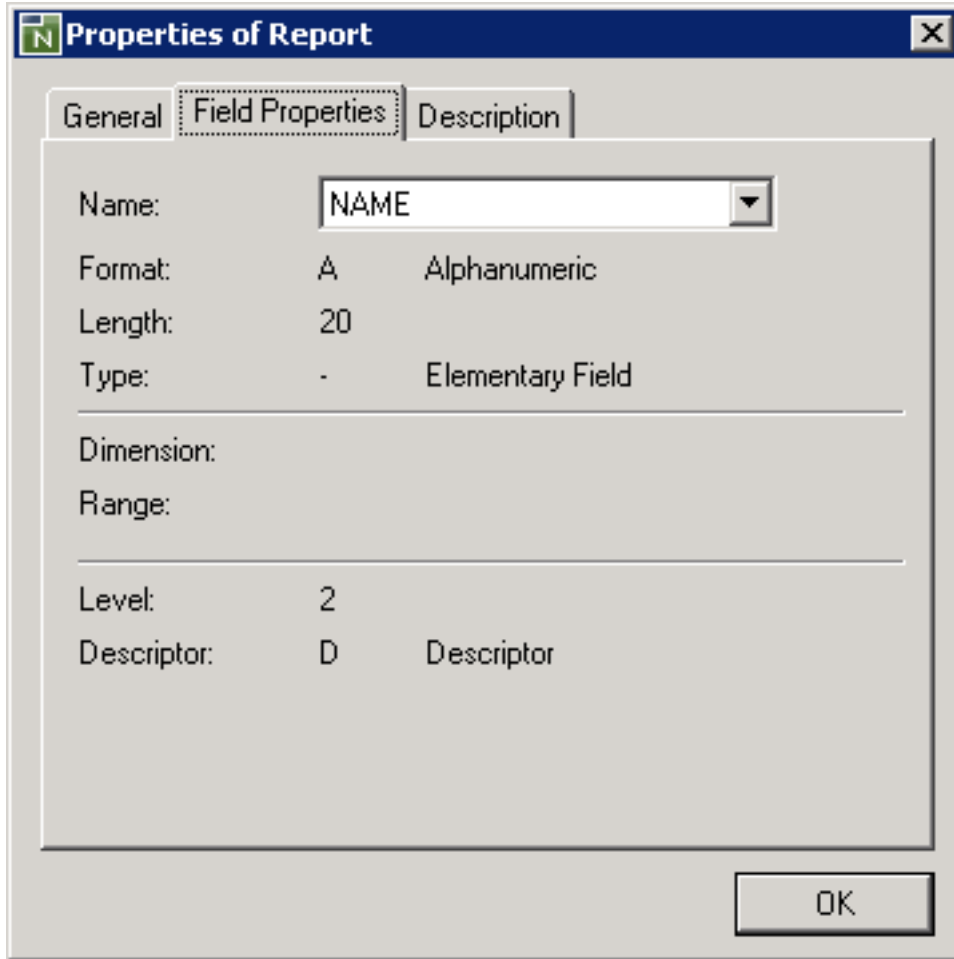
The screenshot shows a dialog box titled "Properties of Report" with three tabs: "General", "Field Properties", and "Description". The "General" tab is active. The fields and their values are as follows:

- File: EMPLOYEES
- DBID/FNR/Type: 0 / 11 / ADABAS
- Selection criteria: NAME
- Start value: P
- End value: R
- Record limit: 10
- Result records: 10
- Creation date: 26.10.07, 14:26:55
- Environment: Local
- Library: SAGTEST

An "OK" button is located at the bottom right of the dialog.

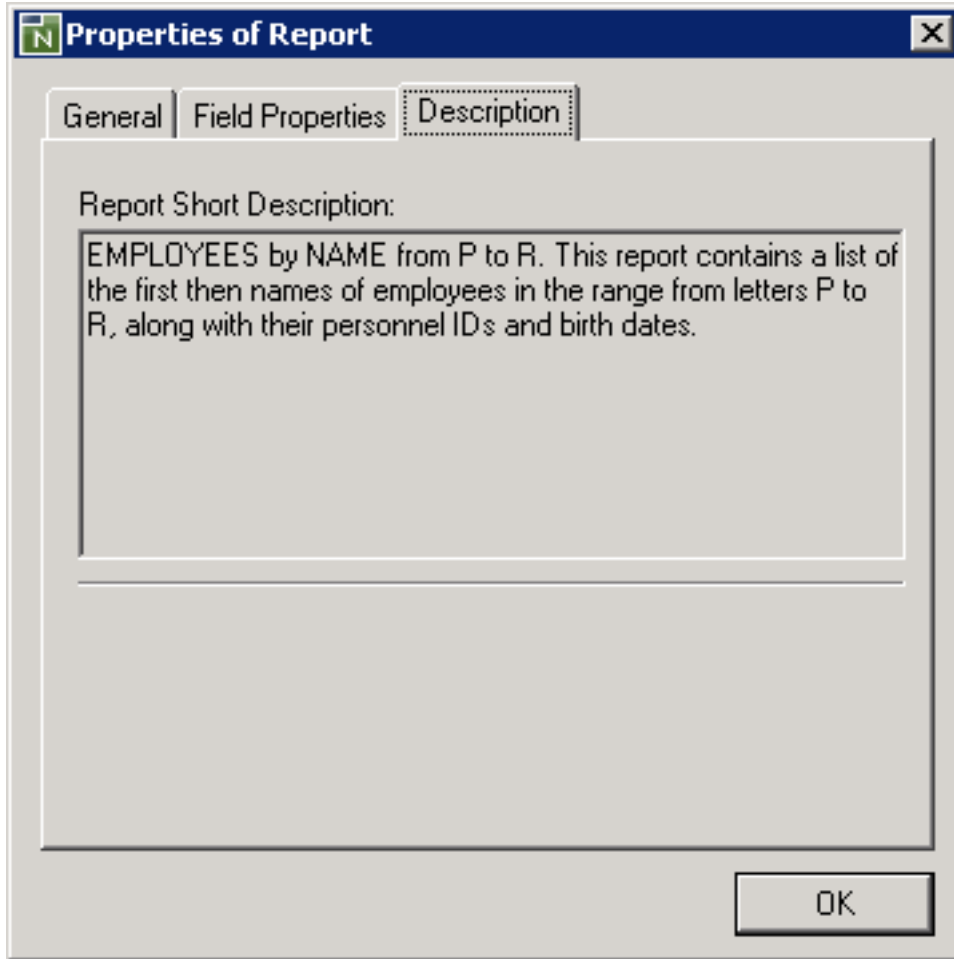
The tabbed page **General** shows the following:

File	DDM name.
DID/FAR/Type	Database ID, file number and database type.
Selection criteria	Selection criteria as entered in the data browser.
Start value	Start value as entered in the data browser.
End value	End value as entered in the data browser.
Record limit	Record limit as entered in the data browser.
Result records	Number of records shown in the report.
Creation date	Date and time when the report was created.
Environment	Name or alias name of the environment where the data were retrieved.
Library	The library from which the data browser was started to create the current report.



The tabbed page **Field Properties** shows the following:

Name	Enables a field name selection to see its properties.
Format	Format as of the DDM.
Length	Length as of the DDM.
Type	Group, periodic group, multiple-value field or elementary field.
Dimension	Shows <i>Periodic</i> for a periodic group or <i>Multiple</i> for a multiple-value field (Adabas), or one or more numbers of array dimensions (1 for SQL or 1 to 3 for XML).
Range	Shows the range of array occurrences specified for a field.
Level	Level as of the DDM.
Descriptor	Descriptor, subdescriptor/superdescriptor, hyperdescriptor or non-descriptor.



The tabbed page **Description** shows the full text of the report description as created by the data browser wizard on **Page 6**.

IV

F TOUCH Utility

4 FTOUCH Utility

- Using the Utility FTOUCH 56
- Syntax of ftouch 57
- Examples of ftouch 60

The FTOUCH utility is used to make a downloaded object executable by Natural. This is done by importing the object into the Natural system file FNAT or FUSER and updating the *FILEDIR.SAG* file.

Related Topics:

- *The File FILEDIR.SAG - Operations* documentation
- *Using NFS to Store Natural Libraries - Operations* documentation
- *Transferring Natural Generated Programs - Programming Guide*

Using the Utility FTOUCH

This section provides instructions for executing the FTOUCH utility.



Note: Terms enclosed in brackets ([]) are optional; bold letters are actual values that must be entered as shown.

▶ To execute the FTOUCH utility

- 1 Go to an operating system command prompt.
- 2 Ensure that the transferred file is in the desired FNAT or FUSER directory (as specified in your global configuration file) and has the correct extension.
- 3 Enter the command `ftouch` using the following syntax:

```
ftouch [fnat=dbid,fnr] [fuser=dbid,fnr][bp=bp-name]
[parm=parm-file] [lib=library-name] [encoding=encoding-name]
[userrep=rep-use] [-ignoreext][-v] [-q] [mode] [kind] files
```

Or:

For migration, use the following syntax:

```
ftouch [fnat=dbid,fnr] [fuser=dbid,fnr]
[parm=parm-file] [lib=library-name][encoding=encoding-name][-q] convert
```

Or:

For endian conversion of the *FILEDIR.SAG* file, use the following syntax:

```
ftouch [fnat=dbid,fnr] [fuser=dbid,fnr]
[parm=parm-file] [lib=library-name][endian=endian-mode]
```

Or:

For encoding of single or multiple objects contained in the *FILEDIR.SAG* file, use the following syntax:

```
ftouch [fnat=dbid,fnr] [fuser=dbid,fnr]
[parm=parm-file] [lib=library-name][objname=object-name][encoding=encoding-name]
```

Or:

For setting the line number suppression state of a library in *FILEDIR.SAG*, use the following syntax:

```
ftouch [fnat=dbid,fnr] [fuser=dbid,fnr]
[parm=parm-file] [lib=library-name][supr]n=library-state]
```

Syntax of ftouch

The following options are provided with the `ftouch` command:

Option	Explanation
<code>fnat=dbid,fnr</code>	Specifies the database ID and file number of the FNAT system file to be used; default is the value specified in the NATPARM parameter file. See also Example 2 .
<code>fuser=dbid,fnr</code>	Specifies the database ID and file number of the FUSER system file to be used; default is the value specified in the NATPARM parameter file. See also Example 2 .
<code>bp=bp-name</code>	Specifies the buffer pool to be used. You can omit the <code>bp-name</code> if you want to use the Natural default buffer pool NATBP; otherwise, you have to specify the appropriate <code>bp-name</code> . Note: 1. If the Natural default buffer pool is not active or if the specified buffer pool does not exist, an appropriate error message is displayed. 2. Do not delete the default buffer pool NATBP, as it is possible that Natural may no longer function properly.

Option	Explanation				
parm= <i>parm-name</i>	Specifies the name of the parameter file to be used if you want to use a parameter file other than the default NATPARM parameter file.				
lib= <i>library-name</i>	Specifies the library to be used. You can omit the <i>library-name</i> if you are already in the appropriate subdirectory; otherwise you have to specify the appropriate <i>library-name</i> .				
userep= <i>rep-use</i>	<p>Specifies whether to use the repository or not. <i>rep-use</i> must be one of the following:</p> <table border="1" data-bbox="490 562 1385 705"> <tr> <td data-bbox="490 562 870 611">ON</td> <td data-bbox="870 562 1385 611">The repository is used.</td> </tr> <tr> <td data-bbox="490 653 870 705">OFF</td> <td data-bbox="870 653 1385 705">The repository is not used.</td> </tr> </table>	ON	The repository is used.	OFF	The repository is not used.
ON	The repository is used.				
OFF	The repository is not used.				
-v	Displays statistics on disk I/Os during processing.				
-q	Indicates that quiet mode is to be used: only error messages but no status messages are displayed.				
-ignoreext	<p>Specifies that files with unknown extensions contained in a library are ignored. The <code>-ignoreext</code> option can be combined with one of the following options:</p> <ul style="list-style-type: none"> - a - d 				
<i>mode</i>	<p>Specifies the programming mode; <code>sm</code> specifies that a program is in structured mode; the default is reporting mode.</p> <p>See also Example 1.</p>				
<i>kind</i>	<p>Specifies the subdirectories SRC and/or GP for input; it can be one of the following:</p> <ul style="list-style-type: none"> - s for source objects (default), - g for cataloged objects/generated programs, - b for both source objects and cataloged objects/generated programs. <p>See also Example 2.</p>				
<i>files</i>	<p>Specifies the files to be processed; you can specify <i>filename.ext</i> for individual files or:</p> <ul style="list-style-type: none"> - a to add new files; all files in the directory which are currently found in <i>FILEDIR.SAG</i> are added (already existing files are not touched). - d to build a new <i>FILEDIR.SAG</i> directory. <p>Caution: Be careful when using this option, since the old <i>FILEDIR.SAG</i> is deleted and rebuilt from scratch.</p> <p>See also Example 4.</p>				

Option	Explanation
-f	Forces an update of the specified object's timestamp in <i>FILEDIR.SAG</i> . This option can only be specified if an individual file has been specified with the <i>files</i> option (see above).
convert	<p>Indicates that an old <i>FILEDIR.SAG</i> file is to be migrated. The <i>FILEDIR.SAG</i> file from a Natural version earlier than Version 6.2 is converted into a new portable <i>FILEDIR.SAG</i> file. A copy of the original (old) <i>FILEDIR.SAG</i> file is saved as <i>FILEDIR.BCK</i> file in the directory of the specified library. If a <i>FILEDIR.BCK</i> file already exists in the specified library, the old <i>FILEDIR.SAG</i> will <i>not</i> be converted.</p> <p>For further information, see <i>Portable Natural System Files</i> in the <i>Operations</i> documentation.</p> <p>See also Example 3 and Example 5.</p>
sync	<p>Indicates that the specified library and system files are to be synchronized between Natural and the repository (Windows only); this function must be executed each time <i>FILEDIR.SAG</i> is modified by FTOUCH.</p> <p>Caution: When specifying <i>sync</i>, ensure that either <i>userrep=ON</i> is set or the Natural profile parameter <i>USEREP</i> is set to <i>ON</i>.</p>
encoding= <i>encoding-name</i>	<p>Specifies the code page to be used for the files contained in <i>FILEDIR.SAG</i>.</p> <p>The <i>encoding</i> option generates or changes the internal code page information maintained in <i>FILEDIR.SAG</i> for each object affected by the <i>ftouch</i> command. This option does <i>not</i> convert the contents of a source object or a cataloged object/generated program.</p> <p>The <i>encoding</i> option can be combined with the following options:</p> <ul style="list-style-type: none"> -a -d convert objname <p><i>encoding-name</i> can be any code page name valid with the CP session parameter specified in the <i>NATPARM</i> parameter file. See also <i>CP - Default Code Page Name</i> in the <i>Parameter Reference</i>.</p> <p>See also Example 4, Example 5, Example 7 and Example 8.</p>
endian= <i>endian-mode</i>	<p>Specifies the endian format to be used for the <i>FILEDIR.SAG</i> directory.</p> <p>The <i>endian</i> option applies to the entire <i>FILEDIR.SAG</i> directory.</p> <p>The option does not apply when adding files to <i>FILEDIR.SAG</i> or when generating a new <i>FILEDIR.SAG</i>.</p> <p><i>endian-mode</i> can be one of the following formats:</p>

Option	Explanation				
	BIG Converts to big endian. LITTLE Converts to little endian. DEFAULT Converts to the endian format used on your current platform. See also Example 6 .				
objname= <i>object-name</i>	Selects the object(s) for which to maintain internal format information in <i>FILEDIR.SAG</i> . The objname option only applies if the encoding option is specified. <i>object-name</i> selects all objects with names equal to the specified value. You can use asterisk (*) notation for a name range. See also Example 7 and Example 8 .				
suprln= <i>library-state</i>	Specifies whether the line number suppression state is set for the specified library. <i>library-state</i> must be one of the following: <table border="1" data-bbox="488 863 1377 1102"> <tbody> <tr> <td data-bbox="488 909 792 1020">ON</td> <td data-bbox="792 909 1377 1020">Source line numbers are not written to the files contained in <i>FILEDIR.SAG</i>, when saving the sources of the objects contained in this library.</td> </tr> <tr> <td data-bbox="488 1020 792 1102">OFF</td> <td data-bbox="792 1020 1377 1102">Source line numbers are written to the files contained in <i>FILEDIR.SAG</i>.</td> </tr> </tbody> </table>	ON	Source line numbers are not written to the files contained in <i>FILEDIR.SAG</i> , when saving the sources of the objects contained in this library.	OFF	Source line numbers are written to the files contained in <i>FILEDIR.SAG</i> .
ON	Source line numbers are not written to the files contained in <i>FILEDIR.SAG</i> , when saving the sources of the objects contained in this library.				
OFF	Source line numbers are written to the files contained in <i>FILEDIR.SAG</i> .				

Examples of ftouch

The following section provides examples of the ftouch command.

Example 1:

Change to the following directory: `fuser-directory/TESTLIB/SRC`

Enter the following command: `ftouch sm TESTFILE.NSP`

As a result, the program TESTFILE in library TESTLIB is available in structured mode to Natural.

Example 2:

Change to the following directory: *fuser-directory*/MYLIB

Enter the following command: `ftouch fnat=21,21 fuser=22,22 -b`

As a result, all files in the directories MYLIB/SRC and MYLIB/GP are available in reporting mode (default) to Natural.

Example 3:

Change to the following directory: *fuser-directory*

Enter the following command: `ftouch lib=MYLIB convert`

As a result, a new portable *FILEDIR.SAG* file is saved for the MYLIB library and the old *FILEDIR.SAG* is saved as *FILEDIR.BCK* file in this library.

Example 4:

Change to the following directory: *fuser-directory*

Enter the following command: `ftouch lib=MYLIB encoding=UTF-8 -a -s`

As a result, the internal format information is generated as UTF-8 for all objects which are added to the *FILEDIR.SAG* directory from the MYLIB/SRC subdirectory.

Example 5:

Change to the following directory: *fuser-directory*

Enter the following command: `ftouch lib=OLDLIB encoding=windows-1251 convert`

As a result, a new portable *FILEDIR.SAG* file is saved for the OLDLIB library and the internal format information changes to windows-1251 for all objects contained in the *FILEDIR.SAG* file.

Example 6:

Change to the following directory: *fuser-directory*

Enter the following command: `ftouch lib=MYLIB endian=BIG`

As a result, the *FILEDIR.SAG* file of the MYLIB library is converted to big endian. The internal format information changes to BIG for all objects contained in the MYLIB library.

Example 7:

Change to the following directory: *fuser-directory*

Enter the following command: `ftouch lib=MYLIB objname=MYPROG1 encoding=UTF-8`

As a result, the internal format information of object MYPROG1 changes to UTF-8 if MYPROG1 is contained in library MYLIB in the *FILEDIR.SAG* file.

Example 8:

Change to the following directory: *fuser-directory*

Enter the following command: `ftouch lib=MYLIB objname=MY* encoding=UTF-8`

As a result, the internal information of all objects with names that start with MY changes to UTF-8 if they are contained in library MYLIB in the *FILEDIR.SAG* file.

Example 9:

Change to the following directory: *fuser-directory*

Enter the following command: `ftouch lib=MYLIB suprln=ON`

As a result, the line number suppression state is set to ON for library MYLIB in the *FILEDIR.SAG* file.

V INPL Utility

5 INPL Utility

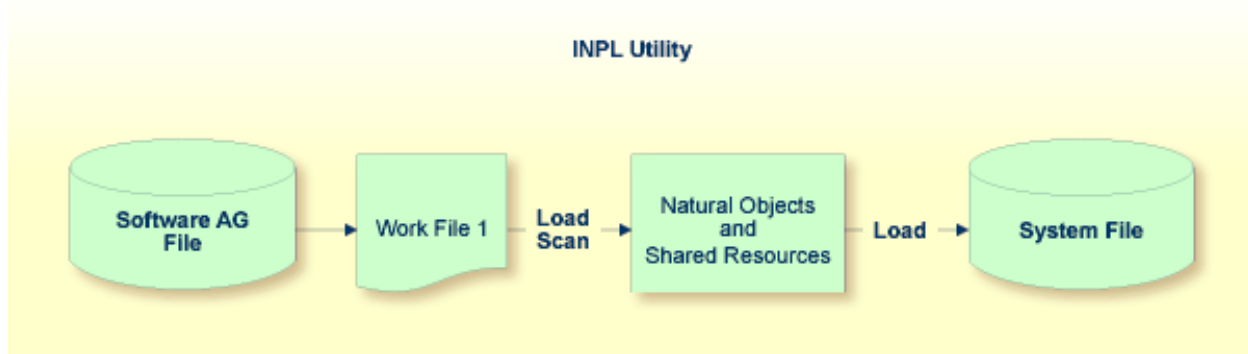
▪ Introducing the INPL Utility	66
▪ Load Libraries Only	71
▪ Load DDMs Only	72
▪ Load Error Messages Only	73
▪ Load All Objects	73
▪ Scan INPL File	74
▪ Natural Security Recover	74
▪ User Exit Routines	75

The INPL utility (Initial Natural Program Load) is used to load or scan Natural objects and shared resources from files supplied by Software AG.

Introducing the INPL Utility

The INPL utility processes Natural objects and shared resources provided by Software AG.

The following diagram is a basic illustration of the INPL functionality:



The Natural objects and shared resources are delivered as installation or update files which are assigned to Work File 1. The INPL utility loads the Natural objects and shared resources from Work File 1 into Natural system files.

The Natural objects and shared resources include cataloged objects and source objects that are contained in libraries in the Natural system files FNAT and FUSER.

In addition to loading Natural objects and shared resources, the INPL utility provides a scan function to check the contents of the file assigned to Work File 1 and a **Natural Security Recover** function which forces initialization of the Natural Security environment.

When loading cataloged objects into Natural system files, the INPL utility deletes any buffer pool entries of cataloged objects with identical names if contained in the same buffer pool used by the INPL utility.

If an error occurs during INPL execution, the INPL will be interrupted and terminate abnormally with Condition Code 40.

This section covers the following topics:

- [Restrictions](#)
- [Special Case](#)
- [Invoking INPL](#)
- [Batch and Direct Command Mode](#)

- [Options Available](#)
- [INPL Report](#)

Restrictions

You can process only files which are marked as “SAG system INPL file”.

Special Case

When an INPL is to be performed in a Natural Security environment, the INPL command can be specified using the dynamic Natural profile parameter `STACK`.

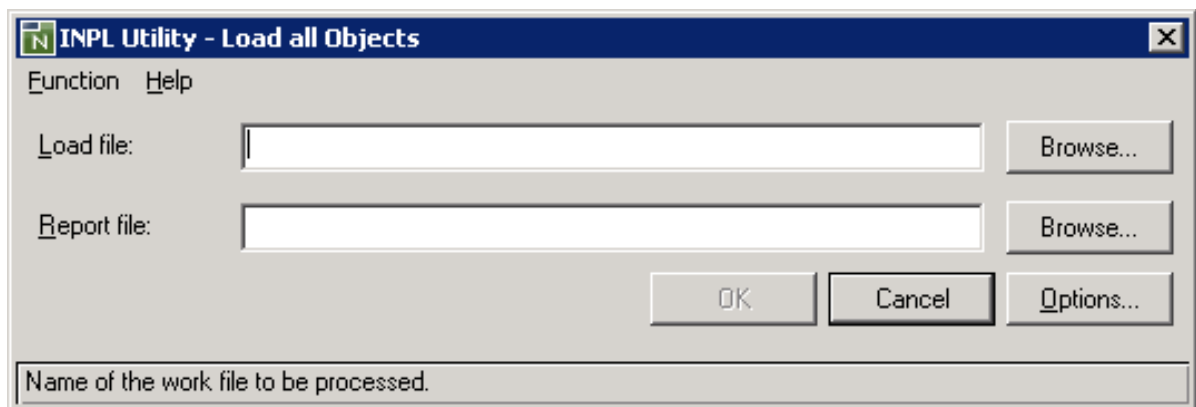
Invoking INPL

▶ To invoke the INPL utility

- 1 Enter the following Natural system command:

```
INPL
```

An INPL utility window similar to the example below is displayed:

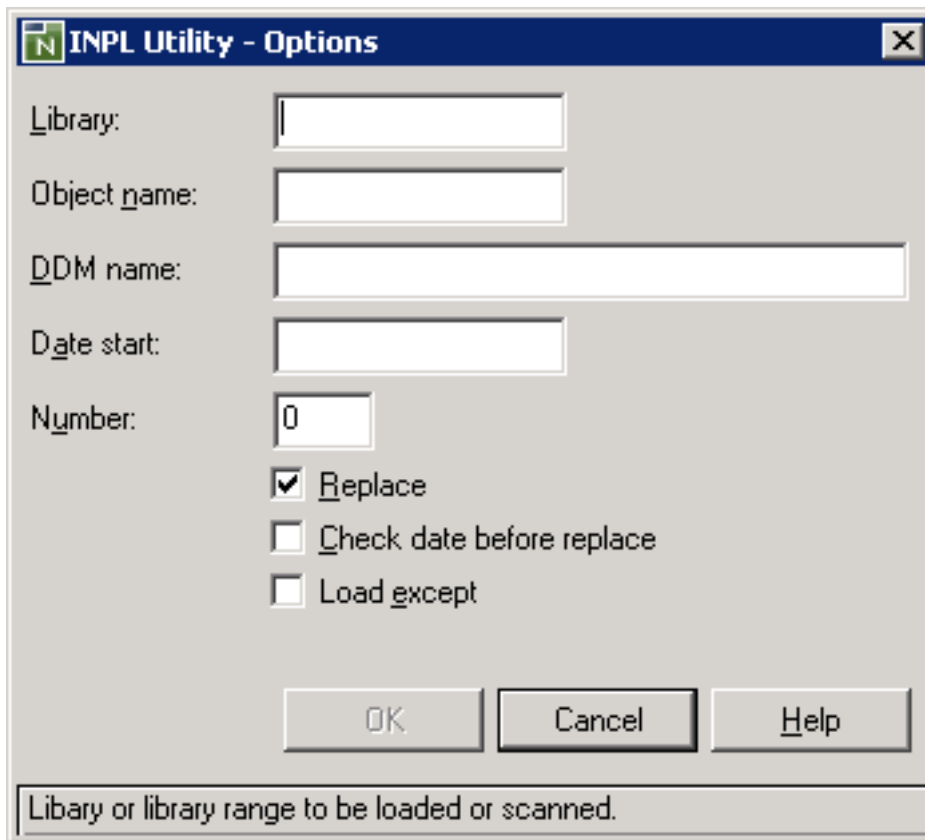


- 2 Select the file you want to load and the file into which the **INPL report** (see below) is to be written. For detailed information, refer to [Options Available](#).
- 3 Choose **OK** to confirm your entries.
- 4 From the **Function** menu, choose one of the following functions:
 - **Load Libraries Only**
 - **Load DDMs Only**
 - **Error Messages Only**
 - **Load All Objects**

- [Scan INPL File](#)
- [Natural Security Recover](#)

For detailed information on these functions, refer to the corresponding sections.

- 5 Choose the **Options** button and, in the **Options** dialog box, enter the parameters to be applied during execution of this function.



For details, refer to [Options Available](#).

Batch and Direct Command Mode

When you run INPL in batch or when you use the direct-command mode, the input data used in connection with the INPL command corresponds to the codes and values that apply to the fields provided with a character user interface as shown in the example of an INPL menu in a UNIX environment below and described in [Options Available](#). For example, assuming the session parameter ID is set to comma (,), the instruction for loading a specific DDM may read:

```
INPL D, , , ddm-name
```



```

11:04:48          ***** NATURAL INPL UTILITY *****          2001-11-09
User: SAG                                               Library: SYSTEM

Code   Function

L     Load Libraries Only
D     Load DDMs Only
E     Load Error Messages Only
B     Load All Objects
S     Scan INPL File
R     Natural Security Recover
?     Help
.     Exit

Code ..... B
Replace .... Y (Y/N/O)      Load Except . N (Y/N)
DDM Name ....
Library .....
Object Name .              Date .....          (YYYY-MM-DD)
Check Date .. N (Y/N)      Number ..... 0
File Type ... D (D/P)
Load File ... $NETWORK/SAGLOAD.sag
Report File . $HOME/report.txt

```

Options Available

The following section describes the text boxes in the INPL utility dialog boxes where you can specify the file to be used for the INPL and one or more parameters as object selection criteria for the INPL function selected from the **Function** menu. The use of a parameter depends on the respective function as indicated in the relevant documentation sections.

Item	Description
Load file	The name of the file to be loaded.
File type (batch or direct commands only)	INPL automatically recognizes the type of the load file such as binary or portable. However, due to compatibility reasons, the File type parameter must still be specified when executing INPL in batch or direct command mode , but it will not be evaluated.
Report file	The name of the file into which the INPL report (see below) is to be written.
Library	The name of a library or a range of names. If you enter a value that ends with an asterisk (*), each library with a name that starts with the specified value is processed. The library name is mandatory if Object name is specified.
Object name	The name of a Natural object (except DDMs) or a range of names. If the value ends with an asterisk (*), each object with a name that starts with the specified value is processed.

Item	Description
	If this text box is empty, all objects contained in the library specified in the Library text box are processed.
DDM name	<p>The name of a DDM or a range of names.</p> <p>If you enter a value that ends with an asterisk (*), each DDM with a name that starts with the specified value is processed. If only an asterisk (*) is entered or if this text box is empty, all DDMs are processed.</p>
Date start	<p>Restricts processing to Natural objects and shared resources which were saved or cataloged on or after the date entered in this text box.</p> <p>The date must be entered in the format <i>YYYY-MM-DD</i> (<i>YYYY</i> = year, <i>MM</i> = month, <i>DD</i> = day).</p>
Number	<p>Limits processing of Natural objects and shared resources to a specified number. All objects are counted which are loaded or scanned according to the specified selection criteria.</p> <p>If the number of Natural objects processed has reached the value entered in the Number text box, processing is terminated with a corresponding message.</p>
Replace	<p>Specifies whether the Natural objects and shared resources to be processed are to replace any that already exist on the system files.</p> <p>Possible settings are:</p> <p>Checked All existing Natural objects and shared resources are replaced. This is the default setting.</p> <p>Unchecked Existing Natural objects and shared resources are <i>not</i> replaced.</p> <p>See also Check date before replace to replace only Natural objects and shared resources that are older than the Natural objects and shared resources to be loaded.</p>
Check date before replace	<p>Specifies whether existing Natural objects and shared resources are to be replaced depending on their time stamp.</p> <p>This parameter has no effect if Replace is not selected.</p> <p>Possible settings are:</p> <p>Checked Only objects which are older than the Natural objects or shared resources of the same name are replaced. An object is older if it was saved or cataloged before the object to be loaded.</p> <p>Unchecked All objects are replaced. This is the default setting.</p>
Load except	<p>Specifies whether to exclude Natural objects and shared resources from processing.</p> <p>This parameter does not apply to error messages.</p> <p>Possible settings are:</p>

Item	Description
	<p>Checked All Natural objects and shared resources are processed except for the objects specified in the text boxes DDM name, Library and/or Object name.</p> <p>Unchecked No exceptions; all Natural objects and shared resources are processed. This is the default setting.</p> <p>Examples of load exceptions:</p> <p>All libraries except the library ABC are loaded: Function = Load Libraries Only Library = ABC</p> <p>All DDMs with a prefix other than XY are loaded: Function = Load DDMs Only DDM name = XY*</p> <p>All objects contained in libraries with a prefix other than AB and all DDMs with a prefix other than CD are loaded: Function = Load All Objects Library = AB* DDM name = CD*</p>
Natural Security recover object	<p>Only applies if the function Natural Security Recover has been selected.</p> <p>If checked, resets the owner information of specified Natural objects.</p>

INPL Report

When the selected INPL function is complete, a corresponding INPL report is displayed in a list box (online mode).

Load Libraries Only

This function of the INPL utility is used to load Natural cataloged objects and source objects and shared resources into specified libraries in the Natural system file FNAT or FUSER.

► To load libraries

- 1 From the **Function** menu, choose **Load Libraries Only**. You can specify parameters to be valid during execution of this function:
 - **Replace**
 - **Load except**

- **Library**
- **Object name**
- **Date start**
- **Check date before replace**
- **Number**

For detailed information on these parameters, refer to *Options Available* in the section *Introducing the INPL Utility*.

- 2 Confirm your entries.

When the function is complete, a corresponding **INPL report** (see the section *Introducing the INPL Utility*) is output.

Load DDMs Only

This function of the INPL utility is used to load DDMs into the libraries indicated in the work file.

▶ To load DDMs

- 1 From the **Function** menu, choose **Load DDMs Only**. You can specify parameters to be valid during execution of this function:

- **Replace**
- **Load except**
- **DDM name**
- **Number**

For detailed information on these parameters, refer to *Options Available* in the section *Introducing the INPL Utility*.

- 2 Confirm your entries.

When the function is complete, a corresponding **INPL report** (see the section *Introducing the INPL Utility*) is output.

Load Error Messages Only

This function of the INPL utility is used to load user-defined error messages or system error messages into specified libraries in the Natural system file FUSER or FNAT respectively.

▶ **To load error messages**

- 1 From the **Function** menu, choose **Load Error Messages Only**. You can specify parameters to be valid during execution of this function:

- **Replace**
- **Library**

For detailed information on these parameters, refer to *Options Available* in the section *Introducing the INPL Utility*.

- 2 Confirm your entries.

When the function is complete, a corresponding **INPL report** (see the section *Introducing the INPL Utility*) is output.

Load All Objects

This function of the INPL utility is used to load all Natural objects (including error messages and DDMs) and shared resources into the libraries indicated in Work File 1.

▶ **To load all objects and shared resources**

- 1 From the **Function** menu, choose **Load All Objects**. You can specify parameters to be valid during execution of this function:

- **Replace**
- **Load except**
- **DDM name**
- **Library**
- **Object name**
- **Date start**
- **Check date before replace**
- **Number**

For detailed information on these parameters, refer to *Options Available* in the section *Introducing the INPL Utility*.

- 2 Confirm your entries.

When the function is complete, a corresponding **INPL report** (see the section *Introducing the INPL Utility*) is output.

Scan INPL File

This function of the INPL utility is used to scan the contents of the file assigned to Work File 1.

▶ To scan an INPL File

- 1 From the **Function** menu, choose **Scan INPL File**. You can specify parameters to be valid during execution of this function:

- **Load except**
- **DDM name**
- **Library**
- **Object name**
- **Check date before replace**
- **Number**

For detailed information on these parameters, refer to *Options Available* in the section *Introducing the INPL Utility*.

- 2 Confirm your entries.

When the function is complete, a corresponding **INPL report** (see the section *Introducing the INPL Utility*) is output.

Natural Security Recover

This function of the INPL utility is used to force initialization of the Natural Security environment.

The following options are provided:

- **Reset Environment**

- [Remove Owners](#)

Reset Environment



Caution: Execution of this function will reset the user profile DBA and the library profile SYSSEC as well as the link between these two objects as they were after the initial installation; all other links to the library SYSSEC will be canceled. Other Natural Security profiles and links will not be modified. Contact Software AG technical support for further information.

▶ To reset the environment

- From the **Function** menu, choose **Natural Security Recover**.

Remove Owners

▶ To remove owners

- 1 From the **Function** menu, choose **Natural Security Recover**.
- 2 Choose the **Options** button.
- 3 In the **Options** dialog box, select the **Natural Security recover object** check box to reset the owner information of specified objects.

User Exit Routines

An INPL user exit routine is supplied as source object INPLSX nn in the Natural system library SYSLIB where nn denotes the ID of the user exit routine.

▶ To activate a user exit routine

- 1 Copy the source code from SYSLIB into a user library.
- 2 Catalog it under the name INPLUX nn .
- 3 Copy it back into the Natural system library SYSLIB.



Note: The source object that you might have modified, and the cataloged object of the user exit routine are renamed to avoid them to be overwritten by an update installation.

The following user exit routines are available:

Name	Function
INPLUX01	Prevent error message texts to be replaced.

INPLUX01

You can use this user exit to define ranges for error messages (user defined or Natural system error messages) that cannot be replaced during an INPL session. For further details, see the source of INPLSX01 in the Natural system library SYSLIB.

VI

Installer

6 Installer

The Natural Installer is used to install in your local environment packed versions of Natural add-on products such as, Natural Security. In addition, you can update Natural add-on products or uninstall Natural add-on products that have been installed with the Natural Installer.

The Natural Installer provides a wizard that guides you through a sequence of windows. These help you perform the installation step by step.

The Natural Installer is stored in the Natural system library SYSOBJH.

▶ To invoke the Natural Installer

- In the Natural main window, from the **Tools** menu, choose **Configuration Tools > Natural Installer**.

Or:

In the Natural main window, from the **View** menu, choose **Command Line** and enter the system command `SYSINST`.

Or:

Execute the Natural program `NATINST`.

`NATINST` is stored in the Natural system library SYSOBJH.

The Natural Installer welcome window is displayed. Follow the instructions specified in this window and all subsequent windows.

VII Object Handler

The Object Handler is designed to process Natural and non-Natural objects for distribution in Natural environments. This is done by unloading the objects in the source environment into work files and loading them from work files into the target environment.

General Information on the Object Handler	Invoking the Object Handler in batch or online mode; applying Natural Security.
Functions	Using the Object Handler menu functions: unload, load, restart load, scan, view, find and administration.
Object Specification	Specifying the objects to be processed with Object Handler menu functions.
Settings	Specifying option and parameter settings for Object Handler menu functions.
Workplans	Using standard procedures to execute Object Handler functions.
Name, Date and Time Specification	Specifying names, dates, times and ranges.
Work Files	Work files used by the Object Handler.
Direct Commands	Using direct commands to perform Object Handler functions.
Batch Condition Codes and User Exit Routines	Condition codes and user exit routines provided in batch mode.
Tools	Displaying status information, setting trace and report options, and transferring work files.
Options	Invoking or activating specific Object Handler options.
Profile Settings	Setting up a profile to define individual defaults and standard procedures.
Migration from SYSTRANS to the Object Handler	Migrating from the utility SYSTRANS to the Object Handler.

7 General Information on the Object Handler

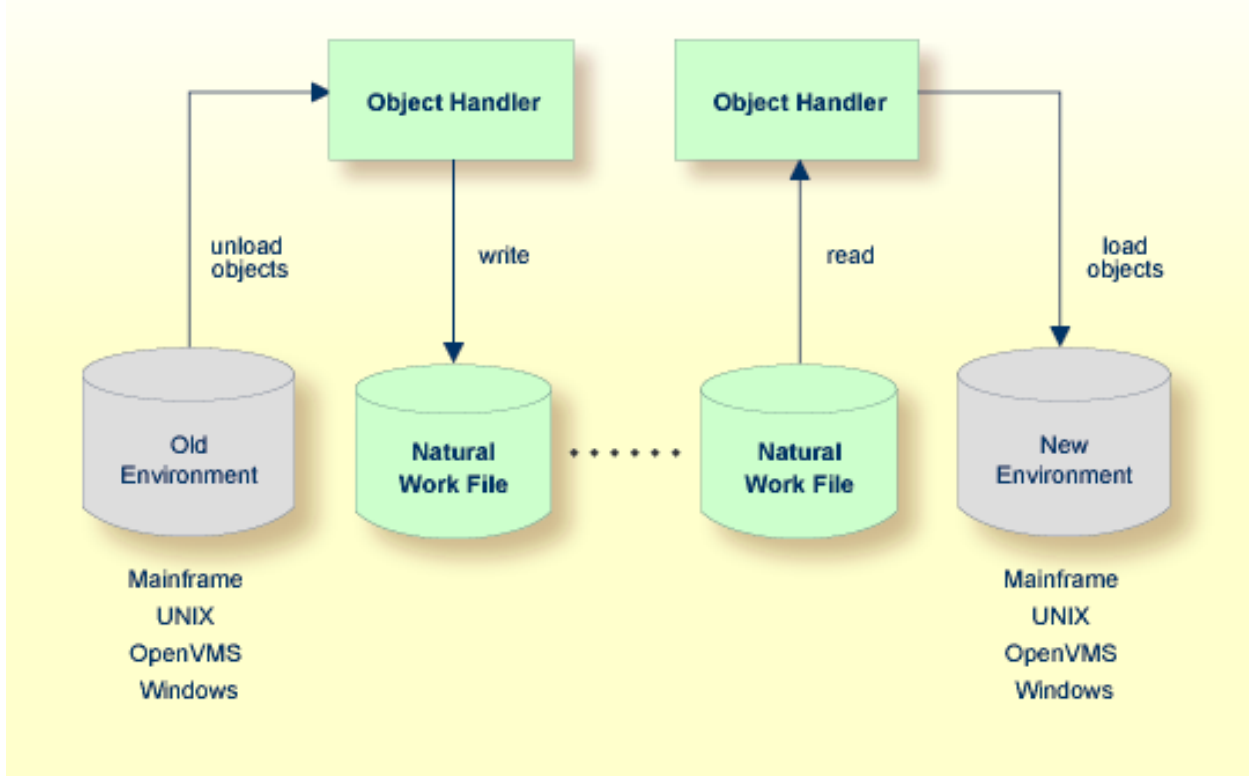
▪ Principles of Object Transfer	84
▪ Invoking the Object Handler	86
▪ Batch or Direct Command Calls	87
▪ Issuing Object Handler Commands from a Natural Program	88
▪ Text Members for Reports, Restarts and Traces	88
▪ Natural Security	88
▪ Using FDDM System Files	89

The Object Handler consists of the utility SYSOBJH which is located in the Natural system library SYSOBJH, and the direct command interface. Additionally, the Application Programming Interface OBJHAPI is provided for executing Object Handler functions from a Natural program.

Principles of Object Transfer

The diagram below illustrates how the Object Handler transfers objects by unloading them from the source environment into work files and loading them from work files into the target environment. If required, an application protocol such as FTP can be used for transferring work files from source to target environments.

When connected to a Natural Development Server in a remote environment located on a Windows, mainframe, UNIX or an OpenVMS platform, the Object Handler can be used to directly unload or load objects from or into a work file that is located on the local Windows client.



Related Topic:

Natural Development Server documentation

This section covers the following topics:

- [Transfer Environment and File Security](#)
- [Objects Processed by the Object Handler](#)
- [Formatting Options](#)

Transfer Environment and File Security

An old or a new environment is an FNAT, FUSER or FDIC system file contained in an Adabas database or a VSAM file system on a mainframe, or in the file system on a UNIX, an OpenVMS or a Windows platform. Natural objects on the FNAT or FUSER system file can be contained in libraries as indicated in the following section.

The file security (that is, passwords and cipher codes) relates to the security that has been defined for a system file in an Adabas or a VSAM environment. If file security has been defined for a system file, you need to specify a password, cipher code and/or VSAM name for the source and/or target system file required before you perform an Object Handler function. Otherwise, Adabas or VSAM will issue an appropriate error message. You do not have to provide security information for the default system files assigned to the Natural session at the start of the Object Handler.

Objects Processed by the Object Handler

The Object Handler transfers Natural source objects (also referred to as saved objects) and cataloged objects which are contained in Natural libraries, Natural error messages, Natural command processor sources, Natural-related objects, Adabas FDTs (Field Definition Tables) and external files.

Formatting Options

You can transfer data of binary or text format, depending on the source and target environment where the objects are processed.

Binary format can be used for source objects and cataloged objects, error messages, Natural command processor sources, Natural-related objects and Adabas FDTs and external files.

Text format applies to source objects, Natural command processor sources, error messages and Adabas FDTs. You can only transfer text data between mainframe and UNIX/OpenVMS/Windows platforms. You can transfer binary data between identical platforms. Between UNIX or OpenVMS and Windows platforms, you can transfer binary data by using portable work files of internal format.

Invoking the Object Handler

To invoke the Object Handler you can either use menu functions or direct commands.

If you start the Object Handler while a remote Natural Development Server environment is active, the Object Handler will be invoked for the remote environment currently mapped and will process only the objects contained in this environment. In Natural Studio, the current environment is indicated in the title of the **SYSOBJH - Object Handler** window, which appears when you invoke the Object Handler as described in the following section.

▶ To invoke the Object Handler using menu functions

- 1 In the Natural Studio window, from the **Tools** menu, choose **Development Tools and Object Handler**.

Or:

In the Natural Studio window, from the **View** menu, choose **Command Line** and enter the following system command:

```
SYSOBJH
```

The **Welcome to the Natural Object Handler** window of the Object Handler appears with the following options:

- Unload
 - Load
 - Administration
- 2 Select the **Advanced user** check box if you do not want to use the Object Handler wizards for function processing.

Select the function required either by choosing the corresponding command button or by selecting the corresponding function from the **Actions** menu.

In addition to the functions listed above, the **Actions** menu provides the following:

- **View**
- **Find**
- **Scan Work File**
- **Change Workplan Library**
- **Restart Load**

See the section *Functions* for descriptions of these functions, and how to process the functions in advanced-user mode or by using wizards.

► **To invoke the Object Handler in batch or direct command online mode**

- Enter the system command `SYSOBJH` followed by a direct command as described in *Batch or Direct Command Calls* and *Direct Commands*.

After execution of a direct command, you can enter either another direct command or a period (.) to exit the Object Handler.

Batch or Direct Command Calls

Several commands can be issued to the Object Handler online or in batch mode. The last command in the command sequence must be a period (.), `STOP`, `END`, `QUIT` or `FIN`, where `FIN` ends the Natural session.

The section covers the following topics:

- [Batch Mode](#)
- [Online Mode](#)

Batch Mode

If you invoke the Object Handler in batch mode, you must issue a direct command to execute an Object Handler function.

The commands to the Object Handler are read from standard input. Each command can be separated into a maximum of 20 command parts/strings by entering input delimiters (session parameter `ID`) after any keyword or keyword value. Each command part/string must not exceed 248 bytes.

If the command is longer than a single line, at the end of every line except the last that belongs to the command, enter the character defined with the session parameter `CF` (default is %) This indicates continuation on the next line. However, this is only possible if you specify the command `SYSOBJH` in a line by itself. That is, you cannot use `CF`, if you enter `SYSOBJH` in the same line where a multi-line command starts.

Example (assuming `ID` is set to ,):

```
UNLOAD * LIB EXAMPLE, WHERE, WORK C:\TEMP\TEST.SAG
STOP
```

Related Topics:

- [Direct Commands](#)

- [Batch Processing in a Remote Environment](#) in *Examples of Using Direct Commands*
- [Natural in Batch Mode - Operations](#) documentation

Online Mode

The command to the Object Handler in the Command line can consist of up to 20 command parts. Several commands can be issued to the Object Handler. The last command in the command sequence must be a period (.), STOP, END, QUIT or FIN, where FIN ends the Natural session.

Example:

```
SYSOBJH UNLOAD * LIB EXAMPLE WHERE WORK C:\TEMP\TEST.SAG  
STOP
```

Issuing Object Handler Commands from a Natural Program

You can issue commands to the Object Handler with a Natural program by using the OBJHAPI Application Programming Interface, which is supplied as a subprogram in the Natural system library SYSOBJH. For the parameters required and examples, see the Natural program DOC-API supplied in the library SYSOBJH.

Text Members for Reports, Restarts and Traces

Only applies to remote environments located on mainframe platforms.

Report, restart and trace data created by the Object Handler are stored as Natural text members (Natural objects of the type Text) in the Workplan library. The Object Handler generates names for text members that have not been explicitly specified in the **Options** window. The names generated are a combination of the weekday and the time. For example: a member with the name 21415568 was created on Tuesday (the second day of the week) at 14:15:56,8.

Natural Security

The use of the Object Handler under Natural Security requires that utility profiles be defined for it in Natural Security. At least, a default profile must be defined. For information on utility profiles, see the section *Protecting Utilities* in the *Natural Security* documentation.

If Natural Security is installed, the Object Handler checks the SYSOBJH utility profiles in Natural Security to find out whether the requested function is allowed.

Should a Natural Security error occur during the load function, the following applies:

- If the **Write report** option is set, in online mode, the error message is written to the report file and processing continues for the current load command.
- If the **Write report** option is set, in batch mode, the error message is written to the report file and the Object Handler terminates after the load command where the error occurred has finished processing.
- If the **Write report** option is not set, an error message is issued and the load command is terminated.

Using FDDM System Files

Natural DDMs (data definition modules) can be stored in libraries or the system file FDDM. See also: *FDDM - Natural System File for DDMs* in the *Parameter Reference* documentation).

To use the system file FDDM for processing DDMs with the load, unload or find function, the Object Handler provides the option **Use FDDM file for processing DDMs**. This option is set by selecting **Use additional options** (see the section *Settings*).

Consider the following when selecting **Use FDDM file for processing DDMs**:



- This option is selected by default if FDDM has been activated in the NATPARM module.
- You cannot process DDMs that are stored in libraries.
- You need to specify the library SYSTEM and the Natural object type V (see *Natural Library Object Details* in the section *Object Specification*).
- If used with the load function, all DDMs are loaded into the system file FDDM. In this case, the parameter `NEWLIBRARY` is ignored.

8 Functions

This section describes the main functions provided by the Object Handler.

If you execute an Object Handler function while a remote Natural Development Server environment is active, this function will process all objects in the remote environment currently mapped in Natural Studio or with a direct command if used.

You can take advantage of the Object Handler wizards to guide you through the steps required to execute the unload, load, scan and administration functions. The wizards are activated by default. If you prefer the unload, load or scan mode for the experienced user instead, select the check box next to **Advanced user** in the **Welcome to the Natural Object Handler** window or adjust the default input mode by using the Object Handler profile option. See also [Profile Settings](#).

-  **Tip:** You can create standard procedures to define recurring settings and object specifications which automate the processing of the unload, load or scan function, see [Workplans](#).
-  **Caution:** You cannot unload, load or scan external files in remote environments located on mainframe platforms. Natural DDMs (data definition modules) can *only* be unloaded, loaded or scanned in remote environments located on mainframe platforms.

This section covers the following topics:

[Wizards](#)

[Advanced User](#)

[Restart Load](#)

[View](#)

[Find](#)

[Scan](#)

[Administration](#)

[Change Workplan Library](#)



Note: *Change Workplan Library* is described in the section *Administration*.

9 Wizards

The Object Handler provides wizards that determine the processing sequence for the following:

- Unloading data from the Natural system environment into Natural work files.
- Loading data from work files into the Natural system environment.
- Finding objects in your Natural environment.
- Scanning the contents of Natural work files.
- Performing administration functions.

This section provides general information on the wizards and detailed instructions for using the unload and load wizards:

General Information on Wizards

Unload Wizard

Load Wizard



Notes:

1. The unload wizard includes the find function and the load wizard includes the scan.
2. The **administration wizard** is described in the section *Administration*.

10

General Information on Wizards

- Invoking Wizards 96
- Navigation and Command Execution 96

This section provides information on invoking the Object Handler wizards, navigating between the windows provided and executing commands.

Invoking Wizards

▶ To invoke the wizards

- In the **Welcome to the Natural Object Handler** window, remove the mark from the **Advanced user** check box if required (the box is not selected by default).

Or:

From the **Options** menu, select **Advanced User**.

Navigation and Command Execution

To navigate between the processing windows of a wizard, choose the command buttons **Next** and **Back**: choose **Next** to confirm settings and continue processing, choose **Back** to modify settings made in a previous step. Choose the **Cancel** command button whenever you want to stop the processing sequence and return to the **Welcome to the Natural Object Handler** window.

Before finally executing a function, the wizard displays the command or command procedure generated for the relevant function and the settings and object specifications made. You can again choose **Back** to return to a previous window and modify the setting, or confirm and execute the command with **Next**.

After you have executed the command or command procedure for the unload, load or scan function, you can continue processing and reuse the settings previously made or choose the **Cancel** command to terminate the function.

11 Unload Wizard

▪ Unload Objects into Natural Work File(s)	98
▪ Find Objects	101
▪ Start Object Handler Command Procedure	102

This section provides instructions for using the unload wizard and describes the items available in wizard windows.

▶ **To start the unload wizard**

- In the **Welcome to the Natural Object Handler** window, remove the mark from the **Advanced user** check box and choose the **Unload** command button.

Or:

From the **Actions** menu, choose **Unload**.

The initial **Unload Wizard** window appears.

The options provided in the **Unload Wizard** window are described in this section.

Unload Objects into Natural Work File(s)

With the wizard function **Unload objects into Natural work file(s)**, you are guided through the sequence of windows described below, where you can specify option and parameter settings and the type of object for the unload to be performed:

- [Set Options](#)
- [Set Parameters](#)
- [Select Objects](#)

Set Options

In the options window of the unload wizard, select any of the options to be used for function processing and, if required, fill the text box:

Item	Explanation
Transfer format	<p>Only valid if Use default options (this is the default) or Use additional options has been selected (see below).</p> <p>If selected, the data to be processed is written in Transfer format to the work file. See also Work File Format in <i>Work Files</i>.</p>
Local work file	<p>Only applies to remote environments.</p> <p>Only valid if Use default options (this is the default) or Use additional options has been selected (see below).</p> <p>Specifies the location of the work file when using Object Handler functions in connection with SpoD (Single Point of Development). If Local work file is selected, the data to be processed is written to the specified work file in the local file system.</p>

Item	Explanation
	See also WFLOC in <i>Direct Commands</i> .
Portable work file	<p>Not applicable to work files located in a remote environment on a mainframe platform.</p> <p>This option is only valid if the following applies:</p> <ul style="list-style-type: none"> ■ Use default options (this is the default) or Use additional options has been selected (see below). ■ Transfer format has <i>not</i> been selected. <p>If Portable work file has been selected, the work file is written or read in portable format. See also Work File Format in <i>Work Files</i>.</p>
Unicode work file	<p>Only applies if Transfer format has been selected.</p> <p>If this option is selected, all object sources are converted to Unicode/UTF-8 (Universal Transformation Format, 8-bit form) before they are written to the work file.</p>
Fixed length	See FIXEDLENGTH in <i>Direct Commands</i> .
Unload file (Server)	<p>Only valid if Use default options (this is the default) or Use additional options has been selected. See below.</p> <p>The name of the work file to be used for the function. See also Work Files. On mainframes, the current Work File 1 is used as the default unload file.</p>
Browse	<p>Not applicable to server unload files.</p> <p>Invokes the browse function to select a work file from a directory.</p>
Use default options	Default options are used (this is the default). For the options available, see Set Additional Options in <i>Settings - Options</i> .
Use additional options	Used in connection with Set (see below).
Set	<p>Only activated if Use additional options has been selected.</p> <p>Invokes the Unload Options window where you can modify the default settings and enter additional options for the processing sequence. See also Set Additional Options in <i>Settings - Options</i>.</p>
Use Option Workplan	<p>If selected, a Workplan of the type OPTION is used.</p> <p>Select a Workplan from the combo box or type the name of a Workplan of the type OPTION.</p> <p>See also Workplans.</p>
List	<p>Only valid if Use Option Workplan (see above) has been selected and the name of a valid Workplan of the type OPTION was entered.</p> <p>Displays the contents of the Workplan specified.</p>

Set Parameters

Applies to the unload function only.

In the parameters window of the unload wizard, select any of the options to be used for function processing and, if required, fill the text box:

Item	Explanation
Do not use parameters	If selected (default setting), no parameters are set.
Use global parameters	If selected, global parameters are used. See also Set Global Parameters in <i>Settings</i> .
Set	Only activated if Use global parameters has been selected. If selected, the global parameters window is invoked. See Set Global Parameters in <i>Settings</i> and parameter-setting (Direct Commands) for descriptions of keywords and valid values.
Use Parameter Workplan	If selected, a Workplan of the type PARAMETER is used. Select a Workplan from the combo box or type the name of a Workplan of the type PARAMETER. See also Workplans .
List	Only valid if Use Parameter Workplan (see above) has been selected and the name of a valid Workplan of the type PARAMETER was entered. Displays the contents of the Workplan specified.

Select Objects



Note: You cannot unload Natural-related objects or external files in remote environments located on mainframe platforms. Natural DDMs can *only* be unloaded in remote environments located on mainframe platforms.

In the object type specification window, choose either of the following methods to specify the type of object you want to process:

1. Direct Specification

Select one of the following types of object:

- [Natural library objects](#)
- [Natural system error messages](#)
- [Natural command processor sources](#)
- [Natural-related objects](#)
- [External files](#)
- [FDTs](#)

■ Natural DDMs

Depending on the type of object selected, a window appears where you can specify selection criteria for the objects to be processed.

Specify the objects and choose **Details** (if available for the type of object selected) for more detailed specifications, if required. For the keywords and valid values that apply to each object type, see the relevant explanations in the section *Object Specification*.

2. Using a Workplan

Select **Use Selection or List** if you want to use a Workplan of the type SELECTION or LIST that predefines object selection criteria: see the section *Workplans* for more information.

In the **Selection or List** window, enter the name of a Workplan of the type SELECTION or LIST by using either option:

- Type in the name of a Workplan.

Or:

From the drop-down list box, choose a name from the list of Workplans available.

Choose the **List** command button if you want to list the contents of the Workplan specified.

Find Objects

The wizard function **Find objects** is used to locate objects in your Natural environment and generate a report list of the objects found.

Find objects guides you through the same sequence of windows and the same setting or object specification options as described earlier for the wizard function **Unload objects into Natural work file(s)** above. Settings or specifications that only apply to the unload function (for example, unload file) are excluded from the respective windows.

The report generated by the **Find objects** function contains several table columns with information on the objects listed. For information on the table columns, refer to the section *Object Specification*.

For general information on the execution of the find command and possible error messages, choose the **Details** button. See also *Last Result* in the section *Tools*.

Start Object Handler Command Procedure

Choose the function **Start Object Handler command procedure** if you want to execute a standard procedure (Workplan) of the type PROCEDURE with predefined settings and object specifications for the unload function to be performed. See also the section [Workplans](#) for further information.

▶ **To start and execute an Object Handler command procedure**

- 1 In the initial **Unload Wizard** window, choose **Start Object Handler command procedure**.

The procedure window appears.

- 2 In the field **Procedure name**, enter the name of a Workplan of the type PROCEDURE (see also [Workplans](#)) by using either option:

- Type in the name of a Workplan.

Or:

From the drop-down list box, choose a name from the list of Workplans available.

- Choose the **List** command button if you want to display the contents of the Workplan specified.

- 3 Confirm the contents of the PROCEDURE Workplan and execute the transaction.

12 Load Wizard

- Load/Scan Objects from/in Work Files 104
- Start Object Handler Command Procedure 107
- Load SYSPAUL Application 108

This section provides instructions for using the load wizard and describes the items available in wizard windows.

▶ **To start the load wizard**

- In the **Welcome to the Natural Object Handler** window, remove the mark from the **Advanced user** check box (if necessary) and choose the **Load** command button.

Or:

From the **Actions** menu, choose **Load**.

The initial **Load Wizard** window appears.

The options provided in the **Load Wizard** window are described in this section.

Load/Scan Objects from/in Work Files

With the wizard function **Load objects from Natural work file(s)** or **Scan objects in Natural work file(s)**, you are routed through the sequence of windows described in this section, where you can specify the option or parameter settings and the type of object for the load or scan to be performed.

- [Set Options](#)
- [Set Parameters](#)
- [Select Objects](#)

Set Options

In the options window of the load/scan wizard, select any of the options to be used for function processing and, if required, fill the text box.

Item	Explanation
Transfer format	<p>Only valid if Use default options (this is the default) or Use additional options has been selected. See below.</p> <p>If selected, the data to be processed is written in Transfer format from the work file. Load/scan data is expected to be in Transfer format. See also Work File Format in <i>Work Files</i>.</p>
Local work file	<p>Only applies to remote environments.</p> <p>Only valid if Use default options (this is the default) or Use additional options has been selected. See below.</p>

Item	Explanation
	<p>Specifies the location of the work file when using Object Handler functions in connection with SpoD (Single Point of Development). If Local work file is selected, the data to be processed is read from the work file specified in the local file system.</p> <p>See also WFLOC in <i>Direct Commands</i>.</p>
Portable work file	<p>This option is not required for the load and scan functions, which automatically choose the appropriate work file type and ignore the option if set.</p> <p>In addition, this option does not apply to work files located in a remote environment on a mainframe platform.</p> <p>Portable work file is only valid if the following applies:</p> <ul style="list-style-type: none"> ■ Use default options (this is the default) or Use additional options has been selected (see below). ■ Transfer format has <i>not</i> been selected. <p>See also Work File Format in <i>Work Files</i>.</p>
Load file or Scan file (Server)	<p>Only valid if Use default options (this is the default) or Use additional options has been selected. See below.</p> <p>The name of the work file to be used for the function. See also Work Files. On mainframes, the current Work File 1 is used as the default load/scan file.</p>
Browse	<p>Not applicable to server load/scan files.</p> <p>Invokes the browse function to select a work file from a directory.</p>
Use default options	<p>Default options are used (this is the default). For the options available, see Set Additional Options in <i>Settings - Options</i>.</p>
Use additional options	<p>Used in connection with Set (see below).</p>
Set	<p>Only activated if Use additional options has been selected.</p> <p>Invokes the Load/Scan Options window where you can modify the default settings and enter additional options for the processing sequence. See also Set Additional Options in <i>Settings - Options</i>.</p>
Use Option Workplan	<p>If this option is selected, a Workplan of the type OPTION is used.</p> <p>Select a Workplan from the combo box or type the name of a Workplan of the type OPTION.</p> <p>See also Workplans.</p>
List	<p>Only valid if Use Option Workplan (see above) has been selected and the name of a valid Workplan of the type OPTION was entered.</p> <p>Displays the contents of the Workplan specified.</p>

Set Parameters

Applies to the load function only.

In the parameters window of the load wizard, select any of the options to be used for function processing and, if required, fill the text box:

Item	Explanation
Do not use parameters	If selected (default setting), no parameters are set.
Use global parameters	If selected, global parameters are used. See also Set Global Parameters in <i>Settings</i> .
Set	Only activated if Use global parameters has been selected. If selected, the global parameters window is invoked. See Set Global Parameters (<i>Settings</i>) and parameter-setting (<i>Direct Commands</i>) for descriptions of keywords and valid input values.
Use Parameter Workplan	If selected, a Workplan of the type PARAMETER is used. Select a Workplan from the combo box or type the name of a Workplan of the type PARAMETER. See also Workplans .
List	Only valid if Use Parameter Workplan (see above) has been selected and the name of a valid Workplan of the type PARAMETER was entered. Displays the contents of the Workplan specified.

Select Objects



Note: You cannot load or scan Natural-related objects or external files in remote environments located on mainframe platforms. Natural DDMs can *only* be loaded or scanned on mainframe platforms.

For loading FDTs, see also [FDTs](#) in the section *Object Specification*.

In the object type specification window, choose any of the following options to specify the type of object you want to process:

1. Select **Load/Scan all objects from work file** if you want to process *all* objects from the work file.
2. Select **Load/Scan selected objects from work file** to process a particular type of object:
 - [Natural library objects](#)
 - [Natural system error messages](#)
 - [Natural command processor sources](#)
 - [Natural-related objects](#)

- **External files**
- **FDTs**
- **Natural DDMs** (remote environments only)

Depending on the type of object selected, a window appears where you can specify selection criteria for the objects to be processed.

Specify the objects and choose **Details** (if available for the type of object selected) for more detailed specifications, if required.

For the keywords and valid values that apply to each object type, see the relevant explanations in the section *Object Specification*.

3. Select **Use Selection or List** if you want to use a Workplan of the type SELECTION or LIST that predefines object selection criteria: see the section *Workplans* for more information.

In the **Selection or List** window, enter the name of a Workplan of the type SELECTION or LIST by using either option:

Type in the name of a Workplan.

Or:

From the drop-down list box, choose a name from the list of Workplans available.

Choose the **List** command button if you want to list the contents of the Workplan specified.

Start Object Handler Command Procedure

Choose the function **Start Object Handler command procedure** if you want to execute a standard procedure (Workplan) of the type PROCEDURE with predefined settings and object specifications. See also the section *Workplans* for further information.

▶ To start and execute an Object Handler command procedure

- 1 In the initial window of a wizard, choose **Start Object Handler command procedure**.

The **Start Procedure** window appears.

- 2 In the field **Procedure name**, enter the name of a Workplan of the type PROCEDURE (see also *Workplans*) by using either option:

- Type in the name of a Workplan.

Or:

From the drop-down list box, choose a name from the list of Workplans available.

- Choose **List** if you want to display the contents of the Workplan specified.
- 3 Confirm the contents of the PROCEDURE Workplan and execute the transaction.

Load SYSPAUL Application

Note that the Object Handler covers the functionality of the utility SYSPAUL, which is no longer installed per default.

SYSPAUL Applications can only be loaded in local environments.

▶ To load a SYSPAUL Application

- 1 Make sure that the utility SYSPAUL has been installed in the current FNAT system file.
- 2 In the **Load SYSPAUL Application** window, choose **Select** to select the name of the file *ap-
plinfo.txt* of the SYSPAUL Application to be loaded. It is located in the first directory of the SYSPAUL Application.
- 3 In the next window, the name of the SYSPAUL Application is displayed in the field **Name**.
- 4 The Object Handler loads the SYSPAUL Application and displays the result. The report file *sysload.log* is located in the temporary directory of Natural.

13 **Advanced User**

- Activating Advanced User 110
- Advanced User Unload 111
- Advanced User Load 112

The Object Handler provides a function processing sequence for the advanced user. The following functions are available if advanced-user mode is activated:

- Unload
- Load
- Administration
- View
- Find
- Scan Work File
- Restart Load

For the functions **administration**, **view**, **find**, **scan work file** and **restart load**, refer to the relevant sections in *Functions*.

This section describes how to activate advanced-user mode and provides a description of the unload or load processing sequence performed in this mode.

Activating Advanced User

▶ To activate advanced-user mode

- In the **Welcome to the Natural Object Handler** window, select the **Advanced user** check box (not selected by default).

Or:

From the **Options** menu, select **Advanced User**.

Or:

Set advanced-user mode as the default by choosing **Profile** from the **Options** menu, and changing the entries in the **SYSOBJH - Modify Profile** window:

From the **New user profile entry** drop-down list box, select **Advanced User**, in the **Entry value** box, enter a Y (Yes), and choose **Add**.

See also the section *Profile Settings*.

Advanced User Unload



Note: You cannot unload Natural-related objects and external files in remote environments located on mainframe platforms. Natural DDMs can *only* be unloaded in remote environments located on mainframe platforms.

▶ To unload objects in advanced-user mode

- 1 In the **Welcome to the Natural Object Handler** window, choose the **Unload** command button.

Or:

From the **Actions** menu, choose **Unload**.

The **Unload** window appears with a list of the types of object currently available in your Natural system environment.

- 2 Select one type of object and, from the context menu, choose **Unload**.

Or:

From the **Actions** menu, choose **Unload** and select the type of object required.

If you want to use a SELECTION or a LIST Workplan, you need to choose the second method and choose **Unload** from the **Actions** menu.

The **Unload** window appears for the type of object selected.

- 3 Specify the objects to be processed:
 - If available for the type of object selected, choose the **Details** button for further object specifications: see also the relevant sections in [Object Specification](#).
 - Choose the **Settings** button to specify unload options and parameters as described in the section [Settings](#).
- 4 Choose the **Unload** command button.

A message appears confirming the execution of the unload.

Advanced User Load

You cannot load Natural-related objects and external files in remote environments located on mainframe platforms. Natural DDMs can *only* be loaded in remote environments located on mainframe platforms.

To load FDTs, see also [FDTs](#) in the section *Object Specification*.

This section contains information on how to execute the load function in advanced-user mode.

▶ To load objects in advanced-user mode

- 1 In the **Welcome to the Natural Object Handler** window, choose the **Load** command button.

Or:

From the **Actions** menu, choose **Load**.

The **Load** window appears with a list of the types of objects currently available in your Natural system environment.

- 2 Select one type of object and, from the context menu, choose **Load**.

Or:

From the **Actions** menu, choose **Load** and select the type of object required.

If you want to use a SELECTION or a LIST Workplan, you need to choose the second method and choose **Load** from the **Actions** menu.

The **Load** window appears for the type of object selected.

- 3 Specify the objects to be processed:
 - If available for the type of object selected, choose the **Details** button for further object specifications: see also the relevant sections in [Object Specification](#).
 - Choose the **Settings** button to specify load options and parameters as described in the section [Settings](#).
- 4 Choose the **Load** command button.

A message appears confirming the load.

14 Restart Load

- Invoking Restart Load 114
- Specifying Permanent Members 115

Only applies to the load function and if Write restart information is set.

You can use the restart load function to resume load functions that terminated abnormally. If the load function terminates before the work file has been processed completely, with the restart load you can continue from the point of termination.

The restart load requires that restart information is written to Work File 6 or a specified restart file in accordance with the selection criteria, options and parameter settings specified for the load.

In local environments, the restart file is located in the local file system. Work File 6 is used for writing and reading restart data.

In remote environments, restart load data is written to a Natural object of the type Text (text member) located in the Workplan library. By default, this text member is a temporary object. We recommend that you specify a permanent text member for restart data as described below.

For information on setting the **Write restart information** option and specifying a restart file, see *Special* in *Set Additional Options* in the section *Settings - Options*.



Note: To display the name of the text member containing the restart data, from the **Tools** menu, choose **Show Status**.

Related Topics:

[RESTART](#) under *option-clause* in the section *Direct Commands*
[Change Workplan Library](#) in the section *Administration*

Invoking Restart Load

▶ **To invoke the restart load function**

- 1 In the **Welcome to the Natural Object Handler** window, from the **Actions** menu, choose **Restart Load**.

The **Restart Load** window appears.

- 2 In local environments, in the field **Restart file**, enter the name of the restart file.

In remote environments, in the field **Restart file**, enter the name of the text member containing the restart data.

Or:

Choose the **Browse** button and select a file from a directory.

Or:

From the drop-down list box, select a file.

Specifying Permanent Members

Applies to remote environments only.

▶ To specify a permanent restart text member

- 1 From the **Actions** menu, choose **Change Workplan Library**.

The **Workplan Library** window appears.

- 2 Select the check box **Use permanent text member for restart data** and enter the name required in the corresponding input field.
- 3 Choose the **OK** command button to confirm your settings.

15 View

▪ Invoking View	118
▪ Terminating View	119
▪ Navigating	119
▪ Saving Object Selections	120
▪ Sorting Objects	120
▪ Listing Objects Separately	120
▪ Deleting Objects	121

This function is used to display objects currently located in your Natural system environment. From the view function you can invoke the **find** function (see the relevant section) to specify further selection criteria for the search.

You can view all objects available on the relevant platform:

In a local Windows environment, or in a remote environment on a Windows, a UNIX or an OpenVMS platform:

- Natural library objects
- Natural system error messages
- Natural command processor sources
- Natural-related objects
- External files
- FDTs

In a remote environment on a mainframe platform:

- Natural library objects
- Natural system error messages
- Natural command processor sources
- Natural DDMs
- Natural-related objects
- FDTs

For information on the table columns and cells that appear in the windows generated by the view function, refer to the section *Object Specification*.

Invoking View

▶ To invoke the view function

- Activate advanced-user mode and, from the **Actions** menu, choose **View**.

The **View** window appears with a list of the types of object available for selection.

Terminating View

▶ **To leave the view function**

- From the **Object** menu, choose **Close**.

Or:

Choose the standard Windows close button.

Navigating

▶ **To navigate between the windows of the view function and to select objects**

- 1 In the initial **View** window and in any subsequent object-specification window, double-click on a list item to go to the next object-specification window.

Or:

Select a list item and, from the **Object** menu, choose **Open** to go to the next specification window.

- 2 From the **Object** menu, choose **Back** to return to a previous object-specification window.

▶ **To view Natural library objects from different system files in a remote environment**

- 1 In the initial **View** window, double-click on the type of object required.

Or:

From the **Object** menu, choose **Open**.

A window appears with a list of specified system files.

- 2 In the **System File** table, double-click on **User-defined System File**.

Or:

Select **user-defined system file** and, from the **Object** menu, choose **Open**.

- 3 If required, in the appropriate text boxes, enter a valid Adabas database ID (**DBID**), file number (**FNR**), password (**Password**) and cipher key (**Cipher key**), and choose the **OK** command button.

Saving Object Selections

▶ To save a list of selected objects as Workplan of the type LIST

- 1 In any of the object selection tables generated by the view function, select the objects required.
- 2 From the **Object** menu, choose **Save Into**.

The **Save into List** window appears.

- 3 In the **Workplan name** text box, enter the name of a new Workplan of the type LIST, and fill the **Workplan description** text box.

Or:

From the drop-down list box, choose a Workplan from a list of all Workplans available.

- 4 Choose the **OK** command button to add the specified objects to the Workplan.

Sorting Objects

▶ To sort a list of selected objects by columns

- In the **View** window, select the entire column by which you want to sort the table and double-click on this column.

Or:

Select the column by which you want to sort the table and, from the context or **Edit** menu, choose **Sort Objects**.

Listing Objects Separately

▶ To list source objects separately from cataloged objects (GPs)

- In the **View** windows, from the **Options** menu, choose **Single Objects**.

Source objects (Src) and cataloged objects (Gp) are listed in separate table rows.

Deleting Objects

▶ **To delete objects**

- 1 In any of the object selection tables generated by the view function, select the objects you want to delete.
- 2 From the **Object** menu, choose **Delete**.

A confirmation box appears.

- 3 Choose the **Yes** command button to execute the deletion.

16 Find

■ Find in Advanced User Mode	124
------------------------------------	-----

This function is used to locate objects in your Natural environment and generate a report list of the objects found. In addition to the [view](#) function (see the relevant section), the find function provides options to specify further criteria for the object selection.

▶ **To invoke the find function**

- Activate advanced-user mode and use the **Find** menu option as described below.

Or:

Use the [Find objects](#) function of the unload wizard as described in the section *Wizard*.

Find in Advanced User Mode

▶ **To invoke the find function in advanced-user mode**

- 1 From the **Actions** menu, choose **Find**.

A window appears where you select one object type:

- Natural library objects
- Natural system error messages
- Natural command processor sources
- DDMs
(remote environments only)
- Natural-related objects
- FDTs
- Use Selection or List

- 2 Choose the **OK** command button.
- 3 Depending on the object type selected, one or more additional windows appear where you can specify selection criteria and option or parameter settings:
 - For the keywords and valid values that apply to each object type, see the relevant explanations in the section [Object Specification](#).
 - For possible settings, see the section [Settings](#).
- 4 After you have made all object specifications and specified the settings, choose the **Find** command button to execute the find function.

The find window appears with an object selection table of all objects available in your current Natural environment that match the object specifications made earlier.

For information on the table columns, refer to the section *Object Specification*.

For the options provided in the find window, see *Table Functions* below.

- 5 To terminate the find function:

From the **Object** menu, choose **Close**.

Or:

Choose the standard Windows close button.

This section covers the following topic:

- [Table Functions](#)

Table Functions

Listed below are the options provided in the object selection table of the find window, along with explanations and instructions on how to invoke them:

Option	Explanation/Instruction
Refresh Table	Rebuilds the table to show the latest status. Choose this function from the context or Object menu.
Details	Invokes the scroll bar for displaying additional table columns. From the Options menu, choose Details so that you can scroll to list further table columns with more information on the objects found.
Single Objects	Lists source objects (Src) and cataloged objects (Gp) in separate table rows. Set this option in the Options menu.
Sort Objects	Sorts the table by columns. Select the entire column by which you want to sort the table and double-click on this column. Alternatively, select the column by which you want to sort the table and, from the context or Edit menu, choose Sort Objects .
Unload	Unloads objects. This function can only be applied, if the unload function of the Object Handler has been activated (see also <i>Advanced User Unload</i> in the section <i>Advanced User</i>). Select one or more objects and, from the context or Object menu, choose Unload .

Option	Explanation/Instruction
Load	Loads objects. This function can only be applied, if the load function of the Object Handler has been activated (see also <i>Advanced User Load</i> in the section <i>Advanced User</i>). Select one or more objects and, from the context or Object menu, choose Load .
Save Into	Saves a list of selected objects as Workplan of the type LIST. Select one or more objects and, from the context or Object menu, choose Save Into .
Delete	Deletes objects. Select one or more objects and, from the context or Object menu, choose Delete .

17 Scan

- Scan in Advanced User Mode 128

This function is used to scan for objects in Natural work files.

The following restrictions apply to the scan function:

- You cannot scan Natural-related objects and external files in remote environments located on mainframe platforms. Natural DDMs can *only* be scanned in remote environments located on mainframe platforms.
- You cannot apply the scan function while you are executing an unload function. If you end the current unload to perform a scan, the unload file will be closed. A subsequent unload then writes the data to a new work file. So, if you do not change the work file name, the existing file will be overwritten.

▶ To invoke the scan function

- Activate advanced-user mode and use the **Scan Work File for** menu option as described below.

Or:

Use the function **Scan objects in Natural work file(s)** of the load wizard as described in the section *Wizards*.

Scan in Advanced User Mode

For information on the table columns and cells that appear in the boxes generated by the scan function, refer to the section *Object Specification*.

▶ To scan objects in advanced-user mode

- 1 From the **Actions** menu, choose **Scan Work File for**.

A window appears where you can select one object type or select all object types:

- All objects
- Natural library objects
- Natural system error messages
- Natural command processor sources
- DDMs
(remote environments only)
- Natural-related objects
- External files
- FDTs

- Use Selection or List
- 2 Choose the **OK** command button.
 - 3 Depending on the object type selected, one or more additional windows appear where you can specify selection criteria and option and parameter settings:
 - For the keywords and valid values that apply to each object type, see the relevant explanations in the section *Object Specification*.
 - For possible settings, see the section *Settings*.
 - 4 After you have made all object specifications and specified the settings, choose the **Scan** command button to execute the function.

The scan window appears with a table of all objects that meet the selection criteria specified and that are contained in the work file.

For information on the table columns, refer to the section *Object Specification*.

For the options provided in the scan window, see *Table Functions* below.

- 5 To terminate the scan function:

From the **Object** menu, choose **Close**.

Or:

Choose the standard Windows close button.

This section covers the following topic:

- *Table Functions*

Table Functions

The following functions are available in the object selection table of the scan window:

Function	Explanation/Instruction
Details	Shows further table columns. From the Options menu, choose the Details button so that you can scroll to list further table columns with more information on the objects found.
Unload	Unloads objects. This function can only be applied, if the unload function of the Object Handler has been activated (see also <i>Advanced User Unload</i> in the section <i>Advanced User</i>). Select one or more objects and, from the context or Object menu, choose Unload .

Function	Explanation/Instruction
Load	Loads objects. This function can only be applied, if the load function of the Object Handler has been activated (see also <i>Advanced User Load</i> in the section <i>Advanced User</i>). Select one or more objects and, from the context or Object menu, choose Load .
Save Into	Saves a list of selected objects as Workplan of the type LIST. Select one or more objects and, from the context or Object menu, choose Save Into .

18 Administration

- Administration Wizard 132
- Advanced User Administration 134
- Change Workplan Library 136

This function is used to maintain Object Handler Workplans.

For information on Workplans and the syntax that applies, refer to the sections [Workplans](#) and [Direct Commands](#).

The Object Handler provides the option to use the administration wizard which determines the processing sequence, or to use the administration function for advanced users.

Note that you can set the default library for Workplans by using the **Workplan-Library** entry of the **Profile** option. See also the section [Profile Settings](#).



Note: The administration wizard offers a restricted set of administration functions that does not provide the option to create, modify, delete, export or import Workplans. To get the full set of administration functions, activate advanced-user mode.

Administration Wizard

The administration wizard provides the **Next** and **Back** command buttons to navigate between the windows (steps). Use the **Cancel** command button to cancel the processing sequence.

▶ To invoke the administration wizard

- In the **Welcome to the Natural Object Handler** window, remove the mark from the **Advanced user** check box if required (not selected by default) and choose the **Administration** button.

The **Administration Wizard** window appears.

Instructions for using the functions provided in the **Administration Wizard** window are explained in the following section:

- [List and Check Workplan](#)
- [Start Object Handler Command Procedure](#)
- [Change Workplan Library](#)

List and Check Workplan

This function is used to list the contents of all Workplans that are available in the Workplan library.

▶ To list and check Workplans

- 1 In the initial **Administration Wizard** window, choose **List and Check Workplan**.

A window appears with the text boxes **Workplan Name** and **Workplan Type**.

- 2 Enter the name of a Workplan.

Or:

From the drop-down list box, choose a name from the list of Workplans available.

- 3 Choose the **Next** command button.

A window appears with the contents of the Workplan specified.

- 4 Choose the **Next** command button.

The Object Handler checks the syntax and displays the result.

Note that this step does not apply to Workplans of the types TEXT and LIST.

- 5 Choose the **Next** command button.

The initial **Administration Wizard** window appears.

Start Object Handler Command Procedure

1. In the initial **Administration Wizard** window, choose **Start Object Handler command procedure**.

A window appears with the **Procedure name** text box.

2. Enter the name of a Workplan of the type PROCEDURE.

Or:

From the drop-down list box, choose a name from the list of all Workplans available.

3. Choose the **Next** command button.

A window appears with the contents of the Workplan specified.

4. Choose the **Next** command button.

The Object Handler executes the command procedure and displays the result.

5. Choose the **Next** command button.

The initial **Administration Wizard** window appears.

Change Workplan Library

For instructions, see the corresponding function *Change Workplan Library* described in *Advanced User Administration*.

Advanced User Administration

▶ **To invoke the administration function in advanced-user mode**

- In the **Welcome to the Natural Object Handler** window, select the **Advanced user** check box and choose the **Administration** button.

The **Administration** window appears with a table of all Workplans available in your Workplan library (see also *Change Workplan Library* below).

The following columns are provided:

Column	Explanation
Name	The name of the Workplan.
Type	The type of Workplan: see <i>Types of Workplan</i> in the section <i>Workplans</i> .
Description	The description of the Workplan.
User	The ID of the user who last modified the Workplan.
Date	The date and time of the last modification.

If the Workplan library does not contain any Workplan, the table is empty.

▶ **To terminate the administration function in advanced-user mode**

- From the **Object** menu, choose **Close**.

Or:

Choose the standard Windows close button.

This section covers the following topic:

- Advanced User Administration Table

Advanced User Administration Table

Listed below are the options provided in the Workplan table, along with explanations and instructions on how to apply them to a Workplan:

Option	Explanation/Instruction
Sort Objects	<p>Sorts Workplans by the columns Name, Type, Description, User or Date.</p> <p>Select the entire column by which you want to sort the table and double-click on this column.</p> <p>Alternatively, select the column by which you want to sort the table and, from the context or Edit menu, choose Sort Objects.</p>
New Workplan	<p>Creates a new Workplan.</p> <p>From the Object menu, select New Workplan and the type of Workplan. Depending on the editing option chosen by selecting or not selecting Free Format Editing from the Options menu, the following applies:</p> <ul style="list-style-type: none"> With Free Format Editing selected (activated), or if you create a Workplan of a type other than OPTION, PARAMETER or SELECTION, a window with an edit area appears. Enter the contents of the Workplan. With Free Format Editing not selected (deactivated; this is the default), windows with input and selection options are provided for Workplans of the type OPTION, PARAMETER or SELECTION. Select the required check boxes and option buttons and fill the text boxes. <p>For information on the syntax used, see the section <i>Direct Commands</i>.</p>
Edit	<p>Modifies an existing Workplan.</p> <p>Select the Workplan required and double-click on it. Alternatively, select the Workplan required and, from the context or Object menu, choose Edit.</p>
Open Workplan	<p>Selects and opens a Workplan from a list of all Workplans available.</p> <p>From the Object menu, choose Open Workplan and, in the Workplan window, enter the name of a Workplan or select a Workplan from the drop-down list box. In the window provided, you can edit the Workplan.</p> <p>For information on the syntax used, see the section <i>Direct Commands</i>.</p>
Delete	<p>Deletes a Workplan.</p> <p>Select the Workplan required and, from the context or Object menu, choose Delete.</p>
Check	<p>Verifies that the correct syntax is used in a Workplan.</p> <p>Select the Workplan required and, from the context or Object menu, choose Check.</p>

Option	Explanation/Instruction
	Note that the Check option does not apply to Workplans of the type TEXT or LIST.
Execute	Executes a Workplan of the type PROCEDURE. Select the Workplan required and, from the context or Object menu, choose Execute .
Import	Imports a Workplan into the file system. Select any Workplan and, from the context or Object menu, choose Import .
Export	Exports a Workplan from the file system. Select one or more Workplans and, from the context or Object menu, and choose Export .

Change Workplan Library

This function is used to change the Workplan library. All Workplans must be stored in a Workplan library, as otherwise Workplans cannot control data processing, such as the function Select OPTION Workplan.

This section covers the following topics:

- [Local Environments](#)
- [Remote Environments](#)

Local Environments

▶ To change the Workplan library in a local environment

- 1 From the **Actions** menu, choose **Change Workplan Library**.

Or:

Using the administration wizard, in the initial **Administration** window, select the option button **Change Workplan library** and then choose the **Next** command button.

A window appears with the following items:

Item	Explanation
Library	The name of the Workplan library. Default is the library WORKPLAN. From the drop-down list box, choose the name of a Workplan library available.
DBID	Specifies the database ID where the Workplan library is located. If no values are specified, the current FUSER or FNAT system file is used.
FNR	Specifies the file number where the Workplan library is located. If no values are specified, the current FUSER or FNAT system file is used.

- 2 Enter or select the data required.
- 3 Choose the **OK** command button.

Or:

Using the administration wizard, choose the **Next** command button.

The Workplan library has been changed, and the window appears from where the function **Change Workplan Library** was invoked (for the administration wizard, this is the initial **Administration** window).

Remote Environments

In remote environments, the function **Change Workplan Library** also provides the option to specify permanent files for reports, traces and restarts of load functions (see also the sections [Tools](#) and [Restart Load](#)).

In remote environments, report, trace and restart data is written to Natural objects (members) of the type Text in the Workplan library. The Object Handler assigns them temporary names and automatically deletes them after two days. When using **Change Workplan Library** with the **Use permanent text member** check box selected, data can be stored in permanent text members that are kept until overwritten by new data or intentionally deleted by the user.

► To change the Workplan library in a remote environment and specify permanent record files

- 1 In the initial **Administration** window, choose the option button **Change Workplan library**.

A window appears with the following items:

Item	Explanation
Library	See Library above.
DBID	See DBID above.
FNR	See FNR above.
Password	The Adabas password of the Adabas file where the Workplan library is located.
Cipher key	The Adabas cipher code of the Adabas file where the Workplan library is located.
Work file text members	To specify a text member for storing report, restart or trace data, select the relevant Use permanent text member check box and enter the name of a Natural object of the type Text in the corresponding text box: For report data, see also Reports in the section Tools . For restart data, see also the section Restart Load . For trace data, see also Traces in the section Tools .

- 2 Enter the data required and confirm your changes by choosing the **OK** command button or by choosing **ENTER**.

19 Object Specification

For the unload, load and scan functions, the Object Handler provides a selection window where you can select the types of object to be processed or specify a Workplan of the type SELECTION or LIST.

For each type of object selected, you are provided object-specification windows. These windows are used to specify selection criteria for the objects to be processed.

This section describes the options provided in the individual object-specification windows.

All Objects

Natural Library Objects

Natural System Error Messages

Natural Command Processor Sources

Natural DDMs

Natural-Related Objects

External Files

FDTs

Use Selection or List

20 Object Specification - All Objects

Applies to the load and scan functions only.

The option **All objects** is used to select all objects available in the work file for processing. For descriptions of keywords and valid input values, see [select-clause](#) in the section *Direct Commands*.

21 Object Specification - Natural Library Objects

- Natural Library Objects 144
- Natural Library Object Details 145
- Natural Library Object Properties 147
- Natural Library Object Exceptions 149
- Natural Library Object Exception Properties 150

This section describes the options provided in the object-specification windows for processing Natural library objects.

Natural library objects are programming objects (including Natural DDMs), user-defined error messages and shared resources.

For descriptions of keywords and valid input values, see also *select-clause* in the section *Direct Commands*.

Natural Library Objects

The specification window for Natural library objects provides the following items:

Item	Explanation
Library	The name of a library or a range of names: see <i>Name</i> in <i>Name, Date and Time Specification</i> . To choose a name from a selection list of all libraries available, open the drop-down list box.
DBID	Only applies to the unload function. The database ID of the system file where the Natural libraries are stored.
FNR	Only applies to the unload function. The file number of the system file where the Natural libraries are stored. If no values (or 0) are specified for DBID and FNR, the current FUSER or FNAT system file is used.
Password	Only applies to the unload function if executed in remote environments located on mainframe platforms. The Adabas password of the system file where the Natural libraries are stored.
Cipher key	Only applies to the unload function if executed in remote environments located on mainframe platforms. The Adabas cipher code of the system file where the Natural libraries are stored.
Name	The name of a Natural programming object or shared resource or a range of names: see <i>Name</i> . The default is an asterisk (*), which selects all objects available. Only evaluated if the check boxes Natural programming objects and/or Shared resources are selected in the details window, which is the default. See also <i>Natural Library Object Details</i> .
Message from/to	A valid range (1 - 9999) of user-defined error messages delimited by the first and the last message number. Only evaluated if the Error messages check box is selected in the details window, which is the default. See also <i>Natural Library Object Details</i> .

Item	Explanation
Details	Invokes an additional window where you can enter more detailed object specifications: see Natural Library Object Details .
Settings	Only applies to functions executed in advanced-user mode. Invokes the Settings window where you can specify option and parameter settings: see the section Settings .
Unload file or Load file or Scan file (Server)	Only applies to functions executed in advanced-user mode. The name of the work file to be used for the function. See also the section Work Files .
Browse	Only applies to functions executed in advanced-user mode. Invokes the browse function to select a work file from a directory.

Natural Library Object Details

The details window for Natural library objects is used to specify further selection criteria for Natural library objects.

For descriptions of keywords and valid input values, see also [select-clause](#) in the section *Direct Commands*.

The details window for Natural library objects provides the following items:

Item	Explanation
Library	The name of a library or a range of names: see Name in <i>Name, Date and Time Specification</i> . To choose a name from a selection list of all libraries available, open the drop-down list box. Ranges are not allowed if the Predict set option is selected.
DBID	See DBID in <i>Natural Library Objects</i> above.
FNR	See FNR in <i>Natural Library Objects</i> above.
Password	See Password in <i>Natural Library Objects</i> above.
Cipher key	See Cipher key in <i>Natural Library Objects</i> above.
Natural programming objects	Natural programming objects. Natural DDMs (data definition modules): In remote environments located on mainframe platforms, DDMs are located in the FDIC system file. They are not

Item	Explanation
	<p>considered Natural library objects. In other system environments, DDMs are considered Natural programming objects, which are stored in Natural libraries.</p> <p>If the FDDM system file has been activated, see also Use FDDM file for processing DDMs in <i>Settings - Options</i>.</p>
Error messages	User-defined error messages.
Shared resources	<p>Any non-Natural file that is used in a Natural environment and is maintained in the Natural library system.</p> <p>Note that shared resources are not defined in remote environments located in mainframe systems.</p>
Name	See Name in <i>Natural Library Objects</i> above.
S/C-Kind	<p>The kind of Natural programming object:</p> <p>Src Source objects only.</p> <p>Gp Generated programs (cataloged objects) only.</p> <p>Any All source and/or cataloged objects (generated programs). This is the default.</p> <p>Stowed All STOWed objects: source and cataloged objects with identical date and time.</p> <p>Both Both source and cataloged objects if both exist.</p> <p>Note: Stowed and Both are valid for the unload function only.</p>
Predict set	<p>Only applies to the unload and find functions and if Predict is installed.</p> <p>This option is used to read the names of the objects to be processed from a retained set. A retained set is created with the save set option of the LIST XREF command.</p> <p>If the Predict set option is selected, the following applies:</p> <ul style="list-style-type: none"> ■ The Name text box must contain asterisk (*) indicating all objects. This is the default setting. ■ The Library list box must contain the name of a single library. Name ranges are not allowed. ■ The Set number box must be filled. <p>For detailed information on Predict sets, refer to the Predict documentation.</p>
Set number	<p>Only applies if Predict set is selected.</p> <p>A one- or two-digit number that identifies the retained set to be used.</p>
Set library	<p>Only applies if Predict set is selected.</p> <p>The name of the library to be searched for a Predict set. If you do not specify a name, the library entered in the Library list box is used by default.</p>

Item	Explanation
Set user	Only applies if Predict set is selected. The ID of the user who created the retained set. If no ID is entered, the ID specified with the system variable *USER (see the <i>System Variables</i> documentation) is used.
Natural Object Types	The types of Natural programming object.
Select All	Selects all types of Natural programming object (this is the default).
Deselect All	Deselect all types of Natural programming object.
User-defined Messages: from/to	A range of user-defined error messages as entered in the Message from/to boxes (see <i>Natural Library Objects</i> above).
User-defined Messages: S/L-Kind	The kind of user-defined error message text: Short Short text. Long Long text. Any Short and/or long text. This is the default. Both Short and long texts if both exist (unload function only).
User-defined Messages: Language codes	Up to 8 valid language codes (for example, code 1 for English) of the specified error messages. An asterisk (*) selects all language codes.
Properties	Invokes an extra window where you can specify additional properties of Natural programming objects: see <i>Natural Library Object Properties</i> below. Once you have specified any properties, you can activate them by selecting the check box to the left of the Properties button, or deactivate them by removing the mark from the check box.
Exceptions	Invokes an extra window where you can specify exceptions to the selection of Natural programming objects: see <i>Natural Library Object Exceptions</i> below. Once you have specified any exceptions, you can activate them by selecting the check box to the left of the Exceptions button, or deactivate them by removing the mark from the check box.

Natural Library Object Properties

The **Properties** window for Natural library objects is used to specify properties for the Natural library objects selected for processing.

For descriptions of keywords and valid input values, see also *select-clause* in the section *Direct Commands*.

The **Properties** window for Natural library objects provides the following items:

Item	Explanation								
User ID	The ID of the user who last saved the object. Specify a single user ID or a range of user IDs: see <i>Name</i> in <i>Name, Date and Time Specification</i> .								
Natural version	The Natural version of the Natural programming objects. You can also specify a range of versions: see <i>Name</i> .								
Mode	<p>The programming mode of the Natural programming objects:</p> <table border="1" data-bbox="496 449 1380 669"> <tr> <td data-bbox="496 449 850 491"></td> <td data-bbox="850 449 1380 491"></td> </tr> <tr> <td data-bbox="496 491 850 533">Structured</td> <td data-bbox="850 491 1380 533">Structured mode only.</td> </tr> <tr> <td data-bbox="496 533 850 575">Report</td> <td data-bbox="850 533 1380 575">Reporting mode only.</td> </tr> <tr> <td data-bbox="496 575 850 669">Any</td> <td data-bbox="850 575 1380 669">No mode check performed. This is the default.</td> </tr> </table>			Structured	Structured mode only.	Report	Reporting mode only.	Any	No mode check performed. This is the default.
Structured	Structured mode only.								
Report	Reporting mode only.								
Any	No mode check performed. This is the default.								
DDM DBID	<p>Not valid in remote environments located on mainframe platforms.</p> <p>The database ID (DBID) of the data definition modules (DDMs). Valid entries are: 1 to 65535 or 0 (all DBIDs)</p>								
DDM FNR	<p>Not valid in remote environments located on mainframe platforms.</p> <p>The file number (FNR) of the DDMs: Valid entries are: 1 to 65535 or 0 (all FNRs).</p>								
Date: Select all objects	Selects all objects, regardless of their date.								
Date: Select objects modified between/and	<p>Selects all objects with a save or catalog date and/or time within the range specified in the Date boxes.</p> <p>Select a value from the combo box or type values for a start date and/or time and/or values for an end date and/or time. For valid input values, see <i>Date</i> and <i>Time</i> in <i>Name, Date and Time Specification</i>. Special dates allowed are: TODAY, YESTERDAY, MONTH and YEAR.</p>								
Date: Select objects modified on	<p>Selects all objects with a save or catalog date and/or time that fits the date/time specified in the Date box.</p> <p>Select or type values for a date and/or time. For valid input values, see <i>Date</i> and <i>Time</i>. Special dates allowed are: TODAY and YESTERDAY.</p>								
Size: Select all objects	Selects all objects, regardless of their size.								
Size: Select objects with size between/and	Selects all objects with a size within the range specified in the Size boxes: type a value for a start size and/or an end size.								
Size: Select objects with size	Selects all objects with a size that fits the size specified in the Size box.								

Natural Library Object Exceptions

The **Exceptions** window for Natural library objects is used to specify exceptions to the selection of Natural library objects.

All objects that match the selection criteria specified in [Natural Library Objects](#), [Natural Library Object Details](#) and [Natural Library Object Properties](#) are checked against the specifications made in the **Exceptions** window. Objects that match *all* specifications defined as exceptions, are exempted from processing.

For descriptions of keywords and valid input values, see also [select-clause](#) in the section *Direct Commands*.

The **Exceptions** window for Natural library objects is basically identical to the [details](#) window. For explanations of the items listed in the table below, see the relevant section. The **Properties** button is used to specify additional properties for Natural programming object exceptions: see [Natural Library Object Exception Properties](#) below.

Item
Location: Library
Object Types: Natural programming objects Error messages Shared resources
Natural Programming Objects and Shared Resources: Name
Natural Programming Objects: S/C-Kind Natural Object Types
Natural System Error Messages: from/to Language codes S/L-Kind
Extras: Properties

Natural Library Object Exception Properties

The **Exception Properties** window for Natural library objects is used to specify exceptions to the properties of the Natural library objects selected for processing.

For descriptions of keywords and valid input values, see also *select-clause* in the section *Direct Commands*.

The **Exception Properties** window for Natural library objects provides the following items:

Item	Explanation
User ID	See User ID in <i>Natural Library Object Properties</i> .
Natural version	See Natural version in <i>Natural Library Object Properties</i> .
Mode	See Mode in <i>Natural Library Object Properties</i> .
DDM DBID	See DDM DBID in <i>Natural Library Object Properties</i> .
DDM FNR	See DDM FNR in <i>Natural Library Object Properties</i> .
Date: Ignore date	Performs no date check. Objects are processed, regardless of their date.
Date: Exclude objects modified between/and	Exempts from processing all objects with a save or catalog date and/or time within the range specified in the Date boxes. Select or type values for a start date and/or time and/or values for an end date and/or time. For valid input values, see <i>Date</i> and <i>Time</i> in <i>Name, Date and Time Specification</i> . Special dates allowed are: TODAY, YESTERDAY, MONTH and YEAR.
Date: Exclude objects modified on	Exempts from processing all objects with a save or catalog date and/or time that fits the date/time specified in the Date box. Select or type values for a date and/or time. For valid input values, see <i>Date</i> and <i>Time</i> . Special dates allowed are: TODAY and YESTERDAY.
Size: Ignore size	Performs no size check. Objects are processed, regardless of their size.
Size: Exclude objects with size between/and	Exempts from processing all objects with a size within the range specified in the Size boxes: type a value for a start size and/or an end size.
Size: Exclude objects with size	Exempts from processing all objects with a size that fits the size specified in the Size box.

22

Object Specification - Natural System Error Messages

- Natural System Error Messages 152
- Natural System Error Message Details 153
- Natural System Error Message Exceptions 153

This section describes the options provided in the object-specification windows for processing Natural system error messages.

For descriptions of keywords and valid input values, see also *select-clause* in the section *Direct Commands*.

Natural System Error Messages

The specification window for Natural system error messages provides the following items:

Item	Explanation
Number from/to	The range of Natural system error messages delimited by the first and the last message number.
DBID	Only applies to the unload function if executed in remote environments located on mainframe platforms. The database ID of the system file where the Natural system error messages are stored.
FNR	Only applies to the unload function if executed in remote environments located on mainframe platforms. The file number of the system file where the Natural system error messages are stored. If no values (or 0) are specified for DBID and FNR , the current FNAT system file is used.
Password	Only applies to the unload function if executed in remote environments located on mainframe platforms. The Adabas password of the system file where the Natural system error messages are stored.
Cipher key	Only applies to the unload function if executed in remote environments located on mainframe platforms. The Adabas cipher code of the system file where the Natural system error messages are stored.
Details	Invokes the details window where you can enter more detailed object specifications: see Natural System Error Message Details below.
Settings	Only applies to functions executed in advanced-user mode. Invokes the Settings window where you can specify option and parameter settings. See Settings .
Unload file or Load file or Scan file (Server)	Only applies to functions executed in advanced-user mode. The name of the work file to be used for the function. See also Work Files .

Item	Explanation
Browse	Only applies to functions executed in advanced-user mode. Invokes the browse function to select a work file from a directory.

Natural System Error Message Details

The details window for Natural system error messages is used to specify further selection criteria for Natural system error messages.

For descriptions of keywords and valid input values, see also *select-clause* in the section *Direct Commands*.

The details window for Natural system error messages provides the following items:

Item	Explanation
Number from/to	See Number from/to in <i>Natural System Error Messages</i> above.
DBID	See DBID in <i>Natural System Error Messages</i> above.
FNR	See FNR in <i>Natural System Error Messages</i> above.
Password	See Password in <i>Natural System Error Messages</i> above.
Cipher key	See Cipher key in <i>Natural System Error Messages</i> above.
S/L-Kind	See S/L-Kind in <i>Natural Library Object Details</i> .
Language codes	See Language codes in <i>Natural Library Object Details</i> .
Exceptions	Invokes an extra window where you can specify exceptions to the selection of Natural system error messages: see <i>Natural System Error Message Exceptions</i> . Once you have specified any exceptions, you can activate them by selecting the check box to the left of the Exceptions button, or deactivate them by removing the mark from the check box.

Natural System Error Message Exceptions

The **Exceptions** window for Natural system error messages is used to specify exceptions to the selection of Natural system error messages.

All Natural system error messages that match the selection criteria specified in *Natural System Error Messages* and *Natural System Error Message Details* are checked against the specifications made in the **Exceptions** window. Error messages that match *all* specifications defined as exceptions, are exempted from processing.

For explanations of the items contained in the **Exceptions** window, see [Natural System Error Message Details](#) above.

For descriptions of keywords and valid input values, see also [select-clause](#) in the section *Direct Commands*.

23

Object Specification - Natural Command Processors

- Natural Command Processor Sources 156
- Natural Command Processor Source Exceptions 157

This section describes the options provided in the object-specification windows for processing Natural command processor sources. Natural command processor sources are stored in Adabas files.

For descriptions of keywords and valid input values, see also *select-clause* in the section *Direct Commands*.

Natural Command Processor Sources

The specification window for Natural command processor sources provides the following items:

Item	Explanation
Library	The name of a Natural command processor library or a range of names: see <i>Name</i> in <i>Name, Date and Time Specification</i> .
DBID	Only applies to the unload function. The database ID where the Natural command processor sources are stored. For details, see the <i>SYSNCP Utility</i> in the <i>Utilities</i> documentation.
FNR	Only applies to the unload function. The file number of the Adabas file where the Natural command processor sources are stored. If no values are specified, the current setting of LFILE 190 is used. For details, see the <i>SYSNCP Utility</i> in the <i>Utilities</i> documentation.
Password	Only applies to the unload function. The Adabas password of the Adabas file where the Natural command processor sources are stored.
Cipher key	Only applies to the unload function. The Adabas cipher code of the Adabas file where the Natural command processor sources are stored.
Name	The name of a Natural command processor source or a range of names: see <i>Name</i> .
Exceptions	Invokes the Exceptions window where you can specify exceptions to the selection of Natural command processor sources: see <i>Natural Command Processor Source Exceptions</i> below. Once you have specified any exceptions, you can activate them by selecting the check box to the left of the Exceptions button, or deactivate them by removing the mark from the check box.
Settings	Only applies to functions executed in advanced-user mode. Invokes the Settings window where you can specify option and parameter settings. See <i>Settings</i> .

Item	Explanation
Unload file or Load file or Scan file (Server)	Only applies to functions executed in advanced-user mode. The name of the work file to be used for the function. See also <i>Work Files</i> .
Browse	Only applies to functions executed in advanced-user mode. Invokes the browse function to select a work file from a directory.

Natural Command Processor Source Exceptions

The **Exceptions** window for Natural command processor sources is used to specify exceptions to the selection of Natural command processor sources.

All objects that match the selection criteria specified in *Natural Command Processor Sources* are checked against the specifications made in the **Exceptions** window. Natural command processor sources that match *all* specifications defined as exceptions, are exempted from processing.

For descriptions of keywords and valid input values, see also *select-clause* in the section *Direct Commands*.

The **Exceptions** window for Natural command processor sources provides the following items:

Item	Explanation
Library	The name of a Natural command processor library or a range of names: see <i>Name</i> in <i>Name, Date and Time Specification</i> .
Name	The name of a Natural command processor source or a range of names: see <i>Name</i> .

24 Object Specification - Natural DDMs

- Natural DDMs 160
- Natural DDM Details 161
- Natural DDM Exceptions 162

Only applicable to remote environments located on mainframe platforms.

This section describes the options provided in the object-specification windows for processing Natural DDMs (data definition modules).

On mainframe platforms, DDMs are stored in the FDIC system file. The DDMs to be processed by the Object Handler are located in the default FDIC file. If you want to specify a different FDIC file, use the *option-setting* clause described in the section *Direct Commands*.

For descriptions of keywords and valid input values, see also the *select-clause* described in the section *Direct Commands*.

Natural DDMs

The specification window for Natural DDMs provides the following items:

Item	Explanation
Name	The name of a DDM or a range of names: see <i>Name</i> in <i>Name, Date and Time Specification</i> .
Details	Invokes an additional window where you can enter further object specifications: see <i>Natural DDM Details</i> .
Display FDIC	This field is available in advanced-user mode only. Displays the current FDIC setting.
Set FDIC	This field is available with the load or unload wizard only. Provides a window where you can modify the current FDIC setting.
Settings	Only applies to functions executed in advanced-user mode. Invokes the Settings window where you can specify option and parameter settings. See <i>Settings</i> .
Unload file or Load file or Scan file (Server)	Only applies to functions executed in advanced-user mode. The name of the work file to be used for the function. See also <i>Work Files</i> .
Browse	Only applies to functions executed in advanced-user mode. Invokes the browse function to select a work file from a directory.

Natural DDM Details

The details window for Natural DDMs is used to specify further selection criteria for Natural DDMs.

For descriptions of keywords and valid input values, see also *select-clause* in the section *Direct Commands*.

The details window for Natural DDMs provides the following items:

Item	Explanation
Name	The name of the DDM or a range of names: see <i>Name</i> in <i>Name, Date and Time Specification</i> .
DDM DBID	See DDM DBID in <i>Natural Library Object Properties</i> .
DDM FNR	See DDM FNR in <i>Natural Library Object Properties</i> .
Date: Select all objects	Selects all objects, regardless of their date.
Date: Select objects modified between/and	See Date in <i>Natural Library Object Properties</i> .
Date: Select objects modified on	See Date in <i>Natural Library Object Properties</i> .
Size: Select all objects	Selects all objects, regardless of their size.
Size: Select objects with size between/and	Selects all objects with a size within the range specified in the Size boxes: type a value for a start size and/or an end size.
Size: Select objects with size	Selects all objects with a size that fits the size specified in the Size box.
Exceptions	<p>Invokes an extra window where you can specify exceptions to the selection of Natural DDMs: see <i>Natural DDM Exceptions</i>.</p> <p>Once you have specified any exceptions, you can activate them by selecting the check box to the left of the Exceptions button, or deactivate them by removing the mark from the check box.</p>

Natural DDM Exceptions

The **Exceptions** window for Natural DDMs is used to specify exceptions to the selection of Natural DDMs.

All objects that match the selection criteria specified in *Natural DDMs* and *Natural DDM Details* are checked against the specifications made in the **Exceptions** window. DDMs that match *all* specifications defined as exceptions, are exempted from processing.

For descriptions of keywords and valid input values, see also *select-clause* in the section *Direct Commands*.

The **Exceptions** window for Natural DDMs provides the following items:

Item	Explanation
Name	The name of a DDM or a range of names: see <i>Name</i> in <i>Name, Date and Time Specification</i> .
User ID	See User ID in <i>Natural Library Object Properties</i> .
DDM DBID	See DDM DBID in <i>Natural Library Object Properties</i> .
DDM FNR	See DDM FNR in <i>Natural Library Object Properties</i> .
Date: Ignore date	Performs no date check. Objects are processed, regardless of their date.
Date: Exclude objects modified between/and	See Date in <i>Natural Library Object Exception Properties</i> .
Date: Exclude objects modified on	See Date in <i>Natural Library Object Exception Properties</i> .
Size: Ignore size	Performs no size check. Objects are processed, regardless of their size.
Size: Exclude objects with size between/and	Exempts from processing all objects with a size within the range specified in the Size boxes: type a value for a start size and/or an end size.
Size: Exclude objects with size	Exempts from processing all objects with a size that fits the size specified in the Size box.

25

Object Specification - Natural-Related Objects

- Natural-Related Objects - Windows, UNIX and OpenVMS 164
- Natural-Related Objects - Mainframes 167

This section describes the options provided in the object-specification windows for processing Natural-related objects located on Windows, UNIX, OpenVMS or mainframe platforms. On Windows, UNIX and OpenVMS platforms, Natural-related objects are objects that exist in a Natural environment but are not located in Natural libraries, such as the Natural parameter module NATPARM, which is located in Natural path *PARM_PATH*. On mainframe platforms, Natural-related objects are profiles, debug environments and DL/I subfiles.

Note that not all of the fields may appear in the object-specification windows because they depend on the object location, the version of the Natural Development Server installed, the type of object selected and the function used.

Process Natural-related objects in internal format, that is, do *not* select the **Transfer format** check box. See also [Work File Format](#) in the section *Work Files*.

For descriptions of keywords and valid input values, see also [select-clause](#) in the section *Direct Commands*.

Natural-Related Objects - Windows, UNIX and OpenVMS

The specification window for Natural-related objects located on Windows, UNIX or OpenVMS platforms provides the following items:

Item	Explanation
Natural path	Enter the name of the path where the Natural-related object is located or select a path from the drop-down list box. Valid path names are: <i>NATDIR, NATVERS, NATBIN, NATERR, NATSAG, PARM_PATH, PROFILE_PATH, TEXT_PATH, TMP_PATH.</i> Load and scan: Enter the name of a path or asterisk (*) to select all paths.
Object name	The name of a Natural-related object. Load and scan: A single name or a range of names: see Name in <i>Name, Date and Time Specification</i> .
Select	Invokes the browse function to select a Natural-related object from a directory.
Details	Invokes the details window where you can enter further object specifications: see Natural-Related Object Details - Windows, UNIX and OpenVMS .
Settings	Only applies to functions executed in advanced-user mode. Invokes the Settings window where you can specify option and parameter settings. See Settings .

Item	Explanation
Unload file or Load file or Scan file (Server)	Only applies to functions executed in advanced-user mode. The name of the work file to be used for the function. See also Work Files .
Browse	Only applies to functions executed in advanced-user mode. Invokes the browse function to select a work file from a directory.

This section covers the following topics:

- [Natural-Related Object Details - Windows, UNIX and OpenVMS](#)
- [Natural-Related Object Exceptions - Windows, UNIX and OpenVMS](#)

Natural-Related Object Details - Windows, UNIX and OpenVMS

The details window is used to specify further selection criteria for Natural-related objects located on Windows, UNIX or OpenVMS platforms.

For descriptions of keywords and valid input values, see also [select-clause](#) in the section *Direct Commands*.

The details window for Natural-related objects located on Windows, UNIX or OpenVMS platforms provides the following items:

Item	Explanation
Natural path	See Natural path in <i>Natural-Related Objects - Windows, UNIX and OpenVMS</i> above.
Object name	See Object name in <i>Natural-Related Objects - Windows, UNIX and OpenVMS</i> above.
Select	Invokes the browse function to select an object from a directory.
Date:	Selects all objects, regardless of their date.
Select all objects	
Date:	See Date in <i>Natural Library Object Properties</i> .
Select objects modified between/and	
Date:	See Date in <i>Natural Library Object Properties</i> .
Select objects modified on	
Size:	Selects all objects, regardless of their size.

Item	Explanation
Select all objects	
Size: Select objects with size between/and	Selects all objects with a size within the range specified in the Size boxes: type a value for a start size and/or an end size.
Size: Select objects with size	Selects all objects with a size that fits the size specified in the Size box.
Exceptions	<p>Invokes an extra window where you can specify exceptions to the selection of Natural-related objects: see Natural-Related Object Exceptions - Windows, UNIX and OpenVMS.</p> <p>Once you have specified any exceptions, you can activate them by selecting the check box to the left of the Exceptions button, or deactivate them by removing the mark from the check box.</p>

Natural-Related Object Exceptions - Windows, UNIX and OpenVMS

The **Exceptions** window is used to specify exceptions to the selection of Natural-related objects located on Windows, UNIX or OpenVMS platforms.

All Natural-related objects that match the selection criteria specified in [Natural-Related Objects - Windows, UNIX and OpenVMS](#) and [Natural-Related Object Details - Windows, UNIX and OpenVMS](#) are checked against the specifications made in the **Exceptions** window for Natural-related objects. Objects that match *all* specifications defined as exceptions, are exempted from processing.

For descriptions of keywords and valid input values, see also [select-clause](#) in the section *Direct Commands*.

The **Exceptions** window for Natural-related objects located on Windows, UNIX or OpenVMS platforms provides the following items:

Item	Explanation
Natural path	See Natural path in Natural-Related Objects - Windows, UNIX and OpenVMS above.
Object name	See Object name in Natural-Related Objects - Windows, UNIX and OpenVMS above.
Select	Invokes the browse function to select an object from a directory.
Date: Ignore date	Performs no date check. Objects are processed, regardless of their date.
Date: Exclude objects modified between/and	See Date in Natural Library Object Exception Properties .

Item	Explanation
Date: Exclude objects modified on	See Date in <i>Natural Library Object Exception Properties</i> .
Size: Ignore size	Performs no size check. Objects are processed, regardless of their size.
Size: Exclude objects with size between/and	Exempts from processing all objects with a size within the range specified in the Size boxes: type a value for a start size and/or an end size.
Size: Exclude objects with size	Exempts from processing all objects with a size that fits the size specified in the Size box.

Natural-Related Objects - Mainframes

The specification window for Natural-related objects located on mainframe platforms provides the following items:

Item	Explanation
Object type	The type of object to be processed: Profile Debug environment DL/I subfile
DBID FNR Password Cipher key	Only applies to the unload function. The database ID (DBID), file number (FNR), password and cipher code (cipher key) of the Adabas file where the objects are stored. If no values (or 0) are specified, the current FNAT system file is used.
Object name	The name of the object. Load and scan: A single name or a range of names: see Name in <i>Name, Date and Time Specification</i> .
Details	Invokes the details window where you can enter further object specifications: see Natural-Related Object Details - Mainframes .
Settings	Only applies to functions executed in advanced-user mode. Invokes the Settings window where you can specify option and parameter settings. See Settings .
Unload file or Load file	Only applies to functions executed in advanced-user mode. The name of the work file to be used for the function.

Item	Explanation
or Scan file (Server)	See also Work Files .
Browse	Only applies to functions executed in advanced-user mode. Invokes the browse function to select a work file from a directory.

This section covers the following topics:

- [Natural-Related Object Details - Mainframes](#)
- [Natural-Related Object Exceptions - Mainframes](#)

Natural-Related Object Details - Mainframes

The details window is used to specify further selection criteria for Natural-related objects located on mainframe platforms.

For descriptions of keywords and valid input values, see also [select-clause](#) in the section *Direct Commands*.

The details window for Natural-related objects located on mainframe platforms provides the following items:

Item	Explanation
Object type	See Object type in <i>Natural-Related Objects - Mainframes</i> above.
DBID FNR Password Cipher key	See DBID/FNR in <i>Natural-Related Objects - Mainframes</i> above.
Object name	See Object name in <i>Natural-Related Objects - Mainframes</i> above.
Subtype	Applies to profiles and DL/I subfiles. The type(s) of profile or the type(s) of DL/I subfile to be processed: Device profile Editor profile Map profile Parameter profile NSB subfile NDB subfile
Library	Applies to debug environments. The name of a library or a range of names: see Name in <i>Name, Date and Time Specification</i> .

Item	Explanation
	To choose a name from a selection list of all libraries available, open the drop-down list box.
Exceptions	<p>Invokes an extra window where you can specify exceptions to the selection of Natural-related objects: see Natural-Related Object Exceptions - Mainframes.</p> <p>Once you have specified any exceptions, you can activate them by selecting the check box to the left of the Exceptions button, or deactivate them by removing the mark from the check box.</p>

Natural-Related Object Exceptions - Mainframes

The **Exceptions** window is used to specify exceptions to the selection of Natural-related objects located on mainframe platforms.

All Natural-related objects that match the selection criteria specified in [Natural-Related Objects - Mainframes](#) and [Natural-Related Object Details - Mainframes](#) are checked against the specifications made in the **Exceptions** window for Natural-related objects on mainframes. Objects that match *all* specifications defined as exceptions, are exempted from processing.

For descriptions of keywords and valid input values, see also [select-clause](#) in the section *Direct Commands*.

The **Exceptions** window for Natural-related objects located on mainframe platforms provides the following items:

Item	Explanation
Object type	See Object type in Natural-Related Objects - Mainframes above.
DBID FNR Password Cipher key	See DBID/FNR in Natural-Related Objects - Mainframes above.
Object name	See Object name in Natural-Related Objects - Mainframes above.
Subtype	See Subtype in Natural-Related Object Details - Mainframes above.
Library	See Library in Natural-Related Object Details - Mainframes above.

26 Object Specification - External Files

- External Files 172
- External File Details 173
- External File Exceptions 174

Not applicable to remote environments located on mainframe platforms.

This section describes the options provided in the object-specification windows for processing external files. External files are files that are located outside Natural and Adabas environments, such as bitmaps.

Process external files in internal format, that is, do *not* select the **Transfer format** check box. See also [Work File Format](#) in the section *Work Files*.

For descriptions of keywords and valid input values, see also [select-clause](#) in the section *Direct Commands*.

External Files

The specification window for external files provides the following items:

Item	Explanation
External path	Enter the name of the path where the external file is located or select a path from the drop-down list box. Load and scan: Enter the name of a path or asterisk (*) to select all paths.
Object name	The name of an external file. Load and scan: A single name or a range of names: see Name in <i>Name, Date and Time Specification</i> .
Select	Invokes the browse function to select an object from a directory.
Details	Invokes the details window where you can enter further object specifications: see External File Details .
Settings	Only applies to functions executed in advanced-user mode. Invokes the Settings window where you can specify option and parameter settings. See Settings .
Unload file or Load file or Scan file (Server)	Only applies to functions executed in advanced-user mode. The name of the work file to be used for the function. See also Work Files .
Browse	Only applies to functions executed in advanced-user mode. Invokes the browse function to select a work file from a directory.

External File Details

The details window for external files is used to specify further selection criteria for external files.

For descriptions of keywords and valid input values, see also *select-clause* in the section *Direct Commands*.

The details window for external files provides the following items:

Item	Explanation
External path	See External path in <i>External Files</i> above.
Object name	See Object name in <i>External Files</i> above.
Select	Invokes the browse function to select an object from a directory.
Date: Select all objects	Selects all objects, regardless of their date.
Date: Select objects modified between/and	See Date in <i>Natural Library Object Properties</i> .
Date: Select objects modified on	See Date in <i>Natural Library Object Properties</i> .
Size: Select all objects	Selects all objects, regardless of their size.
Size: Select objects with size between/and	Selects all objects with a size within the range specified in the Size boxes: type a value for a start size and/or an end size.
Size: Select objects with size	Selects all objects with a size that fits the size specified in the Size box.
Exceptions	Invokes an extra window where you can specify exceptions to the selection of external files: see <i>External File Exceptions</i> . Once you have specified any exceptions, you can activate them by selecting the check box to the left of the Exceptions button, or deactivate them by removing the mark from the check box.

External File Exceptions

The **Exceptions** window for external files is used to specify exceptions to the selection of external files.

All external files that match the selection criteria specified in *External Files* and *External File Details* are checked against the specifications made in the **Exceptions** window. Objects that match *all* specifications defined as exceptions, are exempted from processing.

For descriptions of keywords and valid input values, see also *select-clause* in the section *Direct Commands*.

The **Exceptions** window for external files provides the following items:

Item	Explanation
External path	See External path in <i>External Files</i> above.
Object name	See Object name in <i>External Files</i> above.
Select	Invokes the browse function to select an object from a directory.
Date: Ignore date	Performs no date check. Objects are processed, regardless of their date.
Date: Exclude objects modified between/and	See Date in <i>Natural Library Object Exception Properties</i> .
Date: Exclude objects modified on	See Date in <i>Natural Library Object Exception Properties</i> .
Size: Ignore size	Performs no size check. Objects are processed, regardless of their size.
Size: Exclude objects with size between/and	Exempts from processing all objects with a size within the range specified in the Size boxes: type a value for a start size and/or an end size.
Size: Exclude objects with size	Exempts from processing all objects with a size that fits the size specified in the Size box.

27 Object Specification - FDTs

The specification window for FDTs is used to select Adabas FDTs (Field Definition Tables) for processing.



Note: When loading FDTs, all FDT data is written to Work File 5. You can use the contents of this work file as input for the Adabas utility ADAFDU. To select another work file, choose **Settings** from the **Options** menu, and, in the **Additional Options** window, on the tabbed page **Special**, specify the ADAFDU file.

For descriptions of keywords and valid input values, see also *select-clause* in the section *Direct Commands*.

The specification window for FDTs provides the following items:

Item	Explanation
DBID	The database ID where the FDT is located. Load and scan: A valid DBID or 0 for all DBIDs.
FNR	The file number where the FDT is located. Load and scan: A valid FNR or 0 for all FDTs.
Password	The Adabas password of the Adabas file where the FDT is located.
Cipher key	The Adabas cipher code of the Adabas file where the FDT is located.
Settings	Only applies to functions executed in advanced-user mode. Invokes the Settings window where you can specify option and parameter settings. See Settings .
Unload file or Load file	Only applies to functions executed in advanced-user mode. The name of the work file to be used for the function.

Item	Explanation
or Scan file (Server)	See also Work Files .
Browse	Only applies to functions executed in advanced-user mode. Invokes the browse function to select a work file from a directory.

28

Use Selection or List

This option is used to specify a Workplan of the type SELECTION or LIST. These Workplans specify selection criteria for the objects to be processed. See also the section [Workplans](#).

The specification window provides the following items:

Item	Explanation
Name	The name of the Workplan to be processed. To choose a name from a selection list of Workplans available, open the drop-down list box.
Settings	Only applies to functions executed in advanced-user mode. Invokes the Settings window where you can specify option and parameter settings. See Settings .
Unload file or Load file or Scan file (Server)	Only applies to functions executed in advanced-user mode. The name of the work file to be used for the function. See also Work Files .
Browse	Only applies to functions executed in advanced-user mode. Invokes the browse function to select a work file from a directory.

29 Settings

The settings option is used to specify option settings for the unload, load, find or scan function and parameter settings for the unload or load function.

▶ **To invoke the Settings window**

- In the **Welcome to the Natural Object Handler** window, from the **Options** menu, choose **Settings**.

Or:

In advanced-user mode, during the unload or load function, from the **Options** menu, choose **Settings**.

Or:

In advanced-user mode, during the unload or load function, in an object-specification window, choose the **Settings** button.

The **Settings** window appears with the tabs **Options** and **Parameters**.

This section describes the items contained on the pages that belong to the tabs **Options** and **Parameters** of the **Settings** window and associated windows and window tabs:

[Settings - Options](#)

[Settings - Parameters](#)

30 Settings - Options

- Set Additional Options 183

On the tabbed page **Options** of the **Settings** window you can specify the following:

Item	Explanation
Transfer format	<p>Only activated if Use default options (this is the default) or Use additional options has been selected. See below.</p> <p>Unload: The data to be unloaded is written in Transfer format to the work file.</p> <p>Load and scan: The data to be loaded or scanned are expected to be in Transfer format.</p>
Local work file	<p>Only applies to remote environments.</p> <p>Specifies the location of the work file when using Object Handler functions in connection with SpoD (Single Point of Development). If Local work file is selected, depending on the function used, the data to be processed is written to or read from the work file specified in the local file system.</p> <p>See also WFLOC in <i>Direct Commands</i>.</p>
Portable work file	<p>Not applicable to work files located in a remote environment on a mainframe platform.</p> <p>In addition, this option is not required for the load and scan functions, which automatically choose the appropriate work file type and ignore the option if set.</p> <p>Portable work file is only activated if the following applies:</p> <ul style="list-style-type: none"> ■ Use default options (this is the default) or Use additional options has been selected (see below) and ■ Transfer format has <i>not</i> been selected. <p>If Portable work file has been selected, the work file is written or read in portable format. See also Work File Format in <i>Work Files</i>.</p>
Fixed length	<p>Only available with the unload function.</p> <p>See FIXEDLENGTH in <i>Direct Commands</i>.</p>
Unicode work file	<p>Only applies to the unload function and if Transfer format has been selected.</p> <p>If this option is selected, all object sources are converted to Unicode/UTF-8 (Universal Transformation Format, 8-bit form) before they are written to the work file.</p> <p>If a Unicode work file is specified, you cannot use the transfer options Use conversion table, Substitute line references and Incorporate free rules.</p>
Unload file or Load file or	<p>Only activated if Use default options (this is the default) or Use additional options has been selected (see below).</p> <p>The name of the work file to be used for the function. See also Work Files.</p>

Item	Explanation
Scan file (Server)	
Browse	Not applicable to server unload/load/scan files. Invokes the browse function to select a work file from a directory.
Use default options	Default options are used (this is the default). See also Set Additional Options below.
Use additional options	Used in connection with Set (see below).
Set	Only activated if Use additional options has been selected. Invokes the Options window where you can modify the default settings and enter additional options for the processing sequence. For the options available, see Set Additional Options below.
Use Option Workplan	If this option is selected, a Workplan of the type OPTION is used. Select a Workplan from the combo box or type the name of a Workplan of the type OPTION. See also Workplans .
List (Option Workplan)	Only activated if Use Option Workplan (see above) has been selected and the name of a valid Workplan of the type OPTION is entered. Displays the contents of the Workplan specified.

Set Additional Options

Special unload options are set in the **Additional Options** window.

► To invoke the Additional Options window

- In the **Settings** window, on the tabbed page **Options**, select the option button **Use additional options** and choose the **Set** button.

The **Additional Options** window contains the tabs **General**, **Special**, **Transfer** and **XREF**. Note that not all of the tabs may appear on the screen, because they depend on the function used, the settings defined and the products installed.

The options provided on the tabbed pages are described in the following section. For descriptions of mentioned keywords and valid input values, see also [option-setting](#) in the section *Direct Commands*.

This section covers the following topics:

- [General](#)

- Special
- Transfer
- XREF

General

The tabbed page **General** of the **Additional Options** window contains the group boxes **Report**, **FDIC** and **FSEC** where you can specify the following:

Item	Explanation
Write report	<p>Writes a report of the objects processed to Work File 4. This is the default setting for object processing except for the find function.</p> <p>In remote environments, the report data is written to a Natural text member that is stored in the Workplan library.</p> <p>To display a report, select Show Report File from the Options menu. See also Reports in <i>Tools</i>.</p>
Start new report	<p>Only activated if Write report has been selected.</p> <p>Deletes the contents of Work File 4 before a new report is written.</p>
Use Report option in Find commands	<p>Only applies to the find function and if Write report has been selected.</p> <p>If selected, this option writes a report of the objects found by using the same report settings (for example, the name of the report file and the additional option Start new report) specified for the unload, load or scan function.</p> <p>If this option has not been activated for the current find command, no report is written. In this case, the Show Report File option (see also Reports in <i>Tools</i>) will only display old report data of a previous find, unload, load or scan function, if performed.</p>
Report file	<p>Only available in local environments and if Write report has been selected.</p> <p>The name of a report file. Choose the Browse button to select a name from a directory.</p>
FDIC	<p>Only applies if Predict is installed.</p> <p>With the FDIC option, you specify the Predict file (FDIC) used for processing XRef data:</p> <p>DBID The database ID where the FDIC file is located.</p> <p>FNR The file number where the FDIC file is located.</p> <p>Password Optional. The Adabas password of the Adabas file where the FDIC file is located.</p> <p>Cipher key Optional. The cipher code of the Adabas file where the FDIC file is located.</p>
FSEC	<p>Only applies if Natural Security is installed.</p> <p>With the FSEC option, you specify the Natural Security data file (FSEC) used for security checks:</p>

Item	Explanation
	<p>DBID The database ID where the FSEC file is located.</p> <p>FNR The file number where the FSEC file is located.</p> <p>Password Optional. The Adabas password of the Adabas file where the FSEC file is located.</p> <p>Cipher key Optional. The cipher code of the Adabas file where the FSEC file is located.</p>

Special

The tabbed page **Special** contains the group boxes **Replace**, **Load/Scan** and **Load** where you can specify the following:

Item	Explanation
Replace all	Replaces all objects.
Do not replace	Does not replace any objects. This is the default.
Replace obsolete objects	Replaces objects with a date older than the date of the objects in the load file.
Replace except newer	Replaces all objects except those with a date newer than the date of the objects in the load file.
Write restart information	<p>Only applies to the load function.</p> <p>When this option is set, restart information is provided for the restart load function.</p> <p>See also Restart Load in <i>Functions</i>.</p>
Restart file	<p>Only applies to the load function in local environments and if Write restart information has been selected.</p> <p>The name of the work file to be used for the restart data: Work File 6 (default setting) or the <i>restart-file</i> specified.</p> <p>Choose the Browse button to select a name from a directory.</p> <p>For further information, see Restart Load in <i>Functions</i>.</p>
Check version	<p>Only applies when loading objects in a mainframe environment.</p> <p>Checks the Natural version of the cataloged object to be loaded: the Natural version under which the objects were cataloged and written to the work file is compared with the current Natural version. Objects cataloged under a Natural version higher than the current one will be rejected.</p>
ADAFDU file	<p>See ADAFDUWORKFILE in <i>option-setting</i> in <i>Direct Commands</i>.</p> <p>Choose the Browse button to select a name from a directory.</p>

Item	Explanation
Number of objects to process	<p>Specifies the number of objects to be processed.</p> <p>Enter a value with a maximum of 5 digits. If a value greater than 0 (zero) is specified, the load or scan function stops after the specified number of objects has been processed.</p> <p>Note: If a cataloged Natural object is processed directly after the source object of the same name, they are considered one object.</p>
Use FDDM file for processing DDMs	<p>Only applies in environments where the FDDM system file has been activated in the NATPARM module.</p> <p>If this option has been activated (this is the default), the FDDM system file is used for processing DDMs with the load, unload or find function.</p> <p>Specify the library SYSTEM and the Natural object type V (see Natural Library Object Details in <i>Object Specification</i>) for processing DDMs.</p> <p>If used with the load function, all DDMs are loaded into the FDDM system file. In this case, the parameter NEWLIBRARY is ignored.</p> <p>See also the syntax diagram of the <i>option-clause</i> in <i>Direct Commands</i>.</p>

Transfer

The tabbed page **Transfer** of the **Additional Options** window contains the group boxes **Load**, **Conversion Table**, **Unload** and **Data Area Format**.

The items provided in these group boxes and the functions to which they apply are described in the following section.

Item	Explanation	Function
Translate into upper case	Translates any source code to be loaded to upper case.	Load
Load code page	<p>In this text box, you can enter the name of the code page to be used for the load function.</p> <p>If a code page is specified, all object sources unloaded into a work file in UTF-8 will be converted with the specified code page when they are loaded into a work file. See also Unicode work file.</p> <p>If you enter *CODEPAGE as the code page name, the value assigned to the system variable *CODEPAGE is used (see the <i>System Variables</i> documentation).</p> <p>If no code page name is specified, the source objects are converted with the code page used when unloading them.</p>	Load

Item	Explanation	Function
	If Load code page is specified, you cannot use the options Use conversion table and Translate into upper case .	
Use conversion table	<p>Unload: Converts data to EBCDIC format by using the internal Natural conversion table (System table) or a conversion table defined by the user (User-defined table).</p> <p>Load: Converts data to ASCII format by using the internal Natural conversion table (System table) or a conversion table defined by the user (User-defined table). Note that this only applies if the data in the work file is in EBCDIC format or if a conversion program is specified (see User-defined table).</p>	Unload Load
System table	<p>Only activated if Use conversion table has been selected.</p> <p>Unload: Converts data to EBCDIC format by using the internal Natural conversion table.</p> <p>Load: Converts data to ASCII format by using the internal Natural conversion table.</p>	Unload Load
User-defined table	<p>Only activated if Use conversion table has been selected.</p> <p>Specifies that a user-defined table is to be used for conversion.</p> <p>If the name of a conversion program has been entered in the text box Table Name for Load function or Table name for Unload function (see below), data is converted to EBCDIC or ASCII format by using the conversion program defined.</p> <p>To specify an individual conversion program, the program must be stored in the library SYSOBJH or one of its steplibs. See the example program OTNCONAE in the library SYSOBJH.</p> <p>If no conversion program is specified, by default, the corresponding conversion table in the Natural file NATCONV.INI is used for the unload ([ISO8859_1->EBCDIC]) and the load ([EBCDIC->ISO8859_1]) functions.</p>	Unload Load
Table name for Load function	<p>Only activated if User-defined table has been selected.</p> <p>Enter the name of a user-defined conversion table.</p>	Load
Table name for Unload function	<p>Only activated if User-defined table has been selected.</p> <p>Enter the name of a user-defined conversion table.</p>	Unload
Substitute line references	<p>Only applies if source-code line numbers are used for statement references.</p> <p>If line numbers are used as references in the source code, the line numbers of referenced lines and the line number references are replaced with labels. The sources are not modified in the database.</p>	Unload

Item	Explanation	Function
Include line numbers	If you choose this option, the line numbers will be transferred. (By default, line numbers in Natural objects are not transferred.)	Unload
Incorporate free rules	If Predict is installed, Predict rules associated with a map are incorporated into the map source.	Unload
Leave data areas as they are	Does <i>not</i> convert data area sources to the new internal data area format or the old internal data area format (see below). This is the default.	Unload Load
Convert data areas into new internal format	Converts data area sources to the new internal data area format. For details, see <i>Data Area Editor</i> in the <i>Editors</i> documentation.	Unload Load
Convert data areas into old internal format	Converts data area sources to the old internal data area format. If one or more data area sources cannot be converted to the old internal data area format, the Object Handler issues a corresponding message when unloading is complete. In addition, in the Status column of the unload report generated by the unload function, a corresponding remark appears next to the names of the data area sources affected.	Unload Load

XREF

The tabbed page **XREF** of the **Additional Options** window contains the group boxes **Load** and **Unload**. **XREF** is only available when unloading or loading data in internal format, that is, if the check box **Transfer format** has *not* been selected. Predict must be installed to process XRef data.

The items provided in the group boxes and the functions to which they apply are described in the following section.

Item	Explanation	Function
On	Unload: Unloads GPs (generated programs, that is, cataloged objects) and their cross-reference data, if any. Load: Loads GPs and their cross-reference data if cross-references exist in the work file.	Unload Load
Off	Ignore XRef data. No XRef data is processed.	Unload Load
Doc	Predict definition required. Loads cataloged objects only if Predict entries exist for the objects in the FDIC system file.	Load
Force	XRef data and Predict definition required. Loads GPs (generated programs) and their cross-reference data only if cross-references exist in the work file and if Predict entries exist for the objects in the FDIC system file.	Load
Special	Load GPs and XRef data.	Load

Item	Explanation	Function
	Loads GPs and their cross-reference data (if any).	

31 Settings - Parameters

- Set Global Parameters 192

On the tabbed page **Parameters** of the **Settings** window you can specify the following:

Item	Explanation
Do not use parameters	If selected (default setting), no parameters are set.
Use global parameters	Used in connection with Set (see below). Global parameters are used. See also <i>Set Global Parameters</i> below.
Set (global parameters)	Only activated if Use global parameters has been selected. Invokes the Unload/Load Parameters window. See <i>Set Global Parameters</i> below for descriptions of keywords and valid input values.
Use Parameter Workplan	If selected, a Workplan of the type PARAMETER is used. Select a Workplan from the combo box or type the name of a Workplan of the type PARAMETER. See also <i>Workplans</i> .
List (Parameter Workplan)	Only activated if Use Parameter Workplan (see above) has been selected and the name of a valid Workplan of the type PARAMETER was entered. Displays the contents of the Workplan specified.

Set Global Parameters

Only applies to the load or unload function.

You can use global parameter to change the settings for the objects to be processed with the load or unload function, and to change the target environment for the load function. For example, you can specify new names (or name ranges) under which the selected objects are unloaded to the work file, or you can specify a different library into which the selected objects are loaded from the work file.

If global parameters are specified during the unload function, the parameter settings affect the objects before they are written to the work file. If they are specified during the load function, the parameter settings affect the objects before they are written to the target environment.

Global parameters are set in the **Unload/Load Parameters** window.

▶ To invoke the Unload/Load Parameters window

- In the **Settings** window, on the tabbed page **Parameters**, select the **Use global parameters** option button and choose **Set**.

The tabbed page **Parameters** contains the group boxes **General** and **Load Target**. The items provided in these group boxes and the data you can specify are described in the following section.

- [General](#)
- [Load Target](#)

General

The tabbed page **General** contains a table where the relevant parameters are listed in the **Parameters** column. The values that can be specified to change the settings of the parameters are entered in the columns **Value** and **New Value**.

New Value does not apply to the parameter **Error number difference** and target system file specifications for load functions.

If no value has been entered in **Value**, the value entered in **New Value** affects all objects to which the specific parameter setting applies. If a value has been entered in **Value**, the value entered in **New Value** only affects objects to which the specific parameter setting and the value entered in **Value** apply.

If a value or new value is not relevant to the type of object to be processed, any value entered in either column will be ignored. For example: Natural system error messages have no library name. Therefore, when processing Natural system error messages, a value entered in **Value** or **New Value** for the parameter **Library** will be ignored.

For valid parameter settings, see also [parameter-setting](#) in the section *Direct Commands*.

On the tabbed page **General** you can specify the following:

Parameter	Explanation
Name	Value/New Value: A single object name or a range of names: see Name in <i>Name, Date and Time Specification</i> and Rules for New Values .
Library	Value/New Value: A single library name or a range of names: see Name and Rules for New Values .
Date	Value/New Value: A single date or a range of dates: see Date and Time in <i>Name, Date and Time Specification</i> , and Rules for New Values .
User ID	Value/New Value: A single user ID or a range of user IDs: see Name and Rules for New Values .

Parameter	Explanation
Language code	<p>Only applies when processing Natural system error messages or user-defined error messages.</p> <p>Value/New Value:</p> <p>Up to 8 valid language codes such as code 4 for Spanish. If more than one language code is specified, Value must contain the same number of language codes. In this case, the language code in Value is replaced by the language code in the corresponding New Value.</p> <p>Note: New Value does not apply to the long texts of Natural system error messages for which English (code 1) is the only valid language.</p>
Error number difference	<p>Only applies when processing Natural system error messages or user-defined error messages.</p> <p>A 4-digit positive or negative value (+/-nnnn) to be used as a new number range for error messages. Start and end values must be provided in the Number from/to or Message from/to boxes to validate whether the new range can be applied to the selected error messages.</p> <p>Example:</p> <p>If Number from/to selects message numbers 1 to 10 and Error number difference is set to 2000, the messages will be renumbered from 2001 to 2010. A value of -1000 in Error number difference would cause a validation error.</p>
FDT DBID	<p>Value/New Value:</p> <p>A valid database ID (DBID) for the Adabas FDT.</p>
FDT FNR	<p>Value/New Value:</p> <p>A valid file number (FNR) for the Adabas FDT.</p>
External path	<p>Value/New Value:</p> <p>The name of the path for external files.</p>

Rules for New Values

The following applies to **New Value** for **Name**, **Library**, **Date** and **User ID**:

If **New Value** contains a range with an asterisk (*) such as ABC*, the number of characters before the asterisk (*) determines the number of characters to be replaced in **Value**. This is also valid if **Value** is shorter than the range specified in **New Value** (see the second example in Examples 2 below).

Examples:

1. If **Name** is ABCDEFG and **New Value** is set to ZYX*, the resulting object name is ZYXDEFG.
2. If **Name** is AB and **New Value** is set to ZYX*, the resulting object name is ZYX.

3. If **Date** is 2005-03-26 and **New Value** is set to 2006*, the resulting object date is 2006-03-26.

Load Target

The tabbed page **Load Target** only applies to the load function.

The page contains the group boxes **Load FNAT**, **Load FUSER** and **Load NCP** where you can specify the following:

Item	Explanation
Load FNAT: DBID FNR Password Cipher key	The database ID (DBID) and file number (FNR) of the target FNAT system file. This system file is used for all library objects whose library name starts with SYS, but not SYSTEM. In remote environments, you can also specify the Adabas password and cipher code.
Load FUSER: DBID FNR Password Cipher key	The DBID and FNR of the target FUSER system file. This system file is used for all library objects whose library name does not start with SYS, and for the library SYSTEM. In remote environments, you can also specify the Adabas password and cipher code.
Load NCP: DBID FNR Password Cipher key	The DBID and FNR of the target Adabas file into which the Natural command processor sources are to be loaded. Additionally, you can specify the Adabas password and cipher code.

32 Workplans

- Creating, Selecting and Modifying Workplans 198
- Contents of Workplans 198
- Examples of Workplans 199
- Referencing Workplans 200

Workplans define individual standard procedures for command execution, object selection and parameter or option settings which can be used to further automate function processing.

Workplans are Natural objects of the type Text. They are, by default, stored in the library WORKPLAN located in the current FUSER system file.

Creating, Selecting and Modifying Workplans

You can use the **administration** function (see the relevant section) to create a Workplan, select a Workplan from a list, modify a Workplan, or change the default library for Workplans. The default library can also be changed by using the `Workplan-Library` entry of the **Profile** option (see also *Profile Settings*).

Contents of Workplans

A Workplan consists of a header (generated by the Object Handler) and an associated instructional or textual part. Instructional parts contain Object Handler commands and parameter and/or option settings. Textual parts contain plain text only. Header and instructional or textual parts can contain comments (for example, the short description of the Workplan) that must start with the delimiter characters `/*` and are restricted to one line.

There are six types of Workplan: PROCEDURE, SELECTION, LIST, PARAMETER, OPTION and TEXT.

The table below lists the valid headers (to be entered if creating a Workplan outside the Object Handler) for the corresponding types of Workplan and describes the contents of the instructional or textual part. Additionally, it provides cross references to the clauses that apply when specifying Object Handler direct commands. The Object Handler direct commands provided are explained in the section *Direct Commands*.

Valid Headers	Contents	Related Topic in <i>Direct Commands</i>
TYPE PROCEDURE	An Object Handler command procedure. This Workplan can contain any combination of Object Handler commands available for PROCEDURE. Enter a sequence of commands separated by semicolons (;).	<i>Basic Command Syntax</i>
TYPE SELECTION	Selection criteria for objects. This Workplan can be used in Object Handler Workplan commands.	<i>select-clause</i>

Valid Headers	Contents	Related Topic in <i>Direct Commands</i>
TYPE LIST	A list of objects. This Workplan can be used in Object Handler Workplan commands.	<i>select-clause</i> <i>Object List - LIST Workplan</i>
TYPE PARAMETER	Parameters for the unload or load function. This Workplan can be used to change attributes for the objects to be processed such as the name of a new target library where objects are loaded. TYPE PARAMETER can be used in Object Handler Workplan commands.	<i>parameter-setting</i>
TYPE OPTION	Options for the unload or load function, for example, report settings. This Workplan can be used in Object Handler Workplan commands.	<i>option-setting</i>
TYPE TEXT	Comments or any other text that can be used for documentation purpose.	Not applicable

Examples of Workplans

The following table lists examples of instructional parts contained in a Workplan.

Workplan Type	Instruction	Explanation
PROCEDURE	FINDLIB * LIB TEST	Check whether the library TEST exists.
PROCEDURE	UNLOAD A* LIB TEST	Local environments: Unload from the library TEST into Work File 1 all Natural programming objects and shared resources starting with A, and all user-defined error messages; write the report into Work File 4. Remote environments located on mainframe platforms: Unload from the library TEST into Work File 1 on the server system all Natural programming objects starting with A, and all user-defined error messages; write the report into the corresponding text member of the Workplan library.
SELECTION	* LIB TEST	Process all objects from the library TEST.
TEXT	This is a Workplan comment.	Any text.

This section covers the following topic:

- [Example of Workplan Contents](#)

Example of Workplan Contents

The following is an example listing of a PROCEDURE Workplan where the UNLOAD command is executed:

```
TYPE PROCEDURE /* VERSION=03.01 NATURAL VERSION=06.93.09 PL=0 AUTHOR=SAG ↵
DATE=2010-07-20 09:40:12
/* unload from library TEST with target library PROD01
UNLOAD * LIB TEST OBJTYPE N
WITH NEWLIBRARY PROD01
WHERE REPORT MYREP01
```

Referencing Workplans

You can reference a Workplan by using Object Handler menu functions or direct commands (see also the section [Direct Commands](#)).

The following syntax applies when referencing a Workplan with the Object Handler direct commands described in the section [Direct Commands](#).

```
( workplan-name
  [ LIBRARY library-name ]
  [ DBID dbid [ FNR fnr ] ] [ NAME vsam-name ]
  [ CIPHER cipher ]
  [
    {
      PASSWORD
      PSW
    } password
  ]
)
```

The syntactical options are explained in the following section:

- [Keyword Explanation](#)

Keyword Explanation

The table below describes the keywords and values that apply to the syntax for referencing Workplans.

Keyword	Values	Default Value
<i>workplan-name</i>	The name of the Natural text member in the Workplan library to be used as the Workplan.	No default
LIBRARY	The name of the library where the Workplan is located.	WORKPLAN
DBID	The ID of the Adabas database where the Workplan library is located.	0 (current FNAT/FUSER)
FNR	The number of the Adabas file where the Workplan library is located.	0 (current FNAT/FUSER)
NAME	Only applies to objects on mainframes. The name of a valid VSAM file where the Workplan library is located.	blank (current FNAT/FUSER)
CIPHER	Only applies to objects on mainframes. An 8-digit cipher code.	blank (current FNAT/FUSER)
PASSWORD	Only applies to objects on mainframes. An 8-character Adabas password.	blank (current FNAT/FUSER)

33

Name, Date and Time Specification

- Name 204
- Date 205
- Time 206

You can use a name, a date, a time or a range of names, dates and times to select Natural library objects, Natural command processor sources, Natural-related objects, Natural DDMs (data definition modules on a remote mainframe platform) or external files.

Name

You can specify a name or a range of names.

In the list of options below, *value* is any combination of one or more characters:

	Input	Items Selected
	<i>value</i>	All items with names equal to <i>value</i> .
	*	All items.
	>	
	?	All items with any single character for each question mark (?) entered.
Leading Characters	<i>value</i> *	All items with names that start with <i>value</i> . Example: AB* Selected: AB, AB1, ABC, ABEZ Not selected: AA1, ACB
Wildcard	<i>value</i> ?	All items with names that start with <i>value</i> and end with any single character for each question mark (?) entered. Example: ABC? Selected: ABCA, ABCZ Not selected: AXC, ABCAA
	<i>value</i> ? <i>value</i> ? <i>value</i> * <i>value</i> ? * <i>value</i> ? <i>value</i> *	All items that match <i>value</i> combined with asterisk (*) and question mark (?) in any order. Example: A?C*Z Selected: ABCZ, AXCBBBZ, AN CZ Not selected: ACBZ, ABDEZ, AXCBBBZA
Start Value	<i>value</i> >	All items with names greater than or equal to <i>value</i> . Example: AB> Selected: AB, AB1, BBB, ZZZZZZZ Not selected: AA1, AAB
End Value	<i>value</i> <	All items with names less than or equal to <i>value</i> . Example: AX< Selected: AB, AWW, AX Not selected: AXA, AY



Note: The parameter specification option **New Value** only allows leading characters (asterisk (*) notation). See [Rules for New Values](#) in *Set Global Parameters* in the section *Settings*.

Date

All date values within the Object Handler are specified in international date format.

You can specify a date, a range of dates, a special date or a range of special dates. A date must be specified in the format *YYYY-MM-DD* (*YYYY* = year, *MM* = month, *DD* = day).

In the list of options below, the underlined portion of a keyword represents its valid abbreviation, and *value* is any combination of one or more digits:

	Input Value	Items Selected
Date	<i>YYYY-MM-DD</i>	All items with a date equal to <i>YYYY-MM-DD</i> . Example: 2003-02-15
Leading characters	<i>value*</i>	All items with a date that starts with <i>value</i> . Example: 2002* Selected: 2002-01-01, 2002-12-31 Not selected: 2001-12-31, 2003-01-01
Start value	<i>value></i>	All items with a date greater than <i>value</i> . Example: 2002-05> Selected: 2002-05-01, 2002-12-31, 2003-01-01, 2003-12-31 Not selected: 2002-04-31, 2001-12-31 Special dates can be used as <i>value</i> (see below).
End value	<i>value<</i>	All items with a date less than <i>value</i> . Example: 2003-02< Selected: 2002-05-01, 2002-12-31, 2003-01-01, 2003-01-31 Not selected: 2003-02-01, 2003-05-18 Special dates can be used as <i>value</i> (see below).
Special Dates		
TODAY (+/- nnnn)		All items with the date of the current day. The day can be followed by <i>+nnnn</i> or <i>-nnnn</i> where <i>nnnn</i> has a maximum of 4 digits. The resulting date is computed as the date of the current day plus or minus <i>nnnn</i> days. Example: If the current date is 2003-03-01, TODAY +5 results in 2003-03-06.

	Input Value	Items Selected
YESTERDAY		All items with the date of the day before the current day.
MONTH		All items with the date range of the current month. Example: The current month is 2003-02. Selected: 2003-02-01, 2003-02-30 Not selected: 2003-03-01
		FMDATE: Starts with the first day of the current month. TODATE: Ends with the last day of the current month. If the values of FMDATE and TODATE are identical, the selection is restricted to one day.
YEAR		All items with the date range of the current year. Example: The current year is 2003. Selected: 2003-01-01, 2002-12-31 Not selected: 2002-31-12
		FMDATE: Starts with the first day of the current year. TODATE: Ends with the last day of the current year. If the values of FMDATE and TODATE are identical, the selection is restricted to one year.



Note: The parameter specification option **New Value** only allows leading characters (asterisk (*) notation). See [Rules for New Values](#) in *Set Global Parameters* in the section *Settings*.

Time

You can specify a time or a range of times. The time must be specified in the format *HH:II:SS* (*HH* = hours, *II* = minutes, *SS* = seconds).

In the list of options below, *value* is any combination of one or more digits:

	Input Value	Items Selected
Time	<i>HH:II:SS</i>	All items with a time equal to <i>HH:II:SS</i> . Example: 14:15:16
Leading characters	<i>value</i> *	All items with a time that starts with <i>value</i> . Example: 13:* Selected: 13:00:00, 13:10:53, 13:59:59 Not selected: 12:59:59, 14:00:00

34 Work Files

- Work File Assignment 208
- Work File Format 209

This section describes work files and valid formats that apply to the unload, load and scan functions of the Object Handler.

Object Handler functions invoked in a local environment will only process objects from this local environment, with a work file located in the current local file system. Object Handler functions invoked in a remote environment will only process objects from the same remote environment. The work file used for the load or unload function is located in the same remote environment or in your local file system.

See also [General](#) in *Settings - Options*, the section *Work Files* in the *Operations* documentation, the statement `DEFINE WORK FILE` in the *Statements* documentation and the profile parameter `WORK` in the *Parameter Reference* documentation.

Work File Assignment

This section covers the following topics:

- [Local Environments](#)
- [Remote Environments](#)

Local Environments

The following table lists the work files used by the Object Handler in local environments.

File	Explanation
Work File 1	Used for the unload, load and scan functions. Contains the data unloaded. See also Transfer Work File in <i>Tools</i> .
Work File 3	Used for internal reports. Contains scan and find results.
Work File 4	Used if the option Write report (see <i>Settings - Options</i>) is set. Write report is the default setting for object processing. Contains report data.
Work File 5	The target file for the Adabas FDTs (Field Definition Tables) loaded.
Work File 6	Used for the load function if the option Write restart information (see Restart Load in <i>Functions</i>) is set. Contains restart information data.
Work File 7	An internal work file.
Work File 9	An internal work file.

File	Explanation
Work File 10	Used if the trace mode is set. See also <i>Traces</i> in <i>Tools</i> .
Work Files 11 to 15	Internal work files.

Remote Environments

The following table lists the work files used in remote environments.

File	Location	Explanation
Work File 1	local system	If the option Local work file is set (see <i>Settings - Options</i>), this work file is used for the unload, load and scan functions. It contains the data processed. Additionally, in mainframe environments, this work file is used to transfer work files from the local environment to the server and vice versa as described in <i>Transfer Work File</i> in <i>Tools</i> .
Work File 1	server system	If the option Local work file is <i>not</i> set (see <i>Settings - Options</i>), this work file is used for the unload, load and scan functions. It contains the data processed. Additionally, in mainframe environments, this work file is used to transfer work files from the local environment to the server and vice versa as described in <i>Transfer Work File</i> in <i>Tools</i> . Note that in a mainframe environment work files must be specified properly. See also <i>Natural User Access Method for Print and Work Files in Configuring Natural</i> in the documentation <i>Natural Operations for Mainframes</i> .
Work File 3	local system	An internal work file.
Work File 9	local system	An internal work file.
Work Files 11 to 15	local system	Internal work files.

Work File Format

There are two file formats for unloading objects in the source environment into work files and for loading them from work files into the target environment: an internal format and the Transfer format. Work files must be of internal format to transfer binary data. Work files must be of Transfer format to transfer text data.

This section covers the following topics:

- [Internal Format](#)

- [Transfer Format](#)

Internal Format

The internal format is an internal record layout for work files that are used to transfer Natural sources and cataloged objects, error messages, Natural command processor sources, Adabas FDTs and non-Natural objects from one environment to another.

With the internal format activated, Natural objects are read from the source environment and written to a Natural work file by using the unload function of the Object Handler. This work file can be transported to another environment with standard file transfer services. In the target environment, the objects can then be read from the work file and loaded into the local file or database system with the load function of the Object Handler.

To transfer objects between identical platforms, use work files of internal format. Use portable work files of internal format if you want to transport objects between different UNIX, OpenVMS or Windows platforms, for example, from a little-endian machine to a big-endian machine. See also [Portable work file](#) in *Settings - Options, Portable Natural Generated Programs (Programming Guide)* and `DEFINE WORK FILE` (*Statements* documentation).

The Object Handler uses internal format by default. When using the internal format (**Transfer format** check box *not* selected), Work File 1 must be of binary format. To achieve this, omit the file extension or use the file extension `.sag`.



Notes:

1. Work files created by the utility SYSPAUL must be processed in internal format.
2. For remote environments: Work files created by the utility NATUNLD on the server, must be processed in internal format. The work files must be created on a server of the same platform where NATUNLD was applied.

Transfer Format

See also [Transfer format](#) in the section *Settings - Options*.

The Transfer format is a general record layout for work files that contain load or unload data. This format is platform-independent and can be used to transfer the sources of Natural objects, Natural command processor sources, error messages and Adabas FDTs (Field Definition Tables) from one hardware platform to another and between Windows and mainframe, UNIX or OpenVMS platforms.

With the check box **Transfer format** selected, the unload function of the Object Handler reads Natural objects from a hardware platform and then restructures them.

Formatted records are written to a Natural work file that can be transported to another platform with standard file transfer services. On the target platform, the load function of the Object Handler

then reads the objects from the work file and loads them into the local file or database system. The objects read from the work file are restructured according to the structure of the new hardware platform.

Specifying Work Files in Local Environments

If Transfer format is specified (check box **Transfer format** selected), Work File 1 must be of text (ASCII) format. To achieve this, a file extension must be used, but not the file extension `.sag`. If Transfer format is *not* specified, Work File 1 must be of binary format. To achieve this, omit the file extension or use the file extension `.sag`.

Specifying Work Files in Remote Environments

Work File 1 must be defined in the server environment. If Transfer format is specified (check box **Transfer format** selected), Work File 1 contains data of text (ASCII) format. If Transfer format is *not* specified, Work File 1 contains data of internal format.

Handling Sources in Unicode/UTF-8

Transfer format is also used to unload or load sources of Natural objects in Unicode/UTF-8 (Universal Transformation Format, 8-bit form). If you specify the corresponding unload option (`WORKFILETYPE` set to `UTF-8` in command mode or **Unicode work file** in menu mode), all object sources will be unloaded into a work file in UTF-8. If you specify the corresponding load option (`LOAD-CODE-PAGE` in command mode or **Load code page** in menu mode), all object sources in UTF-8 will be converted with the specified code page when they are loaded into a Natural system file.

Work Files from SYSTRANS

Work files created by the utility SYSTRANS must be processed in Transfer format. Work files that contain object sources encoded in UTF-8 cannot be processed with SYSTRANS.

35 Direct Commands

The Object Handler provides direct commands for the following purposes:

- To execute an Object Handler function such as unloading or loading objects in batch mode or in direct command online mode without using Object Handler menus (see also *Batch or Direct Command Calls*).
- To execute or reference a Workplan (see also the section *Workplans*).
- To be used as an instruction in a Workplan.

This section describes the basic command syntax and the individual clauses, parameter and option settings available to perform these tasks. In addition, you can view examples that illustrate the use of direct commands.

The symbols used in the syntax diagrams shown in this section are explained in *System Command Syntax* in the *System Commands* documentation.

This section covers the following topics:

Basic Command Syntax

select-clause

Object List - LIST Workplan

parameter-setting

option-setting

Examples of Using Direct Commands

36

Basic Command Syntax

This section describes the Object Handler direct commands provided for executing Object Handler functions and Workplans of the type PROCEDURE. It also describes the commands used for migrating from the old utility SYSTRANS to the Object Handler.

For explanations of the variable values contained in the syntax diagrams shown in this section, refer to the relevant sections in the *Object Handler* documentation. For explanations of the symbols used in the syntax diagrams, see *System Command Syntax* in the *System Commands* documentation.

```
EXECUTE (procedure-workplan)
```

Executes a Workplan of the type PROCEDURE. See also the section [Workplans](#).

```
UNLOAD select-clause [parameter-setting] [option-setting]
```

Unloads the objects defined in the *select-clause* with the parameters defined in *parameter-setting* with the options defined in *option-setting*.

```
LOAD select-clause [parameter-setting] [option-setting]
```

Loads the objects defined in the *select-clause* with the parameters defined in *parameter-setting* with the options defined in *option-setting*.

```
LOADALL [parameter-setting] [option-setting]
```

Loads all objects from a work file with the parameters defined in *parameter-setting* with the options defined in *option-setting*.

```
SCAN select-clause [option-setting]
```

Scans a work file for the objects defined in the *select-clause* with the options defined in *option-setting*.

```
SCANALL [option-setting]
```

Scans a work file for all objects with the options defined in *option-setting*.

```
FIND select-clause [option-setting]
```

Finds the objects defined in the *select-clause* with the options defined in *option-setting* and writes a report of the objects found into Work File 3. In addition, a report of the objects found can be written to Work File 4 or a specified report file.

```
FINDLIB select-clause [option-setting]
```

Finds the libraries for Natural objects or Natural command processor sources defined in the *select-clause* with the options defined in *option-setting* and writes a report of the objects found into Work File 3. In addition, a report of the objects found can be written to Work File 4 or a specified report file.

```
DELETE select-clause [option-setting]
```

Deletes the objects defined in the *select-clause* with the options defined in *option-setting*.

Restriction: It is not possible to delete an FDT.

```
UNDELI select-clause [option-setting]
```

Unloads delete instructions for the objects defined in the *select-clause* with the options defined in *option-setting*.

```
RESTART [restart-file]
```

Continues an interrupted load function. This is only possible if information was written to a restart file during the aborted load. Restart load information can be written to Work File 6 or a specified restart file. See also [RESTART](#) in the section *option-setting* (*Direct Commands*) and [Restart Load](#).

DISPLAY STATISTICS

Displays statistics information about the objects processed.

SYSTRANS *systrans-direct-command*

Executes an Object Handler direct command issued in the syntax of the old utility SYSTRANS. See also [*Migration from SYSTRANS to the Object Handler*](#).

37

select-clause

▪ Syntax of select-clause	220
▪ SELECTION or LIST Workplan	220
▪ Natural Library Object and DDM Selection	221
▪ Natural-Related Object Selection	227
▪ Natural-Related Debug Environment Selection	229
▪ Natural-Related Profile Selection	231
▪ Natural-Related DL/I Subfile Selection	232
▪ Natural System Error Message Selection	234
▪ Natural Command Processor Selection	236
▪ External File Selection	237
▪ FDT Selection	239
▪ Application Selection	240
▪ Object Selection for Delete Instructions	242

The *select-clause* comprises either a Workplan of the type SELECTION or LIST, or selection specifications for the objects, FDTs or applications to be processed.

This section describes the syntax that applies to the *select-clause*. The keywords and variable values contained in the syntax diagrams represent the parameters that can be used to specify object selection criteria. If indicated, a variable value must be supplied with a keyword.

Syntax of select-clause

The *select-clause* consists of one of the following options:

<pre>(<i>selection-workplan</i>) (<i>list-workplan</i>) <i>object-selection</i> <i>delete-instruction-selection</i></pre>

The *selection-workplan* and *list-workplan* options are explained in *SELECTION or LIST Workplan* below.

The use of *object-selection* depends on the object type, DDM, FDT or application you want to process, for each of which the appropriate syntax and keywords are explained in the remainder of this section.

The *delete-instruction-selection* options are explained in *Delete Instructions for Selected Objects*.

SELECTION or LIST Workplan

A Workplan of the type SELECTION contains a header (**TYPE SELECTION**) and a selection from one of the following types of object or file: Natural library objects, Natural-related objects, Natural system error messages, Natural command processor sources, external files or Adabas FDTs (Field Definition Tables).

A Workplan of the type LIST contains a header (**TYPE LIST**) and a selection list of objects as described in the section *Object List - LIST Workplan*. Such an object list can be used for the UNLOAD, LOAD or FIND command only.

For further information on using Workplans, see the section *Workplans*.

Natural Library Object and DDM Selection

This selection is used to select Natural objects for processing including Natural DDMs, user-defined error messages and shared resources.

The appropriate syntax is shown and explained in the following section.

- [Syntax of Natural Library Object and DDM Selection](#)

Syntax of Natural Library Object and DDM Selection

```

object-name
LIBRARY library-name
[ DBID dbid FNR fnr
  [NAME vsam-name]
  [CIPHER cipher]
  [ { PASSWORD } password
    PSW ] ]
[OBJTYPE group-type]
[ SETNO set-number [SETUSER set-user] [SETLIBRARY set-library] ]
[NATTYPE object-type]
[SCKIND object-kind]
[MODE object-mode]
[FMNUM error-number-from]
[TONUM error-number-to]
[SLKIND message-type]
[LANGUAGE languages]
[DDMDBID dgm-dbid] [DDMFNR dgm-fnr]
[NATVERS natural-version]
[ DATE date
  [FMDATE date-from] [TODATE date-to] ]
[ [SIZE size]
  [FMSIZE size-from] [TOSIZE size-to] ]
[USERID user-id]
[TID terminal-id]
[except-clause]

```

except-clause

```

EXCEPT
( object-name
[LIBRARY library-name]
[OBJTYPE group-type]
[SCKIND object-kind]
[NATTYPE object-type]
[MODE object-mode]
[SLKIND message-type]
[FMNUM error-number-from] [TONUM error-number-to]
[LANGUAGE languages]
[DDMDBID dgm-dbid] [DDMFNR dgm-fnr]
[NATVERS natural-version]
[
    DATE date
    [FMDATE date-from] [TODATE date-to]
]
[
    SIZE size
    [FMSIZE size-from] [TOSIZE size-to]
]
[USERID user-id]
[TID terminal-id]
)

```

**Notes:**

1. For the command `FINDLIB`, only the following keywords are processed: `LIBRARY`, `DBID`, `FNR`, `NAME`, `CIPHER` and `PASSWORD` or `PSW`.
2. When processing Natural DDMs on mainframes, you must set `OBJTYPE` to `D`. In addition, some keywords do not apply to DDMs, which is described below.

Keyword Explanation of Natural Library Object and DDM Selection

The keywords and valid values for the objects to be processed are described in the following section.

Keyword	Valid Values	Default Value
<i>object-name</i>	<p>A valid object name or a range of names.</p> <p>If <i>object-name</i> contains blank characters, it must be enclosed in double quotation marks (" ").</p> <p>See also <i>Name</i> in <i>Name, Date and Time Specification</i>.</p>	none

Keyword	Valid Values	Default Value
LIBRARY	<p>A valid library name or a range of names.</p> <p>If OBJTYPE (see below) is set to D, the library name is ignored.</p> <p>If SETNO is specified, a range of names is not allowed.</p> <p>See also <i>Name</i>.</p>	none
DBID	<p>Not valid for DDMs on mainframes (OBJTYPE set to D; see below).</p> <p>A valid database ID.</p>	0 (current FNAT/FUSER)
FNR	<p>Not valid for DDMs on mainframes (OBJTYPE set to D; see below).</p> <p>A valid file number.</p>	0 (current FNAT/FUSER)
NAME	<p>Only applies to objects on mainframes. Not valid for DDMs on mainframes (OBJTYPE set to D; see below).</p> <p>A valid VSAM name.</p>	blank (current FNAT/FUSER)
CIPHER	<p>Only applies to objects on mainframes. Not valid for DDMs on mainframes (OBJTYPE set to D; see below).</p> <p>The 8-digit cipher code of the Adabas file where the objects are stored.</p>	blank (current FNAT/FUSER)
PASSWORD or PSW	<p>Only applies to objects on mainframes.</p> <p>Not valid for DDMs on mainframes (OBJTYPE set to D; see below).</p> <p>An 8-character Adabas password.</p>	blank (current FNAT/FUSER)
OBJTYPE	<p>Types of object are:</p> <ul style="list-style-type: none"> D DDMs (objects on mainframes only) E Natural error messages N Natural programming objects R Shared resources * Asterisk (all) <p>or a valid combination.</p> <p>Exception: Object type D cannot be combined with any other type.</p>	*
SETNO	<p>Only applies to the unload and find functions and if Predict is installed. Not applicable to application objects (see also <i>Selecting Application Objects</i>).</p> <p>A one- or two-digit number that identifies the retained set to be used for the names of the objects to be processed. A retained set is created with the save set option of the LIST XREF command.</p>	none

Keyword	Valid Values	Default Value
	<p>If SETNO is specified, the value specified for <i>object-name</i> is ignored.</p> <p>For detailed information on Predict sets, refer to the <i>Predict</i> documentation.</p>	
SETUSER	<p>Only applies to the unload and find functions and if Predict is installed. Not applicable to application objects (see also <i>Selecting Application Objects</i>).</p> <p>The ID of the user who created the Predict set. If no ID is specified, the value of the system variable *USER (see also the <i>System Variables</i> documentation) is used.</p>	*USER
SETLIBRARY	<p>Only applies to the unload and find functions and if Predict is installed. Not applicable to application objects (see also <i>Selecting Application Objects</i>).</p> <p>The name of the library to be searched for a Predict set. If you do not specify SETLIBRARY, the library specified with LIBRARY is used instead.</p>	
NATTYPE	<p>Not applicable if OBJTYPE is set to D.</p> <p>One or more single-character codes for Natural object types:</p> <ul style="list-style-type: none"> P Program N Subprogram S Subroutine C Copycode H Helproutine T Text 7 Function 8 Adapter G Global data area L Local data area A Parameter data area M Map 4 Class 3 Dialog 5 Natural command processor V DDM (not on mainframes) 6 Object view 9 Resource (on mainframes only) * All object types 	*

Keyword	Valid Values	Default Value
SCKIND	<p>Not applicable if OBJTYPE is set to D.</p> <p>The kind of Natural programming objects. Valid input values are:</p> <p>S Source objects: objects that are only stored in source form.</p> <p>C Cataloged objects: objects that are only stored in cataloged form.</p> <p>A All source and cataloged objects.</p> <p>W All STOWed objects: source and cataloged objects with identical date and time.</p> <p>B Source and cataloged objects if both exist.</p> <p>Note: W and B are valid for the UNLOAD and FIND commands only. For LOAD and SCAN, W and B are valid entries, but they are treated like A (all objects). If data is processed in Transfer format, only S (source objects) or A applies.</p>	A
MODE	<p>Not applicable if OBJTYPE is set to D.</p> <p>The programming mode of the Natural programming objects. Valid input values are:</p> <p>A Any.</p> <p>R All objects in reporting mode.</p> <p>S All objects in structured mode.</p>	
FMNUM	<p>A start number of Natural error messages.</p> <p>Valid range: 1 to 9999.</p>	1
TONUM	<p>An end number of Natural error messages.</p> <p>Valid range: 1 to 9999.</p> <p>The value must be greater than or equal to the value of FMNUM, if specified.</p>	9999 or value of FMNUM (if specified)
SLKIND	<p>The kind of Natural error message text. Valid input values are:</p> <p>S Short text. Cannot be applied to the DELETE command (see Basic Command Syntax).</p> <p>L Long text.</p> <p>A Short and/or long text.</p> <p>B Short and long text, if both exist.</p>	A

Keyword	Valid Values	Default Value
LANGUAGE	Up to 8 valid language codes (for example, code 1 for English) of user-defined error messages. An asterisk (*) selects all language codes.	*
DDMDBID	The valid database ID (1 to 65535) of a DDM. UNLOAD, LOAD and SCAN: 0 denotes that no check is performed. DDMs are processed, regardless of their database ID (DBID).	0
DDMFNR	The valid file number (1 to 65535) of a DDM. UNLOAD, LOAD and SCAN: 0 denotes that no check is performed. DDMs are processed, regardless of their file number (FNR).	0
NATVERS	The Natural version of Natural programming objects. You can also specify a range of versions: see Name .	blank (no check)
DATE	The save or catalog date of Natural programming objects, and the date of shared resources. You can add a time by inserting a blank between date and time. For the format and ranges allowed, see Date and Time in <i>Name, Date and Time Specification</i> . Special terms allowed are YESTERDAY and TODAY. See Special Dates in <i>Date</i> .	blank (no check)
FMDATE	A start value: The date on or after which Natural programming objects were cataloged or saved, and the date of shared resources. The format is identical to DATE. See Date . Special terms allowed are YEAR, MONTH, YESTERDAY and TODAY. See Special Dates in <i>Date</i> .	blank (no check)
TODATE	An end value: The date on or before which Natural programming objects were cataloged or saved, and the date of shared resources. The format is identical to DATE. See Date . Special terms allowed are YEAR, MONTH, YESTERDAY and TODAY. See Special Dates in <i>Date</i> .	blank (no check) or high value (if FMDATE specified)
SIZE	The size of Natural programming objects and shared resources (up to 7 digits).	0 (no check)
FMSIZE	A start value: The minimum size of Natural programming objects and shared resources (up to 7 digits).	0 (no check)
TOSIZE	An end value: The maximum size of Natural programming objects and shared resources (up to 7 digits).	0 (no check) or high value

Keyword	Valid Values	Default Value
		(if FMSIZE specified)
USERID	The ID of the user who saved or cataloged the Natural programming objects. You can also specify a range of user IDs: see <i>Name</i> .	blank (no check)
TID	Not applicable if OBJTYPE is set to D. The ID of the terminal where the Natural programming objects were saved or cataloged (provided by the Natural system variable *INIT-ID). You can also specify a range of terminal IDs: see also <i>Name</i> .	blank (no check)
EXCEPT	All items that match the selection criteria entered before EXCEPT are checked against <i>all</i> parameters contained within the parentheses following the keyword EXCEPT. If they match all these parameters too, they are not processed.	not applicable



Notes:

1. Parameters that are irrelevant for **OBJTYPE** are ignored. For example: **DATE**, **SIZE** and **USERID** have no meaning for Natural error messages.
2. **DBID**, **FNR**, **NAME**, **CIPHER** and **PASSWORD** or **PSW** are ignored by the **LOAD** or **SCAN** command. These parameters must instead be specified in the *parameter-setting* clause as described for **LOADFNAT...** and **LOADFUSER...** in *Keyword Explanation of parameter-clause*.
3. If an object for shared resources contains blank characters, it must be enclosed in double quotation marks (" ").

Natural-Related Object Selection

This selection is used to select Natural-related objects for processing.

The appropriate syntax is shown and explained in the following section.

- Syntax of Natural-Related Object Selection

Syntax of Natural-Related Object Selection

```

object-name NATPATH natural-path-name
[
    DATE date
    [FMDATE date-from] [TODATE date-to]
]
[
    SIZE size
    [FMSIZE size-from] [TOSIZE size-to]
]
[ EXCEPT
    (object-name NATPATH natural-path-name
        [
            DATE date
            [FMDATE date-from] [TODATE date-to]
        ]
        [
            SIZE size
            [FMSIZE size-from] [TOSIZE size-to]
        ]
    )
]

```

Keyword Explanation of Natural-Related Object Selection

The keywords and valid input values for the objects to be processed are described in the following section.

Keyword	Valid Values	Default Value
<i>object-name</i>	The name of a Natural-related object. If <i>object-name</i> contains blank characters, it must be enclosed in double quotation marks (" "). See also <i>Name</i> in <i>Name, Date and Time Specification</i> .	none
NATPATH	NATDIR NATGUI_BMP TMP_PATH NATBIN PROFILE_PATH PARM_PATH NATERR	none
DATE	The modification date of Natural-related objects. You can add a time by inserting a blank between date and time. For the format and ranges allowed, see <i>Date</i> and <i>Time</i> in <i>Name, Date and Time Specification</i> . Special terms allowed are: YESTERDAY and TODAY. See <i>Special Dates</i> in <i>Date</i> .	blank (no check)

Keyword	Valid Values	Default Value
FMDATE	A start value: The date on or after which Natural-related objects were modified. The format is identical to DATE. See Date . Special terms allowed are: YEAR, MONTH, YESTERDAY and TODAY. See Special Dates in Date .	blank (no check)
TODATE	An end value: The date on or before which Natural-related objects were modified. The format is identical to DATE. See Date . Special terms allowed are: YEAR, MONTH, YESTERDAY and TODAY. See Special Dates in Date .	blank (no check) or high value (if FMDATE specified)
SIZE	The size of Natural-related objects (up to 10 digits).	0 (no check)
FMSIZE	A start value: The minimum size of Natural-related objects (up to 10 digits).	0 (no check)
TOSIZE	An end value: The maximum size of Natural-related objects (up to 10 digits).	0 (no check) or high value (if FMSIZE specified)
EXCEPT	See EXCEPT in <i>Natural Library Object and DDM Selection</i> .	



Note: The NATPATH clause in the EXCEPT part is evaluated by the LOAD or SCAN command only.

Natural-Related Debug Environment Selection

This selection is used to select Natural-related debug environments for processing.

Note that Natural-related debug environments exist on mainframe platforms only.

The appropriate syntax is shown and explained in the following section.

- Syntax of Natural-Related Debug Environment Selection

Syntax of Natural-Related Debug Environment Selection

```

object-name
NATPATH DEBUG
[LIBRARY library-name]
[
    DBID dbid [FNR fnr]
]
[NAME vsam-name]
[CIPHER cipher]
[
    {
        PASSWORD
        PSW
    } password
]
[ EXCEPT
    (object-name
    [LIBRARY library-name]
    )]

```

Keyword Explanation of Natural-Related Debug Environment Selection

The keywords and valid input values for the debug environments to be processed are described in the following section.

Keyword	Valid Values	Default Value
<i>object-name</i>	A valid debug environment name or a range of names. See also <i>Name</i> in <i>Name, Date and Time Specification</i> .	none
LIBRARY	A valid library name or a range of names. See also <i>Name</i> .	none
DBID	A valid database ID.	0 (current FUSER)
FNR	A valid file number.	0 (current FUSER)
NAME	A valid VSAM name.	blank (current FUSER)
CIPHER	The 8-digit cipher code of the Adabas file where the debug environments are stored.	blank (current FUSER)
PASSWORD or PSW	An 8-character Adabas password.	blank (current FUSER)

Keyword	Valid Values	Default Value
EXCEPT	See EXCEPT in <i>Natural Library Object and DDM Selection</i> .	not applicable



Note: DBID, FNR, NAME, CIPHER and PASSWORD or PSW are ignored by the LOAD or SCAN command. These parameters must instead be specified in the *parameter-setting* clause as described for LOADFNAT... and LOADFUSER... in [Keyword Explanation of parameter-clause](#).

Natural-Related Profile Selection

This selection is used to select Natural-related profiles for processing.

Note that Natural-related profiles exist on mainframe platforms only.

The appropriate syntax is shown and explained in the following section.

- [Syntax of Natural-Related Profile Selection](#)

Syntax of Natural-Related Profile Selection

```

object-name
NATPATH PROFILE
[OBJTYPE profile-type]
[   DBID dbid[FNR fnr] ]
[NAME vsam-name]
[CIPHER cipher]
[   {   PASSWORD   }   password   ]
[ EXCEPT
   (object-name
   [OBJTYPE profile-type]
   )]

```

Keyword Explanation of Natural-Related Profile Selection

The keywords and valid input values for the profiles to be processed are described in the following section.

Keyword	Valid Values	Default Value														
<i>object-name</i>	A valid profile name or a range of names. See also Name in <i>Name, Date and Time Specification</i> .	none														
OBJTYPE	The type of profile: <table border="1" data-bbox="337 394 776 716"> <tr><td></td><td></td></tr> <tr><td>D</td><td>Device profile</td></tr> <tr><td>E</td><td>Editor profile</td></tr> <tr><td>M</td><td>Map profile</td></tr> <tr><td>P</td><td>Parameter profile</td></tr> <tr><td>*</td><td>Asterisk (all profile types)</td></tr> <tr><td></td><td></td></tr> </table> or any combination.			D	Device profile	E	Editor profile	M	Map profile	P	Parameter profile	*	Asterisk (all profile types)			*
D	Device profile															
E	Editor profile															
M	Map profile															
P	Parameter profile															
*	Asterisk (all profile types)															
DBID	A valid database ID.	0 (current FNAT)														
FNR	A valid file number.	0 (current FNAT)														
NAME	A valid VSAM name.	blank (current FNAT)														
CIPHER	The 8-digit cipher code of the Adabas file where the profiles are stored.	blank (current FNAT)														
PASSWORD or PSW	An 8-character Adabas password.	blank (current FNAT)														
EXCEPT	See EXCEPT in <i>Natural Library Object and DDM Selection</i> .	not applicable														



Note: DBID, FNR, NAME, CIPHER and PASSWORD or PSW are ignored by the LOAD or SCAN command. These parameters must instead be specified in the *parameter-setting* clause as described for LOADFNAT... and LOADFUSER... in [Keyword Explanation of parameter-clause](#).

Natural-Related DL/I Subfile Selection

This selection is used to select Natural-Related DL/I subfiles for processing.

Note that Natural-related DL/I subfiles exist on mainframe platforms only.

The appropriate syntax is shown and explained in the following section.

- Syntax of Natural-Related DL/I Subfile Selection

Syntax of Natural-Related DL/I Subfile Selection

```

object-name
NATPATH SUBFILE
[OBJTYPE subfile-type]

[ DBID dbid [FNR fnr] ]

[NAME vsam-name]
[CIPHER cipher]

[ { PASSWORD } password
  PSW ]

[ EXCEPT
  ( object-name
    [OBJTYPE subfile-type]
  ) ]

```

Keyword Explanation of Natural-Related DL/I Subfile Selection

The keywords and valid input values for the DL/I subfiles to be processed are described in the following section.

Keyword	Valid Values	Default Value								
<i>object-name</i>	A valid DL/I subfile name or a range of names. See also Name in <i>Name, Date and Time Specification</i> .	none								
OBJTYPE	The type of DL/I subfile: <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 20%;"></td> <td></td> </tr> <tr> <td>D</td> <td>NDB</td> </tr> <tr> <td>P</td> <td>NSB</td> </tr> <tr> <td>*</td> <td>Asterisk (both subfile types)</td> </tr> </table>			D	NDB	P	NSB	*	Asterisk (both subfile types)	*
D	NDB									
P	NSB									
*	Asterisk (both subfile types)									
DBID	A valid database ID.	0 (current FDIC)								
FNR	A valid file number.	0 (current FDIC)								
NAME	A valid VSAM name.	blank (current FDIC)								
CIPHER	The 8-digit cipher code of the Adabas file where the DL/I subfiles are stored.	blank								

Keyword	Valid Values	Default Value
		(current FDIC)
PASSWORD or PSW	An 8-character Adabas password.	blank (current FDIC)
EXCEPT	See EXCEPT in <i>Natural Library Object and DDM Selection</i> .	not applicable



Note: DBID, FNR, NAME, CIPHER and PASSWORD or PSW are ignored by the LOAD or SCAN command. These parameters must instead be specified in the *parameter-setting* clause as described for LOADFNAT... and LOADFUSER... in [Keyword Explanation of parameter-clause](#).

Natural System Error Message Selection

This selection is used to select Natural system error messages for processing.

The appropriate syntax is shown and explained in the following section.

- [Syntax of Natural System Error Message Selection](#)

Syntax of Natural System Error Message Selection

```

ERROR NATERROR
  DBID dbid FNR
  [ fnr [NAME
    [ vsam-name ] [ { PASSWORD } password ]
    [ CIPHER
    [ cipher ] ] ] ]
  [FMNUM error-number-from] [TONUM error-number-to]
  [SLKIND message-type]
  [LANGUAGE languages]
  [ EXCEPT
    (
      [FMNUM error-number-from] [TONUM error-number-to]
      [SLKIND message-type]
      [LANGUAGE languages]
    ) ]

```

Keyword Explanation of Natural System Error Message Selection

The keywords and valid input values for the Natural system error messages to be processed are described in the following section.

Keyword	Valid Values	Default Value
DBID	Only applies to system error messages on mainframes. A valid database ID.	0 (current FNAT)
FNR	Only applies to system error messages on mainframes. A valid file number.	0 (current FNAT)
NAME	Only applies to system error messages on mainframes. A valid VSAM name.	blank (current FNAT)
CIPHER	Only applies to system error messages on mainframes. The 8-digit cipher code of the Adabas file where the system error messages are stored.	blank (current FNAT)
PASSWORD or PSW	Only applies to system error messages on mainframes. An 8-character Adabas password.	blank (current FNAT)
FMNUM	A start number of system error messages. Valid range: 1 to 9999.	1
TONUM	An end number of system error messages. Valid range: 1 to 9999. The value must be greater than or equal to the value of FMNUM if specified.	9999 or value of FMNUM (if specified)
SLKIND	See SLKIND in <i>Natural Library Object and DDM Selection</i> .	A
LANGUAGE	Up to 8 valid language codes (for example, code 1 for English) of system error messages. An asterisk (*) selects all language codes.	*
EXCEPT	See EXCEPT in <i>Natural Library Object and DDM Selection</i> .	



Note: DBID, FNR, NAME, CIPHER and PASSWORD or PSW are ignored by the LOAD or SCAN command. These parameters must instead be specified in the *parameter-setting* clause as described for LOADFNAT . . . in [Keyword Explanation of parameter-clause](#).

Natural Command Processor Selection

This selection is used to select Natural command processor sources for processing.

The appropriate syntax is shown and explained in the following section.

- [Syntax of Natural Command Processor Source Selection](#)

Syntax of Natural Command Processor Source Selection

```

object-name PROCESSOR ncp-library-name
[
    DBID ncp-dbid FNR ncp-fnr [file-options] ]
[EXCEPT
    (object-name
    [LIBRARY ncp-library-name]
    )]

```

file-options

```

[NAME ncp-vsam-name]
[CIPHER ncp-cipher]
[ { PASSWORD } ncp-password ]
  PSW

```



Note: For the command FINDLIB, only the following keywords are processed: PROCESSOR, DBID, FNR, NAME, CIPHER and PASSWORD or PSW.

Keyword Explanation of Natural Command Processor Source Selection

The keywords and valid input values for the Natural command processor sources to be processed are described in the following section.

Keyword	Valid Values	Default Value
<i>object-name</i>	The name of a valid Natural command processor source or a range of names. See also Name in <i>Name, Date and Time Specification</i> .	none
PROCESSOR	A valid library name or a range of names. See also Name .	none
DBID	The valid database ID of the Adabas file where the Natural command processor sources are stored.	Value of LFILE 190

Keyword	Valid Values	Default Value
FNR	The valid file number of the Adabas file where the Natural command processor sources are stored.	Value of LFILE 190
NAME	Only applies to Natural command processor sources on mainframes. A valid VSAM name.	blank
CIPHER	The 8-digit cipher code of the Adabas file where the Natural command processor sources are stored.	blank
PASSWORD or PSW	The 8-character Adabas password of the Adabas file where the Natural command processor sources are stored.	blank
EXCEPT	See EXCEPT in <i>Natural Library Object and DDM Selection</i> .	



Note: DBID, FNR, NAME, CIPHER and PASSWORD or PSW are ignored by the LOAD or SCAN command. These parameters must instead be specified in the *parameter-setting* clause as described for LOADNCP... in [Keyword Explanation of parameter-clause](#).

External File Selection

This selection is used to select external files for processing.

The appropriate syntax is shown and explained in the following section.

■ Syntax of External File Selection

Syntax of External File Selection

```

external-file-name PATH external-path-name
[
    DATE date
    [FMDATE date-from] [TODATE date-to]
]
[
    SIZE size
    [FMSIZE size-from] [TOSIZE size-to]
]
[ EXCEPT
    (external-file-name [PATH external-path-name]
        [
            DATE date
            [FMDATE date-from] [TODATE date-to]
        ]
        [
            SIZE size
            [FMSIZE size-from] [TOSIZE size-to]
        ]
    )]

```

Keyword Explanation of External File Selection

The keywords and valid input values for the external files to be processed are described in the following section.

Keyword	Valid Values	Default Value
<i>external-file-name</i>	The name of an external file. If <i>external-file-name</i> contains blank characters, it must be enclosed in double quotation marks (" "). See also <i>Name</i> in <i>Name, Date and Time Specification</i> .	none
PATH	The name of the path where the external file is located.	none
DATE	The modification date of external files. You can add a time by inserting a blank between date and time. For the format and ranges allowed, see <i>Date</i> and <i>Time</i> in <i>Name, Date and Time Specification</i> . Special terms allowed are YESTERDAY and TODAY. See <i>Special Dates</i> in <i>Date</i> .	blank (no check)
FMDATE	A start value: The date on or after which external files were modified. The format is identical to DATE. See <i>Date</i> . Special terms allowed are YEAR, MONTH, YESTERDAY and TODAY. See <i>Special Dates</i> .	blank (no check)
TODATE	An end value: The date on or before which external files were modified. The format is identical to DATE. See <i>Date</i> . Special terms allowed are YEAR, MONTH, YESTERDAY and TODAY. See <i>Special Dates</i> .	blank (no check) or high value (if FMDATE specified)
SIZE	The size of external files (up to 10 digits).	0 (no check)
FMSIZE	A start value: the minimum size of external files (up to 10 digits).	0 (no check)
TOSIZE	An end value: The maximum size of external files (up to 10 digits).	0 (no check) or high value (if FMSIZE specified)
EXCEPT	See <i>EXCEPT</i> in <i>Natural Library Object and DDM Selection</i> .	



Note: The NATPATH clause in the EXCEPT part is only evaluated by the LOAD and SCAN commands.

FDT Selection

This selection is used to select Adabas FDTs (Field Definition Tables) for processing.

For loading FDTs, see also [FDTs](#) in the section *Object Specification*.

The appropriate syntax is shown and explained in the following section.

- [Syntax of FDT Selection](#)

Syntax of FDT Selection

```
FDT
DBID dbid
{ FNR fnr [CIPHER cipher] [ { PASSWORD
  PSW } password ] }
  FMFNR fnr-start TOFNR fnr-end }
```

Keyword Explanation of FDT Selection

The keywords and valid input values for the FDTs to be processed are described in the following section.

Keyword	Valid Values	Default Value
DBID	The database ID of the FDT.	none
FNR	The file number of the FDT.	none
CIPHER	The 8-digit Adabas cipher code of the FDT.	none
PASSWORD or PSW	The 8-character Adabas password of the FDT.	none
FMFNR	Only applies to the FIND or UNLOAD command. A start value: The file number (FNR) of an FDT.	none
TOFNR	Only applies to the FIND or UNLOAD command. An end value: The file number (FNR) of an FDT.	none

Application Selection

This selection applies when working in a remote environment located on a mainframe, a UNIX or an OpenVMS platform.

The selection is used for applications maintained in Natural Studio's application workspace and the libraries or objects that belong to these applications.

The appropriate syntax is shown and explained in the following section.

- [Selecting Base and Compound Applications](#)
- [Selecting Application Libraries](#)
- [Selecting Application Objects](#)

Selecting Base and Compound Applications

This selection only applies to the find function.

Syntax

```
APPLICATION APNAME application-name
[APTYPE application-type]
[COMPAAPPLICATION compound-application-name]
[
    EXCEPT
    (APNAME application-name
     [APTYPE application-type]
    )]
```

Selecting Application Libraries

This selection only applies to the find function.

Syntax

```
APPLICATION APLIBRARY application-library-name
[BASEAPPLICATION base-application-name]
[COMPAAPPLICATION compound-application-name]
[
    DBID dbid [FNR fnr] ]
[
    EXCEPT
    (APLIBRARY application-library-name
```

```
[BASEAPPLICATION base-application-name
)]
```

Selecting Application Objects

This selection only applies to the find and unload functions.

Syntax

```
APPLICATION AOBJECTS application-object-name
[BASEAPPLICATION base-application-name]
[COMPAPPLICATION compound-application-name]
[LIBRARY library-name]
[object-specification]
[
    EXCEPT
    (AOBJECT application-object-name
    [LIBRARY library-name]
    [BASEAPPLICATION base-application-name]
    [object-specification]
    )
)]
```

Keyword Explanation of Application Selection

The keywords and valid input values for the applications, application libraries or application objects to be processed are described in the following section.

Keyword	Valid Values	Default Value								
APNAME	A valid name of a Natural application or a range of names. See also <i>Name</i> in <i>Name, Date and Time Specification</i> .	*								
APTYPE	A valid application type: <table border="1" data-bbox="553 1459 1187 1648"> <tbody> <tr> <td></td> <td></td> </tr> <tr> <td>B</td> <td>Base application</td> </tr> <tr> <td>0</td> <td>Compound application</td> </tr> <tr> <td>*</td> <td>All: base and/or compound applications</td> </tr> </tbody> </table>			B	Base application	0	Compound application	*	All: base and/or compound applications	*
B	Base application									
0	Compound application									
*	All: base and/or compound applications									
COMPAPPLICATION	Only applies if APTYPE is set to * or B. The name of a compound application to which the specified base application belongs or a range of names.	none								

Keyword	Valid Values	Default Value
	Only base applications that belong to the specified compound application(s) are selected; base applications that do not belong to a compound application are not selected.	
EXCEPT	See EXCEPT in <i>Natural Library Object and DDM Selection</i> .	not applicable
APLIBRARY	The valid name of a library that belongs to a Natural base or compound application or a range of names. See also Name in <i>Name, Date and Time Specification</i> .	*
BASEAPPLICATION	The valid name of a Natural base application to which an application library or application object belongs. See also Name in <i>Name, Date and Time Specification</i> .	*
DBID	The valid database ID of an application library.	0 (no check)
FNR	The valid file number of an application library.	0 (no check)
APOBJECT	The valid name of an application object that belongs to a base or compound application, or a range of names. See also Name in <i>Name, Date and Time Specification</i> .	*
LIBRARY	A valid library name or a range of names. If OBJTYPE is set to D (see <i>Natural Library Object and DDM Selection</i>), the library name is ignored. See also Name in <i>Name, Date and Time Specification</i> .	*
<i>object-specification</i>	Indicates that additional selection criteria can be specified for application objects as shown in the syntax diagram for Natural library objects and DDMs: all items listed below LIBRARY <i>library-name</i> can also be applied to application objects whereas <i>object-name</i> in the EXCEPT clause is irrelevant for application objects.	not applicable

Object Selection for Delete Instructions

This selection is used to specify delete instructions for Natural library objects, DDMs, user-defined error messages and Natural system error messages. The delete instructions are executed when a work file of internal format is loaded in the target environment with the [DELETEALLOWED](#) option specified.

The appropriate syntax is shown and explained in the following section.

- [Syntax of Delete Instructions for Natural Library Objects and DDMs](#)

- Syntax of Delete Instructions for User-Defined Error Messages
- Syntax of Delete Instructions for Natural System Error Messages

Syntax of Delete Instructions for Natural Library Objects and DDMs

```

object-name
LIBRARY library-name
[ OBJTYPE { N } ]
[ NATTYPE { * } ]
[ SCKIND object-kind ]

```

Keyword Explanation of Delete Instructions for Natural Library Objects and DDMs

The keywords and valid values for the objects to be processed are described in the following section.

Keyword	Valid Values	Default Value
<i>object-name</i>	A valid object name or a start value (<i>value*</i>) for a range of names such as ABC*.	none
LIBRARY	A valid library name. A range specification is <i>not</i> allowed. If OBJTYPE (see below) is set to D, the library name is ignored.	none
OBJTYPE	A valid object-type code: D DDMs (objects on mainframes only) N Natural programming objects Object type D cannot be combined with object type N.	*
NATTYPE	Not applicable if OBJTYPE is set to D. A Natural object type. Valid input values are: * All object types V DDMs (not on mainframes)	*
SCKIND	Not applicable if OBJTYPE is set to D. The kind of Natural programming objects. Valid input values are:	A

Keyword	Valid Values	Default Value
	S Source objects. If used in the <i>except-clause</i> (see <i>Syntax of Natural Library Object and DDM Selection</i>): objects that are stored only in source form.	
	C Cataloged objects. If used in the <i>except-clause</i> : objects that are stored only in cataloged form.	
	A All source and cataloged objects.	

Syntax of Delete Instructions for User-Defined Error Messages

```

*
LIBRARY library-name
OBJTYPE E
FMNUM error-number-from
[TONUM error-number-to]
[SLKIND message-type]
[LANGUAGE languages]

```

library-name denotes the name of a single library; a range specification is not allowed.

For explanations of the other elements used in this syntax, see [Keyword Explanation of Natural Library Object and DDM Selection](#).

Syntax of Delete Instructions for Natural System Error Messages

```

ERROR NATERROR
FMNUM error-number-from
[TONUM error-number-to]
[SLKIND message-type]
[LANGUAGE languages]

```

For explanations of the elements used in this syntax, see [Keyword Explanation of Natural System Error Message Selection](#).

38

Object List - LIST Workplan

- Syntax of object-type-and-location 246
- Syntax of object-name-description 247
- Example of an Object List 249

An object list is a Workplan of the type LIST, which specifies object selection criteria for the objects to be processed in the UNLOAD, LOAD, FIND or DELETE command. An object list can be used as an alternative to the *select-clause* and the SELECTION Workplan.

The following syntax applies to an object list:

```
TYPE LIST
{ object-type-and-location (object-name-description ...) } ...
```

The syntactical options are explained in the following section. The keywords and variable values contained in the syntax diagrams shown in this section represent parameters that are used to specify object selection criteria. If indicated, a variable value must be supplied with a keyword. Each syntax element (except for the ones enclosed in parentheses) must start on a new line and end on the same line.

For explanations of the keywords contained in the syntax diagrams, refer to the section *select-clause*.

Syntax of object-type-and-location

The syntax diagrams that apply to *object-type-and-location* are shown in the following section.

- [Natural Objects and DDMs](#)
- [Natural System Error Messages](#)
- [Natural Command Processor Sources](#)
- [Natural-Related Objects](#)
- [External Files](#)
- [FDTs](#)

Natural Objects and DDMs

```
LIBRARY library-name
[
  DBID dbid FNR fnr [NAME vsam-name] [CIPHER
    cipher] [ { PASSWORD } password ] ]
[ OBJTYPE group-type ]
```



Notes:

1. No ranges are allowed for *library-name*.
2. For DDMs on mainframe platforms, OBJTYPE must be set to D.

Natural System Error Messages


`ERROR NATERROR`

```
[
  DBID dbid FNR fnr [NAME vsam-name] [CIPHER [ { PASSWORD } password ] ]
  cipher
  PSW
]
```

Natural Command Processor Sources

`PROCESSOR ncp-library-name`

```
[
  DBID dbid FNR fnr [NAME vsam-name] [CIPHER [ { PASSWORD } password ] ]
  cipher
  PSW
]
```

 **Note:** No ranges are allowed for *ncp-library-name*.

Natural-Related Objects

`NATPATH natural-path-name`

External Files

`PATH external-path-name`

FDTs

`FDT`

Syntax of object-name-description

The syntax diagrams that apply to *object-name-description* are shown in the following section:

- [Natural Objects](#)
- [Natural System Error Messages](#)
- [Natural Command Processor Sources](#)
- [Natural-Related Objects](#)
- [External Files](#)

- FDTs

Natural Objects

```
object-name [SLKIND object-kind]  
{ error-number [SLKIND message-type] [LANGUAGE languages]  
  FMNUM error-number-from TONUM error-number-to [SLKIND message-type] [LANGUAGE  
  languages] }
```

Natural System Error Messages

```
error-number [SLKIND message-type] [LANGUAGE languages]  
{ FMNUM error-number-from TONUM error-number-to [SLKIND message-type] [LANGUAGE  
  languages] }
```

Natural Command Processor Sources

```
object-name
```

Natural-Related Objects

```
related-object-name
```

External Files

```
external-file-name
```

FDTs

```
DBID dbid FNR fnr [CIPHER cipher] [ { PASSWORD  
  PSW } password ]
```

Example of an Object List

The following is an example of a Workplan of the type LIST:

```
TYPE LIST
  LIBRARY LIB-1 OBJTYPE N      /* process Natural objects from library 'LIB-1'
    ( A* SCKIND S              /* all sources objects whose names start with 'A'
      B1                        /* source and/or cataloged object of 'B1'
      CDE> SCKIND C )          /* all cataloged objects with names greater than/equal ↵
to 'CDE'
  /*                          /* comment line
  LIBRARY LIB-2                /* process Natural objects from library 'LIB-2'
                                /* including error messages and shared resources
  ( *                           /* all source and/or cataloged objects
                                /* including shared resources
  FMNUM 1 TONUM 100            /* error messages from 1 to 100
  )
```


39 parameter-setting

- Syntax of parameter-clause 252
- Keyword Explanation of parameter-clause 253

The *parameter-setting* clause is used to change attributes for the LOAD or UNLOAD command for the objects to be processed and to define target destinations for the LOAD command (for example, FNAT).

The following syntax applies to the *parameter-setting* clause:

```
WITH
{
  (parameter-workplan)
  parameter-clause
}
```

For an explanation of the syntax that applies to *parameter-workplan*, refer to [Referencing Workplans](#) in the section *Workplans*.

This section covers the following topics:

Syntax of parameter-clause

The syntax of the *parameter-clause* is shown in the following diagram. If indicated, a variable value must be supplied with a keyword.

```
[
  [NAME old-name] NEWNAME new-name ]
[
  [LIBRARY old-library-name]
  NEWLIBRARY new-library-name ]
[
  LOADFNATDBID fnat-dbid LOADFNATFNR fnat-fnr
  [LOADFNATNAME vsam-name]
  [LOADFNATCIPHER fnat-cipher]
  [
    {
      LOADFNATPASSWORD
      LOADFNATPSW
    } fnat-password
  ]
  [
  LOADFUSERDBID fuser-dbid LOADFUSERFNR fuser-fnr
  [LOADFUSERNAME fuser-vsam-name]
  [LOADFUSERCIPHER fuser-cipher]
  [
    {
      LOADFUSERPASSWORD
      LOADFUSERPSW
    } fuser-password
  ]
  [
  LOADNCPDBID ncp-file-dbid LOADNCPFNR ncp-file-fnr
  [LOADNCPNAME ncp-file-vsam-name]
  [LOADNCPCIPHER ncp-file-cipher]
  [
    {
      LOADNCPPASSWORD
      LOADNCPPSW
    } ncp-file-password
  ]
]
```


[[FDTDBID <i>old-fdt-dbid</i> FDTFNR <i>old-fdt-fnr</i>] NEWFDTDBID <i>new-fdt-dbid</i> NEWFDTFNR <i>new-fdt-fnr</i>]
[ERRNUMDIFF <i>modification-of-error-message-range</i>]
[[LANGUAGE <i>old-language</i>]]
[NEWLANGUAGE <i>new-language</i>]
[[DATE <i>old-date</i>] NEWDATE <i>new-date</i>]
[[USERID <i>old-userid</i>] NEWUSERID]
[<i>new-userid</i>]
[[TID <i>old-terminal-id</i>] NEWTID]
[<i>new-terminal-id</i>]
[[PATH <i>old-external-path-name</i>]]
[NEWPATH <i>new-external-path-name</i>]

Keyword Explanation of parameter-clause

The keywords and variable values (if relevant) of the *parameter-clause* are explained in the following section.

Keyword	Values	Restricted to Command
NAME	The object name to be checked if NEWNAME is specified.	
NEWNAME	A new object name. Note: Not applicable to DDMs on mainframe platforms.	
LIBRARY	The library name to be checked if NEWLIBRARY is specified.	
NEWLIBRARY	A new library name. Note for the LOAD function: NEWLIBRARY does <i>not</i> affect the library name used in the delete instruction of a work file that is processed with the DELETEALLOWED option.	
LOADFNATDBID	The database ID (DBID) of FNAT libraries.	LOAD
LOADFNATFNR	The file number (FNR) of FNAT libraries.	LOAD
LOADFNATNAME	Only applies to objects on mainframes. An FNAT VSAM file name.	LOAD
LOADFNATCIPHER	Only applies to objects on mainframes. An FNAT cipher code.	LOAD
LOADFNATPASSWORD	Only applies to objects on mainframes. An FNAT Adabas password.	LOAD
or		

Keyword	Values	Restricted to Command
LOADFNATPSW		
LOADFUSERDBID	The DBID of FUSER libraries.	LOAD
LOADFUSERFNR	The FNR of FUSER libraries.	LOAD
LOADFUSERNAME	Only applies to objects on mainframes. An FUSER VSAM file name.	LOAD
LOADFUSERCIPHER	Only applies to objects on mainframes. An FUSER cipher code.	LOAD
LOADFUSERPASSWORD or LOADFUSERPSW	Only applies to objects on mainframes. An FUSER Adabas password.	LOAD
LOADNCPDBID	The DBID of the Adabas file for Natural command processor sources.	LOAD
LOADNCPFNR	The FNR of the Adabas file for Natural command processor sources.	LOAD
LOADNCPNAME	Only applies to objects on mainframes. The VSAM name of the Adabas file for Natural command processor sources.	LOAD
LOADNCPCIPHER	The cipher code of the Adabas file for Natural command processor sources.	LOAD
LOADNCPPASSWORD or LOADNCPPSW	Only applies to objects on mainframes. The Adabas password of the Adabas file for Natural command processor sources.	LOAD
FDTDBID	The DBID of the Adabas FDT (Field Definition Table) to be checked if NEWFDTDBID is specified.	
NEWFDTDBID	A new DBID of the FDT.	
FDTFNR	The DBID of the FDT to be checked if NEWFDTFNR is specified.	
NEWFDTFNR	A new FNR of the FDT.	
ERRNUMDIFF	A number (positive or negative) that is to be added to the Natural error messages during the UNLOAD or LOAD command. ERRNUMDIFF can only be specified if FMNUM and TONUM (see <i>select-clause</i>) have been specified as selection criteria. Otherwise, it is not possible to check for valid results.	
LANGUAGE	Up to 8 valid language codes (for example, code 1 for English) of Natural error messages to be checked if NEWLANGUAGE (see below) is specified. If <i>language</i> contains more than one language code, <i>new-language</i> must contain the same numbers of language codes. Each <i>language</i> language code is replaced by the language code in the corresponding position of <i>new-language</i> .	

Keyword	Values	Restricted to Command
	If <i>language</i> is not specified, <i>new-language</i> must not contain more than one language code.	
NEWLANGUAGE	Up to 8 valid language codes (for example, code 4 for Spanish) for new user-defined error messages. This option does not apply to the long texts of Natural system error messages for which English (language code 1) is the only valid language. See also LANGUAGE above.	
DATE	An object date. You can add a time by inserting a blank between date and time. For the format and ranges allowed, see Date and Time in <i>Name, Date and Time Specification</i> .	
NEWDATE	A new object date. NEWDATE can be a date followed by a time value. You can add a time by inserting a blank between date and time. See also Date and Time in <i>Name, Date and Time Specification</i> .	
USERID	The user ID to be checked if NEWUSERID is specified.	
NEWUSERID	A new user ID.	
TID	Only applies to objects on mainframes. The terminal ID to be checked if NEWTID is specified.	
NEWTID	Only applies to objects on mainframes. A new terminal ID.	
PATH	The path name to be checked if NEWPATH is specified.	
NEWPATH	A new path name.	



Notes:

1. Parameters not applicable to the selection criterion processed are ignored.
2. LOADFNAT . . . , LOADFUSER . . . and LOADNCP . . . are used for the LOAD command only, and ignored otherwise.
3. LOADFNAT . . . is used for libraries starting with SYS (except SYSTEM).
4. LOADFUSER . . . is used for libraries not starting with SYS (but including SYSTEM).
5. LOADNCP . . . is used for Natural command processor sources.

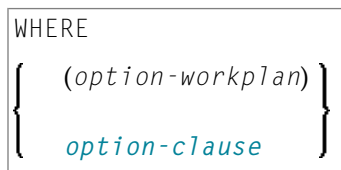
40 option-setting

- Syntax of option-setting 258
- Keyword Explanation of option-setting 260

The *option-setting* clause is used to change the default values of Object Handler command options.

The syntax that applies to the *option-setting* clause is shown and explained in the following section. The keywords and variable values contained in the syntax diagrams shown represent the parameters that are used to specify the default values. If indicated, a variable value must be supplied with a keyword.

Syntax of option-setting

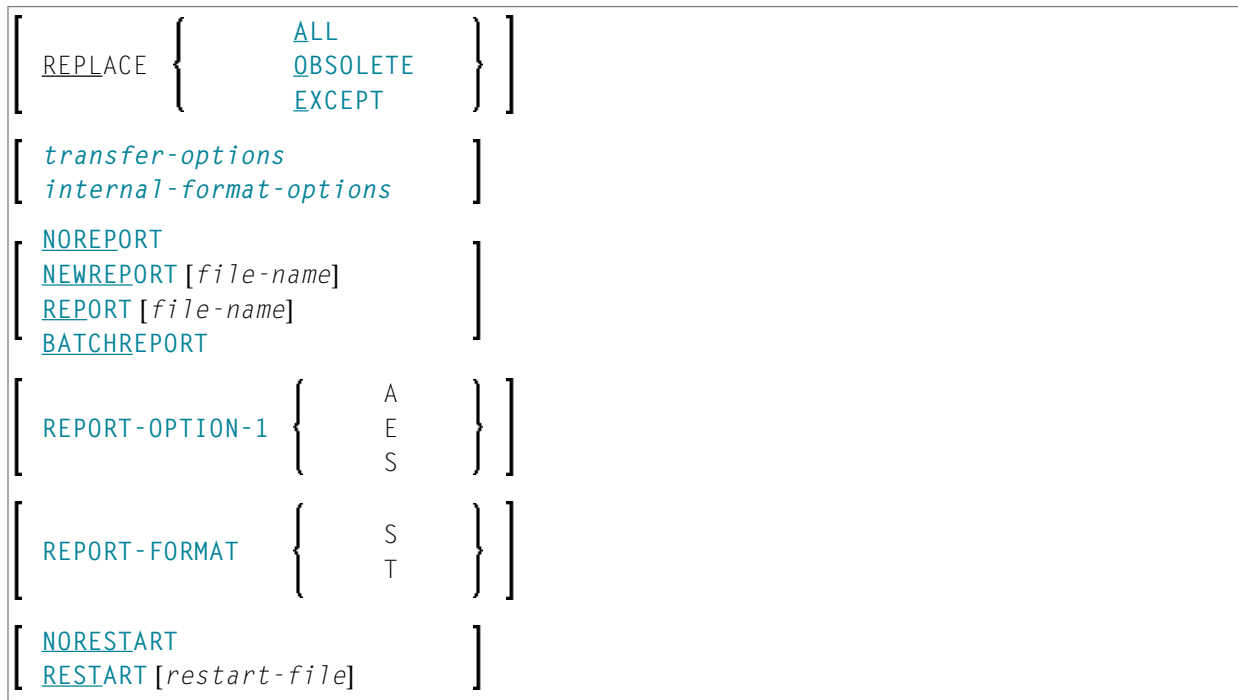


The syntax diagram that applies to *option-workplan* is shown and described in [Referencing Workplans](#) in the section *Workplans*.

The syntax of the *option-clause* is shown in the following section.

- [Syntax of option-clause](#)

Syntax of option-clause



```

[NUMBERPROCESS number]
[FIXEDLENGTH]
[FDIC (dbid,fnr,password,cipher)]
[FSEC (dbid,fnr,password,cipher)]
[ USE-FDDM { YES } ]
[ USE-FDDM { NO } ]
[ { NEWWORKFILE } file-name [ { WORKFILETYPE } { DEFAULT } ] ]
[ { WORKFILE } [ { WETYPE } { PORTABLE } ] ] ]
[ { UTF-8 } ] ] ]
[ADAFDUWORKFILE file-name]
[ { WORKFILELOCATION } { PC } ]
[ { WFLOC } { SERVER } ] ]

```

Separators

Commas must be used as separators between the values following the FDIC and FSEC keywords, or if a value is missing. For example: FDIC (10,21,,2a).

If the session parameter ID (see *ID - Input Delimiter Character* in the *Parameter Reference* documentation) has been set to a comma, use a slash (/) as the separator between values.

transfer-options

```

TRANSFER
[ CONVERSION-TABLE { SYSTEM-TABLE } ]
[ CONVERSION-TABLE { USER-TABLE } ]
[ CONVERSION-TABLE { [conversion-program] } ] ]
[SUBSTITUTE]
[INCLUDE-LINE-NUMBERS]
[UPPERCASE-TRANSLATION]
[INCORPORATE-FREE-RULES]
[LOAD-CODE-PAGE code-page-name]
[DA-FORMAT data-area-format]
[UNSHAPE]

```

internal-format-options

<pre> XREF { ON { OFF { DOC { FORCE { SPECIAL </pre>
[DELETEALLOWED]
[NOSYMBOLTABLE]
[VERSIONCHECK]

Keyword Explanation of option-setting

The keywords and the variable values (if relevant) of *option-setting* are explained in the following section:

Option	Explanation	Restricted to Command
REPLACE	Replaces existing objects according to the option specified: ALL All objects (default setting). OBSOLETE All objects with a date older than the date of the object in the load file. EXCEPT All objects except those with a date newer than the date of the object in the load file.	LOAD LOADALL
TRANSFER	Set Transfer mode. The data is read and written in Transfer format. For valid options, see <i>Keyword Explanation of transfer-options</i> .	UNLOAD LOAD SCAN
NOREPORT	Specifies the report file setting: No data is recorded to a report file. This is the default setting for the FIND and FINDLIB commands.	
NEWREPORT	Specifies the report file setting: Report data is recorded and written to Work File 4 or <i>file-name</i> . An existing file will be overwritten.	
REPORT	Specifies the report file setting: Report data is recorded and written to Work File 4 or <i>file-name</i> . This is the default setting for the commands UNLOAD, LOAD, LOADALL, SCAN, SCANALL and DELETE.	

Option	Explanation	Restricted to Command
BATCHREPORT	<p>Not applicable in a remote Natural Development Server environment.</p> <p>Specifies the report setting for batch processing or when using the OBJHAPI Application Programming Interface:</p> <p>Report data is either written to SYSOUT or output on the screen respectively (report data is <i>not</i> written to a file).</p>	
REPORT-OPTION-1 or REPOPT1 or REP-OPT-1	<p>Specifies the report option to be used when a direct command is executed and a report is to be written:</p> <p><u>A</u> Display all report items (default).</p> <p><u>E</u> Display only error messages. This includes messages from Natural Security and messages that have incurred during the execution of a LOAD command, for instance “not replaced”.</p> <p><u>S</u> For batch mode only. The error report is split into two parts: Report items except error messages are written to the default report device (REPORT(0)/CMPRINT), error messages (including messages from Natural Security and messages that have incurred during the execution of a LOAD command) are written to the second report device (REPORT(1)/CMPRT01). Note that in online mode S has the same effect as A.</p>	UNLOAD LOAD SCAN DELETE
REPORT-FORMAT or REPFMT	<p>Specifies the report format to be used when a direct command is executed and a report is to be written:</p> <p><u>S</u> Display the source data in the unload report, i.e. the data of the unloaded object before parameters (e.g. the library name) are changed (default setting).</p> <p><u>T</u> Display the target data in the unload report, i.e. the data after parameters (e.g. the library name) have been changed.</p>	UNLOAD
NORESTART	No restart information is written to a file.	LOAD
RESTART	Restart information is written to Work File 6 or <i>restart-file</i> .	LOAD
NUMBERPROCESS	<p>Specifies the number of objects to be processed.</p> <p>The LOAD or SCAN command stops execution after the number specified.</p>	LOAD SCAN
FIXEDLENGTH	<p>Sets the format of the unload work file to a maximum record length of fixed size.</p> <p>Every data record contains 256 bytes if written in internal format, or 100 bytes in Transfer format.</p>	UNLOAD
FDIC	<p>Specifies the system file FDIC to be used for processing:</p> <p>the database ID (<i>dbid</i>), file number (<i>fnr</i>), password (<i>password</i>) and cipher code (<i>cipher</i>) of the Adabas file.</p>	UNLOAD LOAD DELETE

Option	Explanation	Restricted to Command
	If no values (or 0) are specified, the current FDIC system file is used.	
FSEC	Specifies the system file FSEC to be used for processing: the database ID (<i>dbid</i>), file number (<i>fnr</i>), password (<i>password</i>) and cipher code (<i>cipher</i>) of the Adabas file. If no values (or 0) are specified, the current FSEC system file is used.	UNLOAD LOAD DELETE
USE - FDDM	Specifies that the FDDM system file is used for processing: see Keyword Explanation of USE-FDDM below.	UNLOAD LOAD FIND DELETE
NEWWORKFILE or WORKFILE	Specifies the work file to be used. The UNLOAD or LOAD data is transferred into/from Natural Work File 1. If NEWWORKFILE is specified, the data overwrites the contents of the existing work file or fills a new work file from the top. Otherwise, the data is appended.	UNLOAD LOAD SCAN
WORKFILETYPE or WFTYPE	Not required by the LOAD and SCAN commands, which automatically choose the appropriate work file type and ignore this keyword if specified. The work file type of Natural Work File 1 when data is read and written in internal format: DEFAULT Default binary work file. PORTABLE Portable work file. UTF-8 Unicode/UTF-8 encoded binary work file. UTF-8 only applies to the unload function and if TRANSFER is specified. If UTF-8 is specified, you cannot use the options CONVERSION-TABLE, SUBSTITUTE and INCORPORATE-FREE-RULES. (See also Work File Format in <i>Work Files</i> .) If WORKFILETYPE has not been specified, the current type is used.	UNLOAD LOAD SCAN
ADAFDUWORKFILE	The complete path name assigned to the work file (Natural Work File 5) into which Adabas FDT data is loaded.	LOAD
WORKFILELOCATION or WFLOC	Only applies to remote environments. Specifies the location of the unload, load or scan work file when using Object Handler functions in connection with SpoD (Single Point of Development). Valid input values are:	UNLOAD LOAD SCAN

Option	Explanation	Restricted to Command
	<p>SERVER The work file is located on the server, in the remote environment. This is the default setting.</p> <p>PC The work file is located in a local Windows directory, on the client.</p>	

The keywords and the variable values (if relevant) of *transfer-options* and *internal-format-options* are explained in the following section:

- [Keyword Explanation of transfer-options](#)
- [Keyword Explanation of internal-format-options](#)
- [Keyword Explanation of USE-FDDM](#)

Keyword Explanation of transfer-options

When using the TRANSFER keyword, you can specify the following options:

Option	Explanation	Restricted to Command
CONVERSION-TABLE	<p>Converts data processed in Transfer format by using either of the following conversion tables:</p> <p>SYSTEM-TABLE: The internal Natural conversion table.</p> <p>USER-TABLE: A user-defined conversion table if <i>conversion-program</i> has been specified. This program must be stored in the library SYSOBJH or one of its steplibs; see the example programs OTNCONAE and OTNCONEA in the library SYSOBJH.</p> <p>If no <i>conversion-program</i> is specified, the corresponding conversion table is used in NATCONV.INI ([ISO8859_1->EBCDIC] or [EBCDIC->ISO8859_1]).</p>	UNLOAD LOAD SCAN
SUBSTITUTE	<p>Replaces line references by labels during the unload in Transfer format.</p> <p>This option only applies if your source-code line numbers are used for statement references. If so, the line numbers of referenced lines and the line number references are replaced by labels. The sources are not modified in the database.</p>	UNLOAD
INCLUDE-LINE-NUMBERS	<p>Transfers line numbers during the unload in Transfer format. By default, line numbers in Natural objects are <i>not</i> unloaded.</p>	UNLOAD

Option	Explanation	Restricted to Command
UPPERCASE - TRANSLATION	<p>Translates any source code into upper case during the load in Transfer format.</p> <p>By default, source code in Natural objects is <i>not</i> translated.</p>	LOAD
INCORPORATE - FREE - RULES	<p>Incorporates source text of Predict free rules associated with a map into a map source during the unload in Transfer format if Predict is installed.</p>	UNLOAD
LOAD - CODE - PAGE	<p>Specifies the code page to be used for converting object sources encoded in Unicode/UTF-8 (Universal Transformation Format, 8-bit form).</p> <p>If you use this option, all object sources unloaded into a work file in UTF-8, will be converted with the specified code page when they are loaded into a work file.</p> <p>If you specify *CODEPAGE as <i>code-page-name</i>, the value assigned to the system variable *CODEPAGE is used (see the <i>System Variables</i> documentation).</p> <p>If <i>code-page-name</i> is not specified, the source objects are converted with the code page used when unloading them.</p> <p>If LOAD - CODE - PAGE is specified, you cannot use the options CONVERSION - TABLE and UPPER CASE - TRANSLATION.</p>	LOAD LOADALL
DA - FORMAT	<p>Specifies format conversion of data area sources.</p> <p>Possible values are:</p> <ul style="list-style-type: none"> N Converts data areas to the new internal data area format. 0 Converts data areas to the old internal data area format. * Does not convert data areas. This is the default. <p>See also the data area conversion options described in Transfer in Settings - Options.</p>	UNLOAD LOAD
UNSHAPE	<p>Replaces shaped Arabic characters (code page IBM420) with the corresponding unshaped characters in the following source objects: Natural programs , user error messages and system error messages.</p>	UNLOAD LOAD

Keyword Explanation of internal-format-options

When using *internal-format-options*, you can specify the following:

Option	Explanation	Restricted to Command		
XREF	<p>Only applies if Predict is installed.</p> <p>Loads or unloads XRef data of cataloged Natural objects. You can specify one of the following values:</p>	LOAD UNLOAD		
	<table border="1"> <tr> <td>ON</td> <td> <p>UNLOAD: Unloads cataloged objects and their cross-reference data (if any).</p> <p>LOAD: Loads cataloged objects and their cross-reference data if cross-references exist in the work file.</p> </td> </tr> </table>		ON	<p>UNLOAD: Unloads cataloged objects and their cross-reference data (if any).</p> <p>LOAD: Loads cataloged objects and their cross-reference data if cross-references exist in the work file.</p>
	ON		<p>UNLOAD: Unloads cataloged objects and their cross-reference data (if any).</p> <p>LOAD: Loads cataloged objects and their cross-reference data if cross-references exist in the work file.</p>	
	OFF		No XRef data is processed. This is the default.	
	DOC		<p>Only applies to LOAD.</p> <p>Loads cataloged objects only if Predict entries exist for the objects in the FDIC system file.</p>	
	FORCE		<p>Only applies to LOAD.</p> <p>Loads cataloged objects and their cross-reference data only if cross-references exist in the work file and if Predict entries exist for the objects in the FDIC system file.</p>	
	SPECIAL		<p>Only applies to LOAD.</p> <p>Loads cataloged objects and their cross-reference data (if any).</p>	
DELETEALLOWED	Processes delete instructions from work files when loading objects in internal format.	LOAD		
NOSYMBOLTABLE	<p>Only applies to objects on mainframes.</p> <p>Unloads cataloged Natural library objects without their corresponding internal Natural symbol tables.</p> <p>This will reduce the amount of disk storage required. However, this is only useful for a production environment, as several application development</p>	UNLOAD		

Option	Explanation	Restricted to Command
	functions which require the symbol tables will then not be available; in addition, the profile parameter RECAT=ON (see the <i>Parameter Reference</i> documentation) will not apply.	
VERSIONCHECK	<p>Only applies to objects on mainframes.</p> <p>Checks the Natural version of the cataloged object to be loaded. The Natural version under which the objects were cataloged and written to the work file is compared with the current Natural version. Objects cataloged under a Natural version higher than the current one will be rejected.</p> <p>VERSIONCHECK can only be used if data is loaded in internal format, that is, if the TRANSFER option is <i>not</i> specified.</p>	LOAD

Keyword Explanation of USE-FDDM

Only applies when processing Natural library objects on UNIX, OpenVMS or Windows platforms.

Specifies that the FDDM system file is used for processing.

If the FDDM file has been activated in the NATPARM module, the default setting is YES.

The following applies when specifying the values YES or NO:

Value	Explanation
YES	<p>UNLOAD, FIND and DELETE:</p> <p>If the parameter NATTYPE is set to V, DDMs are only processed from the library SYSTEM located in the FDDM file or the file specified by the database ID (DBID) and the file number (FNR).</p> <p>No DDMs are processed if the parameter NATTYPE is set to *, or if NATTYPE is a combination of any Natural object types that does not include the type V.</p> <p>LOAD:</p> <p>DDMs are loaded into the library SYSTEM located in the FDDM file.</p> <p>See also NATTYPE in Natural Library Object and DDM Selection in <i>select-clause</i>.</p>
NO	<p>UNLOAD, FIND and DELETE:</p> <p>DDMs are processed from the libraries specified.</p> <p>LOAD:</p> <p>DDMs are loaded into the libraries specified.</p>

41

Examples of Using Direct Commands

▪ Unloading Objects for the Same Platform	268
▪ Unloading Objects for Different Platforms	269
▪ Loading Objects in Internal Format	269
▪ Loading Objects in Transfer Format	270
▪ Batch Processing in a Remote Environment	270

This section provides examples for using Object Handler direct commands.



Tip: For additional examples, you can view the command generated for an Object Handler function. This command is automatically displayed when you use a wizard. In advanced-user mode, you can activate the display of the command by either entering the Object Handler command `SET ADVANCEDCMD ON` or setting the parameter `Display-Cmd-in-Advanced-Mode` to `Y (Yes)` in the Object Handler profile (see also [Profile Settings](#)).

Unloading Objects for the Same Platform

This section contains examples of how to unload objects in internal format to a work file in order to load them on the same platform, within either a local mainframe, UNIX, OpenVMS or Windows environment:

- Unload all Natural programming objects (source objects only) from library ABC:

```
UNLOAD * LIB ABC OBJTYPE N SCKIND S
```

- Unload all Natural programming objects (cataloged objects only) from library ABC:

```
UNLOAD * LIB ABC OBJTYPE N SCKIND C
```

- Unload all Natural programming objects (cataloged objects and source objects) from library ABC:

```
UNLOAD * LIB ABC OBJTYPE N SCKIND A
```

- Unload all Natural programming objects (source objects only) from library ABC to load in library ABCNEW:

```
UNLOAD * LIB ABC OBJTYPE N SCKIND S WITH NEWLIBRARY ABCNEW
```

- On a mainframe: Unload all DDMs whose names start with EMP and which point to database 88:

```
UNLOAD EMP* LIB * OBJTYPE D DDMDBID 88
```

- On UNIX, OpenVMS or Windows: Unload all DDMs whose names start with EMP and which point to database 88:

```
UNLOAD EMP* LIB * OBJTYPE N NATTYPE V DDMDBID 88
```

- On UNIX, OpenVMS or Windows: Unload all DDMs whose names start with EMP from library VLIB to load in library VLIBNEW:


```
UNLOAD EMP* LIB VLIB OBJTYPE N NATTYPE V WITH NEWLIBRARY VLIBNEW
```

- Unload all user-defined error messages from library `ERRLIB` to load in library `NEWERR`:

```
UNLOAD * LIB ERRLIB OBJTYPE E SLKIND A WITH NEWLIBRARY NEWERR
```

- On Windows: Unload all Natural programming objects (cataloged objects and source objects) from library `ABC` to a portable work file on a PC:

```
UNLOAD * LIB ABC OBJTYPE N WHERE WORKFILE C:\WF1.SAG WORKFILETYPE PORTABLE
```

or

```
UNLOAD * LIB ABC OBJTYPE N WHERE WORK C:\WF1.SAG WFT P
```

Unloading Objects for Different Platforms

This section contains command examples of how to unload objects in Transfer format to a work file in order to load them on a different platform such as unloading in a mainframe and loading in a UNIX, an OpenVMS or a Windows environment.

- Unload all Natural programming objects (source objects only) from library `ABC`:

```
UNLOAD * LIB ABC OBJTYPE N WHERE TRANSFER
```

- Unload all Natural programming objects (source objects only) and user-defined error messages from library `ABC`:

```
UNLOAD * LIB ABC WHERE TRANSFER
```

- Unload all Natural programming objects (source objects only) from library `ABC` with fixed record length:

```
UNLOAD * LIB ABC OBJTYPE N WHERE TRANSFER FIXEDLENGTH
```

Loading Objects in Internal Format

This section contains command examples of how to load objects from a work file in internal format.

- Load all objects to library `LIBNEW` and replace any that already exist:

```
LOADALL WITH NEWL LIBNEW WHERE REPLACE ALL
```

- Load all object with target library `TGTLIB` to the new target library `NEWTGT`:

```
LOAD * LIB TGTLIB WITH NEWLIBRARY NEWTGT
```

- Load the user-defined error messages 1000 to 1500 from library `ERRLIB` only:

```
LOAD * LIB ERRLIB OBJTYPE E FMNUM 1000 TONUM 1500
```

Loading Objects in Transfer Format

This section contains command examples of how to load objects from a work file in Transfer format.

- Load all objects to library `LIBNEW` and replace any that already exist:

```
LOADALL WITH NEWL LIBNEW WHERE TRANSFER REPLACE ALL
```

- Load all object with target library `TGTLIB` to new target library `NEWTGT`:

```
LOAD * LIB TGTLIB WITH NEWLIBRARY NEWTGT WHERE TRANSFER
```

Batch Processing in a Remote Environment

You can use direct commands to unload objects in batch mode from a remote Natural Development Server (NDV) environment or load objects in batch into a remote NDV environment.

The examples in this section illustrate the use of direct commands in batch to transfer objects from one remote NDV environment to another.

- [Input Commands in CMSYNIN File](#)
- [Input Data in CMOBJIN File](#)
- [Explanation of File Contents](#)

Input Commands in CMSYNIN File

```
MAP ENVIRONMENT=UX1 SUNNAT63 6312 SAG  
SYSOBJH  
UNMAP  
MAP ENVIRONMENT=MF1 IBM2 4742 SAG  
SYSOBJH  
UNMAP  
FIN
```

Input Data in CMOBJIN File

```
UNLOAD * LIB SAG-TEMP %
WHERE TRANS WFLOC PC WORK D:\NAT-Work\w1.dat REPORT
SHOW STATISTICS
END
LOADALL WHERE TRANS WFLOC PC WORK D:\NAT-Work\w1.dat %
REPLACE ALL REPORT
SHOW STATISTICS
END
```

Explanation of File Contents

<pre>MAP ENVIRONMENT=UX1 SUNNAT63 6312 SAG</pre>	<p>Maps to an NDV environment on a UNIX or an OpenVMS platform.</p>
<pre>SYSOBJH</pre>	<p>Invokes the Object Handler (on the Windows client) that receives the following three commands from the input data in the CMOBJIN file:</p> <pre>UNLOAD * LIB SAG-TEMP % WHERE TRANS WFLOC PC WORK D:\NAT-Work\w1.dat REPORT</pre> <p>Causes the Object Handler to unload all objects from library SAG-TEMP in the remote UNIX or OpenVMS environment into the work file contained in the local Windows directory <i>D:\NAT-Work\w1.dat</i>.</p> <pre>SHOW STATISTICS</pre> <p>Writes statistical data about the unloaded objects to the CMPRINT output file.</p> <pre>END</pre> <p>Terminates the Object Handler.</p>
<pre>UNMAP</pre>	<p>Unmaps the NDV environment on the UNIX or OpenVMS platform.</p>
<pre>MAP ENVIRONMENT=MF1 IBM2 4742 SAG</pre>	<p>Maps to an NDV environment on a mainframe platform.</p>
<pre>SYSOBJH</pre>	<p>Invokes the Object Handler (on the Windows client) that receives the following three commands from the input data in the CMOBJIN file:</p> <pre>LOADALL WHERE TRANS WFLOC PC % WORK D:\NAT-Work\w1.dat REPLACE ALL REPORT</pre> <p>Causes the Object Handler to load all objects from the work file in the local Windows directory <i>D:\NAT-Work\w1.dat</i> into the remote mainframe environment.</p> <pre>SHOW STATISTICS</pre> <p>Writes statistical data about the loaded objects to the CMPRINT output file.</p> <pre>END</pre>

Examples of Using Direct Commands

	Terminates the Object Handler.
UNMAP	Unmaps the NDV environment on the mainframe platform.
FIN	Terminates the Natural batch session.

42

Batch Condition Codes and User Exit Routines

- Condition Codes Returned in Batch 274
- Applying User Exit Routines 274
- User Exit Routines Available 275

This section describes the condition codes returned for Object Handler functions in batch mode and the user exit routines available for function processing.

Condition Codes Returned in Batch

Object Handler processing in batch mode terminates with one of the following condition codes:

Condition Code	Explanation
0	Object Handler process terminated successfully.
30	An internal Object Handler error occurred.
40	An error was detected in the Object Handler command.
50	An error occurred during Object Handler processing.
60	A Natural Security error occurred during Object Handler processing.
99	A Natural error occurred during Object Handler processing.

Applying User Exit Routines

The Object Handler user exit routines are supplied as source objects in the Natural system library SYSOBJH. These source objects are named SRC-EX nn , where nn denotes the number of the user exit routine.

▶ To activate a user exit routine

- CATALOG or STOW source object SRC-EX nn under the name OBJHEX nn in the Natural system library SYSOBJH.

Different names are used to guarantee that the source object (possibly modified according to your requirements) and the cataloged object of the user exit routine are not overwritten by an update installation.

For detailed descriptions of the user exit routines, see the source objects of SRC-EX nn in the library SYSOBJH.

User Exit Routines Available

The following user exit routines are available:

- [OBJHEX01 for Processing Failures](#)
- [OBJHEX02 for Object Rejection](#)

OBJHEX01 for Processing Failures

Whenever a condition code is set to a value greater than 0 (zero) in batch mode, the user exit routine OBJHEX01 (if available) will be invoked before the Object Handler stops processing. With this user exit routine, you can specify whether to continue or terminate Object Handler processing. In the case of termination, you can change the condition code. For further details, see the source of the user exit routine SRC-EX01 in the Natural system library SYSOBJH.

OBJHEX02 for Object Rejection

If the Object Handler load function was executed successfully in batch mode (with Condition Code 0) or in online command mode, but one or more objects were rejected during loading (for example, not replaced), before the Object Handler stops processing, the user exit routine OBJHEX02 (if available) is invoked. With OBJHEX02, you can specify whether to continue or terminate Object Handler processing. In the case of termination, you can set a condition code. For further details, see the source of the user exit routine SRC-EX02 in the Natural system library SYSOBJH.

43 Tools

▪ Status	278
▪ Last Result	278
▪ Traces	278
▪ Reports	279
▪ Transfer Work File	279

The Object Handler provides special features in the **Tools** menu to display status information and reports, check and modify trace settings and transfer work files to and from remote environments.

Status

Displays the Object Handler functions currently used, the user environment, the Workplan library and the current usage of Work Files 1 to 15.

▶ To display the status

- From the **Tools** menu, choose **Show Status**.

Last Result

Displays the last internal command issued by the processing interface of the Object Handler and possible return codes and messages.

▶ To display the last result

- From the **Tools** menu, choose **Last Result**.

Traces

Activates or deactivates the trace function. Traces record internal Object Handler program flows to provide control information for error diagnoses. The trace option is set off by default.

▶ To change the setting

- From the **Tools** menu, choose **Trace Setting**.

▶ To maintain permanent trace files in remote environments

- From the **Actions** menu, choose **Change Workplan Library** and proceed as described in [Remote Environments](#) in *Change Workplan Library* in the section *Administration*.

Reports

Lists the objects processed with the unload, load, scan or find function, and records errors that may interrupt processing. The report option is set on by default and is displayed after the unload, load, scan or find function has been executed.

▶ **To display the contents of the latest report file**

- From the **Tools** menu, choose **Show Report File**.

▶ **To change the settings in local environments**

- 1 From the **Options** menu, choose **Settings** and the tabbed page **General**.
- 2 In the **Settings** window, on the tabbed page **Options**, select the option button **Use additional options** and choose the **Set** button.
- 3 Choose the tabbed page **General**.

For possible settings, see [General](#) in the section *Settings - Options*.

▶ **To maintain permanent report files in remote environments**

- From the **Actions** menu, choose **Change Workplan Library** and proceed as described in [Remote Environments](#) in *Change Workplan Library* in the section *Administration*.

Transfer Work File

Only applicable to remote environments located on mainframe platforms.

Transfers the contents of work files from local environments to mainframe servers and vice versa. The work files must be of corresponding types. See also [Work File Format](#) in the section *Work Files*.

44 Options

- Settings 282
- Profile 282
- Advanced User 282
- Free Format Editing 282
- Details 283
- Single Objects 283
- Display Command 283

The Object Handler provides the **Options** menu to invoke functions and/or activate (deactivate) modes and options. Note that the items on the **Options** menu do not apply to all Object Handler functions, and, therefore, may not be available on all **Options** menus. Additionally, the availability of items depends on the processing mode used (advanced user or wizards) and the status of function processing.

Settings

The **Settings** option invokes the **Settings** window where you can specify option settings for the unload, load or scan function and parameter settings for the unload or load function. For further information, see the section [Settings](#).

Profile

This option is available in the **Options** menu in the **Welcome to the Natural Object Handler** window.

The **Profile** option invokes the **SYSOBJH - Profile** window where you can define an individual profile for your Object Handler utility. See also the section [Profile Settings](#).

Advanced User

This option is available in the **Options** menu in the **Welcome to the Natural Object Handler** window.

The **Advanced User** option activates function processing in advanced-user mode as described in [Advanced User](#) in the section *Functions*.

Free Format Editing

With the **Free Format Editing** option activated, an edit area is supplied for creating a new Workplan: see [New Workplan](#) in *Administration* in the section *Functions*.

Details

This option only applies to functions executed in advanced-user mode.

The **Details** option activates the selection criteria specified in the appropriate **Details** window of objects to be processed. For further information, see the section [Object Specification](#).

Single Objects

This option applies to the result lists generated by Object Handler functions in advanced-user mode. It does not apply to the scan function.

With the **Single Objects** option activated, source objects (Src) and cataloged objects (Gp) are listed in separate table rows.

Display Command

This option applies to functions executed in advanced-user mode. It does not apply to the view function.

With **Display Command** activated, the Object Handler command generated for a function is displayed before the function is executed. This provides you the option to cancel the operation. In addition, you can save the command as Workplan of the type PROCEDURE as described in the section [Workplans](#).

45 Profile Settings

You can define an individual profile for your Object Handler utility by modifying the standard Object Handler user profile and specifying general or user-defined parameter settings, such as activating advanced-user mode by default, or specifying default settings for work files, report files and Workplans.

▶ **To invoke the profile option**

- 1 In the **Welcome to the Natural Object Handler** window, from the **Options** menu, choose **Profile**.

The **SYSOBJH - Profile** window appears which provides all functions required for maintaining individual profiles.

- 2 Select the option(s) required and choose the **OK** command button to save your modifications.

The default is that the settings of the previous Object Handler session are used at session startup.

The Object Handler profile is stored as text file *SYSOBJH.PRU* in the following directory of the Natural Configuration Utility: *Local Configuration File\Installation Assignments\<Path to profile parameters>*.

46 Migration from SYSTRANS to the Object Handler

- Converting Individual Commands 288
- Processing SYSTRANS Commands with OBJHAPI 289
- Unsupported SYSTRANS Options 289

You can migrate from the old utility SYSTRANS to the Object Handler by using the two methods described in this section.

Converting Individual Commands

You can convert SYSTRANS direct commands to the corresponding Object Handler direct commands by using the Object Handler direct command provided for migration. This migration command automatically converts the command syntax used by SYSTRANS to the command syntax used by the Object Handler.

► To convert a single command

- 1 Use the following Object Handler direct command:

```
SYSTRANS
```

followed by a SYSTRANS direct command.

The specified SYSTRANS command is converted to the corresponding Object Handler command.

- 2 Specify any subsequent command for the Object Handler in the syntax that applies to the SYSTRANS utility.

The syntax of SYSTRANS remains valid for the duration of the Object Handler session.

Example of a SYSTRANS Command:

The following is an example of two consecutive SYSTRANS utility commands and their corresponding Object Handler commands.

Old SYSTRANS commands:	TRANSCMD EXECUTE UNLOAD N FROM LIB1 NAME ETID END
New Object Handler command:	SYSOBJH SYSTRANS EXECUTE UNLOAD N FROM LIB1 NAME ETID END
Subsequent Object Handler command in SYSTRANS syntax:	END

Example of SYSTRANS Batch Processing:

The following is an example of processing a SYSTRANS utility command in batch by using map input data, and the corresponding Object Handler command and input data.

Old SYSTRANS batch sequence:

```
SYSTRANS
U
N,N,N,Y,N,N,N,N
N
SRCLIB1,PGM1,*,TGTLIB1
```

New Object Handler batch sequence:

```
SYSOBJH SYSTRANS
U
N,N,N,Y,N,N,N,N
N
SRCLIB1,PGM1,*,TGTLIB1
```

Processing SYSTRANS Commands with OBJHAPI

You can use the OBJHAPI Application Programming Interface (supplied in the Natural system library SYSOBJH) to execute an Object Handler command in the syntax of the SYSTRANS utility.

If you use OBJHAPI for this purpose, you have to specify the parameter `P-EXTENSIONS-EXEC-SYSTRANS-CMD` in the program that invokes OBJHAPI. For details, see the example program DOC-API supplied in the library SYSOBJH.

Unsupported SYSTRANS Options

The Object Handler does not support the following SYSTRANS direct command options: `WORK-FILE-INPUT`, `SPECIAL-CONVERSION`, `RULE-LOAD` and `UNLOAD-RULES`.

VIII

SYSAPI Utility - APIs of Natural Add-On Products

47

SYSAPI Utility - APIs of Natural Add-On Products

- Prerequisites 294
- Invoking and Terminating SYSAPI 294
- SYSAPI Tree View Items 295
- Performing SYSAPI Utility Functions 296

The utility SYSAPI is used to locate and test Application Programming Interfaces (APIs) provided by Natural add-on products such as Entire Output Management. This can be done either in a local Windows environment or in a remote environment located on a Windows, a UNIX, an OpenVMS or a mainframe platform.

The API of a Natural add-on product is a Natural subprogram (cataloged object) that is used for accessing and possibly modifying data or performing services that are specific to an add-on product or a subcomponent.

The API of a Natural add-on product is supplied in the Natural library and/or system file provided for objects that are specific to a particular Natural add-on product. For instructions on using the API of a Natural add-on product, refer to the documentation of the respective add-on product.

For each API of a Natural add-on product, the utility SYSAPI provides a functional description, one example program and API-specific keywords.

Related Topics:

- *SYSEXT - Natural Application Programming Interfaces - Utilities* documentation

Prerequisites

- The appropriate Natural add-on product must be installed at your site.
- The version of the Natural add-on product installed must support the SYSAPI utility features.
- The **Enable Plug-ins** option must be selected. This option is selected by default. For details, see *Workspace Options* in the section *Setting the Options* in the *Using Natural Studio* documentation.

Invoking and Terminating SYSAPI

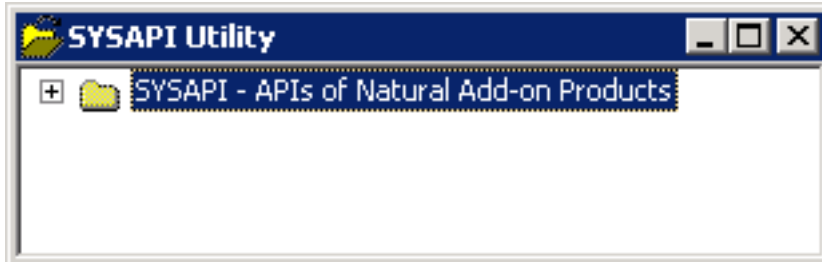
This section provides instructions for invoking and terminating the SYSAPI utility.

▶ To invoke SYSAPI

- Enter the following system command:

```
SYSAPI
```

When invoking SYSAPI, the plug-in for the utility SYSAPI is activated and the SYSAPI utility window appears with the root node **SYSAPI - APIs of Natural Add-on Products** as shown in the example below:



▶ **To restart SYSAPI**

- If you want to restart SYSAPI during the current Natural session, as an alternative to the start method mentioned earlier, from the toolbar, choose the following icon:



This icon appears after initially invoking SYSAPI, when the plug-in for the utility SYSAPI is activated.

▶ **To terminate SYSAPI**

- Choose the standard Windows close function.

SYSAPI Tree View Items

This section describes the structure of the tree in the SYSAPI utility window and the nodes and items contained in the tree view.

Starting with the **SYSAPI - APIs of Natural Add-on Products** root node at the top tree level you can expand the following node hierarchy:

- [Natural Add-On Product Node](#)
- [API Group Node](#)

- [API Node](#)

Natural Add-On Product Node

Each Natural add-on product installed at your site has its own node, which contains one or more subordinate API group nodes.

API Group Node

Each API group node represents a particular API feature provided for the Natural add-on product of the parent node. An API group node contains all API nodes that relate to the particular feature.

API Node

An API node name consists of the name of the API subprogram and a brief description of its functional purpose. API nodes are sorted by API names.

An API node contains the following API-specific items:

Item	Explanation
Keywords	All keywords relevant to the API. See also the functions Select Keyword and List All Keywords .
Description	A text object or program that contains a description of the API. The description comprises purpose, function and calling conventions of the API and keywords relevant to the API.
Example Program	An example program or subprogram of how to invoke the API.

Performing SYSAPI Utility Functions

The SYSAPI utility can be used to perform operations on API-specific text objects (descriptions) and example programs or functions on API groups such as finding APIs relevant to a current task by specifying keywords.

Object operations include functions such as List, Open and Execute, which correspond to the standard functions available when maintaining or executing a Natural object of the type text or program. These functions can be performed by using the context menu associated with each object. For details of these functions, refer to the relevant sections in the *Using Natural Studio* documentation.

The section below describes the functions that can be performed on an API group:

- [Select Keyword](#)
- [List All Keywords](#)

- Refresh

Select Keyword

This function is used to list all API nodes of a selected API group to which a specified keyword applies.

▶ To list APIs by keyword

- 1 Select the API group node of a Natural add-on product, open the context menu and choose **Select Keyword** or press `SHIFT+K`.

The **Select Keyword** window appears.

- 2 From the drop-down list box, select a keyword as shown in the example of a **Select Keyword window** in the SYSEXT utility documentation.
- 3 Choose the **OK** button.

The keyword is indicated in the node name of the selected API group.

- 4 Expand the API group node.

A list of all API nodes to which the specified keyword applies appears for the selected API group. The window looks similar to the corresponding window of the SYSEXT utility (see the **example** in the SYSEXT utility documentation).

- 5 If required, to return to the complete list of all API nodes available in the API group node (default setting), in the **Select Keyword** window, select the asterisk (*).

List All Keywords

This function is used to list all keywords available for a selected API group.

▶ To list all keywords of an API group

- 1 Select an API group node, open the context menu and choose **List All Keywords** or press `SHIFT+A`.

The **All Keywords** window appears where the root node indicates the name of the Natural add-on product and the name of the selected API group.

The window looks similar to the **All Keywords window** of the SYSEXT utility (see the example in the SYSEXT utility documentation).

- 2 Expand the root node.

A list of all keywords available for the selected API group appears. You can expand the keyword nodes and display all API nodes to which the keyword applies.

Refresh

This function updates the current SYSAPI environment settings. This can be required, for example, when you change the language code (system variable *LANGUAGE) to adapt API descriptions and keywords to another language if the Natural add-product supports different languages for its API descriptions and keywords.

▶ To refresh SYSAPI environment settings

- 1 Select the root node **SYSAPI - APIs of Natural Add-on Products**, open the context menu and choose **Refresh** or press SHIFT+R.

A **Refresh** window appears.

- 2 Choose the **OK** button to confirm the refresh or choose **Cancel** to abort the operation.

IX

SYSCP Utility - Code Page Information

48 SYSCP Utility - Code Page Information

- Invoking and Terminating SYSCP 302
- All Code Pages 304
- Unicode Properties 308
- General Information 309

The SYSCP utility is used to obtain information on code pages available in the current Natural Windows environment.

This helps avoid problems that can occur when a code page is not defined in Natural or when source objects are converted to an incorrect code page or Unicode format.

For detailed information on how Natural supports Unicode and code pages and Unicode-specific items, see the descriptions and presentations in the *SYSEXV Utility* and *Related Topics* below.

Related Topics:

- *Unicode and Code Page Support: Natural* documentation
- Unicode: Unicode Consortium at web site at <http://www.unicode.org/>
- ICU: IBM ICU Documentation at web site <http://www-01.ibm.com/software/globalization/icu/index.jsp>
- IBM Converter Explorer documentation at web site <http://demo.icu-project.org/icu-bin/convexp>

Invoking and Terminating SYSCP

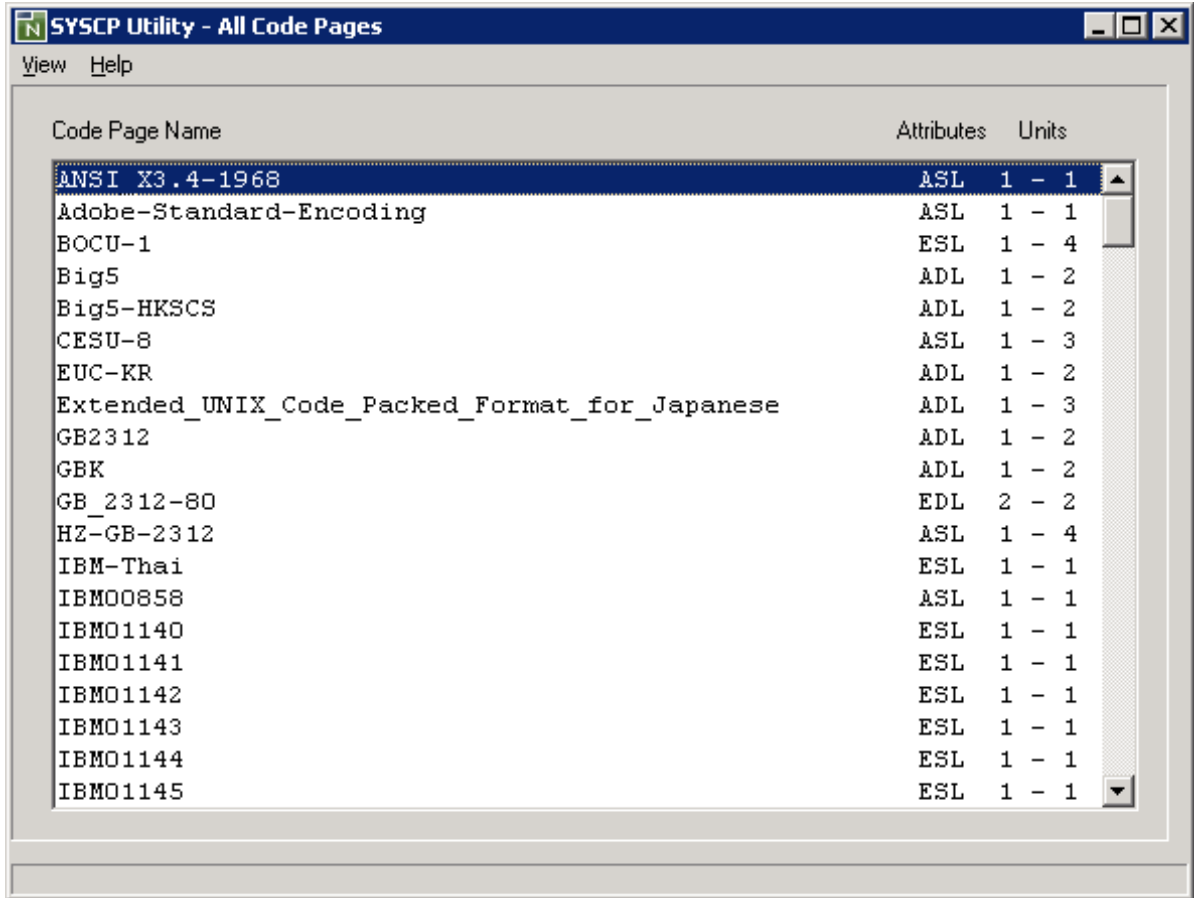
Instructions for invoking and terminating the SYSCP utility are provided in the following section.

▶ To invoke the SYSCP utility

- Enter the following system command:

```
SYSCP
```

A **SYSCP Utility - All Code Pages** window similar to the example below appears:



Code Page Name	Attributes	Units
ANSI X3.4-1968	ASL	1 - 1
Adobe-Standard-Encoding	ASL	1 - 1
BOCU-1	ESL	1 - 4
Big5	ADL	1 - 2
Big5-HKSCS	ADL	1 - 2
CESU-8	ASL	1 - 3
EUC-KR	ADL	1 - 2
Extended_UNIX_Code_Packed_Format_for_Japanese	ADL	1 - 3
GB2312	ADL	1 - 2
GBK	ADL	1 - 2
GB_2312-80	EDL	2 - 2
HZ-GB-2312	ASL	1 - 4
IBM-Thai	ESL	1 - 1
IBM00858	ASL	1 - 1
IBM01140	ESL	1 - 1
IBM01141	ESL	1 - 1
IBM01142	ESL	1 - 1
IBM01143	ESL	1 - 1
IBM01144	ESL	1 - 1
IBM01145	ESL	1 - 1

This window is opened by the **All Code Pages** function, which is executed by default when you invoke the SYSCP utility.

All SYSCP utility functions are provided in the **View** menu. They are explained in the remainder of this documentation.

The **Help** menu provides online help information about the SYSCP utility.

▶ To terminate SYSCP

- Choose the standard Windows close function.

Or:

From the **View** menu, choose **Exit**.

All Code Pages

This function opens the **SYSCP Utility - All Code Pages** window, which provides a list of all code pages available in your current Natural Windows environment. The list is sorted in ascending order by code page name.

The columns contained in the window and the options provided for each code page listed are described in the following section:

- [Columns of Code Page List](#)
- [Context Menu Options](#)

Columns of Code Page List

The columns contained in the list of code pages are described in the following section:

■ Code Page Name

This column lists the IANA name of the code page.

The IANA (Internet Assigned Numbers Authority) name is the standard and unambiguous name of the code page. The IANA name is used by Natural as the default code page name (see the *CP* profile parameter described in the *Parameter Reference* documentation) for conversions to and from Unicode. The IANA name is returned by the `*CODEPAGE` system variable (see the *System Variables* documentation).

If no IANA name is specified, the internal ICU name is listed instead. This is indicated by `ICU Name :`, which is used as a prefix for all ICU names.

You can use the **Show All Names** option of the context menu to display all code page names specified for a code page.

■ Attributes

This column displays a three-letter code, which represents the attributes of the code page: see the **Show Attributes** option of the context menu for details.

■ Units

This column indicates the code units (minimum and maximum numbers of bytes) assigned to the code points.

Context Menu Options

For the code page selected, you can use the context menu to obtain further information on the code page or to test code-point assignments of characters.

▶ To open the context menu

- 1 From the code page list, select the code page required.
- 2 Click the right mouse button.

Or:

Press `SHIFT+F10`.

The context menu appears.

The options provided in the context menu are described in the following section:

- [Show Attributes](#)
- [Show All Names](#)
- [Test Conversion](#)

Show Attributes

This option can be used to display the attributes of the code page selected. Attributes are character sets and languages supported by the code page. Attributes and codes that represent each attribute are listed in the following table:

Code	Attribute
A	ASCII character set.
E	EBCDIC character set.
D	Double-byte or multi-byte character set for languages such as Japanese and Chinese.
S	Single-byte character set.
R	Right-to-left direction.
L	Left-to-right direction for bi-directional languages such as Arabic and Hebrew.

Show All Names

This option can be used to display all names that can be specified for the code page selected:

- **IANA name**

If not specified, this field is blank.

- **ICU name**

- **Alias names**

One or more alternate names used for the code page.

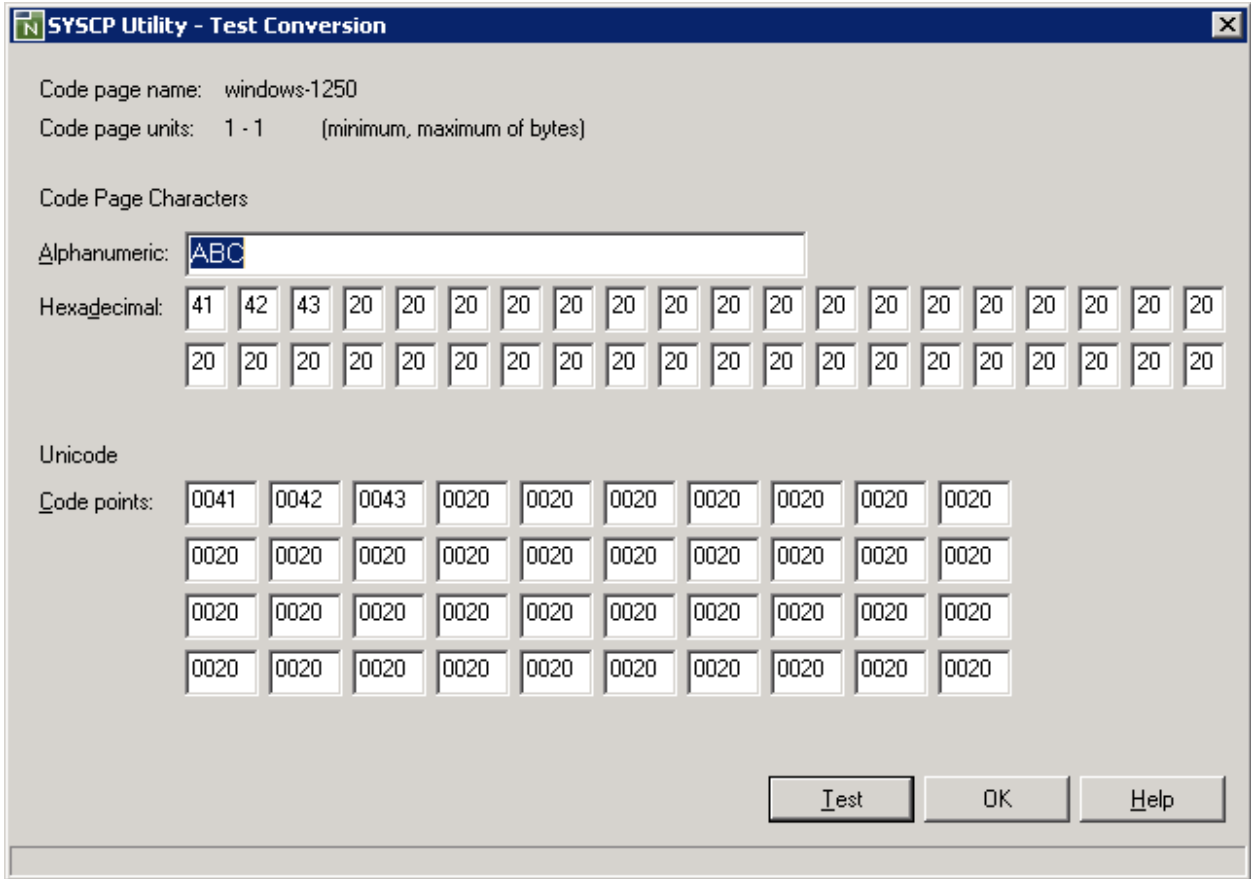
Test Conversion

You can test code-point conversion from a selected code page to the default code page (value of *CODEPAGE) defined with the CP profile parameter:

- from an alphanumeric character string to Unicode code points and vice versa, or
- from hexadecimal values to Unicode code points and vice versa.

The test conversion dialog box of a code page (here: windows-1250) shown in the example below contains the following information:

- the code page name,
- the byte units (minimum and maximum numbers of bytes) assigned to the code points,
- an alphanumeric character string and its equivalent hexadecimal values and
- the corresponding Unicode code points.



► **To convert a character or code point**

- 1 Enter a literal string in the **Alphanumeric** box.

Or:

Enter hexadecimal values in one or more **Hexadecimal** boxes.

Or:

Enter Unicode code points in one or more **Unicode** boxes.

The boxes not used for input are deactivated.

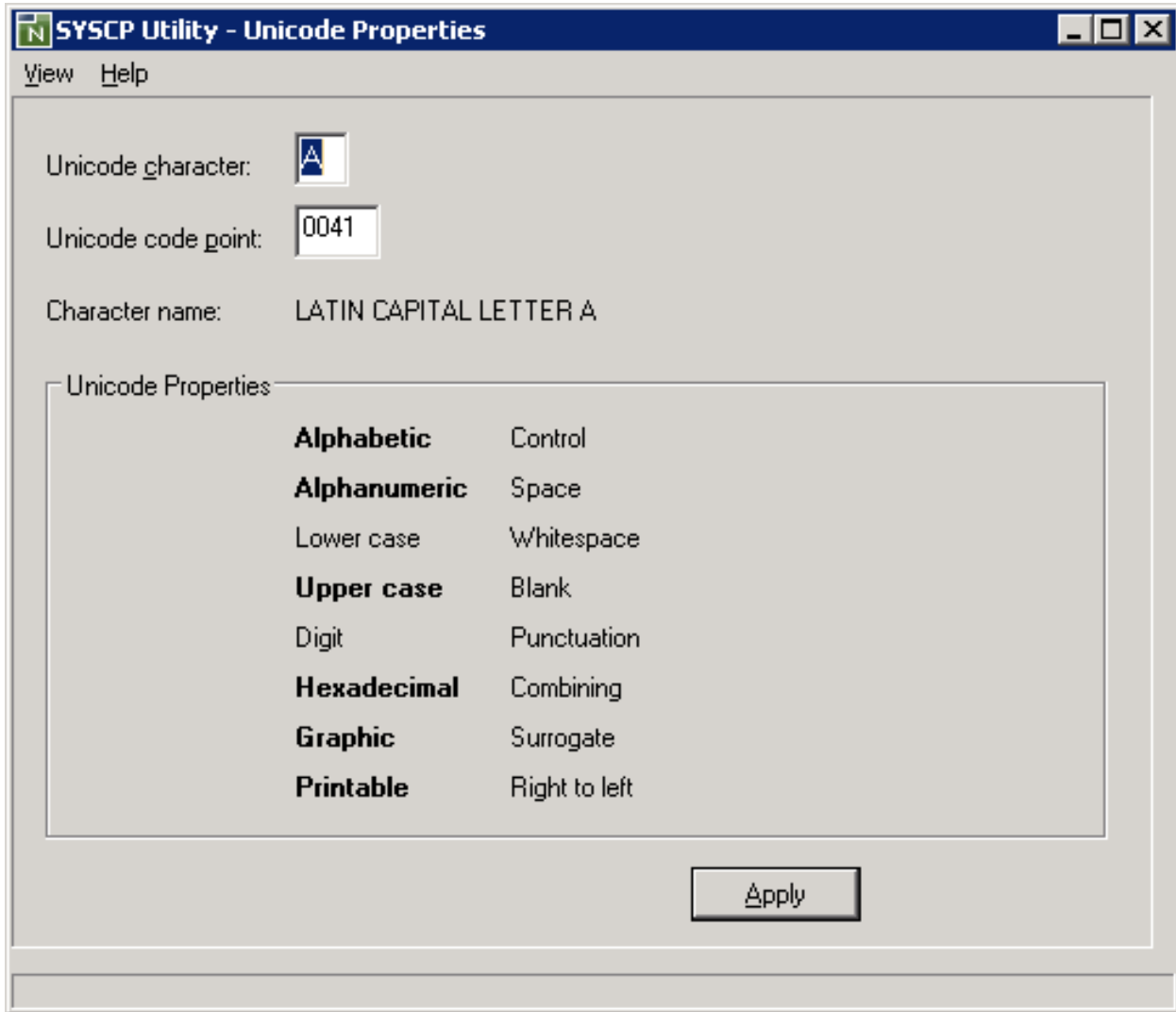
- 2 Choose **Test** or press ENTER.

The value entered in one of the boxes is converted to its equivalent code points or literal string.

- 3 Choose **OK** to close the **Test Conversion** dialog box.

Unicode Properties

This function is used to display the Unicode properties for a Unicode character or a Unicode code point as shown in the example of the letter A below:



The **Unicode Properties** function can also be used to view the code point assigned to a character.

► To display the code-point assignment and Unicode properties

- 1 In the **Unicode character** box, enter the character whose code point and properties you want to view.

Or:

In the **Unicode code point** box, enter the code point whose character and properties you want to view.

- 2 Choose **Apply** or press ENTER.

The value entered in one of the boxes is converted to its equivalent character or code point whose Unicode properties are displayed in the window.

A Unicode property shown in boldface letters denote that this property applies to the character such as **Alphabetic** in the example above.

For explanations of the Unicode character properties displayed in the window, refer to Unicode Consortium's documentation *Unicode Character Database* at web site <http://www.unicode.org/Public/4.1.0/ucd/UCD.html>.

General Information

This function provides information on the current ICU and Unicode versions.

X **SYSERR Utility**

When you develop a Natural application, you may want to separate error or information messages from your Natural code and manage them separately. This makes it easy for you, for example, to standardize messages, to have predefined message ranges for different types of message, to translate messages into other languages or to attach to a message a long text that explains it in more detail.

The SYSERR utility provides the option to write application-specific messages. In addition, you can use the SYSERR utility to customize the texts of the existing Natural system messages.

General Information on Messages

Invoking SYSERR

SYSERR Utility Window and Functions

Converting Natural System Short Messages

Generating Message and Text Files

Managing Messages in Different Libraries

Application Programming Interface USR0020P

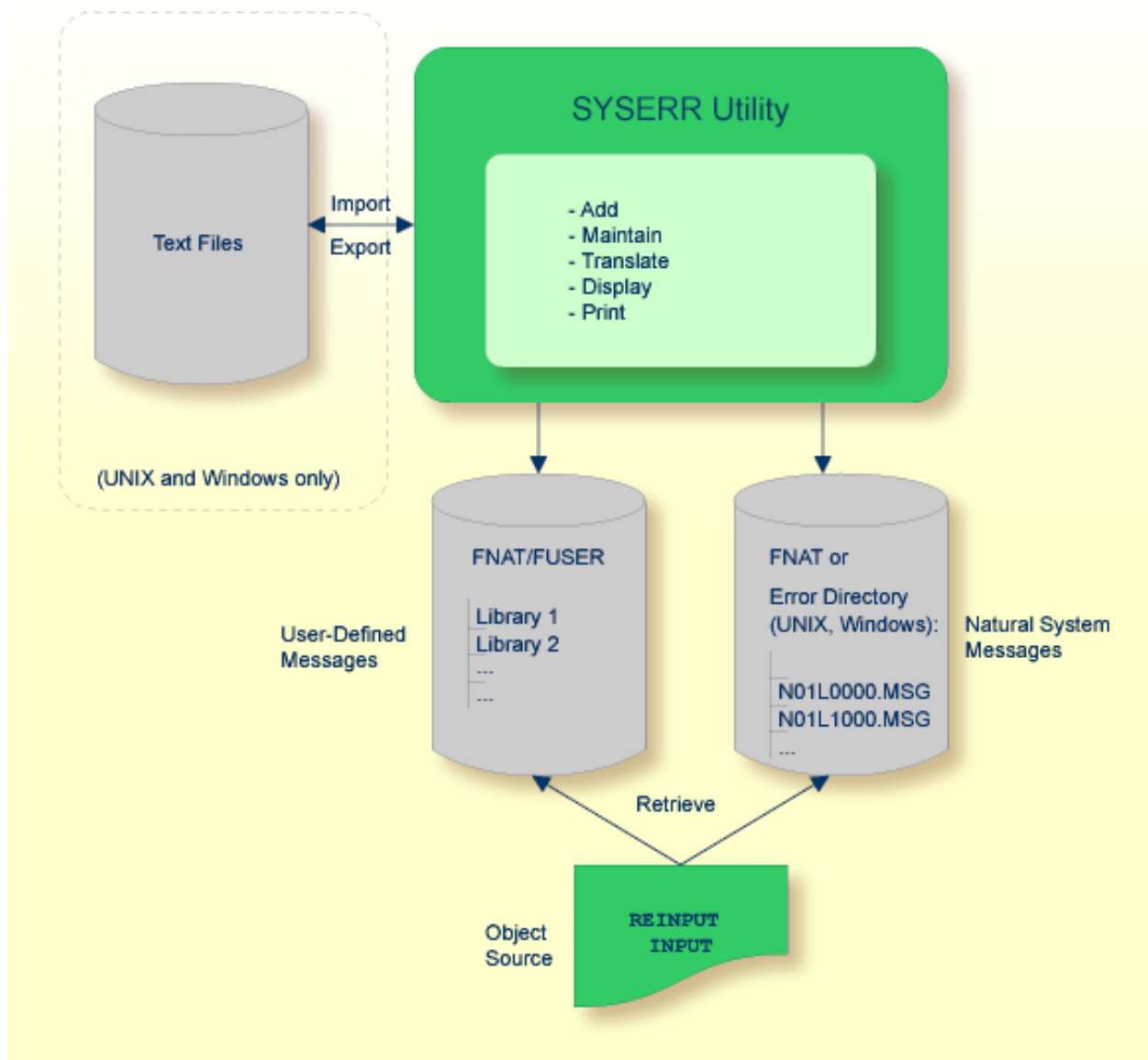
49

General Information on Messages

- Message Types 315
- Message Languages 315
- Issuing Messages 316
- Retrieving Natural System Short Messages 317
- Retrieving User-Defined Short Messages 317
- Obtaining Message Information 318

This section contains information on the types of message and message languages that can be managed with the SYSERR utility and how messages are issued and retrieved in your Natural system environment.

The following graphic illustrates the features of the SYSERR utility and how messages are processed within Natural:



Message Types

There are two types of message: Natural (system) messages and user-defined messages:

Natural system messages are issued by the Natural nucleus and Natural utilities. Natural system messages are delivered by Software AG and stored as message files in the Natural Err directory. Natural system messages begin with `NAT`, followed by a four-digit number, for example, `NAT0230`.

User-defined messages are issued by applications written by a user. User-defined messages are stored as message files in libraries (including `SYS`-libraries) in the system file `FUSER` or `FNAT`.


A message can be translated into different languages. Each language is stored in a separate message file. A maximum of 9999 messages can be stored per library and message file.

There are four types of message text:

- Natural system short message
- Natural system long message
- User-defined short message
- User-defined long message

A short message is the one-line message which is displayed in the message line when the corresponding error situation occurs.

A long message is a detailed explanation of the corresponding short message and includes instructions for solving problems.

 **Caution:** Keep in mind that any modifications of Natural system messages can result in wrong messages or a loss of modifications when a new Natural version is released.

Message Languages

Messages can be created in up to 60 languages as described for the system variable `*LANGUAGE` in the *System Variables* documentation.

The following rules and restrictions apply:

- Natural system short messages must be entered in English first, and can then be translated into any other language.
- Natural system long messages can be entered in English, but cannot be translated into other languages.

- User-defined short messages can be entered in any language, and then be translated into any other language.
- User-defined long messages can be entered in any language, but only if the corresponding short message in the same language already exists.

Issuing Messages

This section contains information on the Natural statements `INPUT` and `REINPUT` that are used to issue a Natural system short message or a user-defined short message in a Natural program.

▶ To issue a Natural system short message in a program

- Specify one of the following Natural statements:

```
INPUT WITH TEXT *-nnnn
```

or

```
REINPUT WITH TEXT *-nnnn
```

where *nnnn* is the number of the requested message (you can omit leading zeros).

▶ To issue a user-defined short message in a program

- Specify one of the following Natural statements:

```
INPUT WITH TEXT *nnnn
```

or

```
REINPUT WITH TEXT *nnnn
```

where *nnnn* is the number of the requested message (you can omit leading zeros).

Dynamic Replacement of Message Text

A message text can contain variable parts that are identified by the notation `:n:`, where *n* represents occurrences 1 to 7. These variable parts are replaced by a value at runtime.

For details, see *operand3* in the section *INPUT Syntax 1 - Dynamic Screen Layout Specification* and *operand3* in the section *REINPUT* in the *Statements* documentation.

Retrieving Natural System Short Messages

When a program references a Natural system short message, Natural looks for the requested message number in the Natural Err directory in the following order:

1. Under the current language code as determined by the system variable `*LANGUAGE`,
2. Under language code 1 (English).

If neither of the above is found, a program references a message that does not exist and you only receive the message number prefixed with `NAT`, for example, `NAT0230`.

Retrieving User-Defined Short Messages

When a program references a user-defined short message, Natural first looks for the requested message number *nnnn* under the current language code as determined by the system variable `*LANGUAGE` (see the *System Variables* documentation). If that message does not exist, Natural looks for the requested message number *nnnn* under language code 1 (English). If that message does not exist either, Natural looks for message number *n000* (where *n* is the first digit of the requested message number) under language code 1.

These three search steps are first performed in the current library. If nothing is found there, further libraries are searched in the same way until a corresponding message is found.

The sequence of libraries for the search is as follows:

1. The current library as determined by the system variable `*LIBRARY-ID`,
2. The steplib; if Natural Security is installed, the sequence in which the steplib are specified in the Natural Security profile of the current library,
3. The default steplib as determined by the system variable `*STEPLIB`,
4. The library `SYSTEM` in the system file `FUSER (*)`,
5. The library `SYSTEM` in the system file `FNAT (*)`.

(*) If the name of the current library begins with SYS, SYSTEM FNAT is searched before SYSTEM FUSER.

Obtaining Message Information

When you receive a short message, you may be looking for additional information on the problem situation.

- With the system command `HELP`, you can display Natural system long messages or user-defined long messages.
- With the system command `LASTMSG`, you can list the short text of the message(s) that occurred last and additional information on the error situation. The information displayed includes associated error messages that possibly preceded the last message.

Both commands are described in the *System Commands* documentation.

50 Invoking SYSERR

- Invoking SYSERR for User-Defined Messages 320
- Invoking SYSERR for Natural System Messages 322

This section describes alternative methods of invoking the SYSERR utility window **SYSERR - Error Messages**. From this window, you can execute all SYSERR functions available for creating and maintaining user-defined or Natural system messages. The window components and maintenance functions are explained in the section *SYSERR Utility Window and Functions*.

Invoking SYSERR for User-Defined Messages

User-defined message files are stored in a library. You can invoke the SYSERR utility either for a list of all messages that exist for a library and the current language code defined for your Natural environment, or messages that exist for a particular language code. In the **Logical View** of the Natural Studio tree view, the messages are stored per language in the library subnode **Error Messages**.

▶ **To invoke SYSERR for messages of the current language in a library**

- 1 From the Natural Studio tree view, select the required library node.

Or:

Issue the following system command:

```
LOGON library-ID
```

where *library-ID* is the name of the required library.

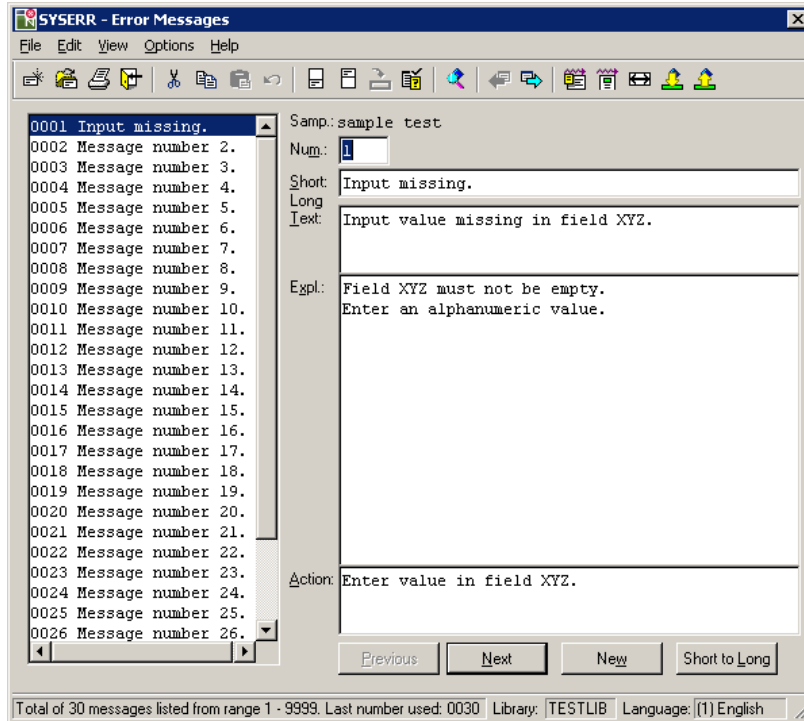
- 2 From the **Tools** menu, choose **Development Tools and Error Messages**.

Or:

Issue the following system command:

```
SYSERR
```

A **SYSERR - Error Messages** window similar to the example below appears:



The window contains a list of all message files stored in the specified library for the current language code.

If no message file exists for the specified library, the window is empty.

▶ **To invoke SYSERR for messages of a particular language in a library**

- 1 Expand the tree node of the required library in the **Logical View**. The messages are listed per language in the **Error Messages** subnode.
- 2 In the **Error Messages** subnode, double-click on the language required.

Or:

Select the required language, invoke the context menu by pressing **SHIFT+F10**, and choose **Open**.

The **SYSERR - Error Messages** window appears with the list of messages that exist for the specified language.

Invoking SYSERR for Natural System Messages

Natural system messages cannot be accessed directly from the Natural Studio tree view or with a Natural system command. You need to invoke the SYSERR utility for any library first and then choose a SYSERR menu function as described in the following instructions.

▶ To invoke SYSERR for Natural system messages

- 1 From the Natural Studio tree view, select the required library node.

Or:

Issue the following system command:

```
LOGON library-ID
```

where *library-ID* can be the name of *any* library.

- 2 From the **Tools** menu, choose **Development Tools and Error Messages**.

Or:

Issue the following system command:

```
SYSERR
```

- 3 From the **File** menu, choose **Open Lib/Lang**.

A **SYSERR - Open Lib/Lang** dialog box appears as shown in the example in the section *File Menu*.

- 4 From the **Library** drop-down list box, select **<natsys>** or remove the library ID and leave the box blank, and, from the **Language** box, select a language.
- 5 Choose **OK** to confirm your selection.

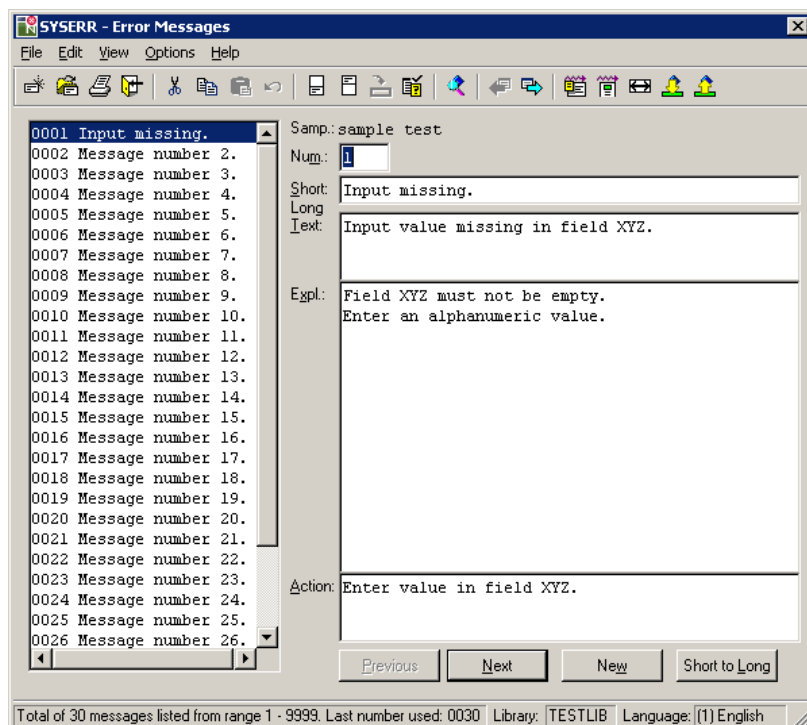
The **SYSERR - Error Messages** window appears with the list of all Natural system messages.

51

SYSERR Utility Window and Functions

▪ List Box	325
▪ Fields	325
▪ Command Buttons	326
▪ File Menu	327
▪ Edit Menu	329
▪ View Menu	331
▪ Options Menu	335
▪ Toolbar Buttons	338
▪ Status Bar	339
▪ Online Help	340

From the **SYSERR - Error Messages** window, you can invoke all SYSERR utility functions available for creating and maintaining user-defined or Natural system messages:



The window is divided into two sections that contain data on messages available in the current library:

- The left section with a **list box** displaying all messages available for the current language.
- The right section displaying the **fields** with the text of the current message. The current message is the message selected (highlighted) in the list box.

If no message exists for the specified library, both sections are empty.

The **SYSERR - Error Messages** window provides menus and **command buttons** for executing SYSERR utility functions. As an alternative to menus and command buttons, you can execute most of the functions by using **toolbar buttons**.

There are context menus available for several elements in the list box and the dialog boxes provided with the SYSERR functions. To invoke the context menu for an element, click on the element required with the right mouse button or select the element and press SHIFT+F10. The commands available are either cut and paste functions or correspond to the commands in the menus or to command buttons.

Some menu items are used to switch between modes or set a status. The check mark next to a menu item indicates which mode or status is active.

List Box

The list box appears in the left section of the **SYSERR - Error Messages** window. It contains the short messages of one language for one library. The short texts are preceded by the message number. The messages are sorted by the message number in ascending order.

If a message file contains more than 200 messages, for performance reasons, not all messages are read in one step. When initially opening the **SYSERR - Error Messages** window, up to 200 messages are read from the message file at once, and about 30 of them are displayed in the list box. Scrolling down the list with the vertical scroll bar, the next 200 messages are read and displayed as soon as the scroll bar reaches the bottom of the list box.

Choose the **Read All** menu option, to read all remaining messages in one step as described in *Edit Menu*.

The selected (highlighted) short message is the current message. It is displayed in the right section of the **SYSERR - Error Messages** window. There, you can modify the short and long texts of the current message. See also *Fields*.

Fields

The following fields appear in the right section of the **SYSERR - Error Messages** window:

Field	Explanation
Samp.	Output field that appears above the Num. field if a sample message exists for the current language and library. Samp. displays the text of the sample message. To create and use a sample message, see Sample in <i>Options Menu</i> .
Num.	Displays the number of the current message. It corresponds to the selected message in the list box. To display another message or to create a new one, replace the current number by another valid number. The maximum message number for a library and language is 9999. The message number 0000 is not allowed.
Short	Modifiable field displaying the short message text of the current message number. The input of the short message is mandatory. A new message can only be saved if text has been entered in the Short field. Therefore, if no text is displayed for the current message, the message number is free and can be assigned to a new message. To extend the default field length, see Short Message Length 72 in <i>Options Menu</i> .

Field	Explanation
	<p>To copy sample messages into the Short field, see Sample in <i>Options Menu</i> and Copy in <i>Command Buttons</i>.</p> <p>To copy text from the Short into the Long field, choose Short to Long.</p>
Long	<p>Modifiable fields displaying the long message text of the current message.</p> <p>Long consists of three input areas:</p> <p>Text Extended version of the short message text.</p> <p>Expl. Further explanation of the message.</p> <p>Action The action to be taken to resolve a problem, if relevant.</p> <p>The input in Long is optional.</p> <p>To copy text from the Short into the Long field, choose Short to Long.</p>

Command Buttons

The following commands can be executed by using command buttons:

Command	Explanation
Copy	<p>This button is only visible if a sample message exists for the current language and library. Copy appears under the toolbar, in the right upper section of the SYSERR - Error Messages window.</p> <p>Copies the text of a sample message (see Sample in <i>Options Menu</i>) into the Short field (see <i>Fields</i>) of the current message.</p>
Previous / Add Previous / Update	<p>Toggles between Previous and Add, and Previous and Update depending on the status of the current message:</p> <p>Previous Scrolls from the current message to the previous message if no modification was made.</p> <p>Add Adds a new message to the message file.</p> <p>Update Saves the current message to the message file after modification.</p> <p>Note: You can only save a message if text was entered in the Short field (see <i>Fields</i>).</p>
Next / Reset	<p>Toggles between Next and Reset, depending on the status of the current message:</p> <p>Next Scrolls from the current message to the next message if no modification was made.</p> <p>Reset Resets modifications made to the current message and displays the original message.</p>

Command	Explanation
New	<p>Searches for the next free message number starting from the current message, and opens input fields to create a message for the number found. Free means that this message number is available and has not been assigned to another message of any language.</p> <p>The search direction is downwards by default. However, the default direction can change to upwards if a previous search was performed in an upward direction by using the alternative function New Message (see <i>Edit Menu</i>).</p> <p>Upwards searches for the next lower message number, downwards searches for the next higher message number from the current message.</p> <p>As an alternative to New or New Message, you can also create a new message by replacing the number in the Num. field.</p>
Short to Long	Copies the text of the Short field to the first line of the Long field.

File Menu

This section describes the functions available in the **File** menu.

Function	Explanation
New Lib/Lang	<p>Only applies to libraries and languages for which no messages exist.</p> <p>Opens a new library and/or adds a new language. See also <i>To switch libraries and/or languages</i>.</p>
Open Lib/Lang	<p>Only applies to libraries and languages for which messages already exist.</p> <p>Opens another library and/or changes the language.</p> <p>See also <i>To switch libraries and/or languages</i>.</p>
Open File	<p>This function does not apply for accessing data on mainframe servers.</p> <p>Selects an existing message file from a Natural Err directory (FNAT or FUSER) or from another directory.</p> <p>See also the section <i>Generating Message and Text Files</i>.</p>
Print	<p>Invokes the print function.</p> <p>A dialog box prompts you to:</p> <ul style="list-style-type: none"> ■ Enter the range of messages. ■ Mark the Long texts printout option if you want to print the long message in addition to the short message. ■ Specify layout parameters and select the output device: Printer or Source. The initial assignment is the default printer set by Windows.

Function	Explanation
	To display the output in the source work area, use the system command LIST. If required, use the system command SAVE and the Natural object type text to save the contents of the source work area. See also <i>To print all Natural system messages.</i>
Exit	Terminates the SYSERR utility.

▶ **To print all Natural system messages**

- 1 Perform the steps described in *Invoking SYSERR for Natural System Messages.*
- 2 From the **File** menu, select **Print**.

▶ **To switch libraries and/or languages**

- 1 From the **File** menu, select **New Lib/Lang** if no messages exist.

Or:

From the **File** menu, select **Open Lib/Lang** if messages exist.

A SYSERR - New Lib/Lang or SYSERR - Open Lib/Lang dialog box similar to the example below appears:



- 2 From the **Library** drop-down list box, select a library and a language.

In the example above, the new language code 6 (Dutch) is selected for the library TESTLIB. TESTLIB already contains messages but no messages for language code 6.

- 3 Choose **OK** to confirm your selection.

An empty **SYSERR - Error Messages** window appears for the specified library and language.

Or:

If the **Open Lib/Lang** menu option was selected, a list of messages appears that exist for the specified library and language.

Edit Menu

This section describes the functions available in the **Edit** menu.

Command	Explanation
Cut	Supported clipboard functions.
Copy	
Paste	
Undo	Standard edit functions.
Delete	
New Message	<p>Searches for the next free message number starting from the current message, and opens input fields to create a message for the number found. Free means that this message number is available and has not been assigned to another message of any language.</p> <p>Upwards or Downwards specifies the search direction: Upwards searches for the next lower message number, Downwards searches for the next higher message number from the current message. See also the alternative New command button.</p> <p>As an alternative to New Message or the New command button, you can also create a new message by replacing the number in the Num. field.</p>
Delete Selected	Removes all messages selected (highlighted) in the list box. A dialog box prompts you to confirm the action.
Delete All	<p>Removes all messages displayed in the list box. A dialog box prompts you to confirm the action.</p> <p>When all messages of a message file have been deleted, the message file is deleted too.</p>
Read All	<p>Reads all messages from a message file into the list box in one step. This command applies to message files with more than 200 messages.</p> <p>Otherwise, for performance reasons, only 200 messages are read by default when the SYSERR - Error Messages window is opened. Additional messages can be displayed by scrolling down the list box to the end or by using the Read All menu option.</p>
Translate	Supports the creation of messages for different languages. Applies to short messages only. See <i>To translate languages</i> below.

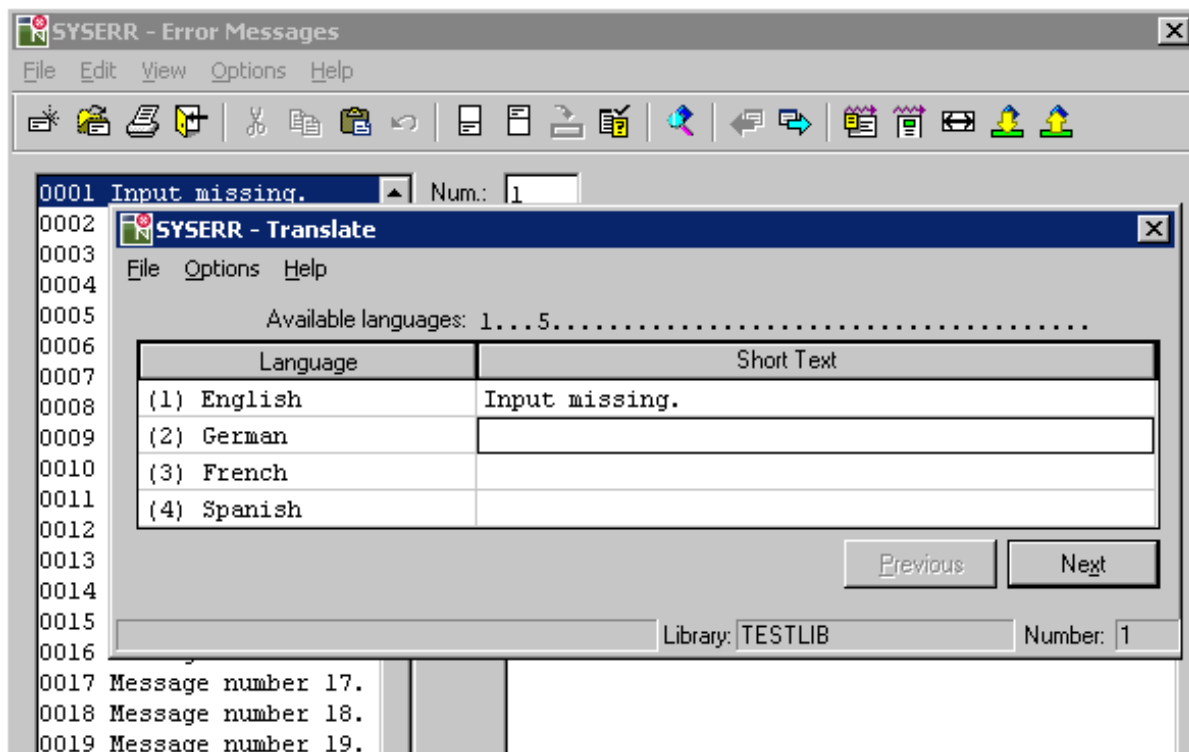
► To translate languages

- 1 From the **Edit** menu, choose **Translate**.

A **SYSERR - Select Languages** dialog box similar to the example below appears:



- 2 Select one language or more into which you wish to translate the current message and choose **OK**.
- 3 A **SYSERR - Translate** dialog box similar to the example below appears:



- 4 The dialog box always lists the current language of the short text (in the example above, English) in the first position followed by the new language(s) selected in **Step 1**.

The **SYSERR - Translate** dialog box provides the following menu options, command buttons and fields:

File > Exit	Closes the SYSERR - Translate dialog box and returns to the SYSERR - Error Messages window.
Options > Select Languages	Invokes a dialog box from which you can select one or more languages to be added to the current message number (unless selected earlier, during Step 1).
Available languages	The language code(s) of the language(s) already available for the current message number.
Language	The code and language of the current language and the new language(s) selected. The current language is always listed in the first position.
Short Text	The text of the short message.
Previous / Update	Same as described for Previous / Update in <i>Command Buttons</i> .
Reset / Next	Same as described for Reset / Next in <i>Command Buttons</i> .
Help > SYSERR Help	Displays help text on SYSERR utility functions.

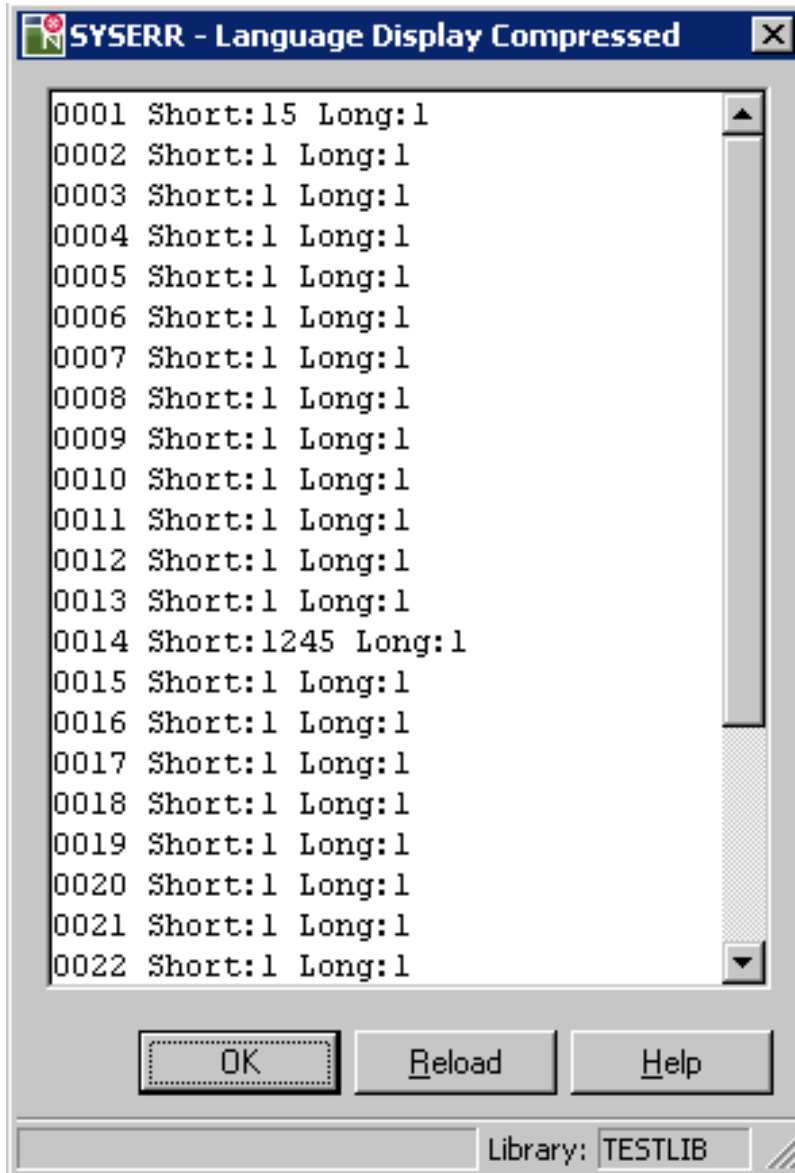
- 5 Enter the translation in the **Short Text** field of the relevant language(s).
- 6 Choose **Update**. The language code(s) of the new language(s) appears under **Available languages**.

View Menu

This section describes the functions available in the **View** menu.

Function	Explanation	
Languages Display	Provides an overview of the languages available for one message number. You can choose among the following views:	
	Compressed	Displays the language code(s) that exists for a short and long message. For short messages, the language codes are listed next to Short , for long messages next to Long . See also <i>Example of Compressed</i> .
	Extended Short	Displays the language code(s) that exists for a short message next to the message number.
	Extended Long	Displays the language code(s) that exists for a long message next to the message number.

Function	Explanation
	<p>You can keep the SYSERR - Language Display dialog box open while maintaining messages, and use the Reload command button any time you want to refresh the overview list.</p>
Filter Display	<p>Applies in connection with the Set Filter function.</p> <p>If Set Filter is enabled, the Filter Display function displays the short messages that match the filter criteria defined with the Set Filter function.</p> <p>If Filter Display is enabled, the filter criteria specified with the Set Filter function are displayed above the list box.</p> <p>To disable the Filter Display and list all messages, again choose the Filter Display option from the menu.</p> <p>See also <i>Filtering Messages</i>.</p>
Set Filter	<p>Defines a filter for displaying short messages and shows the filtered messages in the list box. See also <i>Filtering Messages</i>.</p> <p>Set Filter must be enabled in order to use the Filter Display function.</p>

Example of Compressed

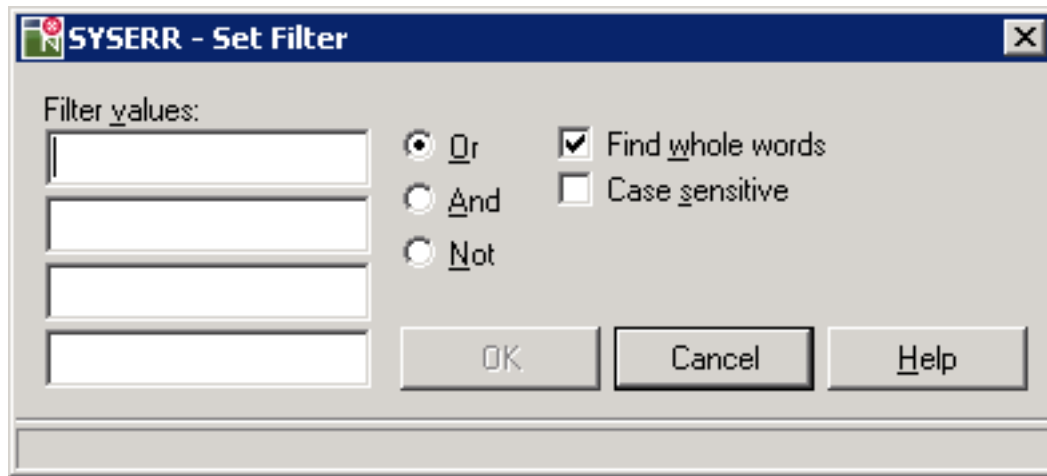
In the example above, the short text (**Short**) for message number 14 exists in English = (1), German (2), Spanish (4), and Italian (5). The long text (**Long**) for message number 14 exists in English (1).

Filtering Messages

▶ **To filter short messages**

- 1 From the **View** menu, choose **Set Filter**.

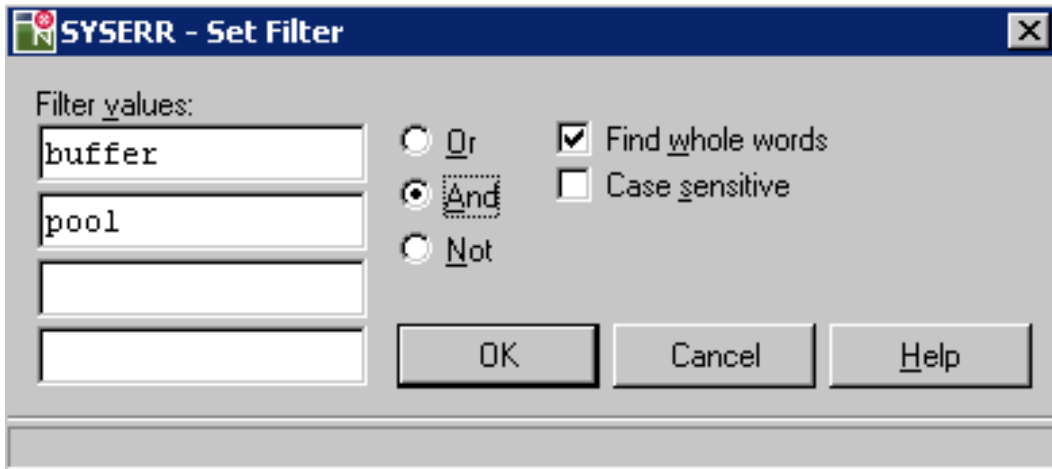
A **SYSERR - Set Filter** dialog box similar to the example below appears:



The dialog box provides the following filtering options:

Filter values	There are four fields in which you can enter the character string(s) to be used in the filter.	
Or / And / Not	You can select one of the following Boolean operators:	
	Or	The filter selects one or more of the character strings entered in Filter values . This is the default setting.
	And	The filter selects all of the character strings entered in Filter values .
	Not	The filter selects none of the character strings entered in Filter values .
	The operator is ignored if you only fill one of the Filter Value fields.	
Find whole words	If marked (default), the filter only allows searching for entire words and not parts of words.	
Case sensitive	If marked, the filter only allows searching for letters that exactly match the lower and/or upper case specified.	

- 2 Specify the criteria to be used for the filter:



In the example above, the filter selects all short messages that contain both the words `buffer` and `pool`.

- 3 Choose **OK**.

If the search criteria entered match the character strings contained in the short messages, the matches are displayed in the list box and the **Set Filter** and **Filter Display** functions are enabled.

- 4 To remove a filter and list all messages, choose **Filter Display** to disable the function.

If the **Set Filter** function is enabled, alternatively, you can execute the filter function by choosing **Filter Display**.

Options Menu

This section describes the functions available in the **Options** menu.

Function	Explanation						
Sample	<p>Invokes the SYSERR - Maintain Sample Message dialog box where you create a sample message to be used as a master for creating new short messages or replace the text of existing short messages.</p> <p>If you enter the string 0000 anywhere in the SYSERR - Maintain Sample Message dialog box (combined with text or not), the string 0000 is replaced by the number of the new message or the current message when copying the message.</p> <p>You can define one sample message for each language and library.</p> <p>If you created a sample message, the Samp. field appears with the text of the sample message.</p> <p>To copy the text of the sample message into the Short field, use the Copy command button (see <i>Edit Menu</i>). Alternatively, you can enter .C in the Short field if this field is empty.</p>						
Layout	See <i>Layout</i> .						
Shift Short Left	If enabled, automatically shifts the text of a short message to the left margin when adding a new message or choosing Update after modification.						
Short Message Length 72	<p>This option does not apply to a remote environment located on a UNIX, OpenVMS or mainframe platform.</p> <p>If enabled, the Short field (see <i>Fields</i>) is extended to a maximum input of 72 characters. The default field length is 65 characters, which is the maximum input in UNIX, OpenVMS and mainframe environments.</p>						
Size	Resizes the dialog box: <table border="1" data-bbox="354 1129 1380 1260"> <tr> <td data-bbox="354 1129 519 1171"></td> <td data-bbox="519 1129 1380 1171"></td> </tr> <tr> <td data-bbox="354 1171 519 1213">Startup</td> <td data-bbox="519 1171 1380 1213">Resizes the dialog box to the size it had at startup.</td> </tr> <tr> <td data-bbox="354 1213 519 1260">Full List Box</td> <td data-bbox="519 1213 1380 1260">Resizes the dialog box to display the short message in full length.</td> </tr> </table>			Startup	Resizes the dialog box to the size it had at startup.	Full List Box	Resizes the dialog box to display the short message in full length.
Startup	Resizes the dialog box to the size it had at startup.						
Full List Box	Resizes the dialog box to display the short message in full length.						
Confirm Window	Enables/disables a pop-up window to confirm: <ul style="list-style-type: none"> ■ The new message number. ■ That the text of the short message is copied into the first line of the Long field if Short to Long was chosen. ■ That the text of the sample message (if available) is copied into the Short field if Copy was chosen. 						

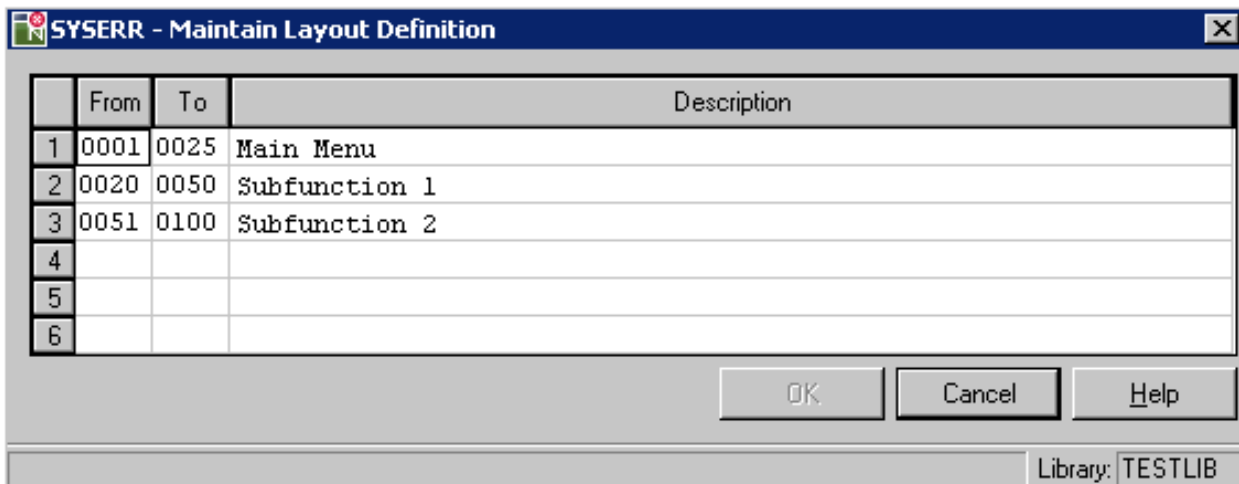
Function	Explanation		
Import Text File	<p>This function does not apply to a remote environment located on a mainframe platform.</p> <p>Imports a text file and converts it into a message file. Note that you always need to specify the full path of a file.</p>		
	<table border="1"> <tr> <td data-bbox="444 432 964 510">From (text file)</td> <td data-bbox="964 432 1474 510">The name of the text file from which the message file will be generated.</td> </tr> </table>	From (text file)	The name of the text file from which the message file will be generated.
	From (text file)	The name of the text file from which the message file will be generated.	
	<table border="1"> <tr> <td data-bbox="444 558 964 869">To (message file)</td> <td data-bbox="964 558 1474 869"> <p>The name of the message file into which the text file will be generated. Default is the full path name of the current library and message file.</p> <p>For user-defined messages, the file name must be <i>NnnAPMSL.MSG</i>. For Natural system messages, the file name must be <i>NnnLmmmm.MSG</i>.</p> </td> </tr> </table>	To (message file)	<p>The name of the message file into which the text file will be generated. Default is the full path name of the current library and message file.</p> <p>For user-defined messages, the file name must be <i>NnnAPMSL.MSG</i>. For Natural system messages, the file name must be <i>NnnLmmmm.MSG</i>.</p>
	To (message file)	<p>The name of the message file into which the text file will be generated. Default is the full path name of the current library and message file.</p> <p>For user-defined messages, the file name must be <i>NnnAPMSL.MSG</i>. For Natural system messages, the file name must be <i>NnnLmmmm.MSG</i>.</p>	
<p>For further information on file formats and generating message and text files, see the relevant section.</p>			
Export Message File	<p>This function does not apply to a remote environment located on a mainframe platform.</p> <p>Exports a message file and converts it into a text file. Note that you always need to specify the full path of a file.</p>		
	<table border="1"> <tr> <td data-bbox="444 1199 964 1509">From (message file)</td> <td data-bbox="964 1199 1474 1509"> <p>The name of the message file from which the text file will be generated. Default is the full path name of the current library and message file.</p> <p>For user-defined messages, the file name must be <i>NnnAPMSL.MSG</i>. For Natural system messages, the file name must be <i>NnnLmmmm.MSG</i>.</p> </td> </tr> </table>	From (message file)	<p>The name of the message file from which the text file will be generated. Default is the full path name of the current library and message file.</p> <p>For user-defined messages, the file name must be <i>NnnAPMSL.MSG</i>. For Natural system messages, the file name must be <i>NnnLmmmm.MSG</i>.</p>
	From (message file)	<p>The name of the message file from which the text file will be generated. Default is the full path name of the current library and message file.</p> <p>For user-defined messages, the file name must be <i>NnnAPMSL.MSG</i>. For Natural system messages, the file name must be <i>NnnLmmmm.MSG</i>.</p>	
	<table border="1"> <tr> <td data-bbox="444 1509 964 1587">To (text file)</td> <td data-bbox="964 1509 1474 1587">The name of the text file that will be generated.</td> </tr> </table>	To (text file)	The name of the text file that will be generated.
To (text file)	The name of the text file that will be generated.		
<p>For further information on file formats and generating message and text files, see the relevant section.</p>			

Layout

The **Layout** option allows specification of valid message ranges to categorize messages. Overlapping of ranges is possible. A new message can only be added if its number is within the range specified in the layout.

The layout definition applies to *all* languages. It is stored in the English message file.

An example layout definition is shown below:














In the fields **From** and **To**, specify the message range. In the **Description** field, enter text that describes the message category.

To insert or delete rows, mark a row and press the INS or DEL key. If the maximum of 18 rows is displayed, you may have to delete or overwrite another row before you can insert a new one.

Toolbar Buttons

The toolbar buttons represent the following menu items or command buttons:

Toolbar Button	Menu Item/Command Button	Menu
	New Lib/Lang	File
	Open Lib/Lang	File
	Print	File
	Exit	File
	Cut	Edit
	Copy	Edit

Toolbar Button	Menu Item/Command Button	Menu
	Paste	Edit
	Undo	Edit
	New Message > Upwards	Edit
	New Message > Downwards	Edit
	Read All	Edit
	Translate	Edit
	Filter Display	View
	Sample	Options
	Layout	Options
	Size > Startup	Options
	Import Text File	Options
	Export Message File	Options
	Previous	-
	Next	-
	Update	-
	Reset	-

Status Bar

The status bar on the bottom of the **SYSERR - Error Messages** window displays the following:

- The total number of messages that exist in the specified library for the current language, the possible message range 1 to 9999, and the last message number used.

This information is only displayed when initializing SYSERR, opening another library or when using the **Read All** menu option described under *Edit Menu*.

- Natural system messages or the ID of the current library.
- The code and name of the current language.

Online Help

You can obtain online information on using the SYSERR utility.

▶ **To view the overview of the online documentation**

- Choose the **Help** menu.

Or:

Press F1 when no dialog box is open.

The overview page of the *SYSERR Utility* documentation appears.

▶ **To view context-sensitive help**

- Choose the **Help** button in a dialog box.

Or:

Press F1 in a dialog box.

Help information specific to the task you are performing appears.

52

Converting Natural System Short Messages

If your terminal does not display certain characters correctly or if your terminal cannot display lower-case characters, it is possible to convert the characters of Natural system short messages with the **SENSYS-D** dialog box.

▶ **To convert characters of messages**

- 1 In the **Logical View** of the Natural Studio tree view, select the **System Libraries** node and then **SYSERR**.

Or:

Issue the following system command:

```
LOGON SYSERR
```

where *library-ID* is the name of the required library.

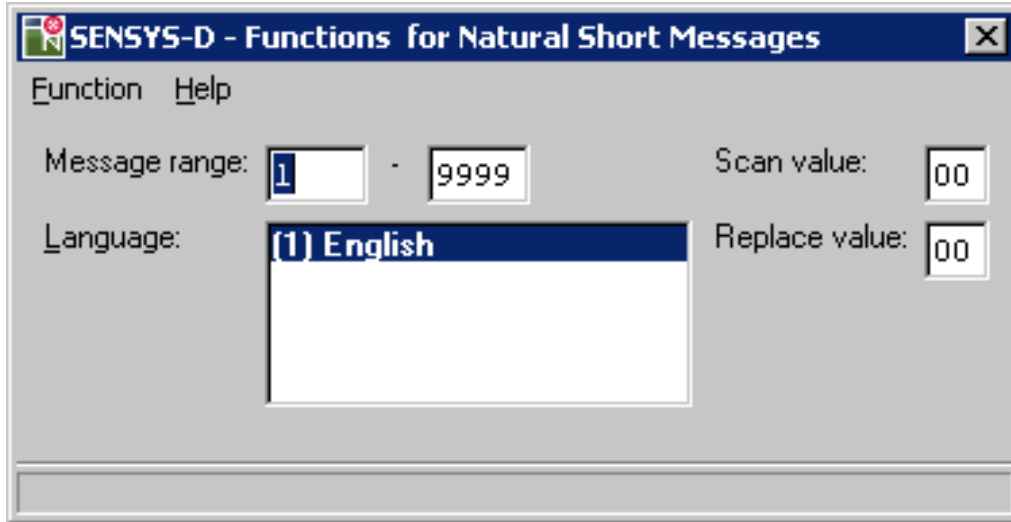
- 2 From the **SYSERR** library, select the **Dialogs** subnode and then **SENSYS-D**.

Or:

In the command line, enter the following:

```
SENSYS-D
```

A **SENSYS-D** dialog box similar to the example below appears for the current language:



3 Specify a range of messages.

If required, enter hexadecimal values in the **Scan value** and **Replace value** boxes.

4 From the **Function** menu, choose one of the following:

Function	Explanation
Uppercase Messages	Converts messages to upper case. Once converted to upper case, you cannot convert them back to lower case. To recover lower-case messages, unload the messages by using the Object Handler, save the transfer file and, when required, reload the messages by using the Object Handler. See also the <i>Object Handler</i> documentation.
Scan Character Hex	Scans for hexadecimal characters entered in the Scan value box.
Scan and Replace	Scans hexadecimal characters entered in the Scan value box and replaces them with the hexadecimal characters entered in the Replace value box. This function may be useful, for example, to replace special signs.
Replace using SECSET-N	Replaces the characters of a message by the character set defined in the SECSET-N subprogram. SECSET-N is stored in the SYSERR subnode Subprograms .
Display Message Hexadecimal	Enables or disables the display of a message in hexadecimal format.
Display Char Table for Terminal	Enables or disables this function to determine the characters your terminal can represent.
Processing Log	Enables or disables the processing log to view the results of the functions executed.

53

Generating Message and Text Files

- Storing a Message File 344
- Creating a Text File 344
- Generating a Message File 345
- Recreating a Text File 346

You can create messages as text files in any environment outside Natural and convert them into message files to be maintained with the SYSERR utility. Message files are created and maintained with the import and export functions of the SYSERR utility.

Message files are created in a platform-independent format, which is portable across any Natural-supported UNIX, OpenVMS and Windows platforms. For example, a message file created in a Natural for Windows environment, can be copied onto a UNIX or an OpenVMS platform without manual conversion; the necessary endian conversion is performed by Natural. For further information, see *Portable Natural System Files* in the *Operations* documentation and *Transferring Natural Generated Programs* in the *Programming Guide*.

Storing a Message File

The message files must be stored with the file extension .MSG in the Natural Err directories.

The message files are stored in the following Natural directories:

```
Natural\NATAPPS\FUSER\library-ID\Err
Natural\Natural version\FNAT\library-ID\Err
Natural\Natural version\Err
```

User-defined message files are stored in the Err subdirectory of the library in the FNAT or FUSER system file from which the application is executed, the steplib, or the SYSTEM library.

For Natural system messages, the message files must be stored in the Err subdirectory in the Natural root directory. Natural system messages are stored in eight message files.

Creating a Text File

For Natural system or user-defined messages, the import function of the SYSERR utility generates a message file from one text file.

To create such a text file, you must use a specific layout, as shown in the following example:

Example:

```

NAT
0010
0100
0010E NO MESSAGE TEXT DEFINED!
0020E MISSING/INVALID SYNTAX; UNDEFINED VARIABLE-NAME.
0025E ERROR IN ENTRY FOR NUMBER OF RECORDS TO BE PROCESSED.
0050E INCORRECT FIELD SPECIFICATION IN 'WHERE' CLAUSE.
#PLEASE CHECK PROGRAM
#FOR ERRORS
0100E FUNCTION NOT AVAILABLE.

```

Explanation:

NAT or <i>library-ID</i>	The prefix of the message number to be displayed with the message. The default prefix is NAT for Natural system messages and the library ID for user-defined messages.
0010	The four-digit starting number of a range of messages.
0100	The four-digit ending number of a range of messages. All message numbers that are defined in this text file must be within this range.
0010E	<p>NO MESSAGE TEXT DEFINED!</p> <p>This is the short message for message number 0010. The E is mandatory and means error. This message will be issued with the following Natural statement:</p> <pre>REINPUT *0010</pre> <p>Explanatory long messages must be placed immediately below this short message; each of these additional lines must start with a hash/number (#) sign. Up to 20 additional lines of long message text are allowed for each short message.</p>

Generating a Message File

The SYSERR utility provides the option to generate a message file from a text file.

For user-defined messages, one output message file can be created in one language for each library. Each message file must be stored in the Err subdirectory of that library.

Naming Conventions

For user-defined messages, the name of the message file must be:

```
NnnAPMSL.MSG
```

where *nn* is the language code (01 - 60), for example 01 for English.

For Natural system messages, the name of the message file must be:

```
NnnLmmmm.MSG
```

where *nn* is the language code to be used and *mmmm* the starting number of the message range. The ranges of message numbers are fixed, as defined during Natural system installation, for example:

```
N01L0000    Messages 1 - 1999
N01L2000    Messages 2000 - 2999
```

▶ To generate a message file

- See the **Import Text File** function of the **Options** menu described in the section *SYSERR Utility Window and Functions*.

Recreating a Text File

The SYSERR utility provides the option to recreate a text file for message text maintenance. This is done by reconvertng a messages file into a text file.

▶ To recreate a message text file

- See the **Export Message File** function of the **Options** menu described in the section *SYSERR Utility Window and Functions*.

54

Managing Messages in Different Libraries

You can transfer messages between different libraries and move, rename, find, list or delete messages in different libraries. For a transfer, you can copy the messages to and from a file.

▶ **To copy/move, rename, find, list or delete messages**

- Use the appropriate Natural Studio functions described in the *Using Natural Studio* documentation.

Or:

Use the *Object Handler* as described in the relevant documentation.

▶ **To transfer messages using files**

- Use the **Export Message File** and **Import Text File** functions described in the section *Options Menu*.

Or:

Use the *Object Handler* as described in the relevant documentation.

55

Application Programming Interface USR0020P

The application programming interface USR0020P in the Natural system library SYSEXT is provided to read messages from the FNAT or FUSER system file. Thus, it is possible, for example, to have long messages displayed in an application (as part of your own user-defined help system) without having to use the Natural system library SYSERR.

Log on to the Natural system library SYSEXT and, in the command line, enter the command `MENU`. In the list provided, mark the program USR0020P with a question mark (?). A window is then displayed, in which you can select the function to be executed for the program. If you enter an `I`, further information on the use of USR0020P is displayed.

XI SYSEXT Utility - Natural Application Programming

Interfaces

56 SYSEXT Utility - Natural Application Programming

Interfaces

▪ Prerequisite	354
▪ Introduction to SYSEXT	354
▪ Invoking and Terminating SYSEXT	356
▪ SYSEXT Tree View Items	357
▪ Performing SYSEXT Utility Functions	358
▪ Interface Versions	361
▪ Reserved Keywords	361
▪ Using a Natural API	361
▪ List of Natural APIs	362

The utility SYSEXT is used to locate and test Natural Application Programming Interfaces (APIs) contained in the current system library SYSEXT either in a local Windows environment or in a remote environment located on a Windows, a UNIX, an OpenVMS or a mainframe platform.

A Natural API is a Natural subprogram (cataloged object) that is used for accessing and possibly modifying data or performing services that are not accessible by Natural statements. Natural APIs refer to Natural, a subcomponent or a subproduct.

Related Topics:

- *Application Programming Interfaces - Natural Security* documentation
- *SYSAPI - APIs of Natural Add-on Products - Utilities* documentation
- *Natural Functions - System Functions* documentation

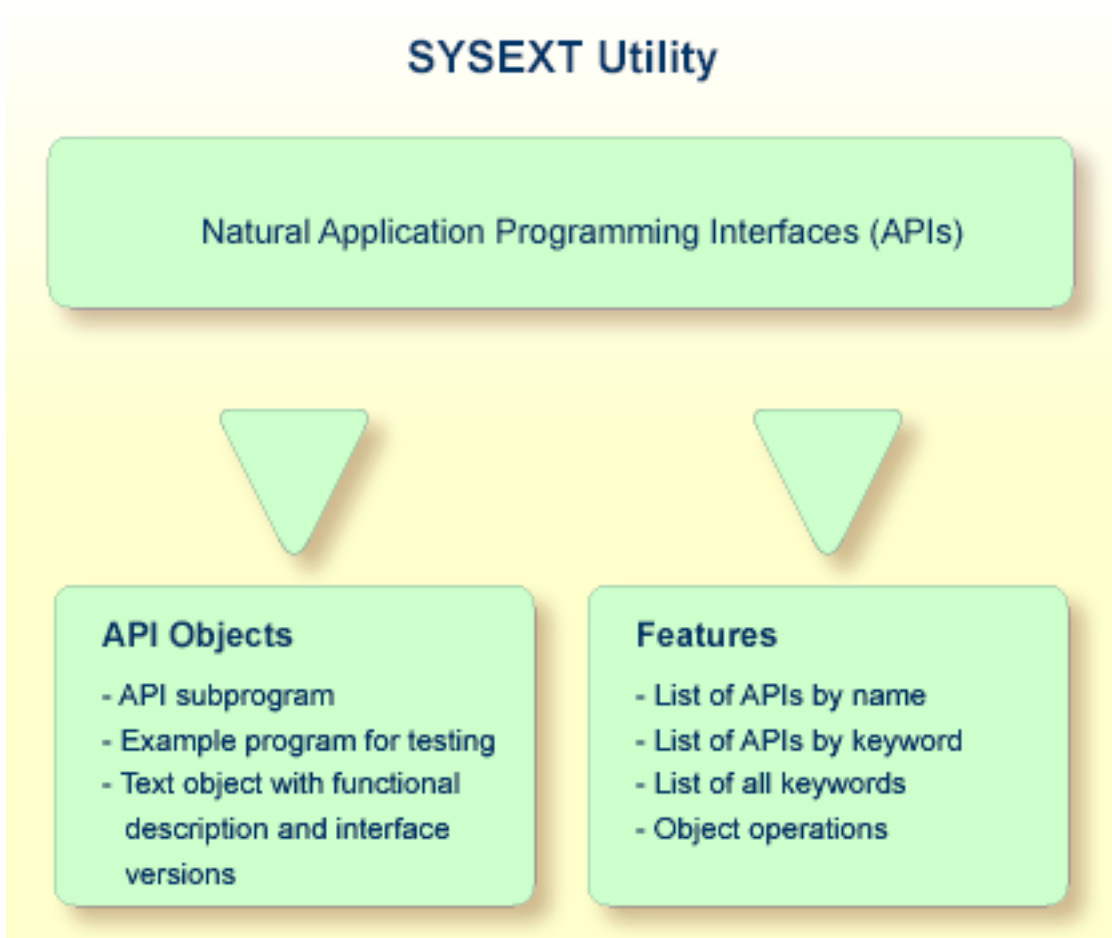
Prerequisite

- The **Enable Plug-ins** option must be selected. This option is selected by default. For details, see *Workspace Options* in the section *Setting the Options* in the *Using Natural Studio* documentation.

Introduction to SYSEXT

For each Natural API, the utility SYSEXT provides a functional description, one example program and API-specific keywords.

The following diagram is an overview of the Natural objects and major features SYSEXT provides:



Objects Provided for Natural APIs

The types of Natural object typically provided for each Natural API are listed in the following section. Additional objects that might exist for a particular API are not covered.

All API-related objects are contained in the library SYSEXT on the system file FNAT.

In the following table, *nnnn* denotes the 4-digit number to identify the API as well as the corresponding example program and text object.

Object Name	Explanation
USR <i>nnnn</i> N	The API subprogram (cataloged object) that performs the designated function.
USR <i>nnnn</i> P	An example program (source object) that can be used to test the effect of the API. The example program invokes the corresponding subprogram USR <i>nnnn</i> N.
USR <i>nnnn</i> T	A text object that contains a description of the corresponding API. The description comprises purpose, function and calling conventions of the API and relevant keywords, category and interface versions .

- ❗ **Caution:** Do *not* modify the source objects EXT-XML1 and EXT-XML2. They are required for configuring the SYSEXT utility and intended for Software AG internal use only.

Invoking and Terminating SYSEXT

This section provides instructions for invoking and terminating the SYSEXT utility.

▶ To invoke SYSEXT

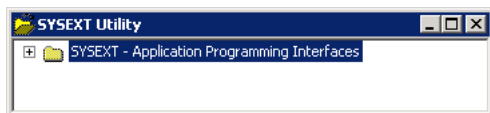
- Enter the following system command:

```
SYSEXT
```

Or:

From the **Tools** menu, select **Development Tools > Application Programming Interfaces**.

When invoking SYSEXT, the plug-in for the utility SYSEXT is activated and the SYSEXT utility window appears with the root node **SYSEXT - Application Programming Interfaces** as shown in the example below:



▶ To restart SYSEXT

- If you want to restart SYSEXT during the current Natural session, as an alternative to the start methods mentioned above, from the toolbar, choose the following icon:



This icon appears after initially invoking SYSEXT, when the plug-in for the utility SYSEXT is activated.

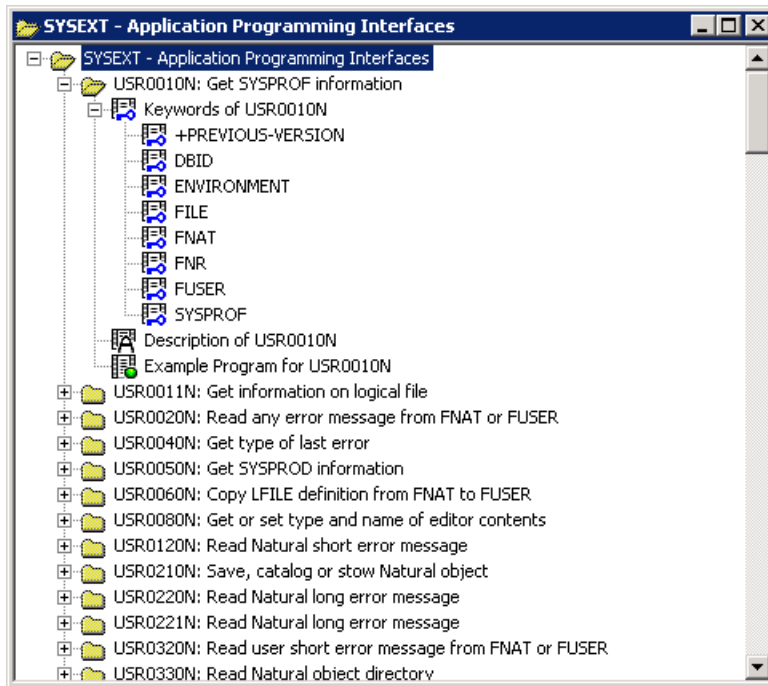
▶ To terminate SYSEXT

- Choose the standard Windows close function.

SYSEXT Tree View Items

This section describes the nodes and items contained in the tree view of the SYSEXT utility window.

If you expand all tree nodes, the tree looks similar to the example below:



Each Natural API is represented by an API node that contains the example program, description and keywords that relate to this API. The API node name consists of the name of the API subprogram (USR $nnnn$ N) and a brief description of its functional purpose. The nodes are sorted by API names.

The following items are provided in an API node:

Item	Explanation
Keywords	All keywords relevant to the API. See also the functions Select Keyword and List All Keywords in <i>Performing SYSEXT Utility Functions</i> .
Description	A text object (USR $nnnn$ T) that contains a description of the API. The description comprises purpose, function and calling conventions of the API and keywords relevant to the API.
Example Program	An example program (source object USR $nnnn$ P) of how to invoke the API.

Performing SYSEXT Utility Functions

The SYSEXT utility functions can be used to perform operations on API-specific text objects (descriptions) and example programs or find APIs relevant to a current task by specifying a keyword.

Object operations include functions such as List, Open and Execute, which correspond to the standard functions available when maintaining or executing a Natural object of the type text or program. These functions can be performed by using the context menu associated with each object. For details of these functions, refer to the relevant sections in the *Using Natural Studio* documentation.

The section below covers the following topics:

- [Select Keyword](#)
- [List All Keywords](#)
- [Refresh](#)

Select Keyword

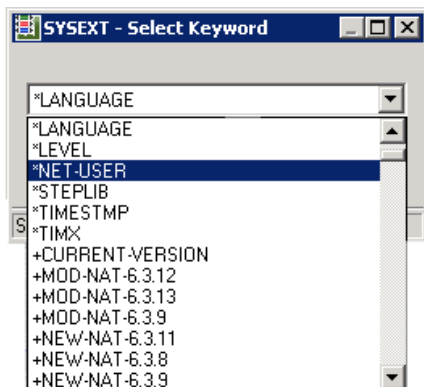
This function is used to list APIs by keyword.

▶ To list APIs by keyword

- 1 Select the root node **SYSEXT - Application Programming Interfaces**, open the context menu and choose **Select Keyword** or press SHIFT+K.

The **Select Keyword** window appears.

- 2 From the drop-down list box, select a keyword as shown in the example below:

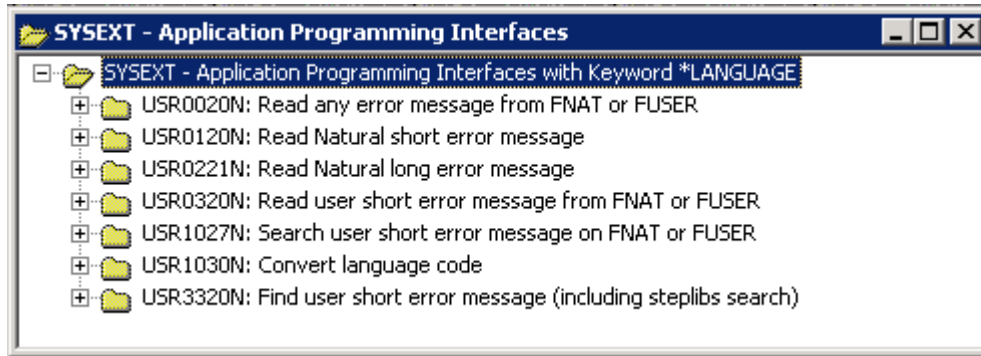


- 3 Choose the **OK** button.

The root node **SYSEXT - Application Programming Interfaces with Keyword** appears for the selected keyword.

- 4 Expand the root node.

The nodes of all APIs to which the specified keyword applies are displayed as shown in the example of keyword *LANGUAGE below:



- 5 If desired, to return to the display of all API nodes (default setting), in the **Select Keyword** window, select the asterisk (*).

List All Keywords

This function is used to list all keywords for the APIs available in the current system environment.

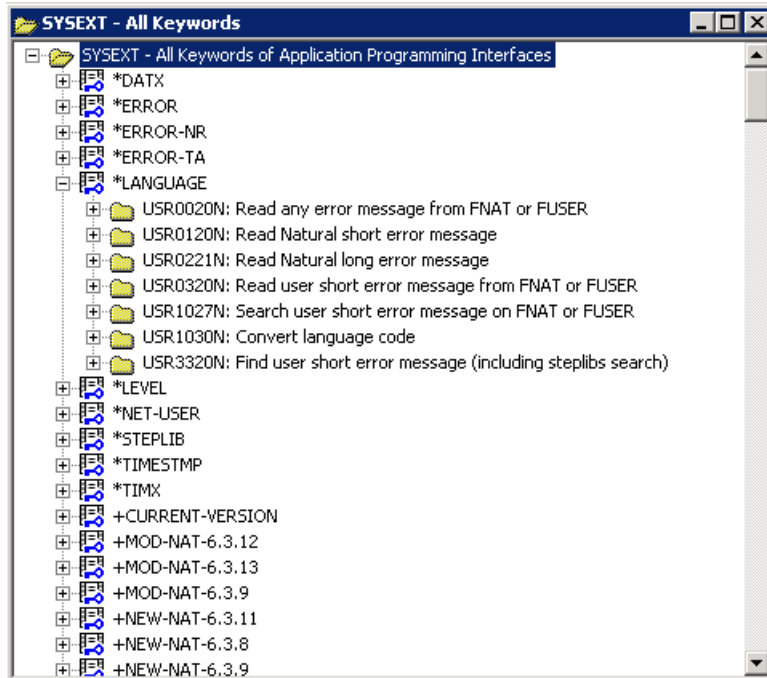
▶ To list all API keywords

- 1 Select the root node **SYSEXT - Application Programming Interfaces**, open the context menu and choose **List All Keywords** or press SHIFT+A.

The **All Keyword** window appears as an additional window with the root node **SYSEXT - All Keywords of Application Programming Interfaces**.

- 2 Expand the root node.

A list of all keywords is displayed as shown in the example below:



Refresh

This function updates API information in the tree view using the data from the objects contained in the current library SYSEXT. The refresh is only required if an API description or a keyword was modified or if a text object was removed.



Note: Do *not* modify the source objects EXT-XML1 and EXT-XML2. They are required for configuring the SYSEXT utility and intended for Software AG internal use only.

▶ To refresh API information

- 1 Select the root node **SYSEXT - Application Programming Interfaces**, open the context menu and choose **Refresh** or press SHIFT+R.

A **Refresh** window appears.

If you access a remote environment on a mainframe, you must specify the parameter SORT in the map environment settings as indicated below:

```
SORT=(WRKSIZE=50)
```

For details on mapping a server, see *Accessing a Remote Development Environment* in the documentation *Remote Development Using SPoD*. For details on SORT, refer to the *Parameter Reference* of Natural for Mainframes.

- 2 Choose the **OK** button to confirm the refresh or choose **Cancel** to abort the operation.

Interface Versions

Interface versions can be seen as a collection of APIs with (almost) the same functionality but with differently extended parameter specifications. Thus, they cover a development cycle to be kept explicit for sake of compatibility (of later versions with earlier versions).

If an API has interface versions, they are displayed in the corresponding text object `USR $nnnn$ T`. Interface versions are ordered within a list according to the version they belong to. The rightmost element belongs to the current version. This status is expressed by the reserved keyword `+CURRENT-VERSION`. All other elements belong to a previous version and are marked with the reserved keyword `+PREVIOUS-VERSION`. APIs without interface versions are called unique.

APIs with interface versions are displayed intensified on the menu.

Reserved Keywords

Reserved keywords refer to meta information on APIs, for example the Natural version in which an API has been added. Reserved keywords always start with a plus sign (+). See the table below for a description:

Reserved Keyword	Description
<code>+CURRENT-VERSION</code>	The current version of an API with interface versions (see Interface Versions).
<code>+PREVIOUS-VERSION</code>	A previous version of an API with interface versions (see Interface Versions).
<code>+NEW-PROD-version</code>	An API that has been added to a specific product in a specific version. For example, <code>+NEW-NAT-6.3.11</code> refers to an API that has been added to the product Natural in version 6.3.11.
<code>+MOD-PROD-version</code>	An API that belongs to a specific product and has been modified in a specific version. For example, <code>+MOD-NAT-6.3.12</code> refers to an API that belongs to product Natural and has been modified in version 6.3.12.

Using a Natural API

If you want to use a Natural API contained in the system library SYSEXT, perform one of the following steps:

- Define the system library SYSEXT in the system file FNAT as a `steplib` library for the user library that contains the Natural objects that use this API. Thus, no API-specific actions are required when upgrading your Natural version.

- Copy the required API to the system library SYSTEM in the system file FNAT. Thus, you only need to check a single library for APIs when upgrading your Natural version.
- Copy the required API to the system library SYSTEM in the system file FUSER (not recommended).
- Copy the required API to the user library (or one of its steplib) in the system file FUSER which contains the Natural objects that use this API (not recommended).

An API can only be used in the Natural version with which it is delivered. It is strongly recommended to store the APIs only in the FNAT system file. This will ensure that the right version is always executed.

▶ To make use of an interface

- 1 In the calling program, use the `DEFINE DATA` statement to specify the parameters listed in the text object `USRnnnnT` of that API. In the example program `USRnnnnP`, the parameters are defined within the `DEFINE DATA LOCAL` statement. Alternatively, you can specify the parameters outside the calling program in a separate LDA (Local Data Area) or PDA (Parameter Data Area), with a `DEFINE DATA LOCAL USING` statement referencing that data area.
- 2 Enter the following statement:

```
CALLNAT 'USRnnnnN' parameters
```

For further information, see the `CALLNAT` statement in the *Statements* documentation.



Note: Non-standard usage is always documented in the respective text object `USRnnnnT`.

List of Natural APIs

The following table gives an overview of available APIs and their function.



Notes:

1. Because there can be situations where a newly added API is not yet covered by the current documentation, we recommend to rely on the list of APIs in the SYSEXT utility in cases where being up-to-date is vital.
2. For detailed information on the use of APIs that apply to Natural Remote Procedure Call (RPC), refer to section *Application Programming Interfaces for Use with Natural RPC* in the *Natural Remote Procedure Call* documentation.
3. APIs with product code PRD are listed and documented separately in the section *Application Programming Interface* in the administration part of the *Predict* documentation.

Interface	Description	Product
USR0010N	Get SYSPROF information	NAT
USR0011N	Get information on logical file	NAT
USR0020N	Read any error message from FNAT or FUSER	NAT
USR0040N	Get type of last error	NAT
USR0050N	Get SYSPROD information	NAT
USR0060N	Copy LFILE definition from FNAT to FUSER	NAT
USR0080N	Get or set type and name of editor contents	NAT
USR0120N	Read Natural short error message	NAT
USR0210N	Save, catalog or stow Natural object	NAT
USR0220N	Read Natural long error message	NAT
USR0221N	Read Natural long error message	NAT
USR0320N	Read user short error message from FNAT or FUSER	NAT
USR0330N	Read Natural object directory	NAT
USR0360N	Modify user short error message on FNAT or FUSER	NAT
USR0420N	Read user long error message from FUSER	NAT
USR0421N	Maintain user long error message on FUSER	NAT
USR0500N	Define title of window	NAT
USR0600N	Get program level information	NAT
USR0610N	Get error information on last database call	NAT
USR0620N	Convert string into ASCII or EBCDIC code	NAT
USR0622N	Reset error counter in ON ERROR statement block	NAT
USR1002N	Save or restore Natural environment parameters	NAT
USR1005N	Get information on certain Natural session parameters	NAT
USR1009N	Convert system variable *TIMESTAMP into numeric variable	NAT
USR1011N	Check name for wildcard or asterisk notation	NAT
USR1012N	Get dynamic error message parts from the last error	NAT
USR1013N	Display current character set	NAT
USR1016N	Get error level for error in nested copycodes	NAT
USR1020N	Add user short error message to FUSER	NAT
USR1021N	Check name for wildcard or asterisk notation	NAT
USR1022N	Get type of database	NAT
USR1023N	Convert date and time format	NAT
USR1025N	Maintain definition of multiple steplibs	NAT
USR1026N	Get information on RETURN data	NAT
USR1027N	Search user short error message on FNAT or FUSER	NAT
USR1028N	Convert bits and bytes	NAT

Interface	Description	Product
USR1029N	Get type of Natural object	NAT
USR1030N	Convert language code	NAT
USR1031N	Check Natural object name	NAT
USR1032N	List cataloged Natural objects with type	NAT
USR1033N	Find DBID and FNR of a cataloged DDM	NAT
USR1034N	Get NTTF file table with translated DBID and FNR	NAT
USR1035N	Maintain objects using the Software AG editor engine	NAT
USR1036N	Maintain user profile of the Software AG editor	NAT
USR1038N	Get platform-specific information	NAT
USR1039N	Modify the clipboard	NAT
USR1040N	Get or set profile parameter UDB	NAT
USR1041N	Install error transaction program (*ERROR-TA)	NAT
USR1042N	Get or set value of UPDATE command	NAT
USR1043N	Perform Adabas direct calls	NAT
USR1048N	Modify PF-key labels	NAT
USR1050N	Get or set work file name	NAT
USR1052N	Send a command to the operating system	NAT
USR1053N	Get the value of an environment variable	NAT
USR1054N	List libraries	NAT
USR1055N	List objects in a library	NAT
USR1056N	List DDMs on the FDIC file or in a library	NAT
USR1057N	Read a Natural source code into an array	NAT
USR1058N	Read a DDM source code into an array	NAT
USR1059N	Initialize the Natural Reporter interface	NAT
USR1060N	Terminate the Natural Reporter interface	NAT
USR1061N	Open an existing report layout	NAT
USR1062N	Close a report	NAT
USR1063N	Select printer to print a report	NAT
USR1064N	Print a report	NAT
USR1065N	Preview a report	NAT
USR1066N	Enable or disable the Natural message 'Executing ...'	NAT
USR1067N	Check Natural library name	NAT
USR1068N	Maintain the number of DBMS calls, MAXCL and MADIO parameters	NAT
USR1069N	Get or set a printer configuration	NAT
USR1071N	Set credentials for RPC server	RPC
USR1072N	Get command ID of a retain set	NAT

Interface	Description	Product
USR2001N	Get information on last error	NAT
USR2004N	Get information on logical file	NAT
USR2005N	Read internal file translation table	NAT
USR2006N	Get information from error message collector	NAT
USR2007N	Get or set data for RPC default server	RPC
USR2009N	Get dynamic error message parts from the last error	NAT
USR2010N	Get error information on last database call	NAT
USR2011N	Get or set work file name	NAT
USR2012N	Get system variable *NET-USER	NAT
USR2013N	Get SYSPROF information	NAT
USR2014N	Maintain objects using the Software AG editor engine	NAT
USR2018N	Read Natural object directory	NAT
USR2019N	Read or save Natural source code from/to the source area	NAT
USR2023N	Get type of database	NAT
USR2026N	Get TECH information	NAT
USR2027N	Define wait interval for current session	NAT
USR2030N	Get dynamic error message parts from the last error	NAT
USR2031N	Get SYSPROD information	NAT
USR2032N	Support commit for CLOSE CONVERSATION	RPC
USR2035N	Get or set parameters for SSL support	RPC
USR2036N	Convert system variable *TIMESTAMP into numeric variable	NAT
USR2071N	Support EntireX Security on client side	RPC
USR2072N	Support EntireX Security on server side	RPC
USR2073N	Ping or terminate an RPC server	RPC
USR2074N	Set new password for Natural Security user in RPC context	RPC
USR2075N	Terminate EntireX Broker service	RPC
USR2076N	Get or set RPC TIMEOUT value	RPC
USR3013N	Get SYSPROF information	NAT
USR3025N	Maintain definition of multiple steplibs	NAT
USR3320N	Find user short error message (including steplibs search)	NAT
USR4003N	Get Natural stack information	NAT
USR4004N	Get dynamic Natural profile parameters at session start	NAT
USR4005N	Read all current PF-key settings	NAT
USR4007N	Get or set profile parameter SYNERR	NAT
USR4008N	Set library for RPC execution	RPC
USR4009N	Set parameters for EntireX	RPC

Interface	Description	Product
USR4010N	Get runtime settings of RPC server	RPC
USR4011N	Create A20 hash value for variable input	NAT
USR4012N	Set application error on RPC server	RPC
USR4025N	Maintain definition of multiple steplibs in Natural Security	NAT
USR4201N	Maintain data area sources	NAT
USR4206N	List objects in a library	NAT
USR4208N	Read or write a Natural resource	NAT
USR4209N	Get short name of subroutine	NAT
USR4210N	Perform base64 conversion of alphanumeric and binary bytes	NAT
USR4212N	Analyze data area	NAT
USR4215N	Get list of resources in a Natural library	NAT
USR4217N	Send data to Optimize for Infrastructure	NAT
USR4218N	Get information on Natural Development Server	NDV
USR4220N	Perform base64 conversion of alphanumeric and binary bytes	NAT
USR5001N	Turn results window on/off	NAT
USR5002N	Maintain bitmaps and icons for a tab	NAT
USR5003N	Maintain context menus for a tab	NAT
USR5004N	Maintain general layout of a tab	NAT
USR5005N	Get or set active tab of results window	NAT
USR5006N	Define UpdateCommandHandler and CommandHandler for tab	NAT
USR5007N	Maintain checked and enabled state for a context menu item	NAT
USR5008N	Maintain columns of a tab	NAT
USR5009N	Get number of columns or rows of a tab	NAT
USR5010N	Maintain default column width and specific column width	NAT
USR5011N	Maintain rows of a tab	NAT
USR5012N	Get or set data of a tab	NAT
USR5013N	Maintain checked state of row(s)	NAT
USR5014N	Maintain selected row(s)	NAT
USR5015N	Get or set row of focus; scroll row into visible area	NAT
USR5016N	Get or set data for UpdateCommandHandler and CommandHandler	NAT
USR5017N	Copy selected row(s) to clipboard	NAT
USR6001N	Call external XSLT processor	NAT
USR6002N	Get the current values of internal counters	NAT
USR6003N	Maintain type/name/library of program editor active window	NAT
USR6004N	Access the program editor active window	NAT
USR6005N	Get information on program editor active window	NAT

Interface	Description	Product
USR6006N	Get path to system file	NAT
USR6007N	Get information on last error in Tamino database	NAT
USR6008N	Get data of object in source area	NAT
USR6009N	Maintain Natural buffer pool	NAT
USR6201N	Write source to Natural Development Server	NDV
USR6202N	Get or set value of an environment variable	NAT
USR6203N	Maintain a resource in a Natural library	NAT
USR6204N	Set profile parameter PROGRAM	NAT
USR6303N	Get Natural stack information	NAT
USR6304N	Get or set reliable state for RPC execution	RPC
USR6305N	Commit or rollback reliable RPC messages	RPC
USR6306N	Get status of UOWs of current EntireX Broker user	RPC
USR6307N	Get or set Request Document timeout value (RQTOUT)	NAT

XII

SYSEXV Utility

57 SYSEXV Utility

- Executing Example Programs of Current Versions 372
- Executing Example Programs of Non-current Versions 372
- Terminating the SYSEXV Utility 373

The utility SYSEXV gives you access to examples of new features available in the current and in some earlier versions of Natural. All programs are available as source objects and display a detailed functionality description when they are executed. Example programs for non-current Natural versions are also available.

This chapter covers the following topics:

Executing Example Programs of Current Versions

▶ To execute an example program of a current version

- 1 Enter the following system command:

```
SYSEXV
```

Or:

Log on to the SYSEXV library:

```
LOGON SYSEXV
```

and continue by entering

```
VERSION
```

As a result, a menu is displayed from which you can choose a Natural version.

- 2 Choose a version from the menu.

As a result, a commented menu is displayed from which you can choose an example program relevant to the selected version.

- 3 Choose a program.

As a result, the program is executed.

Executing Example Programs of Non-current Versions

▶ To execute an example program of a non-current version

- 1 Log on to the SYSEXV library:

```
LOGON SYSEXV
```

- 2 List the text object A-README.



Note: Do not modify the text object A-README.

A-README lists and comments example programs for non-current versions.

- 3 Choose a program and execute it.

Terminating the SYSEXV Utility

▶ To terminate the SYSEXV utility

- In the SYSEXV utility menu, choose **OK**.

XIII

SYSMAIN Utility - Object Maintenance

The SYSMAIN utility is used to perform object maintenance functions such as copy, move, replace, delete and import.

General Information on SYSMAIN

Invoking and Terminating SYSMAIN

Listing Objects

Finding Objects

Copying Objects

Moving Objects

Deleting Objects

Renaming Objects

Importing Objects

Using SYSMAIN with Subprogram

XRef Considerations

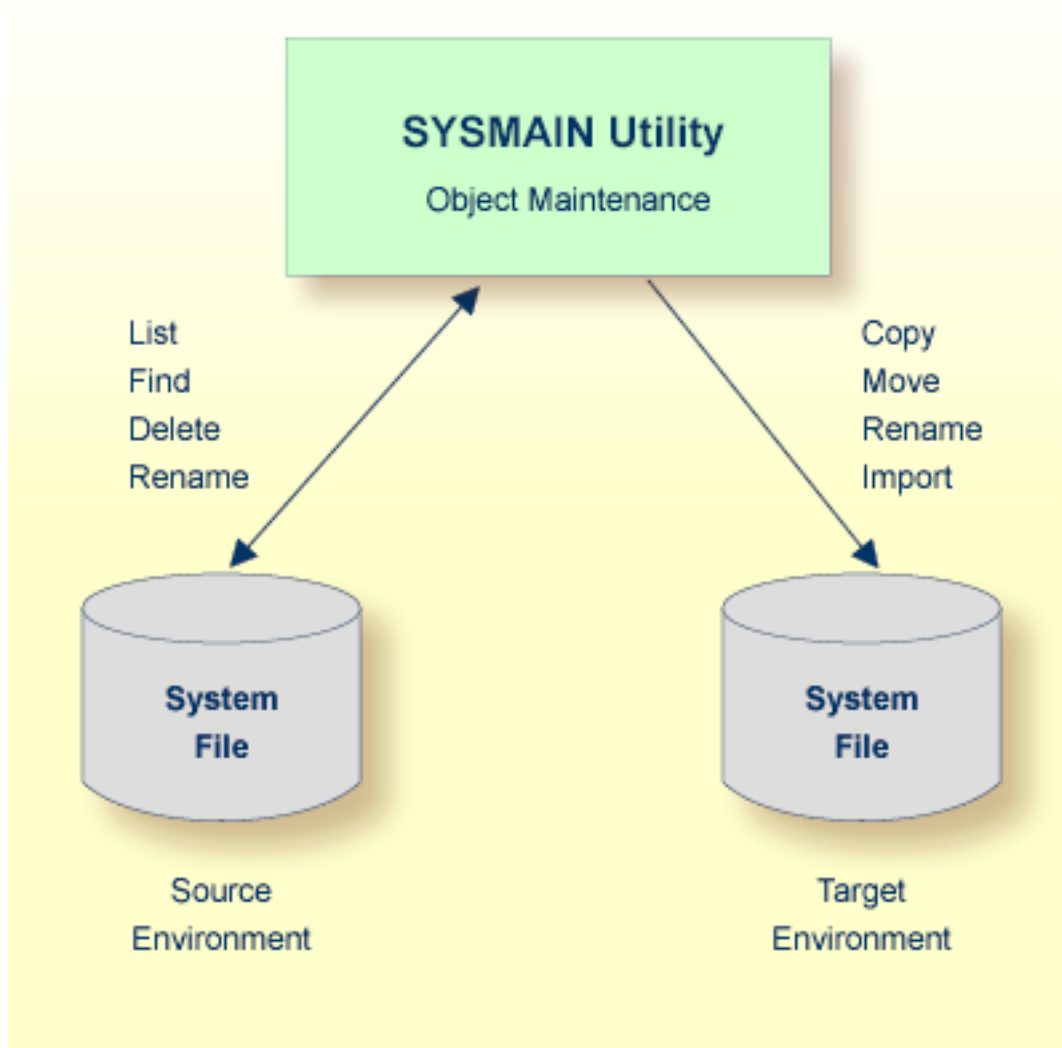
Security Considerations for Administrators

58

General Information on SYSMAIN

The SYSMAIN utility is used to perform object maintenance functions, such as copy, move and delete, within a local Windows environment or within a remote environment located on a Windows, a mainframe, a UNIX or an OpenVMS platform.

The following diagram is a basic illustration of the SYSMAIN functionality:



Objects that can be maintained with the SYSMAIN utility comprise programs, subprograms, maps and data definition modules (DDMs).

In most cases, SYSMAIN utility functions can be accomplished by using drag-and-drop or copy/cut-and-paste functionality or menu functions provided within the library workspace of the Natural Studio (see also *Managing Natural Objects* and *Using Natural Libraries* in the *Using Natural Studio* documentation).

However, there are SYSMAIN utility functions that cannot be covered by library workspace features such as using different system files for object processing, specifying the transfer of cross-reference (XRef) data and performing object maintenance functions online or in batch mode by using a subprogram.

59 Invoking and Terminating SYSMAIN

- Invoking SYSMAIN 380
- Terminating SYSMAIN 382

This section provides instructions for invoking and terminating the SYSMAIN utility online.

For instructions on invoking and executing SYSMAIN online or in batch mode with a subprogram, see *Using SYSMAIN with Subprogram*.

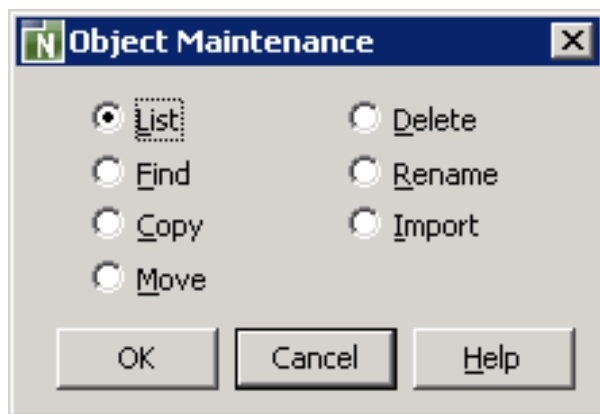
Invoking SYSMAIN

► **To invoke the SYSMAIN utility**

- 1 Enter the following system command:

```
SYSMAIN
```

An **Object Maintenance** dialog box similar to the example below appears with the SYSMAIN utility menu:



- 2 Select the radio button that corresponds to the required function by choosing any of the following methods:

Click a radio button.

Or:

Use DOWN ARROW or UP ARROW to navigate to a radio button.

Or:

Press one of the following shortcut keys:

SHIFT+L for **List**

SHIFT+F for **Find**

SHIFT+C for **Copy**

SHIFT+M for **Move**
 SHIFT+D for **Delete**
 SHIFT+R for **Rename**
 SHIFT+I for **Import**

- 3 Choose **OK**.

(**Cancel** exits the SYSMAIN utility.)

An **Object Maintenance** dialog box appears for the selected function as shown in the following example of **Copy**:

Object Maintenance - Copy

Source

Library: TESTLIB DBID: 30 FNR: 30

FDIC/FSEC...

Name: *

Type

Programming Views (DDM)

Code

Source Cataloged

XREF

User ID: Date: 00.00.0000 Time: 00:00

Target

Library: TESTLIB DBID: 30 FNR: 30

FDIC/FSEC...

Name: *

Confirm on replace Rename

OK Cancel Help

Terminating SYSMAIN

▶ To terminate the SYSMAIN utility

- Choose **Cancel**.

Or:

Choose the standard Windows close button.

60 Listing Objects

The **List** function is used to list a range of objects in a source environment and/or display the source code of single or multiple objects.

This section provides instructions for specifying list options in the **Object Maintenance - List** dialog box.

▶ To list objects

- 1 In the **Library** list box, enter the name of the library that contains the object(s) you want to list or select a library from the drop-down list.

The default is the current library. The **Library** list box does not apply to DDMs (data definition modules) in a remote mainframe environment. Therefore, the box disappears when the **Views (DDM)** check box is selected.

- 2 In the **DBID** (database ID) and **FNR** (file number) boxes, replace the values if the library is not in the current system file. The default is the current FNAT or FUSER, or FDIC for DDMs on mainframes. In a remote environment, you can specify security information for these files as described in *File Security for Remote Environments*.
- 3 In a remote environment, use the **FSEC/FDIC** button if you want to specify security information for the system file FSEC or FDIC for XRef data; see also *File Security for Remote Environments* and *XRef Considerations*.
- 4 In the **Name** box, enter the name of a single object or specify a range of names; see *Specifying a Range of Names*. The default is asterisk (*), indicating all objects.
- 5 In the **Type** group box, select the type(s) of object by choosing one of the following options:

Select **Programming** (default) for all types of object (including program and subprogram) except DDM. You can limit the types of object by choosing **Object Types** and selecting or deselecting the required item(s) from the list provided:

- To select list items:

Click on a single item.

Or:

Press UP ARROW or DOWN ARROW to go to and select the required item.

Or:

Press and hold down CTRL or SHIFT and click on multiple non-consecutive or consecutive items respectively.

Or:

Press SHIFT+UP ARROW or SHIFT+DOWN ARROW to select multiple consecutive items.

Or:

Choose **Select All** to select all items.

- To deselect list items:

Press and hold down CTRL and click on a selected item again.

Or:

Choose **Deselect All** if all items have been selected.

Or:

Select **Views (DDM)** for DDMs only.

- 6 In the **Code** group box, select **Source** and/or **Cataloged** to list all objects for which either the source object or the cataloged object, or both exist. The default is both the source object and the cataloged object.
- 7 In the **User ID** box, enter the ID of a user if you want to list only the objects that were last saved or cataloged by this user. The default is no user ID.
- 8 In the **Date** box, enter a start date to list only objects that were saved or cataloged on or after this date. A date must be entered in the format associated with the preset value in the **Date** box. The default date is 0000.00.00 (no date). The date is given in the format *YYYY.MM.DD* (*YYYY* = year, *MM* = month, *DD* = day).

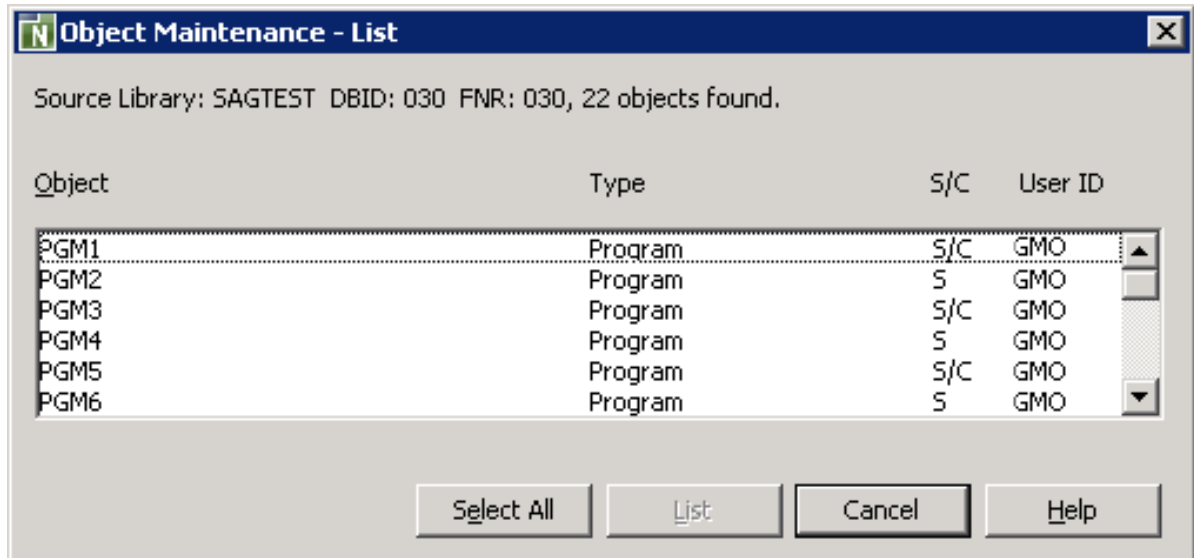
If you have specified a date, you can enter a start time in the **Time** box to list only objects that were saved or cataloged at or after this time. The default time is 00:00 (no time). The time is given in the format *HH:II* (*HH* = hours, *II* = minutes).

- 9 Choose **Object List** when you have finished specifying object selection criteria.

If you have specified a single object name, proceed with [Step 13](#).

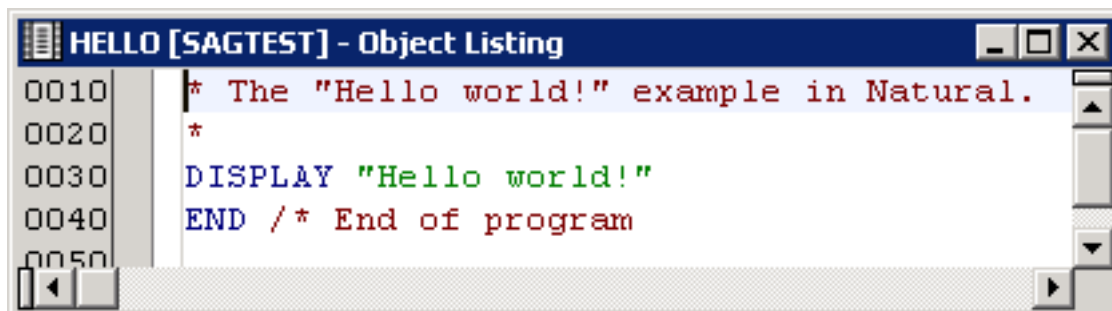
If you have specified a range of object names, proceed with Step 10.

- 10 A dialog box similar to the example below appears with a list of all matching objects:



- 11 Select or deselect single or multiple objects by proceeding as described in [To select/deselect list items](#).
- 12 Choose **List** to display the source code of the selected objects.
(**Cancel** exits the dialog box without any action.)
- 13 The **Object Maintenance - List** dialog box is closed and the **Object Maintenance** menu appears.

In addition, for each object selected, an **Object Listing** window similar to the example below appears, with the source code of the object:



The window title shows the name of the object (here: HELLO) and the name of the library (here: SAGTEST) where the object is stored. The **Object Listing** window is identical to the window that appears when using the LIST system command.

- 14 From the **Object Maintenance** menu, choose **Cancel** to exit SYSMAN and activate an open **Object Listing** window.

61 Finding Objects

The **Find** function is used to locate single or multiple objects in single or multiple libraries and display the source code of the objects found.

This section provides instructions for specifying search options in the **Object Maintenance - Find** dialog box.

▶ To find objects

- 1 In the **Library** list box, enter the name of the library in which you want to search for objects or select a library from the drop-down list. The default is the current library. If you enter an asterisk (*), a dialog box appears (see Step 10), where you can select multiple libraries from a list of all libraries available in the specified system file.

The **Library** list box does not apply to DDMs (data definition modules) in a remote mainframe environment. Therefore, the box disappears when the **Views (DDM)** check box is selected.

- 2 In the **DBID** (database ID) and **FNR** (file number) boxes, replace the values if the library is not in the current system file. The default is the current FNAT or FUSER, or FDIC for DDMs on mainframes. In a remote environment, you can specify security information for these files as described in *File Security for Remote Environments*.
- 3 In a remote environment, use the **FSEC/FDIC** button if you want to specify security information for the system file FSEC or FDIC for XRef data; see *File Security for Remote Environments*.
- 4 In the **Name** box, enter the name of a single object or specify a range of names; see *Specifying a Range of Names*. The default is asterisk (*), indicating all objects.
- 5 In the **Type** group box, select the type(s) of object by choosing one of the following options:

Select **Programming** (default) for all types of object (including program and subprogram) except DDM. You can limit the types of object by choosing **Object Types** and selecting or deselecting the required item(s) from the list provided:

- To select list items:

Click on a single item.

Or:

Press UP ARROW or DOWN ARROW to go to and select the required item.

Or:

Press and hold down CTRL or SHIFT and click on multiple non-consecutive or consecutive items respectively.

Or:

Press SHIFT+UP ARROW or SHIFT+DOWN ARROW to select multiple consecutive items.

Or:

Choose **Select All** to select all items.

- To deselect list items:

Press and hold down CTRL and click on a selected item again.

Or:

Choose **Deselect All** if all items have been selected.

Or:

Select **Views (DDM)** for DDMs only.

- 6 In the **Code** group box, select **Source** and/or **Cataloged** to find all objects for which either the source object or the cataloged object, or both exist. The default is both the source object and the cataloged object.
- 7 In the **User ID** box, enter the ID of a user if you want to find only the objects that were last saved or cataloged by this user. The default is no user ID.
- 8 In the **Date** box, enter a start date to find only objects that were saved or cataloged on or after this date. A date must be entered in the format associated with the preset value in the **Date** box. The default date is 0000.00.00 (no date). The date is given in the format *YYYY.MM.DD* (*YYYY* = year, *MM* = month, *DD* = day).

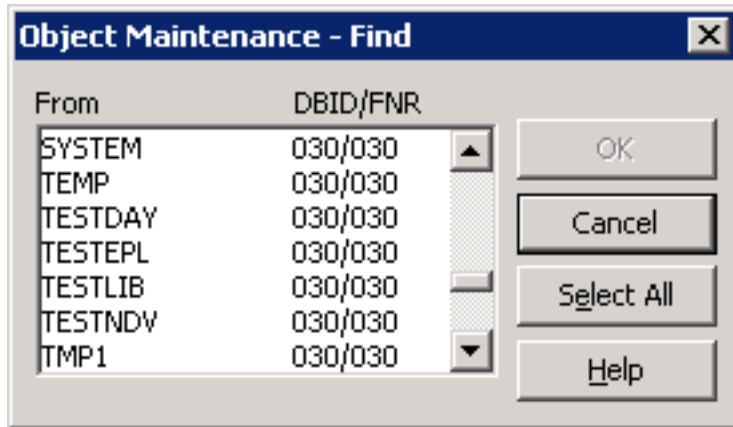
If you have specified a date, you can enter a start time in the **Time** box to find only objects that were saved or cataloged at or after this time. The default time is 00:00 (no time). The time is given in the format *HH:II* (*HH* = hours, *II* = minutes).

- 9 Choose **Object List** when you have finished specifying object selection criteria.

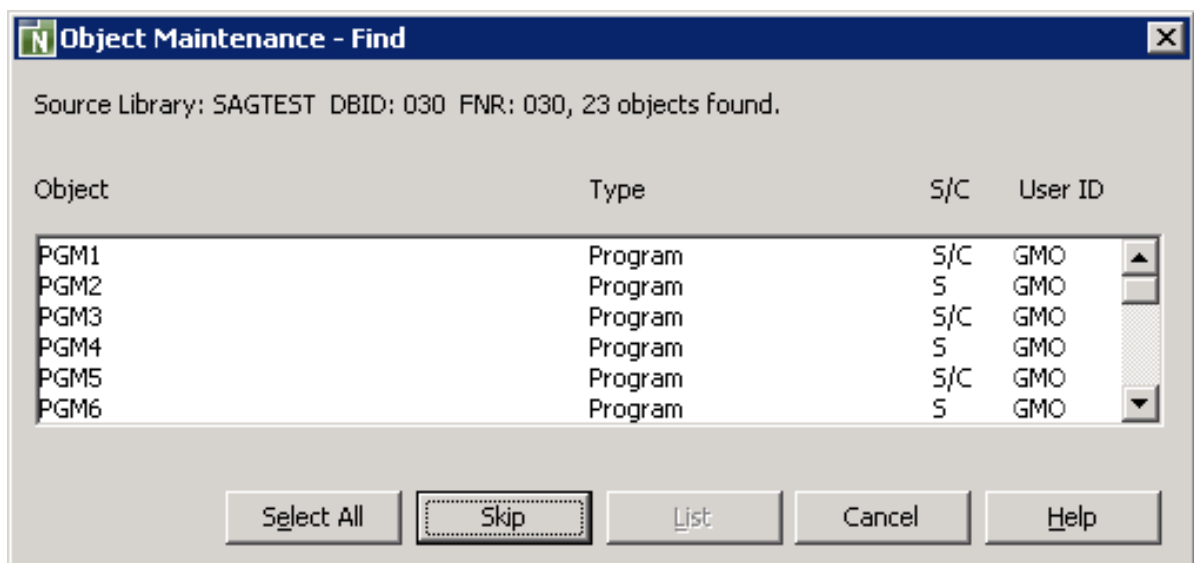
If you have specified a single library, proceed with [Step 13](#).

If you have entered an asterisk (*) in the **Library** list box, proceed with Step 10.

- 10 A dialog box similar to the example below appears with a list of all libraries available in the specified system file:



- 11 Select or deselect single or multiple libraries by proceeding as described in [To select/deselect list items](#).
- 12 Choose **OK**.
(**Cancel** exits the dialog box without any action.)
- 13 A dialog box similar to the example below appears with a list of all matching objects:



The **Skip** button only appears if you selected multiple libraries.

The dialog box shows the library locations, the number of objects found and the object names and types. It also indicates whether a source (S) and/or a cataloged (C) object exists and the ID of the user who saved and/or cataloged an object.

- 14 Select or deselect single or multiple objects by proceeding as described in [To select/deselect list items](#).

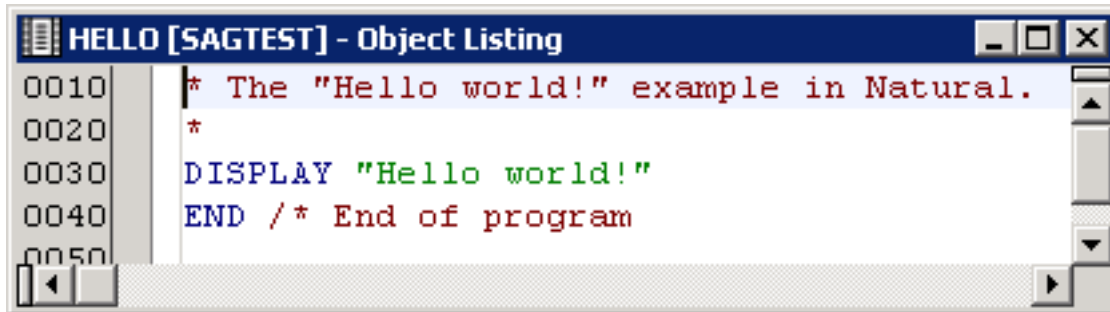
Or:

Choose **Skip** if you do not want to select any objects from the current dialog box but open the dialog box for the next library with matching objects if found. Dialog boxes are opened in alphabetical order of library names.

- 15 Choose **List** if you want to list the source code of the selected objects.

(**Cancel** exits the dialog box without any action.)

- 16 The **Object Maintenance - List** dialog box is closed and the **Object Maintenance** menu appears. In addition, for each object selected, an **Object Listing** window similar to the example below appears, with the source code of the object:



The window title shows the name of the object (here: HELLO) and the name of the library (here: SAGTEST) where the object is stored. The **Object Listing** window is identical to the window that appears when using the LIST system command.

- 17 From the **Object Maintenance** menu, choose **Cancel** to exit SYSMAIN and activate an open **Object Listing** window.

62 Copying Objects

The **Copy** function is used to copy single or multiple objects from a source environment to a target environment. The objects remain unchanged in the source environment. If the target environment already contains an object with the same name as the object to be copied, you can specify whether you overwrite the object in the target environment or give the copied object a new name.

This section provides instructions for specifying copy options in the **Object Maintenance - Copy** dialog box.

▶ To copy objects

- 1 In the **Source** group box, specify the object(s) you want to copy:
 - In the **Library** list box, enter the name of the source library that contains the object(s) you want to copy or select a library from the drop-down list. The default is the current library. The **Library** list box does not apply to DDMs (data definition modules) in a remote mainframe environment. Therefore, the box disappears when the **Views (DDM)** check box is selected.
 - In the **DBID** (database ID) and **FNR** (file number) boxes, replace the values if the source library is not in the current system file. The default is the current FNAT or FUSER system file, or FDIC for DDMs on mainframes. In a remote environment, you can specify security information for these files as described in *File Security for Remote Environments*.
 - In a remote environment, use the **FSEC/FDIC** button if you want to specify security information for the system file FSEC or FDIC for XRef data; see *File Security for Remote Environments* and *XRef Considerations*.
 - In the **Name** box, enter the name of a single object or specify a range of names; see *Specifying a Range of Names*. The default is asterisk (*), indicating all objects.
 - In the **Type** group box, select the type(s) of object by choosing one of the following options:

Select **Programming** (default) for all types of object (including program and subprogram) except DDM. You can limit the types of object by choosing **Object Types** and selecting or deselecting the required item(s) from the list provided:

- To select list items:

Click on a single item.

Or:

Press UP ARROW or DOWN ARROW to go to and select the required item.

Or:

Press and hold down CTRL or SHIFT and click on multiple non-consecutive or consecutive items respectively.

Or:

Press SHIFT+UP ARROW or SHIFT+DOWN ARROW to select multiple consecutive items.

Or:

Choose **Select All** to select all items.

- To deselect list items:

Press and hold down CTRL and click on a selected item again.

Or:

Choose **Deselect All** if all items have been selected.

Or:

Select **Views (DDM)** for DDMs only.

- In the **Code** group box, select **Source** and/or **Cataloged** to copy either the source object or the cataloged object, or both. The default is both the source object and the cataloged object.

If Predict is installed, you can select the **XREF** check box to process XRef data. See also the section [XRef Considerations](#).

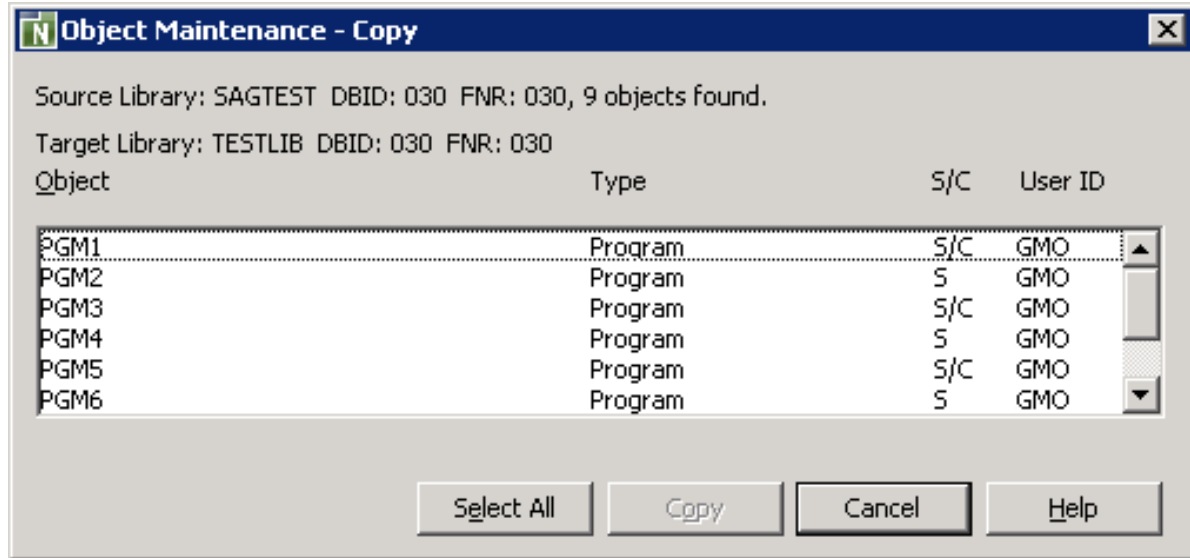
- In the **User ID** box, enter the ID of a user if you want to copy only the objects that were last saved or cataloged by this user. The default is no user ID.
- In the **Date** box, enter a start date to copy only objects that were saved or cataloged on or after this date. A date must be entered in the format associated with the preset value in the **Date** box. This value is determined by the **DTFORM** profile parameter described in the *Parameter Reference* documentation. The default date is 0000.00.00 (no date). The date is given in the format *YYYY.MM.DD* (*YYYY* = year, *MM* = month, *DD* = day).

If you have specified a date, you can enter a start time in the **Time** box to copy only objects that were saved or cataloged at or after this date and time. The default time is 00:00 (no time). The time is given in the format *HH:II* (*HH* = hours, *II* = minutes).

- 2 In the **Target** group box, specify the target environment for the object(s) selected for processing:
 - In the **Library** list box, enter the name of the target library to which you want to copy the object(s) or select a library from the drop-down list. The default is the current library. If you want to create a library, enter the name of a new library. The **Library** list box does not apply to DDMs (data definition modules) in a remote mainframe environment. Therefore, the box disappears when the **Views (DDM)** check box is selected.
 - In the **DBID** (database ID) and **FNR** (file number) boxes, replace the values if the target library is not in the current system file. The default is the current FNAT or FUSER, or FDIC for DDMs on mainframes. In a remote environment, you can specify security information for these files as described in *File Security for Remote Environments*.
 - In a remote environment, use the **FSEC/FDIC** button if you want to specify security information for the system file FSEC or FDIC for XRef data; see *File Security for Remote Environments* and *XRef Considerations*.
 - In the **Name** box, you can enter a new name for the object copied in the target environment or specify a range of new names by using asterisk (*) notation; see *Specifying a Range of Names*. The default asterisk (*) specifies that all objects contained in the target environment are replaced if relevant.
 - Use the **Confirm on replace** check box to confirm (default) or reject an object replacement. See also **Confirm on replace** below.
 - Select the **Rename** check box (selected by default) to give the copied objects new names in the target environment. The check box is dimmed for DDMs in a remote mainframe environment where you cannot rename DDMs. See also **Rename** below.
- 3 Choose **OK** when you have finished specifying the source and target environments.

If you have entered a single name in the **Name** box of the **Source** group box, skip the following instructions and proceed with **Confirm on replace** in Step 5.

If you have specified a range of names in the **Name** box of the **Source** group box, an additional dialog box similar to the example below appears with a list of all matching objects:

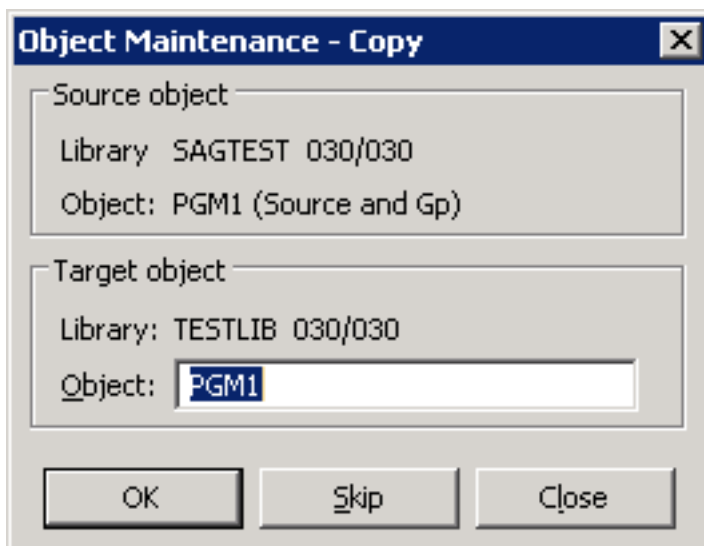


The dialog box shows the library locations, the number of objects found and the object names and types. It also indicates whether a source (S) and/or a cataloged (C) object exists and the ID of the user who saved and/or cataloged an object.

- 4 Select or deselect single or multiple objects by proceeding as described in [To select/deselect list items](#).
- 5 Choose **Copy** to process the selected object(s).

(**Cancel** exits the dialog box without any action.)

- If the **Rename** check box has been selected, an additional dialog box similar to the following example appears that displays, one after the other, each object to be copied. (No additional dialog box appears if a single name, rather than a range, is entered in the **Name** box of the **Source** group box.)



Choose one of the following options:

In the **Object** box of the current object, enter a new name for the object in the target environment. Choose **OK** to confirm the rename.

Or:

Choose **Skip** (or ALT+S) if you want to remove the current object from the list of selected objects and proceed with the next object.

Or:

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

- If the **Confirm on replace** check box has been selected and if the target environment already contains an object with the same name as the object to be copied, an additional dialog box appears with a warning message.

Confirm or reject object replacement by choosing one of the following buttons:

Yes to confirm each object replacement individually one after another.

Or:

Yes to All to confirm all object replacements in one go.

Or:

No to not replace the current object.

Or:

Cancel to exit the dialog box without any action.

- 6 After all objects have been copied, the **Object Maintenance - Copy** dialog box is closed and the **Object Maintenance** menu appears.

63

Moving Objects

The **Move** function is used to transfer an object from a source environment to a target environment. The object is deleted from the source environment and added to the target environment. If the target environment already contains an object with the same name as the object to be moved, you can specify whether you overwrite the object in the target environment or give the copied object a new name.

This section provides instructions for specifying move options in the **Object Maintenance - Move** dialog box.

▶ To move objects

- 1 In the **Source** group box, specify the object(s) you want to move:
 - In the **Library** list box, enter the name of the source library that contains the object(s) you want to move or select a library from the drop-down list. The default is the current library. The **Library** list box does not apply to DDMs (data definition modules) in a remote mainframe environment. Therefore, the box disappears when the **Views (DDM)** check box is selected.
 - In the **DBID** (database ID) and **FNR** (file number) boxes, replace the values if the source library is not in the current system file. The default is the current FNAT or FUSER, or FDIC for DDMs on mainframes. In a remote environment, you can specify security information for these files as described in *File Security for Remote Environments*.
 - In a remote environment, use the **FSEC/FDIC** button if you want to specify security information for the system file FSEC or FDIC for XRef data; see *File Security for Remote Environments* and *XRef Considerations*.
 - In the **Name** box, enter the name of a single object or specify a range of names; see *Specifying a Range of Names*. The default is asterisk (*), indicating all objects.
 - In the **Type** group box, select the type(s) of object by choosing one of the following options:

Select **Programming** (default) for all types of object (including program and subprogram) except DDM. You can limit the types of object by choosing **Object Types** and selecting or deselecting the required item(s) from the list provided:

- To select list items:

Click on a single item.

Or:

Press UP ARROW or DOWN ARROW to go to and select the required item.

Or:

Press and hold down CTRL or SHIFT and click on multiple non-consecutive or consecutive items respectively.

Or:

Press SHIFT+UP ARROW or SHIFT+DOWN ARROW to select multiple consecutive items.

Or:

Choose **Select All** to select all items.

- To deselect list items:

Press and hold down CTRL and click on a selected item again.

Or:

Choose **Deselect All** if all items have been selected.

Or:

Select **Views (DDM)** for DDMs only.

- In the **Code** group box, select **Source** and/or **Cataloged** to move either the source object or the cataloged object, or both. The default is both the source object and the cataloged object.

If Predict is installed, you can select the **XREF** check box to process XRef data. See also the section [XRef Considerations](#).

- In the **User ID** box, enter the ID of a user if you want to move only the objects that were last saved or cataloged by this user. The default is no user ID.
- In the **Date** box, enter a start date to move only objects that were saved or cataloged on or after this date. A date must be entered in the format associated with the preset value in the **Date** box. This value is determined by the **DTFORM** profile parameter described in the *Parameter Reference* documentation. The default date is 0000.00.00 (no date). The date is given in the format *YYYY.MM.DD* (*YYYY* = year, *MM* = month, *DD* = day).

If you have specified a date, you can enter a start time in the **Time** box to move only objects that were saved or cataloged at or after this date and time. The default time is 00:00 (no time). The time is given in the format *HH:II* (*HH* = hours, *II* = minutes).

2 In the **Target** group box, specify the target environment for the object(s) selected for processing:

- In the **Library** list box, enter the name of the target library where you want to place the object(s) or select a library from the drop-down list. The default is the current library. If you want to create a library, enter the name of a new library.

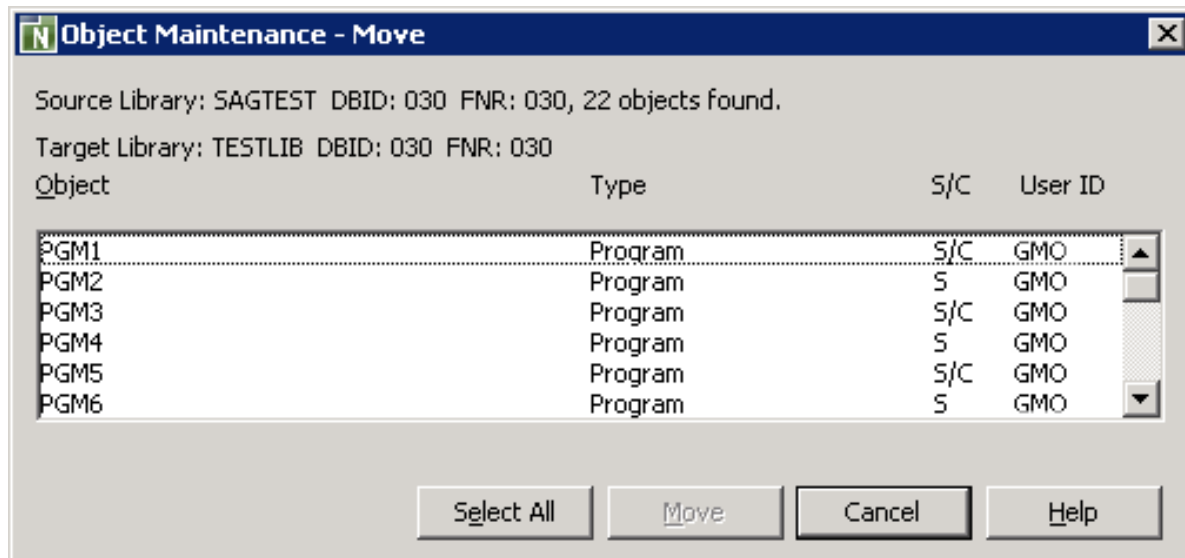
The **Library** list box does not apply to DDMs (data definition modules) in a remote mainframe environment. Therefore, the box disappears when the **Views (DDM)** check box is selected.

- In the **DBID** (database ID) and **FNR** (file number) boxes, replace the values if the target library is not in the current system file. The default is the current FNAT or FUSER, or FDIC for DDMs on mainframes. In a remote environment, you can specify security information for these files as described in *File Security for Remote Environments*.
- In a remote environment, use the **FSEC/FDIC** button if you want to specify security information for the system file FSEC or FDIC for XRef data; see *File Security for Remote Environments* and *XRef Considerations*.
- In the **Name** box, you can enter a new name for the object in the target environment or specify a range of new names by using asterisk (*) notation; see *Specifying a Range of Names*. The default asterisk (*) specifies that all objects contained in the target environment are replaced if relevant.
- Use the **Confirm on replace** check box to confirm (default) or reject object replacement. See also **Confirm on replace** below.
- Select the **Rename** check box (selected by default) to give the moved objects new names in the target environment. The check box is dimmed for DDMs in a remote mainframe environment where you cannot rename DDMs. See also **Rename** below.

3 Choose **OK** when you have finished specifying the source and target environments.

If you have entered a single name in the **Name** box of the **Source** group box, skip the following instructions and proceed with **Confirm on replace** in Step 5.

If you have specified a range of names in the **Name** box of the **Source** group box, an additional dialog box similar to the example below appears with a list of all matching objects:

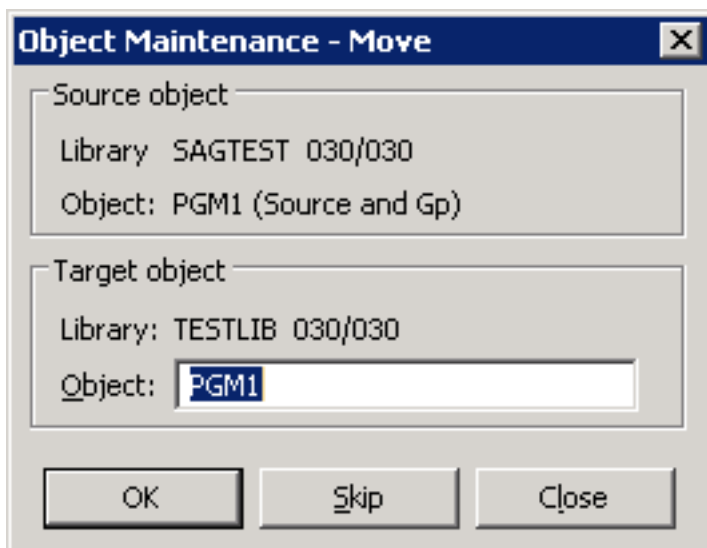


The dialog box shows the library locations, the number of objects found and the object names and types. It also indicates whether a source (S) and/or a cataloged (C) object exists and the ID of the user who saved and/or cataloged an object.

- 4 Select or deselect single or multiple objects by proceeding as described in [To select/deselect list items](#).
- 5 Choose **Move** to process the selected object(s).

(**Cancel** exits the dialog box without any action.)

- If the **Rename** check box has been selected, an additional dialog box similar to the following example appears that displays, one after the other, each object to be moved. (No additional dialog box appears if a single name, rather than a range, is entered in the **Name** box of the **Source** group box.)



Choose one of the following options: In the **Object** box of the current object, enter a new name for the object in the target environment. Choose **OK** to confirm the rename.

Or:

Choose **Skip** (or ALT+S) if you want to remove the current object from the list of selected objects and proceed with the next object.

Or:

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

- If the **Confirm on replace** check box has been selected and if the target environment already contains an object with the same name as the object to be moved, an additional dialog box appears with a warning message.

Confirm or reject object replacement by choosing one of the following buttons:

Yes to confirm each object replacement individually one after another.

Or:

Yes to All to confirm all object replacements in one go.

Or:

No to not replace the current object.

Or:

Cancel to exit the dialog box without any action.

- 6 After all objects have been moved, the **Object Maintenance - Move** dialog box is closed and the **Object Maintenance** menu appears.

64

Deleting Objects

The **Delete** function is used to delete single or multiple objects from a source environment.

This section provides instructions for specifying delete options in the **Object Maintenance - Delete** dialog box.

▶ **To delete objects**

- 1 In the **Library** list box, enter the name of the library that contains the object(s) you want to delete or select a library from the drop-down list. The default is the current library. The **Library** list box does not apply to DDMs (data definition modules) in a remote mainframe environment. Therefore, the box disappears when the **Views (DDM)** check box is selected.
- 2 In the **DBID** (database ID) and **FNR** (file number) boxes, replace the values if the library is not in the current system file. The default is the current FNAT or FUSER, or FDIC for DDMs on mainframes. In a remote environment, you can specify security information for these files as described in *File Security for Remote Environments*.
- 3 In a remote environment, use the **FSEC/FDIC** button if you want to specify security information for the system file FSEC or FDIC for XRef data; see *File Security for Remote Environments* and *XRef Considerations*.
- 4 In the **Name** box, enter the name of a single object or specify a range of names; see *Specifying a Range of Names*. The default is asterisk (*), indicating all objects.
- 5 In the **Type** group box, select the type(s) of object by choosing one of the following options:

Select **Programming** (default) for all types of object (including program and subprogram) except DDM. You can limit the types of object by choosing **Object Types** and selecting or deselecting the required item(s) from the list provided:

- To select list items:

Click on a single item.

Or:

Press UP ARROW or DOWN ARROW to go to and select the required item.

Or:

Press and hold down CTRL or SHIFT and click on multiple non-consecutive or consecutive items respectively.

Or:

Press SHIFT+UP ARROW or SHIFT+DOWN ARROW to select multiple consecutive items.

Or:

Choose **Select All** to select all items.

- To deselect list items:

Press and hold down CTRL and click on a selected item again.

Or:

Choose **Deselect All** if all items have been selected.

Or:

Select **Views (DDM)** for DDMs only.

- 6 In the **Code** group box, select **Source** and/or **Cataloged** to delete all objects for which either the source object or the cataloged object, or both exist. The default is both the source object and the cataloged object.

If Predict is installed, you can select the **XREF** check box to delete XRef data. See also the section *XRef Considerations*.

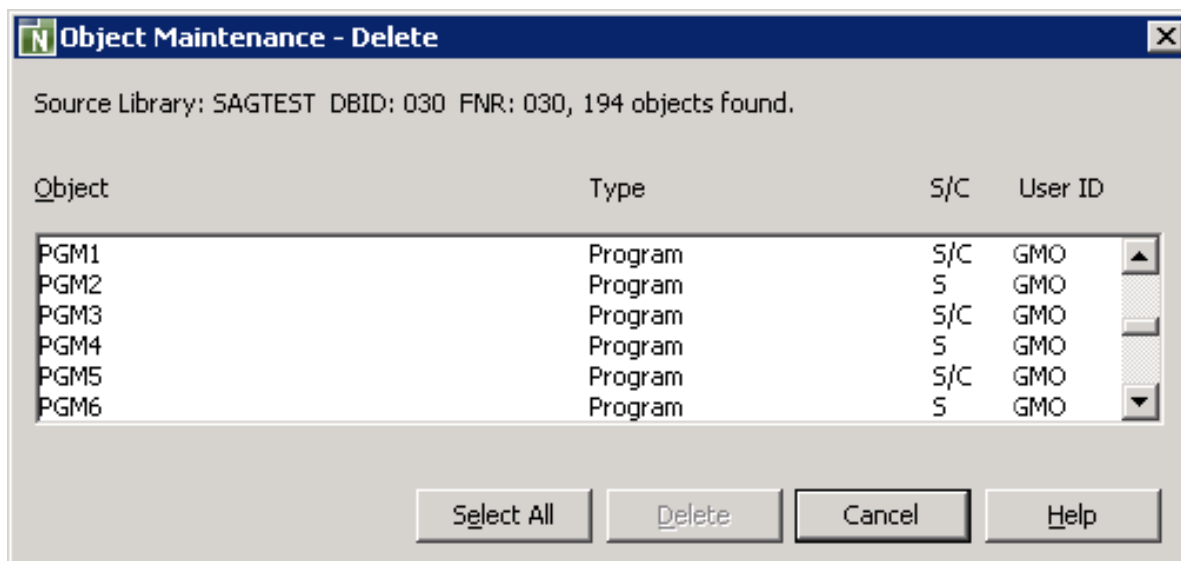
- 7 In the **User ID** box, enter the ID of a user if you want to delete only the objects that were last saved or cataloged by this user. The default is no user ID.
- 8 In the **Date** box, enter a start date to delete only objects that were saved or cataloged on or after this date. A date must be entered in the format associated with the preset value in the **Date** box. The default date is 0000.00.00 (no date). The date is given in the format *YYYY.MM.DD* (*YYYY* = year, *MM* = month, *DD* = day).

If you have specified a date, you can enter a start time in the **Time** box to delete only objects that were saved or cataloged at or after this time. The default time is 00:00 (no time). The time is given in the format *HH:II* (*HH* = hours, *II* = minutes).

- 9 Choose **Object List** when you have finished specifying object selection criteria.

If you have entered a single name in the **Name** box, skip the following instructions and proceed with **Confirm on delete** in Step 11.

If you have specified a range of names in the **Name** box, an additional dialog box similar to the example below appears with a list of all matching objects:



The dialog box shows the library location, the number of objects found and the object names and types. It also indicates whether a source (S) and/or a cataloged (C) object exists and the ID of the user who saved and/or cataloged an object.

- 10 Select or deselect single or multiple objects by proceeding as described in [To select/deselect list items](#).
- 11 Choose **Delete** to process the selected object(s).

(**Cancel** exits the dialog box without any action.)

If the **Confirm on delete** check box has been selected, an additional dialog box appears with a warning message.

Confirm or reject object deletion by choosing one of the following buttons:

Yes to confirm each object deletion individually one after another.

Or:

Yes to All to confirm all object deletions in one go.

Or:

No to not delete the current object.

Or:

Cancel to exit the dialog box without any action.

- 12 After all objects have been deleted, the **Object Maintenance - Delete** dialog box is closed and the **Object Maintenance** menu appears.

65 Renaming Objects

The **Rename** function is used to give single or multiple objects new names within a source environment.

The **Rename** function does not apply to data definition modules (DDMs) in a remote environment on a mainframe platform.

This section provides instructions for specifying rename options in the **Object Maintenance - Rename** dialog box.

▶ To rename objects

- 1 In the **Source** group box, specify the object(s) you want to rename:
 - In the **Library** list box, enter the name of the library that contains the object(s) you want to rename or select a library from the drop-down list. The default is the current library.
 - In the **DBID** (database ID) and **FNR** (file number) boxes, replace the values if the library is not in the current system file. The default is the current FNAT or FUSER, or FDIC for DDMs on mainframes. In a remote environment, you can specify security information for these files as described in *File Security for Remote Environments*.
 - In a remote environment, use the **FSEC/FDIC** button if you want to specify security information for the system file FSEC or FDIC for XRef data; see *File Security for Remote Environments* and *XRef Considerations*.
 - In the **Name** box, enter the name of a single object or specify a range of names; see *Specifying a Range of Names*. The default is asterisk (*), indicating all objects.
 - In the **Type** group box, select the type(s) of object by choosing one of the following options:

Select **Programming** (default) for all types of object (including program and subprogram) except DDM. You can limit the types of object by choosing **Object Types** and selecting or deselecting the required item(s) from the list provided:

- To select list items:

Click on a single item.

Or:

Press UP ARROW or DOWN ARROW to go to and select the required item.

Or:

Press and hold down CTRL or SHIFT and click on multiple non-consecutive or consecutive items respectively.

Or:

Press SHIFT+UP ARROW or SHIFT+DOWN ARROW to select multiple consecutive items.

Or:

Choose **Select All** to select all items.

- To deselect list items:

Press and hold down CTRL and click on a selected item again.

Or:

Choose **Deselect All** if all items have been selected.

Or:

Select **Views (DDM)** for DDMs only (not applicable to DDMs on mainframes).

- In the **Code** group box, select **Source** and/or **Cataloged** to rename either the source object or the cataloged object, or both. The default is both the source object and the cataloged object.

If Predict is installed, you can select the **XREF** check box to process XRef data. See also the section *XRef Considerations*.

- In the **User ID** box, enter the ID of a user if you want to rename only the objects that were last saved or cataloged by this user. The default is no user ID.
- In the **Date** box, enter a start date to rename only objects that were saved or cataloged on or after this date. A date must be entered in the format associated with the preset value in the **Date** box. This value is determined by the `DTFORM` profile parameter described in the *Parameter Reference* documentation. The default date is `0000.00.00` (no date). The date is given in the format `YYYY.MM.DD` (`YYYY` = year, `MM` = month, `DD` = day).

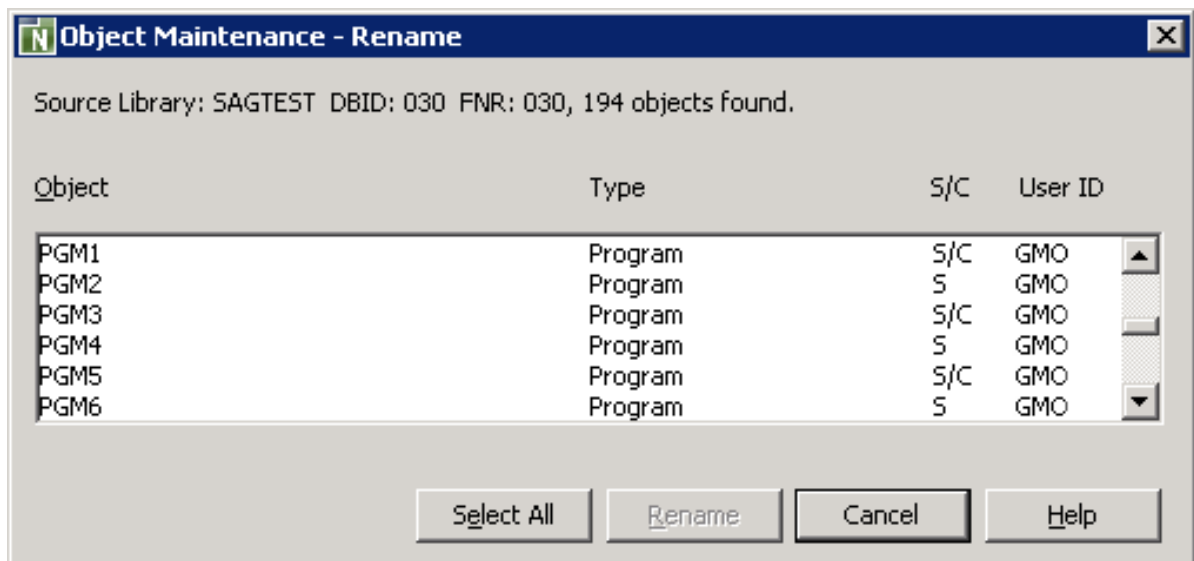
If you have specified a date, you can enter a start time in the **Time** box to rename only objects that were saved or cataloged at or after this date and time. The default time is `00:00` (no time). The time is given in the format `HH:II` (`HH` = hours, `II` = minutes).

- 2 In the **Target** group box, you can specify the following:

- In the **Name** box, enter a new name or a range of new names by using asterisk (*) notation; see [Specifying a Range of Names](#). The default asterisk (*) denotes that all objects specified in the **Name** box of the **Source** group box are to be renamed.
 - Use the **Confirm on replace** check box to confirm (default) or reject object renaming. See also [Confirm on replace](#) below.
- 3 Choose **OK** when you have finished specifying the renaming conditions.

If you entered a single name in the **Name** boxes of the **Source** and the **Target** group boxes, skip the following instructions and proceed with [Confirm on replace](#) in Step 5.

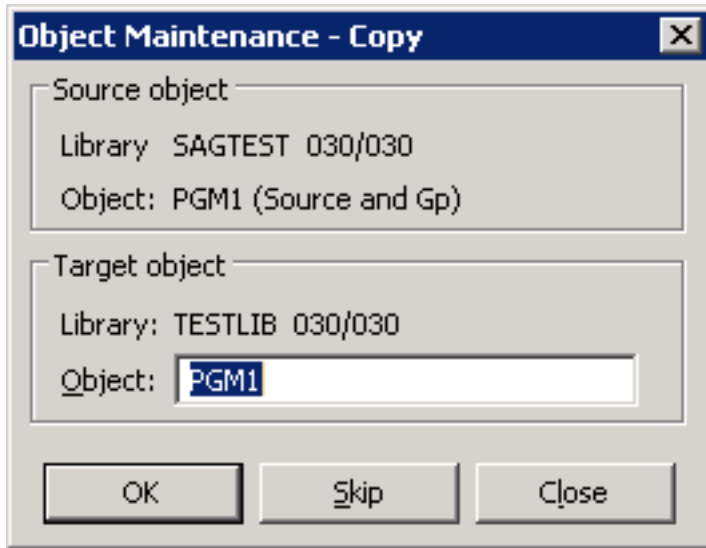
If you entered a range of names in the **Name** box of the **Source** group box, an additional dialog box similar to the example below appears with a list of all matching objects:



The dialog box shows the library locations, the number of objects found and the object names and types. It also indicates whether a source (S) and/or a cataloged (C) object exists and the ID of the user who saved and/or cataloged an object.

- 4 Select or deselect single or multiple objects by proceeding as described in [To select/deselect list items](#).
- 5 Choose **Rename** to process the selected object(s).
- (**Cancel** exits the dialog box without any action.)

- If you specified a range of names in the **Name** box, an additional dialog box similar to the following example appears that displays, one after the other, each object to be renamed:



Choose one of the following options:

In the **Object** box of the current object, enter a new name. Choose **OK** to confirm the rename.

Or:

Choose **Skip** (or ALT+S) if you want to remove the current object from the list of selected objects and proceed with the next object.

Or:

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

- If the **Confirm on replace** check box has been selected and if the target environment already contains an object with the same name as the object to be renamed, an additional dialog box appears with a warning message.

Confirm or reject object replacement by choosing one of the following buttons:

Yes to confirm each object replacement individually one after another.

Or:

Yes to All to confirm all object replacements in one go.

Or:

No to not replace the current object.

Or:

Cancel to exit the dialog box without any action.

- 6 After all objects have been renamed, the **Object Maintenance - Rename** dialog box is closed and the **Object Maintenance** menu appears.

66

Importing Objects

The **Import** function of SYSMAIN is used to copy objects (files) from an external source to a Natural library. Alternatively, you can use the unload and load functions of the **Object Handler** (see the relevant documentation).

When you import objects, the file directory *FILEDIR.SAG* of the target library is automatically updated to contain information on the newly imported objects. Be aware that Natural will *not* update the file directory *FILEDIR.SAG* if you use a non-Natural function or facility (such as the Windows Explorer) to copy objects to a Natural library. As a result, you cannot access the objects contained in this library. *FILEDIR.SAG* contains internal library information required by Natural such as programming mode (structured or reporting), object form (source object and/or cataloged object) and user ID.

The objects to be imported with SYSMAIN must have been created with Natural.



Note: You cannot import shared resources into a remote environment located on a mainframe platform.

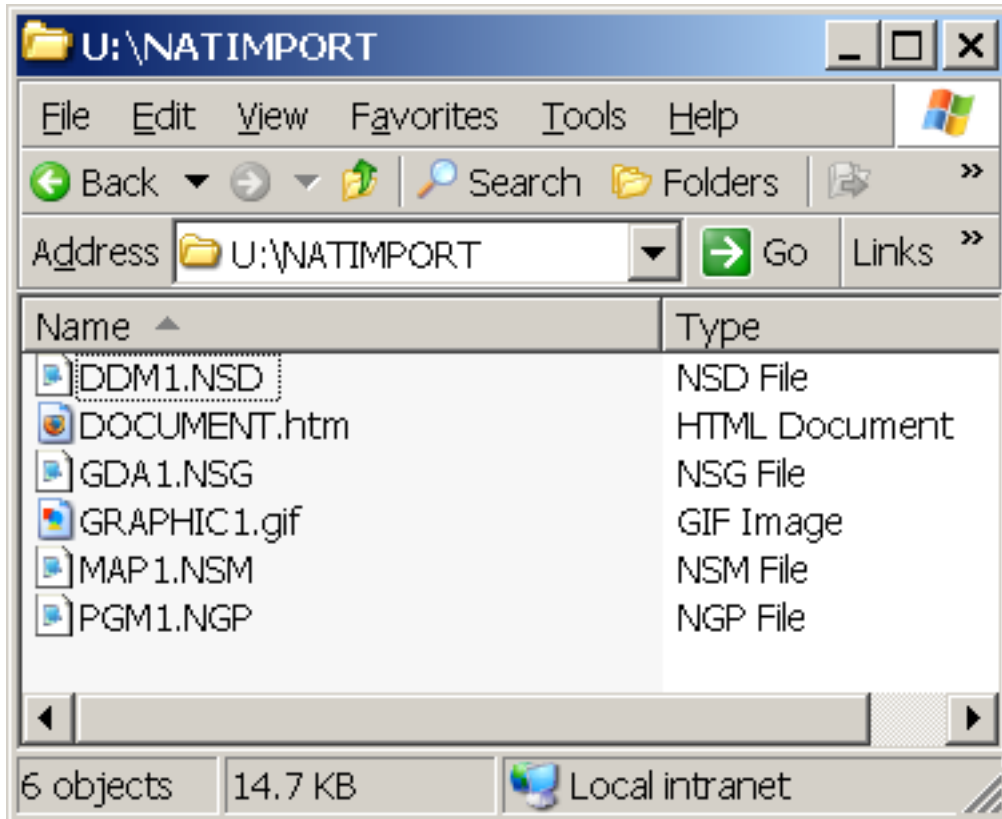
▶ To import an object

- 1 In the **Source** group box, specify the object(s) you want to import:
 - In the **Path** box, enter the name of the path of the folder that contains the objects you want to import. The default is the directory path used when starting Natural.

Or:

From the **Directory** list box, select the path of the folder that contains the objects you want to import.

Remember that a file containing a Natural object must have the appropriate extension as indicated in the following example:



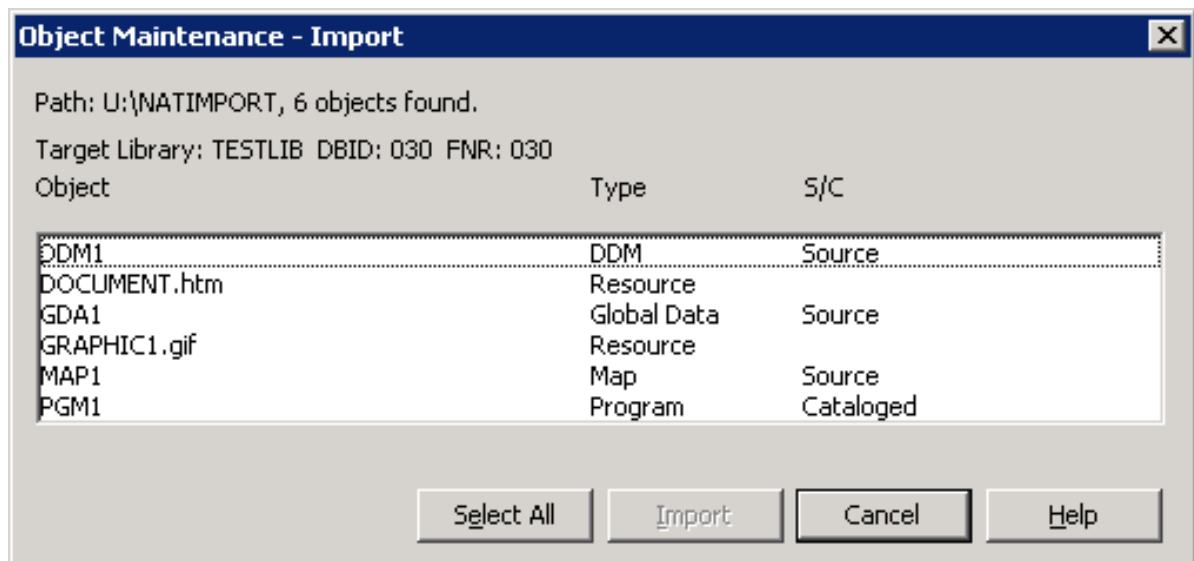
In the **Name** box, enter the name of a single object you want to import or specify a range of names; see [Specifying a Range of Names](#). The default is asterisk (*), indicating all objects.

- In the **Code** group box, select **Source** and/or **Cataloged** to import the object either as a source object or a cataloged object, or both. The default is both the source object and the cataloged object.
- 2 In the **Target** group box, specify the target environment for the objects to be imported:
- In the **Library** list box, enter the name of the library into which you want to import the objects or select a library from the drop-down list. The default is the current library. If you want to create a library, enter the name of a new library.
 - In the **DBID** (database ID) and **FNR** (file number) boxes, replace the values if the target library is not in the current system file. The default is the current FNAT or FUSER, or FDIC for DDMs on mainframes. In a remote environment, you can specify security information for these files as described in [File Security for Remote Environments](#).
 - In a remote environment, use the **FSEC/FDIC** button if you want to specify security information for the system file FSEC or FDIC for XRef data; see [File Security for Remote Environments](#) and [XRef Considerations](#).
 - In the **User ID** box, enter the ID of the user you want to appear in the object properties or object directory information. If you leave the box empty (default), the ID specified with the *USER system variable is used (see also the *System Variables* documentation).

- In the **Mode** box, you can select the programming mode in which the imported object is to be saved. The default is set according to the current setting of the SM parameter. To set or change the programming mode, see *Setting/Changing the Programming Mode*.
 - Use the **Confirm on replace** check box to confirm (default) or reject object replacement. See also **Confirm on replace** below.
- 3 Choose **Object List** when you have finished specifying the source and target environments.

If you entered a single name in the **Name** box, skip the following instructions and proceed with **Confirm on replace** in Step 5.

If you specified a range of names in the **Name** box, an additional dialog box similar to the example below appears with a list of all matching objects:



The dialog box shows the name of the source path, the number of objects found and the object names and types. It also indicates whether a source (S) and/or a cataloged (C) object exists.

- 4 Select or deselect the required object(s) from the list:
- To select list items:
 - Click on a single item.
 - Or:
Press UP ARROW or DOWN ARROW to go to and select the required item.
 - Or:
Press and hold down CTRL or SHIFT and click on multiple non-consecutive or consecutive items respectively.

Or:

Press **SHIFT+UP ARROW** or **SHIFT+DOWN ARROW** to select multiple consecutive items.

Or:

Choose **Select All** to select all items.

- To deselect list items:

Press and hold down **CTRL** and click on a selected item again.

Or:

Choose **Deselect All** if all items have been selected.

- 5 Choose **Import** to process the selected object(s).

(**Cancel** exits the dialog box without any action.)

If the **Confirm on replace** check box has been selected and if the target environment already contains an object with the same name as the object to be imported, an additional dialog box appears with a warning message.

Confirm or reject object replacement by choosing one of the following buttons:

Yes to confirm each object replacement individually one after another.

Or:

Yes to All to confirm all object replacements in one go.

Or:

No to not replace the current object.

Or:

Cancel to exit the dialog box without any action.

- 6 After all objects have been imported into the specified library, the **Object Maintenance - Import** dialog box is closed and the **Object Maintenance** menu appears.

An imported objects is contained in the library folder appropriate for its type. For example, a Natural object of the type program is contained in the **Programs** folder (in the Logical View) of the library. If a type of object is imported for which no folder yet exists, Natural automatically creates the appropriate folder when performing the **Import**.

67

Using SYSMAIN with Subprogram

▪ Invoking and Executing MAINUSER	416
▪ Using Commands	416
▪ LIST and FIND Command Syntax	417
▪ COPY and MOVE Command Syntax	418
▪ DELETE Command Syntax	418
▪ RENAME Command Syntax	419
▪ IMPORT Command Syntax	419
▪ where-clause	420
▪ with-clause	420
▪ Keywords and Variables in Commands	420
▪ Specifying a Range of Names	425

The MAINUSER subprogram is an Application Programming Interface, which allows you to perform SYSMAIN utility functions from any user-written object (subroutine, program or subprogram) as an alternative to using SYSMAIN utility menus. Upon completion of the SYSMAIN function, the utility is terminated and control is returned to the object from which the request was issued. MAINUSER can be used in either online or batch mode. An example of a callable routine is the MAINCALL program, which is supplied in the SYSMAIN system library.

This section provides instructions for using MAINUSER and the syntax that applies when specifying commands for executing SYSMAIN utility functions.

Invoking and Executing MAINUSER

► To invoke and execute MAINUSER

- Issue a CALLNAT statement that contains the following syntax elements:

```
CALLNAT 'MAINUSER' command error message library
```

where the variable values denote the following parameters:

Parameter	Natural Data Format/Length	Explanation
<i>command</i>	A250	The command string to be executed by SYSMAIN: see Using Commands .
<i>error</i>	N4	The return code issued by SYSMAIN at the end of processing to indicate a normal end of processing or an error.
<i>message</i>	A72	The message corresponding to the error given online.
<i>library</i>	A8	The name of the library containing the utility SYSMAIN; by default, this is the library SYSMAIN. (This parameter is provided for compatibility reasons only.)

Using Commands

SYSMAIN functions can be executed using commands issued as a parameter of the MAINUSER subprogram.

A *command* consists of keywords and variable values. For each SYSMAIN function to be performed, the keywords and variable values are shown in the corresponding syntax diagrams below and explained in the section [Keywords and Variables in Commands](#). The symbols in the syntax diagrams

correspond to the syntax symbols used for system commands. These symbols are explained in *System Command Syntax* in the *System Commands* documentation.

The sequence of the command syntax is not completely fixed. The following rules apply:

- SYSMAIN function, object type and object name must be the first three parameters of the command string.
- A period (.) indicates the end of a command. If this character is detected anywhere within a command string, all subsequent data is ignored.
- In the syntax diagrams, FM or IN is shown instead of the FROM keyword to make the diagrams easier to read; however, FROM can always be used as a synonym for FM or IN and vice versa.
- The syntax of the *where-clause* and the *with-clause* is identical for each command.

LIST and FIND Command Syntax

The following command syntax applies to the find and list functions:

$\left\{ \begin{array}{l} \underline{L}IST \\ \underline{F}IND \end{array} \right\}$	$\left[\begin{array}{l} \underline{ALL} \\ \underline{CATALOGED} \\ \underline{SAVED} \\ \underline{STOWED} \\ \underline{VIEW} \end{array} \right]$	$name \left[\underline{IN} \left[\underline{LIBRARY} \right] lib-name \right] \left[\underline{where-clause} \right] \left[\underline{with-clause} \right]$
--	---	---

Examples of LIST and FIND

```
LIST VIEW * IN TESTLIB
```

```
L SAVED TEST* IN TESTLIB TYPE PNS FNR 6
```

```
L SA TEST* IN TESTLIB FNR 6 DBID 2 TYPE PM FMDATE 2007-01-01
```

```
FIND PROG1 IN * DBID 1 FNR 6
```

```
F STOWED MAINMENU IN SYS* WHERE DBID 1 FNR 5
```

```
FIND ALL PROG2 IN PROD* FNR 27 DBID 1
```

COPY and MOVE Command Syntax

The following command syntax applies to the copy and move functions:

{ COPY } { MOVE }	[ALL CATALOGED SAVED STOWED VIEW RESOURCE]	<i>name</i> [FM [LIBRARY] <i>lib-name</i>]	[<i>where-clause</i>]
		TO [LIBRARY] <i>lib-name</i>	[<i>where-clause</i>] [<i>with-clause</i>]

Examples of COPY and MOVE

```
COPY PROG1 FM TESTORD TO ORDERS DBID 1 FNR 6 REP
```

```
C PGM* FM TESTLIB TO PRODLIB WITH REP TYPE PNS
```

```
C VIEW PERS FM OLDLIB FNR 10 TO NEWLIB FNR 16 REPLACE
```

```
MOVE VIEW PERSONNEL FM OLDLIB FNR 20 TO NEWLIB FNR 24
```

```
M PROG1 TO NEWLIB
```

```
M STOWED * FM OLDLIB TO NEWLIB WHERE DBID 100 FNR 160 WITH XREF Y
```

DELETE Command Syntax

The following command syntax applies to the delete function:

DELETE	[ALL CATALOGED SAVED STOWED VIEW RESOURCE]	<i>name</i> [IN [LIBRARY] <i>lib-name</i>]	[<i>where-clause</i>] [<i>with-clause</i>]
--------	---	--	--

Examples of DELETE

```
DELETE SA * IN LIBTEST TYPE GLA
```

```
D * IN TESTORD TYPE PM
```

```
D VIEW FINANCE IN TESTLIB DBID 12 FNR 27
```

RENAME Command Syntax

The following command syntax applies to the rename function:

RENAME	[<ul style="list-style-type: none"> ALL CATALOGED SAVED STOWED VIEW RESOURCE]	name AS <i>new-name</i> [<i>with-clause</i>] FM [LIBRARY] <i>lib-name</i> [<i>where-clause</i>] TO [LIBRARY] <i>lib-name</i> [<i>where-clause</i>]
--------	---	--

Examples of RENAME

```
RENAME PGM1 AS PROG1
```

```
R PGM1 AS PROG1 FM TESTLIB DBID 1 FNR 5 TO PRODLIB DBID 2 FNR 6
```

IMPORT Command Syntax

The following command syntax applies to the import function:

IMPORT	[<ul style="list-style-type: none"> ALL CATALOGED SAVED STOWED VIEW RESOURCE]	name FM [PATH] <i>path-name</i> TO [LIBRARY] <i>lib-name</i> [<i>where-clause</i>] [<i>with-clause</i>]
--------	---	--

Examples of IMPORT

```
IMPORT ALL PGM* FM D:\NAT-PROGRAMS TO IMP-LIB
```

```
I RES res1.bmp FM D:\RESOURCES TO IMP-LIB
```

where-clause

```
[WHERE] [DBID dbid] [FNR fnr]
          [DIC (dbid,fnr,password,cipher)]
          [SEC (dbid,fnr,password,cipher)]
```

Separators

Commas must be used as separators between the values following the DIC and SEC keywords, or if a value is missing. For example: DIC (10, ,secret,2a). If the ID session parameter (see also *ID - Input Delimiter Character in the Parameter Reference* documentation) has been set to a comma, use a slash (/) as the separator between values.

with-clause

```
[WITH] [TYPE type] [FMDATE date] [FMTIME time]
        [USER user-id] [ XREF { Y } ]
        [REPLACE] [RCOP] [NOPROMPT] [HELP]
        [ STRUCT ]
        [ SM ]
        [ REPORT ]
```

Keywords and Variables in Commands

This section explains the keywords and corresponding variable values (if required) used in a command.

Keywords are listed alphabetically. Letters in italics represent variable values that must be supplied with a keyword. For each variable value, the Natural data format and length is indicated.

Keyword	Value	Natural Data Format/Length	Explanation
ALL	<i>name</i>	A9	Only applies to programming objects. The name of the object to be processed or a range of names; see Specifying a Range of Names . Any saved (source) objects and/or cataloged objects are processed.
CATALOGED	<i>name</i>	A9	Only applies to programming objects. The name of the cataloged object to be processed or a range of names; see Specifying a Range of Names .
SAVED	<i>name</i>	A9	Only applies to programming objects. The name of the saved (source) object to be processed or a range of names; see Specifying a Range of Names .
STOWED	<i>name</i>	A9	Only applies to programming objects. The name of an object (or a range of names) for which the saved (source) <i>and</i> the cataloged object are to be processed (see also Specifying a Range of Names). Only an object that exists as both a saved (source) object <i>and</i> a cataloged object is processed. The exceptions to this are copycode and text, neither of which can be cataloged. However, they are included in processing when this option is specified.
VIEW	<i>name</i>	A32	Only applies to DDMs (data definition modules). The name of the DDM to be processed or a range of names; see Specifying a Range of Names .
RESOURCE	<i>name</i>	A255	Only applies to shared resources. The name of the shared resource to be processed or a range of names; see Specifying a Range of Names .
FROM or FM or IN	<i>lib-name</i> or <i>path-name</i>	A8 or A253	Specifies a source library or a source path. The source library or path contains the object to be processed.
TO	<i>lib-name</i>	A8	Specifies a target library.
AS	<i>new-name</i>	A8 or A32 or A255	The new name to be given to an object when it is renamed with the RENAME command. Format/length A8 applies to programming objects, A32 to DDMs and A255 to shared resources.
LIBRARY	<i>lib-name</i>	A8	An optional keyword that indicates the name (<i>lib-name</i>) of a source or a target library. If you omit the keyword and

Keyword	Value	Natural Data Format/Length	Explanation
			<p>respective value, the library where you logged on before you invoked SYSMAIN is used for processing.</p> <p>The source library contains the object to be processed. The target library is the library to which the object is to be copied or moved, or where the object is renamed.</p> <p><i>lib-name</i> must be specified immediately after the FROM/FM/IN or TO keyword. If LIBRARY is used, it must be entered between FROM/FM/IN or TO and <i>lib-name</i>.</p>
PATH	<i>path-name</i>	A253	<p>Only applies to the IMPORT command.</p> <p>An optional keyword that indicates the name (<i>path-name</i>) of a source path.</p> <p><i>path-name</i> must be specified immediately after the FROM/FM/IN or TO keyword. If PATH is used, it must be entered between FROM/FM/IN or TO and <i>path-name</i>.</p>
WHERE	<i>where-clause</i>	-	<p>An optional keyword that indicates the start of a <i>where-clause</i>.</p> <p>The <i>where-clause</i> must always follow the FROM/FM/IN or TO keyword and the library name (<i>lib-name</i>) or path name (<i>path-name</i>) if relevant; the sequence of the keywords and values within the clause can be specified in any order.</p>
DBID	<i>dbid</i>	N5	<p>The database ID (DBID) of a source or a target system file.</p> <p>The source system file contains the object to be processed. The target system file is the system file to which the object is to be copied or moved, or where the object is renamed if relevant.</p> <p>Valid DBIDs are 1 to 65535.</p> <p>If no DBID or FNR (file number) is specified, the following applies: The DBID and FNR of the system file where the current library resides are always used. For example: if you specify a library contained in the FUSER system file, the DBID and FNR of this file are used.</p> <p>See also the section File Security in Remote Environments.</p>
FNR	<i>fnr</i>	N5	<p>The file number (FNR) of a source or a target system file.</p> <p>The source system file contains the object to be processed. The target system file is the system file to which the object is to be copied or moved, or where the object is renamed if relevant.</p> <p>Valid FNRs are 1 to 65535.</p>

Keyword	Value	Natural Data Format/Length	Explanation
			<p>If no DBID or FNR is specified, the following applies: The DBID and FNR of the system file where the current library resides are always used. For example: if you specify a library contained in the FUSER system file, the DBID and FNR of this file are used.</p> <p>See also the section <i>File Security in Remote Environments</i>.</p>
DIC	<i>dbid</i> <i>fnr</i> <i>password</i> <i>cipher</i>	A80	<p>Specifies the environment of the FDIC source and/or target system file: database ID (<i>dbid</i>), file number (<i>fnr</i>), Adabas password (<i>password</i>) and Adabas cipher code (<i>cipher</i>).</p> <p>See also the section <i>File Security in Remote Environments</i>.</p>
SEC	<i>dbid</i> <i>fnr</i> <i>password</i> <i>cipher</i>	A80	<p>Specifies the environment of the FSEC source and/or target system file: database ID (<i>dbid</i>), file number (<i>fnr</i>), Adabas password (<i>password</i>) and Adabas cipher code (<i>cipher</i>).</p> <p>See also the section <i>File Security in Remote Environments</i>.</p>
WITH	<i>with-clause</i>	-	<p>An optional keyword that indicates the start of a <i>with-clause</i>.</p> <p>The keywords and values of the <i>with-clause</i> can be specified in any order, and the <i>with-clause</i> can be placed in any location within the command string, except in the first three positions.</p>
TYPE	<i>type</i>	A20	The type(s) of object to be processed as listed in TYPE Specification below.
FMDATE	<i>date</i>	A10	<p>The start date of a time period: All objects which were saved or cataloged on or after the specified date are processed.</p> <p>A date must be specified in a valid Natural date format. The default format is the international format <i>YYYY-MM-DD</i> (<i>YYYY</i> = year, <i>MM</i> = month, <i>DD</i> = day), for example, 2006-05-20.</p>
FMTIME	<i>time</i>	A5	<p>Only applies if FMDATE is specified.</p> <p>Specifies a start time: All objects which were saved or cataloged at or after the specified time (and date) are processed.</p> <p>A time must be specified in the format <i>HH:II</i> (<i>HH</i> = hours, <i>II</i> = minutes), for example, 11:33.</p>
USER	<i>user-id</i>	A8	<p>A user ID: All objects that were saved or cataloged by the specified user are processed.</p>

Keyword	Value	Natural Data Format/Length	Explanation
XREF	N or Y	A1	<p>Only applies to programming objects and if Predict is installed.</p> <p>Indicates whether XRef data stored in Predict system files is to be processed.</p> <p>You can specify one of the following values:</p> <p>N XRef data is not processed, except when using the DELETE command. If a cataloged object is deleted, SYSMAIN always deletes any existing XRef data for this object.</p> <p>Y All XRef data is processed.</p> <p>See also the section XRef Considerations.</p>
REPLACE	-	-	<p>Activates the replace option used in a <i>with-clause</i>.</p> <p>An object with the same name in the target environment is replaced by the object to be processed.</p> <p>Note: If an object is replaced it is also deleted from the Natural buffer pool; any existing cross-reference records are also deleted.</p>
RCOP	-	-	Specifies that a copy of the object being renamed is to be made.
NOPROMPT	-	-	<p>Not applicable in batch mode.</p> <p>Disables (NOPROMPT) the SYSMAIN prompts.</p> <p>With NOPROMPT, no confirmation screen is displayed.</p> <p>For example, before any deletion, SYSMAIN prompts you for confirmation.</p>
HELP	-	-	This keyword is provided for compatibility reasons only.
STRUCT or SM			<p>Only applies to the IMPORT command.</p> <p>Indicates structured mode described in <i>Natural Programming Modes</i> in the <i>Programming Guide</i>.</p>
REPORT			<p>Only applies to the IMPORT command.</p> <p>Indicates reporting mode described in <i>Natural Programming Modes</i> in the <i>Programming Guide</i>.</p>
.	-	-	A period (.) indicates the end of a command. If this character is detected anywhere within a command string, all subsequent data is ignored.

TYPE Specification

The following table lists all valid object-type codes for programming objects that can be used with the TYPE keyword:

Code	Object Type
P	Program
N	Subprogram
S	Subroutine
M	Map
H	Helproutine
3	Dialog
5	Processor
A	Parameter data area
G	Global data area
L	Local data area
C	Copycode
T	Text
4	Class
7	Function
V	View (DDM)
8	Adapter
*	All programming object types.

Specifying a Range of Names

All SYSMAIN functions provide the option to specify either a name or a range of names for the libraries or the objects to be selected.

The valid asterisk (*) notations for name ranges are listed below where *value* denotes any combination of one or more characters:

Input	Objects or Libraries Selected
*	All objects or libraries.
<i>value</i> *	All objects or libraries with names that start with <i>value</i> . Example: AB* Selected: AB, AB1, ABC, ABEZ Not selected: AA1, ACB

Input	Objects or Libraries Selected
<i>value*value*</i>	All objects or libraries that match <i>value</i> combined with one or two asterisks (*) in any order. Example: A*C* Selected: ABCZ, AXXCBBBZ, ANCZ Not selected: ABDEZ, ACBBBZA

68 XRef Considerations

- Processing of XRef Data 428
- XRef Processing Errors 429

All cross-reference (XRef) data stored in the Predict system file for a cataloged programming object can be processed with SYSMAIN.

The Predict system file is determined by the value assigned to the profile parameter `FDIC` (see *FDIC - Predict System File* in the *Parameter Reference* documentation) in the parameter file or at the start of the Natural Studio session.

You can override the current `FDIC` settings for the duration of the current `SYSMAIN` function by using the **FSEC/FDIC** button and changing the entries in the **FDIC** group boxes (see *File Security in Remote Environments*) or specifying the `DIC` keyword in a `SYSMAIN` command (see also *where-clause* in *Using SYSMAIN with Subprogram*).

The **XREF** check box in the **Code** group box or the `XREF` keyword in a `SYSMAIN` command indicates whether `SYSMAIN` should process XRef data.

If Predict has not been installed, you can clear the **XREF** check box (or set `XREF` to `N`) to not perform a validation of Predict files. This is the default setting.

Processing of XRef Data

No XRef data is processed if the **XREF** check box is cleared or if the `XREF` keyword is set to `N` when using a `SYSMAIN` command.

XRef data is processed if the **XREF** check box is selected or if `XREF` is set to `Y`.

Regardless of what setting you choose, XRef data is always deleted when a programming object is deleted.

If XRef data is to be processed, the following actions are applied during `SYSMAIN` processing:

- `SYSMAIN` checks whether XRef data exists in the Predict source system file for the specified programming object.
- If a programming object is to be deleted from the target environment, XRef data is deleted from the Predict target system file.
- If a programming object is copied to a new environment, the XRef data of the programming object is copied from the Predict source system file to the Predict target system file. The library name is changed accordingly and, in the case of the rename function, the object name is also changed.
- If the move function is executed, the XRef data of the programming object is deleted from the Predict source system file.

XRef Processing Errors

If any of the following inconsistencies occur during SYSMAIN processing of XRef data, all processing for the object or function is terminated and an error message is displayed:

- The value of the XREF option in Natural Security is set to Y and you cleared the **XREF** check box or set XREF to N respectively.
- The **XREF** check box is selected (or XREF set to Y) and the FDIC file(s) being used are not valid Predict files.

69 Security Considerations for Administrators

- File Security in Remote Environments 432
- Natural Security 434

This section describes the security aspects of the SYSMAIN utility.

File Security in Remote Environments

In a remote environment located on a mainframe, a UNIX or an OpenVMS platform, file security (that is, passwords and cipher codes) relates to the security that has been defined for a system file in an Adabas environment. If file security has been defined for a system file, you need to specify a password and a cipher code for the source and/or target system file required before you perform a SYSMAIN function. Otherwise, Adabas will issue an appropriate error message. You do not have to provide security information for the default system files assigned to you at the start of the SYSMAIN utility.

Security information for FSEC and/or FDIC system files can only be specified if Natural Security and/or Predict respectively are installed.

In a remote mainframe environment, the security information of the system files refers to the corresponding profile parameters FNAT, FUSER, FDIC and FSEC described in the *Parameter Reference* documentation.

In a remote UNIX or OpenVMS environment, the security information of the system files refers to the corresponding profile parameters FDIC and FSEC described in the *Parameter Reference* documentation.

The following system files and objects or data contained in the files can be affected by security protection:

- FNAT or FUSER with programming objects and FDIC with DDMs on a mainframe platform;
- FDIC with XRef data (UNIX, OpenVMS and mainframe);
- FSEC with Natural Security profile (UNIX, OpenVMS and mainframe).

▶ To specify security information for FNAT or FUSER, or FDIC for DDMs on mainframes

- 1 In the **Object Maintenance** dialog box of a SYSMAIN utility function, change the entry in the **DBID** or **FNR** box in the **Source** and/or **Target** group boxes.

The **Password** and **Cipher** boxes appear below **DBID** and **FNR**.

- 2 Enter the appropriate security information:

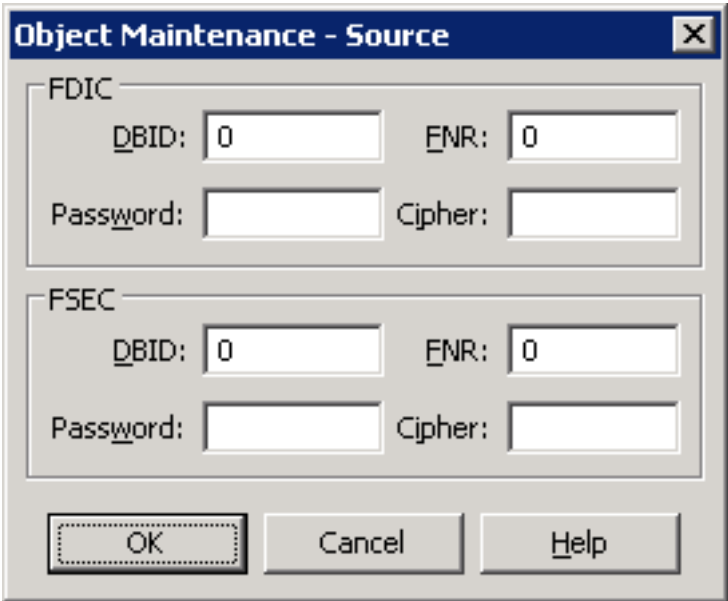
In the **Password** box, enter the 8-character Adabas password for the FNAT or FUSER source and/or target system files.

In the **Cipher** box, enter the 8-character Adabas cipher code for the FNAT or FUSER source and/or target system files.

▶ To specify security information for FDIC (XRef data) or FSEC

- 1 In the **Object Maintenance** dialog box of a SYSMAIN utility function, choose the **FDIC/FSEC** button in the **Source** and/or **Target** group boxes.

An **Object Maintenance - Source** or **Object Maintenance - Target** dialog box similar to the example below appears:



- 2 In the **FDIC** and/or **FSEC** group boxes, enter the appropriate security information for the FDIC system file (if Predict is installed) and/or the FSEC system file (if Natural Security is installed):

DBID	The database ID (DBID) of the source or the target database where the FDIC or FSEC system file is stored. Valid DBIDs are 1 to 65535. The default value is 0 (zero) for the current FDIC or FSEC system file.
FNR	The file number (FNR) of the source or the target database where the FDIC or FSEC system file is stored. Valid FNRs are 1 to 65535. The default value is 0 (zero) for the current FDIC or FSEC system file.
Password	The 8-character Adabas password for the FDIC or FSEC source and/or target system files.
Cipher	The 8-character Adabas cipher code for the FDIC or FSEC source and/or target system files.

The file security specifications in the **Object Maintenance** dialog boxes are retained for the duration of the current SYSMAIN function.

► **To specify security information for system files using commands**

■ For FSEC:

Use the **SEC** keyword of the *where-clause* described in *Using SYSMAIN with Subprogram*.

Or:

For FDIC and XRef data:

Use the **DIC** keyword of the *where-clause* described in *Using SYSMAIN with Subprogram*.

Or:

For FNAT or FUSER:

Use the **DBID** and **FNR** keywords of the *where-clause* described in *Using SYSMAIN with Subprogram*.

Natural Security

Two aspects must be considered when using the SYSMAIN utility within a Natural Security environment:

- [Defining the Natural Security Environment](#)
- [Restricting Use of SYSMAIN under Natural Security](#)

Defining the Natural Security Environment

The source and target libraries can be within one Natural Security environment or within two different Natural Security environments. These environments must be defined to the SYSMAIN utility.

The definition of the Natural Security environment(s) to be used can be specified in the **FSEC** group boxes of the **Object Maintenance - Source** and **Object Maintenance - Target** dialog boxes.

By default, SYMAIN uses the current FSEC settings as specified with the **FSEC** profile parameter in the parameter file or at the start of the Natural Studio session. You can override these settings by changing the entries in the **FSEC** group boxes. The new settings remain in effect for the duration of the current SYSMAIN function. When you execute SYSMAIN with a subprogram using commands (see *Using SYSMAIN with Subprogram*), the **SEC** keyword should be used to specify the file security and assignments of the request.

Once the source and target environments have been determined, SYSMAIN verifies both the source libraries and the target libraries with Natural Security. The source and/or target database and file

must correspond to the database ID (DBID) and file number (FNR) specified in the library security profile; if these values are not specified, default values are taken from the security profile.

Restricting Use of SYSMAIN under Natural Security

The use of the SYSMAIN utility itself can be restricted, or the use of the source and target libraries to be handled with the SYSMAIN utility can be restricted. The use of SYSMAIN utility functions when invoked with the MAINUSER subprogram can be controlled separately. See *Protecting Utilities* in the *Natural Security* documentation for details.

XIV

SYSNCP Utility

70

SYSNCP Utility

▪ Prerequisites for Windows	440
▪ Introducing the SYSNCP Utility	440
▪ Invoking SYSNCP	448
▪ Processor Selection	449
▪ Header Records	450
▪ Keyword Maintenance	459
▪ Function Maintenance	464
▪ Runtime Actions	469
▪ Processor Cataloging	475
▪ Administrator Services	476
▪ Session Profile	483

The utility SYSNCP is used to define command-driven navigation systems for Natural applications.

The Natural Command Processor (NCP) consists of two components: maintenance and runtime. The utility SYSNCP is the maintenance part which comprises all facilities used to define and control navigation within an application. The PROCESS COMMAND statement (see the *Statements* documentation) is the runtime part used to invoke Natural programs.

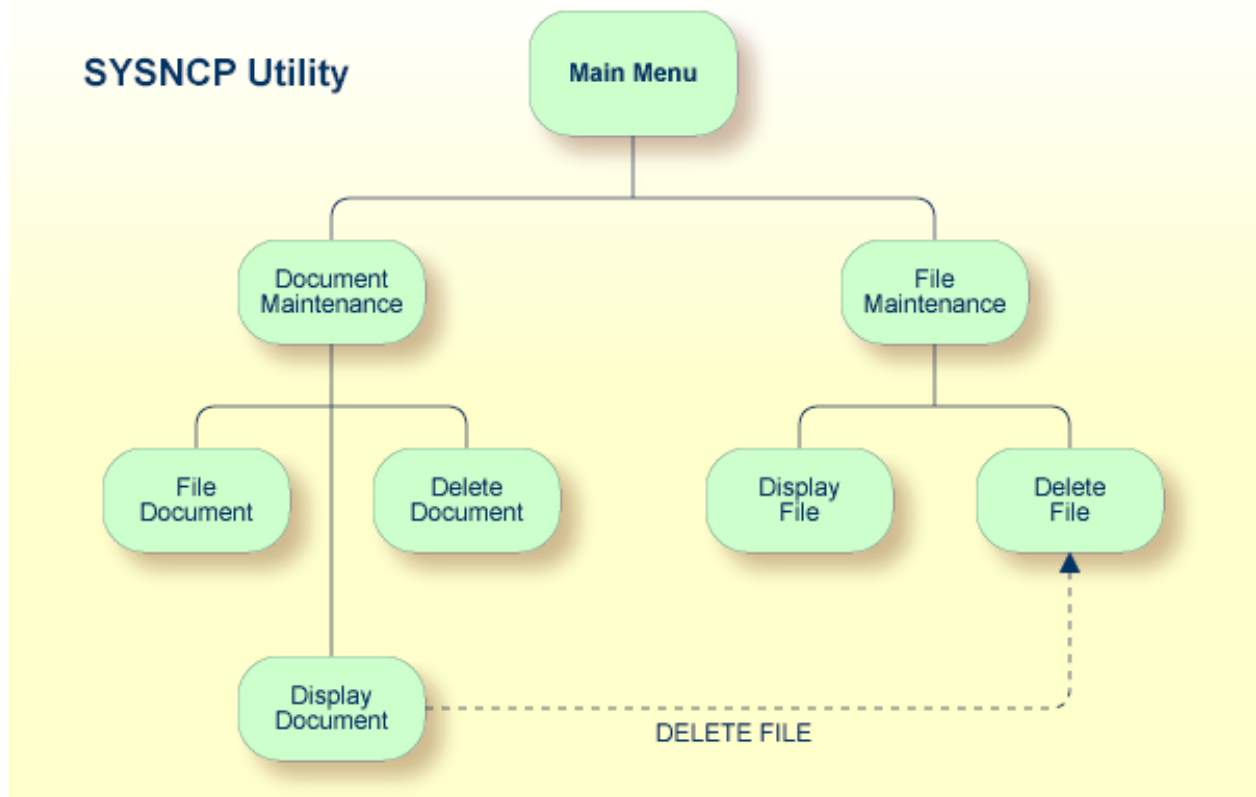
Prerequisites for Windows

This section lists the prerequisites required for installing the command processor under Windows:

- Adabas for Windows.
- Logical file (LFILE) 190 (NCP Command Proc); this file is set as default in the Natural parameter file, do not modify it.
- FDT "SYSTEM-NCP"; see the README file in the folder "demodb".
- Depending on the size of the command processor, the values of the Adabas parameters LP and NH may have to be adjusted. If these values are too small, you receive error NAT3145 (see the Adabas documentation). With Adabas for Windows Version 5.1, the parameters LP and NH have become obsolete.

Introducing the SYSNCP Utility

Applications which enable users to move from one activity to another activity by using direct commands far exceed in usability the ones which force the user to navigate through menu hierarchies to a desired activity.



The figure above illustrates the advantage of using direct commands. In an application in which menu hierarchies form the basis for navigation, a user wishing to advance from the Display Document facility to the Delete File facility would have to return to the Main Menu via the document branch and then enter the file branch. This is clearly less efficient than accessing the Delete File facility directly from the Display Document facility.

Below is information on:

- [Object-Oriented Data Processing](#)
- [Features of the Command Processor](#)
- [Components of the Command Processor](#)
- [What is a Command?](#)

- [Creating a Command Processor](#)

Object-Oriented Data Processing

The Natural command processor is used to define and control navigation within an application. It could be used, for example, to define a command `DISPLAY DOCUMENT` to provide direct access to the Display Document facility. When a user enters this command string in the Command line of a screen (for which this command is allowed), the Natural command processor processes the input and executes the action(s) assigned to the command.

In contrast to menu-driven applications, the command-driven applications implemented with the Natural command processor take a major step toward object-oriented data processing. This approach has the following advantages:

- The design of an application need not depend on the way in which a certain result can be reached, but only on the desired result itself. Thus, the design of an application is no longer influenced by the process flow within its components.
- The processing units of an application become independent of one another, making application maintenance easier, faster and much more efficient.
- Applications can be easily expanded by adding independent processing units. The resulting applications are, therefore, not only easy to use from an end-user's view, but also easier to create from a programmer's view.

The Natural command processor has the following additional benefits:

- **Less Coding**

Instead of having to repeatedly program lengthy and identically structured statement blocks to handle the processing of commands, you only have to specify a `PROCESS COMMAND` statement that invokes the command processor; the actual command handling need no longer be specified in the source code. This considerably reduces the amount of coding required.

- **More Efficient Command Handling**

As the command handling is defined in a standardized way and in one central place, the work involved in creating and maintaining the command-processing part of an application can be done much faster and much more efficiently.

- **Improved Performance**

The Natural command processor has been designed with particular regard to performance aspects: it enables Natural to process commands as fast as possible and thus contributes to improving the performance of your Natural applications.

Features of the Command Processor

The Natural command processor provides numerous features for efficient and user-friendly command handling:

■ Flexible Handling of Commands

You can define aliases (that is, synonyms for keywords), and abbreviations for frequently used commands.

■ Automatic Check for Uniqueness of Abbreviated Keywords

The command processor automatically compares every keyword you specify in SYSNCP with all other keywords and determines the minimum number of characters in each keyword required to uniquely identify the keyword. This means that, when entering commands in an application, users can shorten each keyword to the minimum length required by the command processor to distinguish it from other keywords.

■ Local and Global Validity of Commands

You can specify in SYSNCP whether the action to be performed in response to a specific command is to be the same under all conditions or situation-dependent. For example, you can make the action dependent on which program was previously issued. In addition, you can define a command to be valid under one condition but invalid under another.

■ Error Handling for Invalid Commands

You can attach your own error-handling routines to commands or have error input handled by Natural.

■ Functional Security

With Natural Security, library-specific and user-specific conditions of use can be defined for the tables generated with SYSNCP. Thus, for your Natural applications you can allow or disallow specific functions or keywords for a specific user. This is known as functional security. See also the section *Functional Security* in the *Natural Security* documentation.

■ Help Text

In SYSNCP, you can attach help text to a keyword or a command. Then, by specifying a PROCESS COMMAND ACTION TEXT statement, you can return command-specific help text to the program.

■ Online Testing of Command Processing

If the execution of a command does not produce the intended result, you can find out why the command was not processed correctly by using the PROCESS COMMAND statement (see the *Statements* documentation) and the EXAM* sample test programs (source form) provided in the Library SYSNCP. The endings of the EXAM-* program names appear as abbreviations at the top border line of the relevant action windows (for example, EXAM-C appears as C).

▶ To test a command processor at runtime

- 1 Enter the direct command EXAM to list all test programs. The **Demonstrate PROCESS COMMAND Statement** window is displayed.

- 2 Enter Function Code **O** to open a processor.
- 3 Enter the name of the processor.
- 4 Choose any of the Functions Codes listed (for example, C for CHECK) to apply command actions.
- 5 Enter Function Code **Q** to close the processor.

Components of the Command Processor

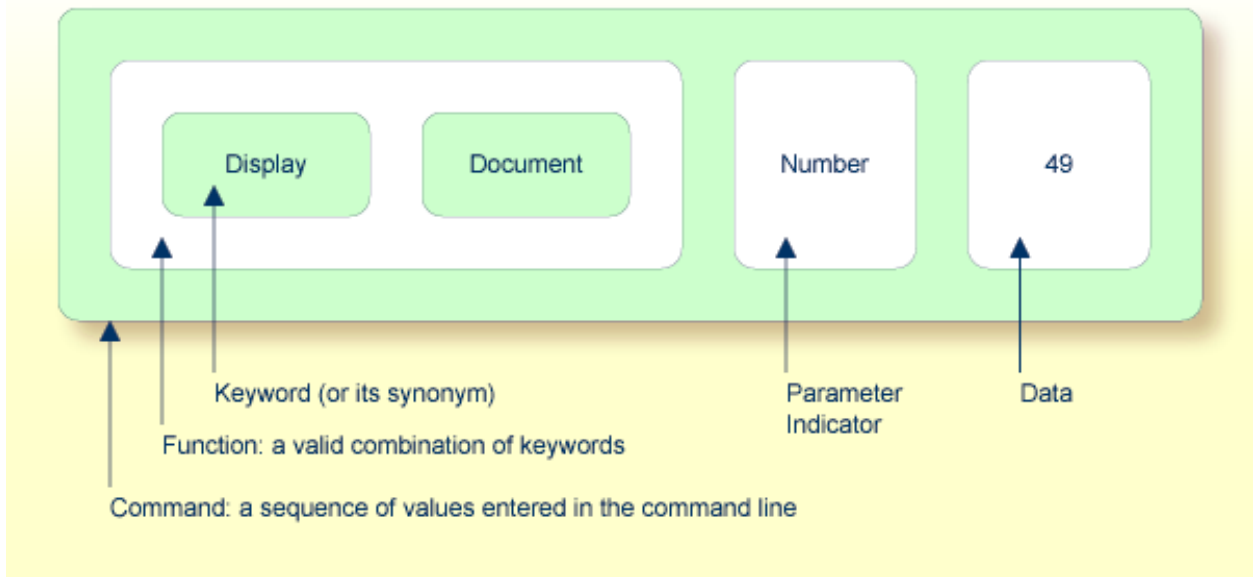
The Natural command processor consists of two parts: a development part and a runtime part:

- The development part is the utility SYSNCP, which is described in this section. With the utility SYSNCP you define commands (as described below) and the actions to be performed in response to the execution of these commands. From your definitions, SYSNCP generates decision tables which determine what happens when a user enters a command. These tables are contained in a Natural member of type Processor.
- The runtime part is the statement PROCESS COMMAND, which is described in the *Statements* documentation. This statement is used to invoke the command processor within a Natural program. In the statement, you specify the name of the processor to be used to handle the command input by a user at this point.

What is a Command?

A command is any sequence of values entered in the Command line which is recognized and processed by an application. Commands can contain up to three elements:

- **Function:**
One or more valid keywords. For example, MENU or DISPLAY DOCUMENT.
- **Parameter Indicator:**
Optional. A keyword which introduces command data.
- **Command Data:**
Information to be sent to a function. Command data can be alphanumeric or numeric, for example, the name or the number of the file to be displayed.



Commands are always executed from a situation within an application; the position where this situation is reached is referred to as a location. Commands take the user from one location to another location; thus, each command can be viewed as a vector:

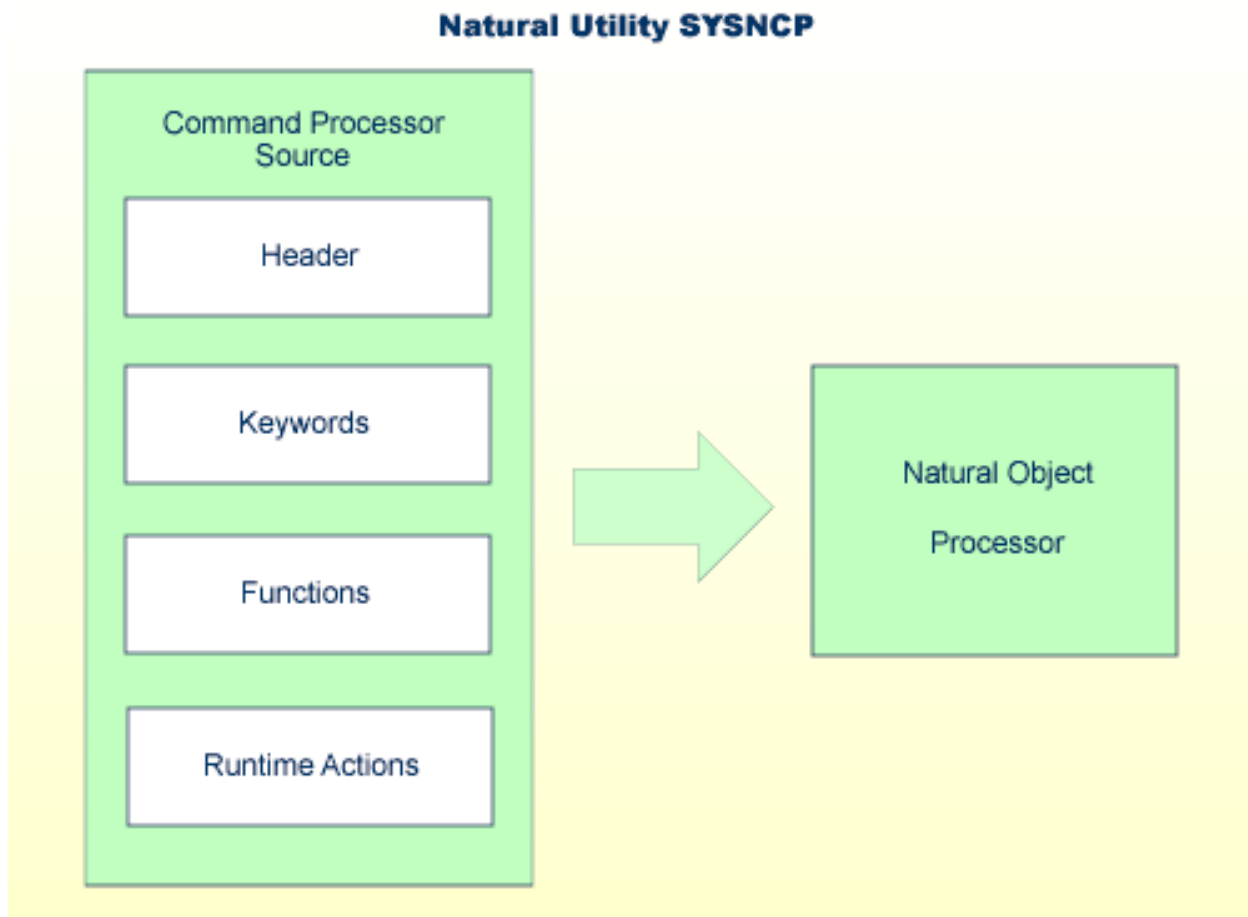


The location from which a certain command can be issued can be restricted on a system-wide or user-specific basis. On a system-wide basis, for example, the functions specified within commands can be local or global. A global function can be issued from *any* location while a local function can only be issued from specified locations. Restrictions can be placed on keywords and functions, however, if Natural Security is active in your environment.

Creating a Command Processor

The utility SYSNCP is used to create and maintain command processors. A command processor contains decision tables which determine what happens when a user enters a valid command.

The creation of a command processor is a cumulative operation involving several steps, from header definition, which establishes general defaults for the processor, to keyword definition, function definition and the linking of actions to functions. Special editors are provided by SYSNCP for the purpose of specifying keywords, functions and actions.



The end product of command processor development is a complex command processor source, which, when cataloged, generates a Natural object of type Processor. Whenever this object is referenced by the Natural statement `PROCESS COMMAND`, the runtime system of the Natural command processor is triggered.

The following is a summary of the steps necessary to create a command processor.

▶ **To create a command processor**

1 Verify/Modify the Session Profile.

SYSNCP itself uses a Session Profile which contains various parameters which control how SYSNCP is to perform certain actions and how information is to be displayed. Desired modifications can be made and the resulting profile can be saved with a given user ID. See the section [Session Profile](#).

2 Initialize the Command Processor.

The name of the command processor and the library into which it is to be stored are specified.

3 Define Global Settings (Header).

Various global settings for the command processor are defined. For example, descriptive text for keywords during editing, minimum and maximum length for keywords, in which sequence keywords are to be processed at runtime, runtime error-handling, and whether PF keys can be used at runtime to invoke functions. See the section [Header Records](#).

4 Define Keywords.

Each keyword which is to be processed by the command processor is defined together with an indication as to whether the keyword is to be entered as the first, second or third entry of a command. Keyword synonyms can also be defined as well as parameter indicators. User text can be defined for each keyword. This text can subsequently be read at runtime using the PROCESS COMMAND ACTION TEXT statement. See the section [Keyword Maintenance](#).

5 Define Functions.

Functions are defined by validating keyword combinations. A function can be defined as local (can only be invoked from a specific location within an application) and/or global (can be invoked from anywhere within an application). See the section [Function Maintenance](#).

6 Define Runtime Actions.

The actions to be taken by the command processor when a command is issued at runtime are specified. Example actions are: fetch a Natural program, place a command at the top of the Natural stack, place data at the top of the Natural stack, change contents of the Command line. See the section [Runtime Actions](#).

7 Catalog Command Processor.

The resulting source is cataloged as a Natural object (type Processor) in the designated Natural library. The command processor can now be invoked by a Natural program using the PROCESS COMMAND statement. See the section [Processor Cataloging](#).

Invoking SYSNCP

▶ To invoke the SYSNCP utility

- Enter the system command SYSNCP.

The Processor Source Maintenance menu is displayed:

```

18:22:53          ***** NATURAL SYSNCP UTILITY *****          2000-05-22
User SAG          - Processor Source Maintenance -

Code  Function

S     Select Processor
N     Create New Processor
H     Modify Header
K     Define Keywords
F     Define Functions
R     Define Runtime Actions
C     Catalog Processor
A     Administrator Services
?     Help
.     Exit

Code .. _      Name .. SAGTEST_  Library .. SYSNCP__

Logon to SYSNCP accepted.
Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Cmd  Exit Last List Flip                               Canc

```

From this menu, you can invoke all functions necessary to create and maintain a command processor. To invoke a function, enter the code letter in the Code field.



Note: When you invoke the SYSNCP utility or restart SYSNCP, the user exit NCP-USR1 is invoked for dynamic customization purposes: see the program NCP-USR1 delivered in the Natural system library SYSNCP.

Help

For help on individual input fields (and also on some output fields) in SYSNCP, place the cursor on the field and press PF1.

Processor Selection

The Select Processor function results in a list of all existing command processor sources with related information. If Natural Security is installed, only those sources are listed which can be cataloged to a library to which you are allowed to log on. These restrictions do not apply to those users who have administrator status.

▶ To invoke the Select Processor function

- 1 In the Processor Source Maintenance menu, enter Function Code S.
- 2 Press ENTER.

The following information is provided for each processor:

Name	The name of the command processor.
Library	The name of the Natural library for which a processor is created. When the processor is cataloged, it is stored in this library.
User ID	The ID of the user who created the processor.
Date	The date the processor was created.
Status	The stage of development of the processor. For possible status values, see Current Status in the section <i>Header Records</i> .
Cat	Indicates if the processor has been cataloged.



Note: With the user exit NCP-SELX (delivered in the Natural system library SYSNCP), you can limit the display to certain processors.

- 3 In the **Ac** field, enter any character to select a processor.

The Processor Source Maintenance menu is displayed, where the name of the selected processor is automatically placed in the Name field.

If you enter a question mark (?) in the Ac field, a window is displayed, listing other possible options.

The name and library name of a command processor can be one to eight characters long. It can consist of upper-case alphabetical characters (A - Z), numeric characters (0 - 9) and the special characters: "-", "/", "\$", "&", "#", "+" and "_".

Header Records

The header maintenance facility defines various global settings for a command processor. These definitions are collectively referred to as a header. Seven header maintenance screens are provided for creating and modifying headers. Header settings for a command processor can be updated at any stage of development (see the following section). After the settings have been modified, the status of a command processor is always set to Header (see also *Current Status*).

Below is information on:

- [Create New Processor](#)
- [Modify Header - General Explanations](#)
- [Keyword Runtime Options - Header 1](#)
- [Keyword Editor Options - Header 2](#)
- [Miscellaneous Options - Header 3](#)
- [Command Data Handling - Header 4](#)
- [Runtime Error Handling - Header 5](#)
- [Statistics - Header 6](#)
- [Status - Header 7](#)

Create New Processor

▶ **To create a new command processor**

- 1 In the Processor Source Maintenance menu, enter Function Code **N** (Create New Processor), the name of the command processor to be created, and the name of the Natural library in which the command processor is to be later cataloged.
- 2 Press ENTER.

The first header maintenance screen is displayed.

The first header maintenance screen and the following ones are filled with default values that can be edited.

Modify Header - General Explanations

The Modify Header function is used to maintain an existing header; that is, to modify the various header settings for a given command processor.

▶ To modify an existing header

- 1 In the Processor Source Maintenance menu, enter Function Code **H** (Modify Header), the name of the corresponding command processor, and the name of the library into which this command processor has been cataloged.

- 2 Press ENTER.

The first header maintenance screen is displayed.

- 3 Modify any input field in the header maintenance screens described below.

- 4 Press ENTER to confirm modifications.

Seven different screens are available for the definition and maintenance of a processor header (for the definition of a header, see the previous section).

▶ To navigate between the header maintenance screens

- Use PF8 (forward) or PF7 (backward).

Each of the screens contains the following information:

Name	The name of the command processor.
Library	The name of the library into which the resulting command processor object is to be placed after being cataloged.
DBID, FNR	The database ID and file in which the specified library is located.
Created by	The user ID of the Natural user who initialized this command processor.
Date	The date the command processor was initially created.
Current Status	The command processor status: <ul style="list-style-type: none"> Init The command processor has been initialized. Header The header for the command processor has been created/modified. Keysave Keywords have been defined and saved. Keystow Keywords have been checked and stowed. Function Keyword combinations have been defined. Action Runtime actions have been defined. Object An object form of the command processor has been created. Frozen The command processor has been frozen.

	Copied	The command processor has been copied.
	Error	An error has been detected.

Keyword Runtime Options - Header 1

When you select the Modify Header function (as described above), the **Processor Header Maintenance 1** screen is displayed:

```

16:40:19          ***** NATURAL SYSNCP UTILITY *****          2000-05-04
User SAG          - Processor Header Maintenance 1 -

Modify Processor          Name SAGTEST  Library SYSNCP  DBID 10   FNR 32
Created by SAG          Date 2000-04-29          Current Status Init ←

Keyword Runtime Options:
-----
First Entry used as ..... Action_____
Second Entry used as ..... Object_____
Third Entry used as ..... Addition_____

Minimum Length ..... _1
Maximum Length ..... 16
Dynamic Length Adjustment .. -

Keyword Sequence ..... 123_____
Alternative Sequence ..... _____
Local/Global Sequence ..... LG_____

Processor Header with name SAGTEST for library SYSNCP has been added.
Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Cmd  Exit Last List Flip -      +      Canc
    
```

Various attributes which are to apply for the keywords defined for the command processor are entered on this screen.

Field	Explanation
First Entry used as	<p>A descriptive text which is to be associated with all keywords which are entered as the first entry (entry type 1) when defining a keyword sequence.</p> <p>For example, if the first keyword of a keyword sequence is to represent the action to be performed (DISPLAY, DELETE, etc.), the descriptive text "Action" could be entered in this field.</p> <p>The first four characters of the text entered in this field appear under the column heading Use in the Keyword Editor as described in the section <i>Keyword Maintenance</i>.</p>
Second Entry used as	<p>A descriptive text which is to be associated with all keywords which are entered as the second entry (entry type 2) when defining a keyword sequence.</p> <p>If, for example, the second keyword of a keyword sequence is to represent the object to be used (DOCUMENT, FILE, etc.), the descriptive text "Object" could be entered in this field.</p> <p>The first four characters of the text entered in this field appear under the column heading Use in the Keyword Editor as described in the section <i>Keyword Maintenance</i>.</p>
Third Entry used as	<p>A descriptive text (TITLE, PARAGRAPH, etc.) which is to be associated with all keywords which are entered as the third entry (entry type 3) when defining a keyword sequence.</p> <p>The first four characters of the text entered in this field appear under the column heading Use in the Keyword Editor as described in the section <i>Keyword Maintenance</i>.</p>
Minimum Length	The minimum length permitted when defining a keyword. Valid values are 1 - 16 characters. The default is one character.
Maximum Length	The maximum length permitted when defining a keyword. Valid values are 1 - 16 characters. The default is 16 characters.
Dynamic Length Adjustment	<p>The following values are permitted:</p> <ul style="list-style-type: none"> + At runtime, each keyword must be entered in its entirety. - At runtime, each keyword can be abbreviated provided that it retains uniqueness with respect to other keywords. <p>S The number of characters which must be entered for a given keyword is to be specified during keyword definition in the ML field of the Keyword Editor as described in the section <i>Keyword Maintenance</i>.</p>
Keyword Sequence	The sequence in which keyword entries are to be processed at runtime. Possible values are 1, 2, 3 and P (for parameter indicator); the default sequence is 12, which means first the first keyword entry and then the second keyword entry. See also the field E as described in the section <i>Keyword Maintenance</i> .
Alternative Sequence	An alternative sequence in which keywords are to be processed at runtime in the event that the default sequence (specified above) results in an error during runtime.
Local/Global Sequence	This option specifies the order of command validation to be performed at runtime. Possible values are:

Field	Explanation
	<p>L Command is to be validated as a local command.</p> <p>G Command is to be validated as a global command.</p> <p>The default validation sequence is LG, which means that the command is to be validated first as a local command and then (if necessary) as a global one.</p>

Keyword Editor Options - Header 2

Further keyword attributes can be entered on the **Processor Header Maintenance 2** screen:

Field	Explanation
Header 1 for User Text	These two fields are used to enter a descriptive text which appears in the Keyword Editor above the column reserved for user text. This text is also output during runtime when the TEXT option is specified with the PROCESS COMMAND statement as described in the <i>Statements</i> documentation.
Header 2 for User Text	
Prefix Character 1	<p>This field and the next three are used to attach a hexadecimal prefix to keywords. This enables the processing of internal keywords which cannot be represented by a normal keyboard. When the command processor is cataloged, all prefix characters in keywords are replaced by the hexadecimal values specified.</p> <p>If a non-blank character is entered in one of the Prefix Character fields, the specified character is replaced by the hexadecimal value specified in the Hexadecimal Replacement field.</p>
Hex. Replacement 1	The value specified in this field replaces the character specified in the field Prefix Character and is used as a prefix for a keyword at runtime.
Prefix Character 2	See above Prefix Character 1.
Hex. Replacement 2	See above Hex. Replacement 1.
Keywords in Upper Case	<p>This option specifies whether keywords are to be translated to upper case in the Keyword Editor and the application:</p> <p>Y Keywords entered in the Keyword Editor are automatically converted to upper case. In the application, end-users can enter the keywords in upper or lower case.</p> <p>N Keywords entered in the Keyword Editor are not converted to upper case. In the application, end-users must enter the keywords <i>exactly</i> as they appear in the Keyword Editor.</p>
Unique Keywords	<p>This option specifies whether keywords within the processor must be unique.</p> <p>Y Each keyword defined must be unique within this processor, regardless of its type.</p> <p>N Each keyword defined for a given keyword type (1, 2, 3 or P) must be unique.</p>

Miscellaneous Options - Header 3

Miscellaneous options can be entered on the **Processor Header Maintenance 3** screen:

Field	Explanation
Invoke Action Editor	<p>This option specifies whether the Runtime Action Editor is to be activated from the Function Editor (see the sections <i>Runtime Action Editor</i> and <i>Define Functions</i>).</p> <p>Y The Runtime Action Editor is invoked whenever a valid keyword combination is defined in the Function Editor.</p> <p>N The Runtime Action Editor is suppressed in the Function Editor.</p> <p>Note: If you use the user exit NCP-REDM (delivered in the Natural system library SYSNCP), you should set this option to Y; otherwise, invalid runtime action values cannot be detected in time and can lead to runtime errors.</p>
Catalog User Texts	<p>This option specifies whether user texts are to be cataloged with the command processor.</p> <p>Y Text portions of the edit line (Keyword Editor; see the section <i>Define Keywords</i>) and the user text portion of the action line (Runtime Action Editor) are bound to the associated keyword or function when the command processor is cataloged. This text can then be read at runtime using the TEXT option of the PROCESS COMMAND statement.</p> <p>N Texts are not cataloged with the command processor and cannot be read at runtime.</p>
Security Prefetch	<p>This option specifies whether security checking is to be performed when the command processor is initially invoked during runtime or at each command evaluation.</p> <p>Y If Natural Security is installed, security checking is performed for all keywords when the processor is invoked.</p> <p>N If Natural Security is installed, security checking is performed with the evaluation of each keyword.</p> <p>If option Y is selected, security checking is performed only once for all keywords when the command processor is invoked. Since the checking procedure takes time, evaluation of the first command is comparatively slow at runtime, while the evaluation of all remaining commands is comparatively fast. Conversely, if option N is selected, the evaluation time for each command is always the same because security is checked for each keyword individually before it is evaluated.</p>
Command Log Size	<p>Commands processed at runtime can be stored in a command log area by the command processor. Specify in the input field the number of KBs storage space allocated to command logging:</p> <p>0 No storage space is allocated to command logging. Command logging is inactive.</p> <p>1 1 KB of storage space is allocated to command logging. Command logging is active.</p>
Implicit Keyword Entry	<p>This option specifies whether a keyword of type 1 is to be retained as an implicit keyword for all subsequent commands.</p>

Field	Explanation
	<p>1 If a command is entered which only contains a keyword of type 2, the command processor assumes the most recently entered keyword of type 1 as implicit keyword.</p> <p>N Option is disabled.</p>
Command Delimiter	<p>This option specifies the character used to separate commands if more than one command is specified in the Command line. At runtime, only the first command will be executed.</p> <p>For example:</p> <p>DISPLAY CUSTOMER; MODIFY CUSTOMER; PRINT.</p>
PF-Key may be Command	<p>This option specifies whether commands can be allocated to PF keys: if the command processor receives at runtime a command line which contains all blanks, it checks if a PF key has been pressed by the user.</p> <p>Possible values are:</p> <p>A The identifier for this PF key (system variable *PF-NAME) is used as the command.</p> <p>K The content of the *PF-KEY system variable is used as the command.</p> <p>Y If *PF-NAME is empty, the content of the *PF-KEY system variable is used instead.</p> <p>N PF keys cannot be used as command, Natural error NAT6913 is issued with message "Command line not accepted".</p> <p>For more information on the system variables *PF-NAME and *PF-KEY see the <i>System Variables</i> documentation.</p>

Command Data Handling - Header 4

The attributes to be entered on the **Processor Header Maintenance 4** screen specify how command data are handled for a function; command data are optional.

Options are:

Field	Explanation
Data Delimiter	<p>Specifies the character to be used to precede data. Default data delimiter is "#".</p> <p>Example: ADD CUSTOMER #123</p>
Data Allowed	<p>Specifies if data input is allowed at runtime.</p> <p>N A runtime error occurs if data is found.</p> <p>D Data is dropped if present.</p> <p>S Data is placed at the top of the Natural stack. No verification is performed.</p>

Field	Explanation
	<p>Y Data is checked and keyword entries of type P (parameter indicator) are evaluated.</p> <p>Example of Y: DISPLAY CUSTOMER NAME=SMITH</p>
More than one Item Allowed	<p>Only applies if the option Data Allowed is set to Y. Specifies whether more than one data string is permitted.</p> <p>N A runtime error occurs if more than one data string is found.</p> <p>D All data after the first data string are dropped.</p> <p>Y More than one data string is permitted.</p> <p>Example: ADD ARTICLE #111 #222</p> <p>As long as uniqueness is guaranteed, the data delimiter can be omitted.</p> <p>Example: ADD ARTICLE 123</p>
Maximum Length of one Item	<p>Only applies if the option Data Allowed is set to Y.</p> <p>Specifies the maximum number of characters allowed for a data string. If the specified maximum is exceeded, a runtime error occurs. Valid range: 1 - 99.</p>
Item Must be Numeric	<p>Only applies if the option Data Allowed is set to Y. Specifies whether each data value must be an integer value.</p> <p>Y Data input must be a positive integer value. If not, a runtime error occurs.</p> <p>N Data can be of any type.</p>
Put to Top of Stack	<p>Only applies if the option Data Allowed is set to Y. Specifies where data is to be placed.</p> <p>Y Data is placed at the top of the Natural stack.</p> <p>1-9 Data is placed in the <i>n</i>th occurrence of the DDM field RESULT-FIELD. If the occurrence has already been filled as a result of a runtime action, it is overwritten.</p>
If Error, Drop all Data	<p>Only applies if the option Data Allowed is set to Y or N. Specifies the reaction to a data evaluation error:</p> <p>Y If an error occurs during evaluation of the data, data is discarded and processing continues.</p> <p>N If an error occurs during data evaluation, control is given to the error handler as described below.</p>

Runtime Error Handling - Header 5

The attributes to be entered on the **Processor Header Maintenance 5** screen specify how to handle runtime errors:

Field	Explanation
General Error Program	<p>The name of the program which is to receive control when an error is detected during runtime processing by the command processor. The Natural stack contains the following information when this program is invoked:</p> <p>Error Number (N4) Line Number (N4) Status (A1) Program Name (A8) Level (N2)</p> <p>If no error program and no specific error handling is specified (see below), the program with the name as contained in the Natural system variable *ERROR-TA is invoked; otherwise, a Natural system error message is issued.</p>
Keyword not found	Indicates whether an action has been specified that is to be performed if a keyword could not be found.
Keyword missing	Indicates whether an action has been specified that is to be performed if the keyword type is missing.
Keyword Sequence Error	Indicates whether an action has been specified that is to be performed in the case of a keyword sequence error.
Command not defined	Indicates whether an action has been specified that is to be performed in the case of an undefined command.
Data disallowed	Indicates whether an action has been specified that is to be performed in the case of disallowed data.
Data Format/Length Error	Indicates whether an action has been specified that is to be performed in the case of a format/length error.
General Security Error	Indicates whether an action has been specified that is to be performed if an error is detected during a general security check.
Keyword Security Error	Indicates whether an action has been specified that is to be performed if an error is detected during a keyword security check.
Command Security Error	Indicates whether an action has been specified that is to be performed if an error is detected during a command security check.

Statistics - Header 6

The **Processor Header Maintenance 6** screen contains only output fields which report statistical data about the keywords specified for a command processor.

The following statistical information is provided:

Field	Explanation
Entry n Keywords	The number of keywords of type n defined in the command processor (not including synonyms).
Entry n Keywords + Synonyms	The sum of keywords of type n and their assigned synonyms.
Highest IKN for Entry n	The largest Internal Keyword Number for the keyword of type n .
Possible Combinations	The number of possible combinations for keywords defined.
Cataloged Functions	The number of keyword combinations currently cataloged.

Status - Header 7

The **Processor Header Maintenance 7** screen contains only output fields which report the time and the date when parts of the command processor were executed or modified.

Keyword Maintenance

Keywords are the basic components for defining functions. Before it is possible to define keywords, the header maintenance records must be created (see the section [Header Records](#)).

- [Define Keywords](#)
- [Editor Commands](#)
- [Positioning Commands](#)
- [Line Commands](#)

Define Keywords

Keywords used in commands are created with the Define Keywords function and the Keyword Editor. The Keyword Editor is similar to existing Natural editors except that lines of the editor are broken up into separate fields. Most of the [editor commands](#) (see the relevant section) and the [line commands](#) (see the relevant section) which are used in the Natural program editor can also be used in the Keyword Editor.

▶ To invoke the Keyword Editor

- 1 In the Processor Source Maintenance menu, enter Function Code **K** (Define Keywords).
- 2 Press ENTER.

The Keyword Editor screen is displayed.

The Keyword Editor screen is shown below. Several keywords have already been defined to serve as examples for this section.

```


09:42:39                - SYSNCP Keyword Editor -                2000-05-04
Modify Keywords          Name SAGTEST   Library SYSNCP   DBID 10   FNR 32  ←

I Line E Use  Keyword          IKN   ML Comment          ←
-----
   1 1 Acti MENU           1004   1
   2 1 Acti DISPLAY       1002   2
   3 S Syno SHOW          1002   1
   4 1 Acti DELETE        1001   2
   5 S Syno PURGE         1001   1
   6 S Syno ERASE         1001   1
   7 1 Acti FILE          1003   4
   8 P Parm NAME          4002   2
   9 2 Obje FILE          2001   4
  10 P Parm NUMBER        4001   2
  11 2 Obje DOCUMENT      2003   2
  12 1 Acti INFORMATION   1005   1
  13
  14
----- All -----

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Cmd  Exit  Last  List  Flip  -1   +1   Top  Bot  Info  Canc

```

Enter in the Keyword Editor all the keywords which you want to have in your command language. These can be entered in any order desired, except synonyms, which must immediately follow the keywords they are related to. To each keyword you assign a type which specifies to which part of command syntax the keyword belongs. Rules of command syntax for a command processor are specified in the processor header; see *Keyword Runtime Options - Header 1* in the section *Header Records*. For example, you can specify whether a keyword is to be of type 1 (entered in first position in a command), type 2, type 3, a synonym for another keyword or a parameter indicator.

 **Note:** A command language requires a strict syntax because, to date, no computer is capable of understanding semantics. Word type is, therefore, the only practical way to communicate meaning in a command language.

In the example above, the keywords DELETE and DISPLAY are defined as keywords of type 1. As specified in the processor header, these keywords denote actions. The keyword DOCUMENT is defined as a keyword of type 2 and it denotes an object. The keyword FILE, however, is defined

as both type 1 and 2, and it can, therefore, denote an action or an object, depending on where it is positioned in the command. It is possible to compose the two keyword types to make commands, such as DELETE FILE and FILE DOCUMENT.

You can save the keywords you have entered by issuing the SAVE or STOW command from the Command line. In addition to saving the keyword definitions in source form, the STOW command performs a consistency check on them. Once a keyword is stowed successfully, it is given an internal keyword number (IKN) which is used at runtime to evaluate a command. Synonyms are always linked to a master keyword and always take the IKN of their master.

Each line in the Keyword Editor contains the following fields:

Field	Explanation
I	Output field. An information field which can contain the following values: E Indicates that a definition error has been detected. X Line is marked with X. Y Line is marked with Y. Z Line is marked with both X and Y. S Scan value found in this line.
Line	Output field. The line number of the editor.
E	Specifies the entry type for a keyword; that is, the position the keyword is to be entered in a command: first, second or third position, synonym or parameter indicator. For instance, in the Keyword Editor screen example above the keyword DELETE is of entry type 1 and DOCUMENT of type 2. Using these keywords, the command DELETE DOCUMENT can be defined. The field takes any of the following characters as input: 1 The keyword defined in this line is to be used as the first titem in a command sequence. 2 The keyword defined in this line is to be used as the second titem in a command sequence. 3 The keyword defined in this line is to be used as the third titem in a command sequence. S The keyword defined in this line is to be used as a synonym for the preceding keyword with titem type 1, 2, 3 or P. P The keyword defined in this line is to be used as a parameter indicator in a command sequence. * No keyword is to be defined in this line. Instead, the line is to be used solely as a comment line. ? This symbol is an output value which indicates an invalid keyword specification.
Use	Output field. The value displayed is determined by the value entered in the preceding field E:

Field	Explanation
	<p>1-3 The first four characters of the user text specified in the processor header for the first, second and third keyword entries respectively are displayed. See also <i>Keyword Editor Options - Header 2</i> in the section <i>Header Records</i>.</p> <p>S SYNO, the abbreviation for synonym, is displayed.</p> <p>P PARM, the abbreviation for parameter indicator, is displayed.</p>
Keyword	<p>Enter the keyword to be defined. Embedded blanks are not permitted. If you have specified in the processor header that keywords can only be upper case, then keywords are always translated to upper case, regardless of how they are entered. Otherwise, the case remains as entered.</p> <p>The maximum and minimum length of keywords depends on the settings specified in the header (default: 1 - 16 characters). Keywords must be unique unless specified otherwise in the header. Keyword prefixes can be used as described in <i>Keyword Editor Options - Header 2</i> in the section <i>Header Records</i>.</p>
IKN	<p>Output field. The Internal Keyword Number (IKN) is an identifier assigned to each valid keyword. IKNs are useful for testing and debugging. They are allocated only when a keyword is successfully stowed (see also the STOW command under <i>Editor Commands</i>). Each keyword is assigned a unique IKN, except synonyms, which take the IKN of their master term (see the <i>Keyword Editor screen</i> example above: DISPLAY and SHOW).</p>
ML	<p>Input and output field indicating the minimum length of a keyword. The field is an input field if S is specified in the Dynamic Length Adjustment field of the processor header as described in <i>Keyword Runtime Options - Header 1, Header Records</i>. In this case, you must specify the number of characters which must be entered for the keyword. For all other input, this field contains the minimum number of characters of a keyword a user must specify to avoid ambiguity with other keywords.</p> <p>For instance, in the <i>Keyword Editor screen</i> example above, keyword MENU requires only input of M while keyword DISPLAY requires input of DI to avoid ambiguity with keyword DELETE.</p>
Comment	<p>Enter free text for a keyword. There are no input restrictions. The user text is included in the cataloged command processor if the field Catalog User Texts is set to Y in the header definition as described in "Miscellaneous Options - Header 3", <i>Header Records</i>. It can be read at runtime using the TEXT option of the PROCESS COMMAND statement. The header text appearing at the top of this column is controlled by the header definition fields "Header for User Text 1" and "Header for User Text 2".</p>

Editor Commands

In the Command line of the Keyword Editor, you can enter the following commands:

Command	Function
ADD	Adds ten empty lines to the end of the editor.
CANCEL	Returns to Processor Maintenance Menu.
CHECK	Tests the keyword source for consistency.
EXIT	Returns to Processor Maintenance Menu.
HELP	Displays valid escape characters and other useful processor settings.
INFO	Displays information on the keyword on which your cursor is positioned.
LET	Undoes all modifications made to the current screen since the last time ENTER was pressed.
POINT	Positions the line in which a line command .N is entered to the top of the current screen.
RECOVER	Returns keyword source that existed before last SAVE/STOW.
RESET	Deletes the current X and Y line markers.
SAVE	Keyword source is saved.
SCAN	Scans for the next occurrence of the scan value.
STOW	Keyword source is stowed and Internal Keyword Numbers (IKNs) are generated for valid keywords.

Positioning Commands

Editor positioning commands are the same as the ones provided for the Natural program editor. For more information, see the description of the program editor in the *Editors* documentation.

The last line of the editor contains an output field which informs you of where your display is located in the editor. The following output values are displayed:

Top	Editor is currently positioned at the top of the keyword source.
Mid	Editor is currently positioned at the center of the keyword source.
Bot	Editor is currently positioned at the bottom of the keyword source.
Emp	Editor is currently empty.
All	The entire source is contained on the current screen.

Line Commands

Line commands in the Keyword Editor are the same as in the Natural program editor with the exception of the commands .J and .S, which cannot be used.

Each command is entered beginning in the E field; the remaining part of the command is entered in the **Keyword** field, as illustrated in the screen below:

```

09:42:39          - SYSNCP Keyword Editor -          2000-05-04
Modify Keywords      Name SAGTEST   Library SYSNCP   DBID 10   FNR 32
-----
I Line E Use  Keyword          IKN   ML Comment
-----
  1 1 Acti MENU          1004   1
  2 1 Acti DISPLAY      1002   2
  3 S Syno SHOW         1002   1
  4 . Acti i(3)TE       1001   2
  5 S Syno PURGE        1001   1

```



Caution: When you move (.M) or copy (.C) lines, ensure that individual keywords are always moved or copied together with their synonyms.

When you delete (.D) lines, the corresponding keywords and any functions containing these keywords will not be deleted from the database until you issue the STOW editor command. As long as you do not issue the STOW command, these functions will still be displayed within the Function Editor.

Function Maintenance

Functions are composed of the keywords entered in the Keyword Editor. Before it is possible to define functions, the keywords must be successfully stowed (see the section [Keyword Maintenance](#)).

- [Define Functions](#)
- [Editor Commands](#)
- [Direct Command QUICK-EDIT](#)
- [Local and Global Functions](#)
- [Procedure for Validating Functions](#)

Define Functions

Use the Define Functions function and the Function Editor to specify functions and compose valid commands which can be accessed from a specific location.

▶ To invoke the Function Editor

- 1 In the Processor Source Maintenance menu, enter Function Code **F** (Define Functions).
- 2 Press ENTER.

The Function Editor screen is displayed.

The Function Editor displays all possible combinations of the keywords stowed in the Keyword Editor.

The screen below, shows the Function Editor with keywords used as examples in the [Keyword Editor screen](#) in the section *Keyword Maintenance*:

```

09:45:53          ***** NATURAL SYSNCP UTILITY *****          2000-05-04
User SAG          - Function Editor -
Edit Global Combinations   Name SAGTEST   Library SYSNCP   DBID 10   FNR 32

Global

I Ac  Action          Object          Addition          Global Local Any Loc
- - - - -
      DELETE
      DELETE          DOCUMENT          Yes
      DELETE          FILE              Yes
      DISPLAY
      DISPLAY          DOCUMENT          Yes
      DISPLAY          FILE              Yes
      FILE
      FILE             DOCUMENT          Yes
      FILE             FILE              Yes
      INFORMATION
      INFORMATION     DOCUMENT
      INFORMATION     FILE
Repos: _____
Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Cmd  Exit Last List Flip          +   Top   Loc  Loc+  Canc

```

You have to validate each keyword combination that you want to designate as a valid function in your application. A keyword combination can be validated as a global function, local function or both. A global function can be invoked from anywhere in an application, whereas a local function can only be invoked from a specific location within an application.

Two fields in the upper left corner of this screen indicate the current validation mode (local or global) and the location for which keyword combinations can currently be validated. In the screen above, the text "Edit Global Combinations" indicates that global mode is active. If the local mode were active, the text "Edit Local Combinations" would appear here. In the screen above, the text "Global" appears below this text. This indicates that global validation can be performed for all of the combinations listed. In local mode, in this field the name of the location appears for which local validation can be performed (for example, "Local DISPLAY FILE").

The Function Editor contains the following columns:

Column	Explanation
I	Output field. The following values are output as a result of function editing. E Runtime action edited. D Referenced locations displayed. V Validation issued. R Validation removed.
Ac	Action to be taken. The following values can be entered: VG Validate as global function. VL Validate as local function. RG Remove validation as global function. RL Remove validation as local function. DL Display all functions which reference the specified function as a local function. EG Invoke the Runtime Action Editor for a global function (see <i>Runtime Action Editor</i> in the section <i>Runtime Actions</i>). EL Invoke the Runtime Action Editor for a local function (see <i>Runtime Action Editor</i> in the section <i>Runtime Actions</i>). +G Invoke global mode, so that you can maintain any global functions. +L Invoke local mode for the current line, so that you can maintain local functions for this line. IN Information about keywords in this line.
Action	These three columns are used to display all possible combinations of currently defined keywords.
Object	The text which appears at the top of each keyword column is controlled by the fields First Entry used as , Second Entry used as and Third Entry used as as specified in the processor header (see <i>Keyword Runtime Options - Header 1</i> in the section <i>Header Records</i>).
Addition	
Global	If the function has been defined as a global command, Yes appears in this field.
Local	If the function has been defined as a local command, Yes appears in this field for the current location (only displayed in local mode).
Any Loc	Any Location. If the function has been defined as a local command anywhere else within the processor, Yes appears in this field for any other location.

Editor Commands

In the Command line of the Function Editor, you can enter the following commands:

Command	Function
ANY ON	Enable the column Any Loc.
ANY OFF	Disable the column Any Loc (the column will be filled with question marks). This allows for faster scrolling in the Function Editor. Moreover, the third repositioning field is available. Also, processing-in-progress information windows will not be displayed.
FIELD	Display keyword-specific combinations.
GLOBAL	Activate global mode.
LOC	Position to next location group.
LOC+	Position forward by one location.
SINGLE ON	Display only single-word functions.
SINGLE OFF	Display all possible combinations.
TOP	Position to top of list.

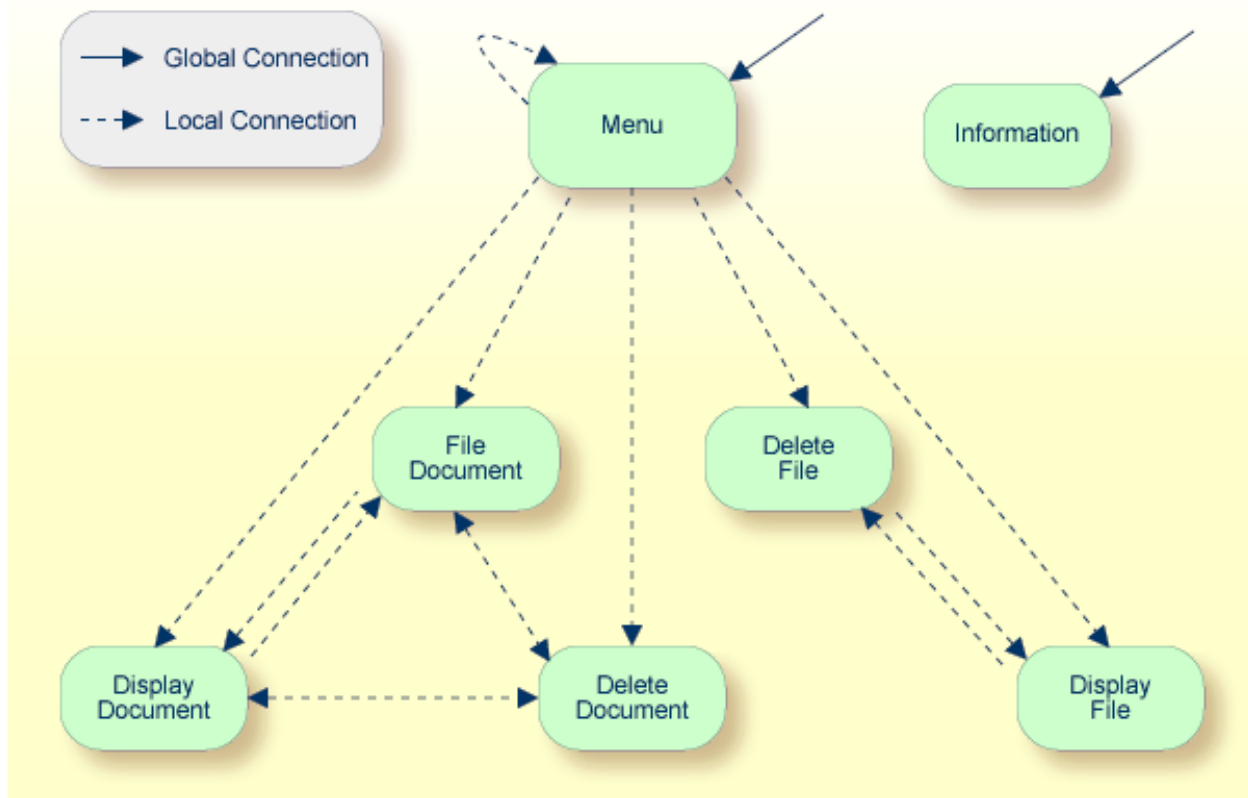
Direct Command QUICK-EDIT

The direct command QUICK-EDIT enables you to quickly define local/global functions, as well as the corresponding runtime actions, by entering keywords or IKNs directly. This may be helpful for extremely large command processors. Note, however, that the location from which the command can be issued is not verified and navigation may not function correctly at runtime.

Local and Global Functions

To understand the concept of local and global functions, you have to picture each valid keyword combination as a location in your application (for example, a location called Display File). In the Function Editor, you specify the commands which can be issued from this location, as well as from which locations this location can be reached using the command DISPLAY FILE.

Local and Global Connections within a Sample Application:



In the sample application above, the Menu and Information locations are the only locations which have been designated as global. Thus, they can be accessed directly from all of the remaining locations in the application. All locations have been designated as local to the location Menu, except Information. The only way to get from the location Display File to Display Document is via Menu.

Procedure for Validating Functions

The Function Editor operates in two modes: global and local. From global mode you can validate global functions and from local mode you can validate global and local functions. Global mode is the default mode. You can determine whether the editor is in global or local mode by the output field above the I field in the editor. If the editor is in global mode, then Global is displayed. If the editor is in local mode, then the location for which local functions are to be validated is displayed. Below is a general procedure for validating global and local functions for an application.

▶ To validate global and local functions

- 1 With the Function Editor in global mode, enter **VG** (validate global) in the Ac field next to the corresponding action to validate all global functions.

Press ENTER.

The **Runtime Action Definition** screen appears.

- 2 Press PF3 to return to the Function Editor.

Yes appears under the column heading Global beside the validated functions.

- 3 Enter **+L** in the **Ac** field for each global function validated in the previous step, to switch to local mode.

Press ENTER.

- 4 Enter **VL** (validate local) in the **Ac** field for each function that is to serve as a location for this global function.

Press ENTER.

The Runtime Action Definition screen appears.

- 5 Press PF3 to return to the Function Editor.

Yes appears under the column heading Local beside the validated functions.

- 6 To validate local functions for a *local* location: Enter **+L** (invoke local mode) in the **Ac** field for each location validated in the previous step, to validate all local functions which are to be used from this location.

Press ENTER.

- 7 Enter **VL** (validate local) in the **Ac** field for each function that is to serve as a local function for the current location.

- 8 Press PF3 to return to the Function Editor.

Yes appears under the column heading Local beside the validated functions.



Note: If in the command processor header (Processor Header Maintenance 3) the field Invoke Action Editor is set to Y, in addition, the window Runtime Action Definition (see [Runtime Action Editor](#) in the section *Runtime Actions*) is displayed for each action.

Runtime Actions

Once valid keyword combinations have been identified as either local or global functions in the Function Editor, it is possible to link each function with one or more runtime actions. Runtime actions consist of one or more steps which are to be carried out whenever a function is issued.

Below is information on:

- [Define Runtime Actions](#)

- Runtime Action Editor

Define Runtime Actions

There are two different locations in SYSNCP from which you can define runtime actions: the Function Editor (see the section *Function Maintenance*) and the Result Editor. The Result Editor is explained in this section, including how to specify runtime actions for a function.

▶ To invoke the Result Editor

- 1 In the **Processor Source Maintenance** menu, enter Function Code **R** (Define Runtime Actions).
- 2 Press ENTER.

The Result Editor screen is displayed:

```

09:47:03          ***** NATURAL SYSNCP UTILITY *****
2000-05-04
User SAG          - Result Editor -

List defined combinations   Name SAGTEST   Library SYSNCP   DBID 10   FNR
32

I Ac Location          Command          Result
-----
< Global >            MENU            KR
< Global >            INFORMATION     SF
DELETE FILE           DISPLAY FILE     SF
DELETE DOCUMENT       DISPLAY DOCUMENT SF
DISPLAY FILE          DELETE FILE      SF
DISPLAY DOCUMENT      DELETE DOCUMENT  SF
DISPLAY DOCUMENT      FILE DOCUMENT    SF
FILE DOCUMENT         DELETE DOCUMENT  SF
FILE DOCUMENT         DISPLAY DOCUMENT SF
MENU                  DELETE FILE      KCS
MENU                  DELETE DOCUMENT KCCS
MENU                  DISPLAY FILE     KRCS

Repo _____

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Cmd  Exit Last List Flip      +      Top  Loc-- Loc+ Canc

```

The Result Editor contains all of the local and global functions specified in the Function Editor. Each line in the editor represents the location from which a command can be issued (Location field), the command itself (Command field) and an abbreviated summary of the action to be carried out when the command is issued (Result field).

The fields of the screen are explained in detail in the table below:

Field	Explanation
I	Output field. Information on the last action carried out on this line.
Ac	Action to be taken. The following values can be entered: DI Display the runtime action definitions for this function. ED Edit the runtime action definitions for this function. PU Purge this function.
Location	Output field. The location within the application from which the command (see Command field below) can be issued. If the function is global, then < Global > appears in this field (the command can be issued from any location).
Command	Output field. The command. The contents of the Location and Command fields may be truncated if very long keywords are used.
Result	Output field. Contains an abbreviated summary of the action to be performed when the command is issued. The first character represents the Keep Location information (see the following section); for all other characters, see the Runtime Action Definition table below.

Runtime Action Editor

The Runtime Action Editor is used to define the actions to be taken when a command is issued from a specific location. The editor can only be invoked for functions which have been defined as global or local functions. The editor can be invoked either from the Function Editor or the Result Editor.

▶ To invoke the Runtime Action Editor from the Function Editor

1 In the **Ac** field, enter **EG** (edit global) for global functions.

Or:

In the **Ac** field, enter **EL** (edit local) for local functions.

2 Press **ENTER**.

▶ To invoke the Runtime Action Editor from the Result Editor

1 In the **Ac** field, enter **ED**.

2 Press **ENTER**.

The **Runtime Action Definition** window is displayed:

```

                                Runtime Action Definition

Location .... DISPLAY DOCUMENT
Command ..... DELETE DOCUMENT

Keep Location .... S
Data allowed ..... Y   More than one .... N   Max. Length ..... 99
Numeric ..... N   TOP of STACK ..... Y   Error: Drop ..... Y

A Runtime Action Definition
- -----
F DE-PGM_____
- _____
- _____
- _____
- _____
- _____
- _____
- _____

```

Actions are always associated with an origin and a destination. The origin is the location from which the command is issued, and the destination is the command itself. Thus, it is possible to link different actions to a command based on the context in which it is used.

In the Runtime Action Editor, you also specify whether the location is to remain the same after the actions have been carried out, or whether the command itself is to become the new current location.

Actions are specified by entering a single-letter code in the left column of the editor. Enter any parameters accompanying an action in the field next to the code. If the characters "/"* are entered in this field, all subsequent input is considered a comment. If you omit a required parameter, you will be prompted for input.

The sequence in which actions are performed at runtime is determined by the order of entry in the editor (from top to bottom). Thus, if a FETCH is specified, all of the actions specified below it are not to be performed.

The Runtime Action Editor contains the following fields:

Field	Explanation
Location	Output field. The location from which the command is issued. If the function is defined as global, the field shows < Global >.
Command	Output field. Command for which actions are to be specified.
Keep Location	Specifies whether the current or a new location is to be active once the actions have been performed. A value in this field only affects commands with a specified EXEC option. Possible values are: K Keep current location. The actions to be performed affect the current location only. S Set new location (global/local). Once the actions are performed, the command processor makes the command the new current location. Every command entered subsequently has to be either a local command of this new location or a global command. Note: The defined actions themselves have no influence on the location; that is, any action performed does <i>not</i> cause the current location to be changed.
Other Options	All other options are related to the handling of parameters provided with this command sequence. For further information, see Command Data Handling - Header 4 in the section <i>Header Records</i> . To activate the header defaults of these options, enter an asterisk (*).

► **To define runtime actions**

- 1 Invoke the **Runtime Action Definition** window as described earlier.
- 2 In the field **A**, enter an action code and the corresponding action in the field opposite to it:

Code	Runtime Action Definition
V	Default value. No runtime action is specified.
T	Text which can be read at runtime using the TEXT or GET option of the PROCESS COMMAND statement.
M	Modify command line. The data are placed in the command line.
C	Command. This command is placed at the top of the Natural stack. If an asterisk (*) is specified here, the name of the program which issued this PROCESS COMMAND statement is put on top of the stack (STACK TOP COMMAND '*PROGRAM'). (*)
D	Data. These data are placed on top of the Natural stack. (*)
F	Natural program name. The program is invoked with a FETCH statement. (*)
S	Natural STOP statement. The statement is executed at runtime. (*)
E	The value specified in this line is to be moved immediately into the system variable *ERROR-NR.
R	A return code is entered in the DDM field RETURN-CODE as described in PROCESS COMMAND in the <i>Statements</i> documentation.
1 to 9	A text string. This value is entered into the multiple DDM field RESULT-FIELD as described in PROCESS COMMAND in the <i>Statements</i> documentation.

Code	Runtime Action Definition
*	Comment line.

* These actions are only performed with the EXEC option of the PROCESS COMMAND statement.

- 3 Press PF3 to leave the **Runtime Action Definition** window.



Note: The user exit NCP-REAM allows you to use some or all of the above codes. The user exit NCP-REEM allows you to modify the line that follows the heading of the Runtime Action Definition table. The user exit NCP-REDM allows you to define default values for runtime action definitions (if you use this user exit, see also *Invoke Action Editor* in the section *Header Records*). All user exits mentioned above are delivered in the Natural system library SYSNCP.

Processor Cataloging

Once you have specified runtime actions for all of the functions you want to use in your command processor, you should catalog the command processor. Cataloging a command processor generates a Natural object of type Processor.

▶ To catalog a command processor

- 1 In the Processor Maintenance menu, enter Function Code C (Catalog Processor), the name of the command processor to be cataloged, and the name of the Natural library in which the command processor is to be cataloged.
- 2 Press ENTER.



Note: If you have Natural Security installed, you have to allow the use of your command processor as described in the *Natural Security* documentation in the section *Functional Security*.

Note for Windows, UNIX and OpenVMS:

Unlike on mainframes, SYSNCP does not create a report when cataloging a command processor.

Administrator Services

SYSNCP provides facilities for the administration of command processors. Only system administrators, as defined in *Natural Security*, are authorized to access these services.

▶ **To access the administrative services**

- 1 In the **Processor Source Maintenance** menu, enter Function Code **A** (Administrator Services).
- 2 Press ENTER.

The Administrator Services screen is displayed:


```

09:49:11          ***** NATURAL SYSNCP UTILITY *****          2000-05-04
User SAG          - Administrator Services -                               ↵

                Code  Function
                S     Select Processor
                C     Copy Processor Source
                D     Delete Processor Source
                P     Print Source/Object/NCP-Buffer
                U     Unload Processor to Work File 3
                L     Load Processor from Work File 3
                F     Freeze Processor Source
                R     References from Natural Security
                ?     Help
                .     Exit

                Code .. _      Name .. SAGTEST_  Library .. SYSNCP__

Command ==>
↵
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Cmd  Exit Last List Flip                               Canc ↵
    
```

 **Note:** If you do not have Natural Security installed, be aware that all other users have administrator status.

Below is information on:

- Select Processor
- Copy Processor Source
- Delete Processor Source
- Print Source/Object/NCP Buffer
- Unload Processor
- Load Processor
- Freeze Processor Source
- References from Natural Security

Select Processor

See the section *Processor Selection*.

Copy Processor Source

In copying processor sources, you have the choice of copying the entire processor or only selected sources (header, keywords, functions, runtime action definitions).

▶ To copy a command processor

- 1 In the Administrator Services menu, enter Function Code C.
- 2 Press ENTER.

The Copy Processor Source window is displayed to provide source and target information:

Copy Processor Source		
	Source	Target
Name	SAGTEST_	_____
Library	SYSNCP__	SYSNCP__
DBID	10__	10__
FNR	32__	32__
Password		
Cipher Key ..		
Replace	NO_	

- 3 In the Source fields, enter the name of the processor to be copied, and the library, database ID (DBID) and file number (FNR) in which the processor is stored. The default values correspond to the processor specified on the **Administrator Services** menu.

In the **Target** fields, enter the name of the processor to be copied to, and the library, database ID (DBID) and file number (FNR) into which the processor is to be copied.

In the **Cipher Key** field, enter the appropriate password and/or cipher key if the source and/or target file is protected by a password and/or cipher key.

In the **Replace** field, enter YES if you want to overwrite a processor in the target environment. The default for this field is NO.

- 4 Press ENTER.

The following window is displayed to select sources:

Copy Processor Source					
Mark	Copy	Source	Target		
—	Header	yes	no		
—	Keywords	yes	no		
—	Functions	yes	no		
	Runtime Action Definitions ..	no	no		
Source Name SAGTEST		Library SYSNCP	DBID 10	FNR 32	
Target Name TEST2		Library SYSNCP	DBID 10	FNR 32	
Replace ... NO					

- 5 In the appropriate **Mark** fields, enter any character to select the sources you want to copy.
- 6 Press ENTER.

Delete Processor Source

This function is used to delete processor sources.

▶ **To delete a command processor**

- 1 In the **Administrator Services** menu, enter Function Code **D**.
- 2 Press ENTER.

The **Delete Processor Source** window is displayed.

- 3 Specify the name of the processor to be deleted, and the library, database ID and file number in which the processor is stored. If the file is protected by a password and/or cipher key, you also have to enter the appropriate password and/or cipher key.

- 4 Press ENTER.

The following window is displayed to select the sources to be deleted:

Delete Processor Source					
Mark	Delete				Available
----	-----				-----
_	Header				yes
_	Keywords				yes
_	Functions				yes
_	Runtime Action Definitions ..				yes
Name SAGTEST Library SYSNCP DBID 10 FNR 32					

To the right of each processor source (header, keywords, functions, runtime action definitions) is a field which indicates whether the source exists. As command processor creation is a cumulative activity, you cannot delete a source without deleting all sources which are based on it. Thus, for example, in the screen above, you cannot delete the source of the functions without also deleting the source of the runtime action definitions.

- 5 In the appropriate **Mark** fields, enter any character to select each source indicated as **Available**.
- 6 Press ENTER.

Print Source/Object/NCP Buffer

In addition to processor sources, you can also print the processor object and the NCP.

▶ To print a command processor item

- 1 In the **Administrator Services** menu, enter Function Code **P**.
- 2 Press ENTER.

The **Print Source/Object/NCP-Buffer** window is displayed.

- 3 Specify the name of the processor to be printed, and the library, database ID and file number in which the processor is stored. If the file is protected by a password and/or cipher key, you also have to enter the appropriate password and/or cipher key.
- 4 Press ENTER.
- 5 The following window is displayed to select items for printing:

```

                                Print Source/Object/NCP-Buffer

Mark  Print                                     Available
-----
_     Header ..... yes
_     Keywords ..... yes

_     Functions ..... yes
_     Runtime Action Definitions .. yes

_     Processor Object ..... yes
      NCP-Buffer ..... no

      Printer ..... _____

Name SAGTEST   Library SYSNCP   DBID 10   FNR 32

```

To the right of each processor source (header, keywords, functions, runtime action definitions) is a field which indicates whether the item exists.

Possible input values for the **Printer** field are the logical printer ID, VIDEO or SOURCE; see also DEFINE PRINTER in the *Statements* documentation.

- 6 In the appropriate **Mark** fields, enter any character to select the items you want to have printed and enter the logical printer name or the value VIDEO or SOURCE in the Printer field.
- 7 Press ENTER.

Unload Processor

▶ **To unload a command processor**

- 1 In the **Administrator Services** menu, enter Function Code **U**.
- 2 Press ENTER. The **Unload Processor to Work File 3** window is displayed:

```

                                Unload Processor to Work File 3

                                Source          Target

Name ..... SAGTEST_
Library ..... SYSNCP__          SYSNCP__
DBID ..... 10__
FNR ..... 32__
Password ....
Cipher Key ..

Report ..... NO_

```

- 3 In the **Source** fields, enter the name of the processor to be unloaded, the library, database ID, and file number in which the processor can be found; the default value is the processor specified in the **Administrator Services** menu. Enter the appropriate password and/or cipher key if the file is protected by a password and/or cipher key.
- 4 In the **Report** field, enter YES if you want a report to be produced. Default is NO. You do not have to use a file extension. If you wish to use an extension, you must use the file extension ".sag".
- 5 Press ENTER.

When the processor is unloaded, all processor sources (header, keywords, functions, runtime action definitions) are written to Work File 3.



Note: Use the **Object Handler** to transfer command processors from one hardware platform to another.

Load Processor

▶ To load a command processor

- 1 In the **Administrator Services** menu, enter Function Code L.
- 2 Press ENTER.

The **Load Processor from Work File 3** window is displayed for loading processors from Work File 3 to a Natural library:

```
Load Processor from Work File 3
```

```
Replace existing processors .. N  
Produce load report ..... NO_
```

- 3 In the **Replace existing processors** field, enter **Y** or **N** (default is **N**) to specify whether existing processors with the same name are to be replaced by the processor to be loaded.
- 4 In the **Produce load report** field, enter **YES** (default is **NO**) if you want a report to be produced.
- 5 Press **ENTER**.



Note: Input for the processor name and the library into which the processor is to be loaded is taken from the work file.

Freeze Processor Source

You can freeze a processor in its current state to prevent users from modifying it further.

▶ To freeze a command processor

- 1 In the **Administrator Services** menu, enter Function Code **F**.
- 2 Press **ENTER**. The **Freeze Processor Source** window is displayed.
- 3 Specify the name of the processor to be frozen, and the library, database ID and file number in which the processor is stored. If the file is protected by a password and/or cipher key, you also have to enter the appropriate password and/or cipher key.
- 4 Press **ENTER**.
- 5 In the following window, specify with **Y** or **N** whether modification of the processor sources is to be allowed or not. Default is **Y**.
- 6 Press **ENTER**.

References from Natural Security

This function is only available if Natural Security is active in your environment. It is used to delete functional security references from Natural Security.

If functional security is defined for a processor in Natural Security, references are created automatically. These references are stored in the FNAT/FUSER system files along with the processor sources, not in FSEC.

▶ **To invoke References from Natural Security function**

- 1 In the **Administrator Services** menu, enter Function Code **R**.
- 2 Press ENTER.

The **Delete References** window appears.

- 3 Specify the name of the processor, and the library, database ID and file number in which the processor is stored. If the file is protected by a password and/or cipher key, you also have to enter the appropriate password and/or cipher key.
- 4 Press ENTER.
- 5 In the following window, you can delete main references, function references and auxiliary references.

For further information on functional security for command processors, refer to the section *Functional Security* in the *Natural Security* documentation.

Session Profile

A session profile is a collection of user-definable defaults which determine how the SYSNCP screens appear or how SYSNCP reacts to input. In a session profile, for example, you can determine which command processor you want as default for a session or which colors you want assigned to screen attributes. In SYSNCP, there is a standard session profile called STANDARD which is issued to all new users. You can create several different session profiles and activate them as required.

Administrators for SYSNCP can access and modify any session profile in SYSNCP. Other users can access all session profiles, but can modify only those session profiles which are created under their user ID or which have the same name as their user ID.

▶ **To define or modify a session profile**

- Issue the PROFILE command from the Command line of the **Processor Source Maintenance** menu.

The first of three session profile maintenance screens is displayed.

Below is information on:

- [Session Profile Name](#)
- [Session Parameters - Profile 1](#)
- [Color Attributes - Profile 2](#)

- Miscellaneous Attributes - Profile 3

Session Profile Name

The standard profile STANDARD or the value of the system variable *USER is taken as default for the profile name.

If you are defining a new session profile, the parameters/attributes are defaults. You can modify these defaults as required and save them by entering the new name and pressing PF5.

The field Session Profile Name on each profile screen is both an input and output field. Thus, it is possible to define, read or save another profile from any of these screens by entering its name in the Profile Name field and pressing PF5 or PF4, respectively.

Session Parameters - Profile 1

On the first profile maintenance screen, you can modify the following fields:

Field	Explanation
Apply Terminal Control 1	These fields can be used to enter the parameters of a SET CONTROL statement to be issued by SYSNCP at startup. For example, when you enter Z in any of the fields, SYSNCP issues the statement SET CONTROL 'Z'.
Apply Terminal Control 2	
Default Processor Name	The default command processor name to be used for this session.
Default Processor Library	The Natural library to be used to store a command processor.
Cancel Reaction	Specifies whether a warning is to be issued whenever the requested modification is not completed and the CANCEL command is issued. W Issue warning. B Back out and cancel without issuing warning.
Clear Key Allowed	Specifies whether clear key is allowed. N Clear key disallowed. Y Clear key active and has same effect as CANCEL.
Default Cursor Position	Specifies placement of the cursor. 1 Cursor to be positioned in first field of the screen. C Cursor to be positioned in command line.
Exec/Display Last Command	Specifies action to be taken as a result of the LAST command: E Execute last command issued in command line. D Display last command issued in command line.

Color Attributes - Profile 2

On the second profile maintenance screen, you can assign colors to various screen attributes, or overwrite existing color assignments.

By specifying the following color codes, you can assign the following colors:

Code	Color
BL	Blue
GR	Green
NE	Neutral
PI	Pink
RE	Red
TU	Turquoise
YE	Yellow

For color assignments to screen attributes, see also the terminal command %= in the *Terminal Commands* documentation.

Miscellaneous Attributes - Profile 3

The following attributes can be specified on the third profile maintenance screen:

Field	Explanation
Message Line Position	The line on which messages are to be displayed. The value 21 is recommended. See also the terminal command %M in the <i>Terminal Commands</i> documentation for more information.
Text for PF5 Key	The PF5 function key is reserved for global (session-wide) use. The text to be displayed on the PF-key line for PF5 can be entered in this field.
Command for PF5 Key	The PF5 function key is reserved for global (session-wide) use. The command to be executed when PF5 is pressed can be entered in this field.

In addition, the screen displays when and by which user this profile was last modified.

XV

SYSRPC Utility

The utility SYSRPC is used to maintain remote procedure calls on the client side.

Basic SYSRPC Functions

Service Directory Maintenance

Generating Interface Objects

Calculating Size Requirements

Server Command Execution

Related Topics:

- For information on how to apply the SYSRPC utility functions to establish a framework for communication between server and client systems, refer to the *Natural Remote Procedure Call (RPC)* documentation.
- For explanations of expressions relevant to the SYSRPC utility, see also the section *Natural RPC Terminology* in the *Natural Remote Procedure Call (RPC)* documentation.
- The use of SYSRPC can be controlled by Natural Security: see *Protecting Utilities* in the *Natural Security* documentation.
- For information on Application Programming Interfaces provided to maintain remote procedure calls, see *Application Programming Interfaces for Use with Natural RPC* in the *Natural Remote Procedure Call (RPC)* documentation.

71 Basic SYSRPC Functions

▪ Invoking SYSRPC	490
▪ Terminating SYSRPC	491
▪ Service Directory Tree	492
▪ Menu Bar	492
▪ Toolbar	495
▪ Context Menu	496

This section provides instructions for starting and terminating the SYSRPC utility and describes the main features and functions provided by the utility.

Invoking SYSRPC

▶ To invoke the SYSRPC utility

- 1 From the library workspace in the Natural Studio tree view, select a library.
- 2 From the **Tools** menu, choose **Configuration Tools** and **Remote Procedure Call**.

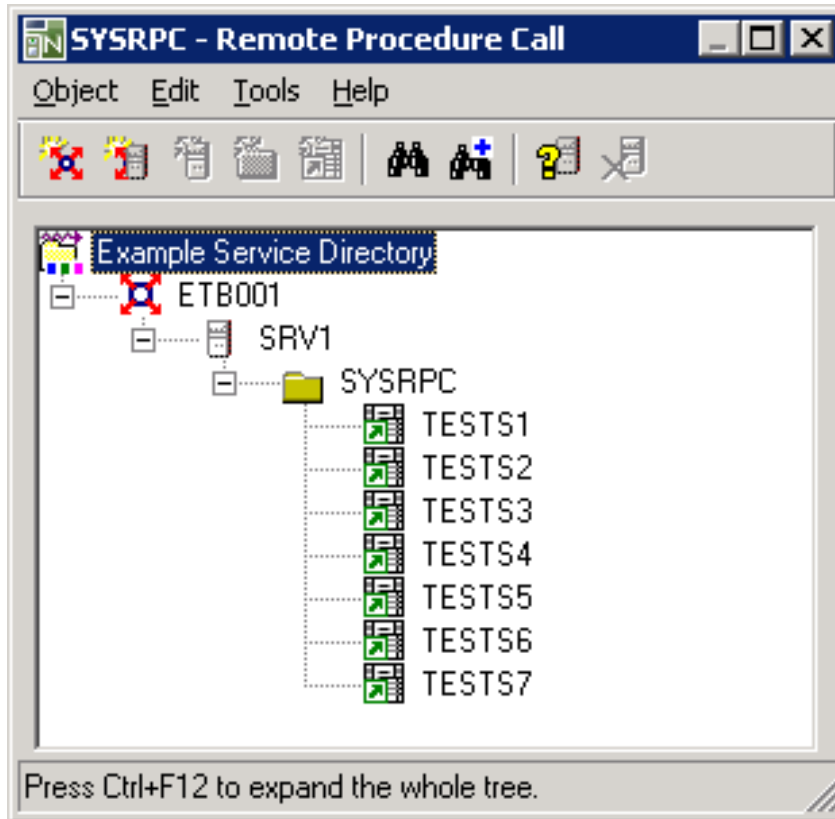
Or:

In the Command line, enter the following command:

```
SYSRPC
```

The **SYSRPC - Remote Procedure Call** window appears and displays the service directory tree view for the library specified. This is indicated in the name of the directory root node: **Service Directory** [*library-name*].

When you invoke SYSRPC for the first time for a library, as shown in the example below, the tree view contains example (dummy) data. The name of the service directory root node is **Example Service Directory** which will change to **Service Directory** [*library-name*] when you use the **Save** or **Save As** function (see [Menu Bar](#) below), regardless of any tree view modifications. For a list of possible root names, see [Root Node Names](#) in the section *Service Directory Maintenance*.



The menus and toolbar buttons in the **SYSRPC - Remote Procedure Call** window provide all functions required to maintain a service directory, generate interface objects and execute server commands.

Terminating SYSRPC

▶ To terminate the SYSRPC utility

- In the **SYSRPC - Remote Procedure Call** window, from the **Object** menu, choose **Exit**.

Or:

Choose ALT+F4.

Or:

Choose the standard Windows close function.



If a window appears with a message saying that subprogram NATCLTGS is missing (needed at runtime), choose **Yes** to confirm the generation of NATCLTGS, or choose **No** to cancel the operation.

Service Directory Tree

The tree view in the **SYSRPC - Remote Procedure Call** window displays all items (tree nodes) required for service directory maintenance. For explanations of the tree nodes, see [Tree Nodes](#) in *Service Directory Maintenance*. For explanations of the tree node hierarchy, see [Service Directory Concept](#) in *Service Directory Maintenance*.

Do not confuse a tree node of the service directory with the node to which an RPC connection is established (see *Natural RPC Terminology* in the *Natural Remote Procedure Call* documentation).

You can manipulate the tree nodes by using the functions provided with the **menu bar**, the **toolbar** and the **context menu** described below.

You can expand or collapse tree nodes by choosing the toggle  (expand) or the toggle  (collapse) in front of a tree node. Alternatively, you can choose the **ARROW** keys. Double-click on an expandable tree node to display all subordinate items.

▶ **To expand all nodes of a tree**

- In the **SYSRPC - Remote Procedure Call** window, select the root node and choose **Expand Tree** from the context menu.
















Or:





Choose **CTRL+F12**.

Menu Bar

The menus, menu items and equivalent shortcut keys (if available) provided to execute SYSRPC functions are described in the following section.

Menu	Menu Item and Shortcut	Explanation
Object	Open	<p>Opens a service directory:</p> <p>From the Type drop-down list box, select the type of service directory (default is SERVDIRX or NATCLTGS) and, if required, in the Library text box, replace the name of the library (default is the current library).</p> <p>The non-modifiable text box underneath Library displays the database ID (DBID), the file number (FNR) and the type of Natural file (FNAT = system, FUSER = user) for the library specified.</p>
	Save CTRL+S	Saves the service directory in the current library.










Menu	Menu Item and Shortcut	Explanation										
	Save As	Saves the service directory in another library.										
	Properties	<p>Invokes the Properties of Service Directory dialog box. It provides information on the generation of the service directory:</p> <p>Object The name of the service directory.</p> <p>Generated by The name of the Natural utility.</p> <p>Environment If the directory was generated in a local environment, the value <code>local</code> is displayed. Otherwise, the name of a remote server is displayed.</p> <p>Library The name of the library in which the service directory was generated.</p> <p>User The ID of the user who generated the service directory.</p> <p>Time stamp The date and the time at which the service directory was last modified.</p> <p>Tree view nodes The number of tree nodes including the service directory root node.</p> <p>Expiration time The remote directory data is loaded at runtime. The expiration time in seconds determines the period of validity of this data. If directory data is requested after the expiration time set, it will automatically be reloaded. If the expiration time is set to 0, the remote directory data will not be reloaded.</p>										
	Exit CTRL+F4	Terminates SYSRPC.										
Edit	New Item	<p>Creates a new tree node item. Depending on the tree node selected, you have the following choices:</p> <table border="1"> <tbody> <tr> <td>Node</td> <td>Corresponding toolbar button: </td> </tr> <tr> <td>Logical Service (EntireX)</td> <td>Corresponding toolbar button: </td> </tr> <tr> <td>RPC Server</td> <td>Corresponding toolbar button: </td> </tr> <tr> <td>Library</td> <td>Corresponding toolbar button: </td> </tr> <tr> <td>Service (Subprogram)</td> <td>Corresponding toolbar button: </td> </tr> </tbody> </table> <p>See also the section <i>Natural RPC Terminology</i> in the <i>Natural Remote Procedure Call (RPC)</i> documentation and Location Transparency in the section <i>Service Directory Maintenance</i> for explanations of relevant expressions.</p>	Node	Corresponding toolbar button: 	Logical Service (EntireX)	Corresponding toolbar button: 	RPC Server	Corresponding toolbar button: 	Library	Corresponding toolbar button: 	Service (Subprogram)	Corresponding toolbar button: 
Node	Corresponding toolbar button: 											
Logical Service (EntireX)	Corresponding toolbar button: 											
RPC Server	Corresponding toolbar button: 											
Library	Corresponding toolbar button: 											
Service (Subprogram)	Corresponding toolbar button: 											
	Rename CTRL+F2	Modifies the name of a tree node.										

Menu	Menu Item and Shortcut	Explanation
	Delete	Removes a tree node.
	Cut CTRL+X	Cuts, copies or pastes a tree node.
	Copy CTRL+C	
	Paste CTRL+V	
	Find CTRL+F	<p>Invokes the Find Item window to search for a name:</p> <p>Find what Enter an alphanumeric search string of up to 32 characters.</p> <p>Case sensitive Select this check box to distinguish between uppercase and lowercase characters.</p> <p>Whole words only Select this check box to search for complete character strings only.</p> <p>Choose OK to start searching and move to the first hit, which is highlighted.</p> <p>Corresponding toolbar button: </p>
	Find Next CTRL+F3	<p>Searches for additional instances of the search string specified in the Find Item window and moves to the next hit if one exists.</p> <p>Corresponding toolbar button: </p>
Tools	Ping CTRL+F9	<p>Sends an internal message to verify server connections described in the section <i>Server Command Execution</i>.</p> <p>Corresponding toolbar button: </p>
	Terminate Server	<p>Sends an internal message to terminate server connections as described in the section <i>Server Command Execution</i>.</p>
	Terminate EntireX Broker Service	<p>Requests termination of an EntireX Broker service as described in the section <i>Server Command Execution</i>.</p> <p>Corresponding toolbar button: </p>
	Single Interface Object Generation CTRL+F8	<p>Generates single interface objects as described in <i>Generating Single Interface Objects with Parameter Specification</i> in the section <i>Generating Interface Objects</i>.</p>
	Interface Object Mass Calculation CTRL+F5	<p>Performs calculations for size requirements of RPC calls as described in the section <i>Calculating Size Requirements</i>.</p>
	Interface Object Mass Generation CTRL+F6	<p>Generates single or multiple interface objects online or batch as described in <i>Using the Interface Object Mass Generation Function</i> in the section <i>Generating Interface Objects</i>.</p>

Menu	Menu Item and Shortcut	Explanation
	Interface Object Generation from IDL CTRL+F7	Generates interface objects or parameter data areas (PDAs) from IDL files as described in <i>Generating Interface Objects or PDAs from IDL Files</i> in the section <i>Generating Interface Objects</i> .
Help		Displays SYSRPC help text: SYSRPC Utility - Overview Server Command Execution (Ping, Terminate) Expiration Time Logon Option Transport Protocol Location Transparency Service Directory Tree Generating Single Interface Objects with Parameter Specification Generating Multiple Interface Objects Generating Interface Object or PDAs from IDL Files Calculating Size Requirements

Toolbar

The toolbar buttons that provide quick access to frequently used SYSRPC functions are described in the following section.

Toolbar Button	Function
	Adds a new tree node item. See also the corresponding menu: Edit > New Item > Node .
	Adds a new logical service (EntireX) item. See also the corresponding menu: Edit > New Item > Logical Service (EntireX) .
	Adds a new RPC server item. See also the corresponding menu: Edit > New Item > RPC Server .
	Adds a new library item. See also the corresponding menu: Edit > New Item > Library .
	Adds a new service (subprogram) item. See also the corresponding menu: Edit > New Item > Service (Subprogram) .
	Finds a character string. See also the corresponding menu: Edit > Find .
	Finds the next character string. See also the corresponding menu Edit > Find Next .
	Pings RPC servers. See also the corresponding menu: Edit > Ping .
	Terminates an EntireX Broker service. See also the corresponding menu: Edit > Terminate EntireX Broker Service .

Context Menu

The context menu provides alternative ways of executing the commands and functions provided by the menus and the toolbar in the **SYSRPC - Remote Procedure Call** window.

In addition, the context menu provides the following:

- the command **Expand Tree** (described earlier in *Service Directory Tree*),
- the **Logon Option** (see *Service Directory Maintenance*) and
- the option to specify the **Transport Protocol** (see *Service Directory Maintenance*).

72 Service Directory Maintenance

▪ Service Directory Concept	498
▪ Tree Nodes	500
▪ Logon Option	504
▪ Transport Protocol	504

The SYSRPC utility provides functions used to maintain a service directory in order to connect the client's calling program to a subprogram on a server.

The service directory information is stored in the NATCLTGS subprogram and the XML-formatted SERVDIRX file (Natural text object) in the library that is defined with the profile parameter `RPCSDIR` (see the *Parameter Reference* documentation). If `RPCSDIR` is set, the service directory maintenance functions reference the library specified with `RPCSDIR`. If `RPCSDIR` is not set (this is the default), the library where you are logged on is referenced. In this case, log on to the library (or one of its steplibs) used by the client at runtime before you perform a service directory maintenance function.

The name of the library referenced for service directory maintenance is indicated in the root node of the service directory tree view (see *Tree Nodes*). If `RPCSDIR` is set, the root node contains **Central**, which indicates that the library displayed in the window is *not* the library where you are currently logged on, but the central library specified with `RPCSDIR`.

For further information on how to apply the service directory maintenance functions, refer to *Specifying RPC Server Addresses* described in *Operating a Natural RPC Environment* in the *Natural Remote Procedure Call (RPC)* documentation.

Attention:

If NATCLTGS is stored in the Natural system library SYSRPC, we strongly recommend that you move NATCLTGS to the application library (or one of its steplibs) used by the client.

Service Directory Concept

The main items of a service directory are node, server, library and service (subprogram). The hierarchical structure of these items is displayed as a tree view in the **SYSRPC - Remote Procedure Call** window (see also *Service Directory Tree* in *Basic SYSRPC Functionality*). The highest level (root node) of the tree view is **Service Directory** and the lowest is service (subprogram).

The node and server names specified in the service directory are either physical names or logical names and logical services.

This section covers the following topics:

- [Physical Nodes and Servers](#)
- [Location Transparency](#)

- [Example of a Service Directory](#)

Physical Nodes and Servers

Physical node and server names denote the names of real nodes (valid TCP/IP or Entire Net-Work addresses) and servers.

In the [Example of a Service Directory](#) below, two servers are defined for one node. Both servers are connected to the same node: ETB045. The remote CALLNAT to subprogram SUB1 is executed on server NRPC001, whereas subprograms SUB2 and SUB3 are executed on server NRPC002.

The server names specified here must be identical to the server names specified for the server with the profile parameter SRVNAME described in the *Parameter Reference* documentation. Analogously, the node name in the service directory must be identical to the node name specified for the server with the profile parameter SRVNODE described in the *Parameter Reference* documentation.

Location Transparency

Location transparency is a concept where physical node names can be replaced by logical node names, and a combination of physical node and server names can be replaced by logical services.

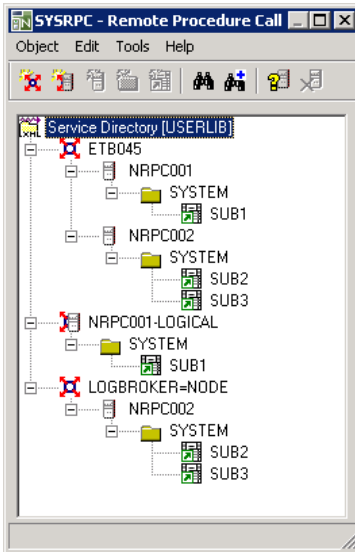
Logical node names and logical services are defined with EntireX and are assigned to physical node and server names at Natural runtime.


See also [Example of a Service Directory](#) and [Using Logical Services and Logical Node Names](#).

Related Topics:

- [Using Location Transparency in Operating a Natural RPC Environment in the Natural Remote Procedure Call \(RPC\) documentation](#).
- The relevant sections in the EntireX documentation.

Example of a Service Directory

















In the example of a service directory above, the icon  indicates that NRPC001-LOGICAL is a logical service. LOGBROKER=NODE indicates that NODE is the logical node name.

Tree Nodes

This section describes the tree nodes contained in the service directory tree view. Each tree node is identified by a different icon.

You can manipulate the tree nodes by using the functions provided by the **menu bar**, the **toolbar** and the **context menu** described in *Basic SYSRPC Functionality*.

Icon	Tree Node	Explanation
	Service Directory root	<p>The service directory root node indicates the name of the library from which the service directory was read: Service Directory [library-name].</p> <p>For example: If you invoked the SYSRPC utility from the library USERLIB, the root reads Service Directory [USERLIB].</p> <p>If the profile parameter RPCSDIR is set, the root node reads Central Service Directory [USERLIB], which indicates that the service directory is read from a central library specified with RPCSDIR.</p> <p>For explanations of other root node names that can occur, see Root Node Names in Error Situations.</p>
	Node	The name of the node to which the remote CALLNAT is sent.


Icon	Tree Node	Explanation
		<p>Maximum name length:</p> <p>Physical nodes: 32 characters Logical nodes: 192 characters</p> <p>Depending on the setting of the Logon Option, either of the following icons is displayed:</p> <p> Logon = No</p> <p> Logon = Yes</p> <p>See also Logon Option below.</p>
 	Server	<p>The name of the server on which the CALLNAT is to be executed.</p> <p>Maximum name length: 32 characters</p> <p>Depending on the setting of the Logon Option, either of the following icons is displayed:</p> <p> Logon = No</p> <p> Logon = Yes</p> <p>See also Logon Option below.</p>
 	Logical Service	<p>The name of a logical service.</p> <p>Maximum name length: 192 characters</p> <p>Depending on the setting of the Logon Option, either of the following icons is displayed:</p> <p> Logon = No</p> <p> Logon = Yes</p> <p>See also Logon Option below.</p>
	Library	<p>SYSTEM or the name of the library to which your client application is logged on during the execution of the remote CALLNAT.</p>
	Service (Subprogram)	<p>The name of the remote subprogram to be accessed from the client.</p> <p>Maximum number of entries: 500 subprograms.</p>

The section below contains information on:

- [Root Node Names in Error Situations](#)
- [Selection Criteria for Node and Server](#)
- [Using Logical Services and Node Names](#)

Root Node Names in Error Situations

This section lists root node names that can occur if subprograms or text objects required by the service directory are missing, explains possible reasons and provides resolution advice.

Node Name	Reason	Resolution
Service Directory from NATCLTGS [<i>library-name</i>]	The text object SERVDIRX is missing. This is indicated by the icon  .	From the Object menu, choose Save As or Save . SERVDIRX is generated into the library for which SYSRPC was invoked and the name of the root node changes to Service Directory [<i>library-name</i>].
Example Service Directory	The subprogram NATCLTGS and the text object SERVDIRX are missing.	From the Object menu, choose Save As or Save . NATCLTGS and SERVDIRX are generated into the library for which SYSRPC was invoked and the name of the root node changes to Service Directory [<i>library-name</i>].
Empty tree	NATCLTGS, SERVDIRX and the subprogram DEF-GS (delivered in the Natural system library SYSRPC) are missing. DEF-GS contains example data.	1. Create at least one new item for a node and a server or create at least one logical service. 2. Save the modifications. NATCLTGS and SERVDIRX are generated into the library for which SYSRPC was invoked and the name of the root node changes to Service Directory [<i>library-name</i>].

Selection Criteria for Node and Server

At Natural runtime, the selection of a node and a server depends on the value of the service (subprogram) and library tree nodes. Comply with the following conditions:

Non-conversational CALLNAT

1. The library tree node must contain the name of the current application library or SYSTEM.
2. The subprogram referenced in the CALLNAT statement must be contained in the service (subprogram) tree node, which belongs to the library tree node in point (1).

Conversational CALLNAT

1. The library tree node must contain the name of the current application library or SYSTEM.
2. All subprograms specified in the OPEN CONVERSATION statement must be contained in a service (subprogram) tree node, which belongs to the library tree node in point (1).

The node and server used for a non-conversational or conversational CALLNAT are taken from the superior node and server tree nodes of the library tree node in point (1).

Using Logical Services and Node Names

You can define logical services and logical node names. For an example of logical services and logical node names, see [Example of a Service Directory](#).

▶ To define a logical service

- Select the **Service Directory** root node, open the **Edit** menu and choose **New Item** and **Logical Service (EntireX)**.

Or:

Select the **Service Directory** root node and choose **New Item** and **Logical Service (EntireX)** from the context menu.

▶ To define a logical node name

- Select the tree node for the required node, choose **Rename** from the context menu or press F2, and replace the existing value by the following:

```
LOGBROKER=node - name
```

where *node - name* denotes the logical EntireX Broker name.

▶ To remove a logical node name

- Select the tree node for the required node, choose **Rename** from the context menu or press F2, and replace the string LOGBROKER= by an EntireX Broker name.

▶ To display physical names defined for logical services or nodes

- Use the **ping** command as described in the section [Server Command Execution](#).

Ping invokes a window that displays the physical node and server names defined for a logical service or a physical node name defined for a logical node.

Logon Option



If the **Logon Option** is set, for each `CALLNAT` request, the client initiates a Natural logon to the server using the current library name on the client, regardless of the library specified in the service directory. You can use the Application Programming Interface `USR4008N` to specify a different library; see also *Logging on to a Different Library* in *Using the Logon Option* in the *Natural Remote Procedure Call (RPC)* documentation.

After the remote `CALLNAT` has been executed (successfully or not), the server library is reset to its previous state. For more information, see *Using the Logon Option* in the *Natural Remote Procedure Call (RPC)* documentation.



The **Logon Option** can be set at server or node level and applies to all definitions made at a hierarchically lower level. If the **Logon Option** has been set for a certain server, it applies to all associated library and subprogram definitions.

▶ To set the Logon Option

- 1 In the **Service Directory** tree view, select the tree node of a node, a server or a logical service and choose **Logon Option** from the context menu.
- 2 Choose **Yes** to set the **Logon Option** for the server. (The default is **No**.)

If the **Logon Option** has been set successfully for the node selected, the icon indicating a node changes from  to .

If the **Logon Option** has been set successfully for a logical service, the icon indicating a logical service changes from  to .

If the **Logon Option** has been set successfully for the server selected, the icon indicating a server changes from  to .

Transport Protocol

▶ To specify the transport protocol

- In the **Service Directory** tree view, select the tree node of a node, a server or a logical service and choose **Transport Protocol** and **ACI** for EntireX Broker ACI protocol from the context menu.

73

Generating Interface Objects


An interface object is a Natural subprogram that is used to connect the client's calling program to a subprogram on a server.

Interface objects are actually not required if automatic Natural RPC (Remote Procedure Call) execution is used with the one important exception described below. However, it can be advantageous to generate interface objects as explained in *Interface Objects and Automatic RPC Execution* in the section *Operating a Natural RPC Environment* in the *Natural Remote Procedure Call (RPC)* documentation.

Note for EntireX RPC Servers:

An interface object is required if the IDL (Interface Definition Language) definition of the subprogram you want to call on an EntireX RPC server contains a group structure. In this case, you have to define the same group structure in the interface object by using the appropriate **YSRPCInterface Object Generation** function described in this section.

You can generate an interface object from new parameter definitions or from existing definitions in a subprogram.

 **Caution:** The subprogram used for generating an interface object can no longer be referenced in the local environment on the client side. The function **Interface Object Generation** completely changes the source of the subprogram so that it becomes unusable for local program calls.

The following sections describe the functions and commands provided to generate single or multiple interface objects:

- [Generating Single Interface Objects with Parameter Specification](#)
- [Generating Multiple Interface Objects](#)
- [Generating Interface Objects or PDAs from IDL Files](#)

74 Generating Single Interface Objects with Parameter

Specification

- Using the Interface Object Generation Function 508
- Specifying Parameters 510
- Examples of Interface Object Generation 512

The function **Single Interface Object Generation** provides the option to generate single interface objects online on a separate window. You either type in the parameter definitions required or read them in from an existing subprogram.

Using the Interface Object Generation Function

This section provides instructions for generating single interface objects by using the function **Single Interface Object Generation**.

► **To generate a single interface object**

- 1 In the **SYSRPC - Remote Procedure Call** window, from the **Tools** menu, choose **Single Interface Object Generation**.

Or:

In the **SYSRPC - Remote Procedure Call** window, press CTRL+F8.

The **Input for Interface Object Generation** dialog box appears.

- 2 In the **Name** text box, enter the name of the interface object to be generated.

The name of the interface object must be identical to the name of the remote **CALLNAT** program.

If required, in the **Library** text box, enter the name of the library into which you want to generate the interface object. The text box is preset to the name of the current library.

DBID, FNR is a non-modifiable text box that displays the database ID (DBID), the file number (FNR) and the type of Natural file (FNAT = system, FUSER = user) for the specified library.

From the **Compression** drop-down list box, select compression type **0, 1** or **2** (default is **1**); see *Using Compression* described in *Operating a Natural RPC Environment* in the *Natural Remote Procedure Call (RPC)* documentation.

- 3 Choose **OK**.
 - If the name entered in the **Name** text box corresponds to the name of an object that already exists in the specified library, a window appears with an appropriate message:

Choose **Yes** if you want to modify existing parameter definitions. If the specified name is identical to a cataloged object of the type subprogram, the **Interface Object Generation** window appears where the parameters definitions of the respective subprogram are contained in a table. If the specified name is identical to an interface object for which also a source object exists, all field attributes (see also *Specifying Parameters*) from a previous generation are retained. Otherwise, all field attributes are set to **M** (modifiable).

Or:

Choose **No** if you want to create new parameter definitions. The **Interface Object Generation** window appears with empty table cells.

Note that you can still keep old definitions, even after you have entered new values if you abort execution by choosing **Cancel**.

- If the name entered in the **Name** box does *not* correspond to the name of an object contained in the specified library, an empty **Interface Object Generation** window appears.
- 4 In the **Interface Object Generation** window, add or modify the parameters to be used in the interface object: Enter a value or select it from a drop-down list box as described in [Specifying Parameters](#).
 - 5 Choose **OK** to generate the interface object and to exit the **Interface Object Generation** window.

A window appears which confirms that the interface object is generated into the specified library. In addition, the window indicates the size the interface object requires for sending data from the client to the server or vice versa. The size includes internal RPC information used for the interface object. The indication of the size helps you configure the middleware layer used; for example, the Broker attribute file when EntireX Broker is used.

The following message appears in the window when you generate an interface object from the example subprogram TESTS5 (see [Example 1](#) below):

```
Interface Object TESTS5 is generated in library TEST
It requires:
    Send length: 2249 bytes
    Receive length: 2221 bytes
```

If dynamic parameters, X-arrays or X-group arrays are used, this message only indicates the minimum length requirements. The actual length requirements can only be determined during program execution and may be different from call to call.

If the `Send length` or the `Receive length` exceeds the Entire Net-Work limit of 32000 bytes, a window appears with a corresponding warning:

Choose **Y** (Yes) to continue, or **N** (No) to cancel the generation.

If you choose **Y** (Yes), this setting is kept for the entire SYSRPC session, that is, you can continue generating interface objects without receiving further warnings.

If the total data (without internal RPC information) sent or received exceeds the limit of 1073739357 bytes (which is 1 GB minus 2467 bytes of internal RPC information), SYSRPC stops processing and issues a corresponding error message. This error message displays the subtotal of the data in bytes that could be transferred at the field up to which the subtotal was

calculated. The corresponding field is then marked. In this case, reduce the amount of data before you continue generating the interface object.

If the interface object was generated in the Natural system library SYSRPC, you it object to the application library or steplib using the Natural transfer utility SYSMAIN or the Object Handler. Note that you may have to recatalog the source of the interface object in the target environment.

Specifying Parameters

In the table cells provided in the **Interface Object Generation** window, you can enter the parameter definitions that are used in the interface object. You can specify a maximum of 5000 parameters. Unless indicated in the table below, input in the boxes is mandatory.

Field	Description
Level	<p>The level of the field.</p> <p>A level can be a number in the range from 01 (highest level) to 99 (lowest level). The leading 0 is optional.</p> <p>See also <i>Defining Groups</i> and <i>Example 2</i> for an example of a group definition.</p>
Attribute	<p>The attribute of the parameter:</p> <p>M (modifiable - INOUT), 0 (output - OUT) or I (input - IN).</p> <p>Parameters assigned a level number of 2 or greater are considered to be a part of a group. Parameters within a group must have the same attribute as the immediately preceding group that is assigned one level higher. For nested groups, this is the attribute of the group with the highest level. For an example of a group definition, see <i>Example 2</i>.</p> <p>If an interface object has been generated from a subprogram, the attribute is M by default, which may need modification.</p> <p>If an interface object has been generated from another interface object, the attribute values specified for the original object are retained.</p> <p>The generated interface object contains a comment that indicates the attribute specified for the parameter: IN, OUT or INOUT.</p>
Type	<p>A Natural data format such as N (numeric) and G (group), or K (Kanji). Natural data formats C (attribute control) and Handle are not allowed.</p> <p>For a description of Natural data formats, see <i>Format and Length of User-Defined Variables</i> and <i>Special Formats</i> in the section <i>User-Defined Variables</i> in the <i>Programming Guide</i>.</p>
Length	<p>The length of the parameter or DYNAMIC.</p>

Field	Description
	<p>This field does not apply to the following Natural data formats: D (date), G (group), L (logical) and T (time).</p> <p>The Natural data format A is restricted to 1073739357 bytes, Natural data format B is restricted to 536869678 bytes.</p> <p>DYNAMIC indicates a dynamic parameter and applies to the Natural data formats A and B.</p>
Precision	<p>Only applies to Natural data formats N (numeric) and P (packed). Optional.</p> <p>The precision of the parameter, that is, the number of digits after the decimal point.</p>
Dimension 1/2/3	<p>Only applies to arrays. Optional.</p> <p>The first, second and third dimension of the parameter.</p> <p>An X-array or an X-group array is specified by entering an asterisk (*) for a dimension.</p> <p>See also <i>Defining X-Arrays and X-Group Arrays</i>.</p>

The section below contains information on:

- [Defining Groups](#)
- [Defining X-Arrays and X-Group Arrays](#)

Defining Groups

You only need to define a group structure for a client Natural object that calls a non-Natural object located on an EntireX RPC server. The group structure must correspond to the IDL definition in EntireX. A group structure is not required for a client Natural object that calls a subprogram located on a Natural RPC server.

Group arrays and X-group arrays passed from a client Natural object to an interface object must be contiguous. Therefore, we strongly recommend that you always pass a complete array to the object by using asterisk (*) notation for all dimensions. We also strongly recommend that you use identical data definitions in the client Natural program, the interface object and the server program.




Caution: Any group definitions in a subprogram will be ignored when an interface object is generated from this subprogram. In this case, you have to define the group again on the **Interface Object Generation** screen and adapt the dimension of the group elements accordingly. (Dimensions defined within a group are propagated to the parameter definitions at a lower level.) If you generate an interface object from another interface object that contains a group, the group definitions will be retained.

See also [Example 2](#) for an example of a group definition.

Defining X-Arrays and X-Group Arrays

If any dimension of a parameter is extensible, all other dimensions of the parameter are also extensible. If you define extensible and fixed dimensions for a parameter in a subprogram, the **Interface Object Generation** function issues a warning and automatically changes the fixed dimension to an extensible dimension as demonstrated in [Example 3](#). In a group structure, you can define either an extensible or a fixed dimension for each level. There is no automatic change of a fixed dimension to an extensible dimension between levels.

Natural RPC only supports extensible upper bounds. All X-arrays and X-group arrays in the generated `DEFINE DATA PARAMETER` area of the interface object are therefore defined as `(1:*)`.

 **Caution:** If you generate an interface object from a subprogram that contains an X-array or X-group array with an extensible lower bound, the extensible lower bound will be converted to an extensible upper bound.

For an example of a group with an extensible dimension, see [Example 3](#).

Examples of Interface Object Generation

This section provides examples of Natural subprograms and the interface objects generated from them.

The parameter definitions indicated below are extracted from example subprograms, which are supplied in the Natural system library `SYSRPC`.

Example 1

The following `DEFINE DATA PARAMETER` area (example subprogram `TESTS5`) shows four modifiable parameters and the corresponding parameter definitions in the **Interface Object Generation** window:

```
DEFINE DATA
PARAMETER
  01 #IDENTIFIER (A10)
  01 #N-OF-ID (I4)
  01 #FREQ (P5.2)
  01 #A100 (A100/5,4)
```

Interface Object Generation								
	Level	Attribute	Type	Length	Precision	Dimension 1	Dimension 2	Dimension 3
1	01	M	A	10				
2	01	M	I	4				
3	01	M	P	5	2			
4	01	M	A	100		5	4	

Example 2

The following DEFINE DATA PARAMETER area (example subprogram TESTS6) shows a nested group structure and the corresponding parameter definitions in the **Interface Object Generation** window:

```

DEFINE DATA
PARAMETER
  01 GROUP-1(10)
    02 A (A20)
    02 B (A20)
    02 GROUP-2(20)
      03 C (A10/5)
      03 D (A10)
  01 LINE (A) DYNAMIC
    
```

Interface Object Generation								
	Level	Attribute	Type	Length	Precision	Dimension 1	Dimension 2	Dimension 3
1	01	M	G			10		
2	02	M	A	20				
3	02	M	A	20				
4	02	M	G			20		
5	03	M	A	10		5		
6	03	M	A	10				
7	01	M	A	DYNAMIC				

Example 3

The following DEFINE DATA PARAMETER area (example subprogram TESTS7) shows a nested group structure with extensible dimensions and the corresponding parameter definitions in the **Interface Object Generation** window.

```

DEFINE DATA
PARAMETER
 01 GROUP-1(10)
  02 A (A20)
  02 B (A20)
  02 GROUP-2(0:*)
    03 C (A10/5)
    03 D (A10)
 01 LINE (A) DYNAMIC
    
```

Interface Object Generation								
	Level	Attribute	Type	Length	Precision	Dimension 1	Dimension 2	Dimension 3
1	01	M	G			10		
2	02	M	A	20				
3	02	M	A	20				
4	02	M	G			*		
5	03	M	A	10		5		
6	03	M	A	10				
7	01	M	A	DYNAMIC				

75

Generating Multiple Interface Objects

- Using the Function Interface Object Mass Generation 516
- Using the SYSRPC SGMASS Command 520
- Name Specification and Compression 520

You can generate single or multiple interface objects in either online or batch mode by using the function the **Interface Object Mass Generation** or the command `SYSRPC SGMASS`. Either method will invoke a window that indicates the send and receive lengths required by the specified subprogram(s).

Using the command or function, field attributes will be generated as described in *Specifying Parameters* in the section *Generating Single Interface Objects with Parameter Specification*.

You generate interface object from subprograms.

Using the Function Interface Object Mass Generation

This section provides instructions for generating single or multiple interface objects by using the **Interface Object Mass Generation** function.

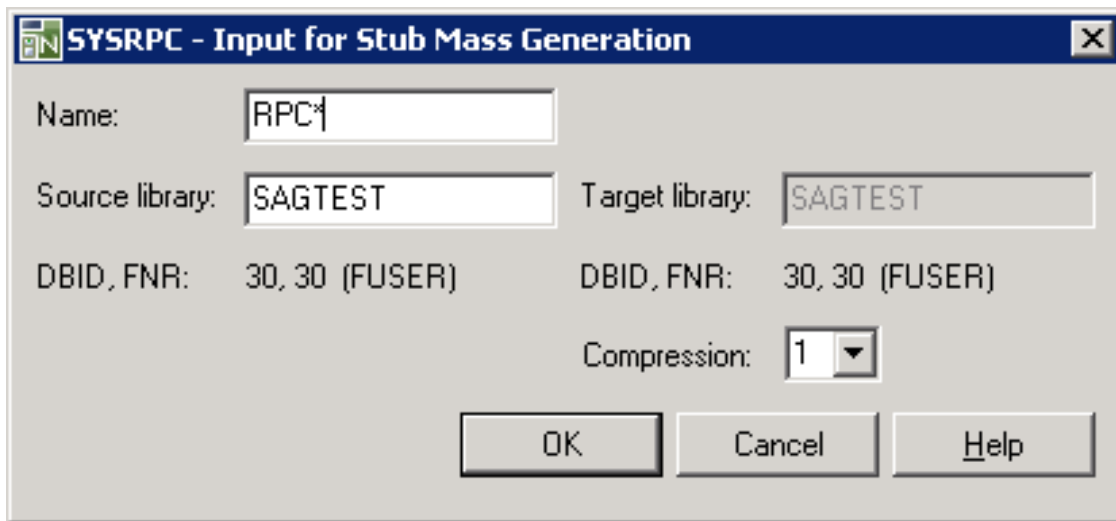
▶ **To perform the Interface Object Mass Generation function**

- 1 In the **SYSRPC - Remote Procedure Call** window, from the **Tools** menu, choose **Interface Object Mass Generation**.

Or:

In the **SYSRPC - Remote Procedure Call** window, press `CTRL+F6`.

An **SYSRPC - Input for Interface Object Mass Generation** dialog box similar to the example below appears:



- 2 In the **Name** text box, enter the name of the interface object to be generated or specify a range of names. The text box is preset to asterisk (*), indicating all subprograms. For valid entries, see *Name* in *Name Specification and Compression*.

The name(s) of the interface object must be identical to the name(s) of the remote CALLNAT program(s).



Important: If you do not specify a name, with few exceptions (see below), all subprograms in the current library will be converted to interface objects.

If required, in the **Source library** text box, enter the name of the library that contains the subprogram(s) from which you want to generate the interface object(s). The text box is preset to the name of the current library.

Target library is a read-only text box that contains the name of the current library into which the interface objects are generated.

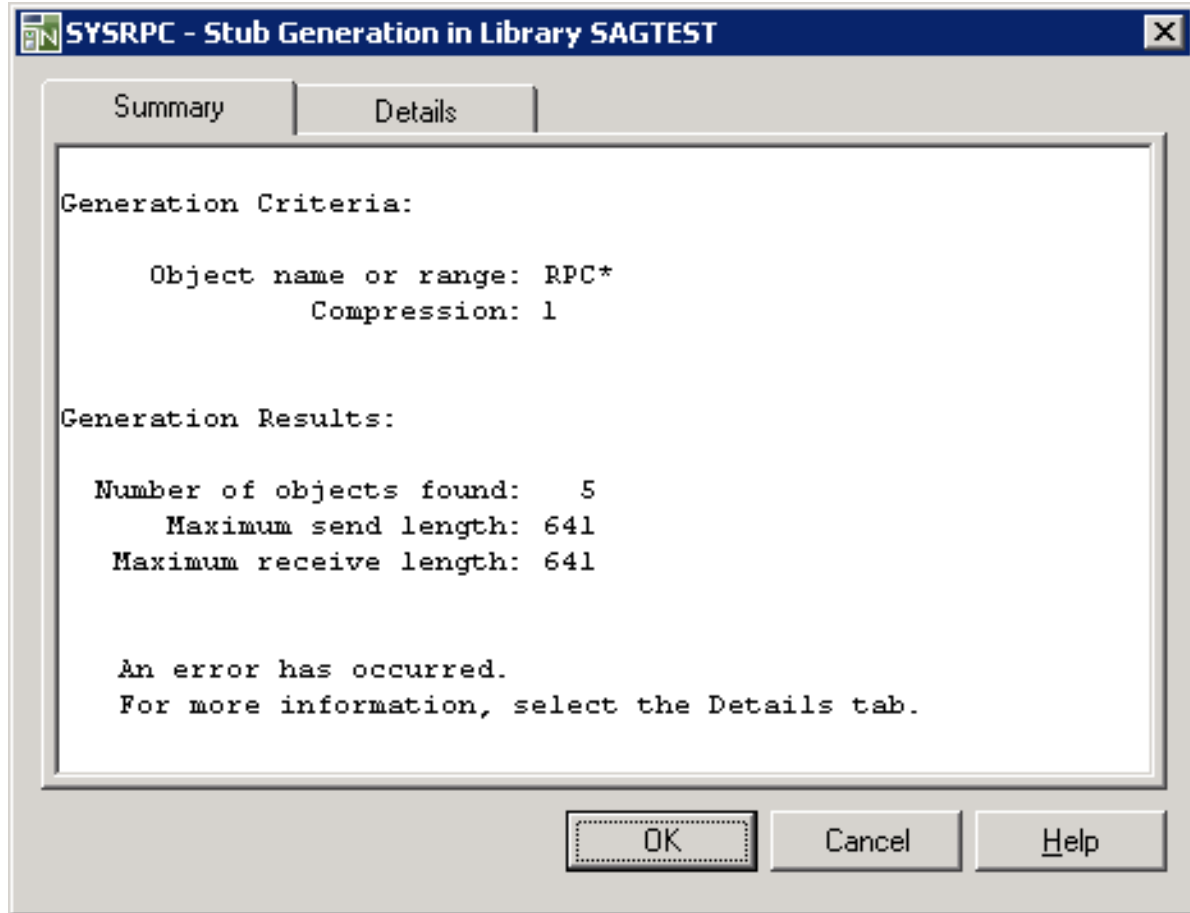
DBID, FNR are non-modifiable text boxes that display the database ID (DBID), the file number (FNR) and the type of Natural file (FNAT = system, FUSER = user) for the source and target libraries entered.

From the **Compression** drop-down list box, select compression type **0, 1** or **2** (default is **1**); see *Using Compression* described in *Operating a Natural RPC Environment* in the *Natural Remote Procedure Call (RPC)* documentation.

- 3 Choose **OK**.

The **SYSRPC - Interface Object Generation in Library** window appears for the library specified (here: **SAGTEST**) with the tabbed pages **Summary** and **Details**.

The **Summary** page contains a report that indicates the send and receive length requirements of the subprograms (objects) selected as shown in the following example:



The report is organized in two sections, which contain the following information:

■ **Generation Criteria:**

The criteria based on which the interface objects were generated: a single object name or a range of names (here: RPC*) and the compression (here: 1).

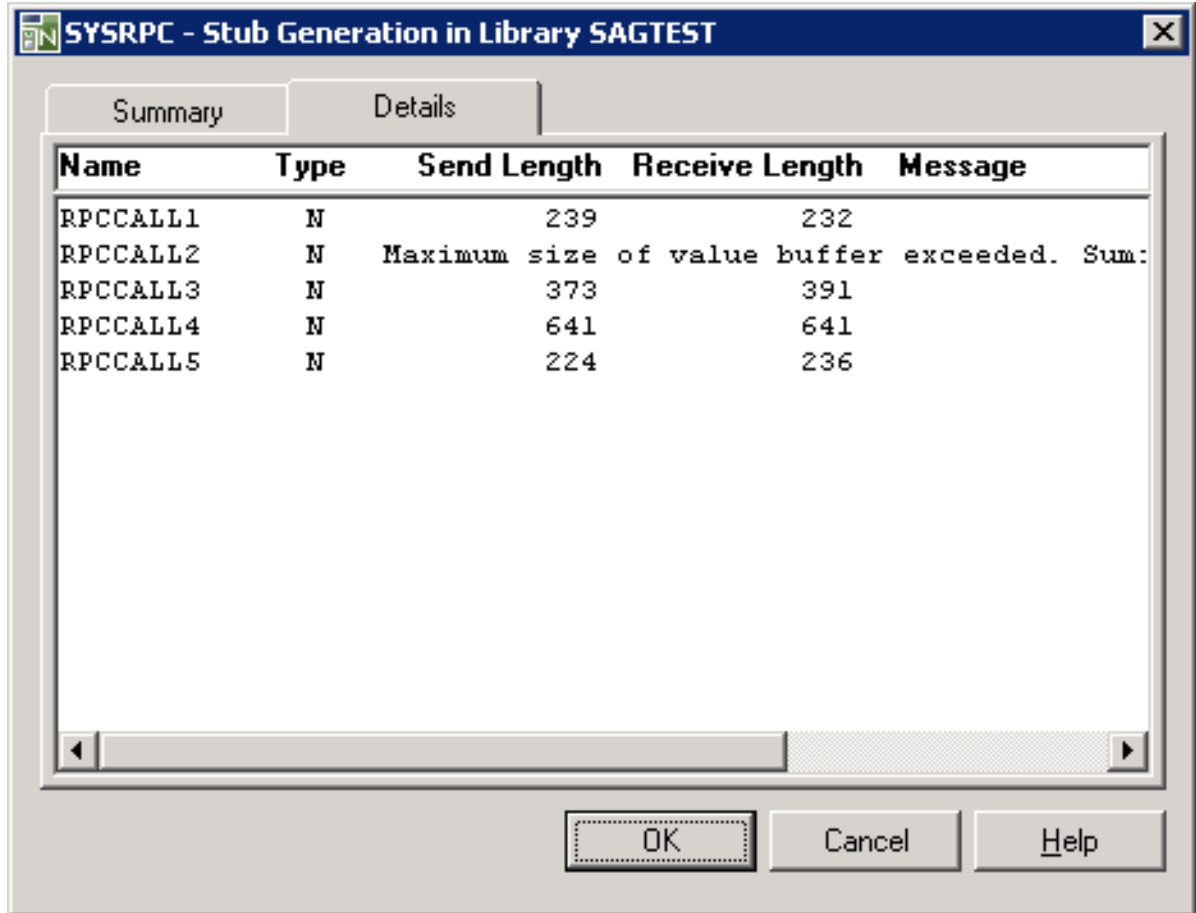
■ **Generation Results:**

The number of objects selected for the interface object generation.

The maximum buffer sizes all generated interface objects require for sending and receiving data from the client.

If the generation of an interface object fails, a message at the bottom of the page indicates this error.

The **Details** page contains a list of all generated interface objects similar to the example shown below:



The list is sorted in alphabetical order of object names (**Name** column) and contains information on the following:

- the type of object from which the interface object was generated (here: N for subprogram) in the **Type** column,
- the buffer sizes each object requires for sending (**Send Length**) and receiving (**Receive Length**) data from the client,
- and a possible comment on each generation of an interface object in the **Message** column.

If the generation of the interface object fails, an error message is displayed next to the objects affected (here: RPCCALL2).

Using the SYSRPC SGMASS Command

You can enter the command `SYSRPC SGMASS` in the Command line for generating interface objects online.

The report produced by the command corresponds to the report described for the **Interface Object Mass Generation** function.

-

The syntax that applies to `SYSRPC SGMASS` is illustrated in the diagram below:

```
SYSRPC SGMASS [name] [compression]
```

The syntactical items *name* and *compression* are explained in the section *Name Specification and Compression*.


Name Specification and Compression

You can specify the objects (subprograms) to be selected for interface object generation and the type of compression to be used:

- Name
- Compression

Name

You can specify an object name or a range of names. The specification of an object name or a range of names is optional.

 **Caution:** If you do not specify an object name or a range of names, with few exceptions (see below), all subprograms in the current library will be converted to interface objects.

Valid name specifications are described below where *value* is any combination of one or more alphanumeric characters:

Input	Objects Selected
*	All subprograms. This is the default setting.
<i>value</i>	A subprogram with a name equal to <i>value</i> .
<i>value</i> *	All subprograms with names that start with <i>value</i> .
<i>value</i> <	All subprograms with names less than or equal to <i>value</i> .
<i>value</i> >	All subprograms with names greater than or equal to <i>value</i> .

Exceptions to Names

In the Natural system library SYSRPC, SYSRPC SGMASS exempts from interface object generation all subprograms with names that start with any of the following prefixes: RDS, RPC, NAT, NAD or NSC.

In user libraries, SYSRPC SGMASS exempts from interface object generation the subprogram NATCLTGS.

Compression

You can specify any of the following compression types: 0, 1, 2. The specification of compression is optional. The default type used for interface object generation is 1.

See also *Using Compression* described in *Operating a Natural RPC Environment* in the *Natural Remote Procedure Call (RPC)* documentation.

76

Generating Interface Objects or PDAs from IDL Files

- Building a Selection List 526

An IDL (Interface Definition Language) file contains definitions of the interface between client and server.

The **Interface Object Generation from IDL** function provides the option to generate interface objects and/or parameter data areas (PDAs) from EntireX IDL files.

You can generate interface objects from IDL files by using either the **Interface Object Generation from IDL** function or the command `SYSRPC SGIDL`.

The following section provides instructions for using the **Interface Object Generation from IDL** function.

▶ **To generate interface objects or PDAs from an IDL file**

- 1 In the **SYSRPC - Remote Procedure Call** window, from the **Tools** menu, choose **Interface Object Generation from IDL**.

Or:

In the **SYSRPC - Remote Procedure Call** window, press `CTRL+F7`.

Or:

In the Command line, enter `SYSRPC SGIDL`.

The **Input for Interface Object Generation from IDL File** dialog box appears.

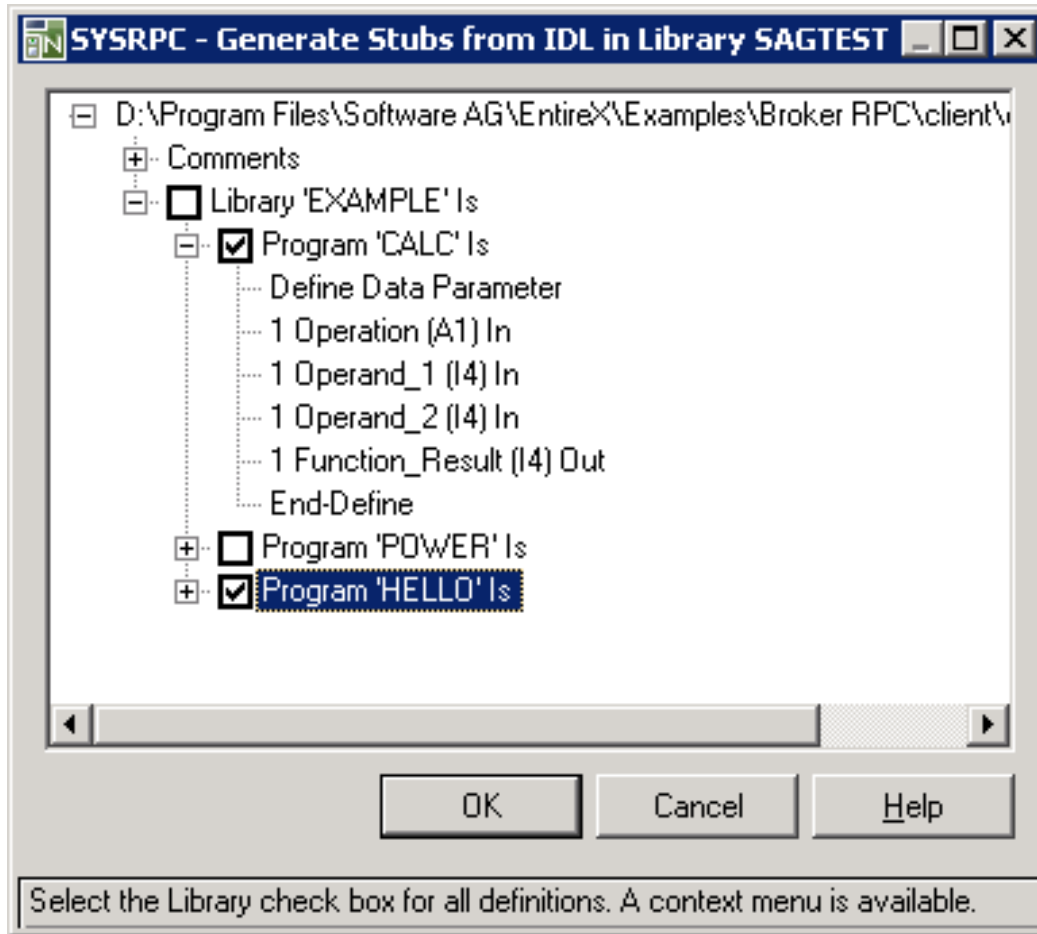
- 2 In the **Path or file name** text box, enter the complete path to the folder or the file that contains the IDL definitions or choose **Browse** to select a file from a folder.

If required, in the **Library** text box, enter the name of the library into which you want to generate the interface object. The text box is preset to the name of the current library.

DBID, FNR is a non-modifiable text box that displays the database ID (DBID), the file number (FNR) and the type of Natural file (FNAT = system, FUSER = user) for the specified library.

- 3 Choose **OK**.

The **Generate Interface Objects from IDL** window similar to the example below appears for the specified library:



All definitions contained in the selected IDL file are displayed in a tree view where the **Library** node indicates an EntireX RPC IDL library and each set of IDL definitions is contained in a **Program** node. The **Comments** node contains commentary text such as a description of the definitions contained in an IDL file.

- 4 Select the node(s) that contain the sets of IDL definitions from which you want to generate the interface object(s) or PDA(s):
 - **For interface objects:**
Select the check box of a **Library** node to generate interface objects from all **Program** nodes contained in that **Library** node.

Or:

Select the check boxes of individual **Program** nodes to generate interface objects from the specified **Program** nodes only.

You can change the name of an interface object by using a selection list as described in [Building a Selection List](#). Otherwise, the interface object will be named after the corresponding **Program** node.

- **For PDAs:**

Select the **Library** node(s) or the required **Program** node(s) and choose **Mark for parameter data area** from the context menu.

A list appears that displays the set(s) of IDL definitions for each node selected as shown in the **example** of a selection list below. The check boxes of all IDL definitions are selected for PDA generation. You can deselect a PDA by clearing the check box next to it in either the tree view or the selection list as described in *Building a Selection List*.

You can change the name of a PDA as described in *Building a Selection List*. Otherwise, the PDA will be named after the corresponding **Program** node where the following conversion rules apply:

- For a node name of 7 characters, an A is added to the PDA name. For example, node name PDATEST1 is converted to PDA name PDATESTA.
- For a node name of 6 characters or less, an A is added to the PDA name in the 8th position and each empty position between node name and A is filled with a dash (-). For example, node name TEST1 is converted to PDA name TEST1--A.

5 Choose **OK**.

For each item selected, a message box appears confirming successful generation of the interface object or the PDA.

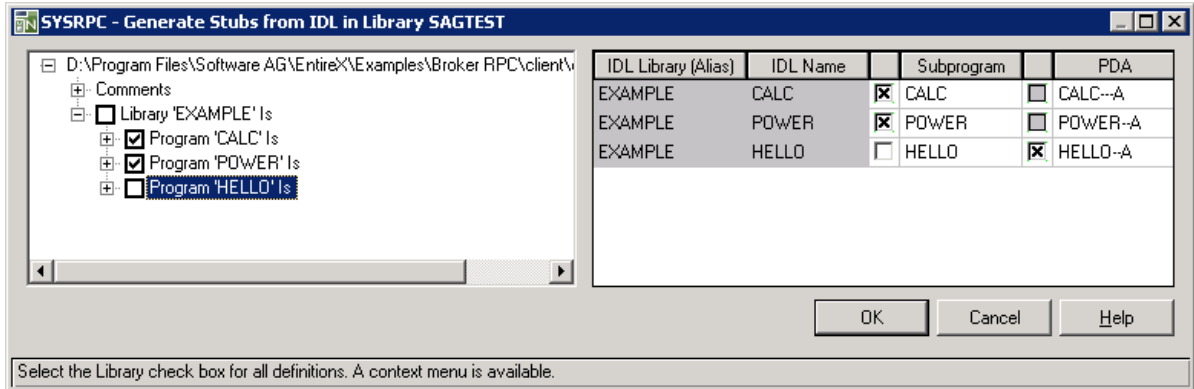
Building a Selection List

You can use a selection list to change the name of an interface object or a PDA, or deselect or select the nodes from which to generate an interface object and/or a PDA.

▶ **To build a selection list**

- 1 In the **Generate Interface Objects from IDL** window, select the check box of a **Library** node or the check box(es) of **Program** node(s) and choose **Build selection list to rename or mark object types** from the context menu.

A selection list appears that displays the selected **Program** nodes as shown in the example below:



The selection list displays all nodes selected in the tree view where the **IDL Library (Alias)** column corresponds to the **Library** node and the **IDL Name** column corresponds to the **Program** node. In the example above, the **Program** nodes `CALC` and `POWER` are selected for interface object generation and the **Program** node `HELLO` is selected for PDA generation.

- 2 In the **Subprogram** and/or **PDA** columns, select the check boxes of the sets of IDL definitions from which you want to generate interface objects and/or PDAs.

You can deselect an item by clearing a check box.

- 3 If you want to change the name of an interface object or a PDA, select the required table cell and replace the current entry.
- 4 If you want to remove one or more table rows, select the required rows and choose **Clear Row(s)** from the context menu.
- 5 If you want to close the selection list, choose **Close Table** from the context menu.

77

Calculating Size Requirements

- Using the Function Interface Object Mass Calculation 530
- Using the SYSRPC CSMASS Command 534
- Name Specification and Compression 534

You can calculate the buffer size (in bytes) stubless RPC calls require for sending data from the client to the server or vice versa. The indication of the size helps you configure the middleware layer used; for example, the Broker attribute file when EntireX Broker is used. If desired, you can also perform size calculations for interface objects, even though sizes are already calculated when the interface objects are generated.

Size calculations are performed with either the **Interface Object Mass Calculation** function or the direct command `SYSRPC CSMASS` whereby the direct command can be used in online or batch mode. Either method will invoke a window that indicates the send and receive lengths required by the specified subprogram(s).

Using the Function Interface Object Mass Calculation

This section provides instructions for calculating size requirements by using the function **Interface Object Mass Calculation**.

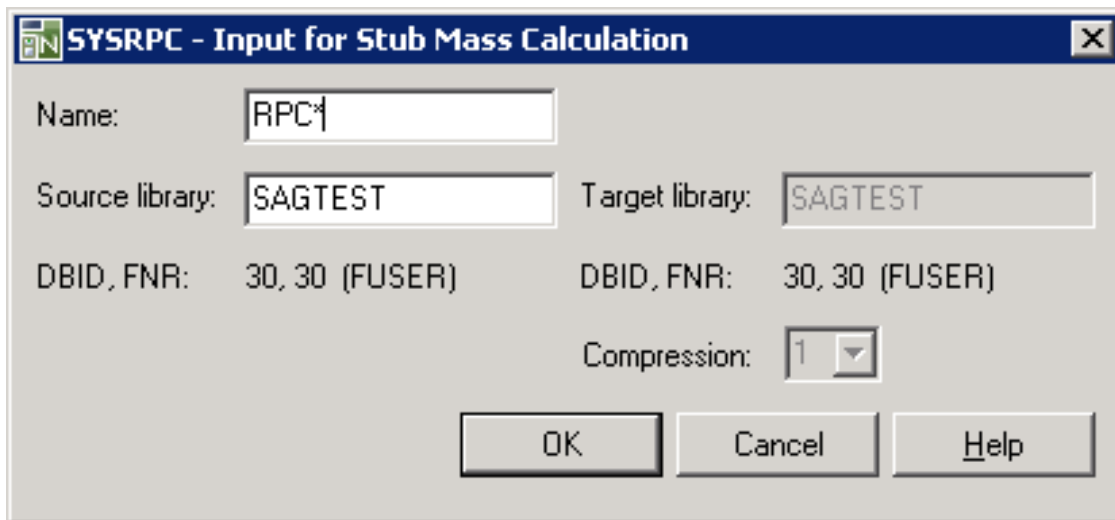
► **To perform Interface Object Mass Calculation**

- 1 In the **SYSRPC - Remote Procedure Call** window, from the **Tools** menu, choose **Interface Object Mass Calculation**.

Or:

In the **SYSRPC - Remote Procedure Call** window, press `CTRL+F5`.

An **SYSRPC - Input for Interface Object Mass Calculation** dialog box similar to the example below appears:



- 2 In the **Name** text box, enter the name of the subprogram for which to calculate the size or specify a range of names. The text box is preset to an asterisk (*) for all subprograms. For valid names, see *Name* in *Specification and Compression*.

If required, in the **Source Library** text box, enter the name of the library that contains the subprograms specified. The text box is preset to the name of the current library.

The **Target Library** text box does not apply to the mass calculation function.

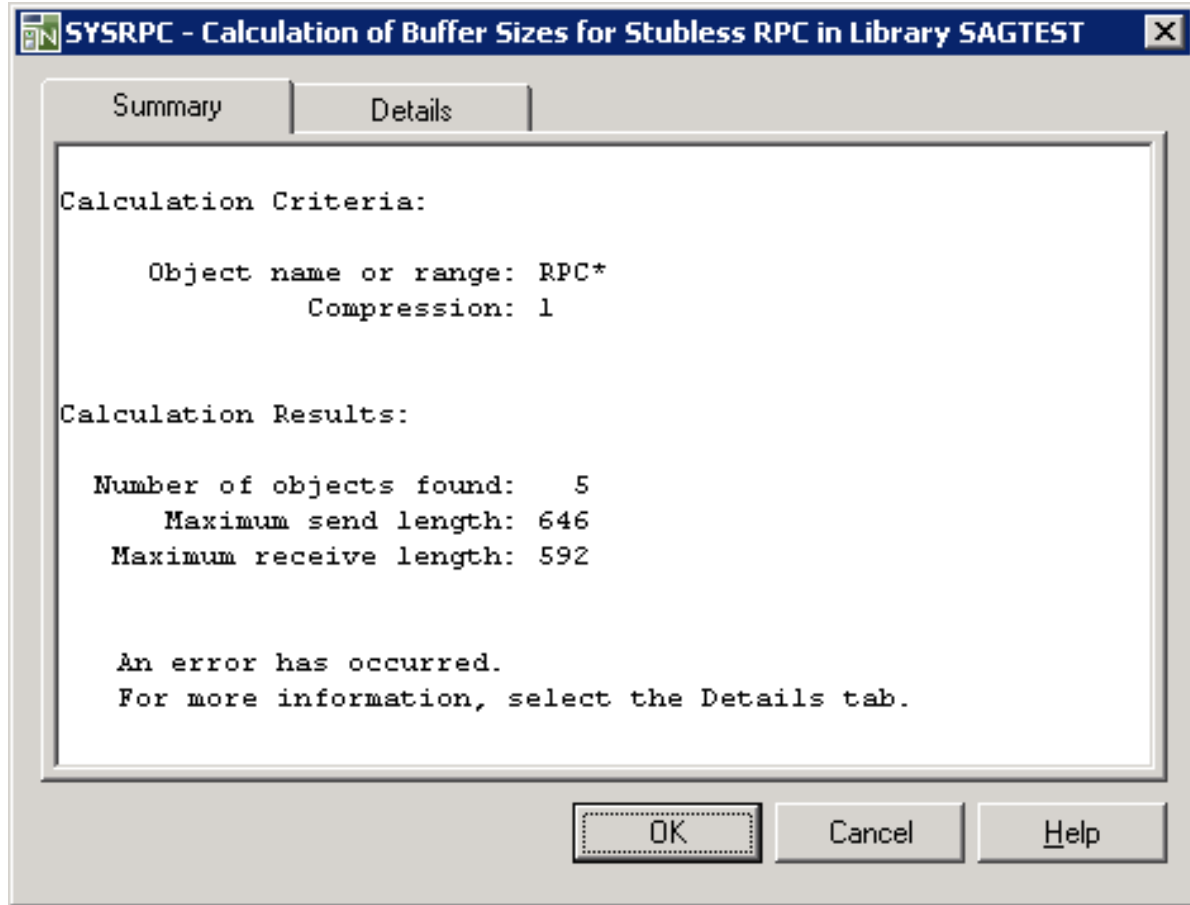
DBID, **FNR** are read-only text boxes that display the database ID (DBID), the file number (FNR) and the type of Natural file (FNAT = system, FUSER = user) for the source and target libraries entered.

From the **Compression** drop-down list box, select compression type **0**, **1** or **2** (default is **1**); see *Using Compression* described in *Operating a Natural RPC Environment* in the *Natural Remote Procedure Call (RPC)* documentation.

- 3 Choose **OK**.

The **SYSRPC - Calculation of Buffer Sizes for Stubless RPC in Library** window appears for the library specified (here: **SAGTEST**) with the tabbed pages **Summary** and **Details**.

The **Summary** page contains a report that indicates the send and receive length requirements of the subprograms (objects) selected as shown in the following example:



The report is organized in two sections, which contain the following information:

■ **Calculation Criteria:**

The criteria based on which the calculation was made: a single object name or a range of names (here: RPC*) and the compression (here: 1).

■ **Calculation Results:**

The number of objects selected for the size calculation. The maximum buffer sizes all selected objects require for sending and receiving data from the client.

If the size calculation fails, a message at the bottom of the page indicates this error.

The **Details** page contains a list of all objects selected for the calculation similar to the example shown below:

Name	Type	Send Length	Receive Length	Message
RPCCALL1	N	239	232	
RPCCALL2	N	Maximum size of value buffer exceeded. Sum:		
RPCCALL3	N	368	246	
RPCCALL4	N	646	592	
RPCCALL5	N	224	215	

The list is sorted in alphabetical order of object names (**Name** column) and contains information on the following:

- the type of object used for the calculation (here: N for type subprogram) in the **Type** column,
- the buffer sizes each object requires for sending (**Send Length**) and receiving (**Receive Length**) data from the client,
- and a possible comment on each object calculation in the **Message** column.

If the size calculation fails, an error message is displayed next to the objects affected (here: RPCCALL2).

Using the SYSRPC CSMASS Command

You can enter the command `SYSRPC CSMASS` in the Command line for calculating size requirements online.

The report produced by the command corresponds to the report described for the **Interface Object Mass Calculation** function.

The syntax that applies to the command `SYSRPC CSMASS` is illustrated in the diagram below:

```
SYSRPC CSMASS [name] [compression]
```

The syntactical items *name* and *compression* are explained in the section *Name Specification and Compression*.

Name Specification and Compression

You can specify the objects (subprograms) to be selected for size calculation and the type of compression to be used:

- Name
- Compression

Name

You can specify an object name or a range of names. If you do not specify a name or a range of names, the size of all subprograms contained in the current library will be calculated.

Valid name specifications are described below where *value* is any combination of one or more alphanumeric characters:

Input	Objects Selected
*	All subprograms. This is the default setting.
<i>value</i>	A subprogram with a name equal to <i>value</i> .
<i>value</i> *	All subprograms with names that start with <i>value</i> .
<i>value</i> <	All subprograms with names less than or equal to <i>value</i>
<i>value</i> >	All subprograms with names greater than or equal to <i>value</i> .

Compression

You can specify any of the following compression types: 0, 1, 2. The specification of compression is optional. The default type used for interface object generation is 1.

See also *Using Compression* described in *Operating a Natural RPC Environment in the Natural Remote Procedure Call (RPC)* documentation.

78 Server Command Execution

- Message Display Window 538
- Pinging a Server 539
- Terminating a Server 540

The SYSRPC utility provides the server execution commands ping and terminate. They are used to control active servers that have been defined in the service directory. The ping command sends an internal message to the server to verify a server connection. Terminate either sends an internal message to the server requesting termination of a single server task, or issues a command to EntireX Broker requesting termination of all server tasks associated with an EntireX Broker service.

The server execution commands reference the service directory in the library that is defined with the `RPCSDIR` profile parameter (see the *Parameter Reference* documentation). If `RPCSDIR` is not set (this is the default), the library where you are currently logged on is used. The name of the library is indicated in the root node of the service directory tree view.

In addition, when a ping or a terminate command is issued, the **Message Display** window appears that displays the following:

- Physical node and server names defined for a logical service, or
- Physical node names defined for logical nodes.

Message Display Window

The ping and terminate commands invoke the **Message Display** window. This window provides the columns **Message**, **Node** and **Server** that display the message returned from the server and the physical name of the node and the server selected (see the instructions below).

Physical names are also displayed for logical specifications: a logical node name is resolved into the physical node name and a logical service is resolved into the physical node and server name as defined in the service directory.

However, logical specifications will *not* be resolved if the following applies:

- **Logical service:**
The pinged or terminated logical service is not active, or the physical node(s) and server(s) defined for the logical service have not been defined in the Location Transparency directory of EntireX Broker. In this case, the **Message Display** window displays a corresponding message from the server, the term `*LOCTRAN` in the **Node** column and the logical service name (as defined in the service directory) in the **Server** column.
- **Logical node:**
The pinged or terminated logical node is not active, or the physical node defined for the logical node has not been defined in the Location Transparency directory of EntireX Broker. In this case, the **Message Display** window displays a corresponding message from the server and the entry `LOGBROKER= node-name` (as defined in the service directory) in the **Node** column.

Pinging a Server

The following section provides instructions for pinging an RPC server by using menu functions.

For an alternative method of pinging an RPC server, see the Application Programming Interface USR2073N described in the *Natural Remote Procedure Call (RPC)* documentation.

▶ To ping a server

- In the **Service Directory** tree view, select a server or a node and choose **Ping** from the context menu.

Or:

Choose CTRL+F9.

Or:

Choose the following toolbar button:



Selecting an EntireX Broker node will ping all servers that belong to this node.

The **Message Display** window appears and displays the physical names of the node(s) and server(s) and the message returned from the server(s).

If the pinged server(s) are active, the server(s) return the message:

```
Server version on operating system
```

where *Server* denotes the type of server;

version denotes the version of the server;

operating system denotes on which operating system the server runs.

Example message: Natural RPC Server 6.3.1.0 on WNT-x86.

Terminating a Server

The SYSRPC utility provides two commands to terminate a server: **Terminate Server** and **Terminate EntireX Broker Service**.

Terminate Server terminates a single server task by sending an internal message to the server. If a server is associated with multiple server tasks (including replicas on mainframe platforms), you can either terminate each server task separately by using **Terminate Server**, or terminate all server tasks in one go by using the **Terminate EntireX Broker Service** command.

Terminate EntireX Broker Service terminates all server tasks associated with an EntireX Broker service by calling EntireX Broker's Command and Information Services (ETBCIS; for details, see the EntireX documentation). The term *service* here summarizes all server tasks that run with the same server name on the same or on different platforms.

The following section provides instructions for terminating a single server task or an EntireX Broker service by using menu functions.

For alternative methods of terminating servers, see *Using Application Programming Interface USR2073N* and *Using Application Programming Interface USR2075N in Terminating a Natural RPC Server* in the *Natural Remote Procedure Call (RPC)* documentation.

▶ To terminate a single server task

- 1 In the **Service Directory** tree view, select a server node and choose **Terminate Server** from the context menu.

The **Message Display** window appears with the name of the node and the server, and the message returned from the server.

If a server is terminated, the server returns the following message:

```
Terminating Server version on operating system
```

where *Server* denotes the type of server;
version denotes the version of the server;
operating system denotes on which operating system the server runs.

Example message: Terminating Natural RPC Server 6.3.1.0 on WNT-x86.

- 2 If the **Logon Option** is set in the service directory, logon data (user ID, password and library name) is sent to the server with the terminate command, as is usual for the CALLNAT. The **Security Token Data** window pops up and requests user ID and password if no Natural Security is installed on the client side and no logon data is set with the Application Programming Interface USR1071N for the current Natural session.

If LOGONRQ=ON (see also *Using Security* in the *Natural Remote Procedure Call (RPC)* documentation) has been set on the server side, logon data must be sent from the client with the terminate command.

If Natural Security is installed on the server, the logon data transferred must enable a logon to the Natural system library SYSRPC.

▶ **To terminate an EntireX Broker service**

- 1 In the **Service Directory** tree view, select a server node and choose **Terminate EntireX Broker Service** from the context menu.

Or:

Choose the following toolbar button:



The **SYSRPC - Terminating EntireX Broker Service** dialog box appears.

- 2 If required for the logon, enter the appropriate user ID and password for EntireX Broker.

If you want to terminate server tasks that are involved in a conversation, select the **Terminate immediately** check box to request immediate termination. If this check box is not selected (this is the default setting), all server tasks involved in a conversation remain operational.

If you do not want this dialog box to appear repeatedly during the current SYSRPC session, choose **Do not show this dialog again**.

- 3 Choose **OK** to terminate the EntireX Broker service.

The **Message Display** window appears, which displays the name of the node and the server and the number of server tasks terminated.

XVI

Tamino Server Extensions

79 Tamino Server Extensions

▪ Overview	546
▪ Developing a Tamino Server Extension	547
▪ Using Callbacks	553
▪ Deploying a Tamino Server Extension	553
▪ Installing a Tamino Server Extension	554
▪ Tamino Server Extension Example	554

Tamino allows you to develop, implement, administrate and execute Server Extensions. Tamino Server Extensions can be used to extend the query and mapping Tamino Server functionality by adding user-defined logic. For a description of the functionality available with Tamino Server Extensions, see the Tamino documentation.

In order to extend the Tamino functionality, you install Tamino Server Extension Packages in Tamino databases. These packages contain (among other data) Tamino Server Extension Objects based on COM or on Java. Methods of these objects can then be used to extend the query or mapping functionality of the Tamino Server.

Server Extension Objects based on COM can be implemented in Natural. Please check the Natural Release Notes for information about the Tamino version needed as a prerequisite.

Overview

This document focuses on the Natural-specific implementation details of Tamino Server Extensions and the Natural tools and techniques used in this process. Essential background information that should help you develop valid Server Extension Function code is contained in the Tamino documentation. You should read the corresponding chapters of the Tamino documentation carefully before starting to develop Tamino Server Extensions.

- To develop Natural-based Server Extensions, you use the Natural Class Builder. A Tamino Server Extension is developed as a NaturalX class that implements interfaces corresponding to a predefined structure. To support the implementation of these interfaces, certain predefined Natural modules are delivered with Natural.
- To deploy a Natural-based Tamino Server Extension in the target environment, you use the usual Natural deployment tools.
- To register your Tamino Server Extension, you use the Natural REGISTER command.

Once you have developed, installed and registered your Natural-based Tamino Server Extension using Natural development tools, you can assign it to a Tamino database and use it in a Tamino schema using the usual Tamino tools. For a detailed description of the usage of these tools, see the Tamino documentation.

- To select methods of your Natural-based Tamino Server Extension into a Server Extension Package and to create a package file, you use the SXS Analyzer.
- To install and administrate Server Extensions in a Tamino database, use the Server Extensions Administration as provided by the Tamino Manager.
- To assign Server Extension functions to a Tamino schema, use the Tamino Schema Editor.
- Server Extensions can be traced using the SXS Trace.

Developing a Tamino Server Extension

The following topics are covered.

- [Overview](#)
- [Set the Library SYSEXSXS as Steplib](#)
- [Create a New Library for Your Project](#)
- [Create a NaturalX Class](#)
- [Create the Object Data Area](#)
- [Edit the Object Data Area](#)
- [Link the Connection Interface](#)
- [Implement the Method Connect](#)
- [Add Server Extension Functions](#)
- [Save and Catalog the Class](#)

Overview

The Natural Class Builder is used to develop a Tamino Server Extension. A Natural-based Tamino Server Extension is a NaturalX class that implements interfaces corresponding to a predefined structure. To support the implementation of these interfaces, certain predefined Natural modules are delivered with Natural:

- An Interface Module (Copycode) containing the declaration of the Connection interface defined by Tamino.
- Parameter Data Areas containing parameter definitions for the different types of Server Extension functions.

Set the Library SYSEXSXS as Steplib

When implementing a Tamino Server Extension in Natural, you can use a number of predefined Natural modules contained in the sample library SYSEXSXS. This makes sure that your Tamino Server Extension conforms to the interface defined by Tamino. In order to use these modules in your Tamino Server Extension project, first define the library SYSEXSXS as steplib.

▶ **To define the library SYSEXSXS as steplib:**

- 1 Start the Natural Configuration Utility.
- 2 Select the Natural parameter file you are working with.
- 3 Choose **Edit > Find** to locate the parameter STEPLIB.
- 4 Enter SYSEXSXS into the list of steplibs.
- 5 Save the Natural parameter file.

- 6 Close the Natural Configuration Utility.
- 7 Restart Natural Studio.

Create a New Library for Your Project

It is recommended to put all Natural modules for one Tamino Server Extension into one Natural library. Therefore first create a new Natural library for your project.

▶ **To create a new library:**

- 1 Select **User Libraries** in the library workspace.
- 2 Select **New** in the context menu.
- 3 Choose a name for the library.

Create a NaturalX Class

A Tamino Server Extension is implemented as a NaturalX Class. Therefore create a new class.

▶ **To create a new class:**

- 1 Select your library in the library workspace.
- 2 Select **New Source > Class** in the context menu.
- 3 Choose a name for the class.
- 4 Select **Save** in the context menu.
- 5 Choose a name for the class module.

Create the Object Data Area

An Object Data Area in a NaturalX class is used to contain variables that shall keep their value during the lifetime of an instance of the Tamino Server Extension.

▶ **To create an Object Data Area and link it to your class:**

- 1 Select your class in the library workspace.
- 2 Select **New > Object Data Area** in the context menu.
- 3 Choose a name for the Object Data Area.

Edit the Object Data Area

The Object Data Area of a Tamino Server Extension should at least contain an object handle to hold a reference to the Tamino callback object during the lifetime of an instance of your Tamino Server Extension. The Tamino callback object is passed by Tamino to the Server Extension when it loads the Extension. You can use the methods of the callback object to call Tamino functionality from inside your Tamino Server Extension functions.

▶ To add an object handle for the callback object to your Object Data Area:

- 1 Double-click on the Object Data Area in the library workspace to edit it.
- 2 In the Data Area Editor select **Insert > Handle**.
- 3 Choose a name for the object handle (e. g. CALLBACK).
- 4 Select "Object" as handle type.
- 5 Click the **Add** button.
- 6 Click the **Quit** button.
- 7 If you so wish, add further variables to the Object Data Area as required by your Server Extension.
- 8 Close the Data Area Editor and save the Object Data Area.

Link the Connection Interface

A Tamino Server Extension must implement the Connection interface ISXSConn. This interface is declared in the interface module ICONN-C in the library SYSEXXSXS. To be able to locate the interface module, you must have defined the library **SYSEXXSXS** as steplib. Link this interface module to your class.

▶ To link the interface module to your class:

- 1 Select your class in the library workspace.
- 2 Select **Link > Interface Module** in the context menu.
- 3 Select the interface module ICONN-C in library SYSEXXSXS.

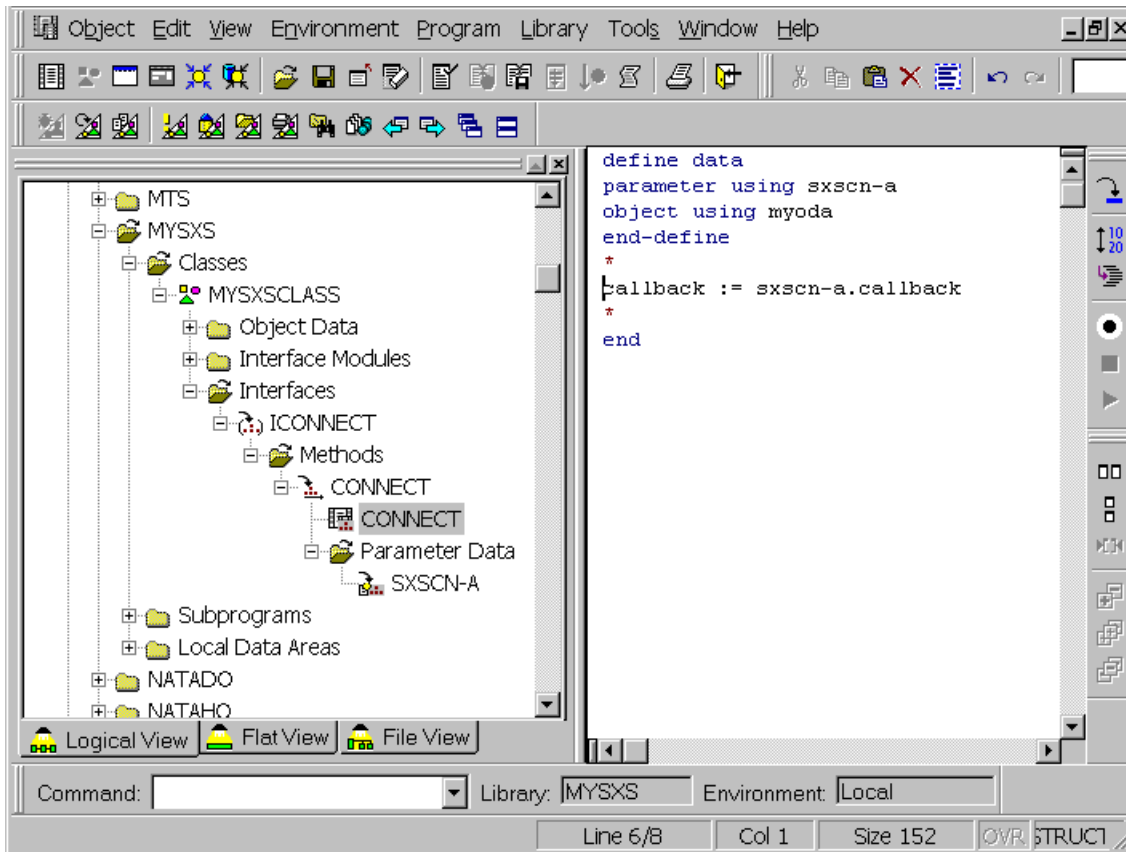
Implement the Method Connect

The method Connect of the ISXSConn should store a handle to the callback object in the Object Data Area. This allows the Server Extension Functions to access Tamino functionality.

▶ **To implement the method Connect:**

- 1 Fully expand the branch "Interfaces" of your class in the library workspace.
- 2 Select the subprogram node "CONNECT". This is the default name for the subprogram that will implement the method Connect.
- 3 If you so wish, rename the subprogram to a name of your choice by selecting Rename in the context menu.
- 4 Double-click on the subprogram node to create and edit the method subprogram.

The implementation of the Connect method must include both the Object Data Area of the class and the Parameter Data Area of the method Connect. The body of the method must assign the Callback object handle from the Parameter Data Area to the corresponding object handle defined in the Object Data Area, as shown in the example below.



You can also add further initialization code to the method `Connect` as required by your Server Extension.

Add Server Extension Functions

You can now start to add your own Server Extension Functions to the class. First you will create a new interface for your class to contain the Server Extension Functions. Then you will add the individual functions to that interface.

▶ **To create a new interface:**

- 1 Select the node "Interfaces" of your class in the library workspace.
- 2 Select **New** in the context menu.
- 3 Choose a name for the interface.


▶ **To create a new Server Extension Function :**

- 1 Select your interface in the library workspace.
- 2 Select **New > Method** in the context menu.
- 3 Choose a name for the method.
- 4 Select the method
- 5 Select **Link > Parameter Data Area**.

Tamino distinguishes different types of Server Extension Functions. Depending on the type of Server Extension Function to be implemented, choose the corresponding Parameter Data Area:

Function	Data
Map In function:	Parameter Data Area SXSMI-A
Map Out function:	Parameter Data Area SXSMO-A
On Delete function:	Parameter Data Area SXSDL-A
Event function:	Parameter Data Area SXSEV-A
Query function:	Parameter Data Area SXSQU-A

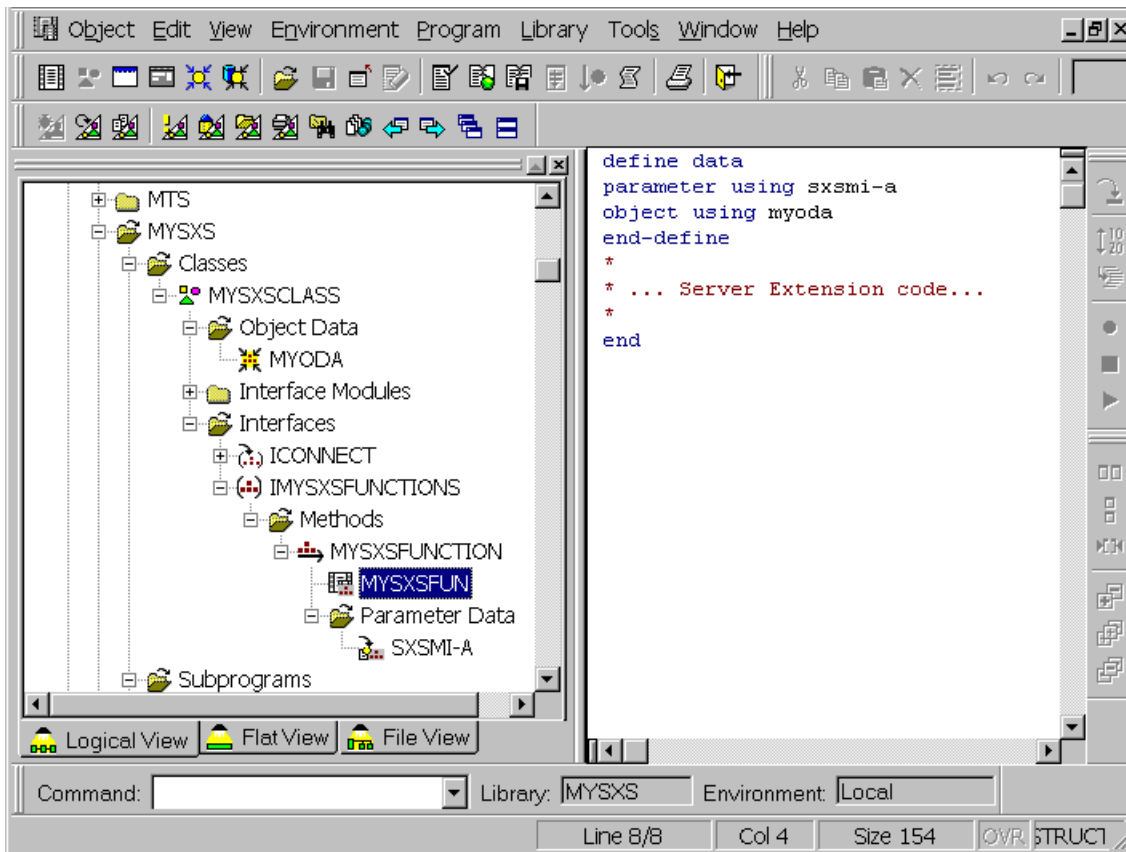
These Parameter Data Areas are contained in the library SYSEXSXS. To be able to locate the Parameter Data Areas, you must have defined the library **SYSEXSXS** as `steplib`.

 **Note:** The Parameter Data Area SXSQU-A is just an example. Query functions can have user-defined parameters. Thus it is not possible to define a common Parameter Data Area for them. If you want to create a query function, please consult the Tamino documentation and check which parameter types are allowed in query functions. Then create your own Parameter Data Area in your project library that matches the needs of your query function.

► **To implement the Server Extension Function:**

- 1 Select the subprogram node that represents the method implementation.
- 2 If you so wish, rename the subprogram to a name of your choice by selecting Rename in the context menu.
- 3 Double-click on the subprogram node to create and edit the method subprogram.

The implementation of the method must include both, the Object Data Area of the class and the Parameter Data Area assigned to the method. The body of the method contains the coding specific to the Server Extension Function.



- Close the Program Editor and save the subprogram.

To add further Server Extension functions, repeat "**To create a new Server Extension Function**".

Save and Catalog the Class

Finally save the class and recatalog the whole project library.

1. Select your class in the library workspace.
2. Select **Save** in the context menu.
3. Select your library in the library workspace
4. Select **Cat All** in the context menu.

Using Callbacks

Tamino callbacks are interfaces of the Tamino Server that can be used in a Server Extension Function. They enable access to both the various databases that can be administrated by the Tamino Server and system information available in the running Tamino Server. To use a Callback function from within a Natural-based Tamino Server Extension, do the following:

- Consult the Tamino documentation to find out the parameters of the callback function you wish to use.
- In the Object Data Area of the NaturalX class that implements your Tamino Server Extension you have defined an object handle as a reference to the callback object. Send a corresponding method call to this object handle.

Deploying a Tamino Server Extension

Having developed the NaturalX class that implements your Tamino Server Extension, deploy it into the target environment with the usual Natural deployment tools, for instance the Object Handler. A Tamino Server Extension must be installed on the same machine where the Tamino server is running.

Register the class in the target environment under an arbitrary COMSERVERID. If necessary, see the NaturalX documentation on COMSERVERIDs and the different options of the REGISTER command.

Installing a Tamino Server Extension

Installing a Natural-based Tamino Server Extension is the same procedure as installing any COM-based Tamino Server Extension:

1. Use the SXS Analyzer to create a Server Extension Package. In the SXS Analyzer, select as the file to analyze the type library of your NaturalX class. As with any NaturalX class, the type library is located in the directory `<natdir>\<natvers>\Natural\Etc\<comserverid>\<classname>\<version>`, where `<natdir>` is the Natural installation directory, `<natvers>` the installed Natural version, `<comserverid>` the COMSERVERID under which the class was registered, `<classname>` the class name and `<version>` the class version (currently always "v1"). Proceed as usual with the SXS Analyzer to create a Server Extension Package.
2. Install the Server Extension Package into a Tamino database using the Server Extensions Administration as provided in the Tamino Manager.
3. Afterwards you can use the Tamino Server Extension as usual, for instance in XQuery functions or to map XML sub-documents in the Tamino Schema Editor.

Tamino Server Extension Example

The Natural sample library SYSEXSXS contains a simple Natural-based Tamino Server Extension as a programming example. The sample works on the Employees schema and maps parts of the schema, the salary data, on Server Extension Functions. These functions work as follows:

- Whenever an Employee element is inserted into the schema, the salary data of this Employee is passed to the Map In function SXSStoreToFile. This function creates a file in the TEMP directory and stores the data into the file.
- Whenever an Employee element is read from the schema, the salary data of this Employee is requested from the Map Out function SXSRetrieveFromFile. This function opens the corresponding file in the TEMP directory and reads the data from the file.
- Whenever an Employee element is deleted from the schema, the On Delete function SXSDeleteFile is called. This function deletes the corresponding file in the TEMP directory.

The sample Employees schema and some sample data are contained in the RES subdirectory of the sample library SYSEXSXS.

The sample can be driven comfortably using the Tamino Demo application contained in the sample library SYSEXINS. To run the sample Tamino Server Extension, proceed as follows:

1. Install the sample Tamino Server Extension in a Tamino database as described above in "[Deploying a Tamino Server Extension](#)" and "[Installing a Tamino Server Extension](#)".

2. Start the dialog MENU in the library SYSEXINS.
3. Set the Tamino URL to your Tamino database.
4. Enter "NATSXSDemoData" as collection name.
5. Execute "Define Tamino Schema" and select the schema "EmployeeSXSSchema.tsd" from the RES subdirectory of the sample library SYSEXXSXS.
6. Execute "Load Tamino Data" and select the data file "EmployeeSXSDData.xml" from the RES subdirectory of the sample library SYSEXXSXS.
7. Execute the sample queries and update records. Note that part of the Employee data (the salary data) is now handled by the Tamino Server Extension.
8. Delete the Employee data.
9. Delete the Employee schema.

