Natural

Tools and Utilities

Version 9.1.3

October 2021

ADABAS & NATURAL

This document applies to Natural Version 9.1.3 and all subsequent releases.

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

Document ID: NATWIN-NNATUTILITIES-913-20211014

Table of Contents

Preface

This document describes the purpose and use of the tools and utilities provided by Natural.

Utility Activation	Describes how Natural invokes a tool or utility.
Component Browser	Views ActiveX components that are available for developing NaturalX applications.
Data Browser	Generates, prints and stores data reports from Natural DDMs (data definition modules).
FTOUCH	Makes a downloaded object executable by Natural.
INPL	Loads or scans Natural objects and shared resources supplied by Software AG.
Natural Profiler	Monitors the internal process flow of a Natural application and analyzes the performance and code coverage of the application.
Object Handler	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	Locates Application Programming Interfaces (APIs) provided by Natural add-on products.
SYSCP	Provides code page information.
SYSERR	Creates application-specific messages. In addition, it can be used to modify the texts of the existing Natural system messages (not recommended).
SYSEXT	Locates Natural Application Programming Interfaces (APIs).
SYSEXV	Provides examples of the new features of the current Natural versions.
SYSLVERS	List objects which have been cataloged within a selected Natural version range.
SYSMAIN	Performs object operations in Natural such as copy, move, delete and import.
SYSNCP	Defines command-driven navigation systems for Natural applications.
SYSPCI	Configures and initializes a product after a first-time installation.
SYSRPC	Establishes and maintains Natural RPC (Remote Procedure Call) environments.
Tamino Server Extensions	Extends the query and mapping Tamino Server functionality by adding user-defined logic.

1 About this Documentation

Document Conventions

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

Online Information and Support

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at **https://documentation.softwareag.com**.

Software AG Empower Product Support Website

If you do not yet have an account for Empower, send an email to empower@softwareag.com with your name, company, and company email address and request an account.

Once you have an account, you can open Support Incidents online via the eService section of Empower at **https://empower.softwareag.com/**.

You can find product information on the Software AG Empower Product Support website at **https://empower.softwareag.com**.

To submit feature/enhancement requests, get information about product availability, and download products, go to **Products**.

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the **Knowledge Center**.

If you have any questions, you can find a local or toll-free number for your country in our Global Support Contact Directory at **https://empower.softwareag.com/public_directory.aspx** and give us a call.

Software AG Tech Community

You can find documentation and other technical information on the Software AG Tech Community website at **https://techcommunity.softwareag.com**. You can:

- Access product documentation, if you have Tech Community credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

Data Protection

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

I Utility Activation

2 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

3 Component Browser

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.

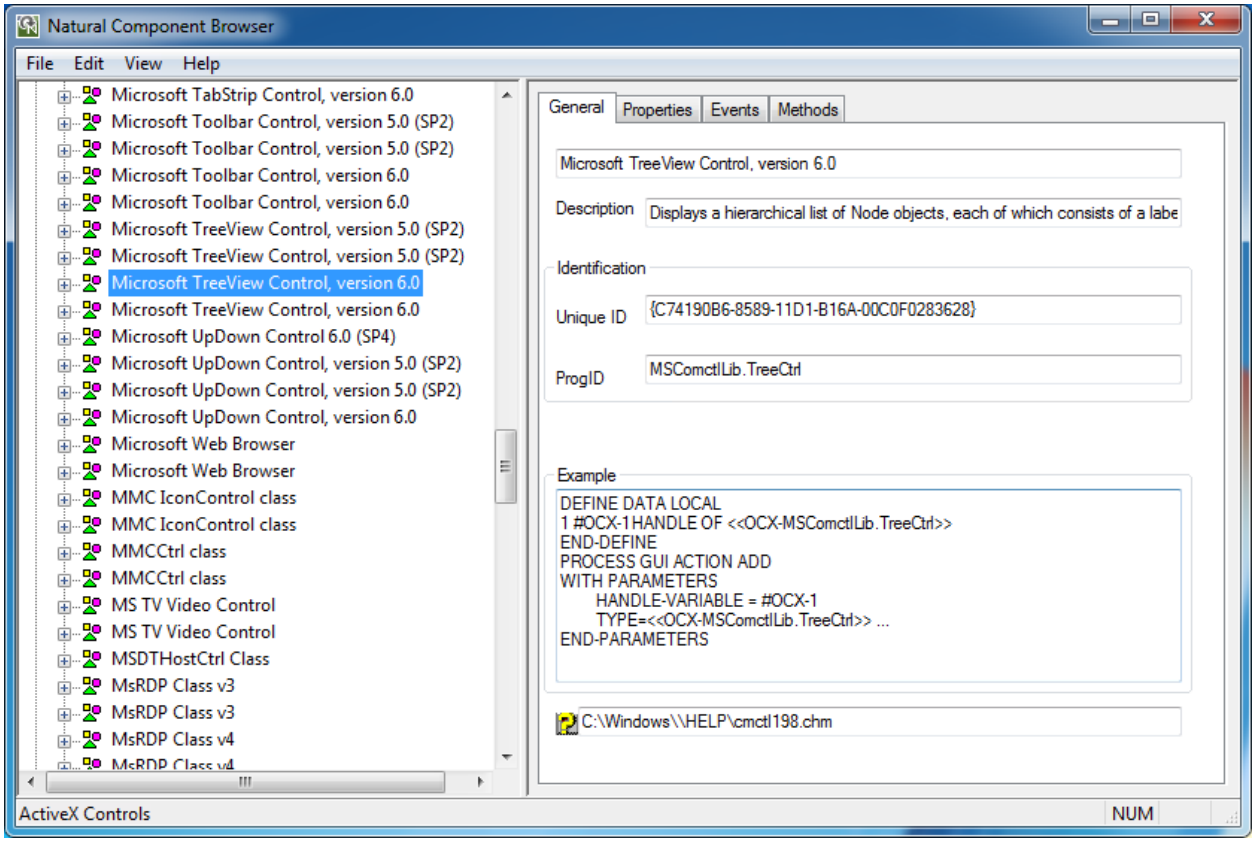
This section describes the tabbed pages that may be displayed in the data view.

- 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.

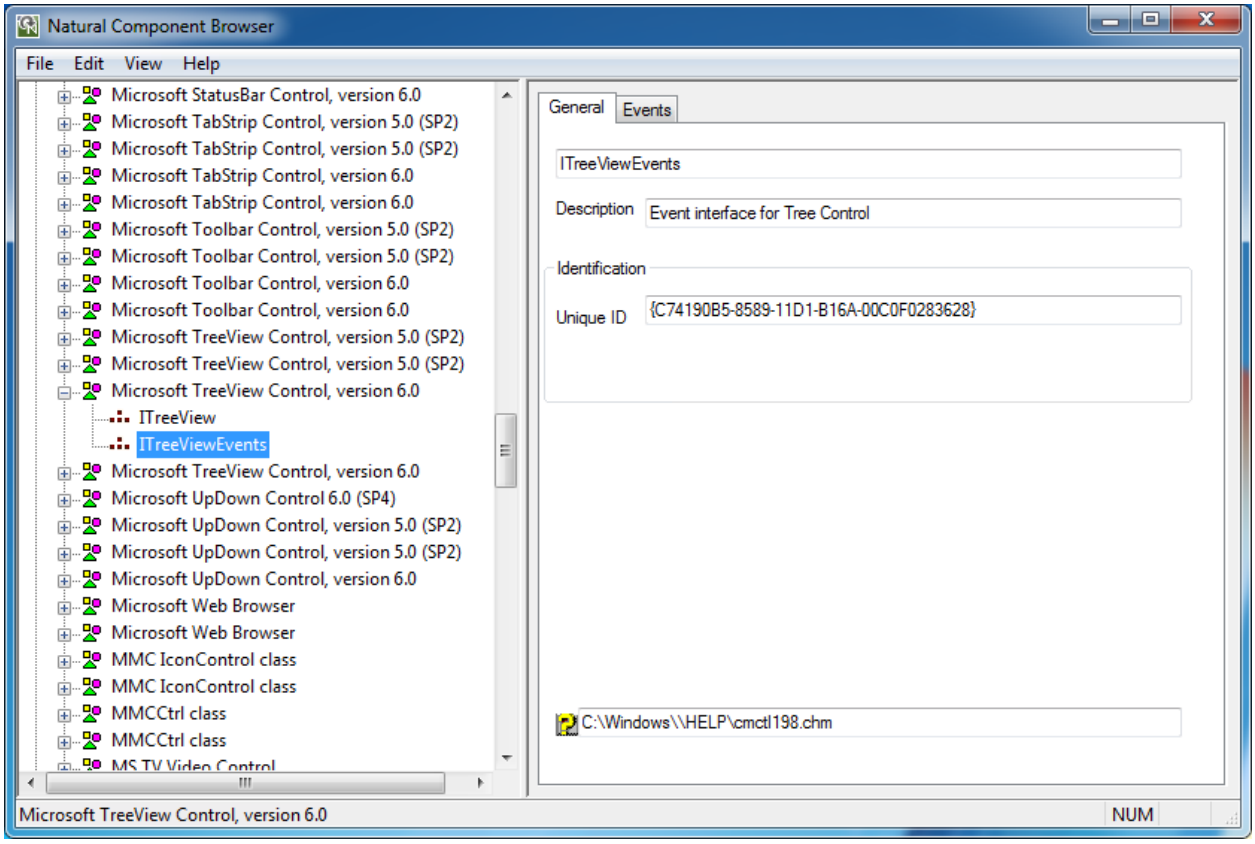
- General
- Properties
- Methods

- Events

General

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

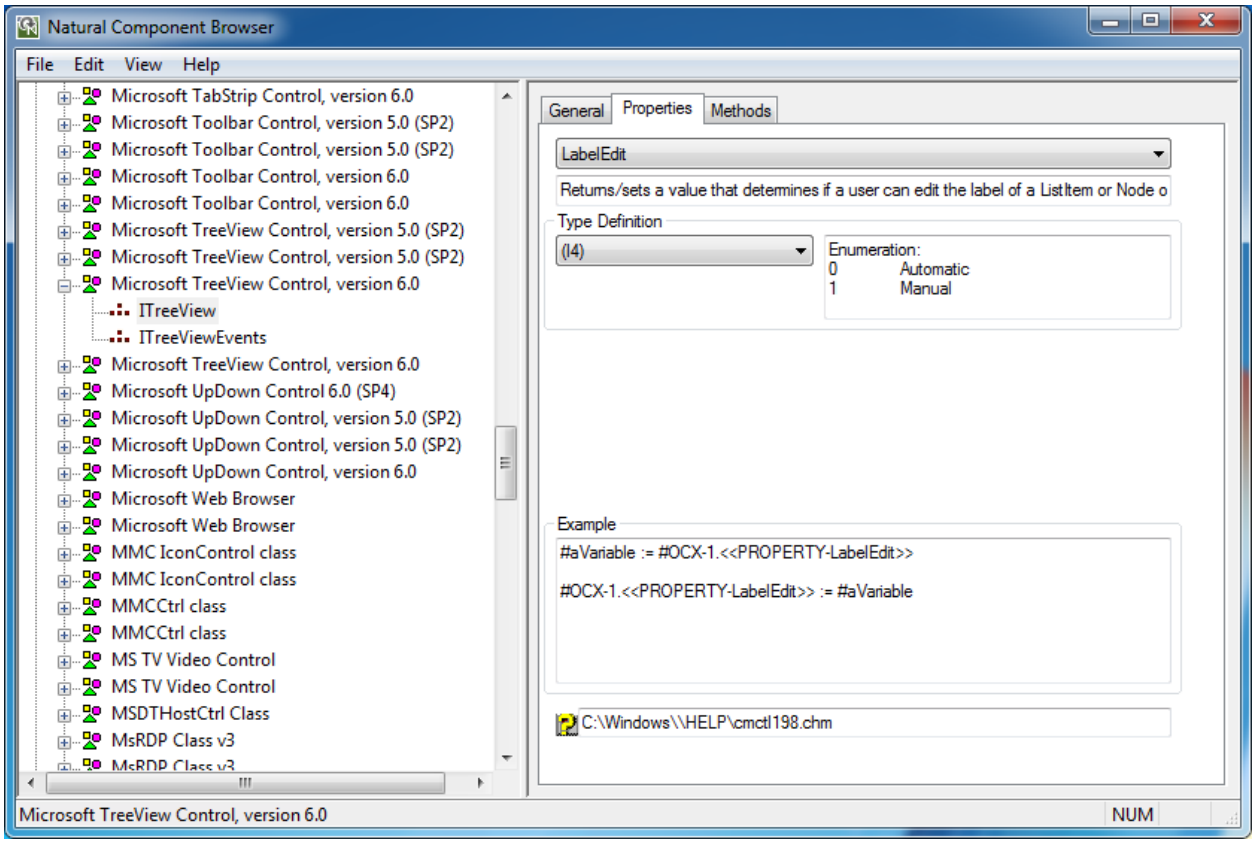
Name	Interface name
Description	Short textual description.
Unique ID	GUID.
Help	Help file name. This file can be opened by choosing 



Properties

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

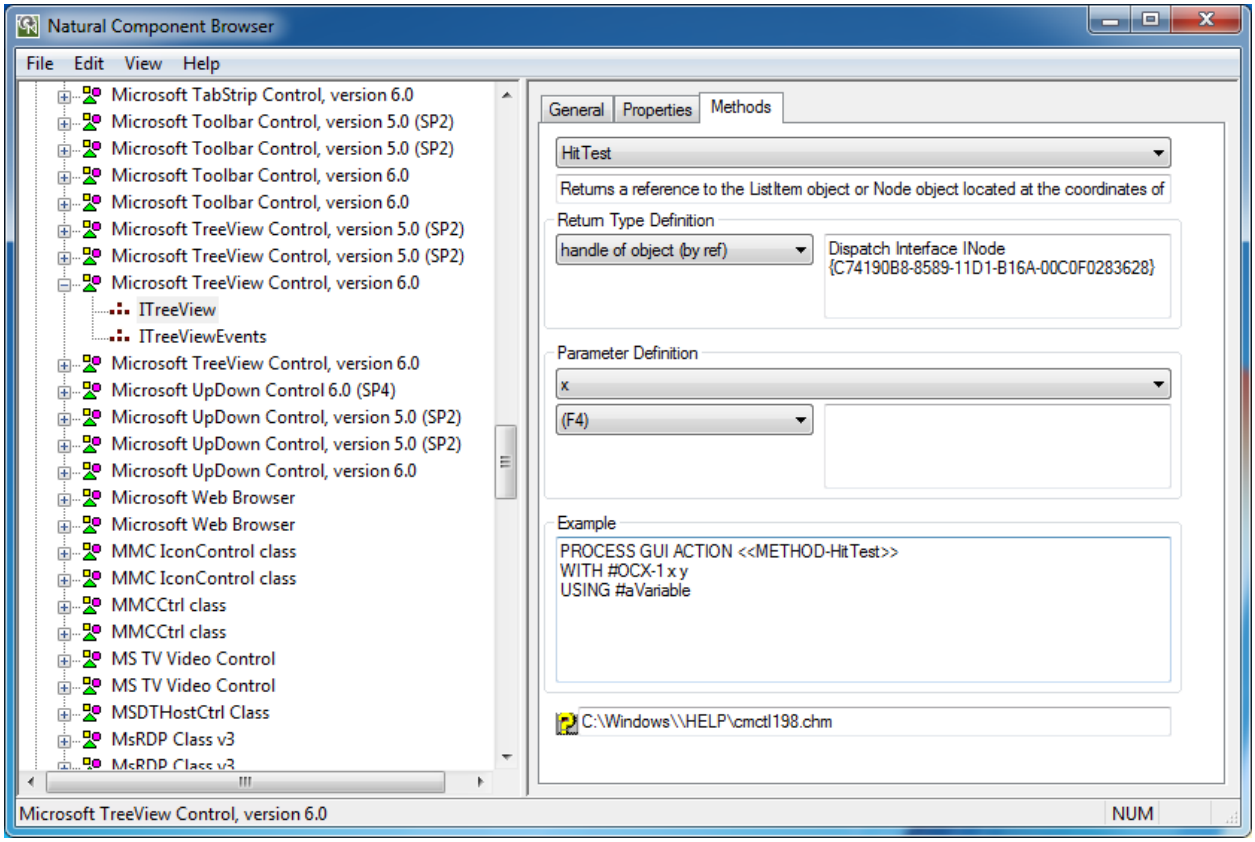
Description	Short textual description for the selected property.
Type Definition	List of valid Natural data formats for the property's type. Additional type info on the selected Natural data format, e.g. valid values for enumeration types.
Example	Example Natural source code. See also *Application Development 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:

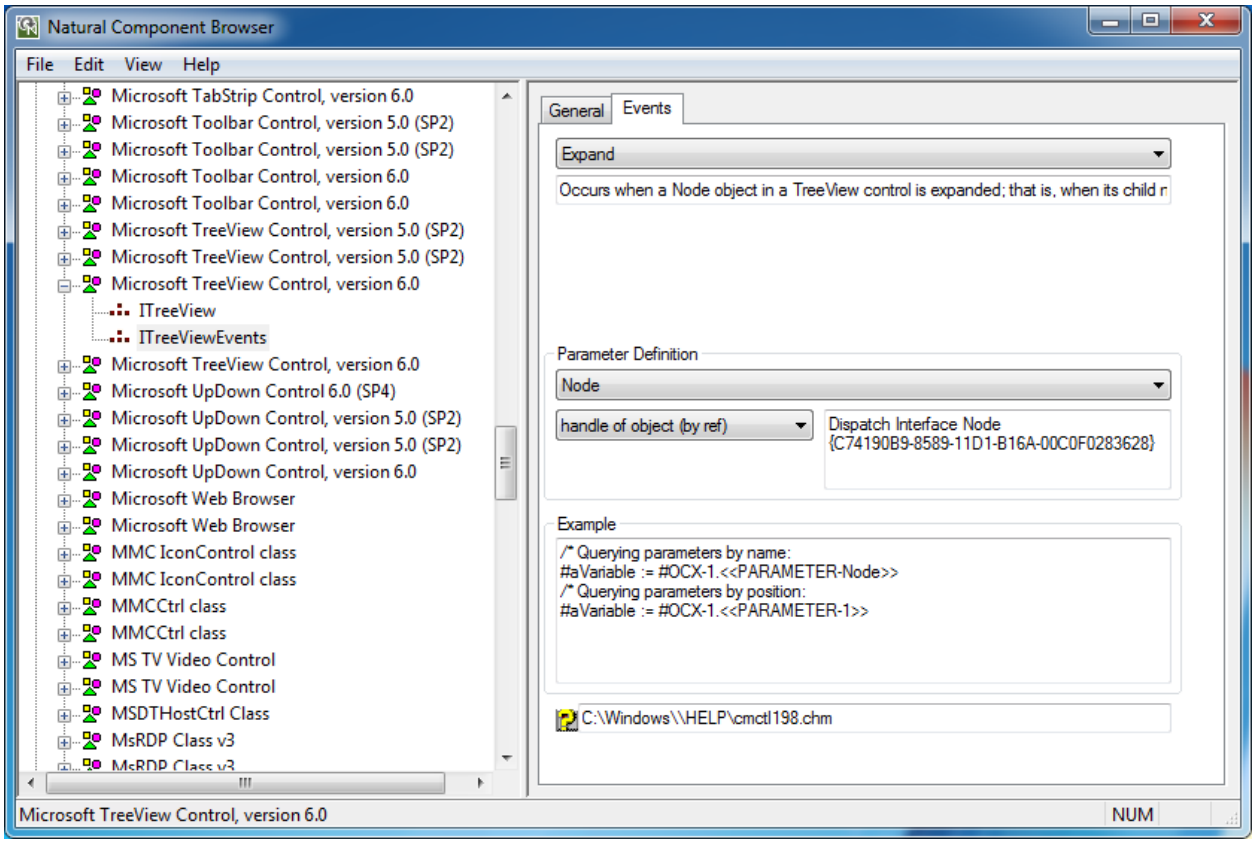
Description	Short textual description for the selected method.
Return Type Definition	List of valid Natural data formats for the method's type. Additional type information on a selected Natural data format, e.g. Name and GUID of the interface if a handle of object corresponds to a dispatch interface. 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

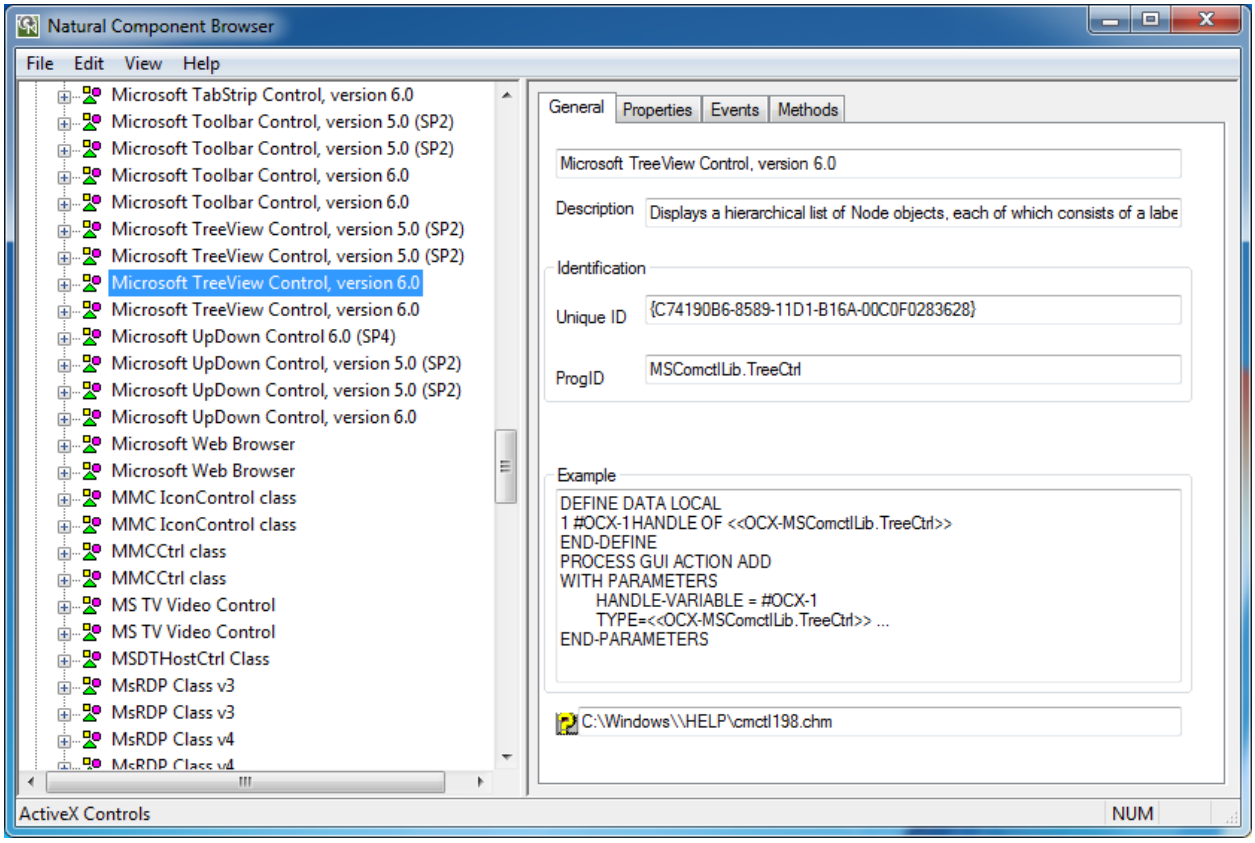
For ActiveX controls, the appropriate `PROCESS GUI` statement is generated that shows how to use these components in Natural applications.

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

- General
- Properties
- Events
- Methods

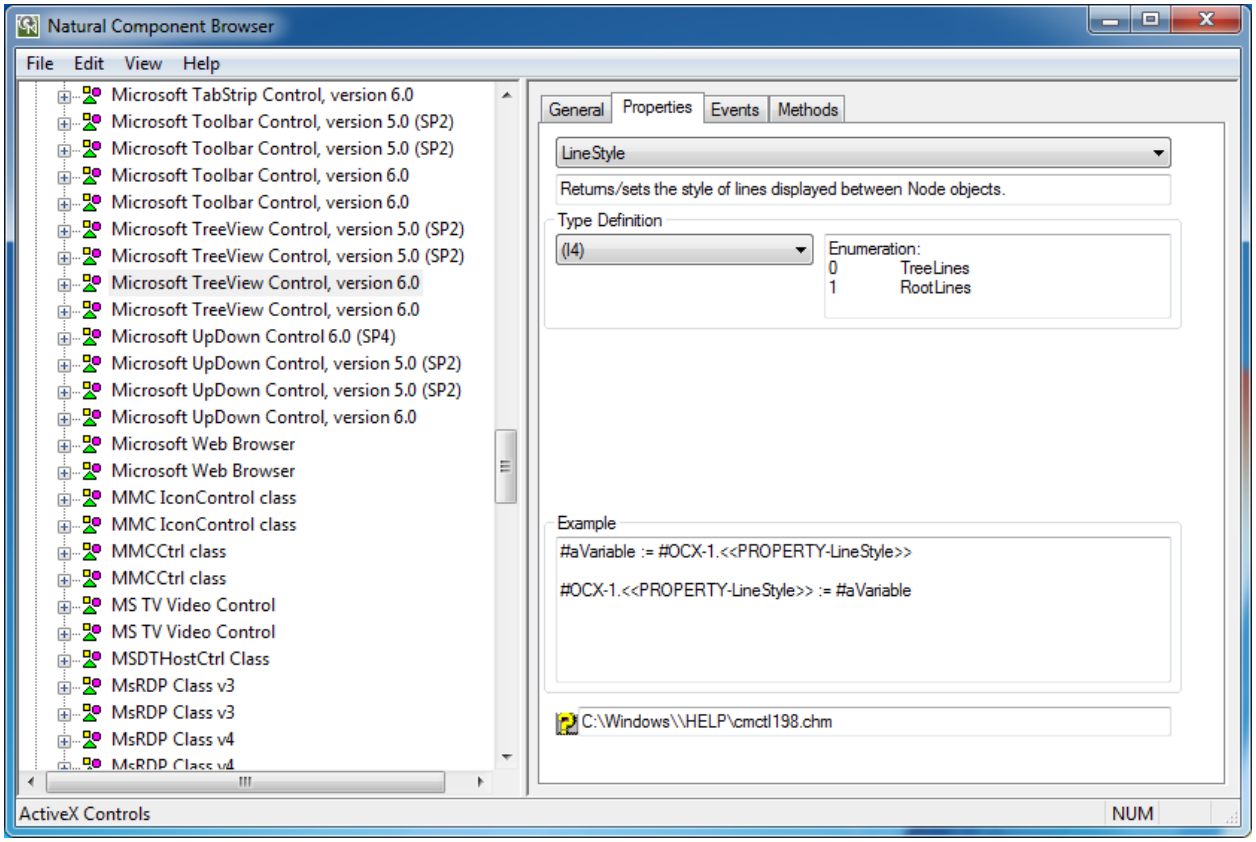
General

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.



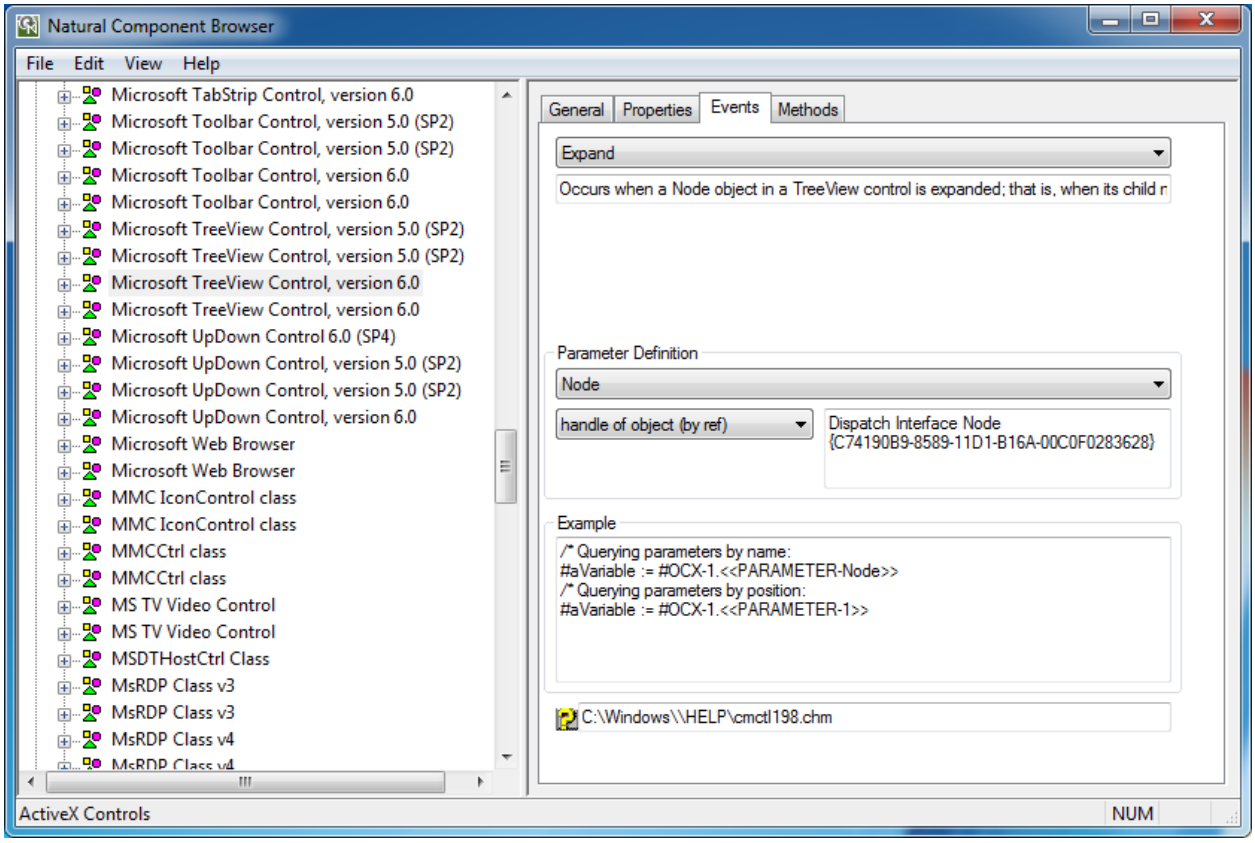
Properties

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.



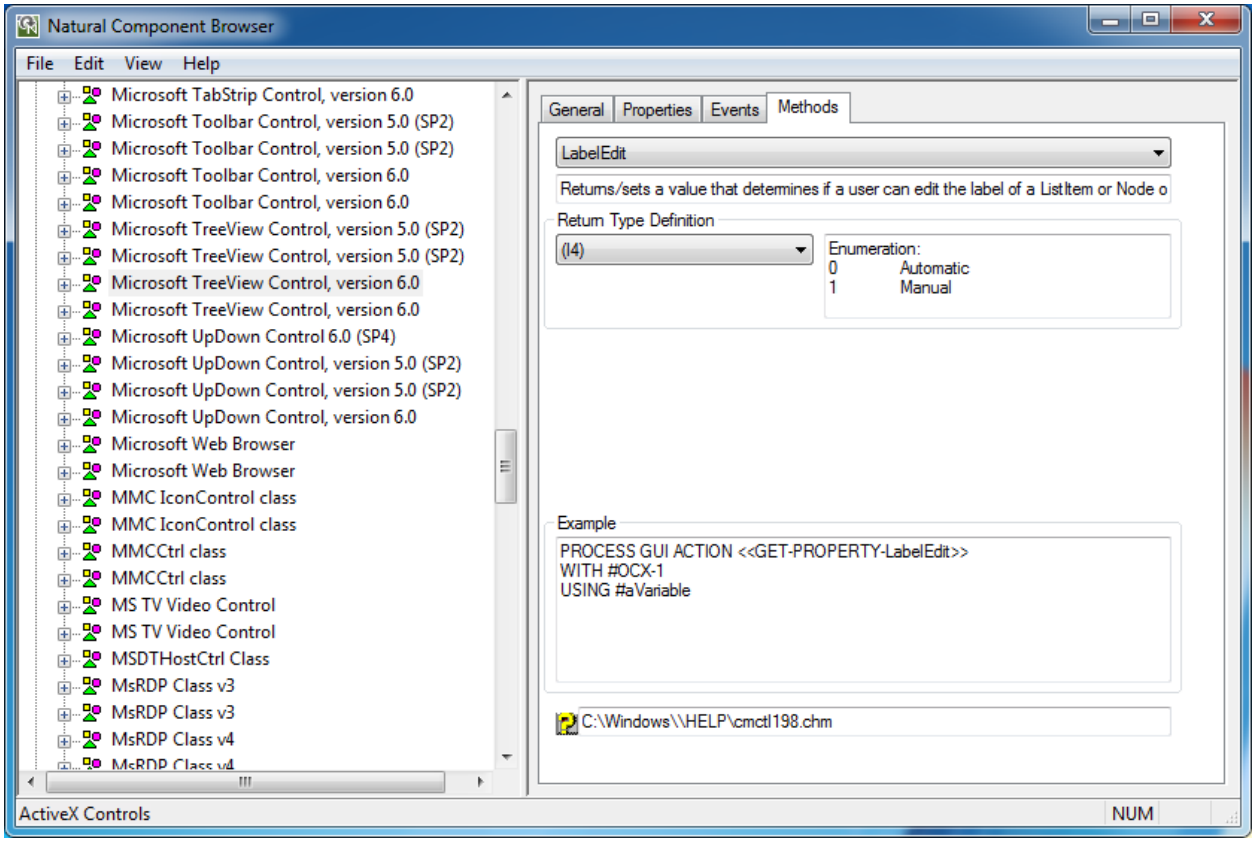
Events

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.



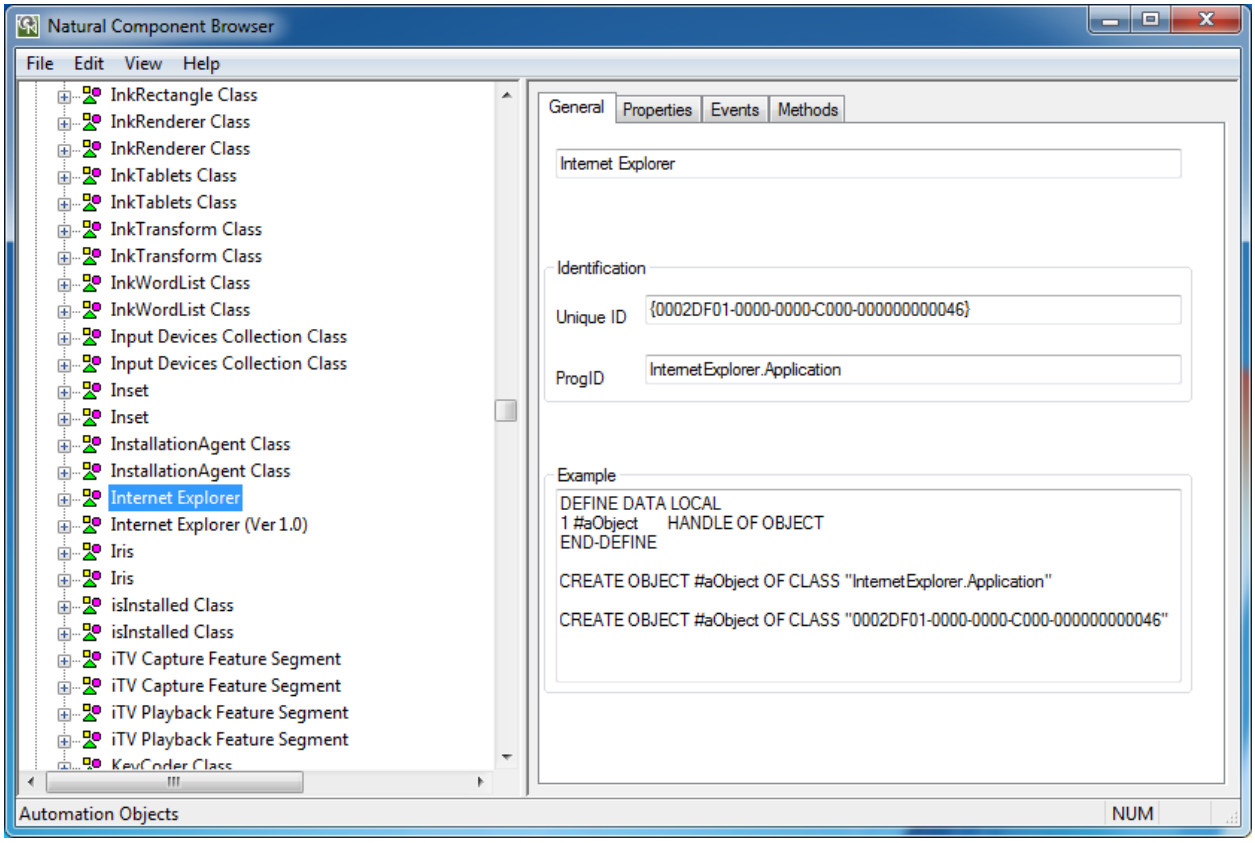
Automation Objects

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

- General
- Properties
- Methods

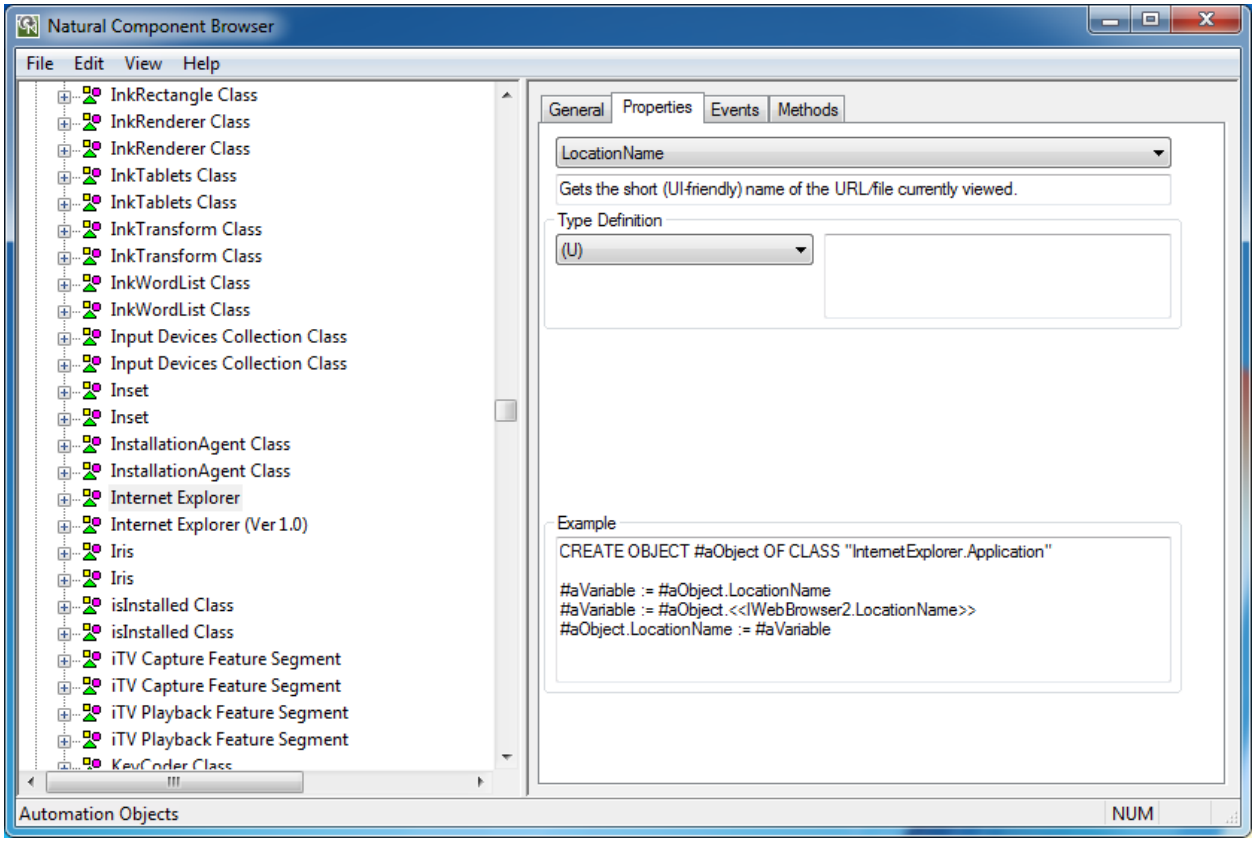
General

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

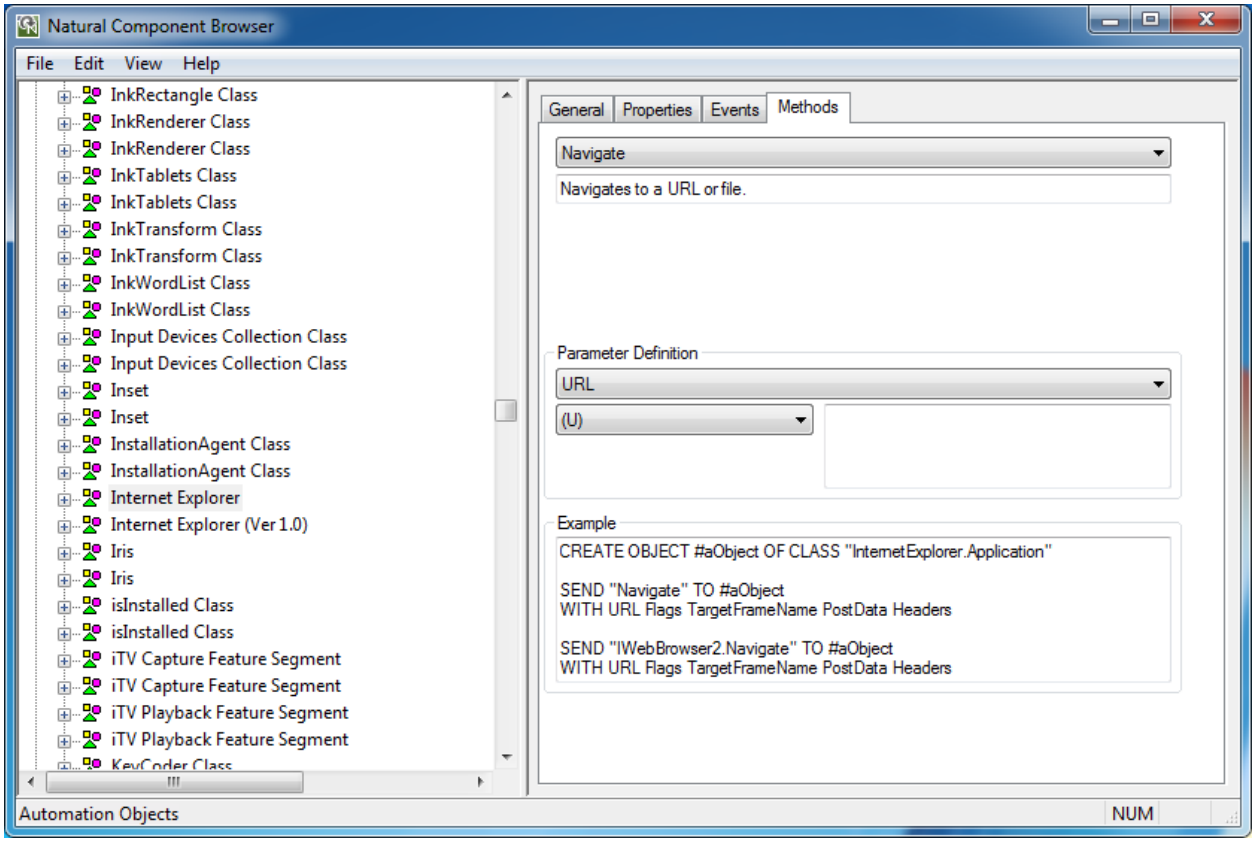
The example on the page **Properties** shows how to assign a property to a variable and how to assign a variable to a property. Here `#aObject` is the defined object handle and `#aVariable` refers to an application-specific variable. Both names can be adapted if required.



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

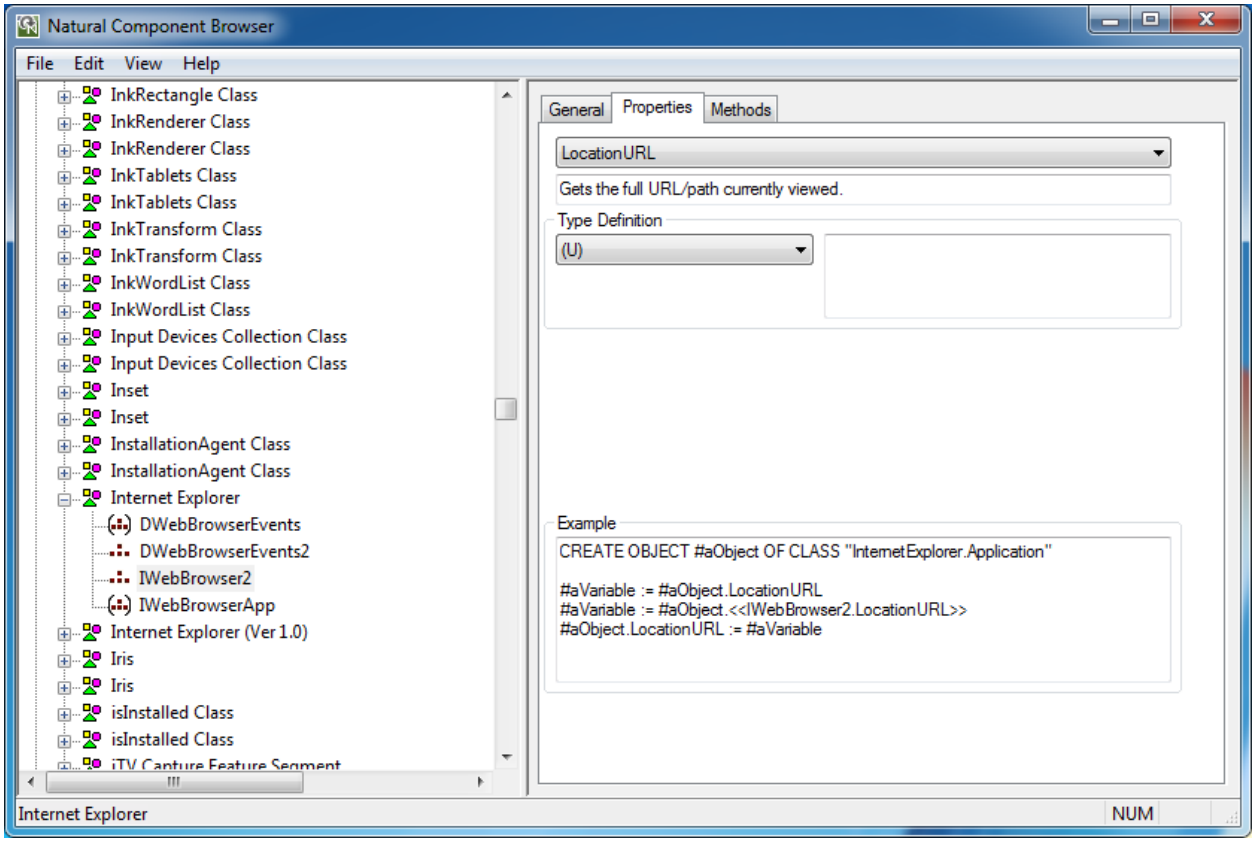
For interfaces that belong to group **Interfaces** and that are not considered in the context of a class, the examples are generated as for Automation Objects. Only the `CREATE OBJECT` statement is left aside.

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

- Properties
- Methods

Properties

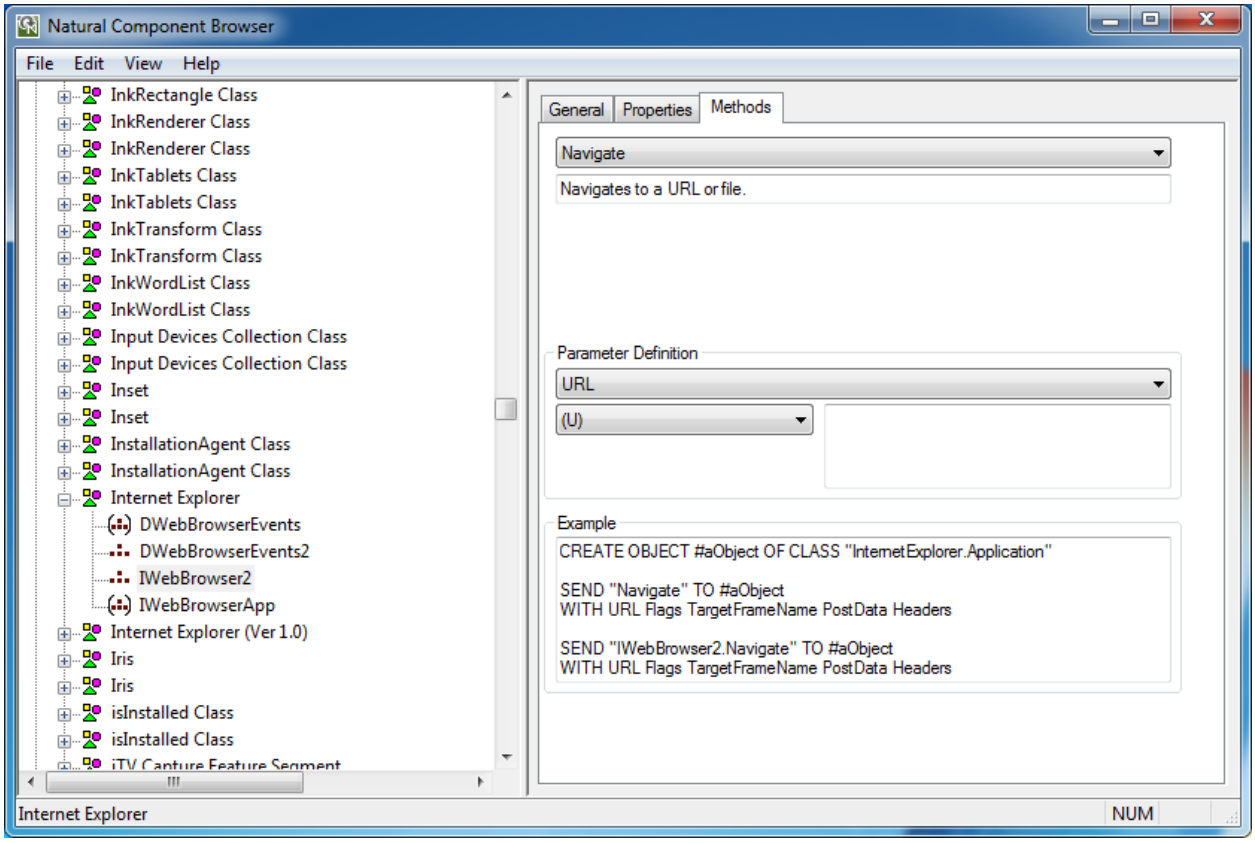
The example on the page **Properties** shows how to assign a property to a variable and how to assign a variable to a property. Here `#aObject` is the defined object handle and `#aVariable` refers to an application-specific variable. Both names can be adapted if required.



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.


Natural Component Browser
File Edit View Help
InkRectangle Class
InkRenderer Class
InkRenderer Class
InkTablets Class
InkTablets Class
InkTransform Class
InkTransform Class
InkWordList Class
InkWordList Class
Input Devices Collection Class
Input Devices Collection Class
Inset
Inset
InstallationAgent Class
InstallationAgent Class
Internet Explorer
DWebBrowserEvents
DWebBrowserEvents2
IWebBrowser2
IWebBrowserApp
Internet Explorer (Ver 1.0)
Iris
Iris
isInstalled Class
isInstalled Class
General
Properties
Methods
Navigate
Navigates to a URL or file.
Parameter Definition
URL
(U)
Example
CREATE OBJECT #aObject OF CLASS "InternetExplorer.Application"
SEND "Navigate" TO #aObject
WITH URL Flags TargetFrameName PostData Headers
SEND "IWebBrowser2.Navigate" TO #aObject
WITH URL Flags TargetFrameName PostData Headers
Internet Explorer
NUM

III Data Browser

4 Data Browser

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.

Starting the Data Browser

To start the data browser

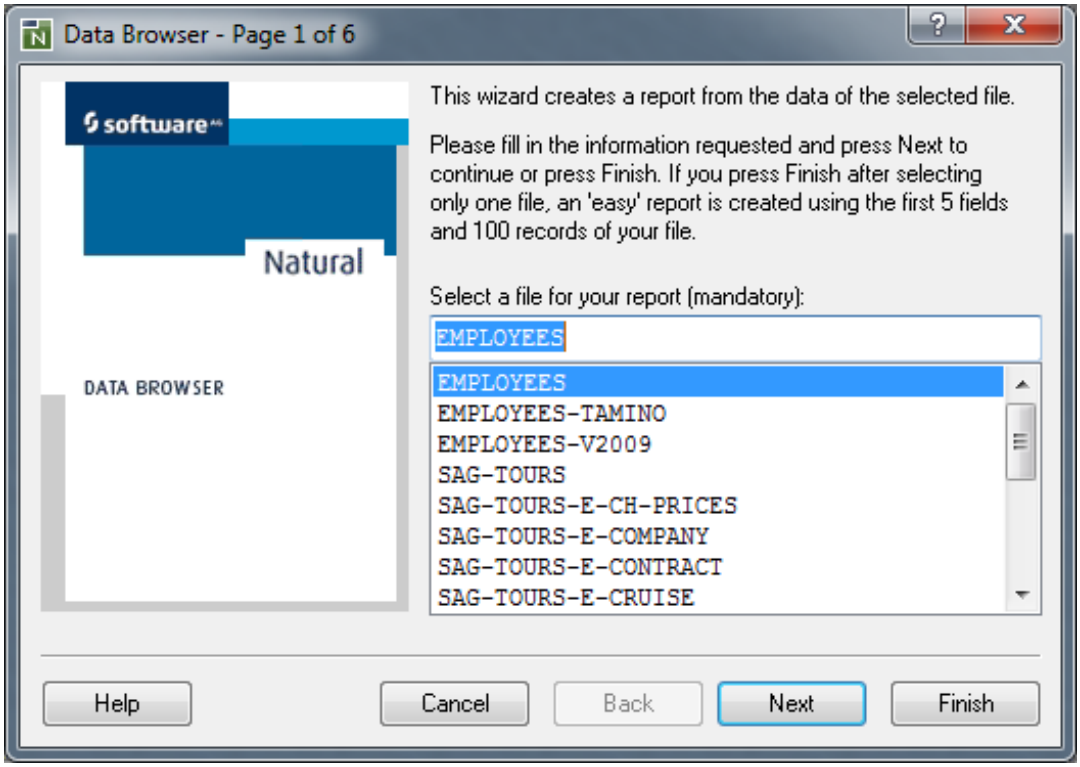
- From the **Tools** menu, choose **Development Tools>Data Browser**.

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 steplibs (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 steplibs 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

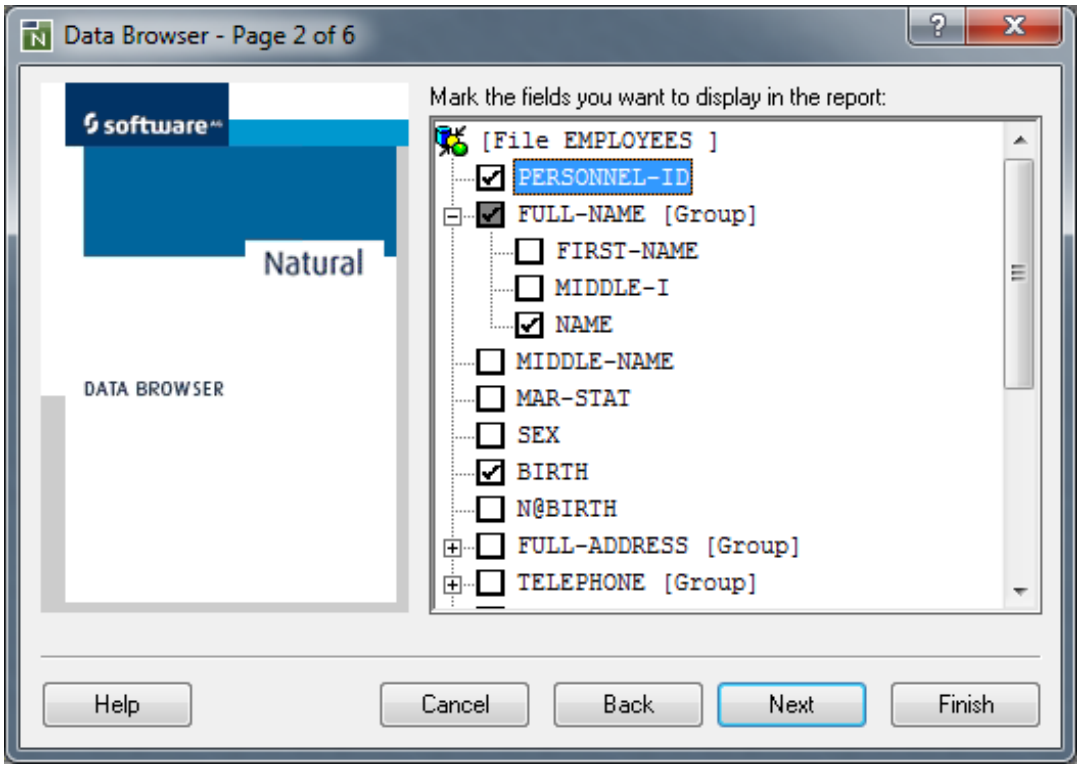
The selection of a DDM is mandatory. After selecting a DDM from the selection list, a check is performed whether the appropriate database is available.

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:

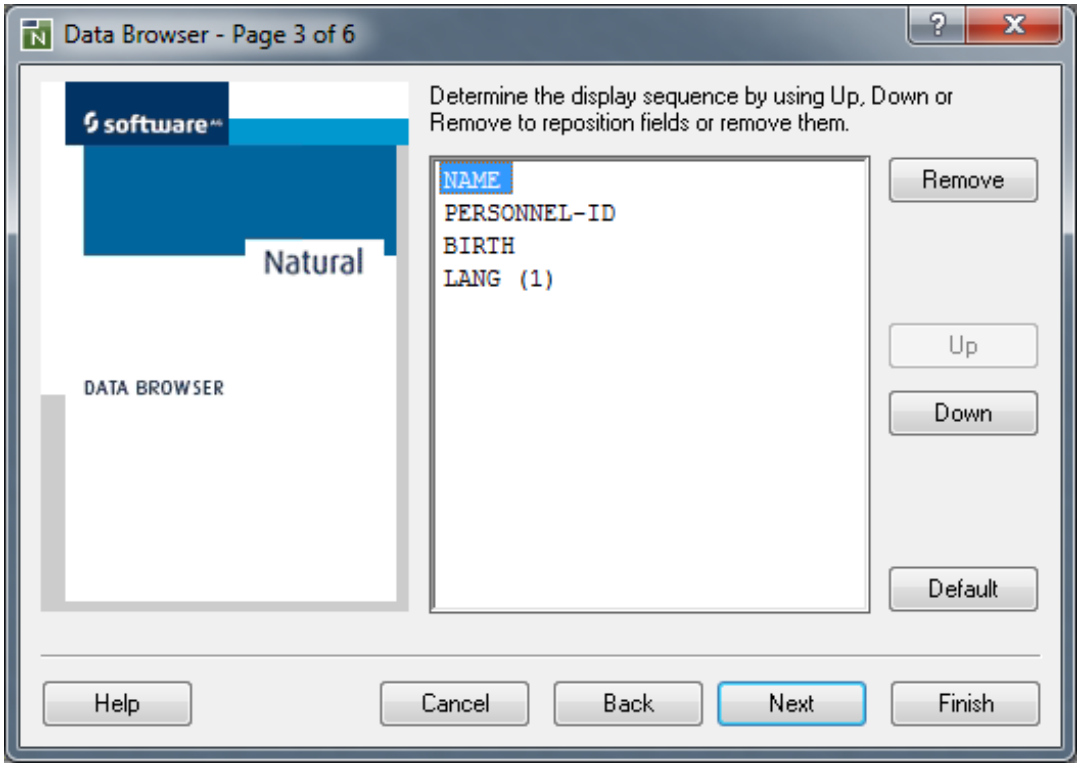
Context Menu

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** und **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:

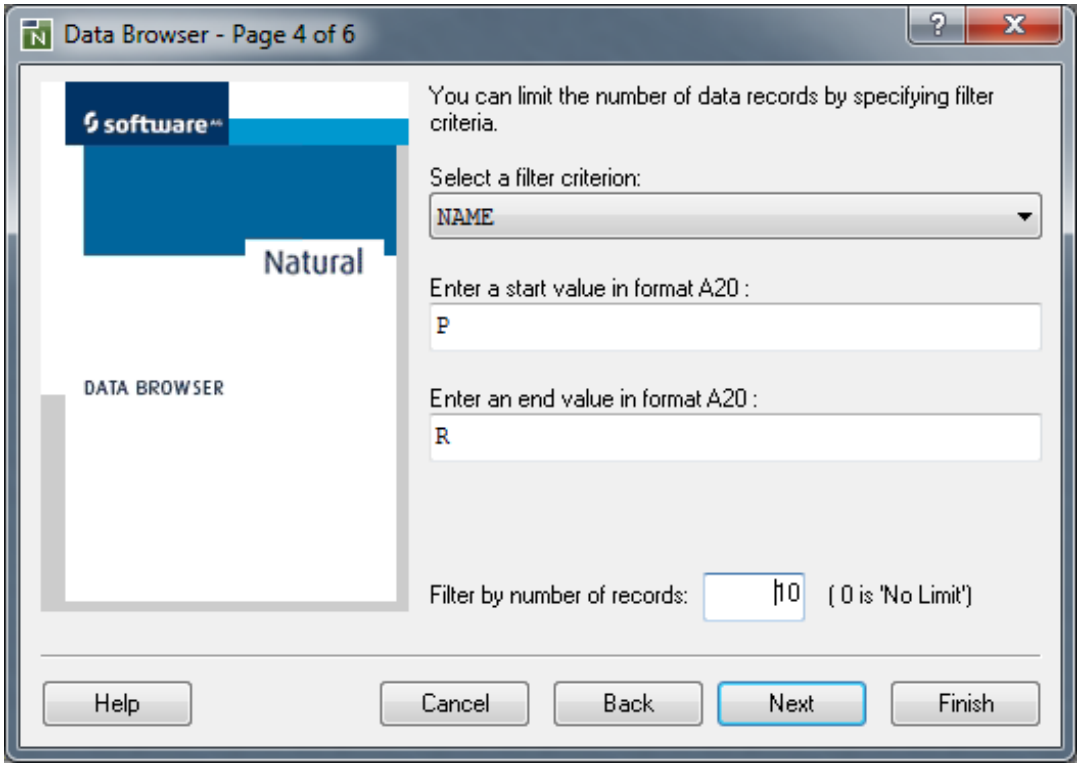
Context Menu

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:



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

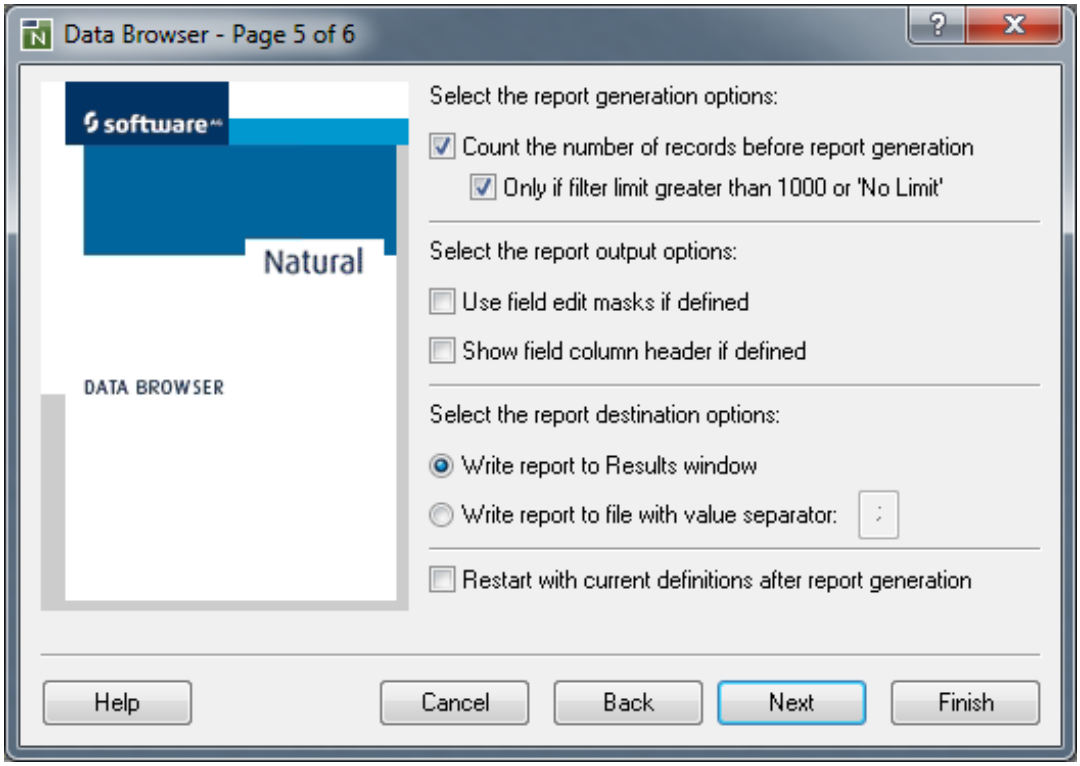
Context Menu

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

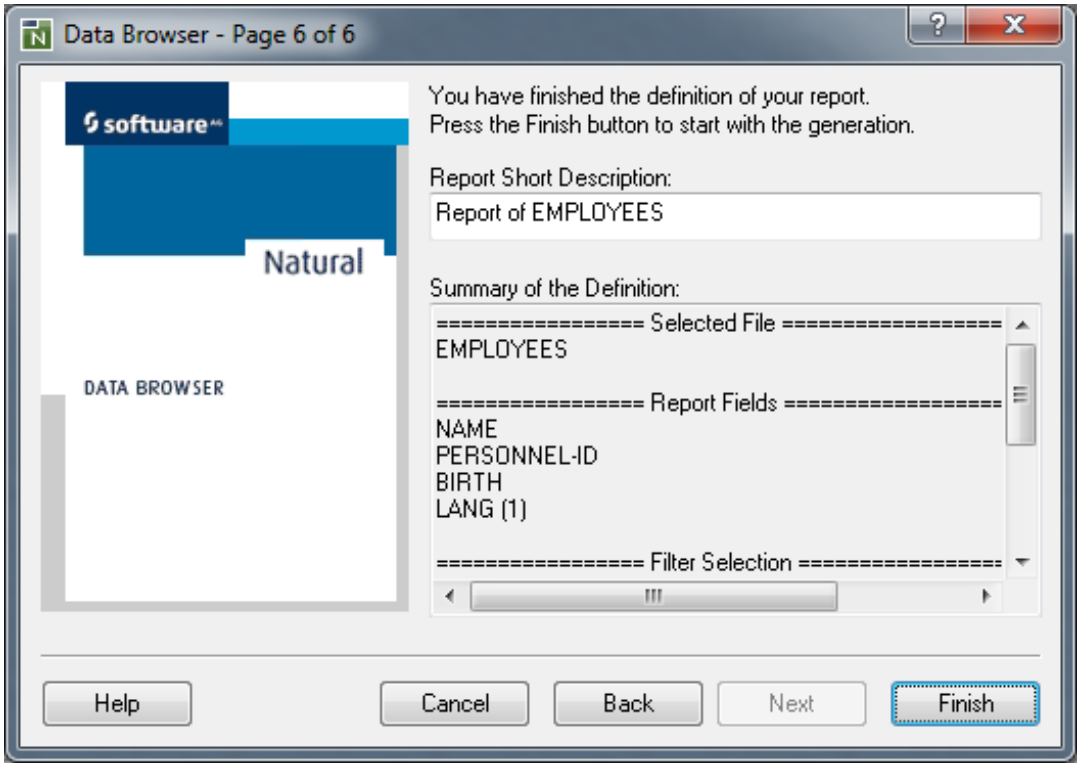
The report options of the data browser are provided on **Page 5**:



Count the number of records before report generation	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. This option is selected by default.
Only if filter limit greater than 1000 or 'No Limit'	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 *Filter Criteria*). This option is selected by default.
Use field edit masks if defined	If defined for the selected fields, uses edit masks for displaying the field values.
Show field column headers if defined	If defined for the selected fields, displays the column headers instead of the field names.
Write report to Results window	Displays the report generated in the **Results** window. This option is selected by default.
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 delimiter 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 displayed 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.

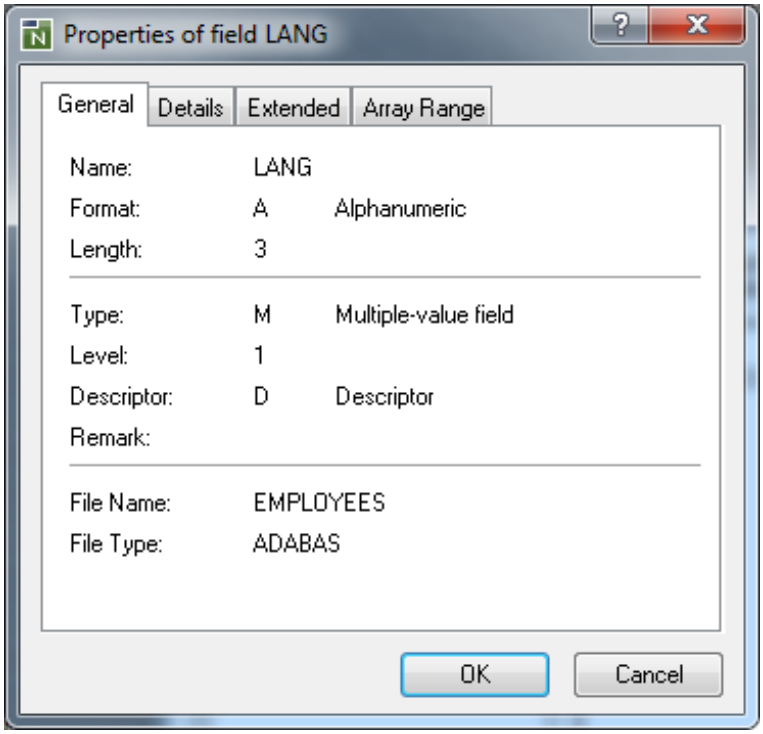
Field Properties

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:

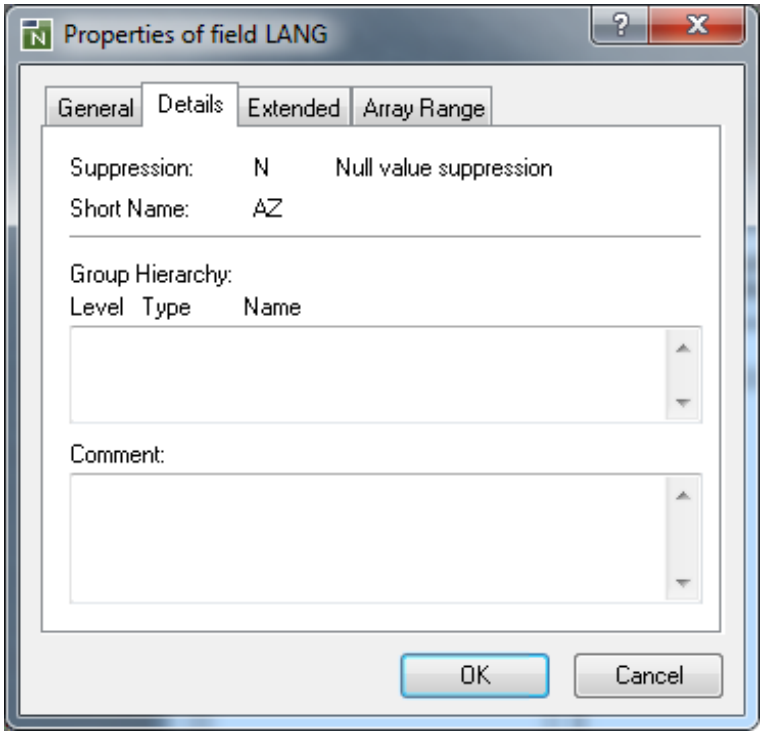


Name	Name as of the DDM.
Format	Format as of the DDM.
Length	Length as of the DDM.
Type	Field type: group, periodic group, multiple-value field or elementary field.
Level	Level as of the DDM.
Descriptor	Descriptor type:

blank	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)

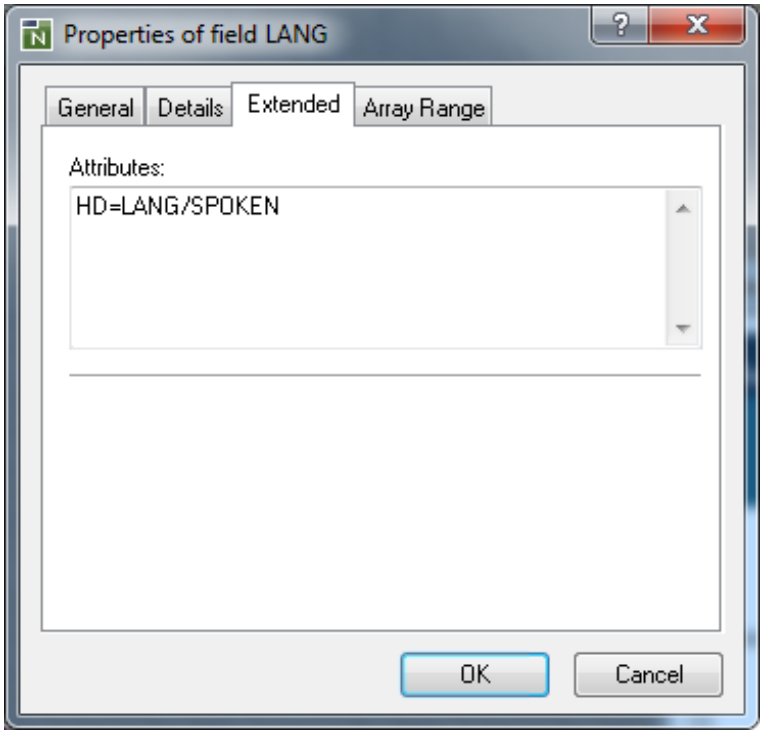
	See further information, see also *Columns of Field Attributes* in the *DDM Editor* documentation.
Remark	Remark as of the DDM.
File Name	File name as of the DDM.
File Type	Database type.

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



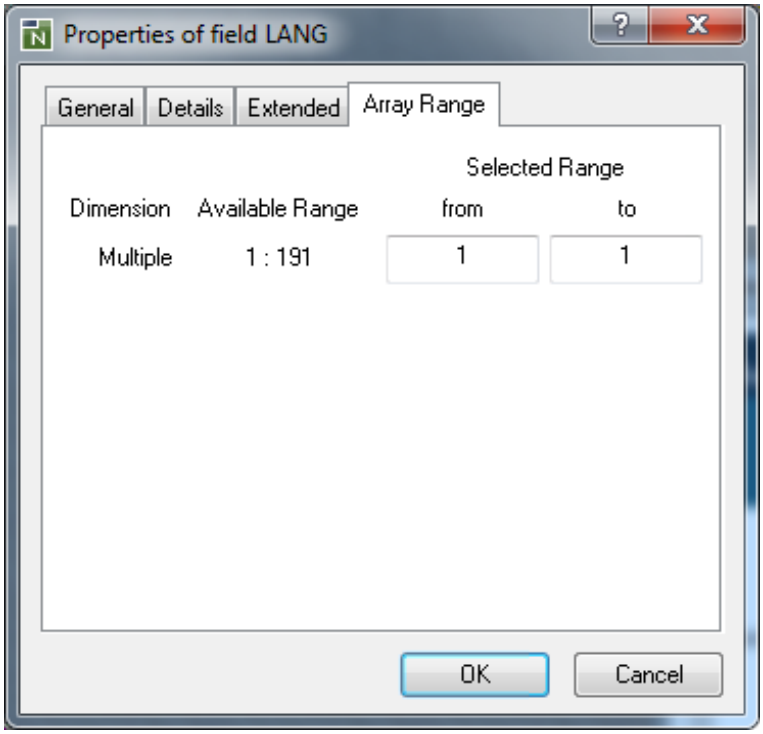
Suppression	Null value suppression, fixed storage, not null.
Short Name	Adabas short name of the field.
Group Hierarchy	All groups and group levels to which a field belongs.
Comment	Comment as of the DDM.

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:



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

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.

The default setting for both fields is 1 for the first occurrence of each dimension.

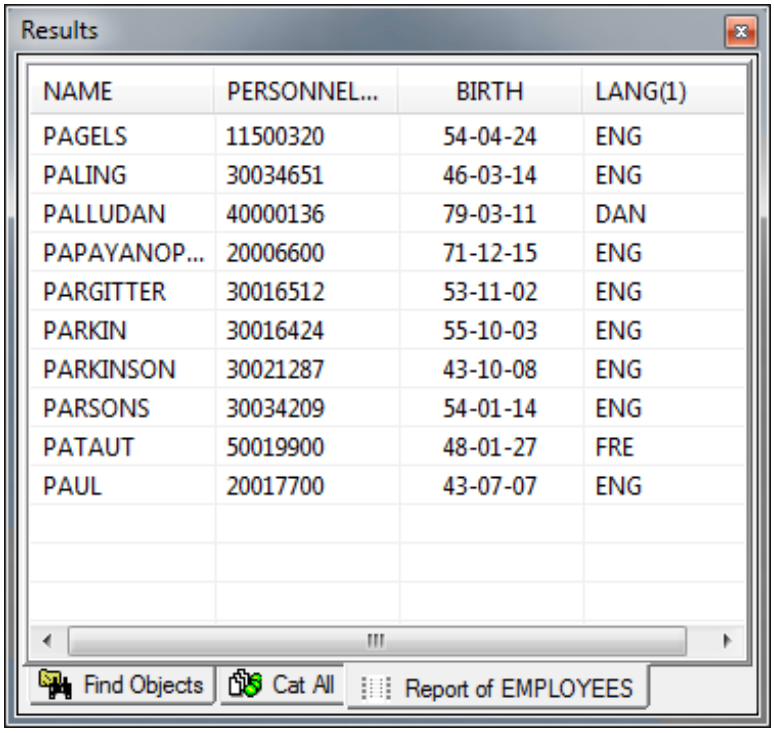
The selected range(s) of occurrences are indicated on Page 3 (see Output Report Fields) and Page 6 (see Summary).

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.

Caution: A wide range of occurrences can result in a large number of columns, which can cause significant performance problems.</td></tr>
</table>

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:



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:

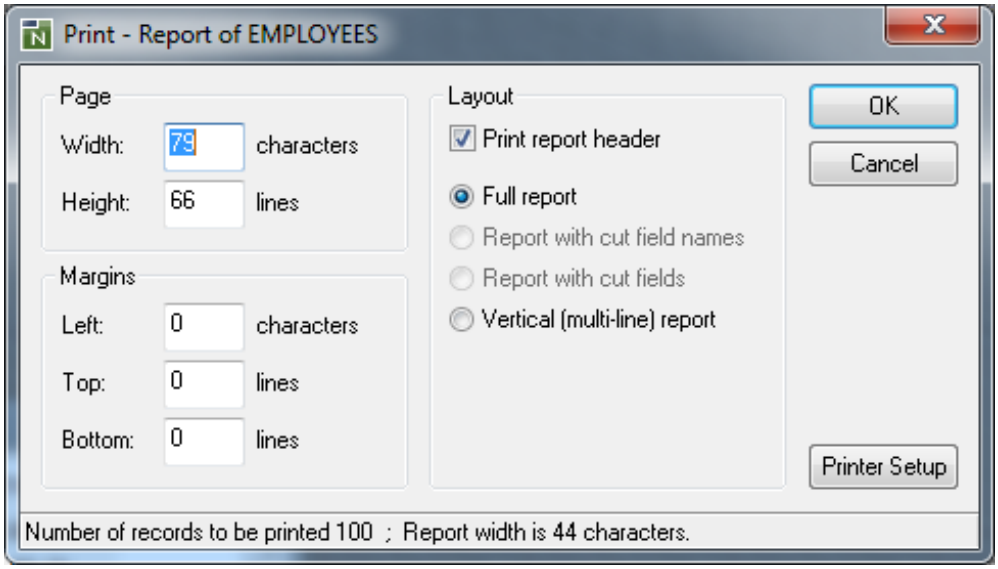
Select All	All lines of the result list are selected.	
Expand All Columns	Expands all columns in the result list.	
Collapse All Columns	Collapses all columns in the result list.	
Output	**Print Selection**	Prints the selected lines of the result list. For more information, refer to ***Print***.
	Export 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:



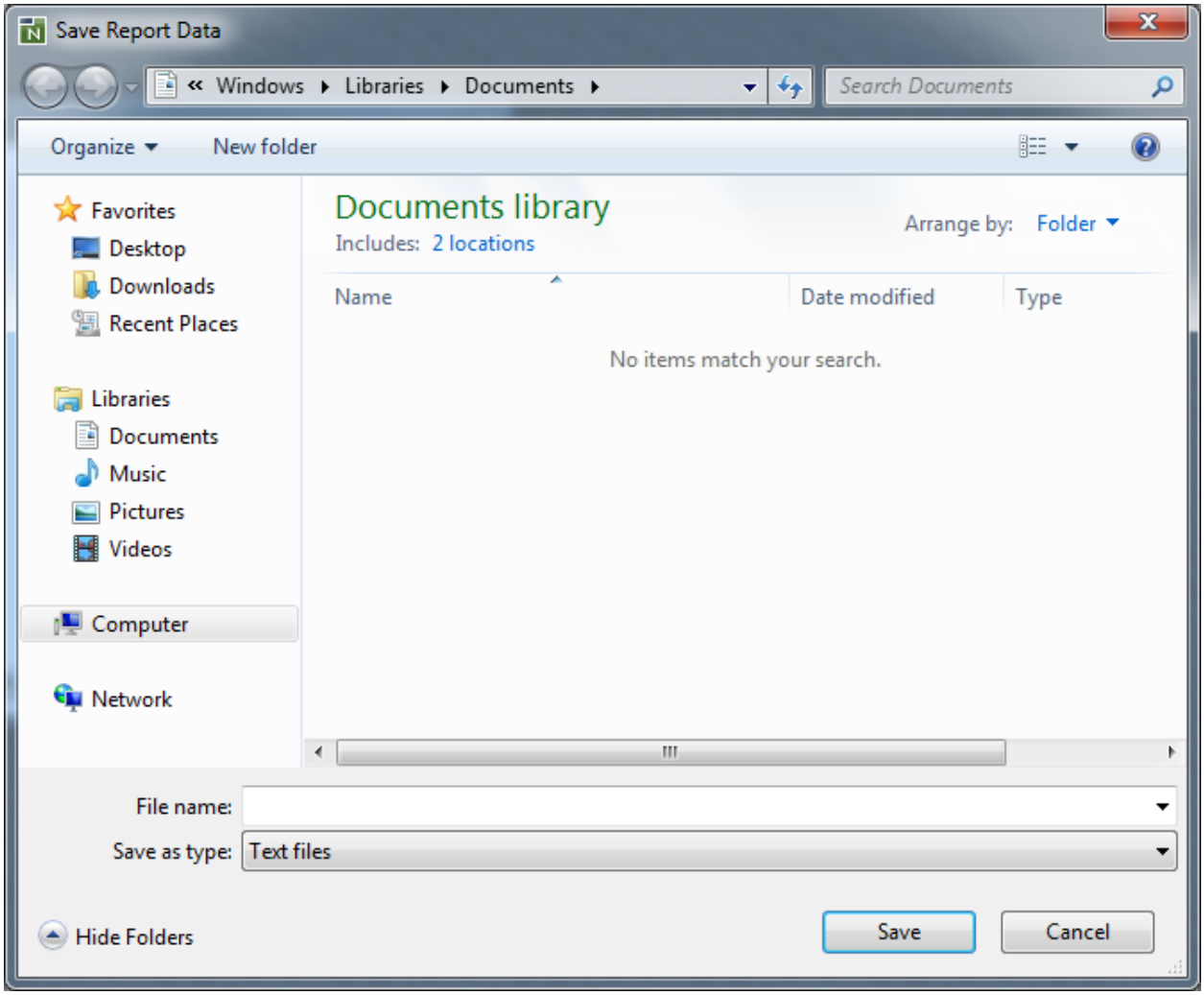
The **Print** window provides the following options:

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	For some criteria you can define the printed report layout. 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**. 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.

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:

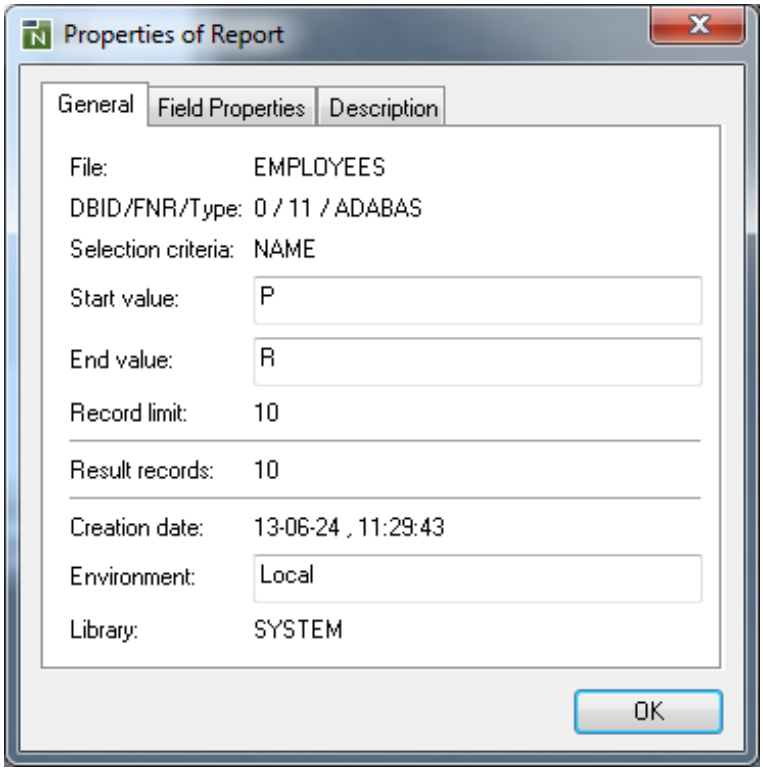


The standard save dialog enables you to enter a file name completed with the file type `.txt`.

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.

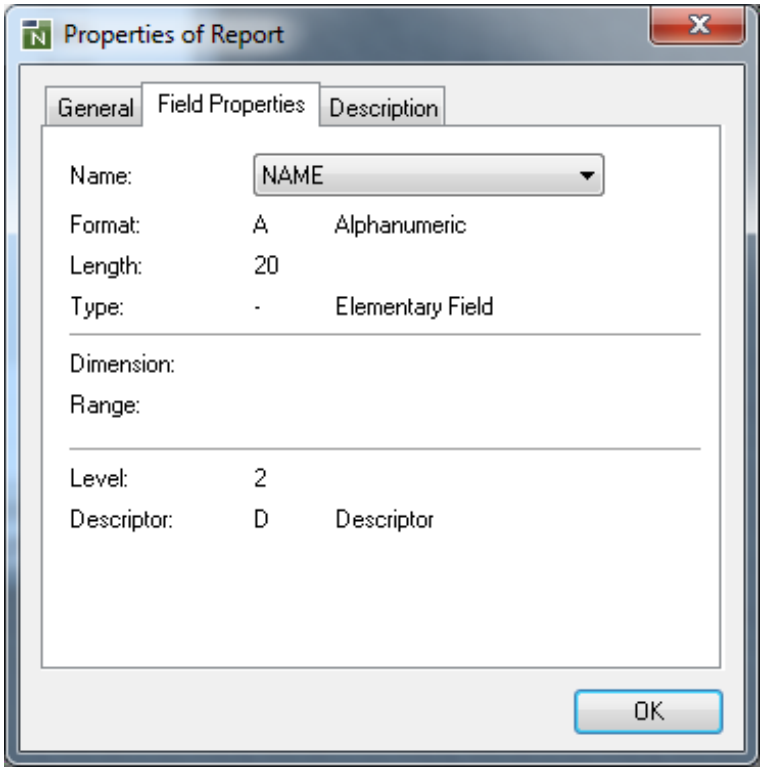
Properties of Report

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



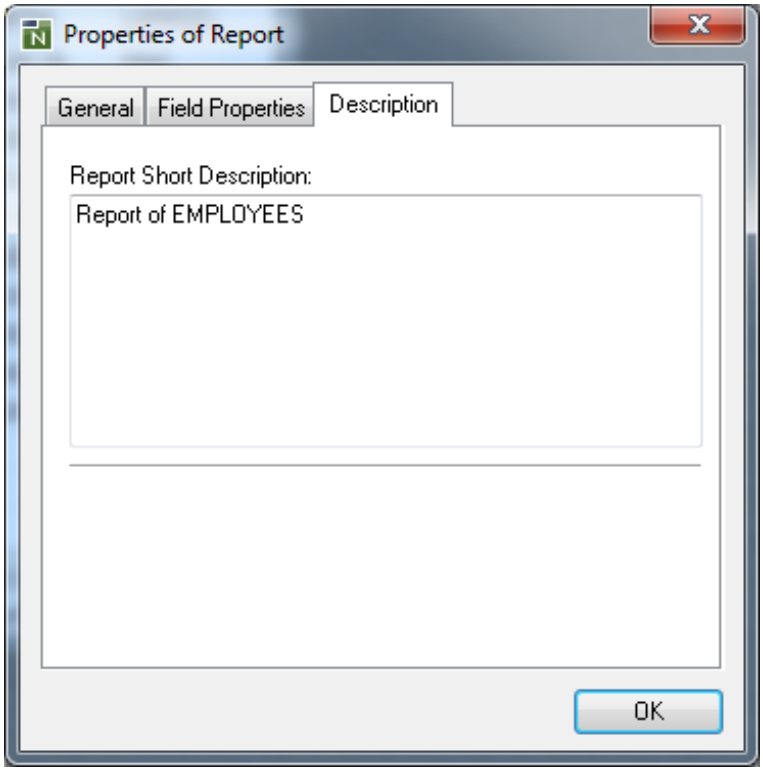
The tabbed page **General** shows the following:

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 `Periodic` for a periodic group or `Multiple` 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 FTOUCH Utility

5 FTOUCH Utility

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.

Note: Using the FTOUCH utility is limited to 60000 objects because this is the maximum number of entries in the file *FILEDIR.SAG*.

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]
[userep=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][suprln=library-state]
```

Syntax of ftouch

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

Option	Explanation
`fnat=dbid,fnr`	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***.
`fuser=dbid,fnr`	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***.

<table>
<tr><th>Option</th><th>Explanation</th></tr>
<tr><td>bp=bp-name</td><td>Specifies the buffer pool to be used. You can omit the bp-name if you want to use the Natural default buffer pool NATBP; otherwise, you have to specify the appropriate bp-name.

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.</td></tr>
<tr><td>parm=parm-name</td><td>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.</td></tr>
<tr><td>lib=library-name</td><td>Specifies the library to be used. You can omit the library-name if you are already in the appropriate subdirectory; otherwise you have to specify the appropriate library-name.</td></tr>
<tr><td>userep=rep-use</td><td>Specifies whether to use the repository or not. rep-use must be one of the following:

ON — The repository is used.

OFF — The repository is not used.</td></tr>
<tr><td>-v</td><td>Displays statistics on disk I/O operations during processing.</td></tr>
<tr><td>-q</td><td>Indicates that quiet mode is to be used: only error messages but no status messages are displayed.</td></tr>
<tr><td>-ignoreext</td><td>Specifies that files with unkown extensions contained in a library are ignored. The -ignoreext option can be combined with one of the following options:

-a
-d</td></tr>
<tr><td>mode</td><td>Specifies the programming mode; sm specifies that a program is in structured mode; the default is reporting mode.

See also Example 1.</td></tr>
<tr><td>kind</td><td>Specifies the subdirectories SRC and/or GP for input; it can be one of the following:

-s for source objects (default),
-g for cataloged objects/generated programs,
-b for both source objects and cataloged objects/generated programs.

See also Example 2.</td></tr>
</table>

Option	Explanation
files	Specifies the files to be processed; you can specify *filename.ext* for individual files or: `-a` to add new files; all files in the directory which are not currently found in *FILEDIR.SAG* are added (already existing files are not touched). `-d` to build a new *FILEDIR.SAG* directory. **Caution:** Be careful when using this option, since the old *FILEDIR.SAG* is deleted and rebuilt from scratch. See also ***Example 4***.
`-f`	Forces an update of the specified object's timestamp in *FILEDIR.SAG*. This option can only be specified if an individual file has been specified with the *files* option (see above).
`convert`	Indicates that an old *FILEDIR.SAG* file is to be migrated. The *FILEDIR.SAG* file from a Natural version earlier than Version 6.2 is converted into a new portable *FILEDIR.SAG* file. A copy of the original (old) *FILEDIR.SAG* file is saved as *FILEDIR.BCK* file in the directory of the specified library. If a *FILEDIR.BCK* file already exists in the specified library, the old *FILEDIR.SAG* will *not* be converted. For further information, see *Portable Natural System Files* in the *Operations* documentation. See also ***Example 3*** and ***Example 5***.
`sync`	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 *FILEDIR.SAG* is modified by FTOUCH. **Caution:** When specifying `sync`, ensure that either `userep=ON` is set or the Natural profile parameter `USEREP` is set to `ON`.

`encoding=`*encoding-name*	Specifies the code page to be used for the files contained in *FILEDIR.SAG*.
	
	The `encoding` option generates or changes the internal code page information maintained in *FILEDIR.SAG* for each object affected by the `ftouch` command. This option does *not* convert the contents of a source object or a cataloged object/generated program.
	
	The `encoding` option can be combined with the following options:
	
	`-a`
	`-d`
	`convert`
	`objname`

Option	Explanation
	encoding-name can be any code page name valid with the CP session parameter specified in the *NATPARM* parameter file. See also *CP - Default Code Page Name* in the *Parameter Reference*. See also ***Example 4***, ***Example 5***, ***Example 7*** and ***Example 8***.
endian=*endian-mode*	Specifies the endian format to be used for the *FILEDIR.SAG* directory. The endian option applies to the entire *FILEDIR.SAG* directory. The option does not apply when adding files to *FILEDIR.SAG* or when generating a new *FILEDIR.SAG*. *endian-mode* can be one of the following formats: 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=*object-name*	Selects the object(s) for which to maintain internal format information in *FILEDIR.SAG*. The objname option only applies if the encoding option is specified. *object-name* 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=*library-state*	Specifies whether the line number suppression state is set for the specified library. *library-state* must be one of the following: ON: Source line numbers are not written to the files contained in *FILEDIR.SAG*, when saving the sources of the objects contained in this library. OFF: Source line numbers are written to the files contained in *FILEDIR.SAG*.

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

6 INPL Utility

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

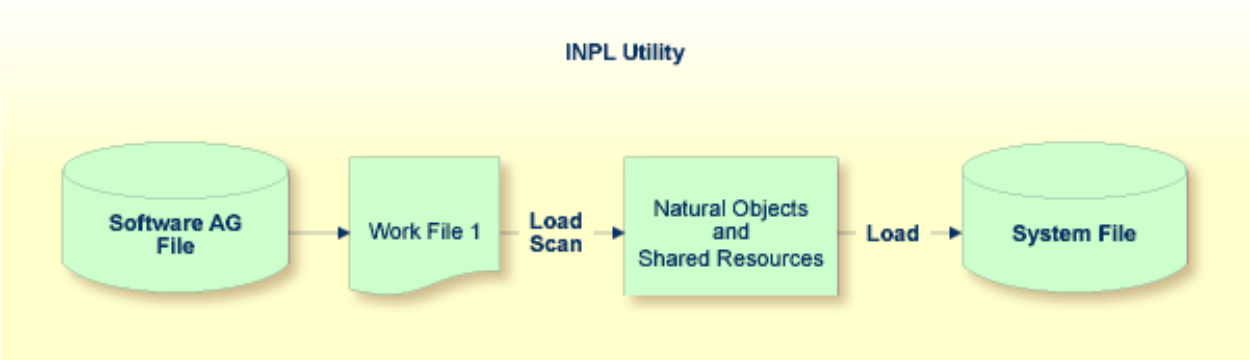
Notation *vrs* or *vr*:

When used in this document, the notation *vrs* or *vr* represents the relevant product version (see also Version in the *Glossary*).

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

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`.

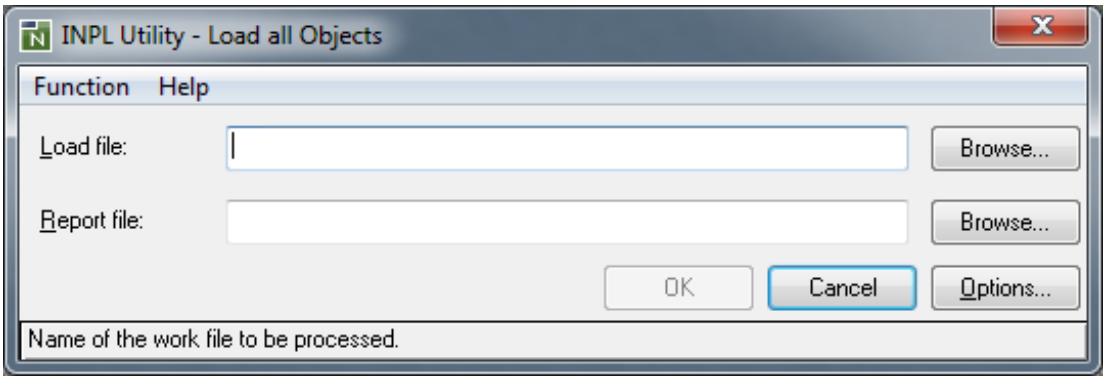
Invoking INPL

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.

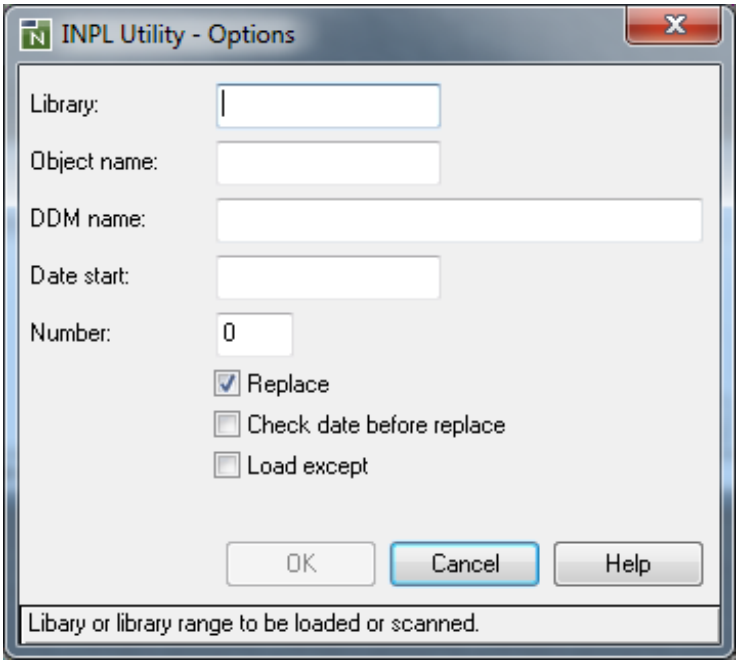
4 From the **Function** menu, choose one of the following functions:

- **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 ... $NATWORK/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	The name of a DDM or a range of names. 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.
Date start	Restricts processing to Natural objects and shared resources which were saved or cataloged on or after the date entered in this text box. The date must be entered in the format *YYYY-MM-DD* (*YYYY* = year, *MM* = month, *DD* = day).
Number	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. If the number of Natural objects processed has reached the value entered in the **Number** text box, processing is terminated with a corresponding message.
Replace	Specifies whether the Natural objects and shared resources to be processed are to replace any that already exist on the system files. Possible settings are: Checked — All existing Natural objects and shared resources are replaced. This is the default setting. Unchecked — Existing Natural objects and shared resources are *not* replaced. 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.
Check date before replace	Specifies whether existing Natural objects and shared resources are to be replaced depending on their time stamp. This parameter has no effect if **Replace** is not selected. Possible settings are: 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. Unchecked — All objects are replaced. This is the default setting.
Load except	Specifies whether to exclude Natural objects and shared resources from processing. This parameter does not apply to error messages. Possible settings are:

Item	Description
	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**. Unchecked — No exceptions; all Natural objects and shared resources are processed. This is the default setting. **Examples of load exceptions:** All libraries except the library ABC are loaded: **Function** = **Load Libraries Only** **Library** = `ABC` All DDMs with a prefix other than XY are loaded: **Function** = **Load DDMs Only** **DDM name** = `XY*` 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*`
Natural Security recover object	Only applies if the function **Natural Security Recover** has been selected. If checked, resets the owner information of specified Natural objects.

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 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.

The *Object Handler* documentation is organized in the following parts:

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: **Natural Library Objects**, **Natural System Error Messages**, **Natural Command Processors**, **Natural DDMs**, **Natural-Related Objects**, **External Files** and **FDTs**.
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.

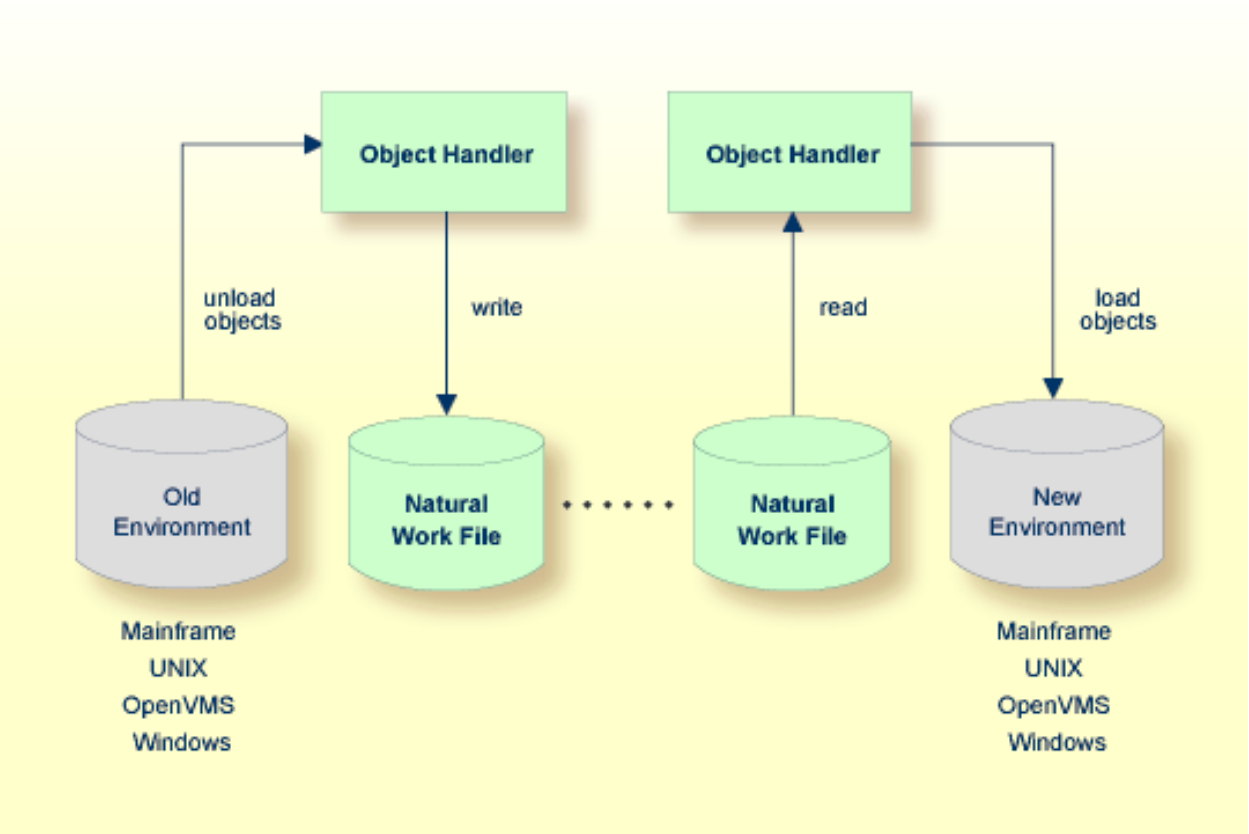
7 General Information on the Object Handler

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

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

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 Objects 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 objects that have not been explicitly specified in the **Options** window. The names generated are a combination of the weekday and the time. For example: an object 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 and the parameter settings are 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 parameter file.
- 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:

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

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

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	Only valid if **Use default options** (this is the default) or **Use additional options** has been selected (see below). If selected, the data to be processed is written in Transfer format to the work file. See also ***Work File Format*** in *Work Files*.
Local work file	Only applies to remote environments. Only valid if **Use default options** (this is the default) or **Use additional options** has been selected (see below). 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.

Item	Explanation
	See also WFLOC in *Direct Commands*.
Portable work file	Not applicable to work files located in a remote environment on a mainframe platform. This option is only valid if the following applies: ■ **Use default options** (this is the default) or **Use additional options** has been selected (see below). ■ **Transfer format** has *not* been selected. If **Portable work file** has been selected, the work file is written or read in portable format. See also ***Work File Format*** in *Work Files*.
Unicode work file	Only applies if **Transfer format** has been selected. 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.
Fixed length	See FIXEDLENGTH in *Direct Commands*.
Unload file (Server)	Only valid if **Use default options** (this is the default) or **Use additional options** has been selected. See below. 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.
Browse	Not applicable to server unload files. Invokes the browse function to select a work file from a directory.
Use default options	Default options are used (this is the default). For the options available, see ***Set Additional Options*** in *Settings - Options*.
Use additional options	Used in connection with **Set** (see below).
Set	Only activated if **Use additional options** has been selected. 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 *Settings - Options*.
Use Option Workplan	If 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	Only valid if **Use Option Workplan** (see above) has been selected and the name of a valid Workplan of the type OPTION was entered. Displays the contents of the Workplan specified.

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 *Settings*.
Set	Only activated if **Use global parameters** has been selected. If selected, the global parameters window is invoked. See ***Set Global Parameters*** in *Settings* 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

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	Only valid if **Use default options** (this is the default) or **Use additional options** has been selected. See below. 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 *Work Files*.
Local work file	Only applies to remote environments. Only valid if **Use default options** (this is the default) or **Use additional options** has been selected. See below.

Item	Explanation
	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. See also `WFLOC` in *Direct Commands*.
Portable work file	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. In addition, this option does not apply to work files located in a remote environment on a mainframe platform. **Portable work file** is only valid if the following applies: ■ **Use default options** (this is the default) or **Use additional options** has been selected (see below). ■ **Transfer format** has *not* been selected. See also ***Work File Format*** in *Work Files*.
Load file or **Scan file** (Server)	Only valid if **Use default options** (this is the default) or **Use additional options** has been selected. See below. 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.
Browse	Not applicable to server 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). For the options available, see ***Set Additional Options*** in *Settings - Options*.
Use additional options	Used in connection with **Set** (see below).
Set	Only activated if **Use additional options** has been selected. 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 *Settings - Options*.
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	Only valid if **Use Option Workplan** (see above) has been selected and the name of a valid Workplan of the type OPTION was entered. Displays the contents of the Workplan specified.

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 *Settings*.
Set	Only activated if **Use global parameters** has been selected. If selected, the global parameters window is invoked. See ***Set Global Parameters*** (*Settings*) and `parameter-setting` (*Direct Commands*) 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 *applinfo.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

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.

Note: This parameter is no longer used and is kept for compatibility reasons only.

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

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 object is a temporary object. We recommend that you specify a permanent text object 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 object 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 object 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 Objects

Applies to remote environments only.

To specify a permanent restart text object

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

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

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 *Advanced User Unload* in the section *Advanced User*). 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 ***Advanced User Load*** in the section *Advanced User*). 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

This functions 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 *Advanced User Unload* in the section *Advanced User*). 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 ***Advanced User Load*** in the section *Advanced User*). 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

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 *Types of Workplan* in the section *Workplans*.
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

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	Sorts Workplans by the columns **Name**, **Type**, **Description**, **User** or **Date**. 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**.
New Workplan	Creates a new Workplan. 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: ■ 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. For information on the syntax used, see the section *Direct Commands*.
Edit	Modifies an existing Workplan. Select the Workplan required and double-click on it. Alternatively, select the Workplan required and, from the context or **Object** menu, choose **Edit**.
Open Workplan	Selects and opens a Workplan from a list of all Workplans available. 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. For information on the syntax used, see the section *Direct Commands*.
Delete	Deletes a Workplan. Select the Workplan required and, from the context or **Object** menu, choose **Delete**.
Check	Verifies that the correct syntax is used in a Workplan. Select the Workplan required and, from the context or **Object** menu, choose **Check**. 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.

Option	Explanation/Instruction
	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 objects 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 object 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.

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

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 ***Name*** in *Name, Date and Time Specification*. 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 ***Name***. 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 ***Natural Library Object Details***.
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 ***Natural Library Object Details***.

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 *Name, Date and Time Specification*. 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 *Natural Library Objects* above.
FNR	See **FNR** in *Natural Library Objects* above.
Password	See **Password** in *Natural Library Objects* above.
Cipher key	See **Cipher key** in *Natural Library Objects* 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
	considered Natural library objects. In other system environments, DDMs are considered Natural programming objects, which are stored in Natural libraries. If the FDDM system file has been activated, see also **Use FDDM file for processing DDMs** in *Settings - Options*.
Error messages	User-defined error messages.
Shared resources	Any non-Natural file that is used in a Natural environment and is maintained in the Natural library system. Note that shared resources are not defined in remote environments located in mainframe systems.
Name	See **Name** in *Natural Library Objects* above.
S/C-Kind	The kind of Natural programming object: **Src** Source objects only. **Gp** Generated programs (cataloged objects) only. **Any** All source and/or cataloged objects (generated programs). This is the default. **Stowed** All STOWed objects: source and cataloged objects with identical date and time. **Both** Both source and cataloged objects if both exist. **Note:** **Stowed** and **Both** are valid for the unload function only.
Predict set	Only applies to the unload and find functions and if Predict is installed. 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. If the **Predict set** option is selected, the following applies: ■ 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. For detailed information on Predict sets, refer to the Predict documentation.
Set number	Only applies if **Predict set** is selected. A one- or two-digit number that identifies the retained set to be used.
Set library	Only applies if **Predict set** is selected. 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.

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 *System Variables* 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 *Natural Library Objects* 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 ***Natural Library Object Properties*** 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 ***Natural Library Object Exceptions*** 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 ***Name*** in *Name, Date and Time Specification*.
Natural version	The Natural version of the Natural programming objects. You can also specify a range of versions: see ***Name***.
Mode	The programming mode of the Natural programming objects: **Structured** — Structured mode only. **Report** — Reporting mode only. **Any** — No mode check performed. This is the default.
DDM DBID	Not valid in remote environments located on mainframe platforms. The database ID (DBID) of the data definition modules (DDMs). Valid entries are: 1 to 65535 or 0 (all DBIDs)
DDM FNR	Not valid in remote environments located on mainframe platforms. The file number (FNR) of the DDMs: Valid entries are: 1 to 65535 or 0 (all FNRs).
Date: **Select all objects**	Selects all objects, regardless of their date.
Date: **Select objects modified between/and**	Selects all objects with a save or catalog date and/or time within the range specified in the **Date** boxes. 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 ***Date*** and ***Time*** in *Name, Date and Time Specification*. Special dates allowed are: `TODAY`, `YESTERDAY`, `MONTH` and `YEAR`.
Date: **Select objects modified on**	Selects 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 ***Date*** and ***Time***. Special dates allowed are: `TODAY` and `YESTERDAY`.
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 *Natural Library Object Properties*.
Natural version	See **Natural version** in *Natural Library Object Properties*.
Mode	See **Mode** in *Natural Library Object Properties*.
DDM DBID	See **DDM DBID** in *Natural Library Object Properties*.
DDM FNR	See **DDM FNR** in *Natural Library Object Properties*.
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 ***Date*** and ***Time*** in *Name, Date and Time Specification*. 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 ***Date*** and ***Time***. 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

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 *Natural System Error Messages* above.
DBID	See **DBID** in *Natural System Error Messages* above.
FNR	See **FNR** in *Natural System Error Messages* above.
Password	See **Password** in *Natural System Error Messages* above.
Cipher key	See **Cipher key** in *Natural System Error Messages* above.
S/L-Kind	See **S/L-Kind** in *Natural Library Object Details*.
Language codes	See **Language codes** in *Natural Library Object Details*.
Exceptions	Invokes an extra window where you can specify exceptions to the selection of Natural system error messages: see ***Natural System Error Message Exceptions***. 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

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 ***Name*** in *Name, Date and Time Specification*.
DBID	Only applies to the unload function. The database ID where the Natural command processor sources are stored. For details, see the ***SYSNCP Utility*** in the *Utilities* 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 ***SYSNCP Utility*** in the *Utilities* 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 ***Name***.
Exceptions	Invokes the **Exceptions** window where you can specify exceptions to the selection of Natural command processor sources: see ***Natural Command Processor Source Exceptions*** 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 ***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.

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 *Name* in *Name, Date and Time Specification*.
Name	The name of a Natural command processor source or a range of names: see *Name*.

24 Object Specification - Natural DDMs

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 ***Name*** in *Name, Date and Time Specification*.
Details	Invokes an additional window where you can enter further object specifications: see ***Natural DDM Details***.
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 ***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.

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 ***Name*** in *Name, Date and Time Specification*.
DDM DBID	See **DDM DBID** in *Natural Library Object Properties*.
DDM FNR	See **DDM FNR** in *Natural Library Object Properties*.
Date: **Select all objects**	Selects all objects, regardless of their date.
Date: **Select objects modified between/and**	See **Date** in *Natural Library Object Properties*.
Date: **Select objects modified on**	See **Date** in *Natural Library Object Properties*.
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 Natural DDMs: see ***Natural DDM Exceptions***. 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 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 ***Name*** in *Name, Date and Time Specification*.
User ID	See **User ID** in *Natural Library Object Properties*.
DDM DBID	See **DDM DBID** in *Natural Library Object Properties*.
DDM FNR	See **DDM FNR** in *Natural Library Object Properties*.
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*.
Date: **Exclude objects modified on**	See **Date** in *Natural Library Object Exception Properties*.
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

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 NATPARM parameter file, 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: *NATROOT*, *NATBIN*, *NATERR*, *NATSAG*, *PARM_PATH*, *PROFILE_PATH*, *TEXT_PATH*, *TMP_PATH*. 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 *Name, Date and Time Specification*.
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 *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: **Select all objects**	Selects all objects, regardless of their date.
Date: **Select objects modified between/and**	See **Date** in *Natural Library Object Properties*.
Date: **Select objects modified on**	See **Date** in *Natural Library Object Properties*.
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	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*. 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-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 *Natural Library Object Exception Properties*.
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 *Name, Date and Time Specification*.
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 *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	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 *Name, Date and Time Specification*.

Item	Explanation
	To choose a name from a selection list of all libraries available, open the drop-down list box.
Exceptions	Invokes an extra window where you can specify exceptions to the selection of Natural-related objects: see *Natural-Related Object Exceptions - Mainframes*. 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-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

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 *Name, Date and Time Specification*.
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 *External Files* above.
Object name	See **Object name** in *External Files* 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 *Natural Library Object Properties*.
Date: **Select objects modified on**	See **Date** in *Natural Library Object Properties*.
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 ***External File Exceptions***. 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 *External Files* above.
Object name	See **Object name** in *External Files* 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*.
Date: **Exclude objects modified on**	See **Date** in *Natural Library Object Exception Properties*.
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:

30 Settings - Options

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

Item	Explanation
Transfer format	Only activated if **Use default options** (this is the default) or **Use additional options** has been selected. See below. Unload: The data to be unloaded is written in Transfer format to the work file. Load and scan: The data to be loaded or scanned are expected to be in Transfer format.
Local work file	Only applies to remote environments. 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. See also `WFLOC` in *Direct Commands*.
Portable work file	Not applicable to work files located in a remote environment on a mainframe platform. 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. **Portable work file** is only activated if the following applies: ■ **Use default options** (this is the default) or **Use additional options** has been selected (see below) and ■ **Transfer format** has *not* been selected. If **Portable work file** has been selected, the work file is written or read in portable format. See also ***Work File Format*** in *Work Files*.
Fixed length	Only available with the unload function. See `FIXEDLENGTH` in *Direct Commands*.
Unicode work file	Only applies to the unload function and if **Transfer format** has been selected. 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. If a Unicode work file is specified, you cannot use the transfer options **Use conversion table**, **Substitute line references** and **Incorporate free rules**.
Unload file or **Load file** or	Only activated if **Use default options** (this is the default) or **Use additional options** has been selected (see below). The name of the work file to be used for the function. See also ***Work Files***.

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	Writes a report of the objects processed to Work File 4. This is the default setting for object processing except for the find function. In remote environments, the report data is written to a Natural text object that is stored in the Workplan library. To display a report, select **Show Report File** from the **Options** menu. See also *Reports* in *Tools*.
Start new report	Only activated if **Write report** has been selected. Deletes the contents of Work File 4 before a new report is written.
Use Report option in Find commands	Only applies to the find function and if **Write report** has been selected. 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. 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 *Tools*) will only display old report data of a previous find, unload, load or scan function, if performed.
Report file	Only available in local environments and if **Write report** has been selected. The name of a report file. Choose the **Browse** button to select a name from a directory.
FDIC	Only applies if Predict is installed. With the **FDIC** option, you specify the Predict file (FDIC) used for processing XRef data: **DBID** The database ID where the FDIC file is located. **FNR** The file number where the FDIC file is located. **Password** Optional. The Adabas password of the Adabas file where the FDIC file is located. **Cipher key** Optional. The cipher code of the Adabas file where the FDIC file is located.
FSEC	Only applies if Natural Security is installed. With the **FSEC** option, you specify the Natural Security data file (FSEC) used for security checks:

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

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	Only applies to the load function. When this option is set, restart information is provided for the restart load function. See also ***Restart Load*** in *Functions*.
Restart file	Only applies to the load function in local environments and if **Write restart information** has been selected. The name of the work file to be used for the restart data: Work File 6 (default setting) or the `restart-file` specified. Choose the **Browse** button to select a name from a directory. For further information, see ***Restart Load*** in *Functions*.
Check version	Only applies when loading objects in a mainframe environment. 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.
ADAFDU file	See `ADAFDUWORKFILE` in `option-setting` in *Direct Commands*. Choose the **Browse** button to select a name from a directory.

Item	Explanation
Number of objects to process	Specifies the number of objects to be processed. 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. **Note:** If a cataloged Natural object is processed directly after the source object of the same name, they are considered one object.
Use FDDM file for processing DDMs	Only applies in environments where the FDDM system file has been activated in the NATPARM parameter file. 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. Specify the library SYSTEM and the Natural object type `V` (see ***Natural Library Object Details*** in *Object Specification*) for processing DDMs. If used with the load function, all DDMs are loaded into the FDDM system file. In this case, the parameter `NEWLIBRARY` is ignored. See also the **syntax diagram** of the *`option-clause`* in *Direct Commands*.

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	In this text box, you can enter the name of the code page to be used for the load function. 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**. If you enter `*CODEPAGE` as the code page name, the value assigned to the system variable `*CODEPAGE` is used (see the *System Variables* documentation). If no code page name is specified, the source objects are converted with the code page used when unloading them.	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	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**). 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**).	Unload Load
System table	Only activated if **Use conversion table** has been selected. Unload: Converts data to EBCDIC format by using the internal Natural conversion table. Load: Converts data to ASCII format by using the internal Natural conversion table.	Unload Load
User-defined table	Only activated if **Use conversion table** has been selected. Specifies that a user-defined table is to be used for conversion. 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. 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. 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.	Unload Load
Table name for Load function	Only activated if **User-defined table** has been selected. Enter the name of a user-defined conversion table.	Load
Table name for Unload function	Only activated if **User-defined table** has been selected. Enter the name of a user-defined conversion table.	Unload
Substitute line references	Only applies if source-code line numbers are used for statement references. 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.	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 *not* 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 *Data Area Editor* in the *Editors* 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

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 *Set Global Parameters* below.
Set (global parameters)	Only activated if **Use global parameters** has been selected. Invokes the **Unload/Load Parameters** window. See *Set Global Parameters* 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 *Workplans*.
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 *Name, Date and Time Specification* 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 *Name, Date and Time Specification*, 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***.
`Language code`	Only applies when processing Natural system error messages or user-defined error messages. **Value/New Value:**

Parameter	Explanation
	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**. 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.
`Error number difference`	Only applies when processing Natural system error messages or user-defined error messages. 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. Example: 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.
`FDT DBID`	**Value/New Value:** A valid database ID (DBID) for the Adabas FDT.
`FDT FNR`	**Value/New Value:** A valid file number (FNR) for the Adabas FDT.
`External path`	**Value/New Value:** The name of the path for external files.

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

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 *Direct Commands*
`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 (;).	*Basic Command Syntax*
`TYPE SELECTION`	Selection criteria for objects. This Workplan can be used in Object Handler Workplan commands.	`select-clause`

Valid Headers	Contents	Related Topic in *Direct Commands*
`TYPE LIST`	A list of objects. This Workplan can be used in Object Handler Workplan commands.	*`select-clause`* ***Object List - LIST Workplan***
`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.	*`parameter-setting`*
`TYPE OPTION`	Options for the unload or load function, for example, report settings. This Workplan can be used in Object Handler Workplan commands.	*`option-setting`*
`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 object 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 } password ]
  [ { PSW      }          ]
)
```

The syntactical options are explained in the following section:

Keyword Explanation

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

Keyword	Values	Default Value
workplan-name	The name of the Natural text object 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

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
	value	All items with names equal to *value*.
	* >	All items.
	?	All items with any single character for each question mark (?) entered.
Leading Characters	*value**	All items with names that start with *value*. Example: AB* Selected: AB, AB1, ABC, ABEZ Not selected: AA1, ACB
Wildcard	*value*?	All items with names that start with *value* and end with any single character for each question mark (?) entered. Example: ABC? Selected: ABCA, ABCZ Not selected: AXC, ABCAA
	value?*value*?	All items that match *value* combined with asterisk (*) and question mark (?) in any order. Example: A?C*Z Selected: ABCZ, AXCBBBZ, ANCZ Not selected: ACBZ, ABDEZ, AXCBBBZA
	*value**value*?	
	value*?*value	
Start Value	*value*>	All items with names greater than or equal to *value*. Example: AB> Selected: AB, AB1, BBB, ZZZZZZZ Not selected: AA1, AAB
End Value	*value*<	All items with names less than or equal to *value*. 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	*YYYY-MM-DD*	All items with a date equal to *YYYY-MM-DD*. Example: 2003-02-15
Leading characters	*value**	All items with a date that starts with *value*. Example: 2002* Selected: 2002-01-01, 2002-12-31 Not selected: 2001-12-31, 2003-01-01
Start value	*value*>	All items with a date greater than *value*. 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 *value* (see below).
End value	*value*<	All items with a date less than *value*. 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 *value* (see below).
Special Dates		
TODAY (+/-*nnnn*)		All items with the date of the current day. The day can be followed by +*nnnn* or -*nnnn* where *nnnn* has a maximum of 4 digits. The resulting date is computed as the date of the current day plus or minus *nnnn* 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	*HH*:*II*:*SS*	All items with a time equal to *HH*:*II*:*SS*. Example: 14:15:16
Leading characters	*value**	All items with a time that starts with *value*. Example: 13:* Selected: 13:00:00, 13:10:53, 13:59:59 Not selected: 12:59:59, 14:00:00

34 Work Files

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

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 *Tools*.
Work File 3	Used for internal reports. Contains scan and find results.
Work File 4	Used if the option **Write report** (see *Settings - Options*) 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 *Functions*) 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 ***Traces*** in *Tools*.
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 *Settings - Options*), 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 ***Transfer Work File*** in *Tools*.
Work File 1	server system	If the option **Local work file** is *not* set (see *Settings - Options*), 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 ***Transfer Work File*** in *Tools*. Note that in a mainframe environment work files must be specified properly. See also *Natural User Access Method for Print and Work Files* in *Configuring Natural* in the documentation *Natural Operations for Mainframes*.
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:

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

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:

```
{ (selection-workplan)
  (list-workplan)
  object-selection
  delete-instruction-selection }
```

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]      [ { PASSWORD } password ]            ]
[ [CIPHER cipher]       [ { 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 ddm-dbid] [DDMFNR ddm-fnr]
[NATVERS natural-version]
[ DATE date [DATECHECK date-check]                           ]
[ [FMDATE date-from] [TODATE date-to]                        ]
[ [DATECHECK date-check]                                     ]
[ [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 ddm-dbid] [DDMFNR ddm-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
object-name	A valid object name or a range of names. If *object-name* contains blank characters, it must be enclosed in double quotation marks (" "). See also ***Name*** in *Name, Date and Time Specification*.	none

Keyword	Valid Values	Default Value
LIBRARY	A valid library name or a range of names. If OBJTYPE (see below) is set to D, the library name is ignored. If SETNO is specified, a range of names is not allowed. See also *Name*.	none
DBID	Not valid for DDMs on mainframes (OBJTYPE set to D; see below). A valid database ID.	0 (current FNAT/FUSER)
FNR	Not valid for DDMs on mainframes (OBJTYPE set to D; see below). A valid file number.	0 (current FNAT/FUSER)
NAME	Only applies to objects on mainframes. Not valid for DDMs on mainframes (OBJTYPE set to D; see below). A valid VSAM name.	blank (current FNAT/FUSER)
CIPHER	Only applies to objects on mainframes. Not valid for DDMs on mainframes (OBJTYPE set to D; see below). The 8-digit cipher code of the Adabas file where the objects are stored.	blank (current FNAT/FUSER)
PASSWORD or PSW	Only applies to objects on mainframes. Not valid for DDMs on mainframes (OBJTYPE set to D; see below). An 8-character Adabas password.	blank (current FNAT/FUSER)
OBJTYPE	Types of object are: D DDMs (objects on mainframes only) E User-defined error messages N Natural programming objects R Shared resources * Asterisk (all) or a valid combination. **Exception:** Object type D cannot be combined with any other type.	*
SETNO	Only applies to the unload and find functions and if Predict is installed. Not applicable to application objects (see also *Selecting Application Objects*). 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.	none

Keyword	Valid Values	Default Value
	If SETNO is specified, the value specified for *object-name* is ignored. For detailed information on Predict sets, refer to the *Predict* documentation.	
SETUSER	Only applies to the unload and find functions and if Predict is installed. Not applicable to application objects (see also ***Selecting Application Objects***). 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 *System Variables* documentation) is used.	*USER
SETLIBRARY	Only applies to the unload and find functions and if Predict is installed. Not applicable to application objects (see also ***Selecting Application Objects***). 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.	
NATTYPE	Not applicable if OBJTYPE is set to D. One or more single-character codes for Natural object types: 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	Not applicable if OBJTYPE is set to D. The kind of Natural programming objects. Valid input values are: S Source objects: objects that are only stored in source form. C Cataloged objects: objects that are only stored in cataloged form. A All source and cataloged objects. W All stowed objects: source and cataloged objects with identical date and time. B Source and cataloged objects if both exist. **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. The search result of an object check also depends on the settings of DATE (or FMDATE and TODATE) and DATECHECK.	A
MODE	Not applicable if OBJTYPE is set to D. The programming mode of the Natural programming objects. Valid input values are: A Any. R All objects in reporting mode. S All objects in structured mode.	
FMNUM	A start number of Natural error messages. Valid range: 1 to 9999.	1
TONUM	An end number of Natural 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	The kind of Natural error message text. Valid input values are: S Short text. Cannot be applied to the DELETE command (see *Basic Command Syntax*). L Long text. A Short and/or long text. B Short and long text if both exist.	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 *Name, Date and Time Specification*. Special terms allowed are YESTERDAY and TODAY. See ***Special Dates*** in *Date*. A date check can also be defined for a specified object kind (source or cataloged object): see DATECHECK.	blank (no check)
DATECHECK	Kind of object (source or cataloged object) to be checked for the date or date range defined with DATE, or FMDATE and TODATE, respectively. Valid input values are: S Check the date of source objects and (if available) the date of objects which exist only as cataloged objects. C Check the date of cataloged objects and (if available) the date of objects which exist only as source objects. The search result of the object check also depends on the setting of SCKIND.	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 *Date*. A date check can also be defined for a specified object kind (source or cataloged object): see DATECHECK.	blank (no check)

Keyword	Valid Values	Default Value
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 *Date*. A date check can also be defined for a specified object kind (source or cataloged object): see DATECHECK.	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 (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 ***Name***.	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 ***Name***.	blank (no check)
EXCEPT	All items that match the selection criteria entered before EXCEPT are checked against *all* 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
object-name	The name of a Natural-related object. If *object-name* contains blank characters, it must be enclosed in double quotation marks (" "). See also ***Name*** in *Name, Date and Time Specification*.	none
NATPATH	NATROOT NATGUI_BMP TMP_PATH NATBIN PROFILE_PATH PARM_PATH NATERR	none

Keyword	Valid Values	Default Value
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 ***Date*** and ***Time*** in *Name, Date and Time Specification*. Special terms allowed are: YESTERDAY and TODAY. See ***Special Dates*** in *Date*.	blank (no check)
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 *Natural Library Object and DDM Selection*.	

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

```
object-name
NATPATH DEBUG
[LIBRARY library-name]
[     DBID dbid [FNR fnr] ]
[NAME vsam-name]
[CIPHER cipher]
[     { PASSWORD }      password ]
      { PSW      }
[ 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
object-name	A valid debug environment name or a range of names. See also ***Name*** in *Name, Date and Time Specification*.	none
LIBRARY	A valid library name or a range of names. See also ***Name***.	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 *Natural Library Object and DDM Selection*.	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    ]
     { PSW      }
[ 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.

<table>
<tr><th>Keyword</th><th>Valid Values</th><th>Default Value</th></tr>
<tr><td>object-name</td><td>A valid profile name or a range of names.

See also Name in Name, Date and Time Specification.</td><td>none</td></tr>
<tr><td>OBJTYPE</td><td>The type of profile:<table><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.</td><td>*</td></tr>
<tr><td>DBID</td><td>A valid database ID.</td><td>0
(current FNAT)</td></tr>
<tr><td>FNR</td><td>A valid file number.</td><td>0
(current FNAT)</td></tr>
<tr><td>NAME</td><td>A valid VSAM name.</td><td>blank
(current FNAT)</td></tr>
<tr><td>CIPHER</td><td>The 8-digit cipher code of the Adabas file where the profiles are stored.</td><td>blank
(current FNAT)</td></tr>
<tr><td>PASSWORD

or
PSW</td><td rowspan="2">An 8-character Adabas password.</td><td rowspan="2">blank
(current FNAT)</td></tr>
<tr><td></td></tr>
<tr><td>EXCEPT</td><td>See EXCEPT in Natural Library Object and DDM Selection.</td><td>not applicable</td></tr>
</table>

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
object-name	A valid DL/I subfile name or a range of names. See also ***Name*** in *Name, Date and Time Specification*.	none
OBJTYPE	The type of DL/I subfile: 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 *Natural Library Object and DDM Selection*.	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

```
ERROR NATERROR
[ DBID dbid FNR fnr [NAME vsam-name] [CIPHER cipher]  [ { PASSWORD } password ] ]
                                                        { PSW      }
[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 *Natural Library Object and DDM Selection*.	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 *Natural Library Object and DDM Selection*.	

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
object-name	The name of a valid Natural command processor source or a range of names. See also ***Name*** in *Name, Date and Time Specification*.	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 *Natural Library Object and DDM Selection*.	

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
external-file-name	The name of an external file. If *external-file-name* contains blank characters, it must be enclosed in double quotation marks (" #). See also ***Name*** in *Name, Date and Time Specification*.	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 ***Date*** and ***Time*** in *Name, Date and Time Specification*. Special terms allowed are YESTERDAY and TODAY. See ***Special Dates*** in *Date*.	blank (no check)
FMDATE	A start value: The date on or after which external files were modified. The format is identical to DATE. See ***Date***. Special terms allowed are YEAR, MONTH, YESTERDAY and TODAY. See ***Special Dates***.	blank (no check)
TODATE	An end value: The date on or before which external files were modified. The format is identical to DATE. See ***Date***. Special terms allowed are YEAR, MONTH, YESTERDAY and TODAY. See ***Special Dates***.	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 EXCEPT in *Natural Library Object and DDM Selection*.	

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

```
FDT
DBID dbid
{ FNR fnr [CIPHER cipher] [ { PASSWORD } password ] }
{                         { PSW      }          ]   }
{ 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]
[COMPAPPLICATION 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]
[ COMPAPPLICATION 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  APOBJECTS application-object-name
[BASEAPPLICATION base-application-name]
[COMPAPPLICATION compound-application-name]
[LIBRARY library-name]
[object-specification]
[                        EXCEPT
                         (APOBJECT 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 ***Name*** in *Name, Date and Time Specification*.	*
APTYPE	A valid application type: B — Base application O — 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 *Natural Library Object and DDM Selection*.	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 *Name, Date and Time Specification*.	*
BASEAPPLICATION	The valid name of a Natural base application to which an application library or application object belongs. See also ***Name*** in *Name, Date and Time Specification*.	*
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 *Name, Date and Time Specification*.	*
LIBRARY	A valid library name or a range of names. If OBJTYPE is set to D (see *Natural Library Object and DDM Selection*), the library name is ignored. See also ***Name*** in *Name, Date and Time Specification*.	*
object-specification	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 *library-name* can also be applied to application objects whereas *object-name* in the EXCEPT clause is irrelevant for application objects.	not applicable

Object Selection for Delete Instructions

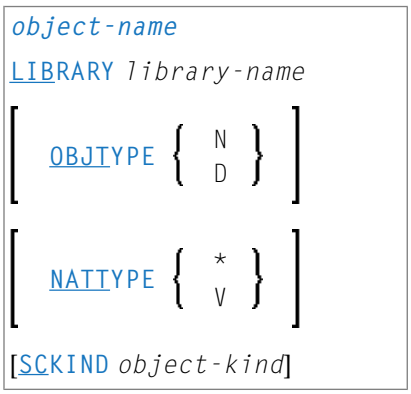
This selection is used to specify delete instructions for Natural library objects, DDMs, user-defined error messages and Natural system error messages. The delete instructions are executed when a work file of internal format is loaded in the target environment with the DELETEALLOWED option specified.

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



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
object-name	A valid object name or a start value (*value**) for a range of names such as ABC*.	none
LIBRARY	A valid library name. A range specification is *not* 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: S Source objects. If used in the *except-clause* (see *Syntax of Natural Library Object and DDM Selection*): objects that are stored only in source form.	A

Keyword	Valid Values	Default Value
	C Cataloged objects. If used in the *except-clause*: 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

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 ] ]
                                                     { PSW      }
[ 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 cipher]    [ { PASSWORD } password ] ]
                                                           [ { PSW      }          ]
```

Natural Command Processor Sources

```
PROCESSOR ncp-library-name
    [ DBID dbid FNR fnr [NAME vsam-name] [CIPHER cipher]    [ { PASSWORD } password ] ]
                                                           [ { 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 [SCKIND 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 } password ]
                                  { PSW      }
```

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

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 }     fnat-password                   ]
       {            LOADFNATPSW      }                                     ]
[      LOADFUSERDBID fuser-dbid LOADFUSERFNR fuser-fnr                      ]
       [LOADFUSERNAME fuser-vsam-name]
       [LOADFUSERCIPHER fuser-cipher]
  [    {            LOADFUSERPASSWORD }    fuser-password                  ]
       {            LOADFUSERPSW      }                                    ]
[      LOADNCPDBID ncp-file-dbid LOADNCPFNR ncp-file-fnr                    ]
       [LOADNCPNAME ncp-file-vsam-name]
       [LOADNCPCIPHER ncp-file-cipher]
  [    {            LOADNCPPASSWORD }     ncp-file-password               ]
       {            LOADNCPPSW      }                                     ]
```

```
[       [FDTDBID old-fdt-dbid FDTFNR old-fdt-fnr] NEWFDTDBID new-fdt-dbid NEWFDTFNR
        new-fdt-fnr                                                                  ]
[ERRNUMDIFF modification-of-error-message-range]
[       [LANGUAGE old-language]                                                      ]
        NEWLANGUAGE new-language
[       [DATE old-date] NEWDATE new-date                                             ]
[       [USERID old-userid]  NEWUSERID                                               ]
        new-userid
[       [TID old-terminal-id] NEWTID                                                 ]
        new-terminal-id
[       [PATH old-external-path-name]                                                ]
        NEWPATH new-external-path-name
```

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 *not* 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 or	Only applies to objects on mainframes. An FNAT Adabas password.	LOAD

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 *select-clause*) 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 *language* contains more than one language code, *new-language* must contain the same numbers of language codes. Each *language* language code is replaced by the language code in the corresponding position of *new-language*.	

Keyword	Values	Restricted to Command
	If *language* is not specified, *new-language* 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 *Name, Date and Time Specification*.	
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 *Name, Date and Time Specification*.	
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:

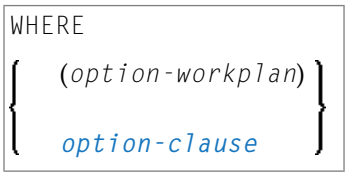
1. Parameters not applicable to the selection criterion processed are ignored.
2. LOADFNAT..., LOADFUSER... and LOADNCP... are used for the LOAD command only, and ignored otherwise.
3. LOADFNAT... is used for libraries starting with SYS (except SYSTEM).
4. LOADFUSER... is used for libraries not starting with SYS (but including SYSTEM).
5. LOADNCP... is used for Natural command processor sources.

40 option-setting

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

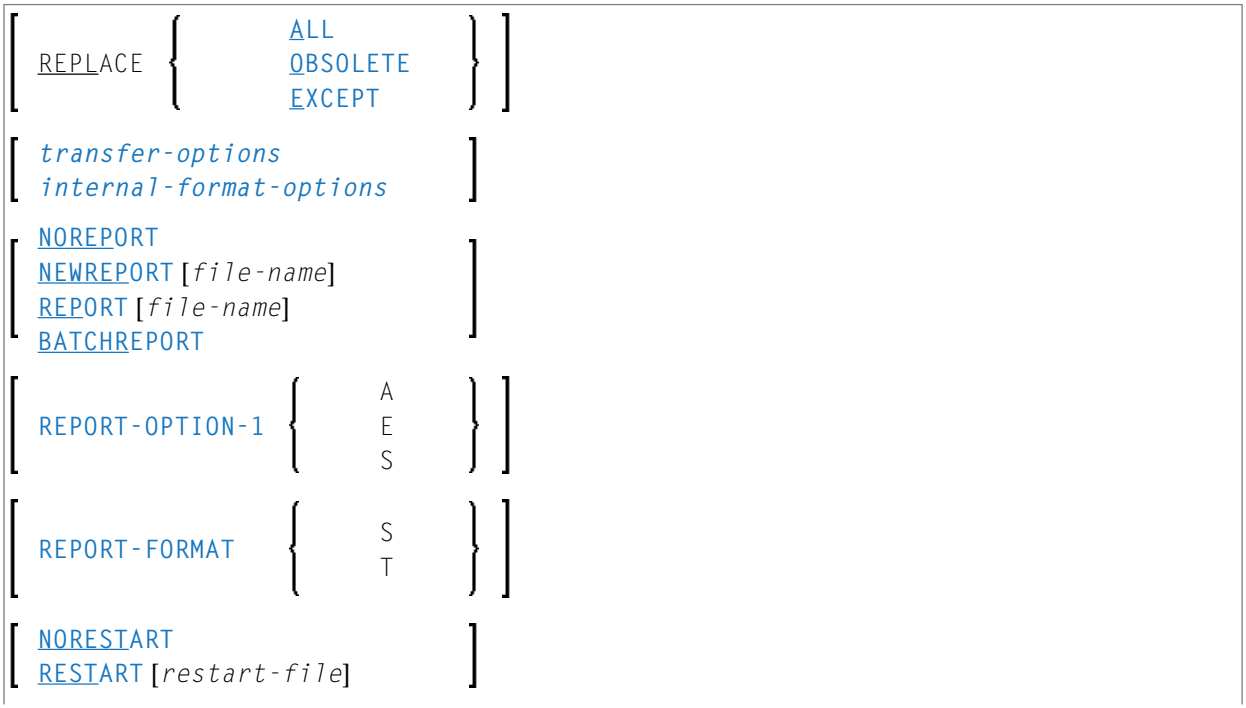


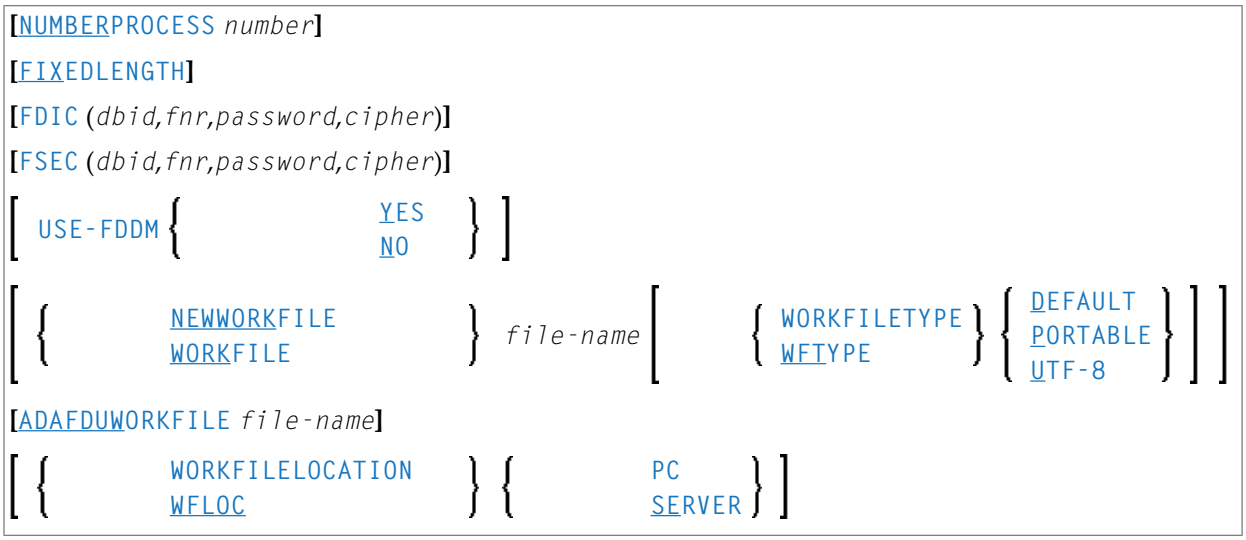
The syntax diagram that applies to *option-workplan* is shown and described in ***Referencing Workplans*** in the section *Workplans*.

The syntax of the *option-clause* is shown in the following section.

- Syntax of option-clause

Syntax of option-clause



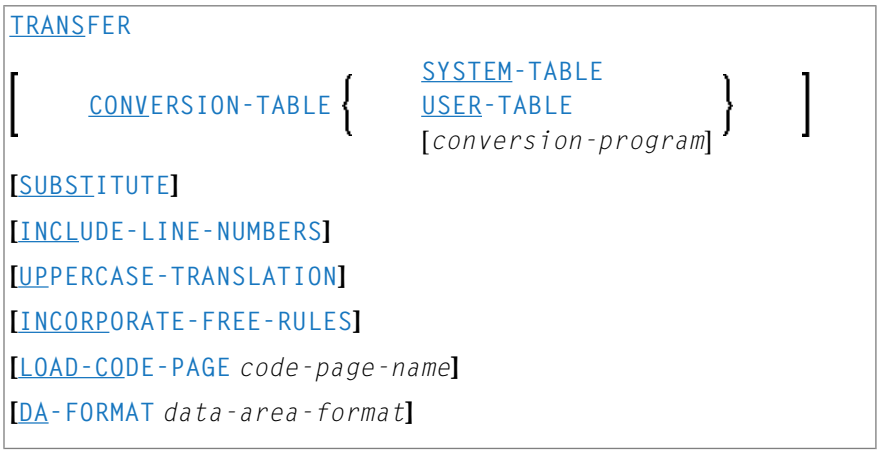


Separators

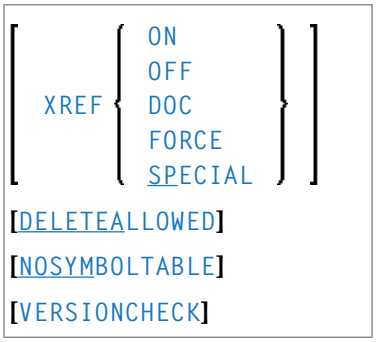
Commas must be used as separators between the values following the `FDIC` and `FSEC` keywords, or if a value is missing. For example: `FDIC (10,21,,2a)`.

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



internal-format-options



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 ***Keyword Explanation of*** *transfer-options*.	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 *file-name*. An existing file will be overwritten.	
REPORT	Specifies the report file setting: Report data is recorded and written to Work File 4 or *file-name*. This is the default setting for the commands UNLOAD, LOAD, LOADALL, SCAN, SCANALL and DELETE.	

Option	Explanation	Restricted to Command
BATCHREPORT	Not applicable in a remote Natural Development Server environment. Specifies the report setting for batch processing or when using the OBJHAPI Application Programming Interface: Report data is either written to SYSOUT or output on the screen respectively (report data is *not* written to a file).	
REPORT-OPTION-1 or REPOPT1 or REP-OPT-1	Specifies the report option to be used when a direct command is executed and a report is to be written: A Display all report items (default). E 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". S 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.	UNLOAD LOAD SCAN DELETE
REPORT-FORMAT or REPFRMT	Specifies the report format to be used when a direct command is executed and a report is to be written: S 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). T Display the target data in the unload report, i.e. the data after parameters (e.g. the library name) have been changed.	UNLOAD
NORESTART	No restart information is written to a file.	LOAD
RESTART	Restart information is written to Work File 6 or *restart-file*.	LOAD
NUMBERPROCESS	Specifies the number of objects to be processed. The LOAD or SCAN command stops execution after the number specified.	LOAD SCAN
FIXEDLENGTH	Sets the format of the unload work file to a maximum record length of fixed size. Every data record contains 256 bytes if written in internal format, or 100 bytes in Transfer format.	UNLOAD
FDIC	Specifies the system file FDIC to be used for processing: the database ID (*dbid*), file number (*fnr*), password (*password*) and cipher code (*cipher*) of the Adabas file.	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 (*dbid*), file number (*fnr*), password (*password*) and cipher code (*cipher*) 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 *Work Files*.) 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
	SERVER The work file is located on the server, in the remote environment. This is the default setting. PC The work file is located in a local Windows directory, on the client.	

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	Converts data processed in Transfer format by using either of the following conversion tables: SYSTEM-TABLE: The internal Natural conversion table. USER-TABLE: A user-defined conversion table if *conversion-program* 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. If no *conversion-program* is specified, the corresponding conversion table is used in NATCONV.INI ([ISO8859_1->EBCDIC] or [EBCDIC->ISO8859_1]).	UNLOAD LOAD SCAN
SUBSTITUTE	Replaces line references by labels during the unload in Transfer format. 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.	UNLOAD
INCLUDE-LINE-NUMBERS	Transfers line numbers during the unload in Transfer format. By default, line numbers in Natural objects are *not* unloaded.	UNLOAD

Option	Explanation	Restricted to Command
USE-LINE-NUMBER-INCREMENT or USE-LNI	UNLOAD If the option INCLUDE-LINE-NUMBERS is not specified, the line number increment of Natural source objects will be unloaded. By default, the line number increment in Natural source objects is *not* unloaded. LOAD If the line number increment was transferred, it is used to rebuild the line numbers of the Natural source objects.	UNLOAD LOAD
UPPERCASE-TRANSLATION	Translates any source code into upper case during the load in Transfer format. By default, source code in Natural objects is *not* translated.	LOAD
INCORPORATE-FREE-RULES	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.	UNLOAD
LOAD-CODE-PAGE	Specifies the code page to be used for converting object sources encoded in Unicode/UTF-8 (Universal Transformation Format, 8-bit form). 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. If you specify *CODEPAGE as *code-page-name*, the value assigned to the system variable *CODEPAGE is used (see the *System Variables* documentation). If *code-page-name* is not specified, the source objects are converted with the code page used when unloading them. If LOAD-CODE-PAGE is specified, you cannot use the options CONVERSION-TABLE and UPPERCASE-TRANSLATION.	LOAD LOADALL
DA-FORMAT	Specifies format conversion of data area sources. Possible values are: N Converts data areas to the new internal data area format. O Converts data areas to the old internal data area format. * Does not convert data areas. This is the default. See also the data area conversion options described in ***Transfer*** in *Settings - Options*.	UNLOAD LOAD

Keyword Explanation of internal-format-options

When using *internal-format-options*, you can specify the following:

Option	Explanation		Restricted to Command
XREF	Only applies if Predict is installed. Loads or unloads XRef data of cataloged Natural objects. You can specify one of the following values:		LOAD UNLOAD
	ON	UNLOAD: Unloads cataloged objects and their cross-reference data (if any). LOAD: Loads cataloged objects and their cross-reference data if cross-references exist in the work file.
	OFF	No XRef data is processed. This is the default.
	DOC	Only applies to LOAD. Loads cataloged objects and their cross-reference data (if any) only if Predict entries exist for the objects in the FDIC system file.
	FORCE	Only applies to LOAD. 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.
	SPECIAL	Only applies to LOAD. Loads cataloged objects and their cross-reference data (if any).
DELETEALLOWED	Processes delete instructions from work files when loading objects in internal format.		LOAD
NOSYMBOLTABLE	Only applies to objects on mainframes. Unloads cataloged Natural library objects without their corresponding internal Natural symbol tables.		UNLOAD

Option	Explanation	Restricted to Command
	This will reduce the amount of disk storage required. However, this is only useful for a production environment, as several application development functions which require the symbol tables will then not be available; in addition, the profile parameter RECAT=ON (see the *Parameter Reference* documentation) will not apply.	
VERSIONCHECK	Only applies to objects on mainframes. 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. VERSIONCHECK can only be used if data is loaded in internal format, that is, if the TRANSFER option is *not* specified.	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 parameter file, the default setting is YES.

The following applies when specifying the values YES or NO:

Value	Explanation
YES	UNLOAD, FIND and DELETE: 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). 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. LOAD: DDMs are loaded into the library SYSTEM located in the FDDM file. See also NATTYPE in ***Natural Library Object and DDM Selection*** in *select-clause*.
NO	UNLOAD, FIND and DELETE: DDMs are processed from the libraries specified. LOAD: DDMs are loaded into the libraries specified.

41 Examples of Using Direct Commands

This section provides examples of 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 with date 2019-10-01 as the catalog date (if both source and cataloged object exist) and the source date (if only a source object exists):

```
UNLOAD * LIB ABC OBJTYPE N SCKIND S DATE 2019-10-01 DATECHECK C
```

- 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

```
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

`MAP ENVIRONMENT=UX1 SUNNAT63 6312 SAG`	Maps to an NDV environment on a UNIX or an OpenVMS platform.

`SYSOBJH`	Invokes the Object Handler (on the Windows client) that receives the following three commands from the input data in the CMOBJIN file:
	
	`UNLOAD * LIB SAG-TEMP %`
	`WHERE TRANS WFLOC PC WORK D:\NAT-Work\w1.dat REPORT`
	
	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 *D:\NAT-Work\w1.dat*.
	
	`SHOW STATISTICS`
	
	Writes statistical data about the unloaded objects to the CMPRINT output file.
	
	`END`
	
	Terminates the Object Handler.

`UNMAP`	Unmaps the NDV environment on the UNIX or OpenVMS platform.
`MAP ENVIRONMENT=MF1 IBM2 4742 SAG`	Maps to an NDV environment on a mainframe platform.

SYSOBJH	Invokes the Object Handler (on the Windows client) that receives the following three commands from the input data in the CMOBJIN file:
	
	`LOADALL WHERE TRANS WFLOC PC %`
	`WORK D:\NAT-Work\w1.dat REPLACE ALL REPORT`
	
	Causes the Object Handler to load all objects from the work file in the local Windows directory *D:\NAT-Work\w1.dat* into the remote mainframe environment.
	
	`SHOW STATISTICS`
	
	Writes statistical data about the loaded objects to the CMPRINT output file.
	
	`END`
	
	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

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

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

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

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`.

VII SYSAPI Utility - APIs of Natural Add-On Products

47 SYSAPI Utility - APIs of Natural Add-On Products

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

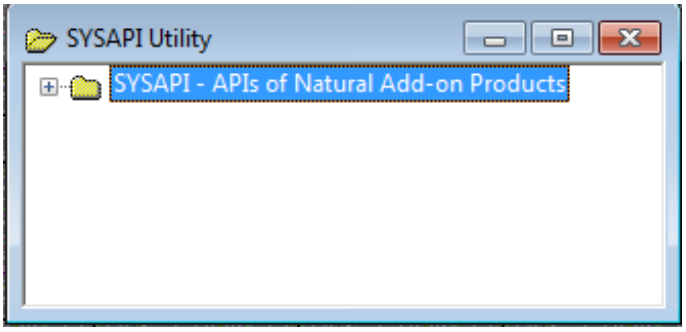
This section provides instructions for invoking and terminating the SYSAPI utility.

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.

VIII SYSCP Utility - Code Page Information

48 SYSCP Utility - Code Page Information

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: International Components for Unicode at web site ***http://site.icu-project.org/***
- ICU: Converter Explorer documentation at web site ***http://demo.icu-project.org/icu-bin/convexp***

Invoking and Terminating SYSCP

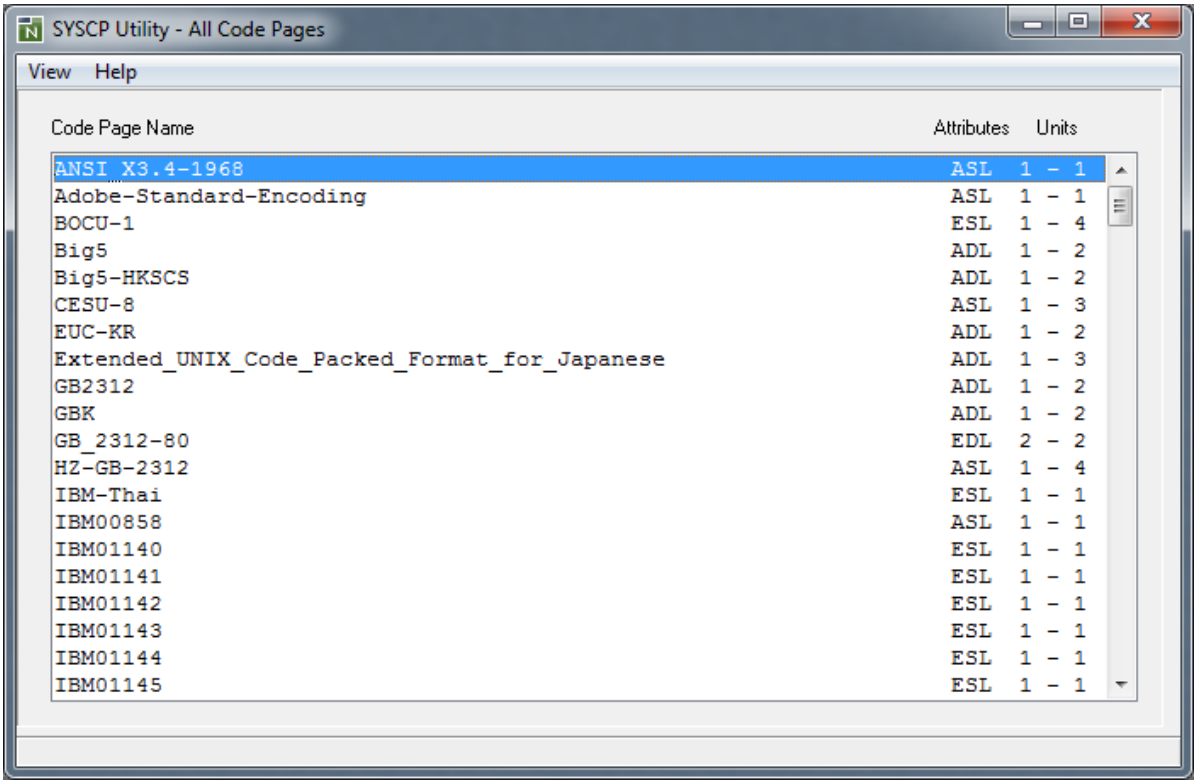
Instructions for invoking and terminating the SYSCP utility are provided in the following section.

To invoke the SYSCP utility

- Enter the following system command:

  ```
  SYSCP
  ```

  A **SYSCP Utility - All Code Pages** window similar to the example below appears:



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

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.

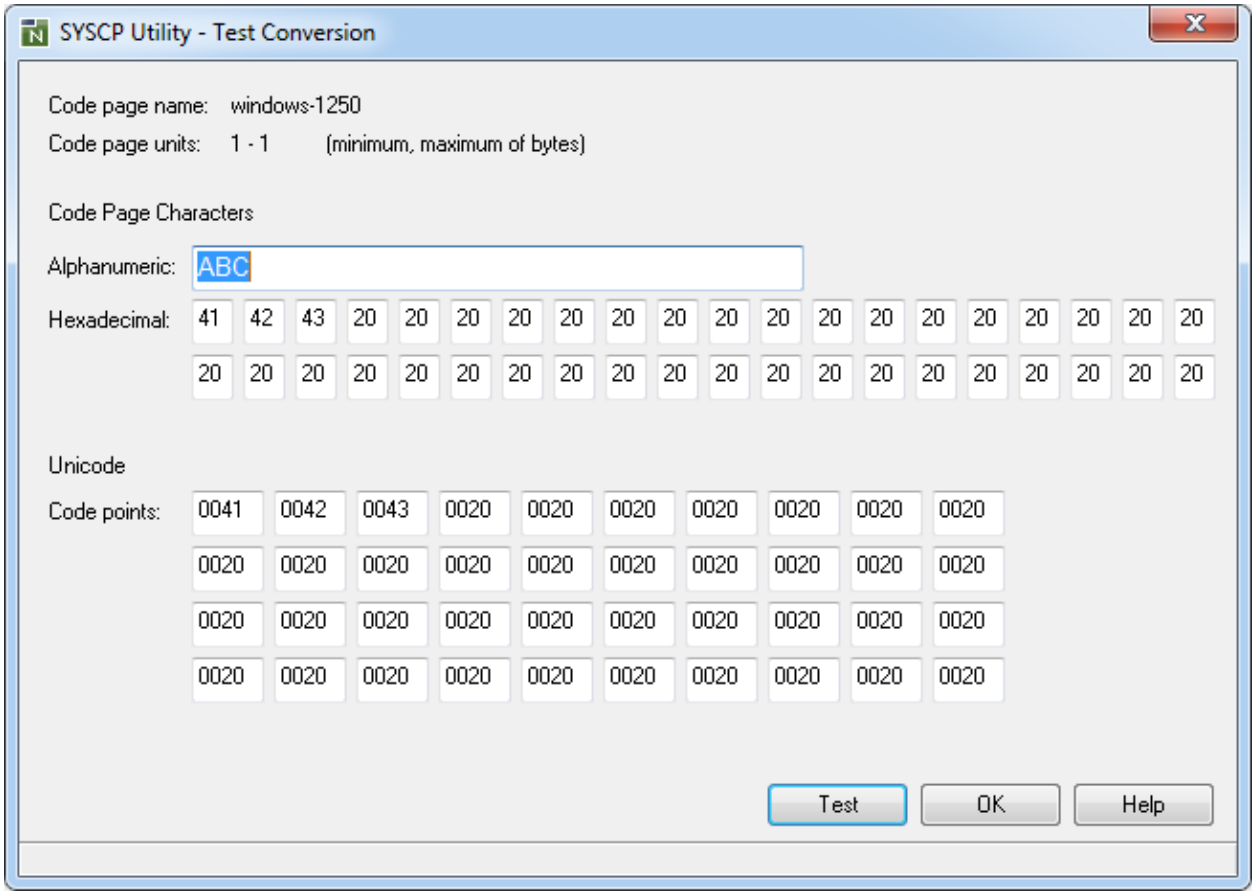
Test Conversion

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.

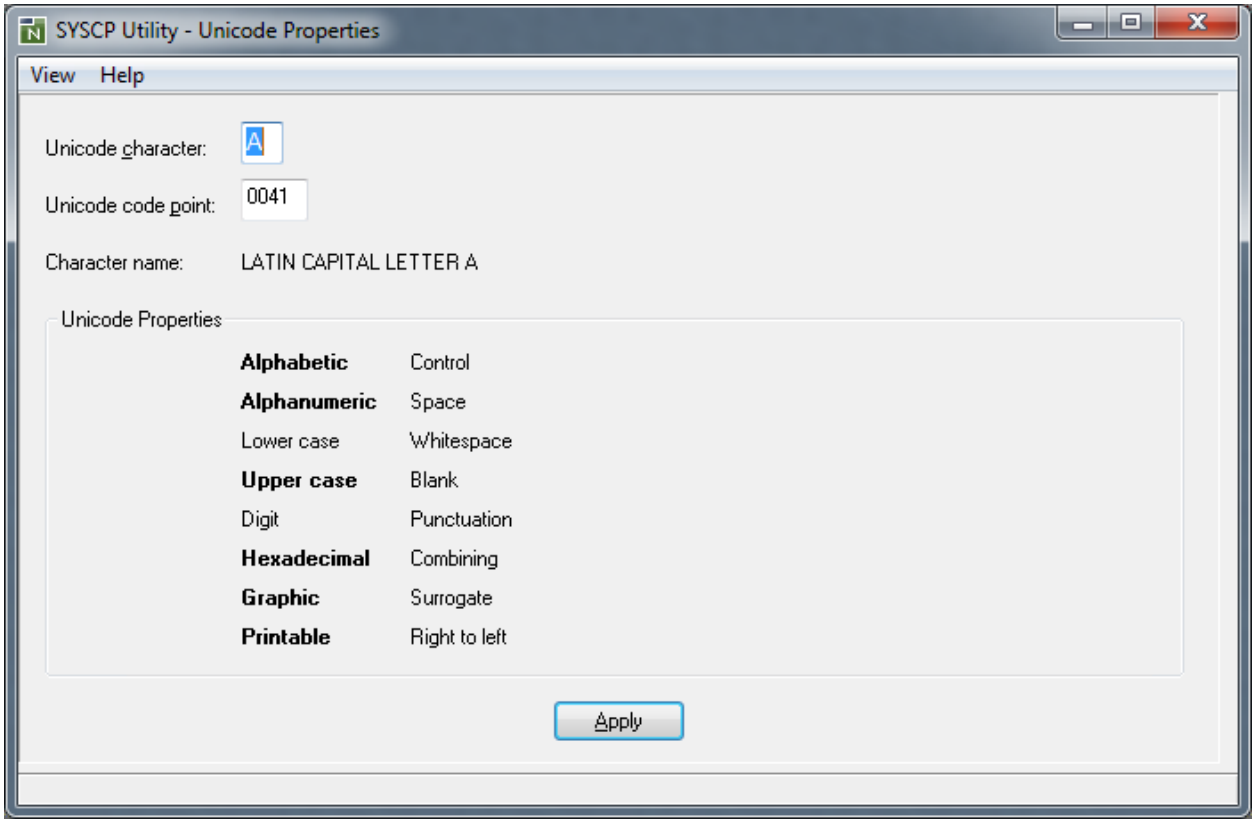
2 Choose **Test** or press ENTER.

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.

IX SYSERR Utility

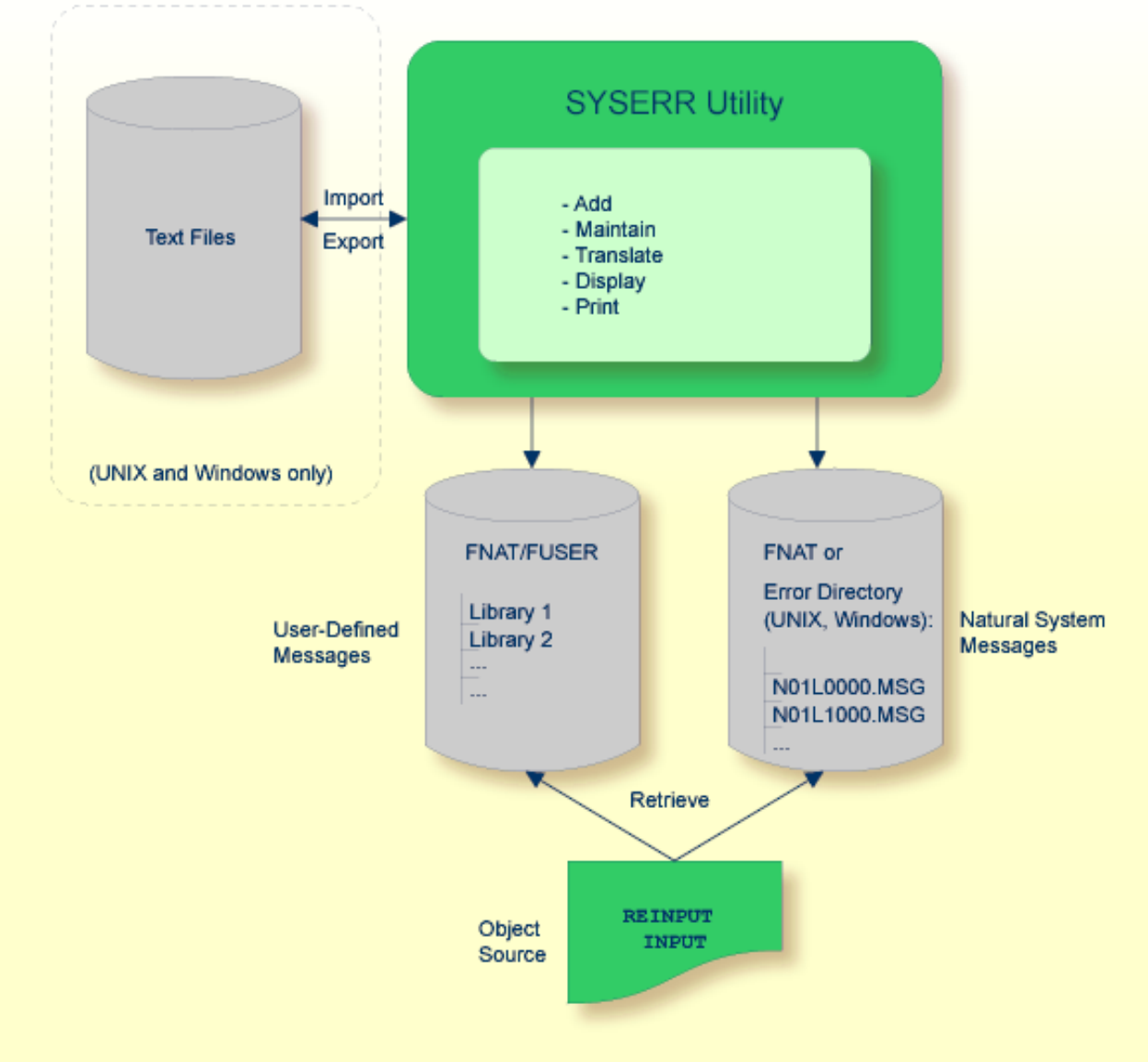
When you develop a Natural application, you may want to separate error or information messages from your Natural code and manage them separately. This makes it easy for you, for example, to standardize messages, to have predefined message ranges for different types of message, to translate messages into 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.

49 General Information on Messages

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 *n*000 (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 steplibs; if Natural Security is installed, the sequence in which the steplibs 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 test 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

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.

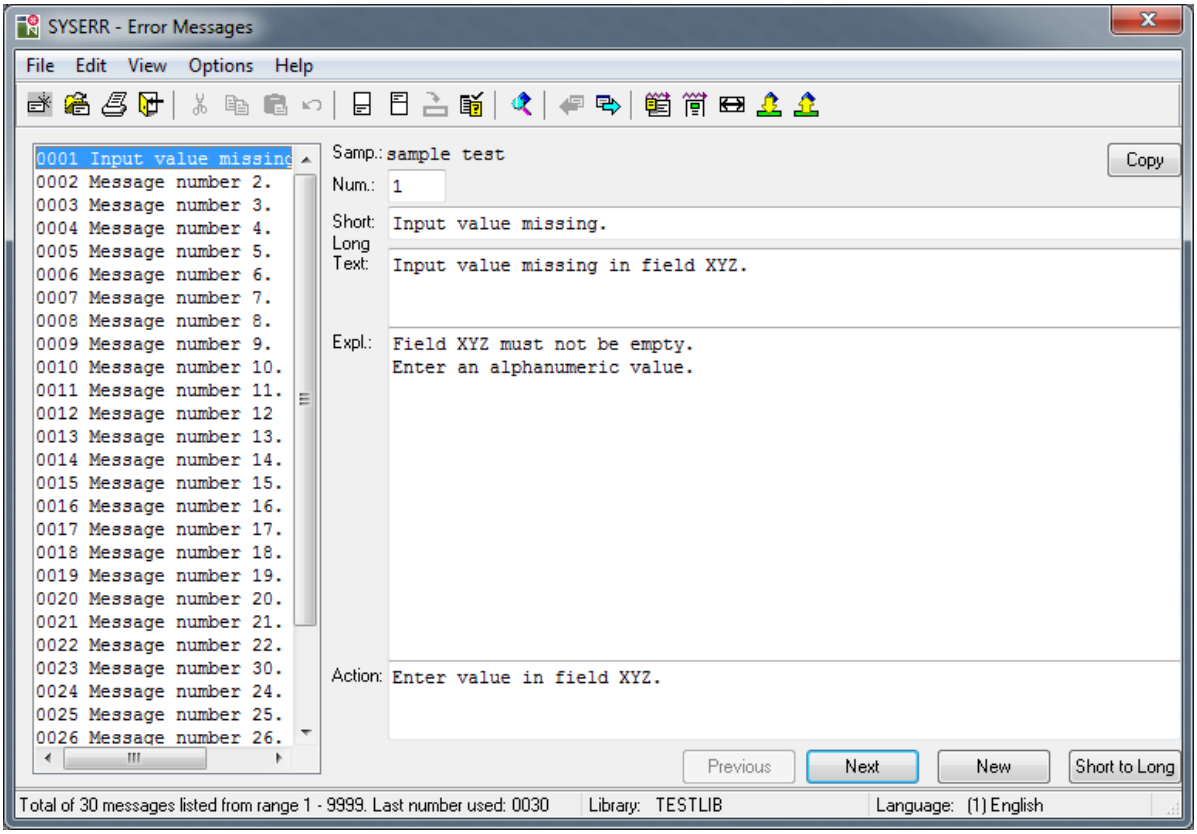
2 From the **Tools** menu, choose **Development Tools and Error Messages**.

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*.

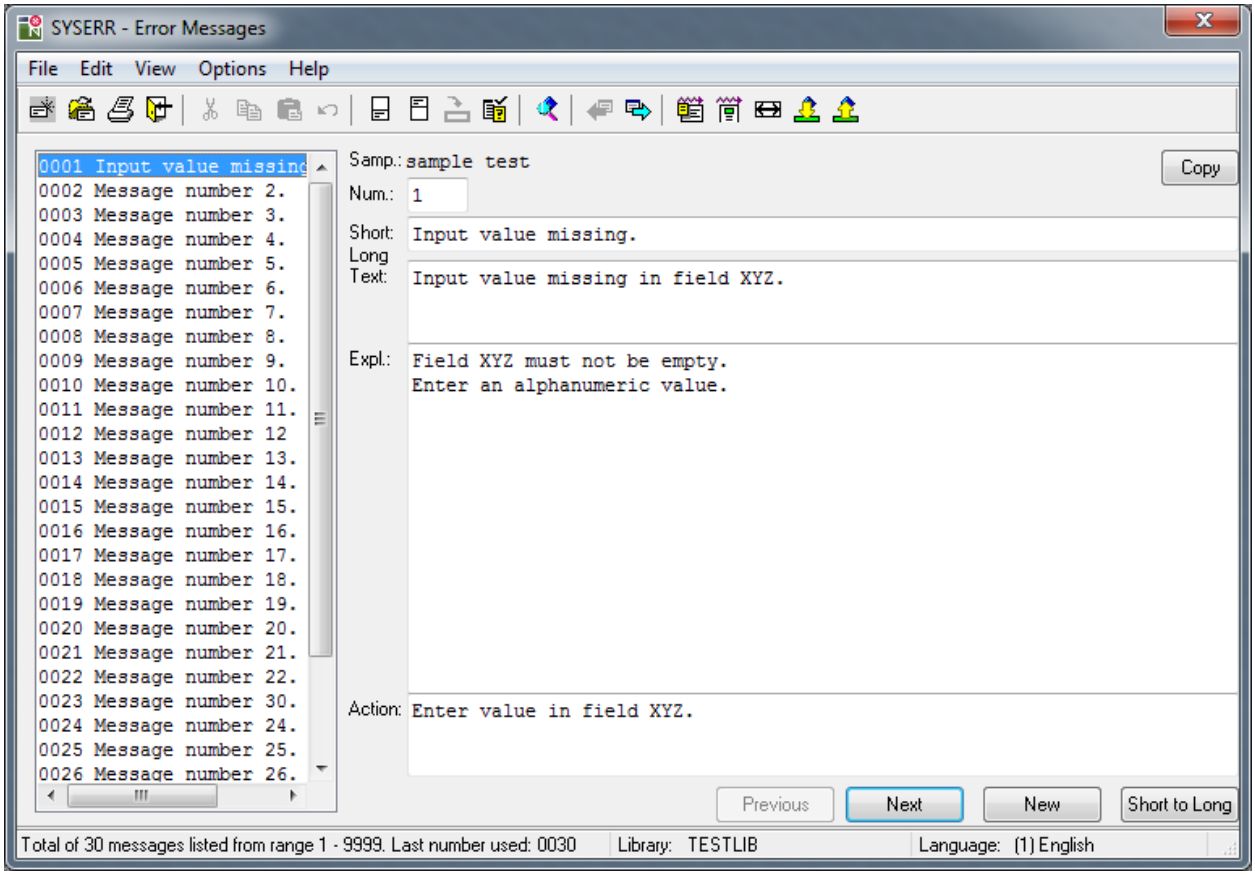
4 From the **Library** drop-down list box, select **<natsys>** or remove the library ID and leave the box blank, and, from the **Language** box, select a language.

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

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 *Options Menu*.
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.

Field	Explanation
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 *Options Menu*. To copy sample messages into the **Short** field, see **Sample** in *Options Menu* and **Copy** in *Command Buttons*. To copy text from the **Short** into the **Long** field, choose **Short to Long**.
Long	Modifiable fields displaying the long message text of the current message. **Long** consists of three input areas: **Text** Extended version of the short message text. **Expl.** Further explanation of the message. **Action** The action to be taken to resolve a problem, if relevant. The input in **Long** is optional. To copy text from the **Short** into the **Long** field, choose **Short to Long**.

Command Buttons

The following commands can be executed by using command buttons:

Command	Explanation
Copy	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. Copies the text of a sample message (see **Sample** in *Options Menu*) into the **Short** field (see *Fields*) of the current message.
Previous / Add **Previous / Update**	Toggles between **Previous** and **Add**, and **Previous** and **Update** depending on the status of the current message: **Previous** Scrolls from the current message to the previous message if no modification was made. **Add** Adds a new message to the message file. **Update** Saves the current message to the message file after modification. **Note:** You can only save a message if text was entered in the **Short** field (see *Fields*).

Command	Explanation
Next / Reset	Toggles between **Next** and **Reset**, depending on the status of the current message: **Next** Scrolls from the current message to the next message if no modification was made. **Reset** Resets modifications made to the current message and displays the original message.
New	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. 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 *Edit Menu*). Upwards searches for the next lower message number, downwards searches for the next higher message number from the current message. As an alternative to **New** or **New Message**, you can also create a new message by replacing the number in the **Num.** field.
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	Only applies to libraries and languages for which no messages exist. Opens a new library and/or adds a new language. See also *To switch libraries and/or languages*.
Open Lib/Lang	Only applies to libraries and languages for which messages already exist. Opens another library and/or changes the language. See also *To switch libraries and/or languages*.
Open File	This function does not apply for accessing data on mainframe servers. Selects an existing message file from a Natural Err directory (FNAT or FUSER) or from another directory. See also the section *Generating Message and Text Files*.
Print	Invokes the print function. A dialog box prompts you to:

Function	Explanation
	■ 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. 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 *To print all Natural system messages*.
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**.

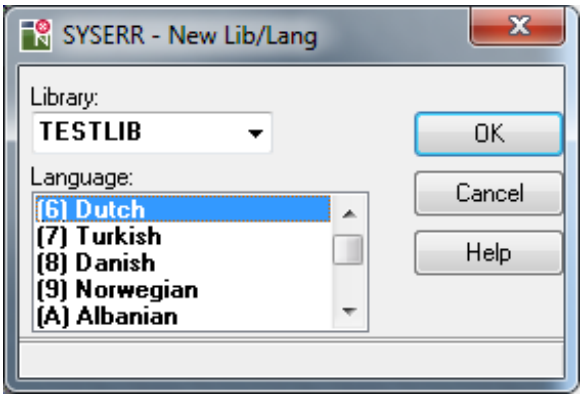
To switch libraries and/or languages

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	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. **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. 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.
Delete Selected	Removes all messages selected (highlighted) in the list box. A dialog box prompts you to confirm the action.
Delete All	Removes all messages displayed in the list box. A dialog box prompts you to confirm the action. When all messages of a message file have been deleted, the message file is deleted too.
Read All	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. 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.
Translate	Supports the creation of messages for different languages. Applies to short messages only. See *To translate languages* below.

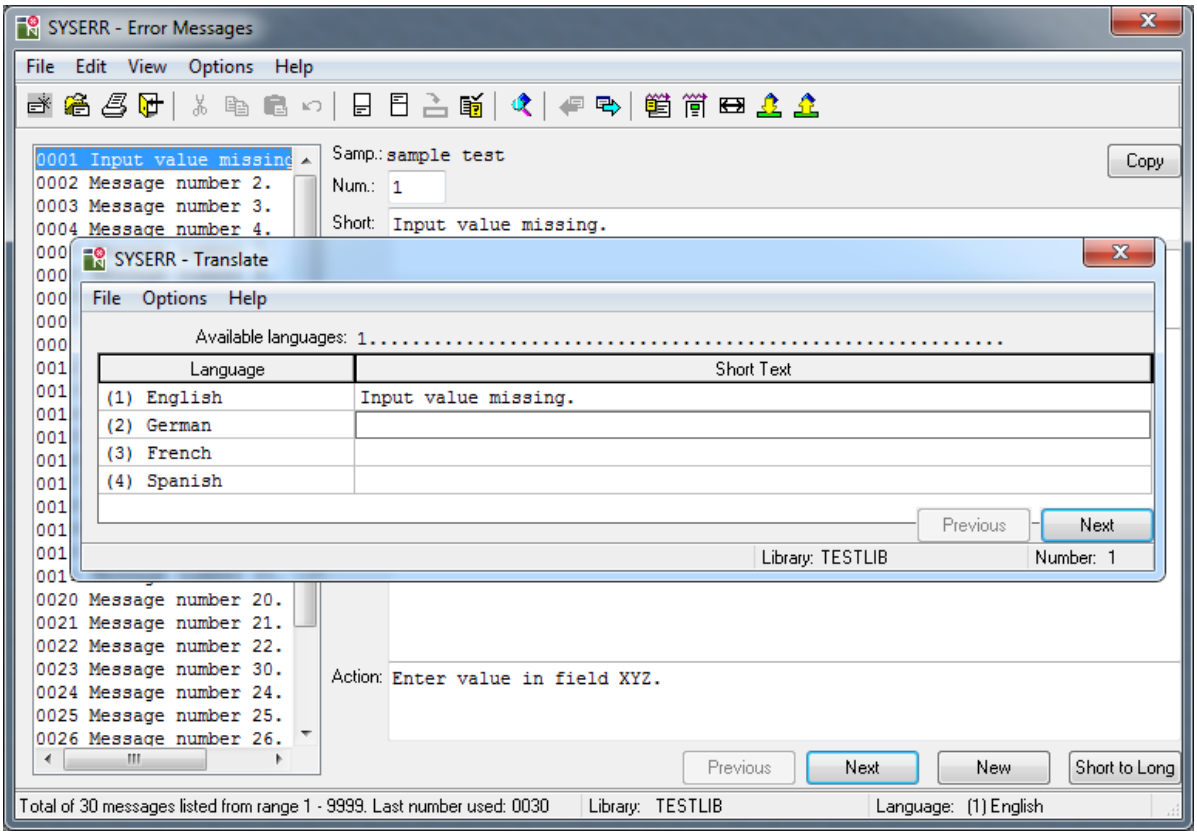
To translate languages

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 *Command Buttons*.
Reset / Next	Same as described for **Reset / Next** in *Command Buttons*.
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**.

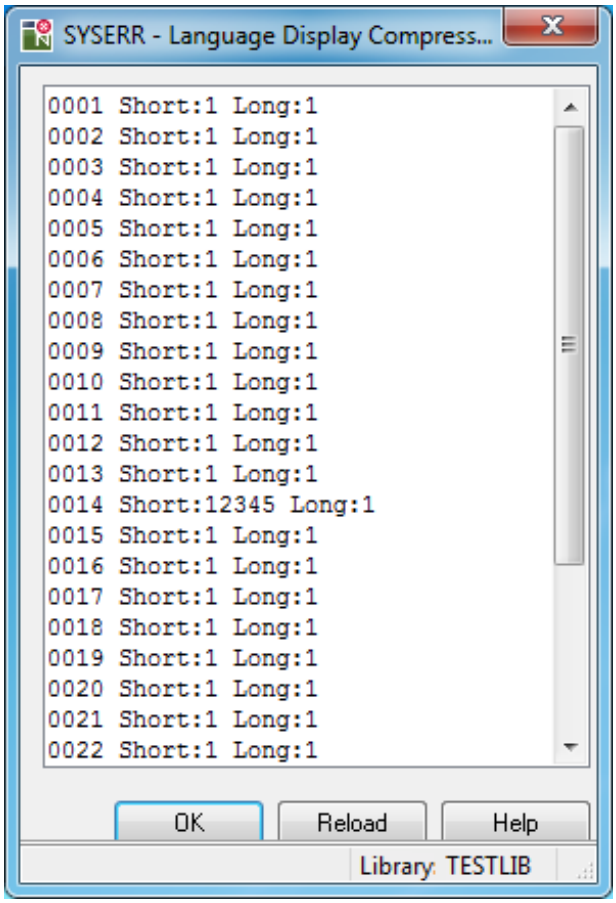
View Menu

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 *Example of Compressed*.
	Extended Short	Displays the language code(s) that exists for a short message next to the message number.

Function	Explanation
	Extended Long — Displays the language code(s) that exists for a long message next to the message number. 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.
Filter Display	Applies in connection with the **Set Filter** function. If **Set Filter** is enabled, the **Filter Display** function displays the short messages that match the filter criteria defined with the **Set Filter** function. If **Filter Display** is enabled, the filter criteria specified with the **Set Filter** function are displayed above the list box. To disable the **Filter Display** and list all messages, again choose the **Filter Display** option from the menu. See also *Filtering Messages*.
Set Filter	Defines a filter for displaying short messages and shows the filtered messages in the list box. See also *Filtering Messages*. **Set Filter** must be enabled in order to use the **Filter Display** function.

Example of Compressed



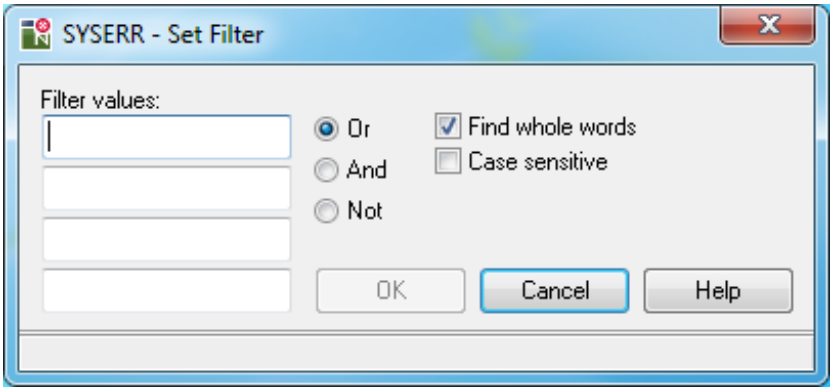
In the example above, the short text (**Short**) for message number 14 exists in English = (1), German (2), Spanish (4), and Italian (5). The long text (**Long**) for message number 14 exists in English (1).

Filtering Messages

To filter short messages

1 From the **View** menu, choose **Set Filter**.

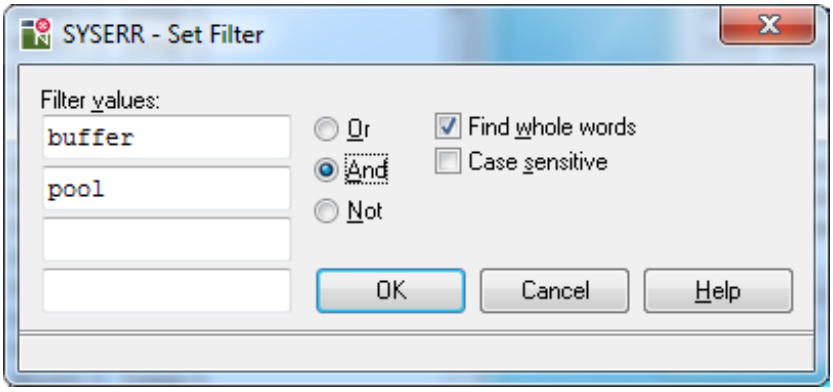
A **SYSERR - Set Filter** dialog box similar to the example below appears:



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	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. 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. You can define one sample message for each language and library. If you created a sample message, the **Samp.** field appears with the text of the sample message. To copy the text of the sample message into the **Short** field, use the **Copy** command button (see *Edit Menu*). Alternatively, you can enter `.C` in the **Short** field if this field is empty.
Layout	See ***Layout***.
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	This option does not apply to a remote environment located on a UNIX, OpenVMS or mainframe platform. If enabled, the **Short** field (see ***Fields***) 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.

<table>
<tr><th>Function</th><th>Explanation</th></tr>
<tr><td>Size</td><td>Resizes the dialog box:

<table><tr><td>Startup</td><td>Resizes the dialog box to the size it had at startup.</td></tr><tr><td>Full List Box</td><td>Resizes the dialog box to display the short message in full length.</td></tr></table></td></tr>
<tr><td>Confirm Window</td><td>Enables/disables a pop-up window to confirm:

■ 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.</td></tr>
<tr><td>Import Text File</td><td>This function does not apply to a remote environment located on a mainframe platform.

Imports a text file and converts it into a message file. Note that you always need to specify the full path of a file.

<table><tr><td>From (text file)</td><td>The name of the text file from which the message file will be generated.</td></tr><tr><td>To (message file)</td><td>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.

For user-defined messages, the file name must be <code>NnnAPMSL.MSG</code>. For Natural system messages, the file name must be <code>NnnLmmmm.MSG</code>.</td></tr></table>
For further information on file formats and generating message and text files, see the relevant section.</td></tr>
</table>

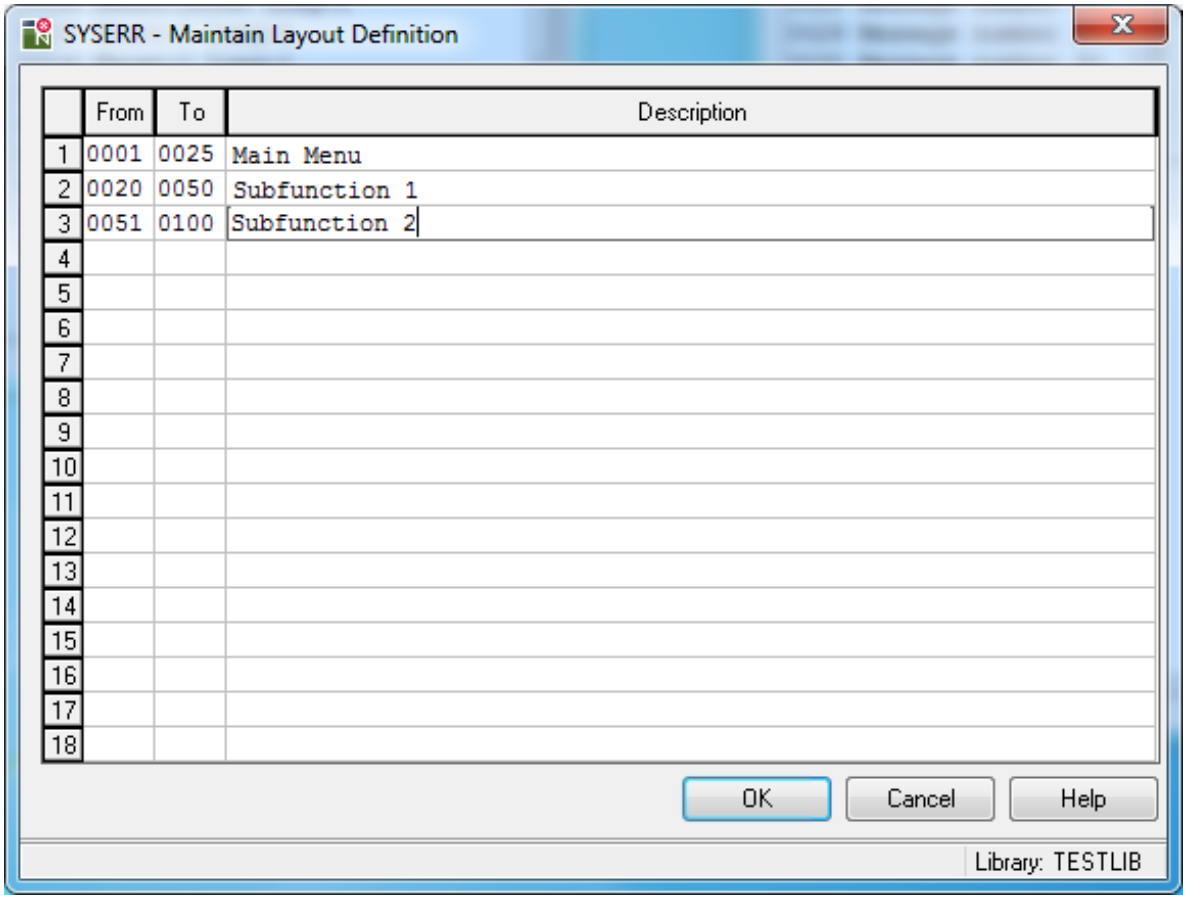
Function	Explanation	
Export Message File	This function does not apply to a remote environment located on a mainframe platform. Exports a message file and converts it into a text file. Note that you always need to specify the full path of a file.	
	From (message file)	The name of the message file from which the text file will be generated. Default is the full path name of the current library and message file. For user-defined messages, the file name must be `N`*`nn`*`APMSL.MSG`. For Natural system messages, the file name must be `N`*`nn`*`L`*`mmmm`*`.MSG`.
	To (text file)	The name of the text file that will be generated.
	For further information on file formats and **generating message and text files**, see the relevant section.	

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

Toolbar Button	Menu Item/Command Button	Menu
	**Copy**	**Edit**
	**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.

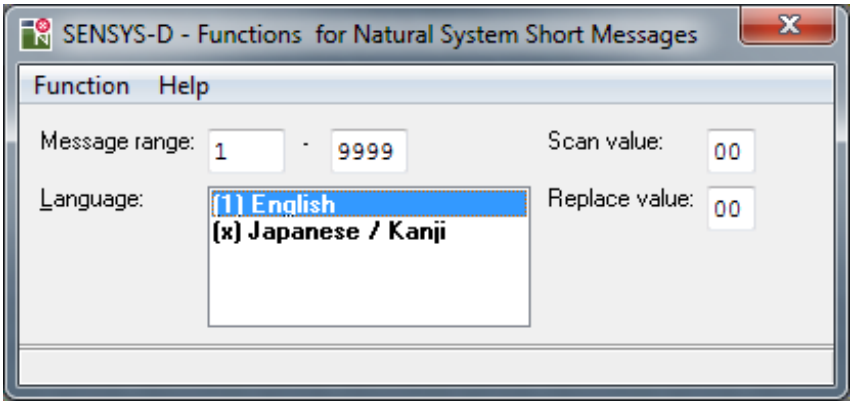
2 From the **SYSERR** library, select the **Dialogs** subnode and then **SENSYS-D**.

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 *Object Handler* 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

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 *`library-ID`*	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`	`NO MESSAGE TEXT DEFINED!`
	
	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:
	
	`REINPUT *0010`
	
	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.

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 reconverting 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.

X SYSEXT Utility - Natural Application Programming Interfaces

56 SYSEXT Utility - Natural Application Programming Interfaces

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

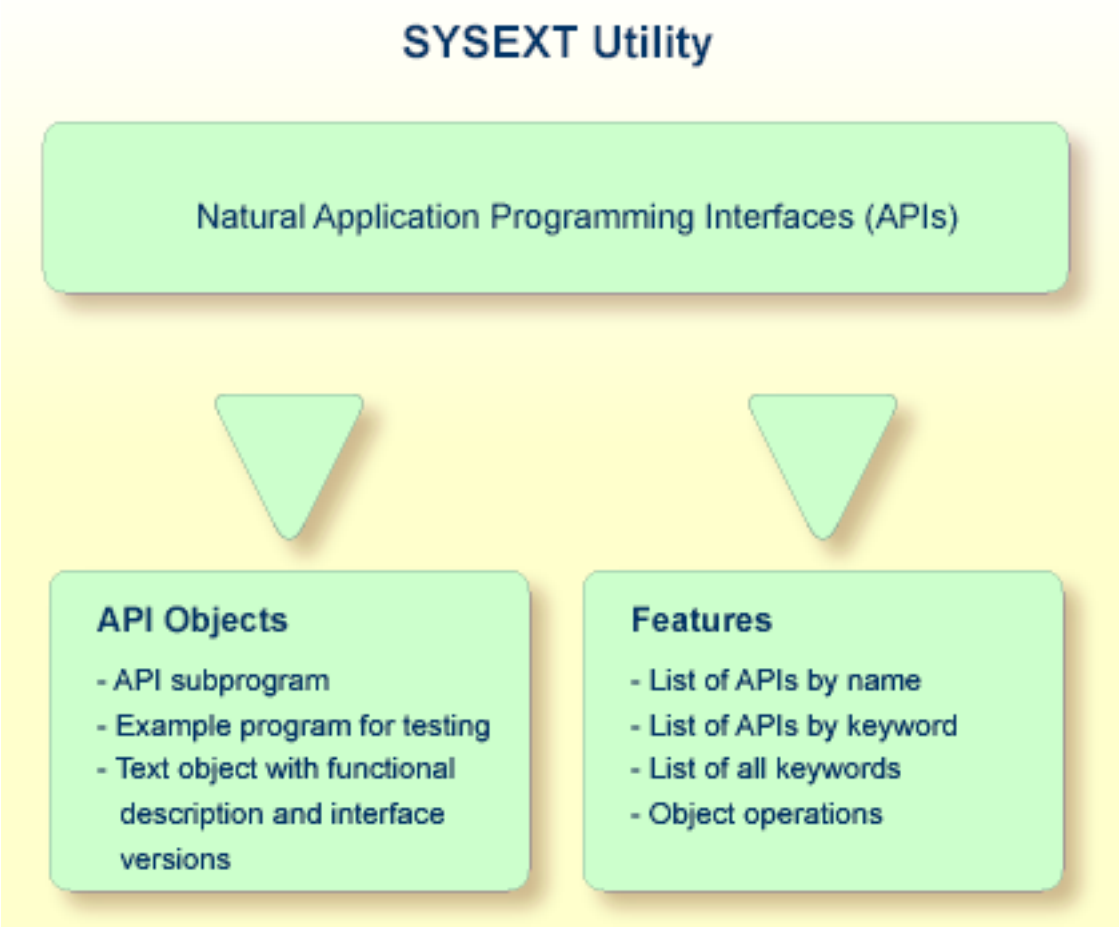
Prerequisite

- 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*nnnn*N	The API subprogram (cataloged object) that performs the designated function.
USR*nnnn*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*nnnn*N.
USR*nnnn*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**.

For some APIs, copycodes are available which provide functions related to the API. The copycodes are named `USRnnnnX`, where `X` is an identification character (such as "Z", "Y" etc.).

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

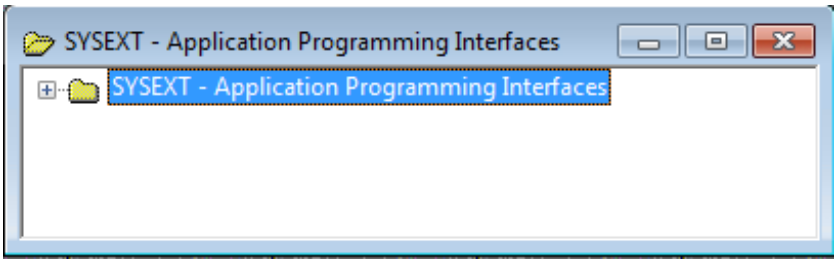
- Enter the following system command:

  ```
  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.

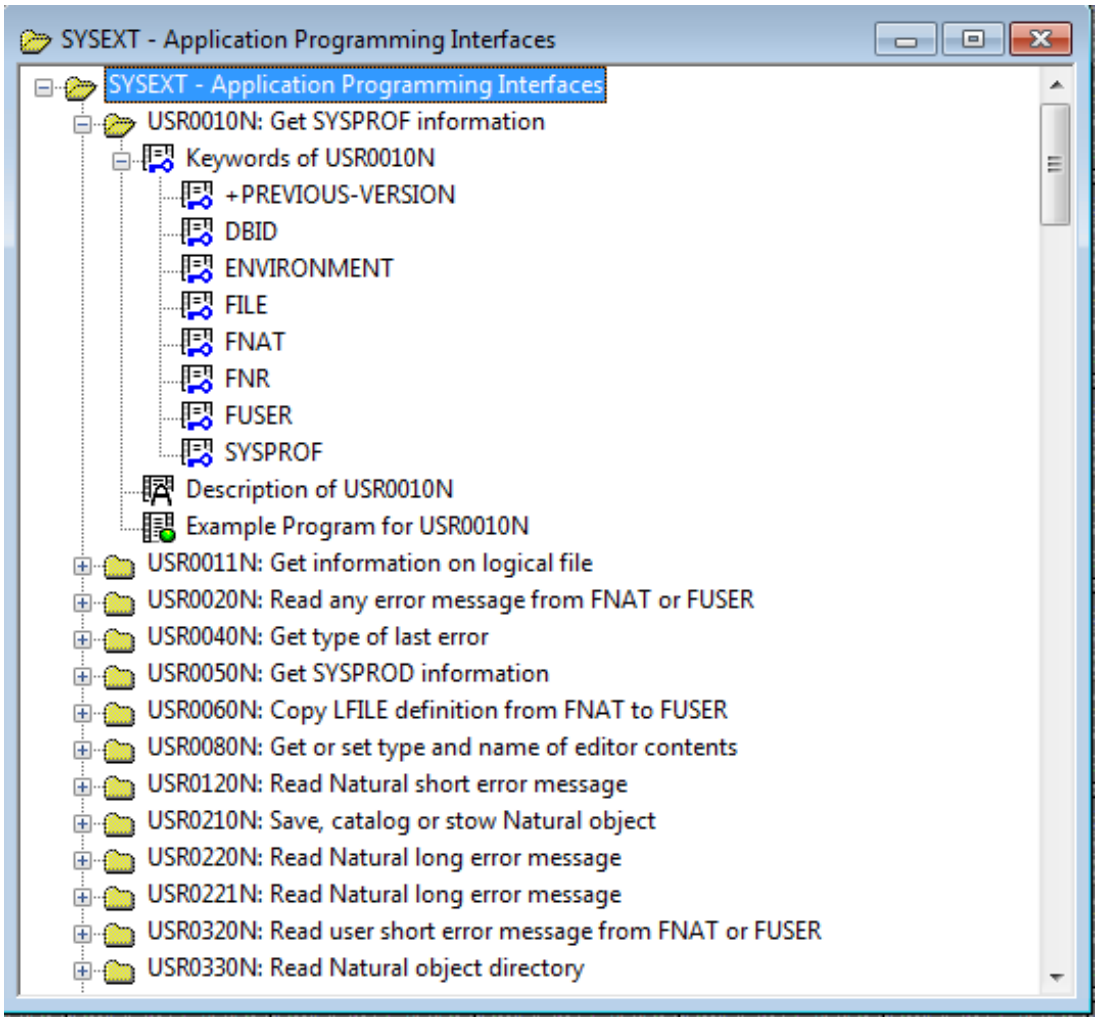
To terminate SYSEXT

- 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 *Performing SYSEXT Utility Functions*.
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

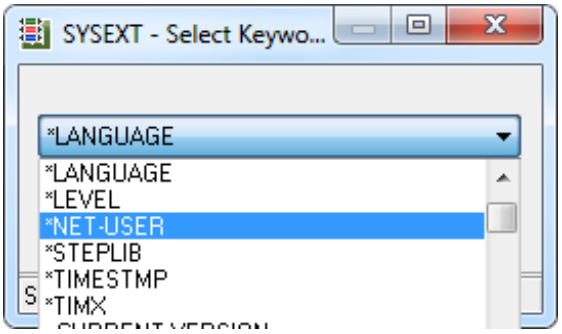
This function is used to list APIs by keyword.

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:

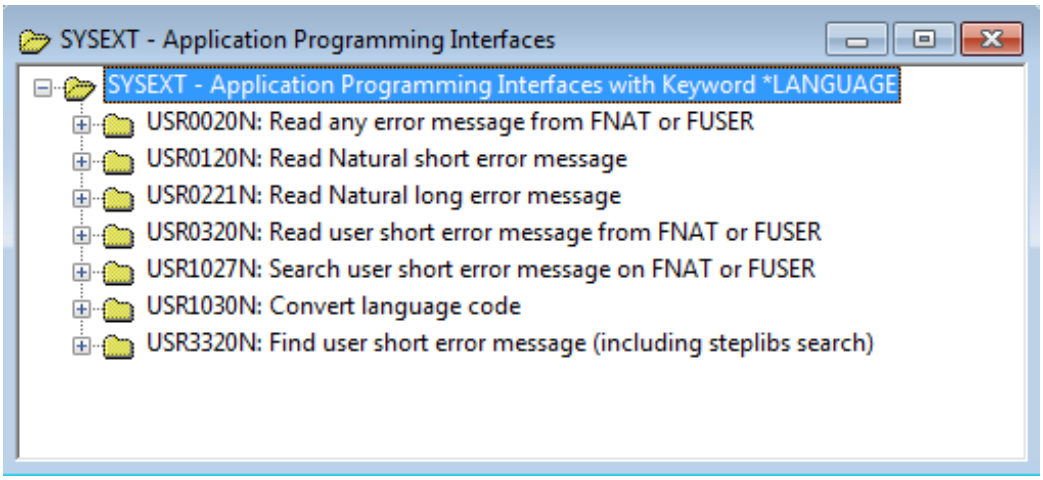


3 Choose the **OK** button.

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.

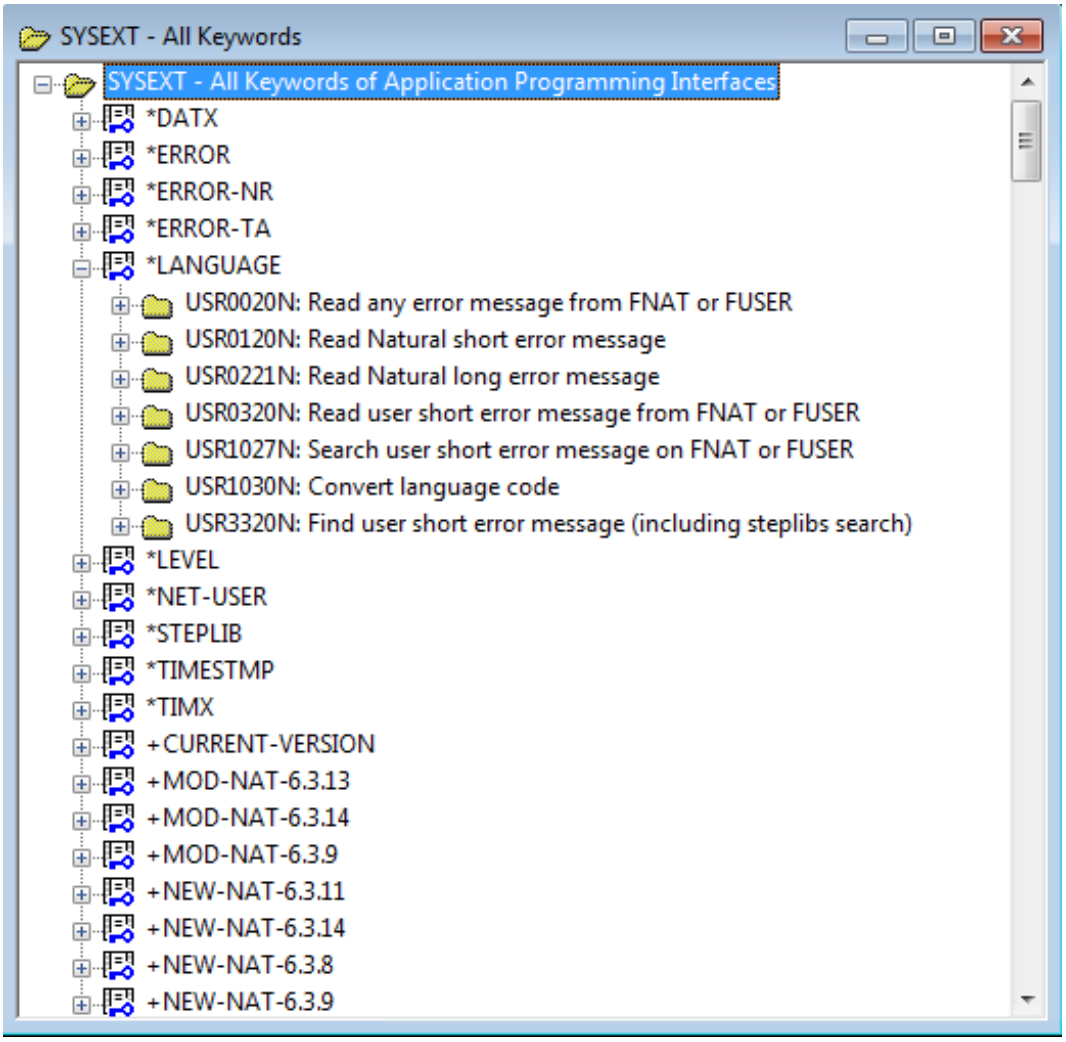
To list all API keywords

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
+CURRENT-VERSION	The current version of an API with interface versions (see *Interface Versions*).
+PREVIOUS-VERSION	A previous version of an API with interface versions (see *Interface Versions*).
+NEW-*PROD-version*	An API that has been added to a specific product in a specific version. For example, `+NEW-NAT-6.3.11` refers to an API that has been added to the product Natural in version `6.3.11`.
+MOD-*PROD-version*	An API that belongs to a specific product and has been modified in a specific version. For example, `+MOD-NAT-6.3.12` 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 steplibs) 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 `USR`*nnnn*`T` of that API. In the example program `USR`*nnnn*`P`, 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`.

If you want to use a Natural copycode contained in the system library SYSEXT, perform the following step:

- Copy the required copycode to the user library in the system file FUSER which contains the Natural objects that use this copycode.

To make use of a copycode

1 In the calling program, use the `INCLUDE` statement to specify the parameters listed in the text object `USRnnnnT` of that API or in the copycode itself. 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 Some copycodes require additional data definitions. These are described in the text object `USRnnnnT` of the API and in the copycode itself.

3 Enter the following statement:

```
INCLUDE USRnnnnX 'parameter'...
```

For further information, see the *INCLUDE* statement in the *Statements* documentation.

Note: Non-standard usage is always documented in the respective text object `USRnnnnT`.

XI SYSEXV Utility

57 SYSEXV Utility

The utility SYSEXV utility provides example programs that demonstrate the use of Natural features introduced in the current or a previous version of Natural.

All example programs are available as source objects. You obtain detailed functionality descriptions when you execute the programs.

Executing Example Programs

To execute an example program

1 Enter the following system command:

```
SYSEXV
```

An **Example Library SYSEXV** dialog opens which lists current Natural version(s) and categories.

2 Click on **Version** in the menu bar.

Or:

Click on **Categories** in the menu bar and select a category.

A commented list of Natural features and corresponding programs is displayed.

3 Choose the required feature/program.

The program is executed.

PF Keys

You can use the following PF keys:

PF Key	Name	Function
PF3	Exit	Close the current window.
PF12	Canc	Exit the SYSEXV utility.

Terminating an Example Program or the SYSEXV Utility

To terminate an example program

- Press PF3 (Exit).

To terminate the SYSEXV utility

- In the **Example Library SYSEXV** dialag, choose **Exit**.

  Or:

  In an example program, press PF12 (Canc).

XII SYSLVERS Utility

58 SYSLVERS Utility

The SYSLVERS utility is used to list objects which have been cataloged within a selected Natural version range.

The following topics are covered:

SYSLVERS in Interactive Mode

To invoke SYSLVERS

- To invoke SYSLVERS, issue the following system command or select it in the NaturalONE Tools and Utilities:

```
SYSLVERS
```

The SYSLVERS menu appears. In this page you can specify a version range, a library range and a Natural system file. All objects in the given library range on the selected Natural system file are processed. The objects which have been cataloged within the selected version range are listed in the Object List page. The Statistics page shows how many libraries and objects have been scanned and how many objects have been found within the selected version range. If desired, you can write the resulting data to a work file in text format or as CSV.

You can make the following specifications in the SYSLVERS menu:

Property	Value	Description
Library from		The library in which the processing starts. If the field is empty, the processing starts from the very first library. If Library to is empty and the Library from value is followed by an asterisk (*), all libraries beginning with this value are processed (*wildcard notation*). Enter a single asterisk (*) if you want to process all libraries of the selected system file. The Library from value is initialized with the name of the current library.
Library to		The library in which the processing ends. If the field is empty, the processing ends at the very last library unless the wildcard notation is used for Library from. In this case the processing ends at the value specified with the wildcard. If an equal sign (=) is specified in this field, Library to uses the same value as Library from. The Library to value is initialized with the name of the current library.
System file		The Natural system file to be processed.

Property	Value	Description
		Default: Blank
	Blank	The Natural system file to be used depends on the value specified in the `Library from` property. If the value starts with `SYS`, but is not equal to `SYSTEM`, the `FNAT` system file is used. Otherwise the `FUSER` system file applies.
	U	The `FUSER` system file is used.
	N	The `FNAT` system file is used.
`Version from`	*vv.rr*	The (inclusive) beginning of the Natural version range, where ■ *vv* is the major version, and ■ *rr* is the minor version. Default: If the field is empty, the first version is used. If the minor version is not specified, the first minor version is taken.
`Version to`	*vv.rr*	The (inclusive) end of the Natural version range, where ■ *vv* is the major version, and ■ *rr* is the minor version. Default: If the field is empty, the last version is used. If the minor version is not specified, the last minor version is taken.
`Export`		Specifies whether the resulting data is written to work file 7 and in which format. Default: N
	N	No data is written to work file 7.
	T	Writes the object list and statistics in free text format.
	C	Writes the object list in CSV format with a comma (,) separator.
	S	Writes the object list in CSV format with a semicolon (;) separator.

If you press ENTER, the selected libraries are scanned and the found objects are displayed in the **Object List** page.

If `Export` is set to T, C or S, the resulting data is written to work file 7.

To leave the `SYSLVERS` menu, press PF3 or enter `END` in the Command field.

PF Keys and Commands

The following PF keys and commands are available in interactive mode:

PF Key	Name	Command	Page	Function
PF1	Help	?	General	Displays help information.
PF3	Exit	END .	General	Exits the current menu.
PF5	Stats	STATISTICS	Object List	Displays the Statistics page.
	Obj	OBJECT	Statistics	Displays the Object List page.
PF6	-	TOP	Object List	Scrolls to the beginning of the list.
PF7	--	PREV	Object List	Scrolls one page up.
PF8 ENTR	+	NEXT	Object List	Scrolls one page down.
PF9	++	BOTTOM	Object List	Scrolls to the end of the list.
PF12	Canc	CANCEL	General	Exits the SYSLVERS command.
		TRACE *value*	General	Start or stop the internal trace. See SYSLVERS **in Batch Mode** for the allowed values. If you enter the TRACE command without a value, the current state of the trace is displayed.

The name of the PF key can also be used as a command. For example, enter the command "++" to scroll to the end of the list.

SYSLVERS in Batch Mode

To invoke SYSLVERS

- To invoke SYSLVERS, enter the following system command into the primary command input data set CMSYNIN:

```
SYSLVERS
```

SYSLVERS in batch mode reads keywords from the batch input file *CMSYNIN* or *CMOBJIN* until it reaches the END keyword or a dot (.).

SYSLVERS uses the following syntax format for the input lines:

```
Keyword[=value]
```

The following rules apply:

- Lines starting with an asterisk (*) are ignored.
- A keyword may be specified without the equal sign and value. The value is set to blank in this case.

- Blanks can be added before or after the keyword or value.
- Keywords and values can be specified in upper or lower case.
- The maximum input line length is 78 characters.

The following keywords are available:

Keyword	Value	Description
LIBRARY-FROM		The library in which the processing starts. If the value is empty, the processing starts from the very first library. If LIBRARY-TO is empty and the LIBRARY-FROM value is followed by an asterisk (*), all libraries beginning with this value are processed (*wildcard notation*). Enter a single asterisk (*) if you want to process all libraries of the selected system file. Default: Asterisk (*)
LIBRARY-TO		The library in which the processing ends. If the value is empty, the processing ends at the very last library unless the wildcard notation is used for LIBRARY-FROM. In this case the processing ends at the value specified with the wildcard. If an equal sign (=) is specified in this field, LIBRARY-TO uses the same value as LIBRARY-FROM. Default: Blank
SYSTEM-FILE		The Natural system file to be processed. Default: Blank
	Blank	The Natural system file to be used depends on the value specified in the LIBRARY-FROM keyword. If the value starts with SYS, but is not equal to SYSTEM, the FNAT system file is used. Otherwise the FUSER system file applies.
	FUSER	The FUSER system file is used.
	FNAT	The FNAT system file is used.
VERSION-FROM	*vv.rr*	The (inclusive) beginning of the Natural version range, where ■ *vv* is the major version, and ■ *rr* is the minor version. If the field is empty, the first version is used. If the minor version is not specified, the first minor version is used. Default: Blank
VERSION-TO	*vv.rr*	The (inclusive) end of the Natural version range, where

Keyword	Value	Description
		■ *vv* is the major version, and ■ *rr* is the minor version. If the field is empty, the last version is used. If the minor version is not specified, the last minor version is taken. Default: Blank
EXPORT		Specifies whether the resulting data is written to work file 7 and in which format. Default: NO
	NO	No data is written to work file 7.
	TEXT	Writes the object list and statistics in free text format.
	COMMA	Writes the object list in CSV format with a comma (,) separator.
	SEMICOLON	Writes the object list in CSV format with a semicolon (;) separator.
TRACE		Set the level of internal trace of the SYSLVERS command. The internal trace is written to the standard output (CMPRINT data set). In general, a higher trace level also contains the information of the lower trace levels. Default: OFF
	0-99	Trace level.
	OFF	No trace is written. Corresponds to trace level 0.
	ON	Activates the internal trace. Corresponds to trace level 99.
	ERROR	If an error occurs at the reading of a library or an object, a message is printed. Corresponds to trace level 1.
	WARNING	If a warning occurs at the reading of a library or object, a message is printed. Corresponds to trace level 2.
	TIME	Activates the performance measurement. The result is provided in the Statistics page. Available in batch mode only. Corresponds to trace level 3.
	DETAIL	Returns a more detailed error message. Corresponds to trace level 8.
	REJECT	Lists rejected objects (those outside the version range) in the Object List and writes them to the standard output (CMPRINT data set). Available in batch mode only. Corresponds to trace level 80.
END .		The keyword END or a period (.) indicates the end of the SYSLVERS input.

Example

The following SYSLVERS execution in batch lists all objects on the *FUSER* system file which have been cataloged with a Natural version prior to version 5. The command writes the result to the Batch Output File *CMPRINT* and additionally to work file 7 in CSV format with a semicolon (;) separator.

```
SYSLVERS
*
* List all objects on FUSER with catalog version < 5
*
LIBRARY-FROM = *
LIBRARY-TO   =
SYSTEM       = FUSER
VERSION-FROM =
VERSION-TO   = 4
EXPORT       = SEMICOLON
END
```

Object List

For the specified library range the Object List page displays all objects which have been cataloged within the selected version range. The following information is listed:

- Count – Counter of the objects found
- Library – Object library
- Object – Object name
- Type – Type of the object like "Program"
- User ID – ID of the user who has cataloged the object
- Catalog Date – Date when the object was cataloged
- Version – Natural version used to catalog the object

The following information is additionally available if the Object List is written to a work file:

- Size – Size of the cataloged object
- Mode - Programming mode of the object like "Structured"

If you press PF5 or enter `STATISTICS` in the Command field in the interactive mode, the **Statistics** page is displayed.

To leave the Object List page, press PF3 or enter `END` in the Command field.

Note: If you are using Natural Security (NSC), only those libraries are scanned which you can access. If you have only restricted access to a library, the library is skipped. In this case you may activate the `SYSLVERS` trace to retrieve further information.

Statistics

The Statistics page displays the selected input values and the following statistical information:

- Libraries scanned – Number of libraries processed
- Objects scanned – Number of objects processed
- Objects found – Number of objects which satisfied the selected criteria

In batch mode, if trace level 3 or higher is selected, the following information is displayed additionally:

- Date – The date when the processing started
- Start time – The time when the processing started
- End time – The time when the processing ended
- Elapsed time – The elapsed time needed to process the `SYSLVERS` command

If you press PF5 or enter `OBJECT` in the Command field in the interactive mode, the **Object List** page is displayed.

To leave the Statistics page, press PF3 or enter `END` in the Command field.

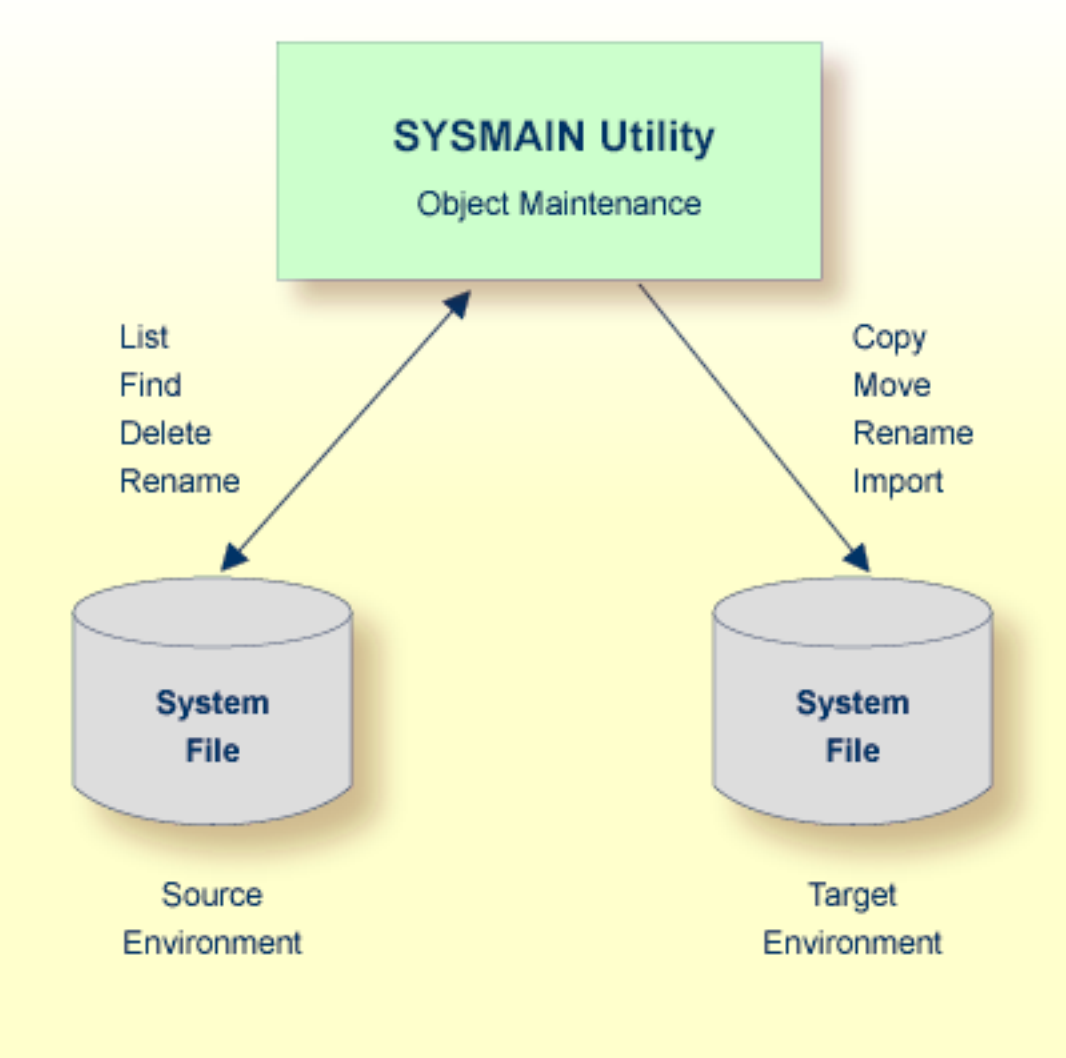
XIII SYSMAIN Utility - Object Maintenance

The SYSMAIN utility is used to perform object maintenance functions such as copy, move, replace, delete and import.

59 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.

60 Invoking and Terminating SYSMAIN

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*.

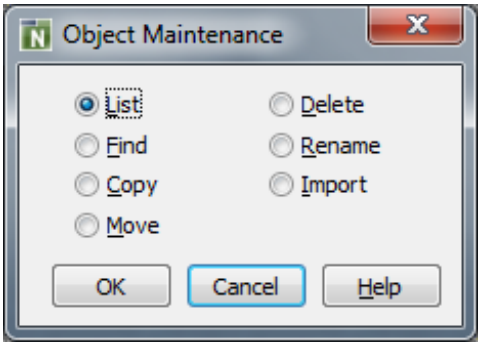
Invoking SYSMAIN

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:

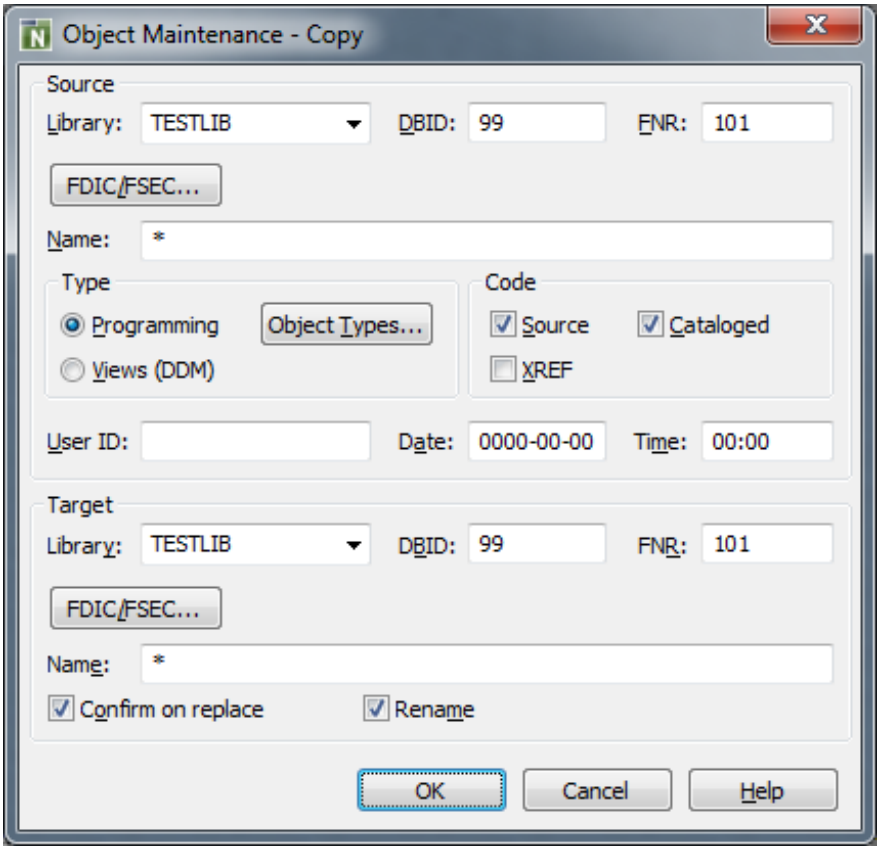
SHIFT+L for **List**
SHIFT+F for **Find**
SHIFT+C for **Copy**
SHIFT+M for **Move**
SHIFT+D for **Delete**

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**:



Terminating SYSMAIN

To terminate the SYSMAIN utility

- Choose **Cancel**.

  Or:

  Choose the standard Windows close button.

61 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).

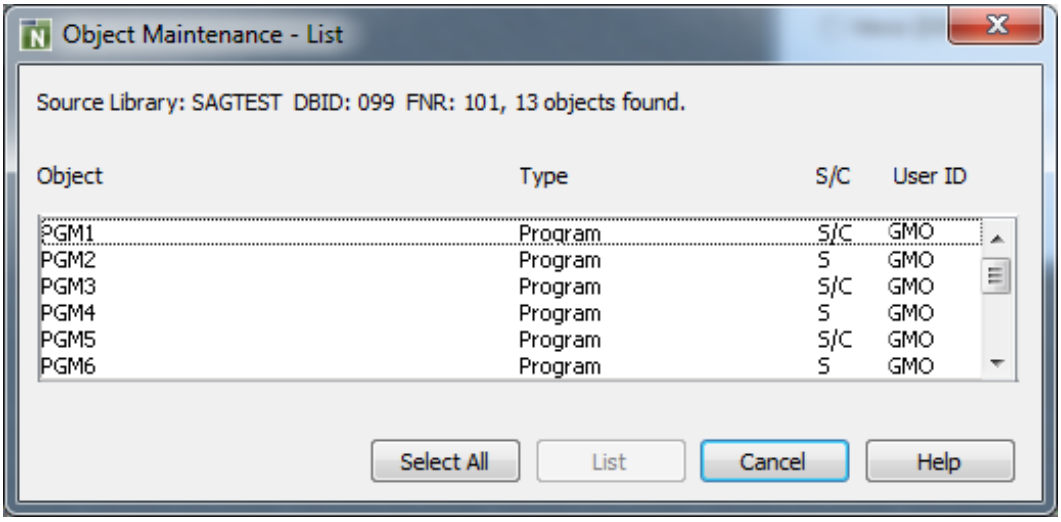
If you have specified a date, you can enter a start time in the **Time** box to list only objects that were saved or cataloged at or after this time. The default time is `00:00` (no time). The time is given in the format *HH*:*II* (*HH* = hours, *II* = minutes).

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:



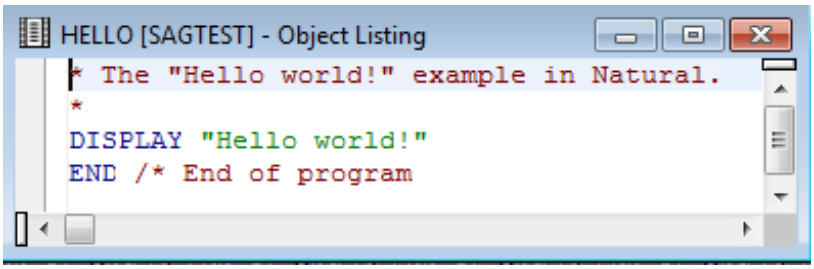
11 Select or deselect single or multiple objects by proceeding as described in *To select/deselect list items*.

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 SYSMAIN and activate an open **Object Listing** window.

62 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).

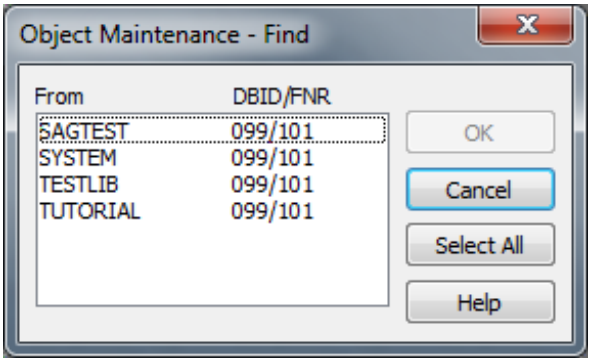
If you have specified a date, you can enter a start time in the **Time** box to find only objects that were saved or cataloged at or after this time. The default time is `00:00` (no time). The time is given in the format *HH*:*II* (*HH* = hours, *II* = minutes).

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:

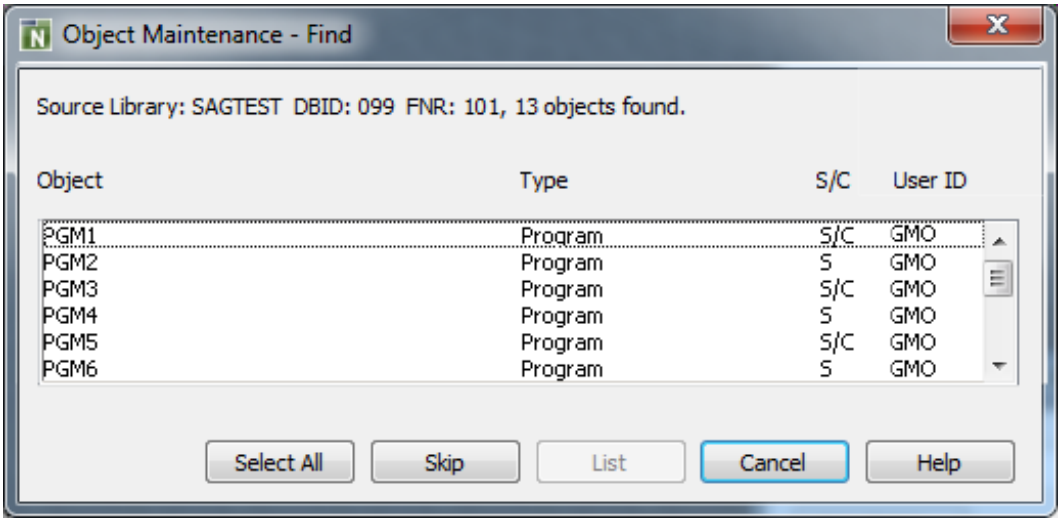


11 Select or deselect single or multiple libraries by proceeding as described in *To select/deselect list items*.

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.

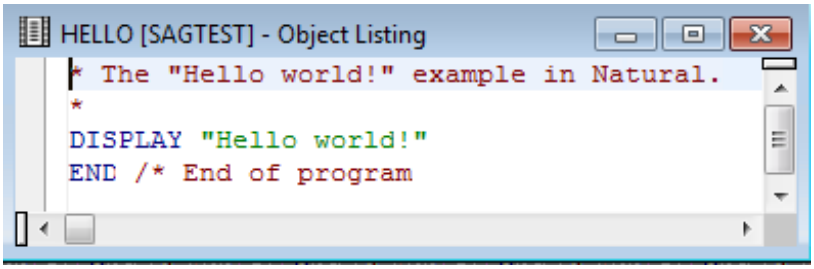
14 Select or deselect single or multiple objects by proceeding as described in *To select/deselect list items*.

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.

63 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).

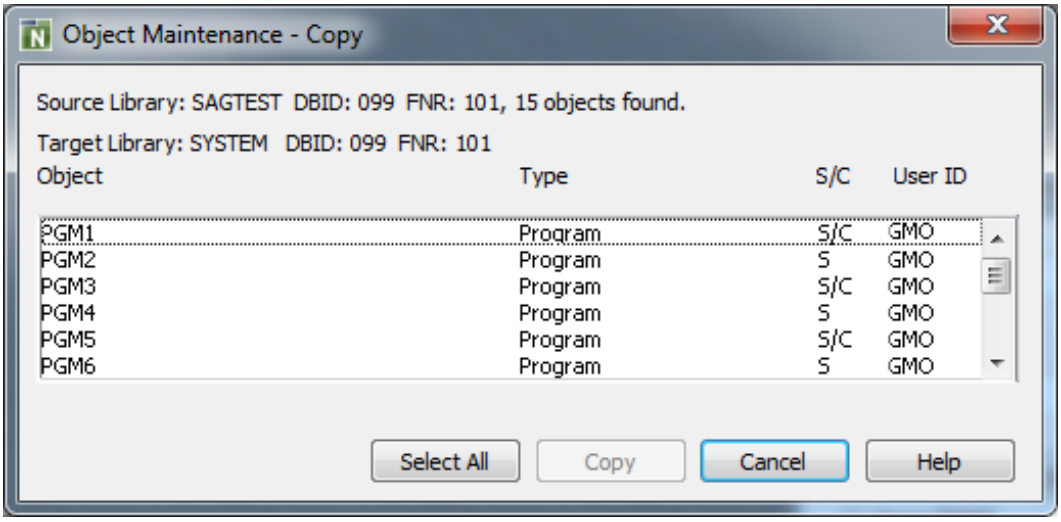
2 In the **Target** group box, specify the target environment for the object(s) selected for processing:

- 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:



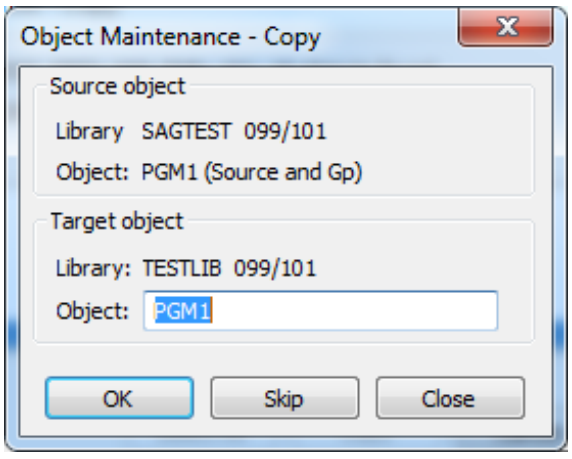
The dialog box shows the library locations, the number of objects found and the object names and types. It also indicates whether a source (`S`) and/or a cataloged (`C`) object exists and the ID of the user who saved and/or cataloged an object.

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.

64 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:

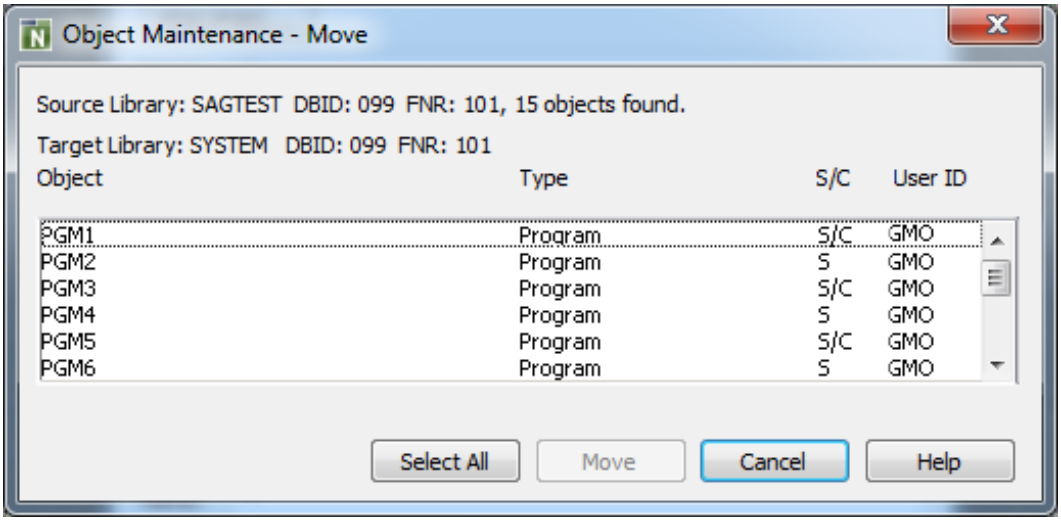
- In the **Library** list box, enter the name of the target library where you want to place the object(s) or select a library from the drop-down list. The default is the current library. If you want to create a library, enter the name of a new library.

  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:



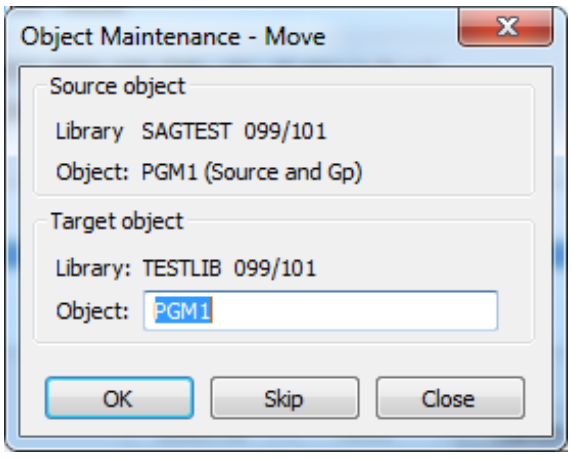
The dialog box shows the library locations, the number of objects found and the object names and types. It also indicates whether a source (`S`) and/or a cataloged (`C`) object exists and the ID of the user who saved and/or cataloged an object.

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.

65 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.

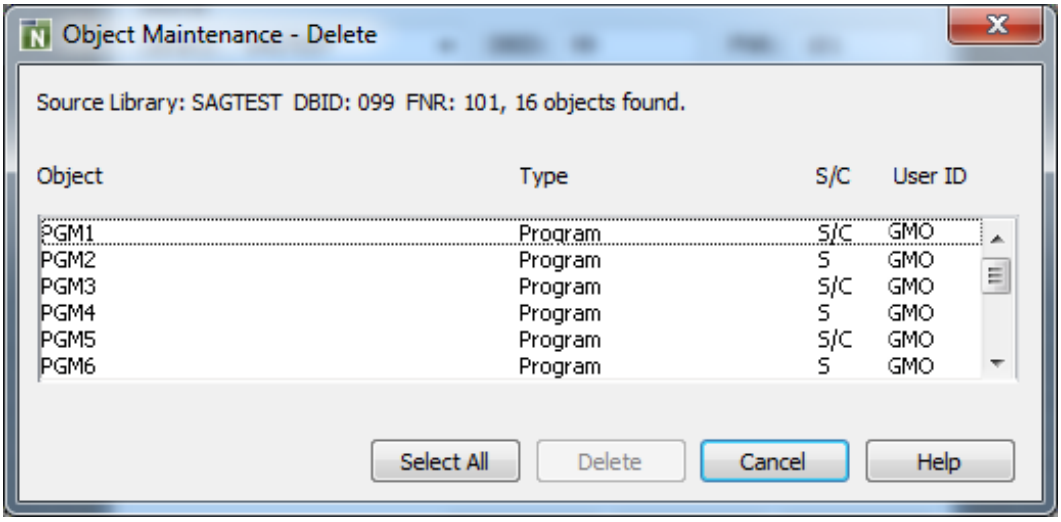
8 In the **Date** box, enter a start date to delete only objects that were saved or cataloged on or after this date. A date must be entered in the format associated with the preset value in the **Date** box. The default date is `0000.00.00` (no date). The date is given in the format *YYYY.MM.DD* (*YYYY* = 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.

66 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).

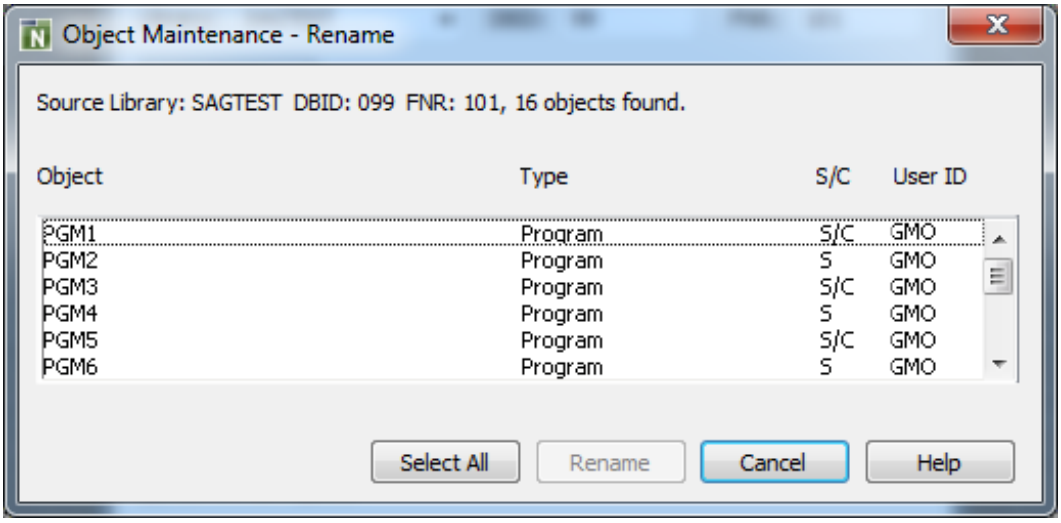
2 In the **Target** group box, you can specify the following:

- 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:



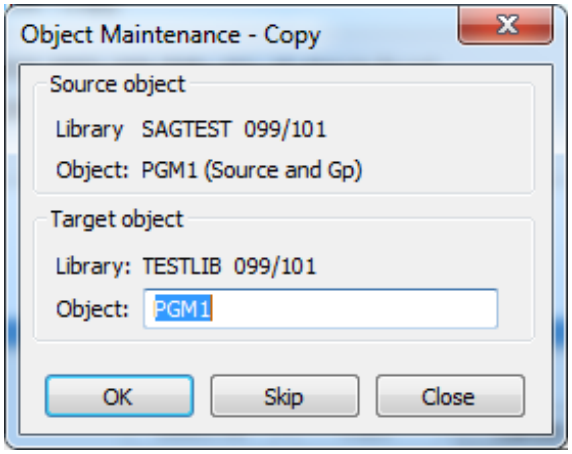
The dialog box shows the library locations, the number of objects found and the object names and types. It also indicates whether a source (`S`) and/or a cataloged (`C`) object exists and the ID of the user who saved and/or cataloged an object.

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.

67 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.

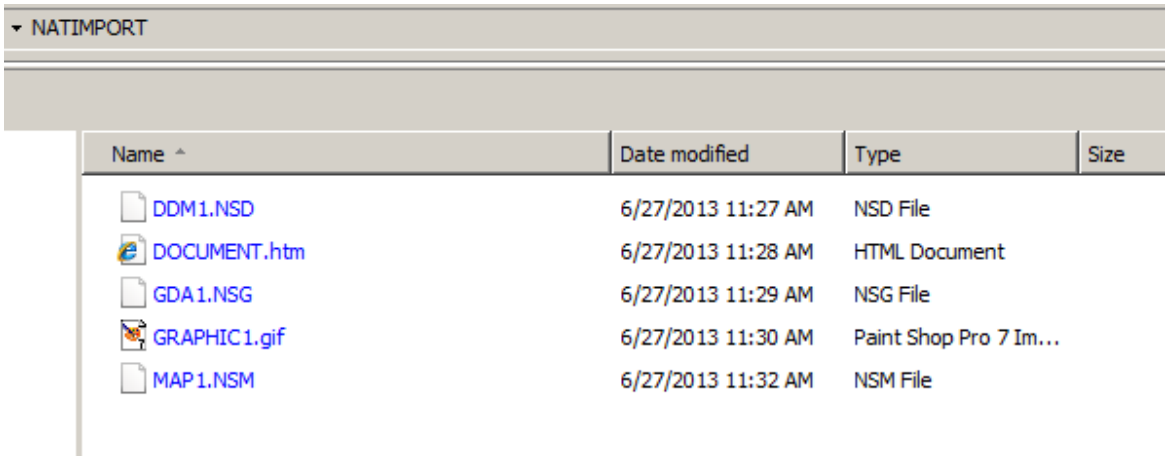
To import an object

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.

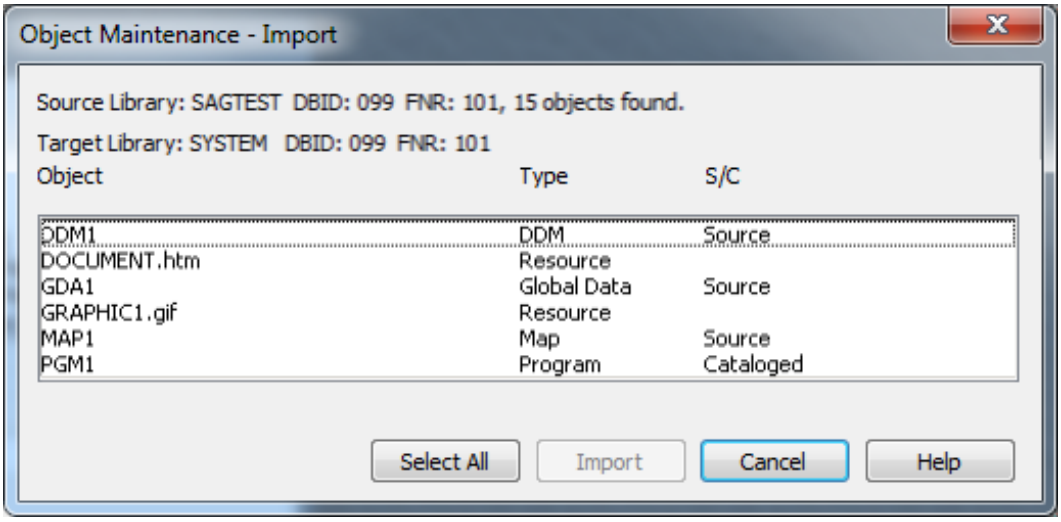
2 In the **Target** group box, specify the target environment for the objects to be imported:

- 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**.

68 Using SYSMAIN with Subprogram

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
command	A250	The command string to be executed by SYSMAIN: see ***Using Commands***.
error	N4	The return code issued by SYSMAIN at the end of processing to indicate a normal end of processing or an error.
message	A72	The message corresponding to the error given online.
library	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

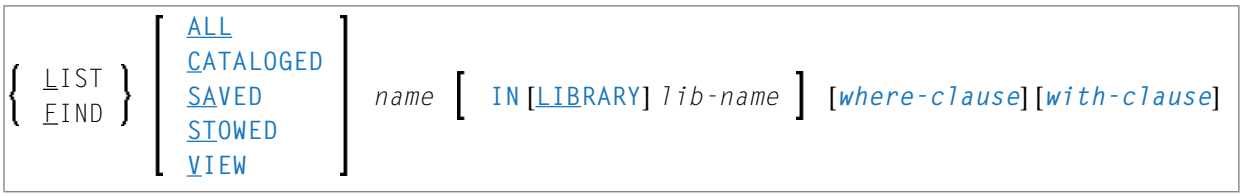
correspond to the syntax symbols used for system commands. These symbols are explained in *System Command Syntax* in the *System Commands* documentation.

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:

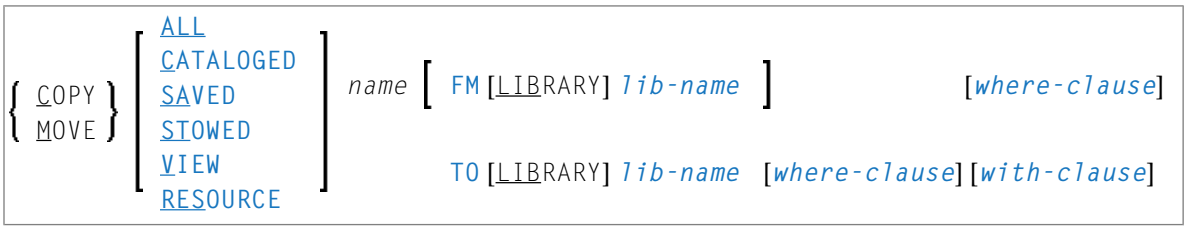


Examples of LIST and FIND

```
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:

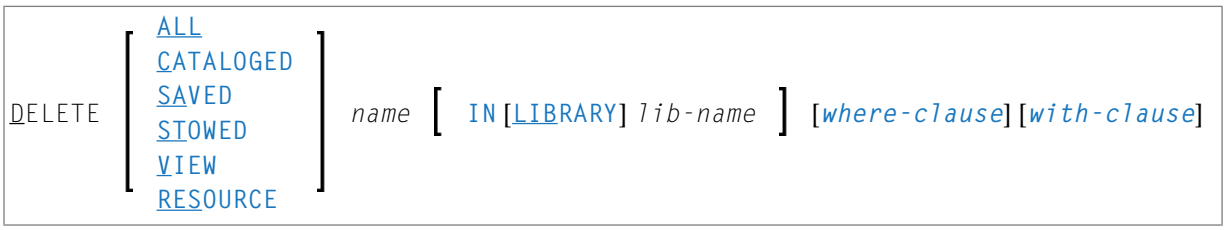


Examples of COPY and MOVE

```
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:

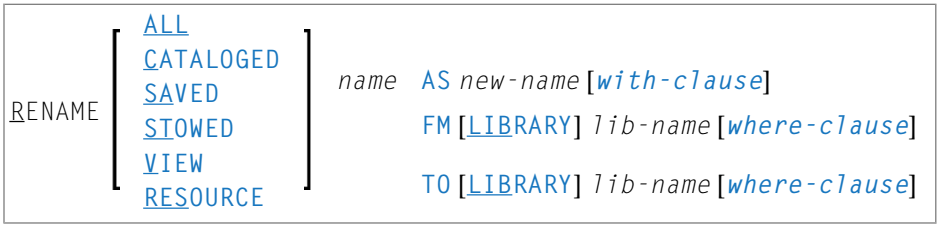


Examples of DELETE

```
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:

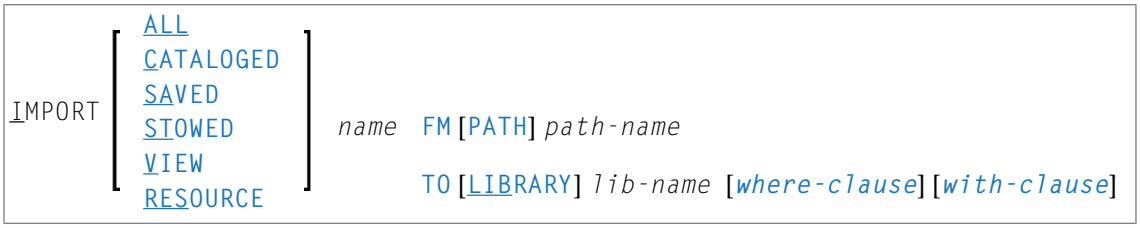


Examples of RENAME

```
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:



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 }   ]
                            [        { N }   ]
        [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	*name*	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	*name*	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	*name*	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	*name*	A9	Only applies to programming objects. The name of an object (or a range of names) for which the saved (source) *and* 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 *and* 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	*name*	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	*name*	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	*lib-name* or *path-name*	A8 or A253	Specifies a source library or a source path. The source library or path contains the object to be processed.
TO	*lib-name*	A8	Specifies a target library.
AS	*new-name*	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	*lib-name*	A8	An optional keyword that indicates the name (*lib-name*) of a source or a target library. If you omit the keyword and

Keyword	Value	Natural Data Format/Length	Explanation
			respective value, the library where you logged on before you invoked SYSMAIN is used for processing. 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. *lib-name* 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 *lib-name*.
PATH	*path-name*	A253	Only applies to the IMPORT command. An optional keyword that indicates the name (*path-name*) of a source path. *path-name* 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 *path-name*.
WHERE	*where-clause*	-	An optional keyword that indicates the start of a *where-clause*. The *where-clause* must always follow the FROM/FM/IN or TO keyword and the library name (*lib-name*) or path name (*path-name*) if relevant; the sequence of the keywords and values within the clause can be specified in any order.
DBID	*dbid*	N5	The database ID (DBID) of a source or a target system file. 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. Valid DBIDs are 1 to 65535. 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. See also the section ***File Security in Remote Environments***.
FNR	*fnr*	N5	The file number (FNR) of a source or a target system file. 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. Valid FNRs are 1 to 65535.

Keyword	Value	Natural Data Format/Length	Explanation
			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. See also the section ***File Security in Remote Environments***.
DIC	*dbid* *fnr* *password* *cipher*	A80	Specifies the environment of the FDIC source and/or target system file: database ID (*dbid*), file number (*fnr*), Adabas password (*password*) and Adabas cipher code (*cipher*). See also the section ***File Security in Remote Environments***.
SEC	*dbid* *fnr* *password* *cipher*	A80	Specifies the environment of the FSEC source and/or target system file: database ID (*dbid*), file number (*fnr*), Adabas password (*password*) and Adabas cipher code (*cipher*). See also the section ***File Security in Remote Environments***.
WITH	*with-clause*	-	An optional keyword that indicates the start of a *with-clause*. The keywords and values of the *with-clause* can be specified in any order, and the *with-clause* can be placed in any location within the command string, except in the first three positions.
TYPE	*type*	A20	The type(s) of object to be processed as listed in ***TYPE Specification*** below.
FMDATE	*date*	A10	The start date of a time period: All objects which were saved or cataloged on or after the specified date are processed. A date must be specified in a valid Natural date format. The default format is the international format *YYYY-MM-DD* (*YYYY* = year, *MM* = month, *DD* = day), for example, 2006-05-20.
FMTIME	*time*	A5	Only applies if FMDATE is specified. Specifies a start time: All objects which were saved or cataloged at or after the specified time (and date) are processed. A time must be specified in the format *HH*:*II* (*HH* = hours, *II* = minutes), for example, 11:33.
USER	*user-id*	A8	A user ID: All objects that were saved or cataloged by the specified user are processed.

Keyword	Value	Natural Data Format/Length	Explanation
XREF	N or Y	A1	Only applies to programming objects and if Predict is installed. Indicates whether XRef data stored in Predict system files is to be processed. You can specify one of the following values: 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. Y All XRef data is processed. See also the section ***XRef Considerations***.
REPLACE	-	-	Activates the replace option used in a *with-clause*. An object with the same name in the target environment is replaced by the object to be processed. **Note:** If an object is replaced it is also deleted from the Natural buffer pool; any existing cross-reference records are also deleted.
RCOP	-	-	Specifies that a copy of the object being renamed is to be made.
NOPROMPT	-	-	Not applicable in batch mode. Disables (NOPROMPT) the SYSMAIN prompts. With NOPROMPT, no confirmation screen is displayed. For example, before any deletion, SYSMAIN prompts you for confirmation.
HELP	-	-	This keyword is provided for compatibility reasons only.
STRUCT or SM			Only applies to the IMPORT command. Indicates structured mode described in *Natural Programming Modes* in the *Programming Guide*.
REPORT			Only applies to the IMPORT command. Indicates reporting mode described in *Natural Programming Modes* in the *Programming Guide*.
.	-	-	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.
*value**	All objects or libraries with names that start with *value*. Example: `AB*` Selected: AB, AB1, ABC, ABEZ Not selected: AA1, ACB

Input	Objects or Libraries Selected
*value***value**	All objects or libraries that match *value* combined with one or two asterisks (*) in any order. Example: A*C* Selected: ABCZ, AXXCBBBZ, ANCZ Not selected: ABDEZ, ACBBBZA

69 XRef Considerations

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.

70 Security Considerations for Administrators

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**.

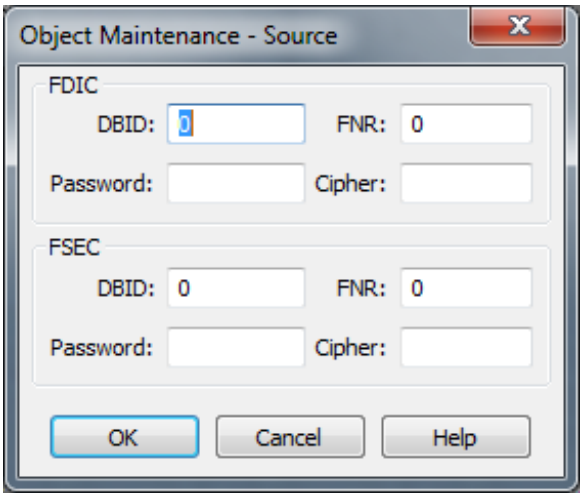
2 Enter the appropriate security information:

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

71 SYSNCP Utility

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.

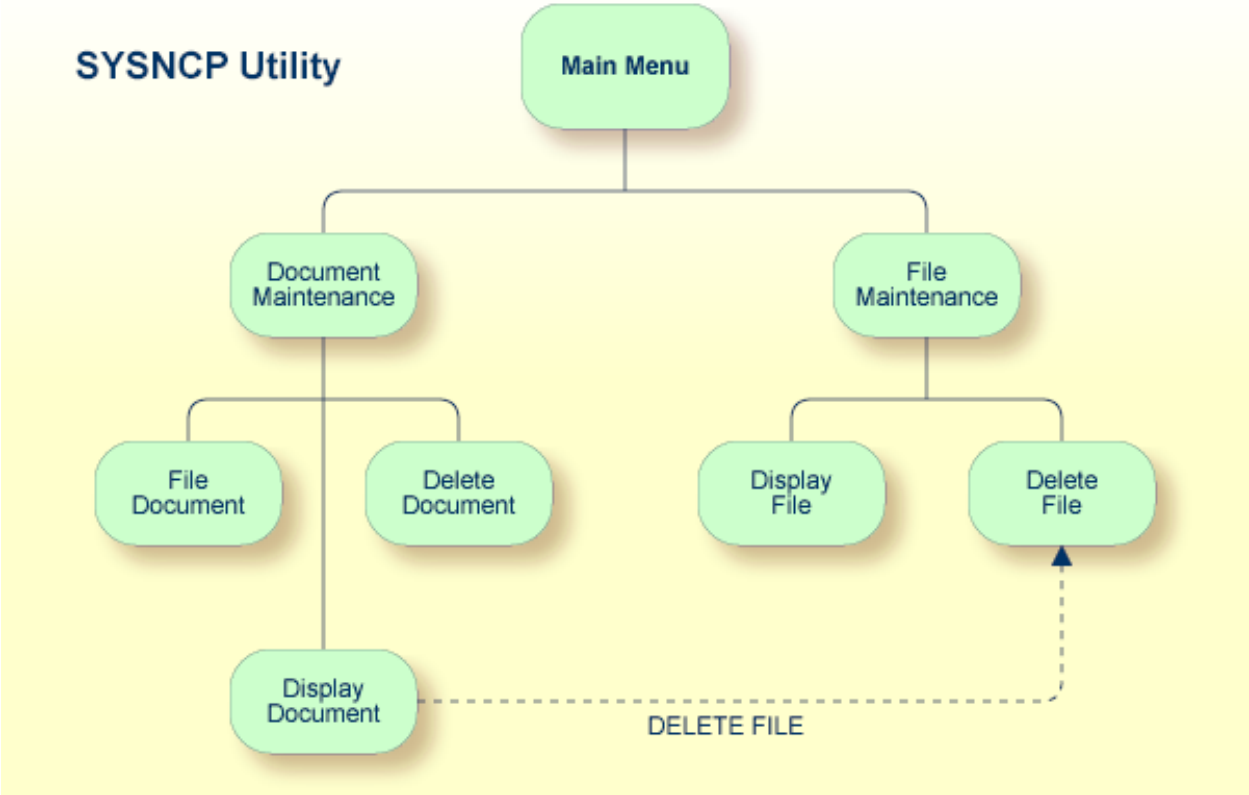
Prerequisites for Windows

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*.

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 `0` in the **Code** field 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

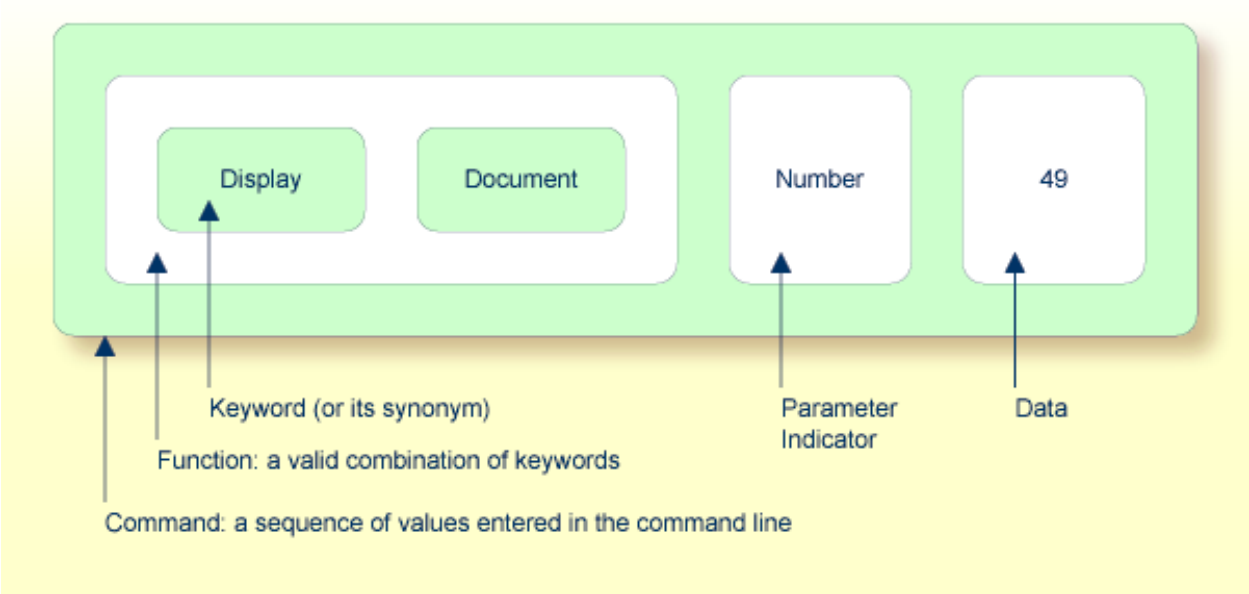
The Natural command processor consists of two parts: a development part and a runtime part:

- 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.

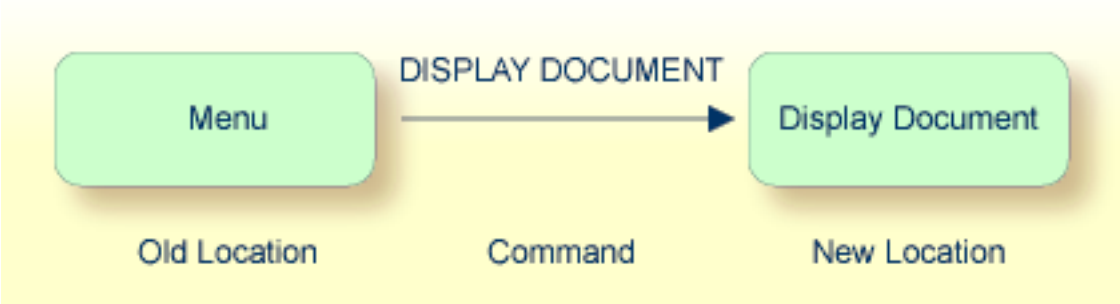
What is a Command?

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:

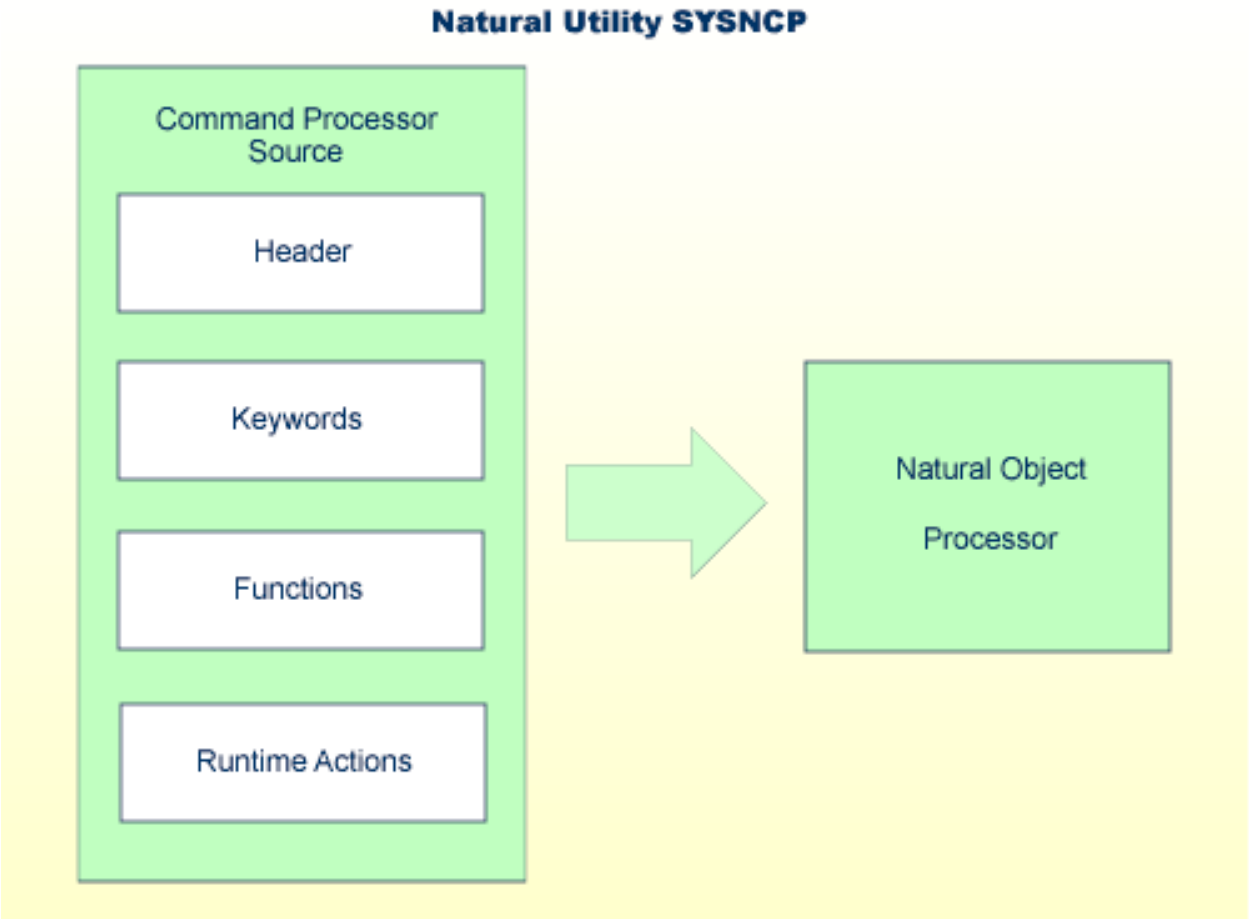


The location from which a certain command can be issued can be restricted on a system-wide or user-specific basis. On a system-wide basis, for example, the functions specified within commands can be local or global. A global function can be issued from *any* location while a local function can only be 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 in the **Code** field.

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 *Header Records*.
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 opens 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 following special characters:

- / $ &#+_

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) in the **Code** field,
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) in the **Code** field,
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: `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	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. 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.

Field	Explanation
	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 *Keyword Maintenance*.
Second Entry used as	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. 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. 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 *Keyword Maintenance*.
Third Entry used as	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. 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 *Keyword Maintenance*.
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	The following values are permitted: `+` 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. `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 *Keyword Maintenance*.
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 **E field** described in the section *Keyword Maintenance*.
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: `L` Command is to be validated as a local command. `G` Command is to be validated as a global command. 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.

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 Header 2 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 *Statements* documentation.
Prefix Character 1	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. 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.
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	This option specifies whether keywords are to be translated to upper case in the Keyword Editor and the application: `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. `N` Keywords entered in the Keyword Editor are not converted to upper case. In the application, end-users must enter the keywords *exactly* as they appear in the Keyword Editor.
Unique Keywords	This option specifies whether keywords within the processor must be unique: `Y` Each keyword defined must be unique within this processor, regardless of its type. `N` Each keyword defined for a given keyword type (1, 2, 3 or P) must be unique.

Miscellaneous Options - Header 3

Miscellaneous options can be entered on the **Processor Header Maintenance 3** screen:

Field	Explanation
Invoke Action Editor	This option specifies whether the Runtime Action Editor is to be activated from the Function Editor (see the sections *Runtime Action Editor* and *Define Functions*). Possible values are: `Y` The Runtime Action Editor is invoked whenever a valid keyword combination is defined in the Function Editor. `N` The Runtime Action Editor is suppressed in the Function Editor. **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.
Catalog User Texts	This option specifies whether user texts are to be cataloged with the command processor: `Y` Text portions of the edit line (Keyword Editor; see the section *Define Keywords*) 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. `N` Texts are not cataloged with the command processor and cannot be read at runtime.
Security Prefetch	This option specifies whether security checking is to be performed when the command processor is initially invoked during runtime or at each command evaluation. Possible values are: `Y` If Natural Security is installed, security checking is performed for all keywords when the processor is invoked. `N` If Natural Security is installed, security checking is performed with the evaluation of each keyword. 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.
Command Log Size	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: `0` No storage space is allocated to command logging. Command logging is inactive. `1` 1 KB of storage space is allocated to command logging. Command logging is active.

Field	Explanation
Implicit Keyword Entry	This option specifies whether a keyword of type 1 is to be retained as an implicit keyword for all subsequent commands. Possible values are: `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. `N` Option is disabled.
Command Delimiter	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. Example: `DISPLAY CUSTOMER; MODIFY CUSTOMER; PRINT`
PF-Key may be Command	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. Possible values are: `A` The identifier for this PF key (system variable `*PF-NAME`) is used as the command. `K` The content of the `*PF-KEY` system variable is used as the command. `Y` If `*PF-NAME` is empty, the content of the `*PF-KEY` system variable is used instead. `N` PF keys cannot be used as command, Natural error NAT6913 is issued with message `Command line not accepted`. For more information on the system variables `*PF-NAME` and `*PF-KEY` see the *System Variables* documentation.

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	Specifies the character to be used to precede data. Default data delimiter is the number sign (#). Example: `ADD CUSTOMER #123`
Data Allowed	Specifies if data input is allowed at runtime. Possible values are: `N` A runtime error occurs if data is found. `D` Data is dropped if present. `S` Data is placed at the top of the Natural stack. No verification is performed.

Field	Explanation
	Y Data is checked and keyword entries of type P (parameter indicator) are evaluated. Example of Y: `DISPLAY CUSTOMER NAME=SMITH`

More than one Item Allowed	Only applies if the option **Data Allowed** is set to Y. Specifies whether more than one data string is permitted. Possible values are:
	
	N A runtime error occurs if more than one data string is found.
	D All data after the first data string are dropped.
	Y More than one data string is permitted.
	
	Example:
	
	`ADD ARTICLE #111 #222`
	
	As long as uniqueness is guaranteed, the data delimiter can be omitted.
	
	Example:
	
	`ADD ARTICLE 123`

Maximum Length of one Item	Only applies if the option **Data Allowed** is set to Y. 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.
Item Must be Numeric	Only applies if the option **Data Allowed** is set to Y. Specifies whether each data value must be an integer value: Y Data input must be a positive integer value. If not, a runtime error occurs. N Data can be of any type.
Put to Top of Stack	Only applies if the option **Data Allowed** is set to Y. Specifies where data is to be placed: Y Data is placed at the top of the Natural stack. 1 - 9 Data is placed in the *n*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.
If Error, Drop all Data	Only applies if the option **Data Allowed** is set to Y or N. Specifies the reaction to a data evaluation error: Y If an error occurs during evaluation of the data, data is discarded and processing continues. If an error occurs during data evaluation, control is given to the error handler as described below.

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	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: Error Number (N4) Line Number (N4) Status (A1) Program Name (A8) Level (N2) 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.
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 **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) in the **Code** field.

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 type 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 item in a command sequence. `2` The keyword defined in this line is to be used as the second item in a command sequence. `3` The keyword defined in this line is to be used as the third item in a command sequence. `S` The keyword defined in this line is to be used as a synonym for the preceding keyword with item 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.

Field	Explanation
Use	Output field. The value displayed is determined by the value entered in the preceding **E** field: `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 ***Keyword Editor Options - Header 2*** in the section *Header Records*. `S` `SYNO`, the abbreviation for synonym, is displayed. `P` `PARM`, the abbreviation for parameter indicator, is displayed.
Keyword	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. 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 ***Keyword Editor Options - Header 2*** in the section *Header Records*.
IKN	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 *Editor Commands*). Each keyword is assigned a unique IKN, except synonyms, which take the IKN of their master term (see the **Keyword Editor screen** example above: `DISPLAY` and `SHOW`).
ML	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 *Keyword Runtime Options - Header 1* in the section *Header Records*. 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. For instance, in the **Keyword Editor screen** example above, keyword `MENU` requires only input of `M` while keyword `DISPLAY` requires input of `DI` to avoid ambiguity with keyword `DELETE`.
Comment	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* in the section *Header Records*. 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**.

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 the **Processor Source Maintenance** menu.
CHECK	Tests the keyword source for consistency.
EXIT	Returns to the **Processor Source 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 editorprogram 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) in the **Code** field.

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                                          Yes
       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 ***Runtime Action Editor*** in the section *Runtime Actions*). `EL` Invoke the Runtime Action Editor for a local function (see ***Runtime Action Editor*** in the section *Runtime Actions*). `+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. 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 ***Keyword Runtime Options - Header 1*** in the section *Header Records*).
Object	
Addition	
Global	If the function has been defined as a global command, `Yes` appears in this field.

Column	Explanation
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.

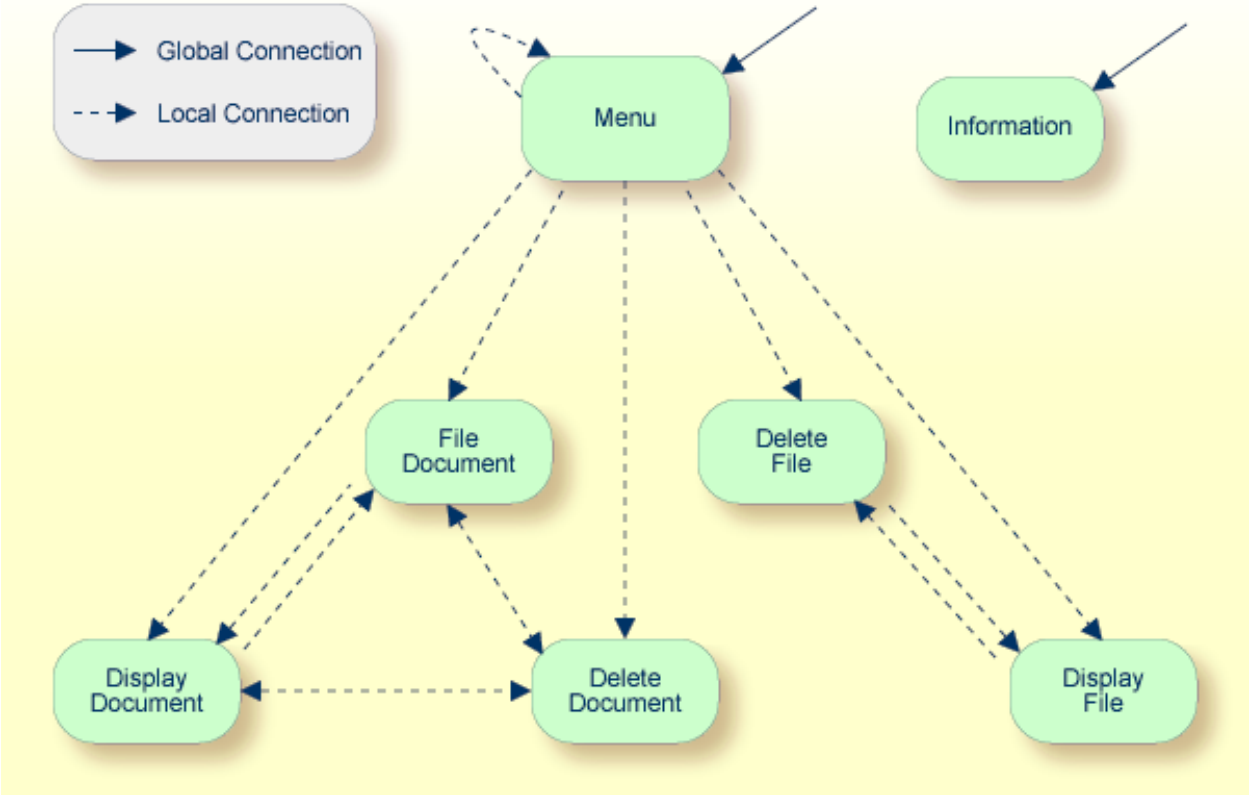
Direct Command QUICK-EDIT

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) in the **Code** field.

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 in the following section.

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 *not* 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 *Header Records*. 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 *Statements* 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 *Statements* 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 Source Maintenance** menu, enter function code C (Catalog Processor) in the **Code** field,
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) in the **Code** field.

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` in the **Code** field.

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 in 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` in the **Code** field.

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` in the **Code** field.

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` in the **Code** field.

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` in the **Code** field.

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` in the **Code** field.

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` in the **Code** field.

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 Apply Terminal Control 2	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'`.
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. Possible values are: `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 the Command line.

Field	Explanation
	D Display last command issued in the 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 *Terminal Commands* 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 SYSPCI Utility - Product Configuration and Initialization

72 SYSPCI Utility - Product Configuration and Initialization

The SYSPCI utility is used after a first-time installation of Natural or one of its add-on products which uses the Software AG Installer. It sets up a number of files, parameters and individual settings depending on your environment.

Using the SYSPCI utility, you can do the following:

- Enter the necessary information for the required Adabas files for your products, and add these files if they do not exist.
- Enter the database IDs of the required Adabas files into Natural's global configuration file.
- Enter the database IDs and file numbers of the new or existing Adabas files into the default parameter files for your products.
- Initialize your product.
- Optional, depending on the selected product: execute additional functions (such as loading product data).

Note: After an update installation, you need not invoke the SYSPCI utility if the required Adabas files and the parameters in the required default parameter files have already been set up previously (for example, after a first-time installation). Previously set up parameters will be kept with an update installation.

You can call the SYSPCI utility in different ways, as described in the following topics:

Configuring the Installed Products Using a Dialog Box

You can configure the installed Software AG products that the SYSPCI utility can detect in your environment.

The description below provides general information on how to use the SYSPCI utility, and it explains the options that are normally available for all products. For detailed information on the files that need to be set up for a specific product, see the installation documentation for that product.

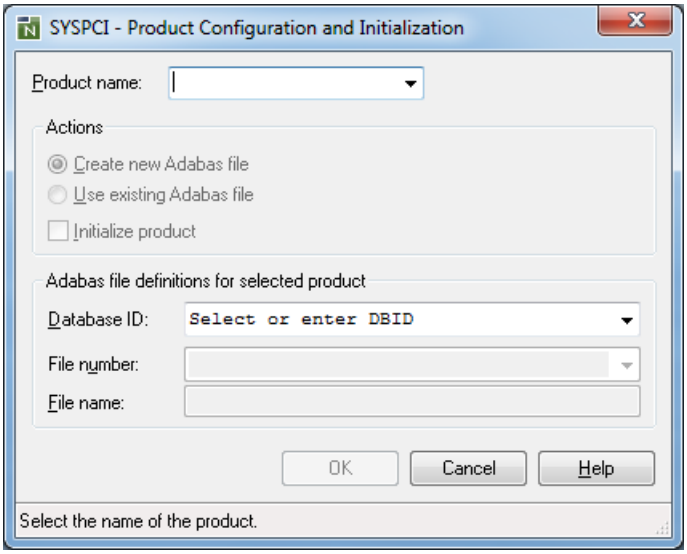
To configure an installed product

1 Enter the following command:

```
SYSPCI
```

Note: If you invoke the SYSPCI utility in an environment which is protected by Natural Security, Natural Security will validate the utility profile for SYSPCI.

The following dialog box appears.



Note: The above dialog box appears for the local environment. When a remote UNIX environment is active while you enter the `SYSPCI` command, a character screen is shown. In this case, see the description for the SYSPCI utility in the Natural for UNIX documentation. The SYSPCI utility only applies for Windows and UNIX. It is not available for remote mainframe environments.

When you open the **Product name** drop-down list box, you can see the installed Software AG products that have been detected in your environment. Exception: if only one installed Software AG product has been detected, a drop-down list box is not available, and the appropriate options for that product are immediately shown.

2 From the **Product name** drop-down list box (if available), select the product that you want to configure.

Important: It is recommended that you configure your products in the same sequence as listed in the **Product name** drop-down list box.

The content of the dialog box may change, depending on the selected product. If an Adabas file is found for the currently selected product, the corresponding drop-down boxes can be used to select the required values.

An action may be shown as disabled (grey) for a selected product. This means, that this option cannot be changed by the user. If a grey option is checked, it will be executed when you choose the **OK** button.

3 Specify the following information for the selected product:

Option	Description
Create new Adabas file or Use existing Adabas file	If **Create new Adabas file** is selected, the first file must not exist and a new Adabas file will be created. Depending on the selected product, there may be more than one file. If one or more of the other files already exist they can be used for the product. If **Use existing Adabas file** is selected, the first file must exist. Depending on the selected product, there may be more than one file. If one or more of the other files do not exist, they can be created. If the Adabas file exists already, however, the SYSPCI utility will only check whether the file has the correct structure (FDT). In both cases (new and existing file), the following actions will be performed: ■ The database ID of each required Adabas file will be entered into Natural's global configuration file. ■ The database ID and file number of each new or existing Adabas file will be entered into the default parameter file for your product.
Initialize product	This option is only available for products which have an initialization program. If selected (default), the initialization program for the selected product will be loaded and executed. If you want to activate the product, you have to select this option. **Note:** If you want to use this option, the database for which you specify the database ID must be online.
Database ID	The database ID of the Adabas file. When you open the drop-down list box, a list of all databases is shown which can be found on the machine. The list also shows whether a database is currently online or offline. The drop-down list box always shows the databases which have been found at the time when the SYSPCI utility was started. If the status of a database is changed afterwards or a new database is created (for example, by another user), the content of the drop-down list box does not change automatically. If you want to refresh the list, clear the contents of the **Database ID** field. Remote databases are also included in the list, provided they are defined in *XTS.config* (Net-Work), in *DBmapping.txt* (ADATCP) or in *natconfig.cfg* (NATTCP). The list also shows the TCP/IP address for the remote databases and the file in which they are defined.
File number	The number of a file in the selected database. This can be the number of an existing file or for a new file. When you open the drop-down list box, a list of all files is shown which can be found for the specified database ID.

Option	Description
	The Adabas system files are also shown in the drop-down list box, so that you can see which file numbers have already been assigned. However, you must not select an Adabas system file. Otherwise, an error will occur. When you specify the number for a new file, make sure that the **Create new Adabas file** option is selected.
File name	The name of the Adabas file. When you have selected an existing file, the corresponding file name is automatically shown. This name cannot be changed. When you have specified a new file number which does not yet exist, you can enter a file name (optional). If you do not enter a file name, a product-specific default name will be used.

For some products (such as Predict), you have to specify additional options. See the installation documentation for that product for further information.

4 Choose the **OK** button to start the configuration of the selected product.

After the selected actions for the selected product have been performed, a message such as the following is shown:

```
Function completed successfully.

The following actions have been performed by the SYSPCI utility:

- Loaded Adabas file with DBID 10 FNR 55 for product NCP
- Updated global configuration file for DBID 10
- Updated Natural parameter file NATPARM and set LFILE 190 to DBID 10 FNR 55
```

5 Choose the **OK** button to close the message box.

The dialog box for the SYSPCI utility is still shown and you can configure further products. The configuration of Natural Security, however, is an exception. In this case, Natural is terminated after the initialization program has been executed.

Calling the SYSPCI Utility with Direct Command Data

You can call the SYSPCI utility using a direct command that consists of keywords and their corresponding values. Thus, you can also use the SYSPCI utility in batch mode.

Exception: When the initialization of the Adabas file for Natural Security has been completed (by loading the initialization program with the INPL utility and executing it), the Natural session is

terminated by the INPL utility. Therefore, it is not possible to execute any additional commands after this step.

You can use the following keywords with the SYSPCI command (see also the examples below):

Keyword	Meaning
PRODUCT *	Product to be processed. Valid values: NCP for Natural Command Processor. PRD for Predict. NEE for Natural Engineer NSL for Natural Security Log. NSC for Natural Security. **Important:** It is recommended that you configure your products in the same sequence as listed above.
DBID *	Database ID of the Adabas file. **Note:** When using FUNCTION ADU or ALU you can leave this blank. In this case, the SYSPCI utility will use the DBID value found in the text member INST-<*productcode*> in library SYSPCI.
DBID2 **	Database ID of the second Adabas file if PRODUCT is PRD or NEE.
FNR *	File number of the Adabas file. **Note:** 1. When using FUNCTION ADU or ALU you can leave this blank. In this case, the SYSPCI utility will use the FNR value found in the text member INST-<*productcode*> in library SYSPCI. 2. When using FUNCTION ADA or ALL you can enter a negative value, for example -10. In this case, the SYSPCI utility will use the next free file number starting from the value given. In the above case it would start from file number 10.
FNR2 **	File number of the second Adabas file if PRODUCT is PRD or NEE.
FUNCTION or FCT	Function to be executed. Valid values: ADA: Create new Adabas file. ADU: Use existing Adabas file. This requires entering a valid DBID and FNR (see above) or entering the value. ADR: Use existing Adabas file on a remote system. This requires entering a valid DBID and FNR (see above) or entering the value. INT: Load and execute initialization program. INR: Load and execute initialization program using Adabas file on a remote system. INW: Initialize product without Adabas file specification. ALL: Both (ADA and INT). Default. ALU: Both (ADU and INT). ALR: Both (ADR and INR).

Keyword	Meaning
FILE-NAME	Name of the Adabas file if FUNCTION is ADA or ALL. Valid values: 16 characters without blanks.
FILE-NAME-2 or FILE-N2	Name of the second Adabas file if PRODUCT is PRD or NEE. Valid values: 16 characters without blanks.
SUBFUNCTION	Additional function to be executed. The valid values depend on the product. If PRODUCT is PRD: ■ PRC: Convert FDIC data. ■ PRP: Load FDIC description. ■ PRD: Load example data. ■ PRA: Both (PRP and PRD). ■ PR1: Convert FDIC data and load FDIC description. ■ PR2: Convert FDIC data and load example data. ■ PR3: Convert FDIC data and load FDIC description and example data.
END, STOP, EXIT, QUIT or .	Exit the SYSPCI utility. The keyword must be entered as a single command.
FIN	Exit the SYSPCI utility and terminate the Natural session. The keyword must be entered as a single command.

Notes:

1. The keywords marked with an asterisk (*) are mandatory.
2. The keywords marked with two asterisks (**) are mandatory for the corresponding products.
3. All other keywords are optional.

Examples

- **Batch Mode**
  Commands in the batch input file which is defined by the CMSYNIN profile parameter:

```
SYSPCI
FIN
```

  Data in the batch input file which is defined by the CMOBJIN profile parameter:

```
FUNCTION ALL PRODUCT PRD DBID 77 FNR 2002 DBID2 12 FNR2 2003
FUNCTION ALL PRODUCT NSC DBID 77 FNR 1600
END
```

See also *Natural in Batch Mode* in the *Operations* documentation.

- **Interactive Mode - Natural Command Line**

```
SYSPCI FUNCTION ALL PRODUCT NSL DBID 77 FNR 1601
```

- **Interactive Mode - Natural Stack**

```
C:\SoftwareAG\Natural\bin\natural stack=(SYSPCI FUNC ALL PROD NCP DBID 77 FNR ↩
1501: PROD NSL DBID 77 FNR 1601; FIN)
```

XVI Natural Profiler

This document provides information on profiling Natural applications in order to analyze program execution and code coverage.

Profiling Natural Applications	General information on the profiling options provided by Natural and NaturalONE.
Code Coverage of Natural Applications	General information on the options for code coverage provided by Natural and NaturalONE.
Basic Concepts of the Profiler Utility	Basic concepts of the Profiler utility.
Using the Profiler Utility	Evaluating the event data from the Profiler resource files and code coverage.
Natural Profiler MashApp	Evaluating Profiler data on an interactive MashZone dashboard.

Note: The features of the NaturalONE Profiler and NaturalONE code coverage are described in the relevant sections of the NaturalONE documentation. The use of the Natural Profiler for UNIX and Windows is described with the `PROFILER` profile parameter in the Natural *Parameter Reference* documentation. The use of Natural code coverage for UNIX and Windows is described with the `COVERAGE` profile parameter in the Natural *Parameter Reference* documentation.

73 Profiling Natural Applications

Introducing Profiling

A profiler is a tool for dynamic program analysis. It measures the frequency and duration of instructions to simplify program optimization.

The Natural Profiler is used to profile Natural applications. It collects profiling data whenever a defined Natural event occurs, for example, when a program starts or before a database is called. The Natural Profiler visualizes the recorded event data as an event trace and the calling structure of the executed Natural objects as a program trace. The performance evaluation provided by the Natural Profiler shows the time consumption and hit count of the executed objects, Natural statements and program lines.

You can view Natural Profiler event data in the Profiler utility output or export the data in text or table format. You can visualize Natural Profiler performance analyses in NaturalONE (Software AG's Eclipse-based development environment) or MashZone (Software AG's tool for creating interactive business dashboards).

A Natural Profiler analysis serves as the basis for performance optimization of a Natural application. The Natural Profiler provides you with a very fast overview about the time-consuming parts of a Natural application. No code modification is required, and moreover, just basic knowledge of the application is sufficient.

Platform-Specific Profiling

You can profile Natural applications on UNIX, Windows and mainframe platforms. How to profile a Natural application depends on the platform and the application processing mode used:

Mainframes

- Mainframe interactive applications are profiled with the NaturalONE Profiler or the Profiler utility in online mode.
- Mainframe interactive applications executed remotely from Natural Studio or RPC are profiled with the Profiler utility in batch mode.
- Mainframe batch applications are profiled with the Profiler utility in batch mode.

UNIX and Windows

- UNIX and Windows interactive applications are profiled with the NaturalONE Profiler or the Natural Profiler for UNIX and Windows, respectively.
- UNIX and Windows batch applications are profiled with the Natural Profiler for UNIX and Windows, respectively.

Profiling Tools

This section summarizes the key features of the Natural profiling tools:

Features of the NaturalONE Profiler

- Profiles interactive Natural applications from UNIX, Windows or mainframe platforms in an Eclipse-based development environment.
- Reads and analyzes Profiler resource files containing event data collected by the mainframe Profiler utility in batch mode or by the Natural Profiler for UNIX and Windows.
- Provides features for big data handling:
  - Event filter,
  - Sampling technique,
  - Data consolidation.
- Performance analyses of programs, statements and program lines:
  - CPU time,
  - Elapsed time,
  - Hit count.
- Displays an event trace.
- Provides direct navigation from a profiled program line to the corresponding source code.
- Saves and reloads the Profiler data as an XML-formatted file.

Features of the Natural Profiler for UNIX and Windows

- Profiles interactive or Natural batch applications from UNIX or Windows platforms.
- Provides features for big data handling:
  - Event filter,
  - Sampling technique,
  - Data consolidation.
- Saves the Profiler data as a Profiler resource file.

Features of the Profiler Utility

Online Mode (Mainframes)

- Profiles interactive Natural applications from mainframe platforms.
- Provides an event filter.
- Displays an event trace.
- Saves the Profiler data in a table format.
- Saves the Profiler data as a Profiler resource file.

Note: The amount of data collected by the Profiler utility in online mode is restricted by the relatively small size of the Natural Data Collector buffer which works in a wrap-around mode. Moreover, when running under CICS or Com-plete, the CPU time is not provided. In general, we recommend that you use the NaturalONE Profiler for profiling interactive Natural mainframe applications because the NaturalONE Profiler has no size restrictions and supports CPU performance analyses.

Batch Mode (Mainframes)

- Profiles Natural batch and Natural RPC applications from mainframe platforms.
- Profiles mainframe interactive applications executed remotely from Natural Studio.
- Provides features for big data handling:
  - Event, program, count and time filters,
  - Sampling technique,
  - Data consolidation.
- Saves Profiler data as a Profiler resource file.
- Reads and analyzes Profiler resource files.
- Prints program and event traces.
- Analyzes program performance.
- Evaluates transaction response times.
- Collects and displays Profiler properties and statistics.
- Exports Profiler data for MashZone visualization.

Batch Mode (UNIX and Windows)

- Reads and analyzes Profiler resource files.
- Provides features for big data handling:
  - Data consolidation.
- Saves consolidated Profiler data as a Profiler resource file.
- Prints program and event traces.

- Analyses program performance.
- Displays Profiler properties and statistics.
- Exports Profiler data for MashZone visualization.

Features of the Natural Profiler MashApp

- Visualizes Profiler data on a graphical, interactive MashZone dashboard.
- Analyzes application performance with selection criteria such as library, program, program line and user:
  - CPU time,
  - Elapsed time,
  - Adabas command time,
  - Hit count.
- Displays Profiler properties and statistics.

Natural Profiler Evaluations

The evaluation criteria provided by the Natural profiling tools are summarized in the following table:

Evaluation	Profiling Tool	Description
Program Summary	Profiler utility (batch)	Shows the CPU time spent for each Natural object that executed and the Natural events that occurred in an object. See also *Example of a Program Summary*.
Line Summary	Profiler utility (batch)	Shows the CPU and elapsed time spent during Natural program execution for each individual source line and the number of Natural events that occurred in the line. See also *Example of a Line Summary*.
Transaction Summary	Profiler utility (batch)	Shows the response time spent for each individual transaction and the number of Natural events that occurred during transaction execution. For more information, see *Transaction Summary*. See also *Example of a Transaction Summary*.

Evaluation	Profiling Tool	Description
Hot Spots	NaturalONE Profiler	Shows the CPU and elapsed time used by Natural objects, statements and program lines and how often an object or statement executed. From a profiled program line you can directly navigate to the corresponding source code line. See also the appropriate description of hot spots in *Using the Natural Profiler* in the *NaturalONE* documentation.
MashZone Evaluation	Natural Profiler MashApp	Visualizes the Profiler data on an interactive MashZone dashboard. You can evaluate the distribution of the CPU and elapsed time, the Adabas command time or the hit count and select criteria for the distribution. See also the example of a **MashZone evaluation**.
Program Trace	Profiler utility (batch)	Shows the program flow of the profiled application in the call hierarchy and the number of events that occurred. See also ***Example of a Program Trace***.
Event Trace	NaturalONE Profiler, Profiler utility (batch)	Lists the recorded event data in chronological order. See also ***Example of an Event Trace*** and the appropriate description in *Using the Natural Profiler* in the *NaturalONE* documentation.
MashZone Profiler Properties, **Profiler Statistics**	Natural Profiler MashApp, Profiler utility (batch)	Lists Profiler properties such as the Profiler revision, and statistics of the monitored application that show, for example, the total CPU and elapsed time. See also the example of **MashZone Profiler properties** and ***Profiler Statistics***.

74 Code Coverage of Natural Applications

This document provides general information on code coverage of Natural applications.

Introducing Code Coverage

In general, code coverage measures the degree to which the source code of a program is executed. It is often used for systematic software testing. The higher the code coverage percentage is, the lower is the chance that the code contains undetected software bugs in code that is not executed.

The Natural code coverage is used to monitor the executed statements of a Natural application. It collects the coverage data while the application is executed and provides tools to analyze the collected data afterwards.

The **Code Coverage** view of NaturalONE (Software AG's Eclipse-based development environment) and the **Program Coverage** table of the Profiler utility show - expressed as a percentage - how many of the statements of the Natural objects have been executed. The Natural Coverage Plugin for Jenkins visualizes the outcome of a Natural coverage cycle directly in the Jenkins job result pages.

In the NaturalONE editor and in the **Statement Coverage** table of the Profiler utility you can see, which individual statement lines of a Natural object have been executed. Here you can also see the statement lines which have been missed or have only been partly covered.

If a statement source uses multiple lines, only the line in which the statement begins is mentioned in the coverage reports.

You can export the coverage data with the Profiler utility output in text or table (CSV) format. The CSV table can be analyzed with a spreadsheet software such as Microsoft Excel.

GP and Source Coverage

If a Natural source contains `INCLUDE` statements, the corresponding copycode is included in the generated object (the **GP**). For the coverage, we can monitor two statement counts:

1. The number of the statements in the GP which includes all copycodes recursively (a copycode can include further copycodes).
2. The number of the statements in the source which does not include the copycodes.

The GP coverage reflects the percentage of the covered statements in the GP including copycodes; whereas the source coverage reflects the percentage of the covered statements in the source not including copycodes.

Platform-Specific Code Coverage

You can perform the Natural code coverage on UNIX, Windows and mainframe platforms. How to proceed depends on the platform and the application processing mode used:

Mainframes

- Code coverage of mainframe interactive applications remotely executed from Natural Studio or RPC is performed using the Profiler utility in batch mode.
- Code coverage of mainframe batch applications is performed using the Profiler utility in batch mode.

Note: Code coverage is not available for mainframe interactive applications running locally on a mainframe or remote from NaturalONE.

UNIX and Windows

- Code coverage of UNIX and Windows interactive applications is performed with the NaturalONE code coverage or the Natural code coverage for UNIX and Windows, respectively.
- Code coverage of UNIX and Windows batch applications is performed with Natural code coverage for UNIX and Windows, respectively.

Code Coverage Tools

This section summarizes the key features of the Natural profiling tools:

- Features of the NaturalONE Code Coverage
- Features of the Natural Code Coverage for UNIX and Windows
- Features of the Profiler Utility

Features of the NaturalONE Code Coverage

- Reads and analyzes Natural code coverage resource files containing coverage data collected by the mainframe Profiler utility in batch mode or by the Natural code coverage for UNIX or Windows.
- Interactive Natural applications from UNIX or Windows can be covered by activating the Natural code coverage for UNIX or Windows and reading the corresponding Natural code coverage resource file.
- The Natural code coverage view shows which percentage of the statements of the Natural objects have been executed.
- From the Natural code coverage view the involved Natural objects can be edited. The NaturalONE editor displays all covered lines with a green background.

Note: Interactive code coverage of Natural applications from mainframe platforms is currently not supported.

Features of the Natural Code Coverage for UNIX and Windows

- Code coverage of interactive or Natural batch applications from UNIX or Windows platforms.
- Provides features for big data handling:
  - automatic event filter,
  - automatic data consolidation.
- Saves the code coverage data as a Natural code coverage resource file.

Features of the Profiler Utility

Batch Mode (Mainframes)

- Code coverage of Natural batch applications from mainframe platforms.
- Code coverage of mainframe interactive applications remotely executed from Natural Studio or against a Natural RPC server.
- Provides features for big data handling:
  - program, count and time filters,
  - automatic event filter,
  - automatic data consolidation.
- Saves code coverage data as a Natural code coverage resource file.
- Reads and analyzes Natural code coverage resource files.

- Lists the **Program Coverage** table for all accessed Natural objects with
  - percentage of the covered statements,
  - number of covered statements,
  - number of missed (not covered) statements and
  - total number of statements of the object.
- The **Statement Coverage** lists the source of each accessed Natural objects and shows for each line the percentage of the covered statements.
- Exports Natural code coverage data in CSV (comma-separated values) format which can be further analyzed with a spreadsheet software (e.g. Microsoft Excel).
- Collects and displays Profiler and code coverage properties and statistics.

Note: On the mainframe, there is no one-to-one relationship between a Natural source code statement and the corresponding object code in the cataloged object. The Natural code coverage on the mainframe monitors the object code rather than the Natural source code. Therefore, multiple Natural statements can be merged into one coverage entry and conversely, one Natural statement can cover multiple coverage entries.

Batch Mode (UNIX and Windows)

- Reads and analyzes Natural code coverage resource files.
- Lists the **Program Coverage** table for all accessed Natural objects with
  - percentage of the covered statements,
  - number of covered statements,
  - number of missed (not covered) statements and
  - total number of statements of the object.
- The **Statement Coverage** lists the source of each accessed Natural objects and shows for each line the percentage of the covered statements.
- Exports Natural code coverage data in CSV (comma-separated values) format which can be further analyzed with a spreadsheet software (e.g. Microsoft Excel).

Note: On Windows and UNIX, missed statements are not collected. Therefore the Statement Coverage can only mark lines containing covered statements and the coverage of these lines is always 100%.

Features of the Natural Code Coverage Spreadsheet

- Template for coloring the Natural code coverage data exported in CSV (comma-separated values) format by the Natural Profiler utility.
- Program and copycode coverage with source and GP counters for
  - percentage of the covered statements,
  - number of covered statements,
  - number of missed (not covered) statements and
  - total number of statements of the object.
- Statement Coverage of the object source whereby the lines are colored in
  - green – if all statements of the line are covered,
  - yellow – if the statements of the line are partly covered,
  - red – if all statements of the line are missed,
  - gray – if the line is empty or contains only comments.
- Profiler and code coverage properties and statistics (for mainframe data).

Note: A Microsoft Excel spreadsheet template for Natural code coverage is available as a resource in the Natural Profiler library `SYSPRFLR` on UNIX and Windows.

Natural Code Coverage Evaluations

This section describes the evaluations provided by the Natural code coverage tools:

- Program Coverage
- Line and Statement Coverage
- Profiler Properties and Statistics

Program Coverage

The program coverage provides you with an overview of the programs executed and the amount of the code that has been covered by the application.

Program Coverage Report

The **Program Coverage** report of the Profiler utility shows the coverage (in percentage of the total number of statements) of each Natural object executed. It shows for each object how many statements have been covered or missed and the total number of statements. In addition, it summarizes the values for all objects in a library and the totals over all libraries.

If the output is written in text format, only the GP coverage is provided. If the data is exported in CSV (comma-separated values) format, the source coverage is given as well. Additionally, the counters for all included copycodes are printed.

The following is an example for text format:

```
Program Coverage
----------------
Library  Object   Ty Coverage%  Covered    Missed    Total
COVDEMO  TESTCOVN N      84.0%       37         7       44
COVDEMO  TESTCOVP P      69.2%        9         4       13
COVDEMO  -------- --      80.7%       46        11       57
Totals   -------- --      80.7%       46        11       57
```

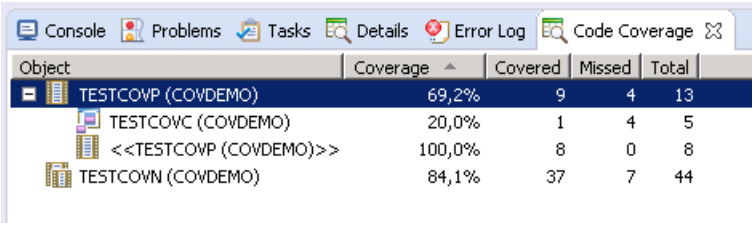
In the **Program Coverage** example above, 69.2% of 13 statements in the `TESTCOVP` program were covered, corresponding to 9 covered and 4 missed statements. 80.7% of the statements of the accessed objects in the library `COVDEMO` were covered, which is also the total value for the whole application run.

Code Coverage View

The **Code Coverage** view of NaturalONE shows the coverage (in percentage of the total number of statements) of each Natural object executed. It shows for each object how many statements have been covered or missed and the total number of statements. If copycodes are included, the object node can be opened to view the coverage of the copycode. In general, the counters reflect the GP coverage (copycodes included). The source coverage (copycodes not included) is displayed in the line where the object name is enclosed in the `<< >>` brackets.

From any line you can directly navigate to the corresponding source code to view the statement coverage.

Example:



In the example above, the `TESTCOVP` program has a GP coverage of 69.2 percent whereby in the program itself all 8 statements are covered (100% source coverage) and in the included copycode `TESTCOVC` only 1 of 5 statements was covered.

Line and Statement Coverage

The statement coverage shows which lines of the program have been executed. For mainframe data, the Profiler utility also indicates which lines containing statements have not been executed or are only executed partly.

Statement Coverage Report

The **Statement Coverage** report of the Profiler utility shows for each source line the coverage of the statements in the line. If the data is exported in CSV (comma-separated values) format, the number of covered or missed statements and the total number of statements in the line are printed as well. The Microsoft Excel spreadsheet template delivered with Natural on UNIX and Windows, can be used to color the lines according to the coverage.

If a source contains an `INCLUDE` statement, the corresponding copycode source is included in the report right after the `INCLUDE` statement.

The following is an example for an export in CSV format colored using a Microsoft Excel spreadsheet:

Line	Source	Coverage%	Covered	Missed	Total
10	* Test function Coverage		0	0	0
20	* Subprogram TESTCOVN		0	0	0
30	DEFINE DATA		0	0	0
40	PARAMETER		0	0	0
50	1 FUNC (I2) /* function		0	0	0
60	1 RET-CODE (I4) /* Return code		0	0	0
70	END-DEFINE		0	0	0
80	*		0	0	0
90	/* Return 0 by default		0	0	0
100	RESET RET-CODE	100	1	0	1
110	*		0	0	0
120	DECIDE ON FIRST VALUE OF FUNC	100	1	0	1
130	VALUE 0	50	1	1	2
140	PRINT 'Test function 0'	0	0	1	1
150	VALUE 1	66	2	1	3
160	PRINT 'Test function 1'	100	1	0	1
170	VALUE 2	100	3	0	3
180	PRINT 'Test function 2'	100	1	0	1
190	VALUE 3	100	3	0	3
200	PRINT 'Test function 3'	100	1	0	1
210	VALUE 4	100	3	0	3
220	PRINT 'Test function 4'	100	1	0	1
230	VALUE 5	100	3	0	3
240	PRINT 'Test function 5'	100	1	0	1
250	VALUE 6	100	3	0	3
260	PRINT 'Test function 6'	100	1	0	1
270	VALUE 7	100	3	0	3
280	PRINT 'Test function 7'	100	1	0	1
290	VALUE 8	100	3	0	3
300	PRINT 'Test function 8'	100	1	0	1
310	VALUE 9	33	1	2	3
320	PRINT 'New test function 9'	0	0	1	1
330	NONE VALUE	100	1	0	1
340	RET-CODE := 1 /* Unsupported function	0	0	1	1
350	END-DECIDE		0	0	0
360	*		0	0	0
370	END	100	1	0	1

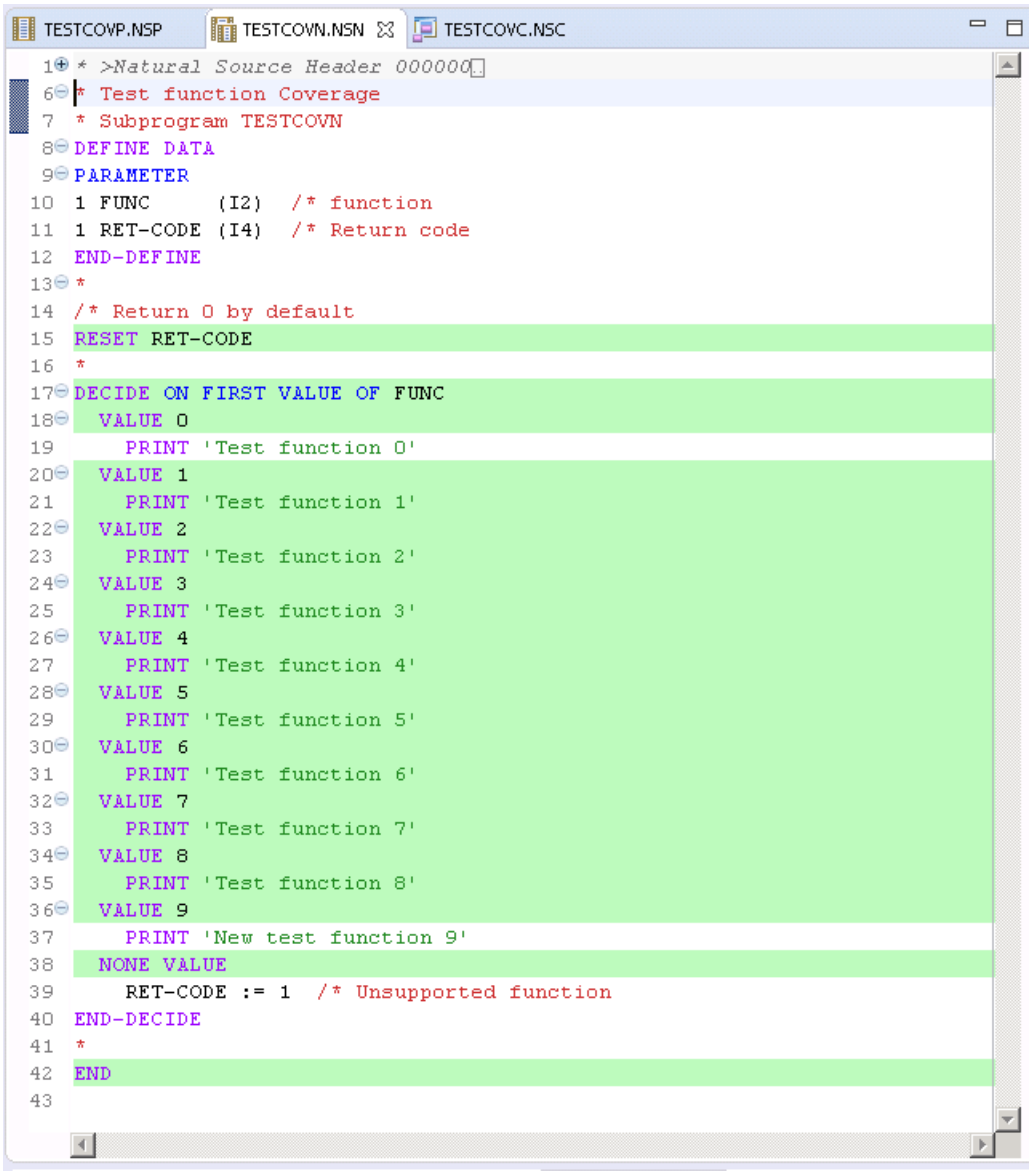
The three red lines of the subprogram `TESTCOVN` have not been executed. Thus the test run does not cover the new test function `9`. It also neither covers the (old) function `0` nor the case when the subprogram is called with an unsupported function.

The data originates from the mainframe. Therefore, the counts refer object code statements rather than Natural statements. A Natural `VALUE` statement can correspond up to 3 object code statements. The yellow lines refer to `VALUE` statements where some of the object code has been covered and some not.

NaturalONE Source Editor

If the source editor is opened from the **Code Coverage** view in NaturalONE, the source is colored according to code coverage. Every line in which one or more statements are covered, is colored with a green background.

Example:



The source editor shows all lines in which at least one statement has been executed with a green background. Therefore, all lines except line 19, 37 and 39 of the `DECIDE` statement have been executed.

Profiler Properties and Statistics

The **Profiler properties and statistics** provided by the Natural Profiler utility lists Profiler properties such as the Profiler revision, and statistics of the monitored application that show, for example, the total CPU time and the elapsed time. For a code coverage run, it shows also the coverage statistics.

Example

```
************************************************************************
* 13:30:48          ***** NATURAL PROFILER UTILITY *****         2017-09-04
* User SAG                     - Statistics -                       COVREAD
*
* General Info
* Machine class ..................... MAINFRAME
* Environment ....................... Batch
...
* Coverage
* Coverage ........................... ON
* Missed statements recorded ......... ON
* Coverage records ................... 60
* Program information records ........ 3
* Coverage records/block ............. 60
* Bytes/coverage record .............. 10.3
* Programs covered ................... 2
* Statement coverage (percent) ....... 80.7
* Statements covered ................. 46
* Statements missed .................. 11
* Statements total ................... 57
```

75 Basic Concepts of the Profiler Utility

The Profiler utility reads and processes Profiler resource files created by the Natural Profiler for UNIX and Windows and Natural code coverage for UNIX and Windows. It provides functions for data consolidation (aggregation), event tracing and program tracing. It offers a program summary, a line summar and a transaction summary and displays the Profiler properties and statistics. For Natural code coverage data, program and statement coverage reports are provided. The resulting data can be exported to a file in text or CSV (comma-separated values) format, or in the format expected by the **Natural Profiler MashApp**.

Additionally, the Profiler utility provides functions to pause and to restart the Profiler data collection.

The Profiler utility runs in batch mode only.

Data Consolidation, Code Coverage and Data Processing

The Profiler utility uses technology introduced with the NaturalONE Profiler such as the NATRDC1 user exit and the Profiler data pool. Therefore, the processing of the event data is restricted to NaturalONE users who can use the NaturalONE Profiler and the Profiler utility to evaluate the event data. The data consolidation and processing functions of the Profiler utility (`CONSOLIDATE`, `READ`, `MASHZONE`, `LIST` and `DELETE`) have to be activated before they can be used. The activation is described in ***Prerequisites***.

This section covers the following topics:

- Data Consolidation
- Natural Code Coverage
- Data Processing

Data Consolidation

When a Natural application is profiled, the Natural Profiler collects one record for each event. Depending on the application, this can produce huge amounts of data, especially when Natural statements are monitored. The more data the Profiler generates, the more time is required to transport the data from the server to the NaturalONE client.

The Profiler utility offers a server-side data consolidation which significantly reduces the amount of data while increasing the transport flow rate. The Profiler data consolidation combines similar records into one consolidated record containing aggregated time values and a hit counter. The consolidated data is written to a resource file which has the same name as the corresponding unconsolidated resource file but an extension `.nprc` (Natural Profiler resource consolidated).

During profiling, the data can be consolidated immediately by switching off the `EVENTTRACE` subparameter of the `PROFILER` parameter. See *PROFILER - Profile a Natural Session* in the *Parameter*

Reference documentation. Unconsolidated data of an NPRF file can be consolidated later with the Profiler utility `CONSOLIDATE` function.

Example

A Natural statement executes 1000 times in a `FOR` loop. The unconsolidated data contains 1000 records for each execution of the statement. Each record contains the event time and the CPU timestamp, besides other information. The Profiler consolidation combines these 1000 records into one consolidated record. All common information (like the library or program name) is kept, the elapsed time and the CPU time of each execution of the statement is determined, summarized and saved in the consolidation record. Additionally, a hit count of `1000` is recorded.

Notes:

1. An NPRC resource file that has been consolidated on the server side contains the same hot spot values as the corresponding unconsolidated NPRF resource but opens much faster with NaturalONE.
2. The consolidated data does not contain the event history (timestamps). Therefore, it is not possible to view the event trace when you open an NPRC resource in NaturalONE.
3. Data consolidation is a prerequisite if you want to analyze the event data in MashZone by using the **Natural Profiler MashApp**.

Natural Code Coverage

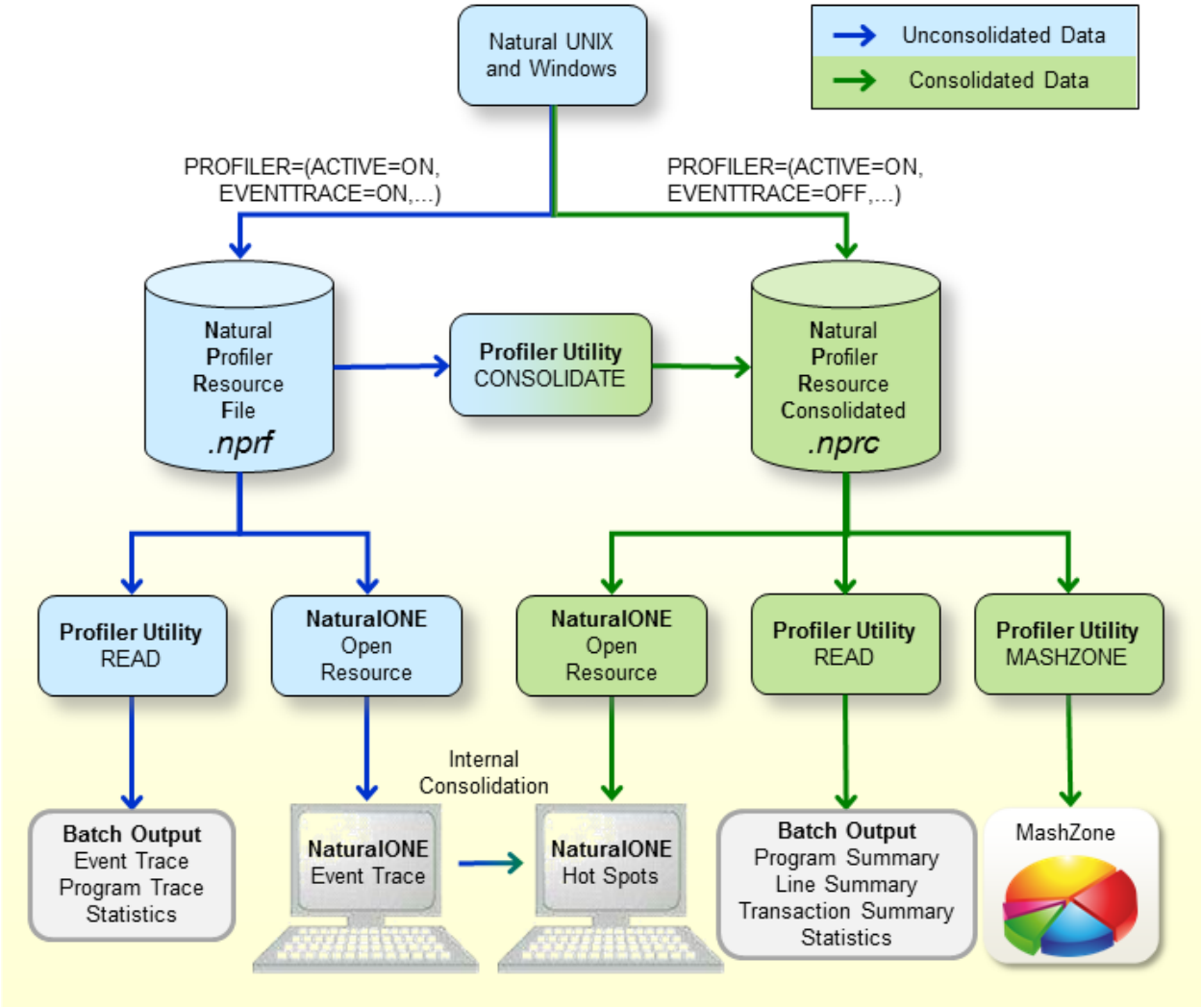
Natural code coverage is used to monitor executed and not-executed statements of a Natural application. It is started by switching on the `ACTIVE` subparameter of the `COVERAGE` profile parameter described in the *Parameter Reference* documentation.

For code coverage, Natural code coverage automatically uses an event filter so that only the program information (`PI`) and Natural statement (`NS`) events are collected. The data is automatically consolidated before it is written to a Natural NCVF resource file.

When the NCVF coverage resource file is analyzed with the Profiler `READ` function, the source of the monitored programs is read and the lines are marked according to the coverage of the statements in the line.

Data Processing

The following graphic shows how the Profiler utility processes unconsolidated and consolidated data:



The graphic is explained in the following section:

- When a Natural application on UNIX or Windows is profiled by switching on the `ACTIVE` subparameter of the `PROFILER` profile parameter, the resulting event data is written to a Natural Profiler resource file (NPRF) or a Natural Profiler resource consolidated (NPRC) file depending on the setting of the `EVENTTRACE` subparameter of the `PROFILER` parameter. For `EVENTTRACE=ON`, the data is written to an NPRF resource file, for `EVENTTRACE=OFF`, it is written to an NPRC resource file.
- The Natural Profiler resource file (extension `.nprf`) contains the event data in an unconsolidated format, which means that there is one record for each event.

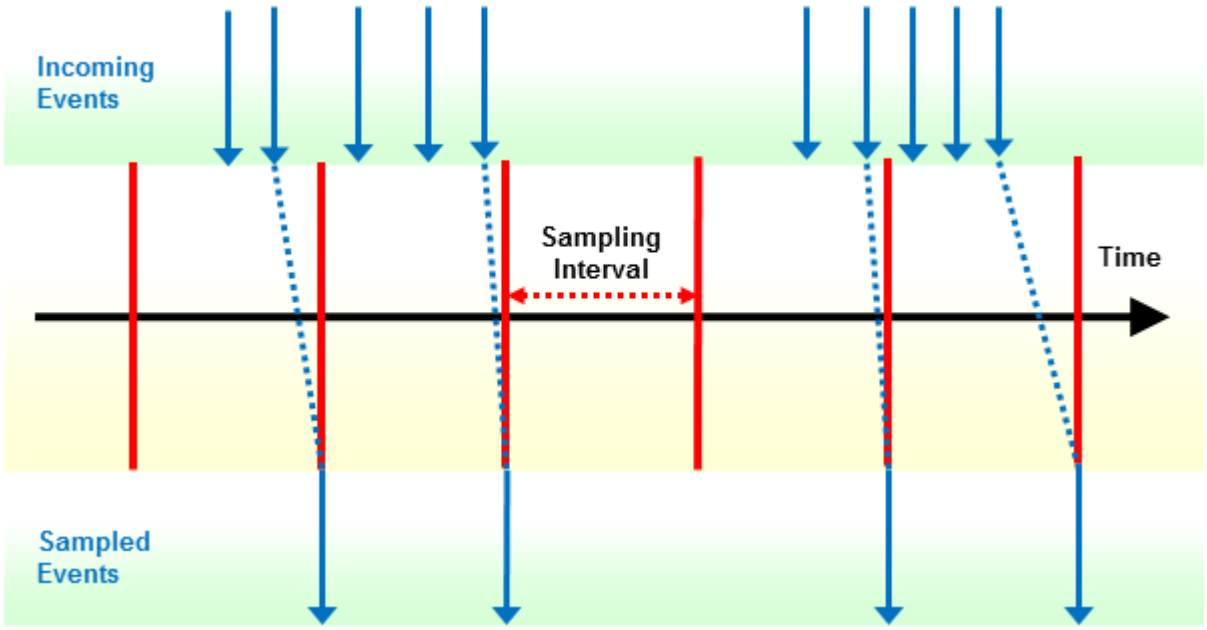
- The Profiler utility `READ` function reads the event data from the NPRF resource file. It provides an event trace, a program trace and the Profiler statistics. The resulting data can be exported to a file in text or CSV (comma-separated values) format.
- If the NPRF resource file is opened from NaturalONE, the unconsolidated event data is listed on the NaturalONE **Event Trace** page.
- The NaturalONE **Hot Spots** page shows the event data in a consolidated form. If the data derives from an NPRF resource file, NaturalONE consolidates the data internally.
- The Profiler utility `CONSOLIDATE` function reads the event data from the NPRF resource file, consolidates it and writes it to an NPRC resource file.
- The Natural Profiler resource consolidated file (extension `.nprc`) contains the event data in a consolidated format, which means that similar records are aggregated in one consolidated record. In general, an NPRC resource file is much smaller than the corresponding NPRF resource file and, therefore, much quicker to process.
- If the NPRC resource file is opened from NaturalONE, the consolidated event data is shown on the **Hot Spots** page. It is not possible to view the event trace because the NPRC resource file does not contain the data of each single event.
- The Profiler utility `READ` function reads the event data from the NPRC resource file. It provides a trace of the consolidated records, a program summary, a line summary, a transaction summary and Profiler statistics. The resulting data can be exported to a file in text or CSV (comma-separated values) format.
- The Profiler utility `MASHZONE` function reads the event data from the NPRC resource file and exports it in CSV (comma-separated values) format as expected by the **Natural Profiler MashApp**.
- The **Natural Profiler MashApp** visualizes the Profiler event data and statistics in MashZone.

Sampling

In general, profilers are classified into event-based or statistical profilers. Statistical profilers, which operate by sampling, interrupt the operating system at regular intervals to receive the profiling data. The resulting data is not exact but a statistical approximation.

The Natural Profiler is an event-based profiler. It receives control and collects the profiling data whenever a Natural event occurs. Although the Natural Profiler does not interrupt the operating system, it offers a sampling technique that generates the same profiling data as statistical profilers.

Natural Profiler sampling works like a filter: it eliminates all events except the last one in a sampling interval. Additionally, it replaces the event CPU timestamp by the subsequent sampling time. This way, the Natural Profiler only collects those events that were active at the beginning of a sampling interval.



If you use Profiler sampling, consider the following:

- Natural Profiler sampling provides a good estimation of the consumed CPU time. It does not provide other estimations such as hit counts, elapsed times, and Adabas times.
- Natural Profiler sampling is a statistical approach which reduces the number of events severely with nearly the same CPU time results.
- The smaller the sampling interval, the more accurate the result.
- The higher the sampling interval, the less data is produced.
- The resulting event duration is a multiple of the sampling interval.
- The sampling generates at most one record per sampling interval.
- Events which spent more time than a sampling interval need one record only.
- The session termination (`ST`) event is recorded unchanged.

If the total application CPU time is known and sampling is used, the number of events can be estimated:

$$\text{Number of events} \approx \frac{\text{Total CPU time in microseconds}}{\text{Sampling interval}}$$

Example

In the following example application, the program `XPROF` calls three subprograms. The application is profiled twice:

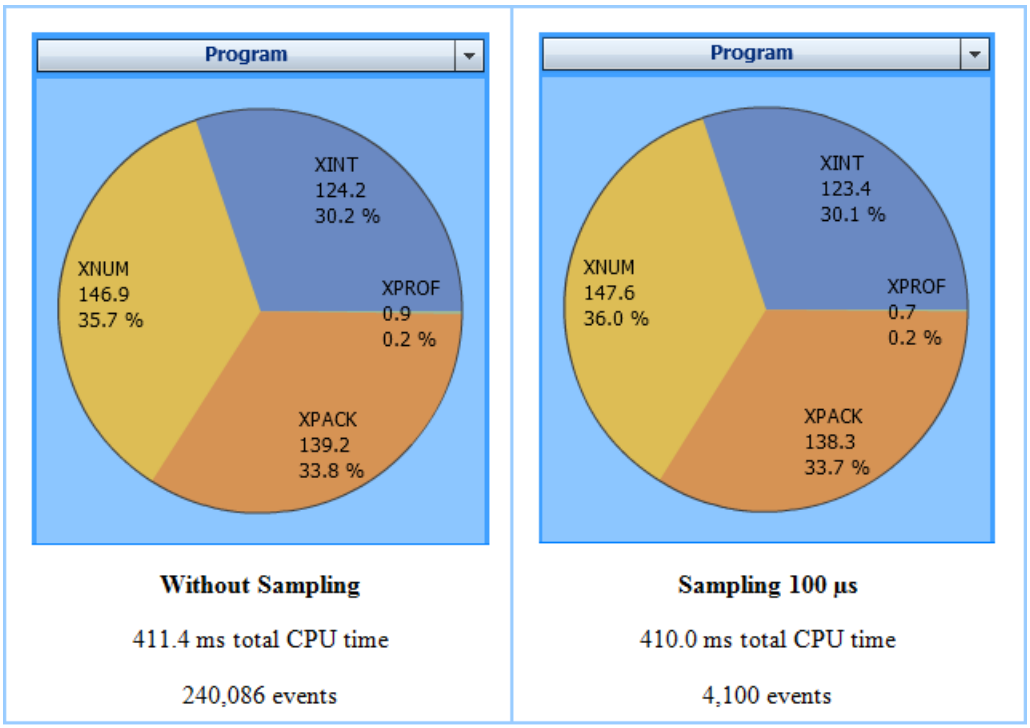
1. Without sampling.

2. With sampling whereby a sampling interval of 100 microseconds is used.

For sampling, the following subparameters of the `PROFILER` profile parameter are used:

```
PROFILER=(ACTIVE=ON,SAMPLING=ON,INTERVAL=100)
```

The Natural Profiler MashZone pie charts below show for each program the name of the program, the CPU time spent (in units of milliseconds) and the CPU time percentage with respect to the total CPU time. The left chart reflects the run without sampling and the right chart the run with sampling. Although the number of events has been reduced by the sampling to about 1.7 percent, the resulting CPU time and distribution are nearly the same.



Without Sampling	Sampling 100 µs
411.4 ms total CPU time	410.0 ms total CPU time
240,086 events	4,100 events

Profiling Long-Running Applications

Profiling a long-running batch application can produce a huge amount of data, especially when Natural statements are monitored.

This section describes how to minimize the number of events to be monitored while keeping essential information:

- Start and Pause Profiling
- Set Filters
- Use Sampling for CPU Analysis

- Use Server-Side Data Consolidation

Start and Pause Profiling

- If a Natural session executes multiple Natural applications, pause the Profiler for applications which are not of interest and restart it for applications of interest.
- Eventually, use the application programming interface (API) to start and pause profiling at specific points in the application.

Example

A Natural batch session executes three Natural applications. From these three applications, only the second one is of interest for a Profiler analysis.

Pause profiling before the first application executes, start profiling before the second application executes, and pause profiling again before the third application executes as in the example below:

```
PRFPAUSE
APP-01
PRFSTART
APP-02
PRFPAUSE
APP-03
FIN
/*
```

This way, profiling only affects the second application and has no impact on the performance of the other applications. Note that the programs `PRFPAUSE` and `PRFSTART` have to be copied into the user library.

Set Filters

- Statement events have the most impact on the performance and quantity. The other events have only a low impact on the performance but enlarge the quantity. Monitor statement events only if you really need them. Monitor from the non-statement events only those you want to analyze.

  For example, if you want to view in NaturalONE the program hot spots but neither the statement nor the line hot spots, the following setting of the `PROFILER` profile parameter is sufficient:

  ```
  PROFILER=(ACTIVE=ON,EVENT=(S,P),...)
  ```

  With this setting, only the program and session events needed for the program hot spots are monitored.

Use Sampling for CPU Analysis

For the CPU analysis of a long-running application, we recommend **sampling**. If you use already filter settings to reduce the number of events, you can additionally activate sampling to reduce the number of events further.

Most event data is generated when statements are collected. Therefore, sampling will often be used in conjunction with statement collection. For very long-running applications, however, it might be helpful to use sampling even if no statements are collected. If you use sampling without statement collection, we recommend a sampling interval that is higher than that specified when statements are collected.

Sampling has only restricted impact on the Profiler performance but it can reduce the amount of data dramatically. The formula in the section ***Sampling*** rearranged here can be used to choose a sampling interval so that the number of events is equal to or less than an approximate value:

$$\text{Sampling interval} \geq \frac{\text{Total CPU time in microseconds}}{\text{Approximate number of events}}$$

For example, a batch application requires 40 minutes of CPU time (2,400,000,000 µs). Sampling should restrict the number of events to at most 500,000 events. The corresponding sampling interval can be calculated with the formula above.

$$\text{Sampling interval} \geq \frac{2{,}400{,}000{,}000}{500{,}000} = 4{,}800$$

Set the `PROFILER` profile parameter as follows:

```
PROFILER=(ACTIVE=ON,SAMPLING=ON,INTERVAL=4800)
```

See *PROFILER - Profile a Natural Session* in the *Parameter Reference* documentation.

Use Server-Side Data Consolidation

If you want to analyze the performance of the event data and do not require an event or program trace, we recommend that you consolidate the event data on the server side. The Profiler data consolidation combines similar records into one consolidated record containing aggregated time values and a hit counter.

The event data can be consolidated during data collection by switching off the `EVENTTRACE` subparameter of the `PROFILER` profile parameter. See *PROFILER - Profile a Natural Session* in the *Parameter Reference* documentation.

Unconsolidated event data of an NPRF (Natural Profiler resource file) resource file can be consolidated with the Profiler utility `CONSOLIDATE` function as described in the section ***Consolidating Event Data***.

Consolidated data is written to an NPRC (Natural Profiler resource consolidated) resource file which is in general significantly smaller than the corresponding NPRF resource file. It opens much faster from NaturalONE and provides the same hot spots as the NPRF resource file.

Note: Natural code coverage data written to an NCVF resource file is automatically consolidated by Natural code coverage.

Related Topics

- The Natural Profiler for UNIX and Windows is activated with the `PROFILER` profile parameter described in the *Parameter Reference* documentation.
- The use of the Profiler utility can be controlled by Natural Security, see *Protecting Utilities* in the *Natural Security* documentation.
- The use of the NaturalONE Profiler and NaturalONE code coverage is described in the *NaturalONE* documentation.

76 Using the Profiler Utility

The Natural Profiler is used to monitor the internal process flow of a Natural application and to analyze the performance and the code coverage of the application. Profiling of Natural applications is activated by switching on the `ACTIVE` subparameter of the `PROFILER` profile parameter. The Natural Profiler writes the collected Profiler event data to a Profiler resource file. See *PROFILER – Profile a Natural Session* in the *Parameter Reference* documentation.

Code coverage of Natural applications is activated by switching on the `ACTIVE` subparameter of the `COVERAGE` profile parameter (see the *Parameter Reference* documentation). Natural code coverage writes the collected coverage data to a code coverage resource file.

The Profiler utility runs in batch mode only. It provides functions to control the Profiler and code coverage data collection and to process the resulting data.

1. With the Profiler data collection functions, the data collection can be paused and restarted (see also ***Starting and Pausing Data Collection***).
2. The **data processing** functions read and process the event data from the Profiler resource file. Unconsolidated event data can be consolidated.

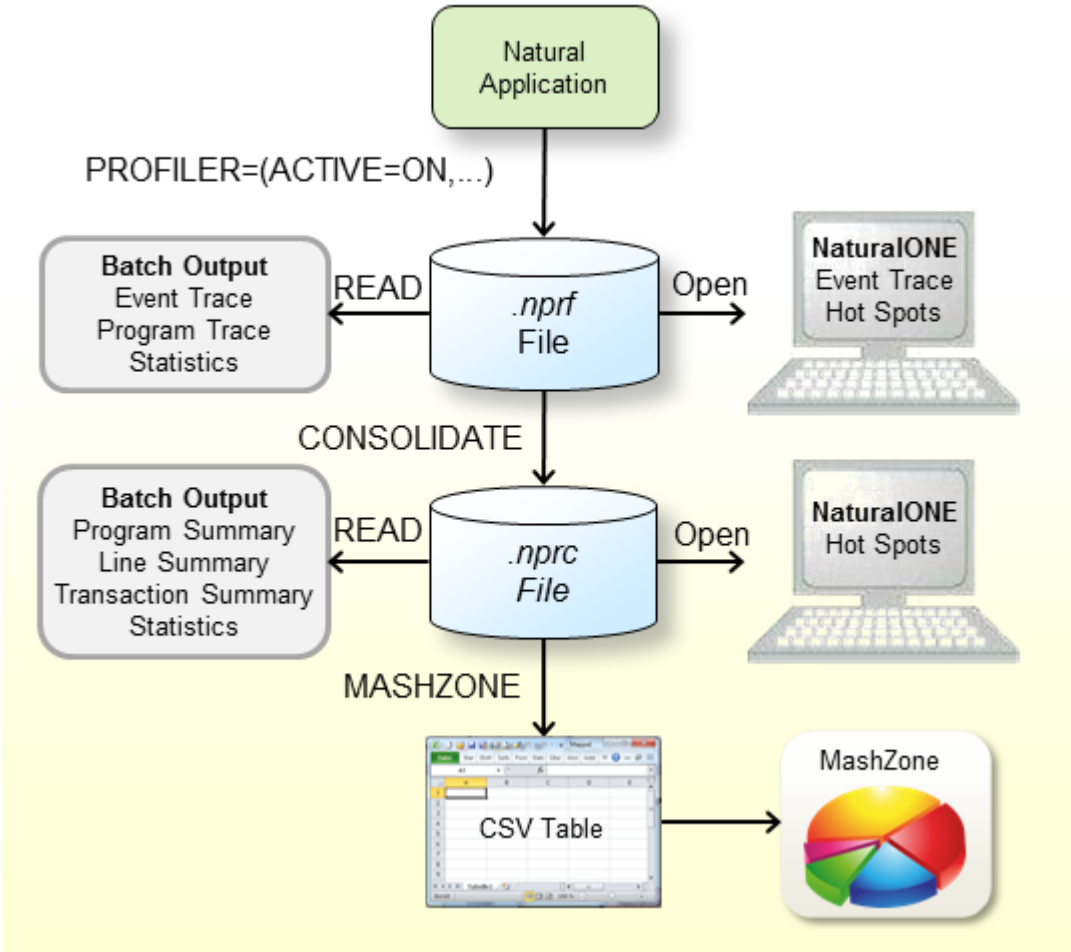
   You can output statistics, a program summary, a line summary and a transaction summary, a program trace, an event trace with the most important data, and reports on program and statement coverage. You can export the resulting data in text or CSV (comma-separated values) format.

The Profiler resource file can be read by NaturalONE which displays the full event trace and provides a performance analysis (hot spots) of the Natural batch application. Coverage data can be inspected in the NaturalONE **Coverage** view and in the NaturalONE source editor. The exported profiling event data can be analyzed with the **Natural Profiler MashApp** which visualizes the data on an interactive MashZone dashboard.

Quick Start for Profiling

This section briefly describes the steps required for profiling Natural applications and viewing the results. The instructions provided here may serve as a guideline when starting to use the Natural Profiler. Detailed information regarding the steps is provided in the remainder of this chapter.

The steps to take depend on the evaluation you want to perform for your application as illustrated in the following graphic:



1. Check that the **prerequisites** are met.
2. Activate the profiling of the Natural session by switching on the `ACTIVE` subparameter of the `PROFILER` profile parameter in the NATPARM parameter file or dynamically when invoking Natural. Example for dynamic parameter specification:

   ```
   PROFILER=(ACTIVE=ON,RESNAME=ResName,RESLIB=RESLIB)
   ```

   In the example above, the Profiler event data is written to a resource file with the name `ResNam.nprf` in the library `RESLIB`. See *PROFILER – Profile a Natural Session* in the *Parameter Reference* documentation.
3. Open the NPRF resource in NaturalONE to view the hot spots and the event trace.
4. Submit a Natural batch job with the Profiler utility `READ` function to print an event trace, a program trace and the Profiler statistics. Example:

```
FUNCTION=READ            /* Read Profiler data
  RESOURCE-LIB=RESLIB    /* Resource library
  RESOURCE-TYPE=NPRF     /* Use resource type NPRF
  EVENT=ON               /* Print event trace
  PROGRAM=ON             /* Print program trace
  STATISTICS=ON          /* Print statistics
```

See also *Profiler Utility READ Function*. This section also describes how to generate a transaction summary.

5. Submit a Natural batch job with the Profiler utility `CONSOLIDATE` function to consolidate (aggregate) the event data. Example:

```
FUNCTION=CONSOLIDATE     /* Consolidate Profiler data
  RESOURCE-LIB=RESLIB    /* Resource library
  REPLACE=YES            /* Replace resource
```

The consolidated Profiler event data is written to the resource `ResNam.nprc` in the library `RESLIB`. See *Consolidating Event Data*.

6. Open the NPRC resource in NaturalONE to view the hot spots.
7. Submit a Natural batch job with the Profiler utility `READ` function to generate a program summary, a line summary and the Profiler statistics. Example:

```
FUNCTION=READ            /* Read Profiler Data
  RESOURCE-LIB=RESLIB    /* Resource library
  RESOURCE-TYPE=NPRC     /* Use resource type NPRC
  PROGRAM=ON             /* Print program summary
  LINE=ON                /* Print line summary
  STATISTICS=ON          /* Print statistics
```

See also *Profiler Utility READ Function*. This section also describes how to generate a transaction summary.

8. Submit a Natural batch job with the Profiler utility `MASHZONE` function to write the data to Work File 7 in the format expected by the Natural Profiler MashApp. Use as Work File 7 a CSV (comma-separated values) file in the Natural Profiler data directory in the MashZone environment. Example:

```
FUNCTION=MASHZONE        /* Write MashZone format to Work File 7
  RESOURCE-LIB=RESLIB    /* Resource library
```

See also *Exporting Event Data for MashZone*.

9. Enter a reference to the new file in the `Overview.csv` file in the `resources\Profiler` directory.

Open the **Natural Profiler MashApp** and select the corresponding input file to evaluate the event data.

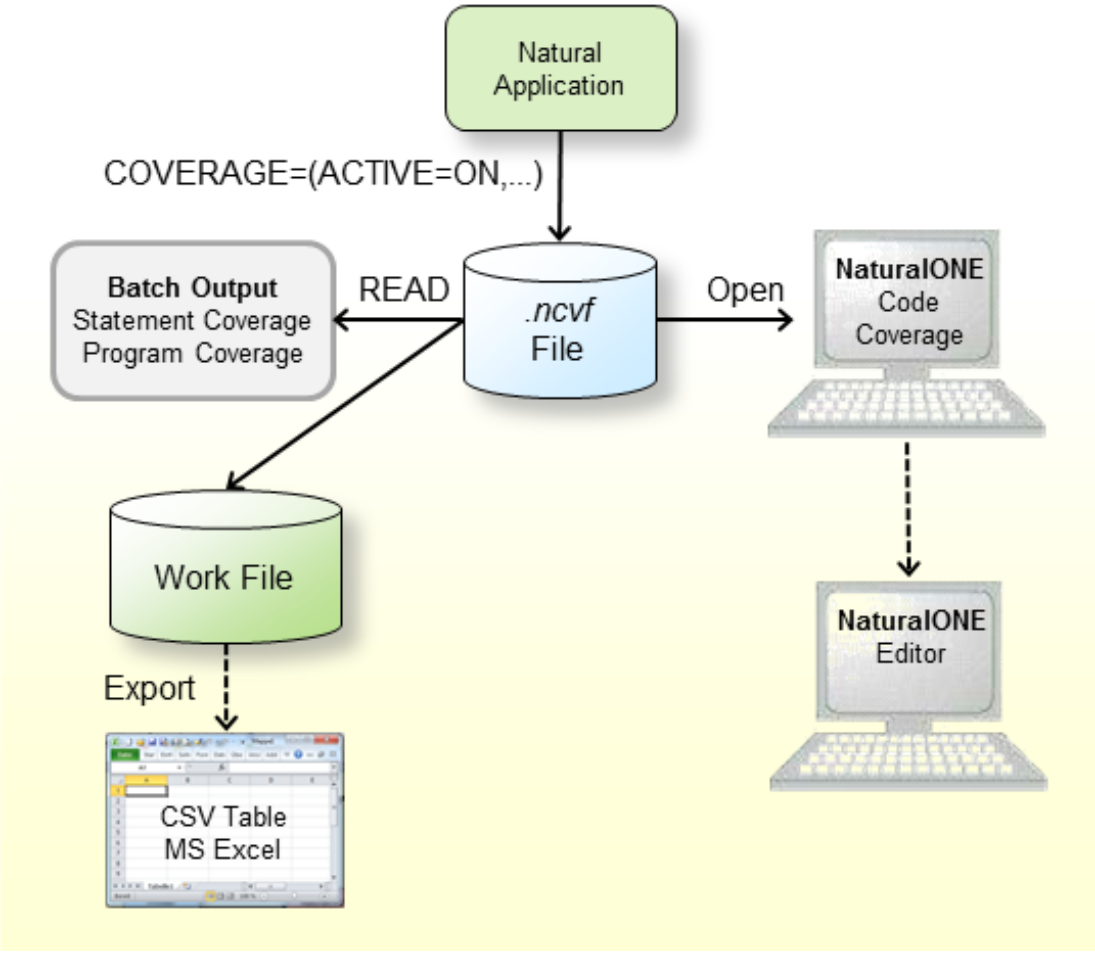
Notes:

1. If the resource name is not explicitly specified in the `READ`, `CONSOLIDATE` or `MASHZONE` function of the Profiler utility, the last created NPRF or NPRC resource in the library is used.
2. If you plan to profile a long-running batch application, refer to the section ***Profiling Long-Running Applications***. It covers strategies of how to minimize the number of events to be monitored.
3. The NaturalONE Profiler is described in the *NaturalONE* documentation.

Quick Start for Code Coverage

This section briefly describes the steps required for performing the code coverage of a Natural batch applications and viewing the results. The instructions provided here may serve as a guideline when starting to use **Natural code coverage**. Detailed information regarding the steps is provided in the remainder of this chapter.

The steps to take depend on the evaluation you want to perform for your application as illustrated in the following graphic:



1. Check that the prerequisites are met.
2. Activate code coverage of the Natural session by switching on the `ACTIVE` subparameter of the `COVERAGE` profile parameter (see the *Parameter Reference* documentation) in the NATPARM parameter file or dynamically when invoking Natural. Example for dynamic parameter specification:

   ```
   COVERAGE=(ACTIVE=ON,RESNAME=ResName,RESLIB=RESLIB)
   ```

   In the example above, the coverage data is written to a resource file with the name `ResNam.ncvf` in the library `RESLIB`.
3. Open the NCVF resource in NaturalONE to obtain the **Code Coverage** view.
4. From the NaturalONE **Code Coverage** view, you can directly edit the source. The editor shows all lines containing covered statements with a green background.
5. Submit a Natural batch job with the Profiler utility `READ` function to print the program and statement coverage.

   Example:

```
FUNCTION=READ            /* Read Profiler data
  RESOURCE-LIB=RESLIB    /* Resource library
  RESOURCE-TYPE=NCVF     /* Use resource type
  EVENT=ON               /* Print statement coverage
  PROGRAM=ON             /* Print program coverage
  EXPORT=ON              /* write to work 7
  FORMAT=C               /* Semicolon/Comma/Text
```

If the EXPORT keyword of the Profiler utility READ function is switched on, the output is written to Work File 7. If FORMAT is specified as C or S, the result is written as comma-separated values (CSV) where a comma or a semicolon is used as a separator, respectively.

6. Export the data of Work File 7 with any tool (such as FTP) as a CSV-formatted file to a Windows environment if you want to process it further in Microsoft Excel.

Notes:

1. If the resource name is not explicitly specified in the READ function of the Profiler utility, the NCVF resource created last in the library is used.
2. The NaturalONE **Code Coverage** view and editor are described in the *NaturalONE* documentation.

Prerequisites

The following prerequisite must be met before you can use the Profiler utility:

- Natural Parameter Settings
- Activate Data Processing

Natural Parameter Settings

The Profiler utility data processing functions (CONSOLIDATE, READ, MASHZONE and LIST) cannot be executed if profiling is active. Deactivate the profiling infrastructure with the following (default) parameter setting:

```
PROFILER=(ACTIVE=OFF)
```

For details regarding the PROFILER parameter, see *PROFILER – Profile a Natural Session* in the *Parameter Reference* documentation.

Activate Data Processing

If NaturalONE is installed at your site, you can activate the Profiler utility data processing functions (`CONSOLIDATE`, `READ`, `MASHZONE` and `LIST`) with the following steps:

1. Start NaturalONE.
2. In the **Natural Server** view, map to the environment where the Profiler resources reside.
3. Add the program `ACTIVATE` contained in the system library `SYSPRFLR` to a new or existing project in NaturalONE.
4. Profile the program `ACTIVATE` with the context menu function **Profile As** > **Natural Application**.
5. Verify that the user-defined event data on the **Event Trace** page of the NaturalONE Profiler contains the activation success message.

When the program `ACTIVATE` is profiled, a NaturalONE Profiler key is generated and written to the Natural resource `NaturalONEProfilerKey.nprk` in the system library `SYSPRFLR`. Each Profiler data processing function reads this resource and checks the key. If the key is valid, the function is performed. A newly generated key is valid for one year. It can always be regenerated.

The Profiler data processing function starts issuing a warning 9 days before the key expires, and returns an error message if no key is found or if the key is not valid.

Notes:

1. Natural statement events (`NS`) are only generated for profiling if the corresponding Natural object was compiled with the profile parameter `GPGEN` set to `(PROFILER=ON)`.`GPGEN` is described in the *Parameter Reference* documentation.
2. Code coverage of a Natural application can only be performed if the corresponding Natural objects were compiled with the profile parameter `GPGEN` set to `(COVERAGE=ON)`.

Invoking and Terminating the Profiler Utility

This section provides instructions for invoking and terminating the Profiler utility in batch mode.

To invoke the Profiler utility

- Enter the following system command into the primary command input data set `CMSYNIN`:

```
PROFILER
```

Note: After the `PROFILER` system command, the Profiler expects one or more lines with Profiler keyword entries.

To terminate the Profiler utility

- Enter the following Profiler keyword into the primary command input data set `CMSYNIN`:

```
END-PROFILER
```

Or:

```
END
```

Or:

```
.
```

Syntax and Keywords

The Profiler utility in batch mode reads the Profiler keywords that control the profiling from the primary command input data set `CMSYNIN`. The Profiler reads the input lines until it reaches the `END-PROFILER` keyword (or `END` or `.`).

This section covers the following topics:

Profiler Utility Syntax

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

You enter a Profiler utility command using either of the following syntax formats:

```
keyword[=value][,keyword[=value]]...
```

Or:

```
keyword
[value]
...
```

Notes:

1. If a value is associated with a keyword but no equal sign is found, the Profiler expects the value in a separate input line without any other keyword (second syntax format).
2. The first syntax format expects input in delimiter mode (`IM=D`).
3. The second syntax format can be used if the Profiler is to be executed with the Natural `STACK` profile parameter or if the data is entered in forms mode (`IM=F`).

The following rules apply:

- Empty lines and lines starting with an asterisk (`*`) are ignored.
- All characters in a line from `/*` to `*/` or to the end of the line are ignored.
- Some keywords have no associated value.
- Blanks can be added before or after the keyword or value.
- Multiple keywords in a line are separated by commas (applies to the first syntax format only).
- A value can be enclosed in apostrophes (`'value'`).
- A value must not contain a comma.
- Keywords and values can be specified in upper or lower case.
- The maximum input line length is 78 characters.

The Profiler utility can be executed multiple times in one Natural session. For example, it is first executed with the `START` function, and then, after the execution of a user program, it is executed with the `PAUSE` function.

Example

The following Natural batch example runs the Profiler utility `READ` function:

```
natural BATCHMODE CMSYNIN=cmd.txt CMOBJIN=data.txt CMPRINT=out.txt CMWRK07=wrk07.txt
```

The content of the batch input file `cmd.txt` which contains the `PROFILER` command is shown below:

```
PROFILER
FIN
```

The content of the input file `data.txt` which contains the input for the Profiler utility is shown below:

```
************************************************************************
* Read Profiler NPRF Resource
************************************************************************
TRACE=3                  /* Set Profiler trace level
FUNCTION=READ            /* Read Profiler resource
  RESOURCE-NAME='Demo01' /* Resource name
  RESOURCE-LIB=PRFDATA   /* Resource library
  RESOURCE-TYPE=NPRF     /* Resource type
  EVENT=ON               /* List events
  STATISTICS=ON          /* List properties and statistics
  PROGRAM=ON             /* Program trace
  PRINT=ON               /* Write to standard output
  EXPORT=ON              /* Write to work file 7
  FORMAT=TEXT            /* Use text format
END-PROFILER             /* End profiler input
```

After execution, `out.txt` contains the Profiler utility output and the internal trace (`TRACE=3`). `wrk07.txt` contains the Profiler utility output in text format (`FORMAT=TEXT`).

The following Natural example demonstrates how the Profiler utility `PAUSE` function is to be executed with the Natural `STACK` profile parameter:

```
natural PROFILER=(ACTIVE=ON,RESNAME=ResName,RESLIB=RESLIB)
STACK=(
PROFILER FUNCTION:PAUSE:END-PROFILER;
LOGON PRFDEMO
)
```

After execution, profiling of the Natural session is activated but the data collection is paused. The data collection can be started later with the Profiler utility `START` function.

Profiler Utility Keywords

The main keywords used in the syntax of the Profiler utility in batch mode are described in the following table. Any additional (subordinate) keywords available for a main keyword are described in the sections referenced in the table. In general, a subordinate keyword value must follow the main keyword value, for example:

```
FUNCTION=READ
  PRINT=ON
```

A subordinate keyword specified before the first `FUNCTION` or `FILTER` keyword is treated as a subordinate keyword of the first `FUNCTION` or `FILTER` keyword.

The following main keywords are available:

Keyword	Value	Description
`FUNCTION`		Perform a Profiler utility function.
	`CONSOLIDATE`	Consolidate (aggregate) resource data. See ***Consolidating Event Data***.
	`LIST`	List Profiler resources. See ***Listing Profiler Resource Files*** in *Maintaining Profiler Resource Files*.
	`MASHZONE`	Export resource data in MashZone format. See ***Exporting Event Data for MashZone***.
	`PAUSE`	Pause the data collection. See ***Starting and Pausing Data Collection***.
	`READ`	Read and evaluate resource data. See ***Evaluating Event Data***.
	`START`	Start or restart the data collection. See ***Starting and Pausing Data Collection***.
`TRACE`	`0 - 10`	Set the level of internal trace of the Profiler trace session. The internal trace contains information such as Profiler errors and is written to the standard output of the trace session (`CMPRINT` data set). See ***Internal Trace***. Default: `2` (warning)
`HELP`		A summarized description of the Profiler keywords is written to standard output.
`INCLUDE`	*`object-name`*	The name of the Natural text object that contains Profiler input data. See also ***Including Profiler Input from Natural Text Objects***.
`INCLUDE-LIB`	*`library-name`*	The name of the Natural library that contains the text object specified with the `INCLUDE` keyword. If the Natural system variable `*LIBRARY-ID` is specified, the name of the current library is used. The library name is used for all following `INCLUDE` keywords. Default: If `INCLUDE-LIB` is not specified before an `INCLUDE` keyword, the Natural system library `SYSPRFLR` is used by default.

Keyword	Value	Description
		See also *Including Profiler Input from Natural Text Objects*.
END-PROFILER or END or .		End of Profiler input. The keyword END-PROFILER, END or a period (.) indicates the end of the Profiler input.

Events and Data Collected

This section describes the events and data processed by the Profiler utility.

- Events
- Data Collected

Events

During a Natural session, different types of events can occur (for example, a program start) where the Profiler collects data specific to the event in a trace record. Each event is associated with an event type, that is, a one or two letter code. Related event types are combined into an event group which is denoted by a one letter code.

The following events, event types and event groups are available:

Event	Event Type	Event Group	When the Event Occurs
Session **I**nitialization	SI	S	When a Natural batch session is initialized. Because the Profiler monitor session starts after the trace session, this event cannot be monitored.
Session **T**ermination	ST	S	When a Natural batch session is terminated. The Profiler always monitors this event.
Program **L**oad	PL	P	When a program (Natural object) is loaded or when it is already located in the buffer pool.
Program **S**tart	PS	P	When a program (Natural object) is started.
Program **T**ermination	PT	P	When a program (Natural object) is terminated.
Program **R**esume	PR	P	When a program (Natural object) resumes control after another Natural object has been executed or when control returns to level 0 (no program active).

Event	Event Type	Event Group	When the Event Occurs
Program **I**nformation	PI	P	When a program (Natural object) is accessed for the first time. This event is only triggered at Natural code coverage.
Before **D**atabase Call	DB	D	Before a database call is executed.
After **D**atabase Call	DA	D	After a database call has been executed.
Before Terminal **I**/O	IB	I	Before a terminal input/output is executed.
After Terminal **I**/O	IA	I	After a terminal input/output has been executed.
Before External Program **C**all	CB	C	Before an external program call (CALL statement) is executed.
After External Program **C**all	CA	C	After an external program call (CALL statement) has been executed.
Runtime **E**rror	E	E	When a Natural runtime error has occurred.
Natural **S**tatement	NS	N	When a Natural statement is executed. For technical reasons, there is no one-to-one relationship between a Natural source code statement and the corresponding object code in the cataloged object. Therefore, multiple Natural statements can be merged into one NS event and conversely, one Natural statement can cover multiple NS events.
Inbound **R**PC Message	RI	R	When the Natural RPC server layer receives the client request.
Start of **R**PC Request Execution	RS	R	When the Natural RPC server layer calls the Natural server program.
Outbound **R**PC Message	RO	R	When the Natural RPC server returns the result to the client.
RPC **W**ait for Client	RW	R	When the Natural RPC server waits for the next message from the client.
User-Defined **E**vent	U	U	When a user-defined event was generated.
Monitor **P**ause	MP	M	When the data collection is paused. A pause event can be caused by an explicit pause request, at the start of a block filter or when the data pool is full. The duration of a pause is not considered for the application performance analysis.

With each collected event, a CPU and an event timestamp are recorded. In general, a timestamp is taken at the beginning of an event. The duration of an event therefore equals the time that elapses between the timestamp of the event and the timestamp of the event that follows.

Data Collected

This section describes the data collected by the Natural Profiler:

General Data

The following data elements are collected at every event:

- Event counter
- Event type
- Event time in units of microseconds
- Session CPU time in units of microseconds
- Trace session ID
- Natural Security user group ID
- Natural user ID
- Natural application name
- Program library
- Program name
- Program level
- Copycode library
- Copycode name
- Statement line number
- Statement op-code
- Coverage flag (for Natural code coverage)

Notes:

1. The Natural Profiler for UNIX and Windows does not yet collect RPC-related events.
2. Natural code coverage only collects `NS` and `PI` events.
3. Natural code coverage does not collect time values.
4. A `PI` event is collected for each object accessed and for all copycodes included in the object (recursively).

Event-Specific Data

The following data is only collected at the following events:

Event	Data Elements
Session Initialization	None
Session Termination	Termination return code Natural termination message code NAT99*nn* Name of back-end program Monitor CPU time in units of microseconds
Program Load	Name of program to be loaded Name of load library Invocation type
Program Resume	None
Program Start/Termination	Program type Database ID of program library File number of program library
Program Information	Program type Number of statements in the program or copycode First statement item `INCLUDE` line number Parent copycode ID
Database Call	Database type Command code Command ID Database ID File number Response code (event type `DA`) Error subcode (event type `DA`) Adabas command time (event type `DA`)
Terminal I/O	Number of bytes sent Number of bytes read Total session storage allocated Compressed session storage length
External Program Call	Name of program called Calling mode such as dynamic or static mode Program link location Parameter type such as reference or value Response code (event type `CA`)
Runtime Error	Natural system error message code External abend code Name of error handling program
Natural Statement	Profiling: None Natural code coverage: Statement item identifier (GP offset)
Start of RPC Request Execution	Environment (`C` = client, `S` = server) Subprogram name Adabas user ID (ETID) Conversation status

Event	Data Elements	
	Logon indicator (Y = logon performed) Impersonation indicator of RPC request (Y = impersonation performed)	
Outbound/Inbound RPC Message / RPC Wait for Client	Environment (C = client, S = server) Transport protocol RPC function Type of client user ID Length of message RPC return code External conversation ID Client user ID Server node (event types RO and RW) Server name (event types RO and RW)	
User-Defined Event	Subtype of the user-defined event Up to 249 bytes of user-defined information	
Monitor Pause	Type of monitor pause Possible values:	
	R	Monitor pause requested. This value is also set when the session is initialized with the Pause option.
	F	Start of a block of filtered-out events. Block filters are: library, program, line, FNAT, event count, or time filter.
	W	Trace session waits because of a data pool full situation.

Starting and Pausing Data Collection

When a Natural session is profiled, all event data of the session is collected by default.

You can start and pause data collection with the following methods:

- Using Profiler Utility Functions
- Using Profiler Utility Programs

- Using the Application Programming Interface

Using Profiler Utility Functions

The Profiler utility `START` and `PAUSE` functions are used to start and pause data collection. The following syntax applies:

```
FUNCTION=START [COUNT={0|count-number}]
FUNCTION=PAUSE
```

Syntax Description:

Keyword for START	Value	Description
`COUNT`	*`count-number`*	Set the event counter of the next monitored event to the specified value. Valid values for *`count-number`*: `0` to `2147483647` The event counter remains unchanged if a value of zero (`0`) is specified.

Using Profiler Utility Programs

The following Natural programs in the system library `SYSPRFLR` are supplied to perform Profiler utility functions:

Program	Description
PRFSTART	Start the data collection.
PRFPAUSE	Pause the data collection.
PRFSTATE	Get the state of the data collection.
PRFFCT	Execute a Profiler utility function: `START`, `PAUSE` or `STATE`.

To use Profiler utility programs

- Logon to the library `SYSPRFLR` or copy the programs to the library `SYSTEM`, to the appropriate steplib library, or to the required library.

  If `PRFFCT` is used, the application programming interface `USR8210N` has to be copied as well (see the following section).

  If `PRFFCT` is used in a client/server environment, copy `PRFFCT` to the client library and `USR8210N` to the server library.

  Note: `PRFFCT` expects as input the value `START`, `PAUSE` or `STATE` to perform the corresponding function.

To start the data collection

- Execute the following program:

```
PRFSTART
```

Or:

```
PRFFCT
START
```

To pause the data collection

- Execute the following program:

```
PRFPAUSE
```

Or:

```
PRFFCT
PAUSE
```

To retrieve the current state of the data collection

- Execute the following program:

```
PRFSTATE
```

Or:

```
PRFFCT
STATE
```

Using the Application Programming Interface

The data collection can be started and paused from the profiled Natural application by calling the application programming interface (API) `USR8210N`. The API can also be used to get the current state of the monitoring process. The API is delivered in the `SYSEXT` library. For more information, see *SYSEXT Utility - Natural Application Programming Interfaces*.

To use the API

- Copy the subprogram USR8210N to the library SYSTEM, to the appropriate steplib library, or to the required library.

  Note: USR8210N expects as the first parameter the value START, PAUSE or STATE to perform the corresponding function. The parameter values can be specified in uppercase or lowercase. On return, P-RETURN contains the return code and P-MESSAGE the success or error message.

To start the data collection

- Use the interface with the CALLNAT statement:

```
CALLNAT 'USR8210N' 'START' P-RETURN P-MESSAGE /* Start Profiler
```

To pause the data collection

- Use the interface with the CALLNAT statement:

```
CALLNAT 'USR8210N' 'PAUSE' P-RETURN P-MESSAGE /* Pause Profiler
```

To retrieve the current state of the data collection

- Use the interface with the CALLNAT statement:

```
CALLNAT 'USR8210N' 'STATE' P-RETURN P-MESSAGE /* Get Profiler state
```

The state is coded in the field P-RETURN:

P-RETURN	Description
0	Natural Profiler data collection is started.
1	Natural Profiler data collection is paused.

Consolidating Event Data

The Profiler utility `CONSOLIDATE` function consolidates event data.

For general information regarding data consolidation, see ***Data Consolidation*** in the section *Basic Concepts of the Profiler Utility*.

Syntax of `CONSOLIDATE`:

```
FUNCTION=CONSOLIDATE
 [RESOURCE={ON|OFF}]
 [RESOURCE-NAME=resource-name]
 [RESOURCE-LIB=library-name]
 [REPLACE={YES|NO}]
 [TRANSACTION={ON|OFF}]
 [IO-TIME={ON|OFF}]
 [EXPORT={ON|OFF}]
 [FORMAT={TEXT|COMMA|SEMICOLON}]
 [TRACE-EVENT={ON|OFF}]
 [TRACE-CONSOLIDATE={ON|OFF}]
```

Syntax Description:

Keyword for CONSOLIDATE	Value	Description
RESOURCE		Specifies whether the consolidated event data is written to a Natural Profiler resource consolidated (NPRC) resource file.
	ON	The consolidated event data is written to an NPRC resource file.
	OFF	The consolidated event data is not written to an NPRC resource file. This setting is useful if you want to print the event trace or statistics or export the data and you do not need the consolidated NPRC resource file.
RESOURCE-NAME	*resource-name*	The name of the Natural Profiler resource file (NPRF) you want to consolidate. The file extension `.nprf` is added automatically. Default: The name of the last created NPRF resource file in the library

Keyword for CONSOLIDATE	Value	Description
		If RESOURCE=ON, the consolidated data is written to an NPRC resource file with the same resource name.
RESOURCE-LIB	*library-name*	The name of the Natural library that contains the NPRF resource file you want to consolidate. Default: The name of the current library. This library is also used as the target library for the consolidated NPRC resource file.
REPLACE		Specifies whether an existing NPRC resource file is replaced.
	YES	Replace an existing NPRC resource file with the same name.
	NO	Do not replace an existing NPRC resource file with the same name. A message is returned if a resource file with the same name already exists. No consolidation is performed in this case.
TRANSACTION		Specifies whether transaction identifiers are added to the consolidated event data.
	ON	Transaction identifiers are added to the consolidated event data for transaction evaluation purposes. For more information, see ***Transaction Summary***. **Note:** The generated NPRC resource file requires more space if ON is set.
	OFF	Transaction identifiers are not added to the consolidated event data.
IO-TIME		Specifies whether I/O time (IB event) and Natural RPC client time (RW event) are included in the consolidated data.
	ON	I/O and Natural RPC client time are included in the consolidated data.
	OFF	I/O and Natural RPC client time are not included in the consolidated data.
EXPORT		Specifies whether the consolidated event data is written to Work File 7.
	ON	Write to Work File 7.
	OFF	Do not write to Work File 7.
FORMAT		Specifies the format in which the exported data is written to Work File 7.
	TEXT	Write the data in free text format.
	COMMA	Write the data in CSV format with a comma (,) separator.
	SEMICOLON	Write the data in CSV format with a semicolon (;) separator.

Keyword for CONSOLIDATE	Value	Description
TRACE-EVENT		Specifies whether the Profiler event trace is written to standard output. See *Event Trace*.
	ON	Write the Profiler event trace.
	OFF	Do not write the Profiler event trace.
TRACE-CONSOLIDATE		Specifies whether the Profiler consolidation trace is written to standard output. See *Consolidation Trace*.
	ON	Write the Profiler consolidation trace.
	OFF	Do not write the Profiler consolidation trace.

Example of a Consolidation

The following example consolidates the Profiler resource `Test.nprf` in the library `PRFDATA` and writes the consolidated data to the Profiler resource `Test.nprc`. I/O and Natural RPC client time are included in the consolidated data.

In addition, the consolidated data is written in CSV (semicolon-separated values) format to Work File 7.

The event and consolidation traces are switched off.

```
FUNCTION=CONSOLIDATE    /* Consolidate Profiler data
  RESOURCE=ON           /* Write to resource
  RESOURCE-NAME='Test'  /* Resource name
  RESOURCE-LIB=PRFDATA  /* Resource library
  REPLACE=YES           /* Replace resource
  TRANSACTION=OFF       /* Do not add transaction identifiers
  IO-TIME=ON            /* Include I/O and RPC client times
  EXPORT=ON             /* Write to Work File 7
  FORMAT=S              /* CSV format with semicolon separator
  TRACE-EVENT=OFF       /* No event trace
  TRACE-CONSOLIDATE=OFF /* No consolidation trace
```

Evaluating Event Data

When a Natural application is profiled, the Natural Profiler utility writes the event data to an NPRF resource file. Consolidated data is stored in an NPRC resource file and coverage data is stored in an NCVF resource file. The Profiler utility `READ` function reads and evaluates the Profiler resource data and writes the results to standard output or to a Natural work file. The evaluations performed depend on the type of the resource file read as described in the following table:

Resource File Type	Evaluation	Description
NPRF	Event trace	Chronological list of the Profiler event data
	Program trace	Program flow of the profiled application
	Statistics	Statistics of profiling and the NPRF resource file
NPRC	Consolidation trace	List of the consolidated data with hit counts and summarized elapsed time and CPU time
	Program summary	Table of executed Natural objects The table shows which events occurred during object execution and the CPU time spent executing the object.
	Line summary	Table of executed Natural source lines The table shows how many events occurred during line execution and the CPU and elapsed time spent executing the line.
	Transaction summary	Table of executed transactions. The table shows which events occurred during transaction execution and the response time (elapsed time) of the transactions.
	Statistics	Statistics of profiling, the consolidation and the NPRC resource file
NCVF	Statement coverage	List of statements covered in in the source lines The list shows the percentage of statement coverage for each statement line in the source of the accessed programs.
	Program coverage	Table of code coverage results of executed Natural objects The program coverage table lists all Natural objects which have been executed during the coverage run. For each object, it shows the percentage of coverage, the number of covered and missed statements, and the total number of statements.
	Statistics	Statistics for profiling, coverage and the NCVF resource file

This section covers the following topics:

- Statistics

Profiler Utility READ Function

The Profiler utility `READ` function reads and evaluates the resource data.

Syntax of `READ`:

```
FUNCTION=READ
 [RESOURCE-NAME=resource-name]
 [RESOURCE-LIB=library-name]
 [RESOURCE-TYPE={NPRF|NPRC|NCVF}]
 [EVENT={ON|OFF}]
 [PROGRAM={ON|OFF}]
 [LINE={ON|OFF}]
 [TRANSACTION={ON|OFF}]
 [STATISTICS={ON|OFF}]
 [PRINT={ON|OFF}]
 [EXPORT={ON|OFF}]
 [FORMAT={TEXT|COMMA|SEMICOLON}]
```

Syntax Description:

Keyword for READ	Value	Description
`RESOURCE-NAME`	*resource-name*	The name of the NPRF, NPRC or NCVF resource file you want to read. If no file extension is specified, the extension specified with the keyword `RESOURCE-TYPE` is added automatically. Default: The name of the last created NPRF, NPRC or NCVF resource file in the library depending on the `RESOURCE-TYPE` specification
`RESOURCE-LIB`	*library-name*	The name of the Natural library that contains the NPRF, NPRC or NCVF resource you want to read. Default: The name of the current library
`RESOURCE-TYPE`		Specifies the default resource type (extension) to use if no extension is specified with `RESOURCE-NAME`.
	`NPRF`	The default resource type is NPRF with extension `.nprf`.
	`NPRC`	The default resource type is NPRC with extension `.nprc`.
	`NCVF`	The default resource type is NCVF with extension `.ncvf`.
`EVENT`		Specifies whether the Natural Profiler evaluates events. See also *Event Trace*, *Consolidation Trace* and *Statement Coverage*.

Keyword for READ	Value	Description
	ON	NPRF: Write the Natural Profiler event trace. NPRC: Write the Natural Profiler consolidation trace. NCVF: Write the statement coverage result.
	OFF	Do not evaluate events.
PROGRAM		Specifies whether the Natural Profiler evaluates programs. See also *Program Trace*, *Program Summary* and *Program Coverage*.
	ON	NPRF: Write the Natural Profiler program trace. NPRC: Write the Natural Profiler program summary. NCVF: Write the program coverage table.
	OFF	Do not evaluate programs.
LINE		This option is only available for NPRC resources. Specifies whether the Natural Profiler evaluates executed source lines. See also *Line Summary*.
	ON	Write the Natural Profiler line summary.
	OFF	Do not evaluate executed source lines.
TRANSACTION		This option is only available for NPRC resources. Specifies whether the Natural Profiler evaluates transactions. For more information, see *Transaction Summary*.
	ON	Evaluate transactions to generate a transaction summary and show transaction-related values in the **program summary** and the **line summary**.
	OFF	Do not evaluate transactions.
STATISTICS		Specifies whether the Natural Profiler writes statistics. See also *Profiler Statistics*. **Note:** Statistics data for Natural code coverage is not collected on UNIX and Windows.
	ON	Write statistics.
	OFF	Do not write statistics.
PRINT		Specifies whether the result is written to standard output.
	ON	Write to standard output.
	OFF	Do not write to standard output.
EXPORT		Specifies whether the evaluated data is written to the Work File 7.
	ON	Write to Work File 7.
	OFF	Do not write to Work File 7.
FORMAT		Specifies the format in which the exported data is written to Work File 7.

Keyword for READ	Value	Description
	TEXT	Write the data in free text format.
	COMMA	Write the data in CSV format with a comma (,) separator.
	SEMICOLON	Write the data in CSV format with a semicolon (;) separator.

Example of READ

The following example reads the Natural Profiler resource `Test.nprf` in the library `PRFDATA` and writes the event trace, program trace and the Profiler statistics to standard output and to Work File 7 in text format.

```
FUNCTION=READ             /* Read Profiler Data
  RESOURCE-NAME='Test'    /* Resource name
  RESOURCE-LIB=PRFDATA    /* Resource library
  RESOURCE-TYPE=NPRF      /* Use resource type NPRF
  EVENT=ON                /* Print event trace
  PROGRAM=ON              /* Print program trace
  STATISTICS=ON           /* Print statistics
  PRINT=ON                /* Write to standard output
  EXPORT=ON               /* Write to Work File 7
  FORMAT=TEXT             /* Export in text format
```

Event Trace

If `EVENT=ON` is specified for an NPRF resource file, the Profiler event trace is generated.

The event trace shows the data of each Natural event which occurred while the application executed. The trace can be referenced if detailed information of an event is required. For example, if a Natural error occurred during application execution, the event trace shows the corresponding error number and message.

If the event trace is written to standard output (`PRINT=ON`) or exported in text format (`EXPORT=ON`, `FORMAT=TEXT`), it is similar to the event trace written by the Profiler monitor session while the application was profiled (see *Event Trace*). If the data is exported in CSV (comma-separated values) format, it contains all data fields provided by the Profiler (see *Data Collected*).

Example of an Event Trace

The following example shows an extract of an event trace:

```
Natural Profiler Event Trace
----------------------------
     Count Time             CPU-Time (ms) Ev Lev Library  Program  Line CC-Lib   CC-Name  Statement  Local-Data
         0 10:20:58.219911        63.318   MP 003 SYSPRFD  PRBINIT  8350                  Call       Monitor pause requested
       102 10:20:58.277586        76.106   PL 000                   0000                             Execute SYSEDMD/MENU
       103 10:20:58.277591        76.139   PS 001 SYSEDMD  MENU     0000                  PgmStart   00010/02430 Type: P
       103 10:20:58.277594        76.151   NS 001 SYSEDMD  MENU     0250                  Compute    Assign/Compute/Move
       103 10:20:58.277596        76.155   NS 001 SYSEDMD  MENU     0270                  Fetch      Fetch
       104 10:20:58.277598        76.169   DB 001 SYSEDMD  MENU     0270                  Fetch      00010/02430 S1
...
```

Explanations:

- The **Count** column shows the number of the event.
- The **Time** and **CPU-Time** columns show the event time and the CPU timestamp of the event execution, respectively.
- The event with the number `104` is a Database Before (`DB`) event caused by an Adabas S1 command issued against the file `00010/02430` which was triggered by a `FETCH` statement in the line `0270` of the Natural object `MENU`.

For further explanations of the trace columns and event types, see the sections ***Event Trace*** and ***Events and Data Collected***.

Consolidation Trace

If `EVENT=ON` is specified for an NPRC resource file, the Natural Profiler consolidation trace is generated. The consolidation trace is also generated if `TRACE-CONSOLIDATE=ON` is set for the Profiler utility `CONSOLIDATE` function.

The consolidation trace shows general event data, summarized values of the elapsed time and CPU time and the hit count of the consolidated record. If two trace entries show the same general event data, they have different event-specific data which is not displayed in the consolidation trace.

The consolidated records are used as the basis for further evaluations like the NaturalONE hot spots or the **Natural Profiler MashApp**. The consolidation trace can be used to validate the consolidated data.

If the consolidation trace is written to standard output (`PRINT=ON`), it is similar to the consolidation trace written by the Profiler data consolidation (see ***Consolidating Event Data***). If the data is exported, it contains all consolidated data fields provided by the Profiler.

Example of a Consolidation Trace

The following example shows an extract of a consolidation trace:

```
Natural Profiler Consolidation Trace
------------------------------------
    Count Transact Ev User     Lev Library  Program  Line CC-Lib   CC-Name  Statement   Hit-Count  Elapsed(ms)      CPU(ms)
        1          DA PRF082D  000                   0000                                      1       75.692         0.870
        2          DA PRF082D  000                   0000                                      1        0.002         0.004
        3          DA PRF082D  000                   0000                                      1        0.006         0.025
        4          NS PRF082D  006 SYSLIBS  A82CLS   0010 SYSAOSSU C-COPYRT Reset            43        0.043         0.118
        5          NS PRF082D  006 SYSTEM   NOMSTCS  4360                   End                1        0.000         0.003
        6          PL PRF082D  006 SYSTEM   NOMSTCS  0970                   Callnat            1        0.008         0.058
        7          PL PRF082D  006 SYSTEM   NOMSTCS  1020                   Perform            1        0.004         0.017
...
```

Explanations:

- The **Count** column shows the number of the consolidated record.
- The **Transact** column shows the transaction identifier.

  The transaction identifier starts with `1` and is increased with every `IA` (after terminal I/O) or `RI` (inbound RPC message) event. Transaction identifiers are only available if data is consolidated with the option `TRANSACTION=ON`.
- The consolidated record `4` shows that the `RESET` statement in the line `0010` of the copycode `C-COPYRT` (included in the Natural object `A82CLS`) executed `43` times spending a total elapsed time of `0.043` milliseconds (`ms`) and a total CPU time of `0.118` ms.

For further explanations of the trace columns and event types, see the sections ***Event Trace*** and ***Events and Data Collected***.

Program Trace

If `PROGRAM=ON` is specified for an NPRF resource file, the Profiler program trace is generated. The program trace shows the program flow of the profiled application. In general, the program trace exclusively shows program and session events (see ***Events and Data Collected*** for a list of possible event types).

If the program trace is written to standard output (`PRINT=ON`) or exported in text format (`EXPORT=ON`, `FORMAT=TEXT`), the program names are indented (see the example below) according to the program level to provide a quick overview of the application calling structure.

If the data is exported in CSV (comma-separated values) format, the program names are not intended. In addition to the output in text format, the exported data contains the CPU timestamp and the summarized Adabas time.

Example of a Program Trace

The following example shows an extract of a program trace and the totals of the application run:

```
Natural Profiler Program Trace
------------------------------
Time            Ev Library  CC-Name  Line Lev Program  Events
10:20:58.309812 PL                   0000 000
10:20:58.309817 PS SYSEDMD           0000 001 .OPTTEST  D=4 N=2
10:20:58.357694 PL SYSEDMD           5620 001 .OPTTEST
10:20:58.357704 PS SYSEDMD           0000 002 ..CALLMON3 N=3
10:20:58.385263 PL SYSEDMD           0980 002 ..CALLMON3
10:20:58.385274 PS SYSEDMD           0000 003 ...OP3DISC  D=3 N=4
10:20:58.412207 PL SYSEDMD           1670 003 ...OP3DISC
10:20:58.412221 PS SYSEDMD           0000 004 ....OPTINFO  N=57
10:20:58.443203 PL SYSEDMD           5830 004 ....OPTINFO
10:20:58.443210 PS SYSEDMD           0000 005 ....:OPTPARM1 D=3 N=19
10:20:58.449549 PL SYSEDMD           1960 005 ....:OPTPARM1
10:20:58.449555 PS SYSEDMD           0000 006 ....:.OPTPARM2 D=3 N=10
10:20:58.458286 PL SYSEDMD           0560 006 ....:.OPTPARM2
10:20:58.458300 PS SYSEDMD           0000 007 ....:..OPTPARM3 N=16
10:20:58.458390 PL SYSEDMD           1530 007 ....:..OPTPARM3
10:20:58.458408 PS SYSLIBS           0000 008 ....:...NAT41004 D=10 C=6 N=7345
10:20:58.471017 PT SYSLIBS           5235 008 ....:...NAT41004
10:20:58.471017 PR SYSEDMD           1530 007 ....:..OPTPARM3 N=2898
10:20:58.473293 PL SYSEDMD           1530 007 ....:..OPTPARM3
10:20:58.473297 PS SYSLIBS           0000 008 ....:...NAT41004 D=5 C=6 N=1416
10:20:58.475581 PT SYSLIBS           5235 008 ....:...NAT41004
10:20:58.475581 PR SYSEDMD           1530 007 ....:..OPTPARM3 N=466
10:20:58.475957 PT SYSEDMD           2190 007 ....:..OPTPARM3
10:20:58.475957 PR SYSEDMD           0560 006 ....:.OPTPARM2 N=283
10:20:58.476187 PT SYSEDMD           0860 006 ....:.OPTPARM2
10:20:58.476187 PR SYSEDMD           1960 005 ....:OPTPARM1 N=42
10:20:58.476222 PT SYSEDMD           7510 005 ....:OPTPARM1
10:20:58.476222 PR SYSEDMD           5830 004 ....OPTINFO  D=3 N=10
10:20:58.497926 PL SYSEDMD           6080 004 ....OPTINFO
10:20:58.521954 PR SYSEDMD           1670 003 ...OP3DISC  N=241
10:21:11.205102 PR SYSEDMD           0980 002 ..CALLMON3 D=7 N=6070
10:21:41.704996 PR SYSEDMD           5620 001 .OPTTEST  D=8 I=3 N=26
10:21:41.731229 PT SYSEDMD           7370 001 .OPTTEST
10:21:41.731229 PR                   0000 000          D=14 I=1
10:21:42.248348 ST                   0000 000

Totals
------
Ev Event                    Count
S  Session ................ 1
P  Program ................ 5297
D  Database Call .......... 2140
I  Terminal I/O ........... 12
C  External Program Call .. 6510
E  Runtime Error .......... 43
N  Natural Statement ...... 857384
R  RPC Request............. 0
U  User-Defined Event ..... 0
M  Monitor Pause .......... 2
```

Explanations:

- For each event listed, the time when the event occurred, the active library, program (Natural object), copycode, line number and program level is displayed.

- The program name is followed by the number of events that occurred from one program event to the next program event.
- Events which belong to one event group are combined into one count using the maximum count of the corresponding event types. Example: One Database Before (`DB`) and one Database After (`DA`) event are combined into one Database event (`D=1`).
- In the example above, the Natural object `OPTTEST` was started at the level `1`. This program calls the subprogram `CALLMON3` which calls further subprograms. The highest Level 8 is reached when the subprogram `NAT41004` executes. During the first execution, this subprogram performs 10 database calls (`D=10`), 6 external program calls (`C=6`) and 7345 Natural statements (`N=7345`).
- The `Totals` section at the end of the program trace shows the maximum count of each event group. For example: a total of `2140` database calls corresponds to 2140 Database Before (`DB`) and 2140 Database After (`DA`) events.
- The totals of the Session (`S`) and Program (`P`) event groups are only listed under `Totals`; they are not listed next to the program name.

For further explanations of the trace columns, see the section ***Event Trace***.

For explanations of event types and associated event groups, see the section ***Events***.

Program Summary

If `PROGRAM=ON` is specified for an NPRC resource file, the Profiler program summary is generated.

The program summary shows for each Natural object how many Natural events have occurred, the total CPU time (in milliseconds) and the percentage of the CPU time spent by the Natural object with respect to the total CPU time.

Monitor Pause events and events at Level 0 are not taken into account for the program summary. Events which belong to one event group are combined into one count: see ***Events***.

Program starts and load requests are listed separately.

If the data is exported in CSV (comma-separated values) format, the count of each event type is listed. Additionally, the elapsed time and the Adabas times (absolute and percentage values) are displayed. The exported time values are indicated in microseconds.

Example of a Program Summary

The following example shows the extract of a program summary:

```
Natural Profiler Program Summary
--------------------------------
Library  Program      Start      Load  Database       I/O  External     Error Statement      User CPU-Time (ms)  CPU %
SYSEDMD  ADA-CL          41         0        40         0        41         0       621         0         3.785   0.14
SYSEDMD  ADA-RC          45         0        44         0        45         0       545         0         4.704   0.17
SYSEDMD  AOS-CL         115        97        15         0         0         0      2507         0        42.890   1.63
SYSEDMD  AOS-OP         169       154        22         0         0         0      6975         0        70.286   2.68
SYSEDMD  BYTE             1         0         0         0         0         0        11         0         0.034   0.00
SYSEDMD  CALLMON3         1         5        23         0         0         0      7089         0        20.001   0.76
SYSEDMD  CALLNOM          6         6        19         0         0         0        18         0         1.342   0.05
SYSEDMD  CALLNOPM         2         2         4         0         0         0        16         0         0.395   0.01
SYSEDMD  CALLNOPN         1         1         4         0         0         0         8         0         0.244   0.00
SYSEDMD  CALLNOPS         3         4        23         0         0         1        31         0         1.841   0.07
SYSEDMD  DISNOP           1         7         6         0         0         0       515         0         2.260   0.08
SYSEDMD  DISNO4I          1        47         3         0         1         0      8075         0        25.516   0.97
SYSEDMD  DISNO4IS        57         0         0         0       624         0     36877         0       105.650   4.03
SYSEDMD  DISNRS           1         0         0         0        44         0       511         0         3.343   0.12
SYSEDMD  DISNSP           1        18        15         0         0         0      1850         0         6.074   0.23
SYSEDMD  DISNTMZ          1         4        11         0         0         0       324         0         2.309   0.08
SYSEDMD  MENU             1         1         3         0         0         0         2         0         0.235   0.00
SYSEDMD  MONACSH          1         6         6         0         0         0      1217         0         3.470   0.13
SYSEDMD  MONADA           1      3176        71         0         0         0    272180         0       680.214  25.98
SYSEDMD  MONAREP          1         9        28         0         0         0      1964         0         6.378   0.24
...
Total                  5294      5293      2122         7      6510        43    857384         0      2617.326 100.00
```

Explanations:

- The Natural object `MONADA` consumed the most CPU time: `680.214` ms which corresponds to `25.98` percent of the total CPU time.
- `MONADA` was started once, it loaded `3176` other Natural objects, performed `71` database calls and `272180` Natural statements. There was no I/O, no external call and no error in the program.
- At the end of the program summary, the `Total` counts of the profiling are listed.

Line Summary

If `LINE=ON` is specified for an NPRC resource file, the Profiler line summary is generated.

The line summary shows for each source line in a Natural object, the number of Natural events that occurred (hit count), the CPU and elapsed time (in milliseconds and percent) spent by the line. The percentage of times is calculated in relation to the total times of the application.

The line summary does not count Monitor Pause events and events at Level 0.

If the data is exported in CSV (comma-separated values) format, the count of each event type is listed. Additionally, the Adabas times (absolute and percentage values) are displayed. The exported time values are indicated in microseconds.

Example of a Line Summary

The following example shows the extract of a line summary:

```
Natural Profiler Line Summary
-----------------------------
Library  Program  Line CC-Lib   CC-Name  Hit-Count       CPU (ms)  CPU %  Elapsed (ms)  Ela %
PRFTEST  XINT     0000                          1          0.016   0.46         0.003   0.01
PRFTEST  XINT     0140                          1          0.005   0.14         0.001   0.00
PRFTEST  XINT     0150                          1          0.006   0.17         0.002   0.01
PRFTEST  XINT     0160                          1          0.004   0.11         0.001   0.00
PRFTEST  XINT     0170                         23          0.128   3.75         0.029   0.18
PRFTEST  XINT     0180                         10          0.049   1.43         0.012   0.07
PRFTEST  XINT     0190                         10          0.054   1.58         0.010   0.06
...
Total                                         371          3.408 100.00        15.992 100.00
```

Explanations:

- Line 0170 in the Natural object XINT consumed 0.128 ms of the CPU time and 0.029 ms of the elapsed time. This corresponds to 3.75 percent of the total CPU time and 0.18 percent of the total elapsed time. 23 events (hit count) were executed in the line.
- At the end of the line summary, the Total counts of the profiling are listed.

Transaction Summary

A transaction is the code executed between two consecutive terminal I/O operations. The elapsed time spent on executing a transaction is called response time.

The transaction summary generated by the Profiler shows how many Natural events have occurred for each transaction, the response time (in milliseconds) and the percentage of the response time spent by the transaction with respect to the total response time.

Prerequisites Required for Transaction Evaluations

The Profiler data must be consolidated with the CONSOLIDATE function set to TRANSACTION=ON so that the consolidated data records in the NPRC resource file contain transaction identifiers required for evaluating the transaction response time.

TRANSACTION=ON is specified for the READ function.

Event Data Evaluated for Transactions

In general, one terminal I/O event relates to one transaction. The number of terminal I/O events is not listed in standard output (for PRINT=ON).

Data exported in CSV (comma-separated values) format contains the count of each event type, including terminal I/O. Additionally, the data list contains the CPU time and the Adabas times (absolute and percentage values). Exported time values are indicated in microseconds.

The transaction summary does not consider Before Terminal I/O (IB) events, RPC Wait for Client (RW) events, Monitor Pause (MP) events and events at Level 0. Events that belong to one event group are combined into one count.

For explanations of event types and associated event groups, see the section *Events*.

Modified Line and Program Summary

If TRANSACTION=ON is specified for the READ function of an NPRC resource file, the program summary and the line summary also consider the transaction identifier and aggregate the program and line values for each transaction separately. In the **transaction line summary**, the **ID** column indicates the program line in which the transaction was started.

Example of a Transaction Summary

The following example shows a transaction summary:

```
Natural Profiler Transaction Summary
------------------------------------
Transact TA-Lib   TA-Prog  TA-CC TA-Line   Program  Database  External     Error Statement      User  Elapsed (ms)  Time%
       1                                        1         0         1         0         0         0         0.829   0.50
       2 SYSPRFLR PRFMENM           0020         9        13         8         0         0         0        17.618  10.65
       3 SYSEDM   MOPTTEST          0020         4         0         0         0         0         0         6.167   3.72
       4 SYSEDM   OPTWLS80          0470         0         0         0         0         0         0         1.108   0.66
       5 SYSEDM   OPTWLS80          0470         2         0         0         0         0         0         1.133   0.68
       6 SYSEDM   OPTTEST           1750         1         0         0         0         0         0         1.180   0.71
       7 SYSEDM   MOPTTEST          0020        27        33        30         0         0         0        94.236  56.96
       8 SYSEDM   OPTTEST           5590         1         0         0         0         0         0         1.185   0.71
       9 SYSEDM   MOPTTEST          0020        37        33        56         0         0         0        37.311  22.55
      10 SYSEDM   OPTTEST           5760         1         0         0         0         0         0         1.096   0.66
      11 SYSEDM   MOPTTEST          0020         6         0         6         0         0         0         2.164   1.30
      12 SYSPRFLR PRFMENM           0020         2         0         3         0         0         0         1.394   0.84
Total                                         86        79       103         0         0         0       165.421 100.00
```

Explanations using the example of Transaction `7`:

- Transaction `7` spent the longest response (elapsed) time: `94.236` milliseconds (`ms`) which correspond to `56.96` percent of the total response time.
- Transaction `7` started an I/O operation in the library `SYSEDM` (`TA-Lib` column), program `MOPTTEST` (`TA-Prog`), program line `0020` (`TA-Line`).
- Transaction `7` accessed `27` programs and issued `33` database calls and `30` external program calls.
- In the example above, all counters for statement events are `0` because statement events were not collected during this Profiler run.
- At the end of the transaction summary, the `Total` counts of the profiled transactions are listed.

Program Coverage

The program coverage table is generated if `PROGRAM=ON` is specified for an NCVF resource file.

The program coverage table shows the code coverage results for each accessed Natural object. If the table is given in text format, only the GP coverage results (copycodes included) are displayed. In CSV (comma-separated values) format, the table shows lines containing copycode values, additional columns with source counters (copycodes not included) and information regarding `INCLUDE` statements.

In text format, the table provides the coverage count for each accessed library and for the whole application.

The table contains the following columns:

Column	Description	
Evaluation	The type of evaluation. Possible types are:	
	Program	For program coverage data
	Event	For statement coverage data
	Statistics	For Profiler statistics data
Object Count	The count of cataloged objects (GPs) listed in the table.	
Object Type	The type of Natural object such as program and subprogram.	

Column	Description
Library	The Natural library that contains the object.
Object	The name of the Natural object.
Copycode ID	The unique identifier of the copycode instance in the cataloged object (GP). The program gets the copycode ID `0`.
Copycode Library	The library from which the copycode is included.
Copycode Name	The name of the copycode.
GP Coverage%	The percentage of object coverage whereby `INCLUDE` statements are resolved.
GP Covered	The number of covered (executed) statements whereby `INCLUDE` statements are resolved.
GP Missed	The number of missed (not executed) statements in the object whereby `INCLUDE` statements are resolved.
GP Total	The total number of all executable statements in the object whereby `INCLUDE` statements are resolved.
Src Coverage%	The percentage of object coverage whereby `INCLUDE` statements are not resolved.
Src Covered	The number of covered (executed) statements whereby `INCLUDE` statements are not resolved.
Src Missed	The number of missed (not executed) statements in the object whereby `INCLUDE` statements are not resolved.
Src Total	The total number of all executable statements in the object whereby `INCLUDE` statements are not resolved.
First Statement	The ID of the first statement of the object or copycode.
INCLUDE CC-ID	For copycode only. The copycode ID of the object or copycode that includes the copycode.
INCLUDE Object	For copycode only. The name of the object or copycode that includes the copycode.
INCLUDE Line	For copycode only. The line number of the `INCLUDE` statement that includes the copycode.

Example of Program Coverage

The following example shows the result of program coverage in text format:

```
Program Coverage
----------------
Library   Object   Ty Coverage%  Covered    Missed    Total
COVDEMO   TESTCOVN N      78.5%       11         3       14
COVDEMO   TESTCOVP P      57.1%        4         3        7
COVDEMO   -------- --     71.4%       15         6       21
Totals    -------- --     71.4%       15         6       21
```

Explanations:

- The application accesses two objects, the TESTCOVN subprogram (N) and the TESTCOVP program (P).
- In TESTCOVN, there are 14 executable statements from which 11 were covered (executed) and 3 missed (not executed), giving a total coverage of 78.5%.
- The summarized values of the two objects accessed in the library COVDEMO show coverage of 71.4%.
- Total coverage is also 71.4% because only one library is accessed by the objects.

Statement Coverage

Statement coverage is generated if EVENT=ON is specified for an NCVF resource file.

For statement coverage, the Profiler utility reads the source of the monitored objects. If the source is not found or if the source does not match the collected data, source lines are not printed in the statement coverage report. The Profiler utility resolves INCLUDE statements and merges the source of the corresponding copycode into the including program. If the INCLUDE structure cannot be resolved, the copycodes are printed separately.

We recommend that you perform the READ function soon after the coverage run, as long as the sources correspond to the monitored GPs. As soon as the sources have been modified, the Profiler utility can no longer provide the full information.

Statement coverage shows the percentage of statements covered for each source line of the accessed programs. If the result is written in text format, for each object listed in the statistics, the object coverage values are shown before the statement coverage data. If the result is written in CSV (comma-separated values) format, additional information regarding statement coverage is provided.

The table contains the following columns:

Column	Description	
Evaluation	The type of evaluation. Possible types are:	
	Program	For program coverage data
	Event	For statement coverage data
	Statistics	For Profiler statistics data
Object Count	The count of objects (GPs) listed in the table.	
Library	The Natural library that contains the objects.	
Object	The name of the Natural object.	
Copycode ID	The unique identifier of the copycode instance in the related cataloged object. The program gets the copycode ID 0.	
Copycode Library	The library that contains the copycode (if copycode is active).	
Copycode Name	The name of the copycode (if copycode is active).	
Line	The line number in the Natural source object, for example, 0120.	

Column	Description
Source	The Natural source line that contains a statement definition, for example, `MOVE #A TO #B`.
Coverage%	The percentage of statement coverage of the line.
Covered	The number of statements covered (executed) in the line.
Missed	The number of missed (not executed) statements in the line.
Total	The total number of all executable statements (object code instructions) in the line.
Item Coverage	Indicates which statement items (object code instructions) in the line have been covered or missed. Each statement is represented by either `1` or `0`, whereby `1` indicates a covered statement and `0` a missed statement. For example: A value of `x100` indicates that only the first of three statements in the line is covered.
Mark	Indicates the coverage state of the line. The Mark column can be used to visualize the coverage results in tools like Microsoft Excel. Possible Mark values are listed in *Using a Microsoft Excel Template to Visualize Coverage Results*.

Example of Statement Coverage

The following example assumes that the development has delivered a new version of the `TESTCOVN` subprogram to the quality engineering. After running the test programs, statement coverage of the subprogram shows the following result (text format):

```
M Cov% CC-Lib   CC-Name  Line Source
*                        0010 * Test function Coverage
*                        0020 * Subprogram TESTCOVN
+                        0030 DEFINE DATA
+                        0040 PARAMETER
+                        0050 1 FUNC     (I2)  /* function
+                        0060 1 RET-CODE (I4)  /* Return code
+                        0070 END-DEFINE
*                        0080 *
*                        0090 /* Return 0 by default
C 100%                   0100 RESET RET-CODE
*                        0110 *
C 100%                   0120 DECIDE ON FIRST VALUE OF FUNC
+                        0130   VALUE 0
+                        0140     PRINT 'Test function 0'
+                        0150   VALUE 1
C 100%                   0160     PRINT 'Test function 1'
+                        0170   VALUE 2
C 100%                   0180     PRINT 'Test function 2'
+                        0190   VALUE 3
C 100%                   0200     PRINT 'Test function 3'
+                        0210   VALUE 4
C 100%                   0220     PRINT 'Test function 4'
+                        0230   VALUE 5
C 100%                   0240     PRINT 'Test function 5'
```

```
+                         0250   VALUE 6
C 100%                    0260     PRINT 'Test function 6'
+                         0270   VALUE 7
C 100%                    0280     PRINT 'Test function 7'
+                         0290   VALUE 8
C 100%                    0300     PRINT 'Test function 8'
+                         0310   VALUE 9
+                         0320     PRINT 'New test function 9'
+                         0330   NONE VALUE
+                         0340     RET-CODE := 1  /* Unsupported function
+                         0350 END-DECIDE
*                         0360 *
C 100%                    0370 END
```

Explanations:

- The Mark (`M`) column shows whether a line is covered (`C`).
- No test cases cover the functions `Test function 0` and `New test function 9`. The `NONE VALUE` case is also not covered.
- All other test cases are covered (denoted with `C` and 100% coverage).

As a consequence of this coverage analysis, the test cases have to be adjusted so that `Test function 0` and `Test function 9` (and, perhaps, the error case with an unsupported function code) are also covered.

Using a Microsoft Excel Template to Visualize Coverage Results

Prerequisites: Microsoft Excel and Natural for Windows or Natural for UNIX.

If you want to analyze the coverage result with Microsoft Excel, you can use the Microsoft Excel template delivered with Natural for Windows and Natural for UNIX. Perform the following steps:

1. Perform the Profiler `READ` function and write the output data in CSV (comma-separated values) format to Work File 7. For example:

```
FUNCTION=READ           /* Read Profiler Data
  RESOURCE-NAME='Test'  /* Resource name
  RESOURCE-LIB=PRFDATA  /* Resource library
  RESOURCE-TYPE=NCVF    /* Use resource type NCVF
  EVENT=ON              /* Print statement coverage
  PROGRAM=ON            /* Print program coverage
  STATISTICS=ON         /* Print statistics
  PRINT=ON              /* Write to standard output
  EXPORT=ON             /* Write to Work File 7
  FORMAT=COMMA          /* Export in CSV format
```

2. If your Microsoft Excel requires semicolons as separators, specify the following:

```
FORMAT=SEMICOLON      /* Export in CSV format
```

3. Export the data of Work File 7 with any tool (such as FTP) as a CSV-formatted file to a Windows environment.
4. Open the CSV file with Microsoft Excel.
5. Rearrange the data so that each evaluation type (program, event, statistics) is on its own worksheet in the Microsoft Excel file.
6. Open the delivered template `TESTCOV.XLSX` with Microsoft Excel. The template is contained in the **RES** (Resources) subdirectory of the Natural `SYSPRFLR` system library.
7. For each worksheet, copy the format from the template to your Microsoft Excel:
   - Click on the upper left corner of the table in the template to mark all data in the table.
   - Click on the Microsoft Excel **Copy format** function.
   - Click on the upper left corner of the table in your worksheet to copy the format.

   Now, all entries are formatted as in the template. The source lines are colored and marked as follows:

Color	Mark	Description
Green	`C`	All statements in the line are covered.
Yellow	`P`	The statements in the line are partly covered.
Pink	`M`	All statements in the line are missed.
Gray	`*`	A comment or an empty line.
Red	`E`	Error encountered. For example, if the coverage analysis has collected a line number but the corresponding source line is not found.
None (white)	`+`	All other lines such as continuation lines of a statement.

Example of a Microsoft Excel Worksheet

The following example shows a worksheet extract of code coverage for the `TESTCOVP` program with included `TESTCOVC` copycode without the columns that contain the object name and library:

Copycode ID	Copycode Library	Copycode Name	Line	Source	Coverage%	Covered	Missed	Total	Item Coverage	Mark
0			10	* Test Coverage		0	0	0		*
0			20	* Program TESTCOVP		0	0	0		*
0			30	DEFINE DATA LOCAL		0	0	0		+
0			40	1 FUNC (I2) /* function		0	0	0		+
0			50	1 RET-CODE (I4) /* Return code		0	0	0		+
0			60	END-DEFINE		0	0	0		+
0			70	FOR FUNC = 1 TO 8	100	1	0	1	x1	C
0			80	/* Test the subprogram functions		0	0	0		*
0			90	CALLNAT 'TESTCOVN' FUNC RET-CODE	100	1	0	1	x1	C
0			100	INCLUDE TESTCOVC 'RET-CODE' 'FUNC'		0	0	0		+
1	QFTEST	TESTCOVC	10	* Test Coverage		0	0	0		*
1	QFTEST	TESTCOVC	20	* Copycode TESTCOVC		0	0	0		*
1	QFTEST	TESTCOVC	30	IF &1& > 0	100	1	0	1	x1	C
1	QFTEST	TESTCOVC	40	IF &1& = 1		0	0	0		+
1	QFTEST	TESTCOVC	50	PRINT 'Unsupported function' &2&		0	0	0		+
1	QFTEST	TESTCOVC	60	ELSE		0	0	0		+
1	QFTEST	TESTCOVC	70	PRINT 'Return code:' &1&		0	0	0		+
1	QFTEST	TESTCOVC	80	END-IF		0	0	0		+
1	QFTEST	TESTCOVC	90	END-IF		0	0	0		+
0			110	END-FOR		0	0	0		+
0			120	*		0	0	0		*
0			130	END	100	1	0	1	x1	C

Explanations:

- The source lines of the `TESTCOVC` copycode are included in the source of the `TESTCOVP` program and placed right after the corresponding `INCLUDE` statement.
- The lines 40 through 70 of the copycode contain statements which were not executed in the test run.
- All other lines (in green) containing executable statements are covered.

Statistics

If `STATISTICS=ON` is specified, the Profiler statistics are listed.

Note: Statistics are not provided for an NCVF file on UNIX or Windows.

If the data is exported in CSV (comma-separated values) format, the properties and values of the Profiler statistics are added as separate columns to the event or consolidation trace. If coverage data is exported in CSV format, the statistics values are added in additional lines indicated by the value `Statistics` in the Evaluation column.

Example of Statistics

The following example shows an extract of the statistics of an NPRC resource file:

```
************************************************************************
* 17:35:59          ***** NATURAL PROFILER UTILITY *****         2016-01-11
* User SAGTEST1                - Statistics -                       RESDATA
...
*
* Profiler Resource File
* Resource name ..................... EDM-MONITOR.nprc
* Resource type ..................... Natural Profiler Resource Consolidated
* Resource allocation date ........... 2015-07-27 10:36:19.6
* Resource size (bytes) .............. 565160
...
*
* Data Processing
* Number of events .................. 895936
...
*
* Data Consolidation
* Consolidation ..................... ON
* Consolidation records .............. 21624
* Consolidation elapsed time (sec) ... 15.643516
* Consolidation factor ............... 41.4
* Consolidation records/block ........ 191.3
* Bytes/consolidation record ......... 25.8
*
...
************************************************************************
```

Explanations:

- The `EDM-MONITOR.nprc` resource was allocated on `2015-07-27` at `10:36:19` a.m. and has a size of `565160` bytes.
- The profiled application generated a total of `895936` Natural events. The data consolidation took `15.6` seconds and reduced the number of records to `21624` which corresponds to a consolidation factor of `41.4`.

All statistics information provided is explained in the section ***Profiler Statistics***.

Note: The Natural Profiler for UNIX and Windows does not collect statistical data. The statistics provided are values determined by the Profiler utility.

Exporting Event Data for MashZone

You can visualize Profiler event data on an interactive MashZone dashboard by using the **Natural Profiler MashApp**.

The Profiler utility `MASHZONE` function reads the consolidated data of an NPRC resource file and writes the data to Work File 7 in the format expected by the Natural Profiler MashApp. Use as Work File 7 a CSV (comma-separated values) file in the Natural Profiler data directory in the MashZone environment which can be accessed by the Natural Profiler MashApp.

Syntax of `MASHZONE`:

```
FUNCTION=MASHZONE
 [RESOURCE-NAME=resource-name]
 [RESOURCE-LIB=library-name]
```

Syntax Description:

Keyword	Value	Description
`RESOURCE-NAME`	*resource-name*	The name of the Natural Profiler resource consolidated (NPRC) file to be exported for MashZone. The extension `.nprc` is added automatically. Default: The name of the last created NPRC resource file in the library
`RESOURCE-LIB`	*library-name*	The name of the Natural library that contains the NPRC resource file you want to export. Default: The name of the current library

Alternative Function Specifications

READ

The following Profiler utility `READ` function is equivalent to the `MASHZONE` function and generates the same export data:

```
FUNCTION=READ
  RESOURCE-NAME=resource-name
  RESOURCE-LIB=library-name
  RESOURCE-TYPE=NPRC
  EVENT=ON
  PROGRAM=OFF
  LINE=OFF
  TRANSACTION=OFF
  STATISTICS=ON
```

```
  PRINT=OFF
  EXPORT=ON
  FORMAT=SEMICOLON
```

CONSOLIDATE

The **Natural Profiler MashApp** can also process data exported with the Profiler utility `CONSOLIDATE` function if you specify the following keywords:

```
FUNCTION=CONSOLIDATE    /* Consolidate Profiler data
  TRANSACTION=ON        /* Add transaction identifiers
  EXPORT=ON             /* Write to Work File 7
  FORMAT=SEMICOLON      /* CVS format with semicolon separator
  ...
```

Example of MASHZONE

The following example reads the consolidated Profiler resource `Test.nprc` in the library `PRFDATA`. The data is written in CSV (comma-separated values) format to Work File 7 which can be accessed by MashZone.

```
FUNCTION=MASHZONE       /* Export Profiler data for MashZone
  RESOURCE-NAME='Test'  /* Resource name
  RESOURCE-LIB=PRFDATA  /* Resource library
```

Maintaining Profiler Resource Files

In general, Profiler resources are listed as NPRF, NPRC or NCVF files by using the Natural SYSMAIN utility, NaturalONE or Natural Studio. These tools also provide functions to copy, rename and delete resource files.

In addition, you can use Profiler utility functions to list Profiler resource files.

This section covers the following topic:

- Listing Profiler Resource Files

Listing Profiler Resource Files

The Profiler utility `LIST` function lists the Profiler resource files of a given Natural library and the date and time when the resource files were allocated.

Syntax of `LIST`:

```
FUNCTION=LIST
 [RESOURCE-LIB=library-name]
 [RESOURCE-TYPE={NPRF|NPRC|NCVF}]
 [PRINT={ON|OFF}]
 [EXPORT={ON|OFF}]
 [FORMAT={TEXT|COMMA|SEMICOLON}]
```

Syntax Description:

Keyword for LIST	Value	Description
RESOURCE-LIB	*library-name*	The name of the Natural library that contains the Profiler resource files you want to list. Default: The name of the current library
RESOURCE-TYPE		Specifies the type of resource files to be listed: NPRF, NPRC or NCVF. Default: All types are listed if no value is specified here.
	NPRF	List NPRF (Natural Profiler Resource File) resource files only.
	NPRC	List NPRC (Natural Profiler Resource Consolidated) resource files only.
	NCVF	List NCVF (Natural code coverage file) resource files only.
PRINT		Specifies whether the result is written to standard output.
	ON	Write to standard output.
	OFF	Do not write to standard output.
EXPORT		Specifies whether the result is written to Natural Work File 7.
	ON	Write to Work File 7.
	OFF	Do not write to Work File 7.
FORMAT		Specifies the format in which the exported data is written to Work File 7.
	TEXT	Write the data in free text format.
	COMMA	Write the data in CSV format with a comma (,) used as a separator.
	SEMICOLON	Write the data in CSV format with a semicolon (;) used as a separator.

Example of LIST

The following example lists the NPRF Profiler resource files of library PRFDATA. The list is written to standard output and to Work File 7 in text format.

```
FUNCTION=LIST           /* List Profiler resource files
  RESOURCE-LIB=PRFDATA  /* Resource library
  RESOURCE-TYPE=NPRF    /* List NPRF resource files
  PRINT=ON              /* Write to standard output
  EXPORT=ON             /* Write to Work File 7
  FORMAT=TEXT           /* Export in text format
```

Output:

```
Natural Profiler Resources
--------------------------
Library: PRFDATA
Resource type: nprf

Count Date       Time     Name
    1 2015-06-15 14:32:18 Hello1.nprf
    2 2015-06-26 18:39:57 QDTest1.nprf
    3 2015-06-24 22:00:35 QETest1.nprf
    4 2015-06-30 14:32:42 Studio.nprf
    5 2015-07-02 15:02:32 Test.nprf

Number of nprf resources in library PRFDATA: 5
```

Including Profiler Input from Natural Text Objects

The Profiler can read input data from a Natural text object. The syntax of the data in the Natural text object is the same as for the primary command input data set CMSYNIN (see *Syntax and Keywords*).

To include Profiler input data from a Natural text object

- Enter the following Profiler keywords:

```
INCLUDE-LIB=library-name
INCLUDE=object-name
```

The keyword syntax is explained in *Profiler Utility Keywords*.

The data in the Natural text object is added to the Profiler input data in the line after the INCLUDE keyword. The Profiler input data can contain multiple INCLUDE keywords, and the related Natural text objects can also contain INCLUDE keywords. If a Natural text object contains an END-PROFILER

keyword, the Profiler utility terminates and any remaining data in the Natural text object(s) is ignored.

The Natural system library SYSPRFLR supplies text object whose names begin with X which can be used as Profiler input. The individual Profiler functions they perform are described in the sources of these objects.

We recommend that you do not modify any objects in the system library SYSPRFLR because they can be overwritten or removed when a new Natural version is installed. Copy the required object(s) to a user library before you edit it.

Examples of INCLUDE

The following example adds the contents of the Natural MYPROF text object from the library MYLIB to the Profiler input data:

```
INCLUDE-LIB=MYLIB
INCLUDE=MYPROF
```

The following example adds the content of the Natural text member XCONS from the library SYSPRFLR to the Profiler input data. The object consolidates Profiler event data. Additionally, it terminates the Profiler utility so that no further Profiler input is expected after the INCLUDE keyword.

```
INCLUDE=XCONS
```

Event Trace

The Natural Profiler collects detailed information of each Natural event that occurs while a Natural application executes. This data can be viewed in the event trace.

The traces written for Natural code coverage are described in the section *Tracing Natural Code Coverage*.

The Profiler utility provides the following options to write a Profiler event trace:

- Write the trace to standard output while the NPRF data is consolidated. In this case, the event trace shows the delta values of the elapsed time and the CPU time instead of event-specific data.
- Write the trace when reading a Profiler NPRF resource file with the Profiler utility READ function.

Note: The event trace can also be listed in NaturalONE.

To enable the event trace

- Enter the following subordinate keyword of the Profiler utility CONSOLIDATE function:

```
TRACE-EVENT=ON
```

Enter the following subordinate keyword of the Profiler utility `READ` function:

```
EVENT=ON
```

The Profiler event trace contains the following columns:

Column	Description
Count	Event count.
Time	Event time. Unit: *hour*:*minute*:*second*.*microseconds*
CPU-Time	Session CPU time. Unit: microseconds
Ev	Event type; see ***Events and Data Collected***.
Lev	Program level.
Library	Program library.
Program	Program (Natural object) name.
Line	Line number of program statement executed.
CC-Lib	Copycode library (if copycode is active).
CC-Name	Copycode name (if copycode is active).
Statement	Natural statement currently executed. For technical reasons, there is no one-to-one relationship between a Natural source code statement and the corresponding object code in the cataloged object. Therefore, the statements listed in the Profiler event trace can differ from the statements in the source.
Local-Data	Event-specific data like the Adabas database ID (DBID) and file number (FNR). This data is only displayed for the Profiler utility `READ` function.
Elapsed (ms)	Elapsed time spent processing the event. Unit: milliseconds This data is only displayed for the Profiler utility `CONSOLIDATE` function.
CPU-Delta	CPU time spent processing the event. Unit: milliseconds

Example of an Event Trace

In the following example, the Profiler utility `READ` function prints the event trace:

```
FUNCTION=READ                /* Read event data
  EVENT=ON                   /* Write event trace
```

The event trace is written to standard output:

```
Count Time              CPU-Time (ms) Ev Lev Library  Program  Line CC-Lib   CC-Name  Statement  Local-Data
    0 17:38:17.200951         42.324 MP 003 SYSPRFLR PRBINIT  8370                 Call       Monitor pause requested
    0 17:38:17.204508          43.471 MP 003 SYSPRFLR PRBSTART 1760                   Call       Start of block filter
   11 17:38:17.218379          48.874 DB 000                   0000                              00010/00032 S1
   12 17:38:17.218941         48.897 DA 000                  0000                              00010/00032 S1   Rsp: 0
   13 17:38:17.218944          48.910 PL 000                   0000                               Execute PRFDEMO/XPROF
   14 17:38:17.218945          48.916 PS 001 PRFDEMO  XPROF    0000                     PgmStart   00010/00032 Type: P
   15 17:38:17.218956          48.979 IB 001 PRFDEMO  XPROF    0300                     Input      Out: 133 In: 0
   16 17:38:17.219235          49.046 IA 001 PRFDEMO  XPROF    0300                     Input      Out: 133 In: 80
   17 17:38:17.219258          49.182 DB 001 PRFDEMO  XPROF    0370                     Callnat    00010/00032 S1
   18 17:38:17.220426         49.211 DA 001 PRFDEMO  XPROF    0370                   Callnat    00010/00032 S1   Rsp: 0
   19 17:38:17.220427          49.216 DB 001 PRFDEMO  XPROF    0370                     Callnat    00010/00032 S1
   ...
```

Tracing Natural Code Coverage

When the coverage resource is read with the Profiler utility `READ` function, the coverage data can be traced with the internal data trace.

To enable tracing for code coverage

- Enable the internal trace by specifying the following subordinate keyword of the Profiler utility `READ` function:

  ```
  TRACE=9
  ```

The table below describes the properties listed in the trace:

Property	Description
Count	The event count.
Ev	The event type. See *Events and Data Collected*.
Library	The name of the Natural library that contains the program/object.
Program/Object	The name of the Natural program/object.
Ty	The object type such as `P` for program.
CC-Lib	The name of the Natural library that contains the copycode (if copycode is active).
CC-Name	The name of the copycode.
Line	The source line number.

Property	Description
CC-ID	The copycode ID. It uniquely identifies the copycode instance in the GP. The program gets the copycode ID `0`.
Par-CC	For copycode only. The parent copycode ID which is the copycode ID of the object/copycode that includes the current copycode.
FirstS	The ID of the first statement of the object or copycode.
Stmts	The total number of executable statements in the object whereby all `INCLUDE` statements are resolved.
Item	The item ID of the statement. It uniquely identifies the statement in the resource file.
Cover	The coverage flag (`0` or `1`) of the statement. When the GP is read, all flags are initialized with `0`. Whenever a statement is executed, the flag is set to `1`.

Internal Trace

The Profiler internal trace writes Profiler messages such as errors or warnings.

The internal trace can be activated for the following:

- The Profiler data processing functions. The data is written to standard output.

To activate the internal trace for the Profiler data processing functions

- Enter the following Profiler keyword:

```
TRACE=n
```

where *n* is the trace level (see ***Trace Levels***).

Notes:

1. By default (if `TRACE` is not specified), Trace Level 2 (warnings) is used.
2. The trace is activated as soon as the `TRACE` keyword is specified. It is therefore recommended to specify the `TRACE` keyword as soon as possible.
3. If you execute the Profiler utility multiple times in the job, you need to specify the `TRACE` keyword with each execution.

Trace Levels

The trace levels used by the Profiler trace and monitor sessions and by the Profiler data processing functions are listed in the following table. In general, a higher trace level also contains the information of the lower trace levels. For example, if you select Trace Level 3 (statistics), error messages and warnings are also logged.

We recommend that you use at least Trace Level 2 (warnings) so that error messages and warnings are logged.

Trace Level	Name	Description
0	No trace	Profiler internal trace is deactivated.
1	Error	Log error messages.
2	Warning	Log warnings.
3	Statistics	Data consolidation: Print the ***profiler statistics*** including the consolidation statistics.
4	Function	Log messages for used **Profiler utility keywords** (`FUNCTION`, `FILTER`, etc.).
5	Block	Print the statistics of each data block written to the Profiler resource file.
6	Details	Log detailed information.
7		Not used.
8		Not used.
9	Data	Trace the coverage resource data when reading an NCVF coverage resource file.
10	Internal	Internal usage.

Example

In the following example, the Profiler internal trace is set to `4` (function):

```
* Set Profiler internal trace
TRACE=4                     /* Trace level
```

Profiler Statistics

In addition to event data, the Profiler collects statistical data which is written to the Profiler resource file.

The Profiler utility provides the following options to write and view Profiler statistics:

- Write the statistics to standard output while the data is consolidated.
- Write the statistics when reading a Profiler resource file with the Profiler utility `READ` function.
- View the statistics with the **Natural Profiler MashApp**.

To write Profiler statistics, perform one of the following steps

- Enter the following keyword before you start the Profiler utility `CONSOLIDATE` function:

```
TRACE=3
```

  or a higher trace level (see *Trace Levels*).

- Enter the following subordinate keyword of the Profiler utility `READ` function:

```
STATISTICS=ON
```

The Profiler statistical data is displayed in categories combining properties of a similar type. The following categories are available:

- General Information
- Profiler Resource File
- Monitor Session
- Trace Session
- Data Processing
- Event Type Statistics
- Monitor Pause Statistics
- Data Consolidation
- Coverage
- Transaction

Note: The properties listed in the following section are the properties provided by the Profiler in all environments. The Profiler statistics contain only the properties that are relevant for the current run. Therefore, not all of the properties listed in the section are displayed in every case.

General Information

Display environment and Natural Profiler related information.

Property	Description
Machine class	The name of the machine class on which the Natural application is running.
Environment	The environment in which the Natural application is running, such as NaturalONE, batch or RPC.
Codepage	The code page used while the Natural application was monitored.
User	The ID of the user running the application (value of `*USER`).
Profiler version	The internal version of the Profiler. NaturalONE environment: The version of the Profiler on the server.
Profiler revision	The internal revision of the Profiler.

Property	Description
Profiler revision date	The date and time when the Profiler revision was created.
Profiler client version	NaturalONE environment: The version of the Profiler client.
Profiler trace library	NaturalONE environment: The name of the Natural library containing the Profiler internal trace and the Profiler event trace.
Profiler trace level	The level of the Profiler internal trace.
Profiler trace member	NaturalONE environment: The name of the Natural text object containing the Profiler internal trace.
Profiler event trace	Indicates whether the Profiler event trace was activated (`ON/OFF`).
Profiler event trace member	NaturalONE environment: The name of the Natural text object containing the Profiler event trace.
Utility trace level	NaturalONE environment: The Natural utilities trace level.

Profiler Resource File

Display Profiler resource file related information.

Property	Unit	Description
Resource name		The name of the Natural Profiler resource file.
Resource type		The type of the Natural Profiler resource file: Natural Profiler resource file (NPRF), Natural Profiler resource consolidated (NPRC) or Natural code coverage file (NCVF).
Resource short name		Mainframe: The short name of the Natural Profiler resource file.
Resource library		The name of the Natural library containing the Natural Profiler resource file.
Resource DBID		The database ID of the Natural library containing the Natural Profiler resource file.
Resource FNR		The file number of the Natural library containing the Natural Profiler resource file.
Resource allocation date	*yyyy-mm-dd hh:ii:ss.t*	The date and time when the Natural Profiler resource file was allocated.
Resource size	bytes	The size of the Natural Profiler resource file. It comprises the resource headers, the event data and the properties. The resource size is calculated regardless whether the resource is allocated or not.
Resource block size	bytes	The maximum size of a resource block. A resource block consists of a resource block header and a data block.
Resource version		The version of the Natural Profiler resource layout.

Monitor Session

Display statistics of the Profiler monitor session.

Property	Unit	Description
Monitor start time	*yyyy-mm-dd hh:ii:ss.t*	The date and time when the monitor session started.
Monitor end time	*yyyy-mm-dd hh:ii:ss.t*	The date and time when the monitor session ended.
Monitor elapsed time	sec	The total elapsed time consumed by the monitor session.

Trace Session

Display statistics of the Profiler trace session. The Profiler trace session includes also the application execution.

Property	Unit	Description
First library		The first library monitored. The libraries SYSTEM, SYSLIB* and SYSPRF* are ignored.
First program		The first program monitored.
Highest level		Highest level number of the Natural objects monitored.
Trace start time	*hh:ii:ss.microsec*	The start time of the tracing. With NaturalONE this is the time of the SI (session initialization) event. In batch, the session is already initialized when the monitoring starts. Therefore, the start time is the time of the first event (usually a Monitor Pause event).
Trace end time	*hh:ii:ss.microsec*	The end time of the tracing. This is in general the time of the ST (session termination) event.
Trace elapsed time	sec	The elapsed time consumed by the trace session from the start time to the end time.
Application CPU time	ms	The total CPU time consumed by the application.
Monitor CPU time	ms	The total CPU time consumed by the Natural data collector. This time is not measured by the Natural UNIX or Windows server.
Total CPU time	ms	The total CPU time consumed by the trace session. It is the sum of the application CPU time and the monitor CPU time.
Sampling interval	microsec	The sampling interval time (CPU time in microseconds). A value of zero (0) means that no sampling was active.
Data pool empty		The number of Profiler read requests which found the Profiler data pool empty (and a session active).
Data pool empty after full		The number of Profiler read requests which found the Profiler data pool empty although it was full before. If this counter is greater than 0, the Profiler data pool is too small which leads to a poor performance.

Property	Unit	Description
Data pool overflow		The number of Profiler data pool overflows (with data lost). Data pool overflows should no longer happen. This property is only maintained for backward compatibility with previous versions of Natural.
No session active		The number of read requests which found the Profiler data pool empty and no trace session active. This can only happen for Profiler read requests submitted before the session initialization or after the session termination.

Data Processing

Display statistics of the data processing, compression and transfer.

Property	Unit	Description
Number of events		The total number of events.
Highest event number		The highest event number as given by the Natural data collector. Note that the Natural data collector counts only non-statement events when called from NaturalONE. In batch it depends on the statement filter whether statement events are counted or not.
Number of data blocks		The number of event data blocks send to NaturalONE or written to the resource.
Utility buffer size	bytes	The size of the utility buffer used for the data transfer from the server to NaturalONE. In general, the buffer contains the header information and function-specific data.
Data block size	bytes	The maximum amount of event data which can be transferred from the server to NaturalONE in one call. The same data block size is used for storing the event data in the resource file.
RDC data length	bytes	The total size of the data received from the Natural Data Collector.
Uncompressed data length	bytes	The total size of the Profiler data in uncompressed format.
Compressed data length	bytes	The total size of the compressed data as send to NaturalONE or written to the Profiler resource file.
Identical bytes trimmed left		The number of identical bytes trimmed left at the forward data compression.
Blanks trimmed right		The number of blanks trimmed right at the backward data compression.
Compression header length	bytes	The total size of the compression headers saved with each compressed event record.
Compression rate	percent	The percentage of the data reduction by the compression. The higher the compression rate, the less data has to be transferred or saved. The formula of the compression rate is described below.
Events/block		The average number of events contained in one event data block.

Property	Unit	Description
Bytes/event		The average length in bytes of a compressed event data record. This property is not available for consolidated or coverage data.

The compression rate is calculated by the following formula:

$$\text{CompressionRate} := 100 \times \frac{\text{BytesTrimmedLeft} + \text{BytesTrimmedRight} - \text{CompressionHeaderLength}}{\text{UncompressedDataLength}}$$

Event Type Statistics

Display statistics of the event types.

Property	Description
Unknown event	The number of unknown events.
Session initialization	The number of Session Initialization events.
Session termination	The number of Session Termination events.
Program load	The number of Program Load events.
Program start	The number of Program Start events.
Program termination	The number of Program Termination events.
Program resume	The number of Program Resume events.
Program information	The number of Program Information events.
Before database call	The number of Before Database Call events.
After database call	The number of After Database Call events.
Before terminal I/O	The number of Before Terminal I/O events.
After terminal I/O	The number of After Terminal I/O events.
Before external program call	The number of Before External Program Call events.
After external program call	The number of After External Program Call events.
Runtime error	The number of Runtime Error events.
Natural statement	The number of Natural Statement events.
Outbound RPC message	The number of Outbound RPC Message events.
Inbound RPC message	The number of Inbound RPC Message events.
Start RPC request execution	The number of Start of RPC Request Execution events.
RPC Wait for Client	The number of RPC Wait for Client events.
User trace call	The number of User-Defined Events.
Monitor pause	The number of Monitor Pause events.
Monitor filter	The number of monitor filter events. Filter events are not recorded.

Monitor Pause Statistics

Display statistics of the types of Monitor Pause events.

Property	Description
Pause - unknown type	The number of Monitor Pause events with unknown pause type.
Pause - requested	The number of requested Monitor Pause events.
Pause - start of block filter	The number of Monitor Pause events caused by a start of a block filter (library, program, line, FNAT, event count or time filter).
Pause - data pool full	The number of Monitor Pause events caused by a data pool full situation.
Pause - data pool overflow	The number of Monitor Pause events caused by a data pool overflow situation.

Data Consolidation

Display statistics of the data consolidation.

Property	Unit	Description
Consolidation		Indicates whether the Profiler data is consolidated (`ON`/`OFF`). The consolidation aggregates similar events into one consolidation record.
Consolidation records		The total number of consolidation records. In general, a consolidation record comprises multiple events.
Consolidation elapsed time	sec	The elapsed time in seconds required for the data consolidation with the Profiler utility `CONSOLIDATE` function.

Consolidation factor		The average number of events combined into one consolidation record. The higher the consolidation factor, the better the consolidation.
		
		`ConsolidationFactor := NumberOfEvents / ↩`
		`ConsolidationRecords`

Consolidation records/block		The average number of consolidation records contained in one data block.
Bytes/consolidation record		The average length in bytes of a compressed consolidation record.
Consolidate I/O time		Indicates whether I/O and Natural RPC client time are included in the consolidated data.

Coverage

Display statistics of Natural code coverage.

Note: Natural code coverage statistics are collected on the mainframe only.

Property	Description
Coverage	Indicates whether Natural code coverage is performed (ON/OFF).
Missed statements recorded	Indicates whether missed statements are recorded (ON/OFF).
Coverage records	The total number of coverage records. These are program information and Natural statement records.
Program information records	The number of program information records written to the resource file. Each program information record contains program and copycode-related information.
Coverage records/block	The average number of coverage records contained in one data block.
Bytes/coverage record	The average length in bytes of a compressed coverage record.
Programs covered	The number of covered programs.
Programs NOC-ed	The number of covered programs compiled with the Natural Optimizer Compiler.
Statement coverage	The percentage of statements of all accessed programs that have been covered by the application.
Statements covered	The number of covered (executed) statements.
Statements total	The total number of executable statements of all programs accessed.

Transaction

Display statistics of executed transactions.

Note: Transaction statistics are only available if Profiler data has been consolidated with the CONSOLIDATE function set to TRANSACTION=ON.

Property	Unit	Description
Transaction		Indicates whether transaction identifiers are added to the consolidated records (ON/OFF). Transaction identifiers are required to evaluate the response time for transactions.
Number of transactions		The total number of processed transactions. A transaction is the code executed between two consecutive terminal I/O operations.
Total response time	sec	The total elapsed time used to process all transactions.
Average response time	sec	The average elapsed time used per transaction.
Max response time	sec	The maximum (highest) elapsed time used to process transactions.

Property	Unit	Description
ID max response time		The transaction identifier of the transaction with the highest elapsed time. In addition to the transaction identifier, the statisitcs indicates the program name, copycode name (if any) and program line from where the terminal I/O operation was issued.

For more information on evaluating transactions, see ***Transaction Summary***.

77 Natural Profiler MashApp

MashZone is a browser-based application from Software AG which is used to visualize data on a graphical, interactive dashboard, a so-called MashApp. The Natural Profiler MashApps evaluate the Profiler event data and depict it in MashZone.

Preparing to Use the MashApps

This section provides instructions for implementing the MashApps:

Downloading the MashApps

The Natural Profiler MashApps and related data are supplied as a Natural component in a zip file.

To download the MashApps zip file

1 Log in to Software AG's Empower web site at *https://empower.softwareag.com/* (password required).

2 Go to **Products & Documentation** > **Download Components**.

The **Download Components** section is displayed.

3 From the **Download Components** section, select **Natural Profiler MashApp**.

4 Download the `NaturalProfiler_MashApp.zip` file.

In addition to the zip file, Empower also provides the Readme file `Readme_NaturalProfiler_MashApp.txt` which contains the latest update information.

Unpacking the Zip File

You have to unpack the MashApp zip file in the appropriate MashZone directory which depends on the MashZone version installed at your site.

To unpack the MashApp zip file

- Unpack the `NaturalProfiler_MashApp.zip` file in the appropriate user data directory of MashZone:
  - For MashZone Version 9.0 and above:

```
installation-directory\server\bin\work\work_mashzone_server-type\mashzone_data
```

where *server-type* indicates the type of the MashZone server: `s`, `m` or `i`. For example, `work_mashzone_m` for a medium type.

- For MashZone versions below Version 9.0:

```
installation-directory
```

where *installation-directory* is the MashZone installation directory.

After unpacking the zip file, the following subdirectories are available in the user data directory of MashZone:

Directory	Content
`importexport\Profiler_`*date*	MashApps for the Natural Profiler. *date* is the MashApp generation date.
`resources\Profiler`	Parent directory of Profiler resources. Contains the user-modified `Overview.csv` resource file. See also ***Editing the Overview.csv Resource File***.
`resources\Profiler\Definition`	Resources used by the MashApp. Initially, this directory contains the resources which do not have to be edited.
`resources\Profiler\Data`	Profiler data directory (including subdirectories) in which the Profiler data files are stored by default.
`resources\Profiler_src`	Source directory for resources which have to be edited and copied into the `resources\Profiler` directory. See also ***Editing the Overview.csv Resource File***.
`assets\colorschemes`	Color schemes. The color schemes for the Natural Profiler are named `Profiler_*.xml`.

Editing the Overview.csv Resource File

You can edit the `Overview.csv` resource file in the `resources\Profiler_src` directory to adapt the Natural Profiler MashApps to your requirements. The resource file is a CSV-formatted file with semicolon (;) separators which can be edited with any text editor.

The supplied `Overview.csv` file contains one line for the sample Profiler data in the `Profiler_Sample.csv` file in the `resources\Profiler\Data` directory. Add more lines for each Profiler CSV file you want to evaluate. For information on creating Profiler CSV files, see *Preparing the Profiler Data*. You can also add or delete lines in the `Overview.csv` file later, after you have copied it to the `resources\Profiler` directory (see *Activating the MashApp*).

In the columns of the `Overview.csv`, you can specify the following:

Column	Description
`csv File`	Specify the name of the Profiler consolidated data file. If the data file resides in a subdirectory of `resources\Profiler`, specify the relative path and the file name. For example: Specify `Data\ProfilerTrace.csv` if the `ProfilerTrace.csv` data file is contained in `...\resources\Profiler\Data`.
`Description`	Specify a descriptive name for the Profiler consolidated data file. The descriptions are used in the **Input** selection box of the Natural Profiler MashApps. If you do not enter a value, the value of the **csv File** column is used in the **Input** selection box.
`Enable`	If you enter `Y` in this column, the name or description of the Profiler consolidated data file is shown in the **Input** selection box. Otherwise, it is not shown.

Activating the MashApps

Prerequisites for activating the Natural Profiler MashApps are a Professional, Enterprise or Event license file and administrator rights.

To activate the MashApps

1 Copy the resource file from `resources\Profiler_src` to `resources\Profiler`.

2 Invoke MashZone.

3 Go to the **Administration** page (see the corresponding tab at the top of the page) and then to the **Import/Export/Delete** page.

4 Import the MashZone archive files (`*.mzp`) from the `importexport\Profiler_`*`date`* directory by using the **Import** function.

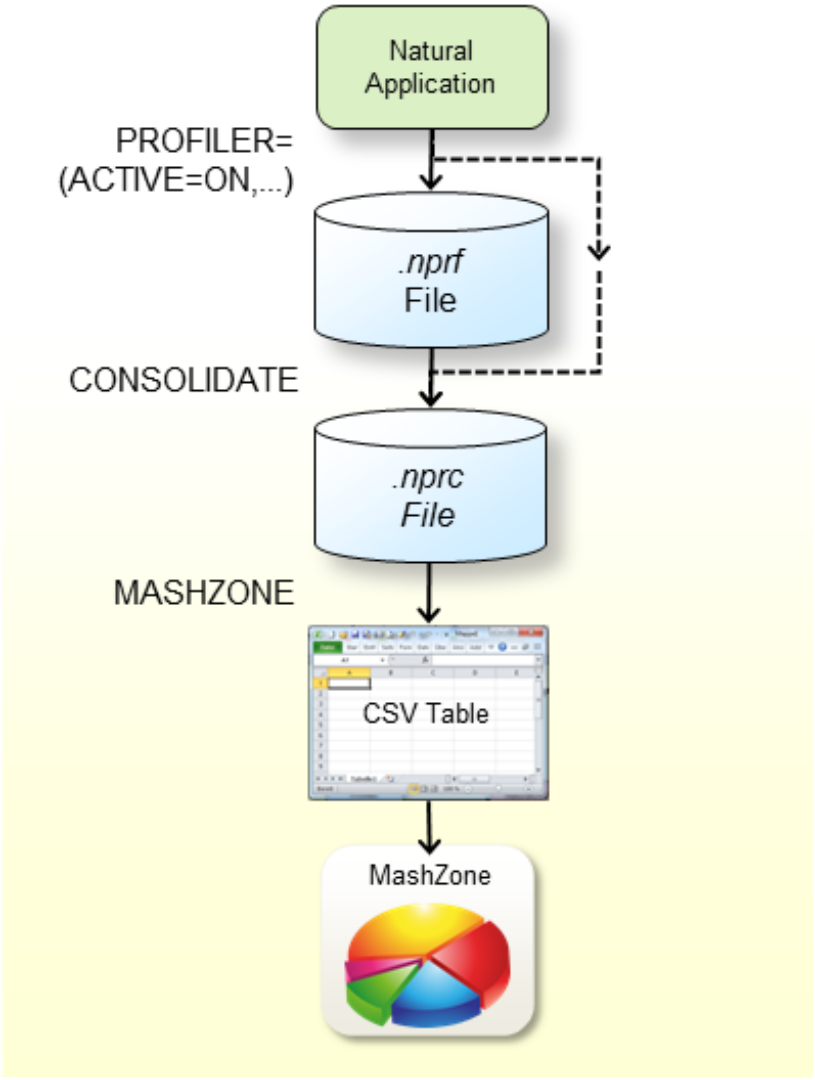
The MashApps in the `importexport\Profiler_`*`date`* directory are named as follows:

`M_MashAppName version_revision_date-time.mzp`

where *MashAppName* is the name of the MashApp in MashZone which can be either of the following:

MashAppName	Purpose
`Natural Profiler`	Evaluate the Natural Profiler data and the Profiler properties and statistics of a monitored application.
`Natural Profiler Compare`	Compare two Natural Profiler data files.

Preparing the Profiler Data



The graphic above illustrates the steps you have to perform before you can evaluate the Natural Profiler data in MashZone:

- Profile the Natural session by switching on the `ACTIVE` subparameter of the `PROFILER` profile parameter in the NATPARM parameter file or dynamically when invoking Natural. See *PROFILER – Profile a Natural Session* in the *Parameter Reference* documentation. The Profiler writes the event data to an `.nprf` Natural Profiler resource file.
- Consolidate the event data using the Profiler utility `CONSOLIDATE` function. The consolidated data is written to an `.nprc` Natural Profiler resource consolidated file.

- Alternatively, you can specify `EVENTTRACE=OFF` with the `PROFILER` profile parameter when you profile the Natural application. In this case, the Profiler writes the event data directly to an `.nprc` Natural Profiler resource file.
- Write the consolidated event data with the Profiler utility `MASHZONE` function in CSV (comma-separated values) format to Work File 7. Use as Work File 7 a CSV (comma-separated values) file in the Natural Profiler data directory (see *Unpacking the Zip File*) in the MashZone environment.
- Enter a reference to the new file in the `Overview.csv` file in the `resources\Profiler` directory.

If you start MashZone, you will find the description of the new file in the **Input** selection box. If you select the line with the description, the Natural Profiler MashApps read the event data from the corresponding CSV file.

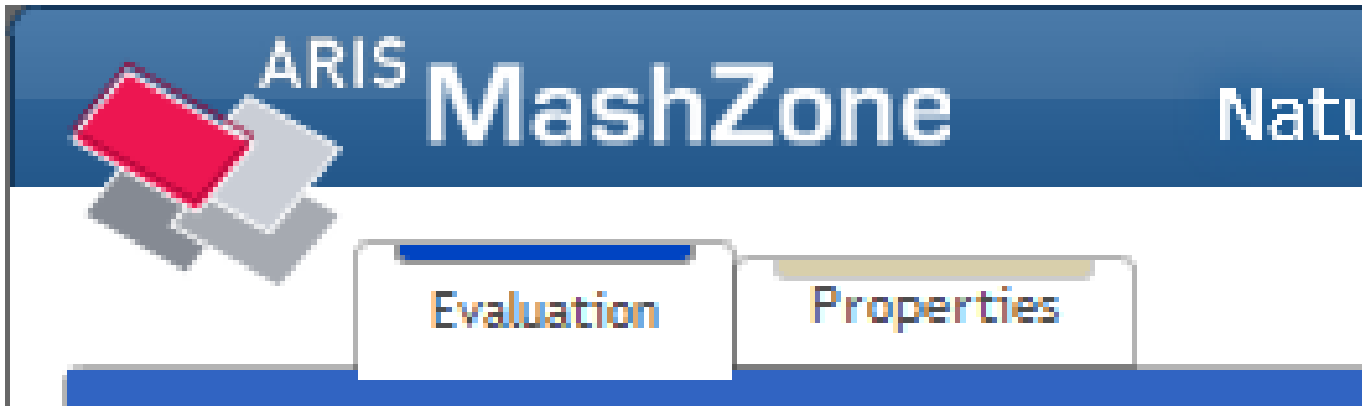
If you already started the Natural Profiler MashApp earlier, MashZone may not immediately detect the new entry in the `Overview.csv` file. In this case, start any other MashApp, and then restart the Natural Profiler MashApp to clear the internal MashZone buffer.

Opening the MashApps

After you have specified all required information as described in the previous sections, you can proceed as follows:

1. Invoke MashZone.
2. Open the Natural Profiler MashApp or the Natural Profiler Compare MashApp.

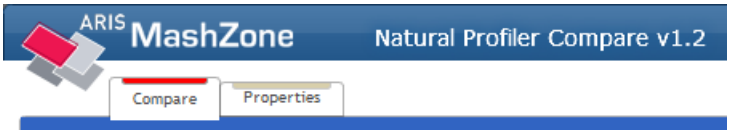
   The Natural Profiler MashApp offers two tabbed pages for analyzing the Profiler event data and viewing the Profiler properties and statistics:



   The **Evaluation page** provides the Profiler event data evaluation.

   The **Properties page** lists the Profiler properties and the statistics of the monitored application.

The Natural Profiler Compare MashApp offers two tabbed pages for comparing the Profiler event data and the Profiler properties and statistics:



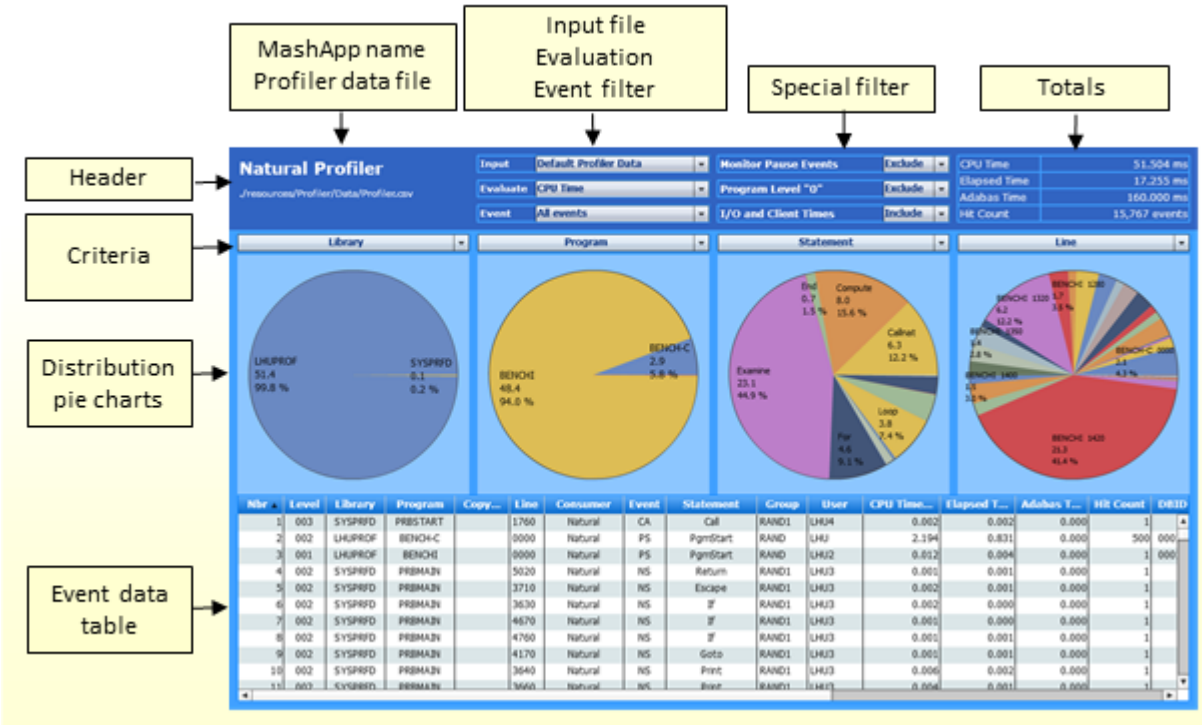
The **Compare page** compares the Profiler event data of two monitored applications.

The **Properties page** lists the Profiler properties and the statistics of the two monitored applications.

The pages are described in the following section.

Evaluation Page

The **Evaluation** page (Natural Profiler MashApp) looks similar to the example below:



The **Evaluation** page is organized in the following sections:

- The **header** at the top of the page with **Input** and KPI selection fields, filters and totals;

- The selection boxes for the distribution criteria and corresponding **distribution pie charts**;
- The **event data table** at the bottom of the page with the consolidated event data.

This section covers the following topics:

Evaluation Header

The header contains the following elements (from left to right and top to bottom):

- The name of the MashApp.
- The path and name of the Profiler data file currently selected.
- The **Input** selection box which is used to select the Profiler data file. The file names listed for selection are taken from the `Description` column in the `Overview.csv` file. See ***Editing the Overview.csv Resource File***. The selected file is used for both pages of the Natural Profiler MashApp.
- The **Evaluate** selection box which is used to select the KPI you want to evaluate in the pie charts. The following KPIs are available:

  CPU Time
  Elapsed Time
  Adabas Command Time
  Hit Count

  The CPU time is evaluated by default. All time values are expressed in milliseconds.
- The **Event** selection box is used to filter the event type you want to evaluate. The event types available for selection depend on the event types collected with the Natural Profiler. The pie charts, the event data table and the totals reflect only the data returned for the selected event types. By default, all event types are evaluated.

  Filtering specific event types is especially useful, for example, to evaluate the hit count of events that seldom occur such as error events.
- The **Monitor Pause Events** selection box is used to filter Monitor Pause events. The filter is valid for the pie charts, the event data table and the totals. By default, the evaluations do not reflect Monitor Pause events. If you include Monitor Pause events, you can see how often monitoring paused, and how long and why it paused.
- The **Program Level "0"** selection box is used to filter events which are executed at Program Level `0`. These events usually relate to the Natural administration rather than the application execution. The filter is valid for the pie charts, the event data table and the totals. By default, the evaluations do not reflect the events at the program level `0`.

- The **I/O and Client Times** selection box is used to filter the I/O time (IB event) and the Natural RPC client time (RW event). These times mainly measure the user reaction (how long it took to press ENTER), especially when the elapsed time for an interactive application is evaluated. They are less relevant for the application performance. The filter is valid for the pie charts, the event data table and the totals. By default, the evaluations reflect the I/O and client times.
- Summarized totals for the CPU time, the elapsed time, the Adabas time and the hit count according to the values that are currently selected in the header and in the pie charts.

Distribution Pie Charts

The **Evaluation** page contains four pie charts. Each pie chart shows the distribution of the KPI (selected in the **Evaluate** selection box) for the criterion selected in the box directly above the pie chart (see the example in *Evaluating Distribution Pie Charts*).

This section covers the following topics:

Criteria for All Event Types

The following criteria are available for all event types:

Consumer

The consumer combines one or more event types into a new criterion. The new criterion depends on the process that consumed the CPU or elapsed time given with the event data. For example, the time returned for a Before Database Call (DB) event is consumed by the database (and therefore belongs to the **Database** consumer), whereas the time returned for an After Database Call (DA) event is consumed by the Natural application (and therefore belongs to the **Natural** consumer).

A consumer evaluation is not relevant for an Adabas time or hit count analysis.

The following consumers are provided:

Consumer	Event Type	Description
Administration	PL, PT	The time Natural used to load and release Natural objects. On the mainframe, the loading of Natural objects from the Natural system file is charged to the **Database** consumer (DB event against FNAT or FUSER system file). On UNIX and Windows, the entire operation is charged to the **Administration** consumer.

Consumer	Event Type	Description
Database	DB	The time consumed for database calls. For the CPU time, it is the time spent in the Natural region.
External	CB	The time spent for external (non-Natural) program calls.
I/O	IB	The time spent for I/O operations. When you analyze the elapsed time of an interactive application, this section shows the user response time. This section is only displayed if **I/O and Client Times** is included in the selection box in the page header.
Pause	MP	The time for which the monitor paused. This section is only displayed if **Monitor Pause Events** is included in the selection box in the page header.
RPC Client	RW	The time spent on the Natural RPC client side. When you analyze the elapsed time of an interactive RPC application, this section shows the user's response time. This section is only displayed if **I/O and Client Times** is included in the selection box in the page header.
RPC Server	RI, RO	The time consumed by the Natural RPC server layer.
Session	SI, ST	The time required to initialize the Natural session.
Natural	CA, DA, E, IA, NS, PR, PS, RS, U	The time Natural spent executing the program code.

Event
The type of the event to be evaluated. All event types are listed in ***Events and Data Collected*** in the section *Using the Profiler Utility*.

Group
The group ID for Natural RPC applications running under Natural Security.

Level
The level at which the profiled program executes.

Library
The Natural library that contains the profiled program.

Line
The source line in which the Natural statement executed by the profiled program is coded.

Line100
Source lines with similar line numbers (rounded down to the next multiple of 100).

Program
The name of the profiled program.

Statement
The Natural statement (for example, EXAMINE) executed in the profiled program.

User
The user ID for Natural RPC applications running under Natural Security.

Criteria for Specific Event Types Only

The following criteria are only available for specific event types. If you select an event-specific criterion, the pie chart will only reflect the data of the related events.

Client User
The Natural RPC client user ID type for RI, RO and RW events.

Command
The Adabas command for DB and DA events.

File

The database ID and file number of the Natural system file for PS and PT events.
The database ID and file number of the Adabas file accessed for DB and DA events.

Return Code

The termination return code for ST events.
The database response and subcode for DA events.
The subprogram response code for CA events.
The error number for E events.
The Natural RPC return code for RI, RO and RW events.

Target Program

The session backend program name for ST events.
The target program name for PL events.
The name of the called subprogram for CB and CA events.
The error handling program name for E events.
The Natural RPC subprogram name for RS events.

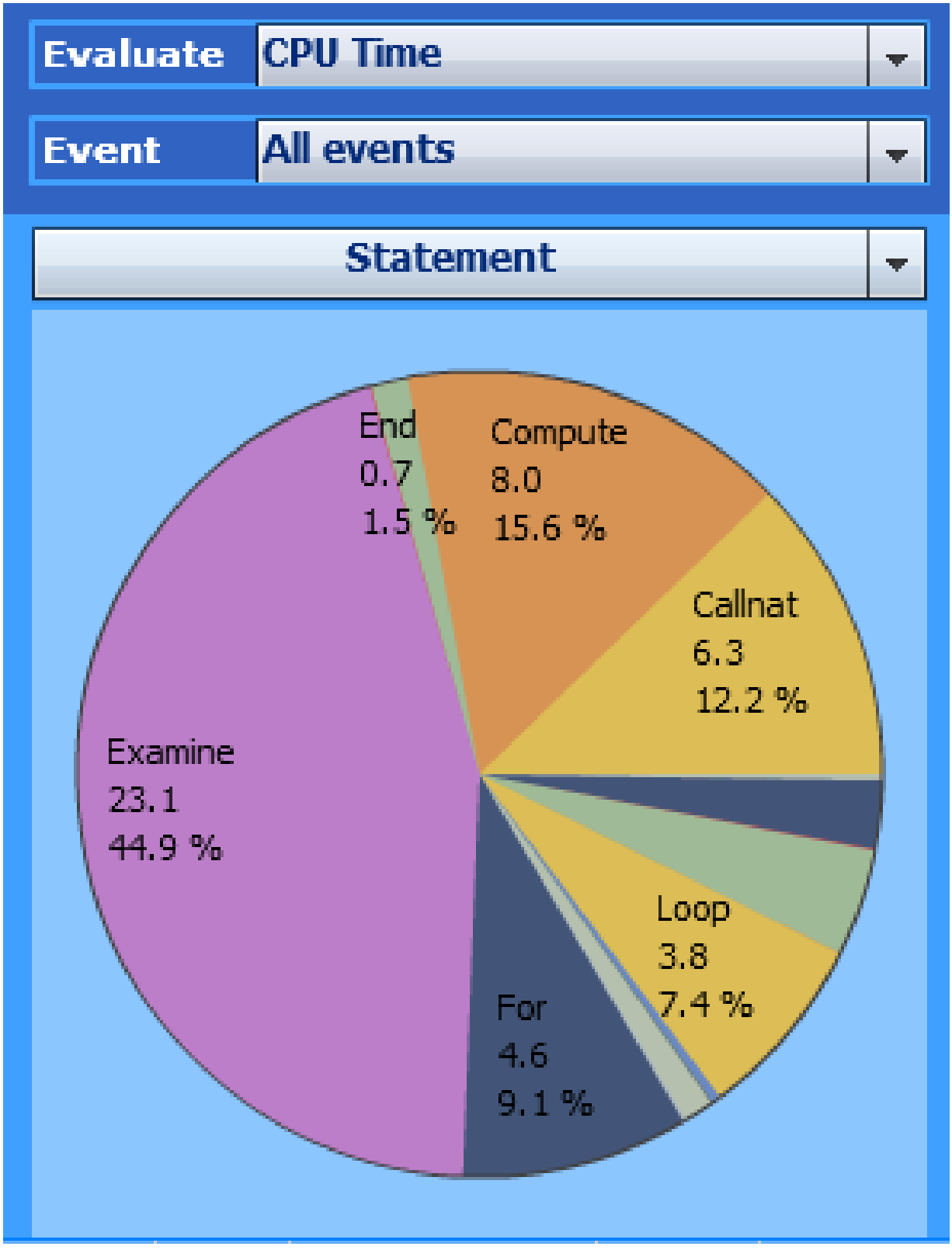
Type

The program type for PS and PT events.
The monitor pause reason for MP events.
The user event subtype for U events.
The return code indicator (system or user) for ST events.

Evaluating Distribution Pie Charts

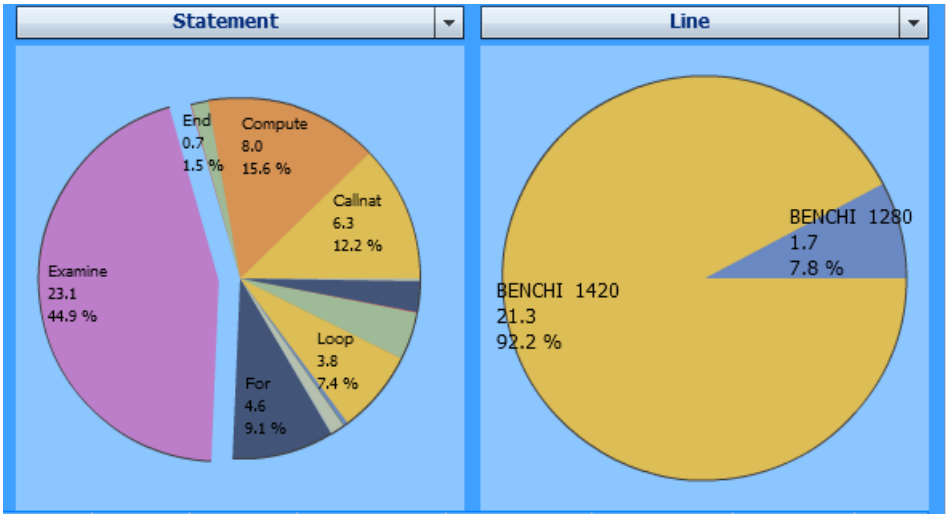
This section describes how you can evaluate distribution pie charts.

- A distribution pie chart shows the distribution for the criterion currently selected in the selection box directly above the pie chart. In the following example, **Statement** has been selected as the criterion for evaluating the CPU time:



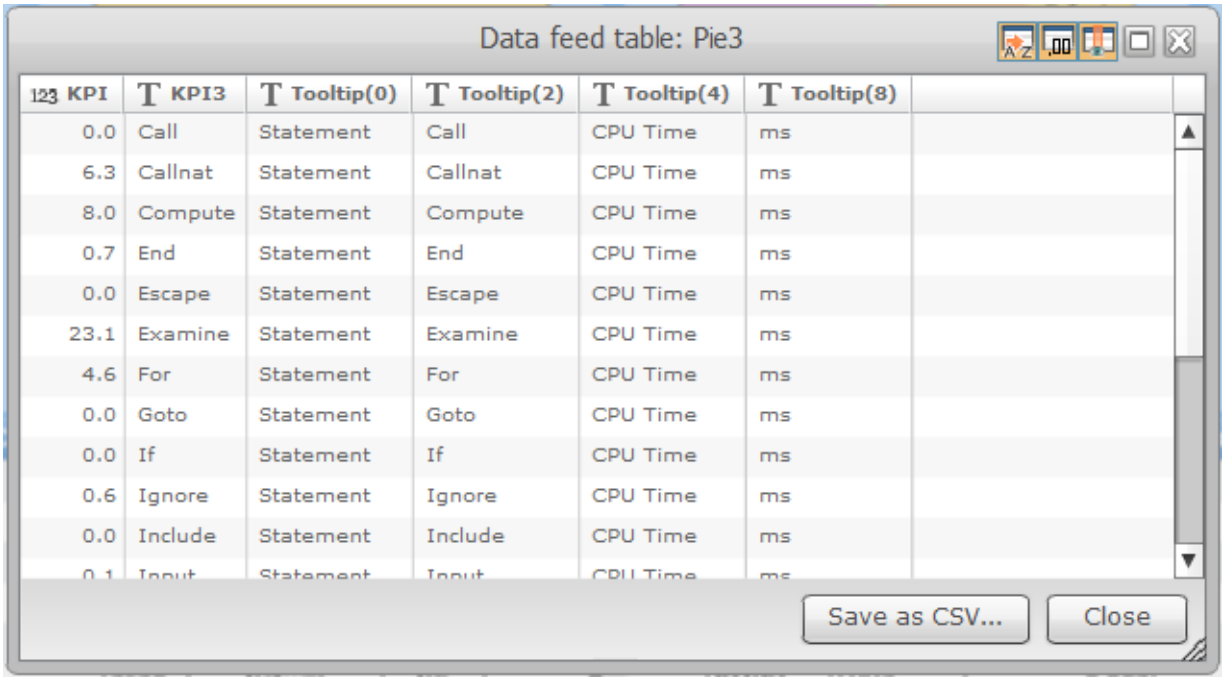
The pie chart shows the distribution of the CPU time for the used Natural statements. It indicates that the **Examine** statement consumed the most CPU time (`23.1` ms / `44.9` percent).

- If you click on a segment in the pie chart, all following pie charts, the event data table and the totals use the selected value as the filter criterion. In the example below, the **Examine** statement in the left pie chart has been selected:



  The right pie chart above displays only those two lines in which an **Examine** statement is executed. The event data table at the bottom of the page and the totals in the page header also reflect the data for the **Examine** statement only.

- To remove a selection, click on the background of a pie chart.
- If you move the cursor to the upper right corner of a pie chart, a drop down list provides the option to save the pie chart as a picture or to display and save the related data. For the display, a window opens with a table containing the data monitored for the Natural statements:



KPI	KPI3	Tooltip(0)	Tooltip(2)	Tooltip(4)	Tooltip(8)
0.0	Call	Statement	Call	CPU Time	ms
6.3	Callnat	Statement	Callnat	CPU Time	ms
8.0	Compute	Statement	Compute	CPU Time	ms
0.7	End	Statement	End	CPU Time	ms
0.0	Escape	Statement	Escape	CPU Time	ms
23.1	Examine	Statement	Examine	CPU Time	ms
4.6	For	Statement	For	CPU Time	ms
0.0	Goto	Statement	Goto	CPU Time	ms
0.0	If	Statement	If	CPU Time	ms
0.6	Ignore	Statement	Ignore	CPU Time	ms
0.0	Include	Statement	Include	CPU Time	ms

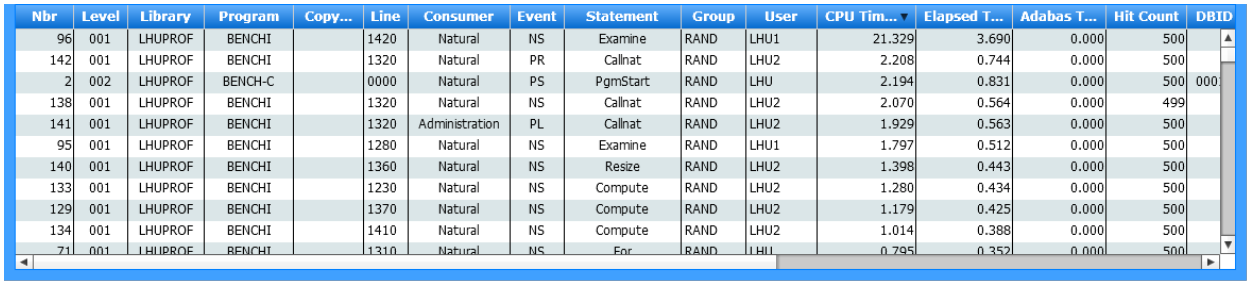
In the example above, the table lists the values of the left pie chart in the previous graphic. The **KPI** column lists the CPU time and the **KPI3** column the corresponding Natural statement.

You can save the table data as a CSV (comma-separated values) formatted file.

Event Data Table

The event data table at the bottom of the **Evaluation** page lists the consolidated Profiler event data according to the values currently selected in the page header and the pie charts. If you click on the table header of a column, the data is sorted by that column.

In the following example, the event data table is sorted by the CPU time (descending):



Nbr	Level	Library	Program	Copy...	Line	Consumer	Event	Statement	Group	User	CPU Tim...	Elapsed T...	Adabas T...	Hit Count	DBID
96	001	LHUPROF	BENCHI		1420	Natural	NS	Examine	RAND	LHU1	21.329	3.690	0.000	500	
142	001	LHUPROF	BENCHI		1320	Natural	PR	Callnat	RAND	LHU2	2.208	0.744	0.000	500	
2	002	LHUPROF	BENCH-C		0000	Natural	PS	PgmStart	RAND	LHU	2.194	0.831	0.000	500	000
138	001	LHUPROF	BENCHI		1320	Natural	NS	Callnat	RAND	LHU2	2.070	0.564	0.000	499	
141	001	LHUPROF	BENCHI		1320	Administration	PL	Callnat	RAND	LHU2	1.929	0.563	0.000	500	
95	001	LHUPROF	BENCHI		1280	Natural	NS	Examine	RAND	LHU1	1.797	0.512	0.000	500	
140	001	LHUPROF	BENCHI		1360	Natural	NS	Resize	RAND	LHU2	1.398	0.443	0.000	500	
133	001	LHUPROF	BENCHI		1230	Natural	NS	Compute	RAND	LHU2	1.280	0.434	0.000	500	
129	001	LHUPROF	BENCHI		1370	Natural	NS	Compute	RAND	LHU2	1.179	0.425	0.000	500	
134	001	LHUPROF	BENCHI		1410	Natural	NS	Compute	RAND	LHU2	1.014	0.388	0.000	500	
71	001	LHUPROF	BENCHI		1310	Natural	NS	For	RAND	LHU	0.795	0.352	0.000	500	

Compare Page

The **Compare** page (Natural Profiler Compare MashApp) compares the Profiler event data of two monitored applications as shown in the following example:

The **Compare** page is organized in the following sections:

- The header at the top of the page with **Input** and KPI selection fields, filters and totals.
- The column chart comparing the values of the two monitored applications:

  Values for the first application are shown in the left (green) column, values for the second application are shown in the right (yellow) column.

This section covers the following topics:

- Compare Header

- Compare Column Chart

Compare Header

The **Compare** header contains the following elements (from left to right and top to bottom):

- The name of the MashApp.
- The paths and names of the two Profiler data files to be compared.
- The **Input 1** and **Input 2** selection boxes with the Profiler data files selected for comparison. The file names listed for selection are taken from the `Description` column in the `Overview.csv` file (see *Editing the Overview.csv Resource File*). The selected files are used for both pages of the Natural Profiler Compare MashApp.
- Summarized totals for the CPU time, the elapsed time, the Adabas time and the hit count according to the values for both applications listed in the header and column chart.
- The **Evaluate** selection box with the KPI to evaluate in the column chart (see *Evaluation Header*).
- The **Event** selection box with the event type to evaluate for (see *Evaluation Header*).
- The **Criterion** selection box with the filter criterion to use for the KPI distribution. The criteria available for selection correspond to the criteria for the distribution pie charts on the **Evaluation** page (see *Evaluating Distribution Pie Charts*.
- The **Monitor Pause Events** selection box with the filter criterion to use for Monitor Pause events (see *Evaluation Header*).
- The **Program Level "0"** selection box with the filter criterion to use for events at the program level `0` (see *Evaluation Header*).
- The **I/O and Client Times** selection box with the filter criterion to use for I/O time (`IB` event) and Natural RPC client time (`RW` event) events; see *Evaluation Header*.
- The **Pre-Selection** values with restrictions for the column chart values to a specific criterion instance, for example to a specific library.

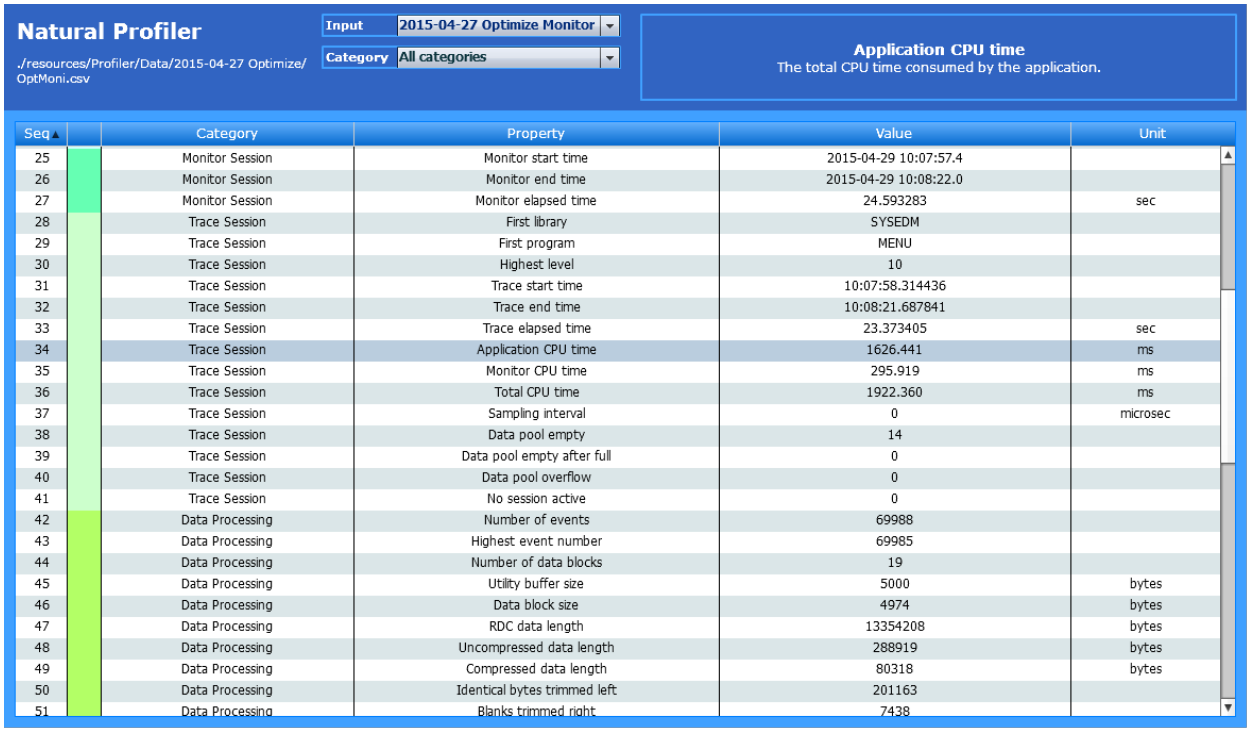
Compare Column Chart

The **Compare** column chart compares the values of the KPI (selected in the **Evaluate** selection box) for the criterion specified in the **Criterion** selection box for both profiled applications.

In the example of a **Compare page** shown earlier, the CPU time (**Evaluate** selection box) of each program (**Criterion** selection box) executed by `Numeric Operations MF` (**Input 1**) is compared with the corresponding time of `Numeric Operations LUW` (**Input 2**). Additionally, a pre-selection has been specified so that only values from the library `PRFDEMO` are considered. The green columns show the CPU times of `Numeric Operations MF`, the yellow columns the CPU times of `Numeric Operations LUW`.

Properties Page

The **Properties** page lists the Profiler properties and the statistics of the monitored application as shown in the following example:



Seq		Category	Property	Value	Unit
25		Monitor Session	Monitor start time	2015-04-29 10:07:57.4	
26		Monitor Session	Monitor end time	2015-04-29 10:08:22.0	
27		Monitor Session	Monitor elapsed time	24.593283	sec
28		Trace Session	First library	SYSEDM	
29		Trace Session	First program	MENU	
30		Trace Session	Highest level	10	
31		Trace Session	Trace start time	10:07:58.314436	
32		Trace Session	Trace end time	10:08:21.687841	
33		Trace Session	Trace elapsed time	23.373405	sec
34		Trace Session	Application CPU time	1626.441	ms
35		Trace Session	Monitor CPU time	295.919	ms
36		Trace Session	Total CPU time	1922.360	ms
37		Trace Session	Sampling interval	0	microsec
38		Trace Session	Data pool empty	14	
39		Trace Session	Data pool empty after full	0	
40		Trace Session	Data pool overflow	0	
41		Trace Session	No session active	0	
42		Data Processing	Number of events	69988	
43		Data Processing	Highest event number	69985	
44		Data Processing	Number of data blocks	19	
45		Data Processing	Utility buffer size	5000	bytes
46		Data Processing	Data block size	4974	bytes
47		Data Processing	RDC data length	13354208	bytes
48		Data Processing	Uncompressed data length	288919	bytes
49		Data Processing	Compressed data length	80318	bytes
50		Data Processing	Identical bytes trimmed left	201163	
51		Data Processing	Blanks trimmed right	7438	

The **Properties** page of the Natural Profiler Compare MashApp lists the properties and statistics of both Profiler data files selected for comparison.

The **Properties** page is organized in the following sections:

- The properties header at the top of the page with **Input** and **Category** selection fields and the property description;
- The properties table with the properties and statistics.

This section covers the following topics:

- Properties Header

- Properties Table

Properties Header

The header contains the following elements (from left to right and top to bottom):

- The name of the MashApp.
- The path and name of the Profiler data file currently selected.
- The **Input** selection box is used to select the Profiler data file. The names listed for selection are taken from the `Description` column in the `Overview.csv` file. See *Editing the Overview.csv Resource File*. The selected file is used for both pages of the Natural Profiler MashApp.
- The **Category** selection box is used to select a category (listed alphabetically). The selection box only offers the categories for which at least one associated property is found in the Profiler data file.

  If you select a category, the table shows the properties of the selected category only. By default, all categories are displayed. The following categories are available:

Category	Description
Data Consolidation	Statistics of the data consolidation such as the consolidation factor
Data Processing	Statistics of the data processing, data compression and data transfer such as the number of events and the compression rate
Event Type Statistics	Statistics of the event types such as the number of Program Load events
General Info	Information related to the environment and the Natural Profiler such as the internal Profiler version
Monitor Pause Statistics	Statistics of Monitor Pause events such as the number of Profiler data pool full situations
Monitor Session	Statistics of the Profiler monitor session such as the monitor elapsed time
Profiler Resource File	Information related to the Profiler resource file such as the resource name and library
Trace Session	Statistics of the Profiler trace session including the application execution such as the CPU time of the total session

- The **Property** description. If you click on a line in the properties table, the name of the corresponding property and a detailed description of it are displayed in the page header.

Properties Table

The properties table lists all collected Profiler properties and application statistics. If you click on an entry in the table header of a column, the entire table is sorted by this column. Each color in the second column corresponds to one category.

All Profiler categories and properties are described in detail in the section ***Profiler Statistics***.

Use Cases

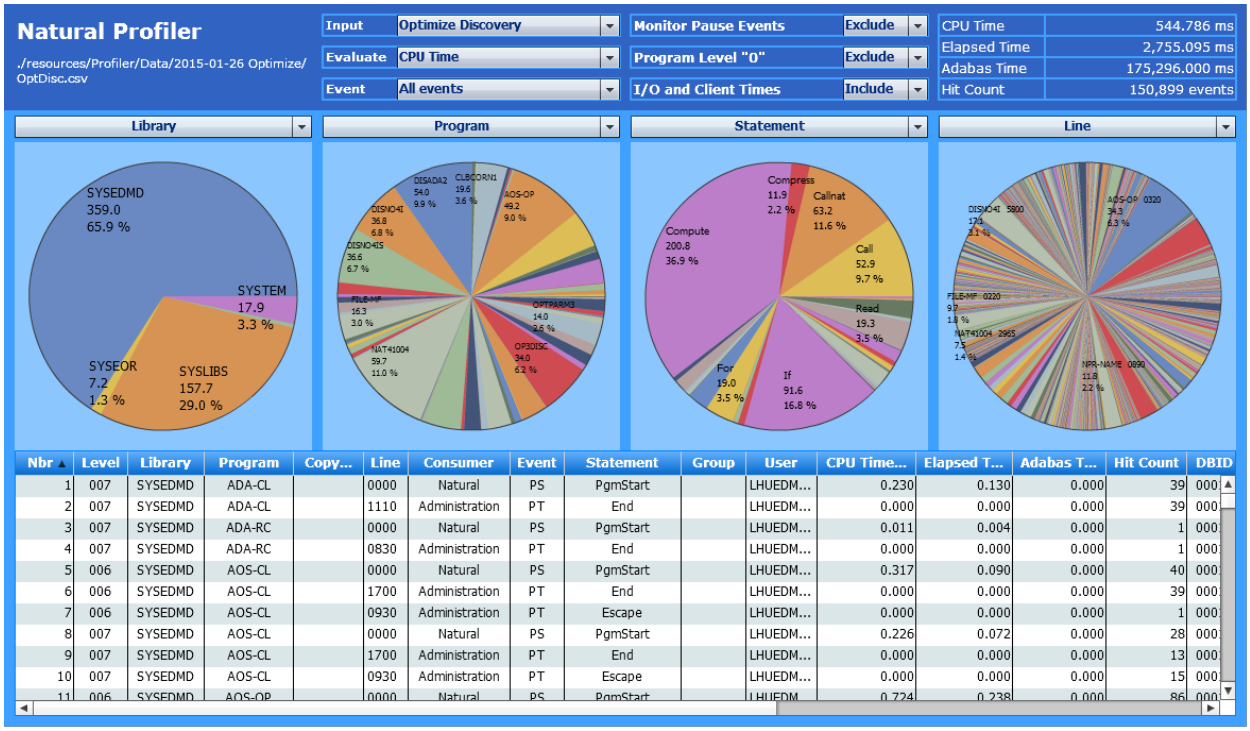
This section describes the following use cases:

- Application Performance Analysis
- Combined Line Numbers
- Consumer
- Natural RPC Server Evaluation
- Natural RPC Server Statistics
- Adabas Command Time Analysis
- Adabas Statistics
- Application Statistics

Application Performance Analysis

By default, the Natural Profiler MashApp is set up to create CPU time performance analyses of libraries, programs, statements and source lines.

Each pie chart in the example below shows the distribution of the CPU time for each criterion selected:



You can immediately see which library, program, statement or line has consumed how much of the CPU time.

The example above uses the following selections:

Evaluate: CPU Time

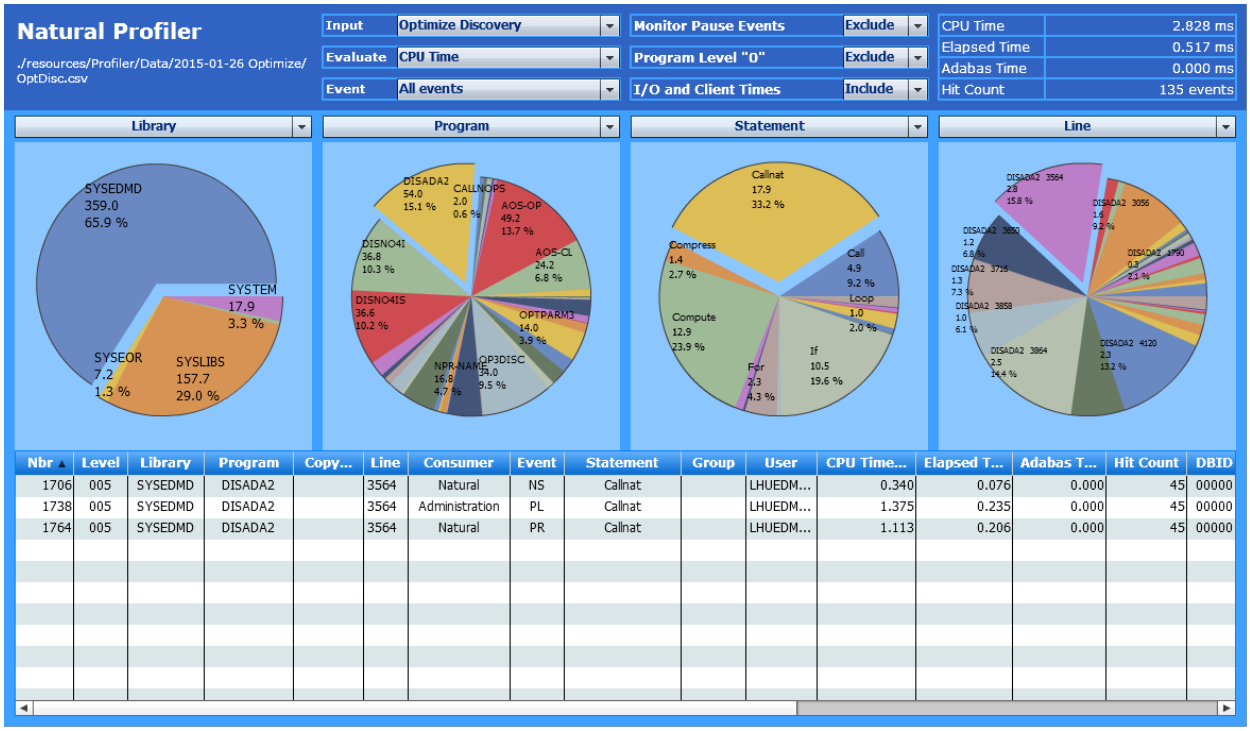
Event: All events

Criteria: Library, Program, Statement, Line

A large application, such as the example above, references many program lines, thus making it difficult to analyze the corresponding pie chart.

If you click on a segment in a pie chart, the corresponding value of that segment is used as a filter and the amount of data which is displayed in the following pie charts is reduced accordingly. In the example above, a click on the segment with the `SYSEDMD` library in the leftmost pie chart would change the contents of the other three pie charts and only show the programs, statements and lines executed in the `SYSEDMD` library.

The following example refers to the previous one and assumes that in addition to the `SYSEDMD` library, the program `DISADA2`, the `Callnat` statement and the line `3564` are selected in the rightmost chart:

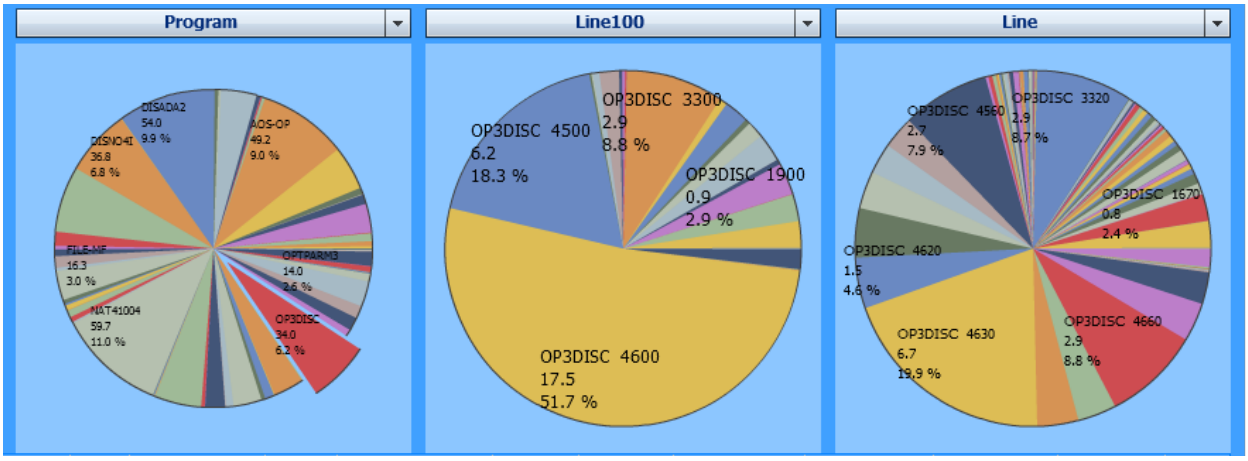


The event data table and the total in the page header now only refer to Line `3564` where the program executes the `CALLNAT` statement. The `CALLNAT` statement caused the event types (`NS`, `PL` and `PR`), each executing `45` times which results in a total **Hit Count** of `135` events.

Combined Line Numbers

The **Line100** criterion is another approach to reduce the number of entries in the line number chart. It replaces the lines by the previous multiples of 100, thus, combining lines with similar line numbers in one segment of the pie chart.

The example below assumes that you want to find out which part of the program `OP3DISC` consumed the most CPU time. Therefore, you select `OP3DISC` in the **Program** chart so that all other charts only display the data for this program:



The **Line** chart clearly indicates that the statement in the segment of line `4630` uses `19.9` percent of the program's CPU time. However, all other segments are rather small and it is difficult to tell them apart.

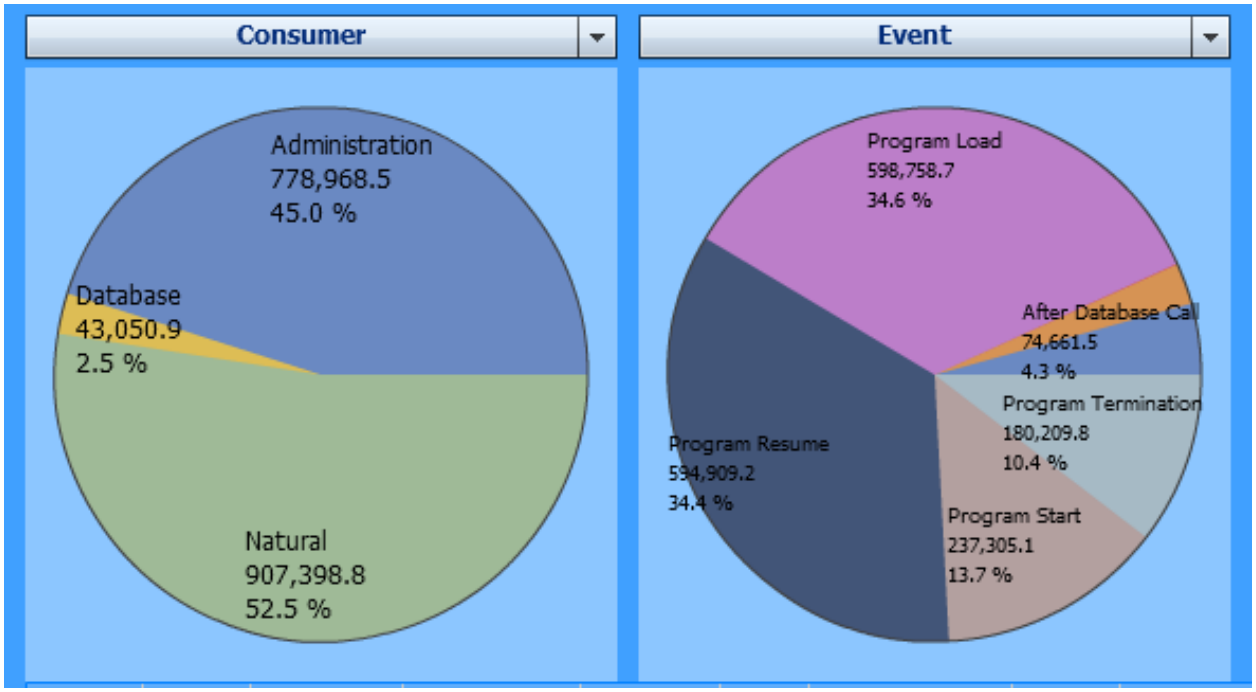
The **Line100** chart shows that more than half of the time was consumed by the statements in the lines ranging from 4600 through 4690. Additionally, considering the statements in the lines ranging from 4500 through 4590, this part of the program even consumes 70 percent of the entire execution time. Thus, this program is most busy with the statements in these lines.

The example above uses the following selections:

Evaluate:	CPU Time
Event:	All events
Criteria:	Program, Line, Line100

Consumer

The **Consumer** analysis gives a quick overview of the processes that consumed the most CPU time such as external programs, database calls, I/O operations, administration tasks or program instructions. For example:



In the example above (Natural for UNIX, without statement events), 45 % of the CPU time was consumed by administration tasks. A potential reason for this can be the usage of small subprograms which solely call other tiny subprograms. This keeps Natural busy with administration tasks (program load with buffer pool management and program termination), while the time used for executing the code itself is relatively short.

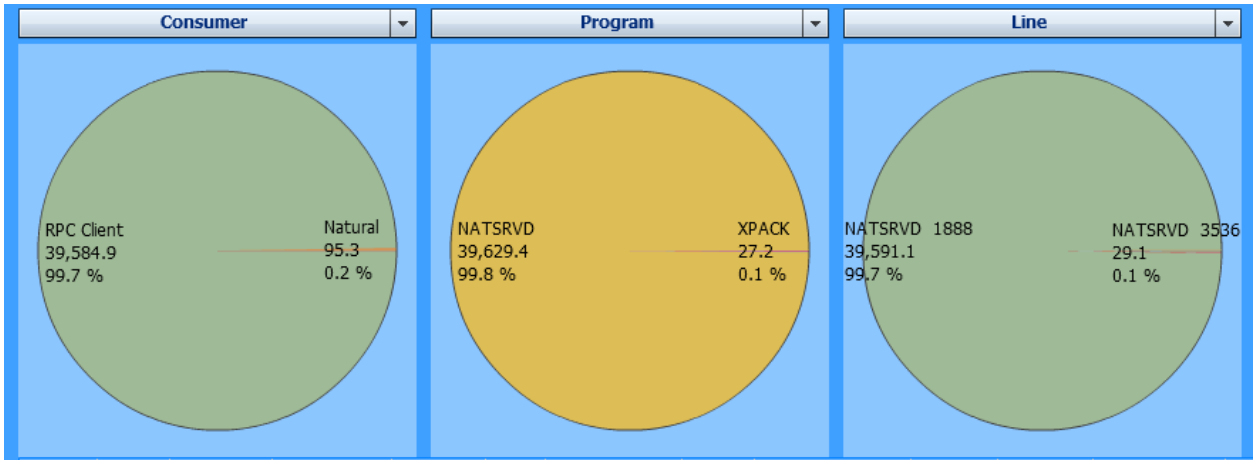
The example above uses the following selections:

Evaluate:	CPU Time
Event:	All events
Criteria:	Consumer, Event

Natural RPC Server Evaluation

When analyzing the elapsed time of an interactive application, waiting for a user response usually takes the most time. For a Natural RPC application, this time is monitored with the RPC Wait for Client (RW) event or the RPC Client consumer.

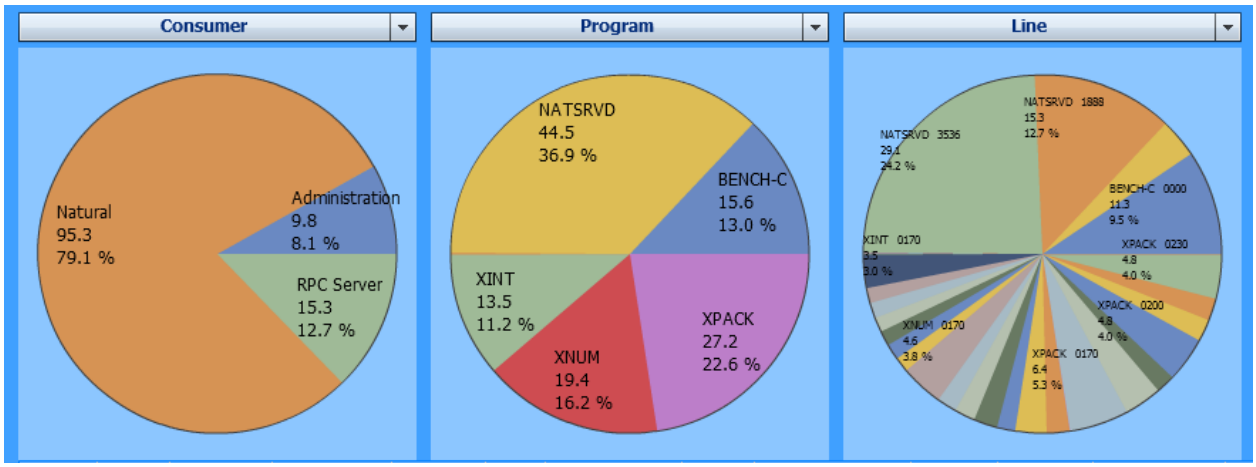
In the example below, the Natural RPC client consumes nearly all of the elapsed time:



The example above uses the following selections:

Evaluate:	Elapsed Time
Event:	All events
I/O and Client Times:	Include
Criteria:	Consumer, Program, Line

The MashApp offers a selection field to exclude the client time. If you exclude **I/O and Client Times**, all individual processes performed in the server application are shown similar to the example below:



The example above uses the following selections:

Evaluate:	Elapsed Time
Event:	All events
I/O and Client Times:	Exclude

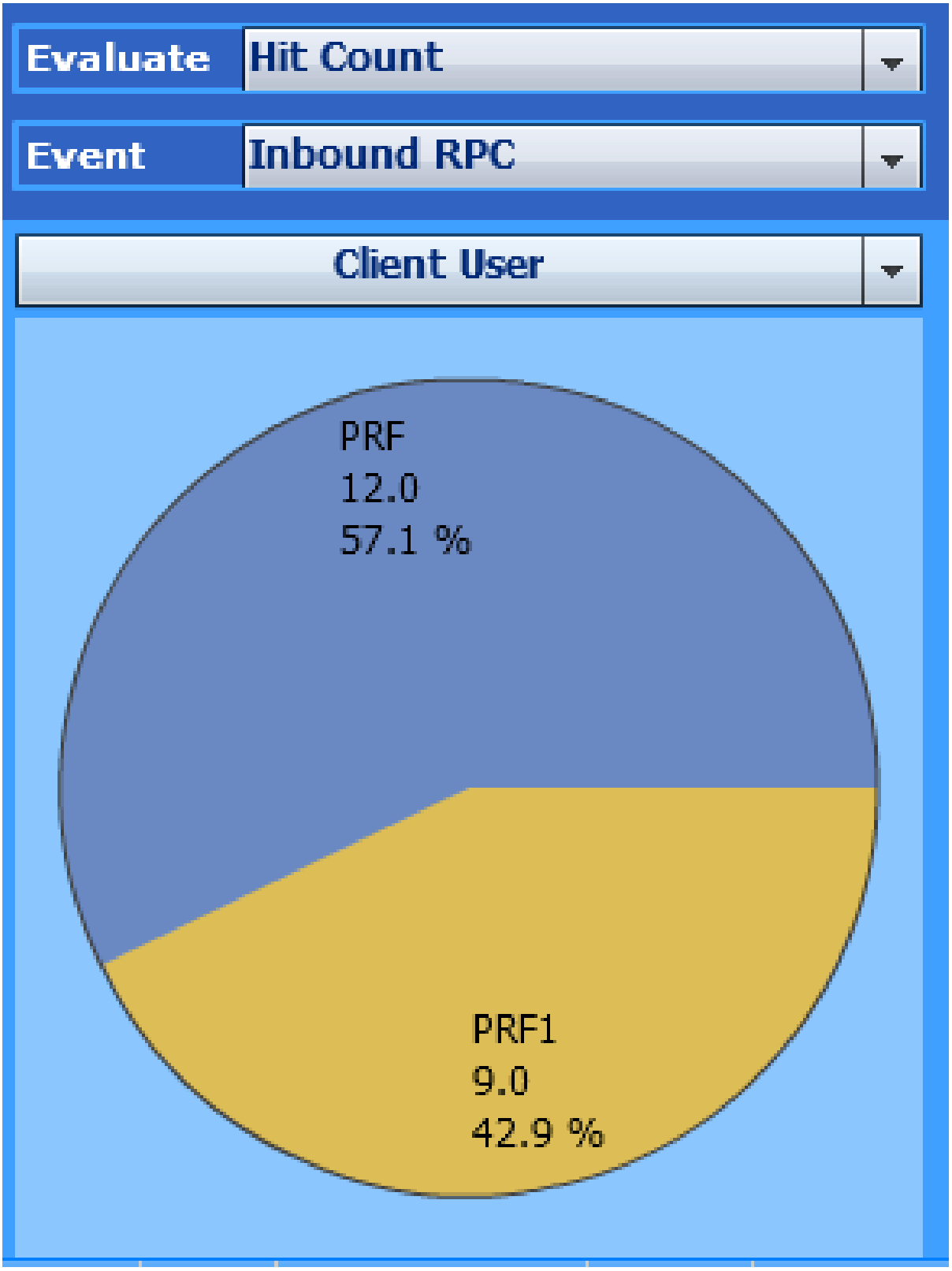
Criteria: Consumer, Program, Line

Natural RPC Server Statistics

You can obtain statistics on remote procedure calls by evaluating the hit count.

Example of a Natural RPC Client User Evaluation

The following example shows which user issued Natural RPC requests and how often:



In the example above, the user `PRF` issued `12` Natural RPC requests.

The example uses the following selections:

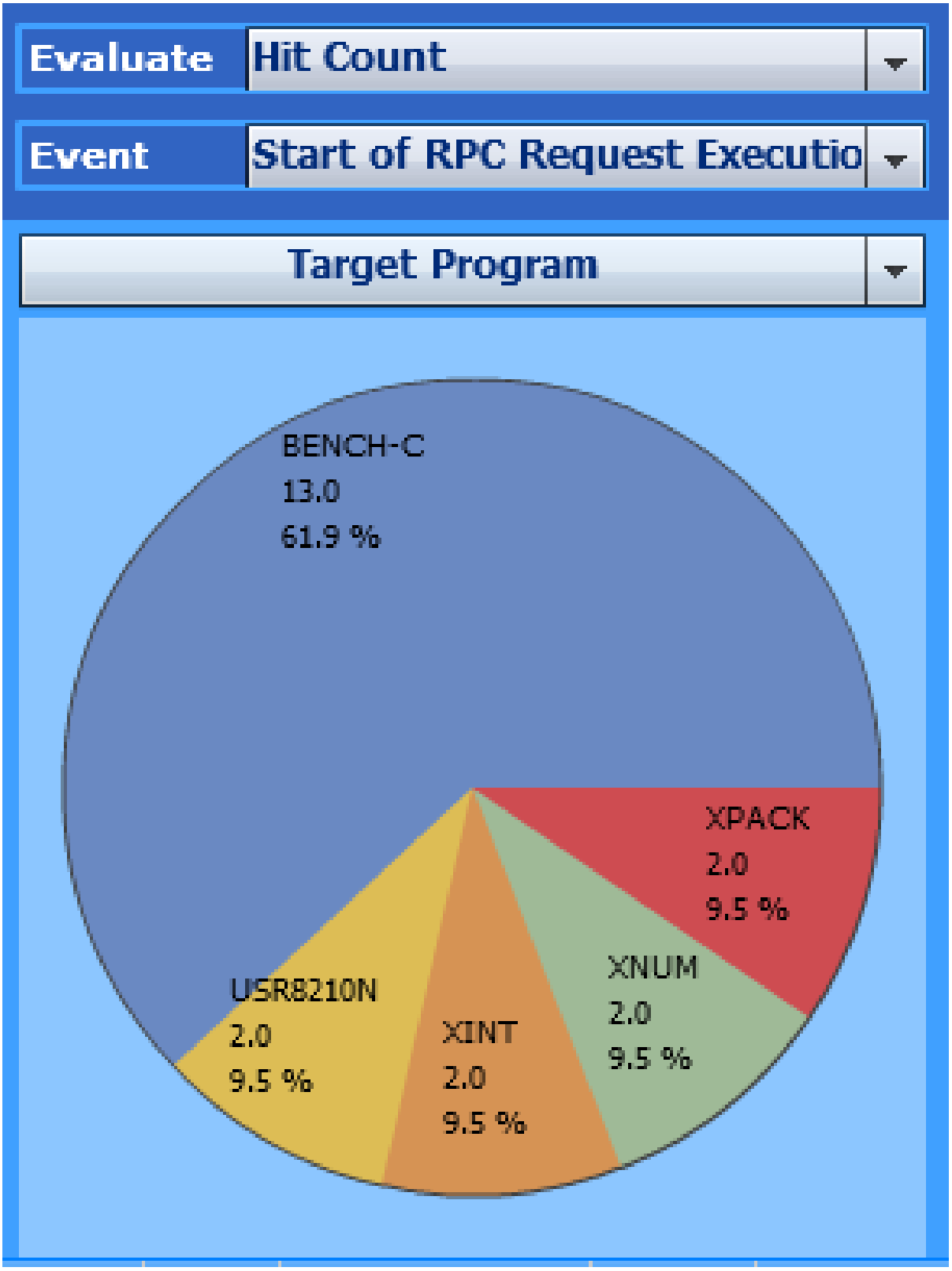
Evaluate: Hit Count

Event: Inbound RPC

Criterion: Client User

Example of a Natural RPC Target Program Evaluation

The following example displays which target program was called on the server and how often:



In the example above, `13` Natural RPC requests were issued for the server program `BENCH-C`.

The example uses the following selections:

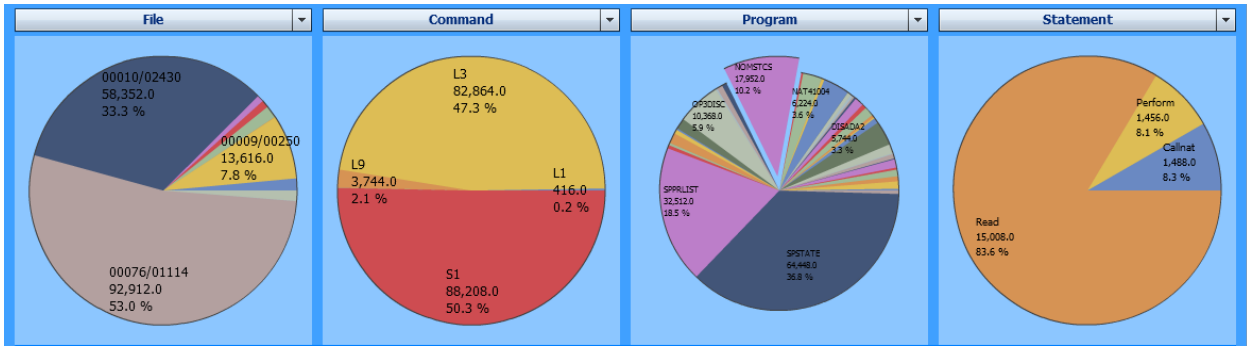
Evaluate: Hit Count
Event: Start of RPC Request Execution
Criterion: Target Program

In both examples shown above, single event types are used for the event selection so not to mix the data with other events. For example, if **All events** is selected, the target programs of external program calls (`CA` events) are also displayed in the chart.

Adabas Command Time Analysis

When the Natural application issues an Adabas command, the database returns the elapsed time the Adabas nucleus required to process the command.

The example below analyzes the distribution of the Adabas command time for the accessed files and for the used Adabas commands. The chart also shows the programs and Natural statements that consumed the Adabas command time.



The most Adabas command time was consumed by calls against the file `1114` of the database `76` and by the Adabas commands `S1` (find record) and `L3` (read logical sequential record).

If you could click on a segment in the pie chart below **File**, you would see the commands issued against the selected file and how much time they consumed. Since the segment of the program `NOMSTCS` is selected in the third pie chart, the fourth pie chart only shows the Adabas command time used by the statements in `NOMSTCS`.

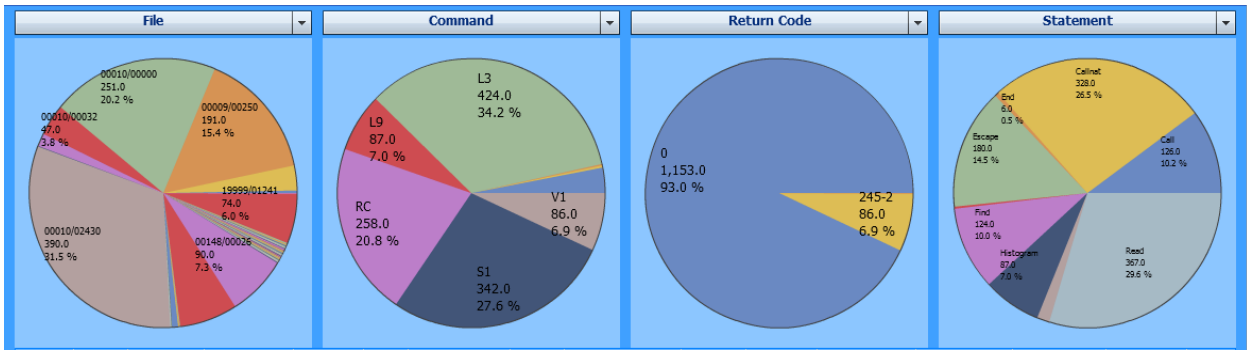
The example uses the following selections:

Evaluate: Adabas Command Time
Event: All events
Criteria: File, Command, Program, Statement

Adabas Statistics

You can obtain statistics on Adabas requests by evaluating the hit count.

The following example shows which files have been accessed, which commands have been issued, which Adabas response codes have occurred and which Natural statements have issued Adabas requests and how often:



The most Adabas requests were issued against the file `2430` of database `10` and the most frequently Adabas command used was `L3` (read logical sequential record). Most calls were successful (Adabas response 0) but `86` calls received an Adabas response `245` with subcode `2`. The fourth pie chart shows that `READ` statements issued `367` Adabas calls.

The example uses the following selections:

Evaluate:	Hit Count
Event:	After Database Call
Criteria:	File, Command, Return Code, Statement

Application Statistics

The following Profiler MashApp examples may answer common statistics questions about monitored Natural applications.

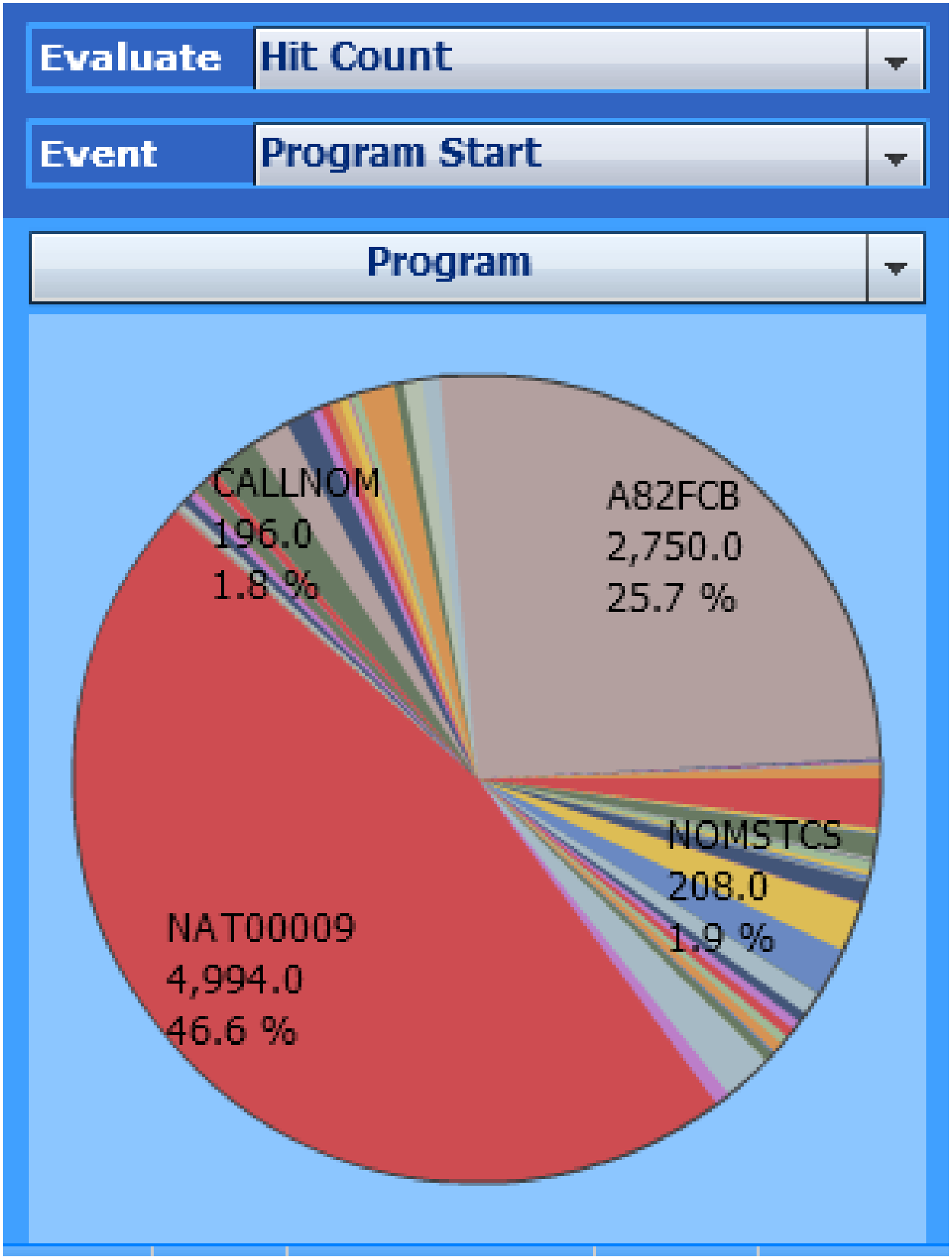
- How often were Natural objects started?
- How many statements were executed in the monitored programs?
- Which objects were called by a selected program and how often?
- How often was a selected object called?

- How many runtime errors occurred?

How often were Natural objects started?

Use the following selections to find out:

Evaluate:	Hit Count
Event:	Program Start
Criterion:	Program

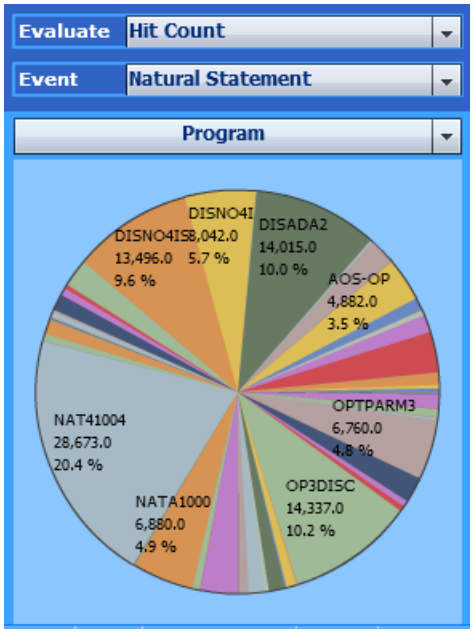

Evaluate
Hit Count
Event
Program Start
Program
CALLNOM
196.0
1.8 %
A82FCB
2,750.0
25.7 %
NOMSTCS
208.0
1.9 %
NAT00009
4,994.0
46.6 %

In the example above, the program NAT00009 started 4,994 times.

How many statements were executed in the monitored programs?

Use the following selections to find out:

Evaluate: Hit Count
Event: Natural Statement
Criterion: Program

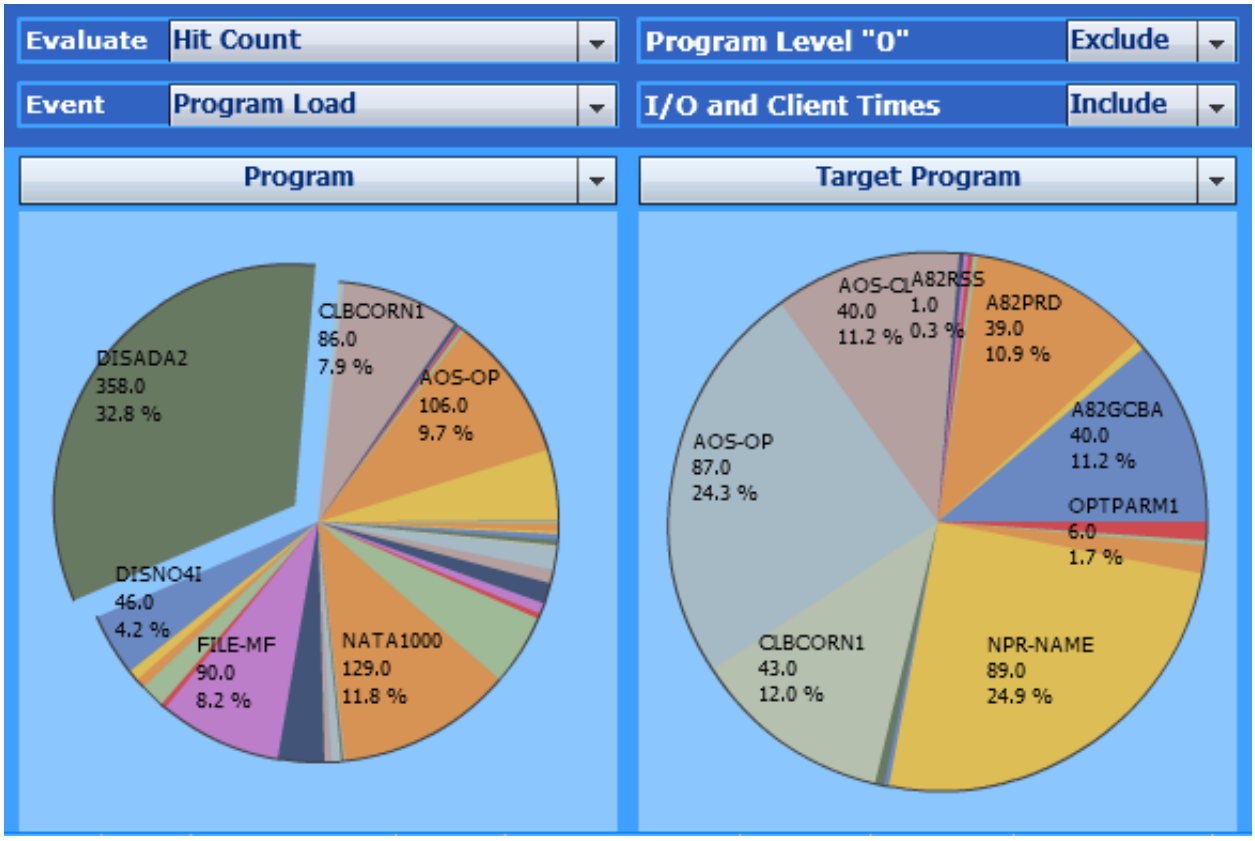


In the example above, the program NAT41004 executed 28,673 statement events.

Which objects were called by a selected program and how often?

Use the following selections to find out:

Evaluate: Hit Count
Event: Program Load
Criteria: Program, Target Program

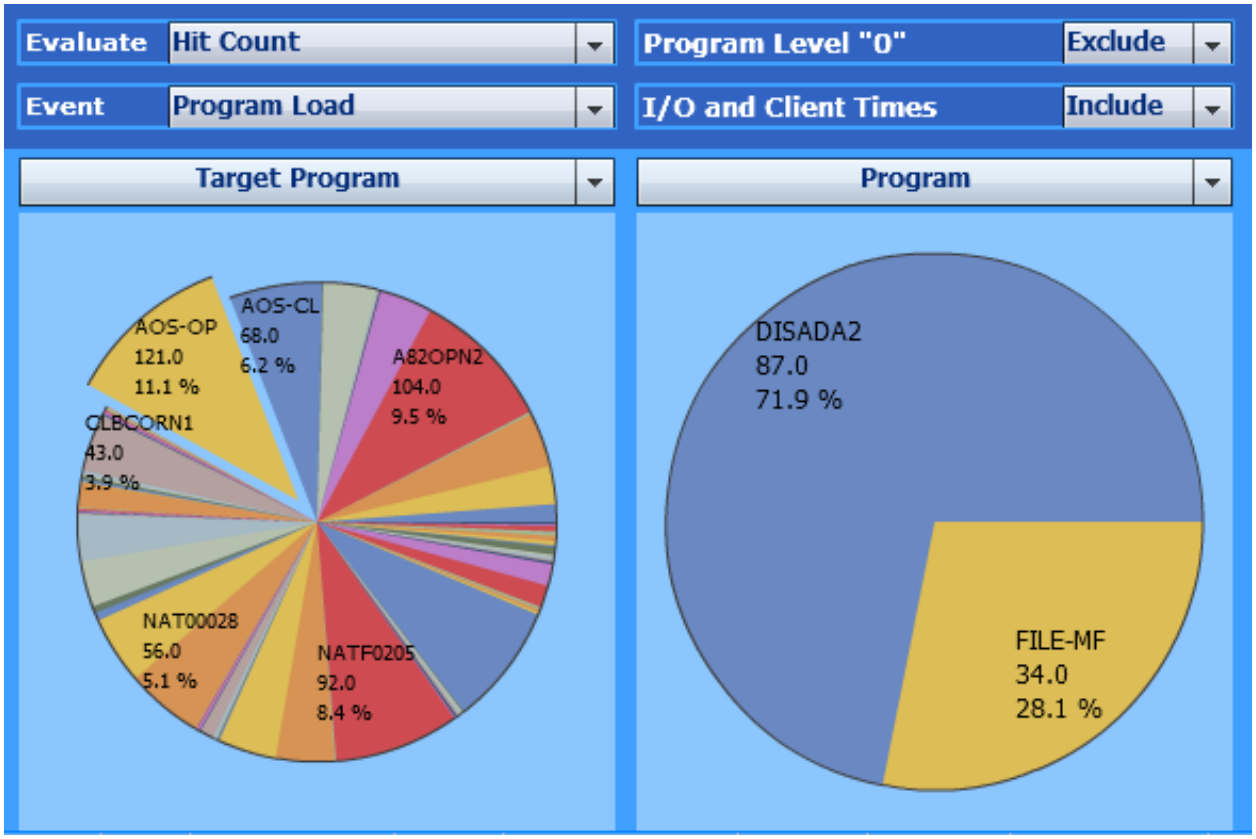


In the example above, the right pie chart shows the Natural objects called by `DISADA2`. The Natural object `NPR-NAME` was called `89` times.

How often was a selected object called?

Use the following selections to find out:

Evaluate:	Hit Count
Event:	Program Load
Criteria:	Target Program, Program

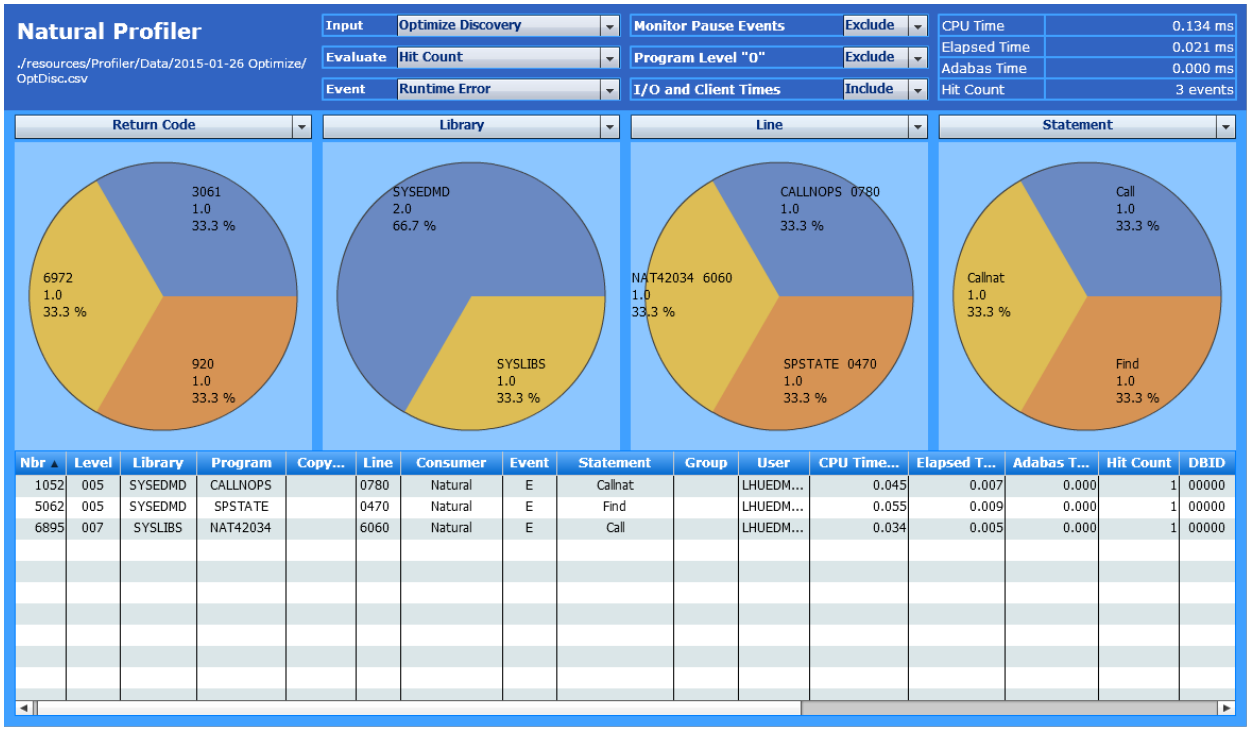


In the example above, the right pie chart shows the objects which called the program `AOS-OP`. `AOS-OP` was called `87` times by the program `DISADA2` and `34` times by the program `FILE-MF`.

How many runtime errors occurred?

Use the following selections to find out:

Evaluate:	Hit Count
Event:	Runtime Error
Criteria:	Return Code, Library, Line, Statement



In the example above, the **Hit Count** in the page header indicates that three runtime errors occurred during application execution. The charts show which errors occurred, the library, program and line where they occurred, and the statements that caused the errors.

XVII SYSRPC Utility

The utility SYSRPC is used to maintain remote procedure calls on the client side.

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 RPC (Remote Procedure Call)* 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 RPC (Remote Procedure Call)* documentation.
- For detailed information regarding EntireX Broker features and components, refer to the appropriate *EntireX Broker* documentation.

78 Basic SYSRPC Functions

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**.

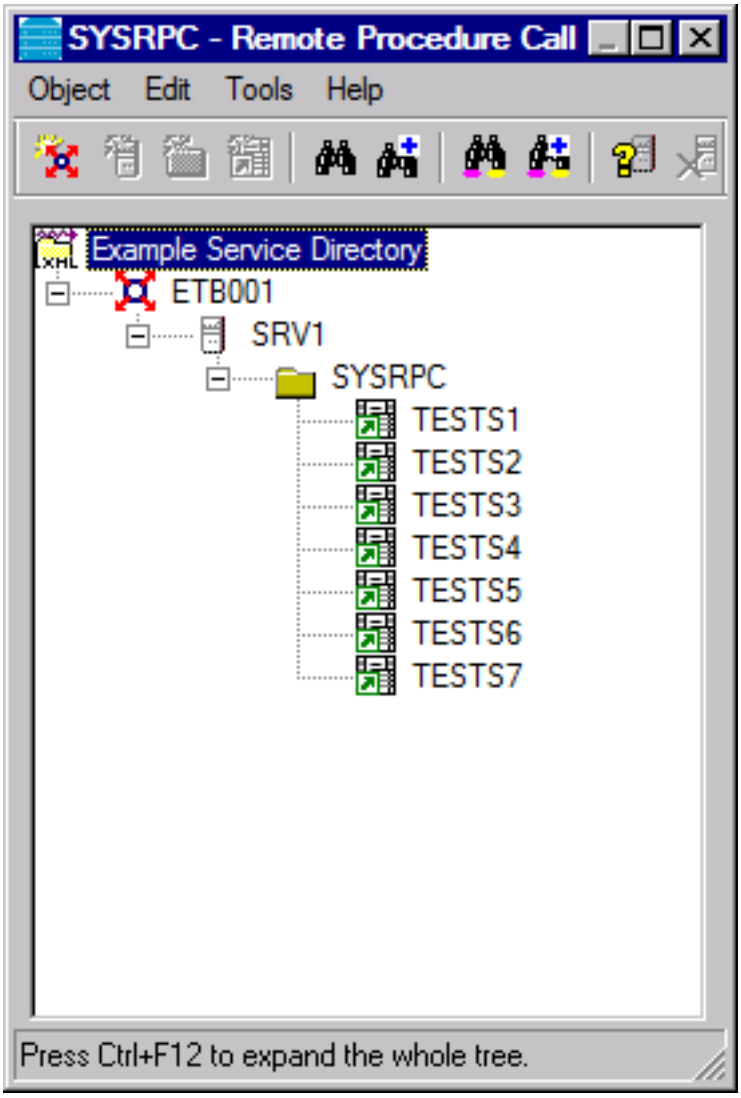
Or:

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 RPC (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.

<table>
<tr><th>Menu</th><th>Menu Item and Shortcut</th><th>Explanation</th></tr>
<tr><td rowspan="5">Object</td><td>Open</td><td>Opens a service directory:

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).

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.</td></tr>
<tr><td>Save
CTRL+S</td><td>Saves the service directory in the current library.</td></tr>
<tr><td>Save As</td><td>Saves the service directory in another library.</td></tr>
<tr><td>Properties</td><td>Invokes the Properties of Service Directory dialog box. It provides information on the generation of the service directory:

Object The name of the service directory.
Generated by The name of the Natural utility.
Environment If the directory was generated in a local environment, the value local is displayed. Otherwise, the name of a remote server is displayed.
Library The name of the library in which the service directory was generated.
User The ID of the user who generated the service directory.
Time stamp The date and the time at which the service directory was last modified.
Tree view nodes The number of tree nodes including the service directory root node.
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.</td></tr>
<tr><td>Exit
CTRL+F4</td><td>Terminates SYSRPC.</td></tr>
<tr><td>Edit</td><td>New Item</td><td>Creates a new tree node item. Depending on the tree node selected, you have the following choices:

Node Corresponding toolbar button: 
RPC Server Corresponding toolbar button:  </td></tr>
</table>

Menu	Menu Item and Shortcut	Explanation
		Library — Corresponding toolbar button:  **Service (Subprogram)** — Corresponding toolbar button:  See also the section *Natural RPC Terminology* in the *Natural RPC (Remote Procedure Call)* for explanations of relevant expressions.
	Rename CTRL+F2	Modifies the name of a tree node.
	Delete	Removes a tree node.
	Cut CTRL+X **Copy** CTRL+C **Paste** CTRL+V	Cuts, copies or pastes a tree node.
	Find CTRL+F	Invokes the **Find Item** window to search for a name: **Find** — Enter an alphanumeric search string of up to 32 characters. **Case sensitive** — Select this check box to distinguish between uppercase and lowercase characters. **Whole words only** — Select this check box to search for complete character strings only. Choose **OK** to start searching and move to the first hit, which is highlighted. Corresponding toolbar button: 
	Find Next F3	Searches for additional instances of the search string specified in the **Find Item** window and moves to the next hit if one exists. Corresponding toolbar button: 
	Replace CTRL+H	Invokes the **Replace Item** window to replace a name: **Find** — Enter an alphanumeric search string of up to 32 characters. **Case sensitive** — Select this check box to distinguish between uppercase and lowercase characters. **Whole words only** — Select this check box to search for complete character strings only. **Replace all** — Select this check box to replace all search strings found.

Menu	Menu Item and Shortcut	Explanation
		Choose **OK** to start searching and replacing the specified character strings. Corresponding toolbar button: 
	Replace Next CTRL+F3	Searches for additional instances of the search string specified in the **Replace Item** window and moves to the next hit if one exists. Corresponding toolbar button: 
Tools	**Ping** CTRL+F9	Sends an internal message to verify server connections described in the section *Server Command Execution*. Corresponding toolbar button:  See also *Pinging an RPC Server*.
	Terminate Server	Sends an internal message to terminate server connections as described in the section *Server Command Execution*. See also *Terminating an RPC Server*.
	Terminate Server with Sequence Number CTRL+F10	Sends a shutdown request with sequence number to the EntireX Broker as described in the section *Server Command Execution*. See also *Terminating an RPC Server*.
	Terminate EntireX Broker Service	Requests termination of an EntireX Broker service as described in the section *Server Command Execution*. Corresponding toolbar button: 
	Single Interface Object Generation CTRL+F8	Generates single interface objects as described in *Generating Single Interface Objects with Parameter Specification* in the section *Generating Interface Objects*.
	Interface Object Mass Calculation CTRL+F5	Performs calculations for size requirements of RPC calls as described in the section *Calculating Size Requirements*.
	Interface Object Mass Generation CTRL+F6	Generates single or multiple interface objects online or batch as described in *Using the Interface Object Mass Generation Function* in the section *Generating Interface Objects*.
	Interface Object Generation from IDL CTRL+F7	Generates interface objects or parameter data areas (PDAs) from IDL files as described in *Generating Interface Objects or PDAs from IDL Files* in the section *Generating Interface Objects*.
Help		Displays SYSRPC help text: SYSRPC Utility - Overview Server Command Execution (Ping, Terminate) Expiration Time Logon Option Transport Protocol

Menu	Menu Item and Shortcut	Explanation
		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 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 options:

- **Expand Tree** described in *Service Directory Tree*,
- **Logon Option** described in *Service Directory Maintenance*,
- **Transport Protocol** described in *Service Directory Maintenance*, and

- **List Natural RPC servers** described in *Viewing a Server List* and *Viewing Additional Server Information* in the section *Listing Servers Registered on EntireX Broker*.

79 Service Directory Maintenance

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 RPC (Remote Procedure Call)* 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.

API for Service Directory Maintenance Functions:

You can use the application programming interface (API) `USR8216N` to perform service directory maintenance functions. `USR8216N` retrieves an existing service directory and adds, changes or deletes entries in the service directory. `USR8216N` is supplied in the Natural SYSEXT system library. For handling instructions, see *Using a Natural API* in the section *SYSEXT Utility*.

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).

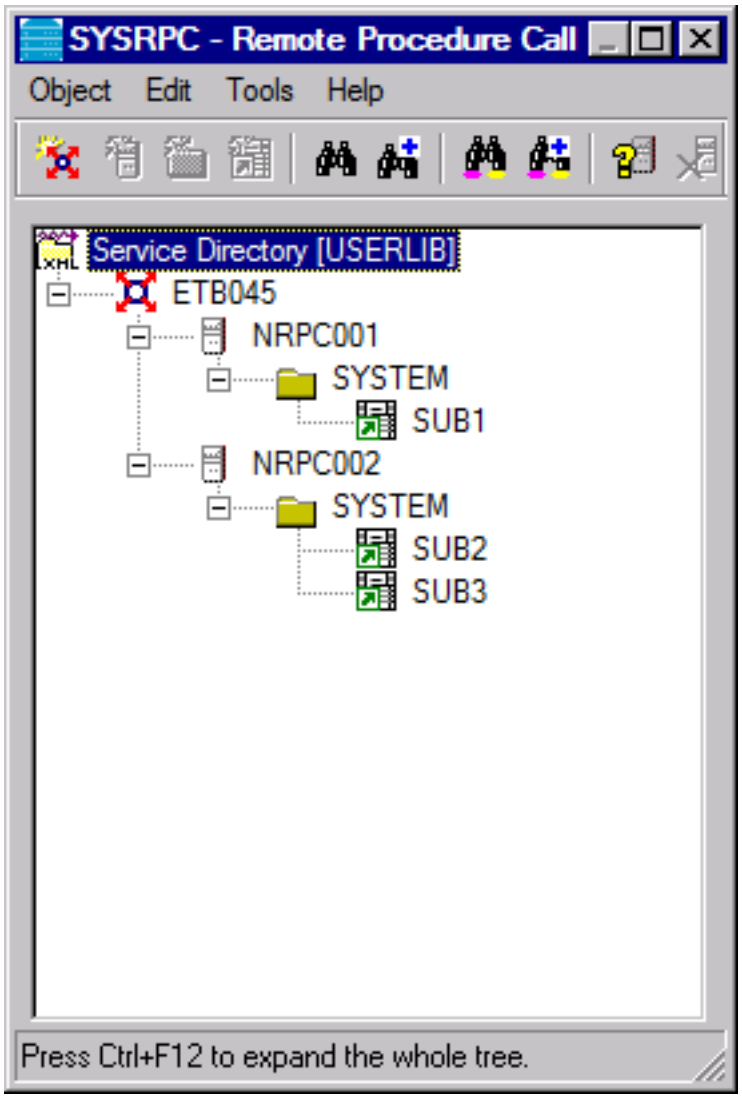
This section covers the following topics:

- Nodes and Servers

Nodes and Servers

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 in the *Parameter Reference* documentation.

In the example of a Service Directory below, two servers are defined for one node:

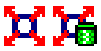


Both servers are connected to the same node: ETB045. The remote CALLNAT to subprogram SUB1 is executed on server NRPC001, whereas subprograms SUB2 and SUB3 are executed on server NRPC002.

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	The service directory root node indicates the name of the library from which the service directory was read: **Service Directory [***library-name***]**. For example: If you invoked the SYSRPC utility from the library USERLIB, the root reads **Service Directory [USERLIB]**. 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`. For explanations of other root node names that can occur, see ***Root Node Names in Error Situations***.
	Node	The name of the node to which the remote `CALLNAT` is sent. Maximum name length: 192 characters Depending on the setting of the **Logon Option**, either of the following icons is displayed:  Logon = No  Logon = Yes See also **Logon Option** below.
	Server	The name of the server on which the `CALLNAT` is to be executed. Maximum name length: 32 characters Depending on the setting of the **Logon Option**, either of the following icons is displayed:  Logon = No  Logon = Yes

Icon	Tree Node	Explanation
		See also ***Logon Option*** below.
	Library	SYSTEM or the name of the library to which your client application is logged on during the execution of the remote `CALLNAT`.
	Service (Subprogram)	The name of the remote subprogram to be accessed from the client. Maximum number of entries: 500 subprograms.

The section below contains information on:

- Root Node Names in Error Situations
- Selection Criteria for Node and Server

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 [*library-name*]	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 [***library-name***]**.
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 [***library-name***]**.
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. 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 [***library-name***]**.

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).

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 RPC (Remote Procedure Call)* 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 RPC (Remote Procedure Call)* 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 or a server 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 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 or a server and choose **Transport Protocol** and **ACI** for EntireX Broker ACI protocol from the context menu.

80 Replacing Items in the Service Directory

You can use the SYSRPC SM REPLACE direct command to replace the names of nodes, servers, libraries and programs defined in a service directory.

SYSRPC SM REPLACE corresponds to the **Replace function** of the **Edit** menu.

SYSRPC SM REPLACE can be used in online and batch mode.

This section contains information on:

Syntax of SYSRPC SM REPLACE

```
SYSRPC SM REPLACE replace-clause
```

replace-clause

```
[ ANY     ]
[ NODE    ]
[ SERVER  ] search-string WITH replace-string [ ALL   ] [WHOLE]
[ LIBRARY ]                                   [ FIRST ]
[ PROGRAM ]
```

The syntax items are explained in the following table:

ANY	Searches for all names specified in the service directory. This is the default value.
NODE	Searches for node names only.
SERVER	Searches for server names only.
LIBRARY	Searches for library names only.
PROGRAM	Searches for program names only.
search-string	An alphanumeric search string of up to 32 characters.
WITH	Introduces the *replace-string*.
replace-string	An alphanumeric replace string of up to 32 characters.
ALL	Replaces all occurrences of the search string found.
FIRST	Replaces only the first occurrence of the search string found. This is the default value.
WHOLE	Replaces only occurrences that match the whole search string.

Note: The search operation is not case-sensitive.

81 Generating Interface Objects - General Considerations

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 RPC (Remote Procedure Call)* 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 SYSRPC**Interface 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**

82 Generating Single Interface Objects with Parameter Specification

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 RPC (Remote Procedure Call)* 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	The level of the field. A level can be a number in the range from `01` (highest level) to `99` (lowest level). The leading `0` is optional. See also ***Defining Groups*** and ***Example 2*** for an example of a group definition.
Attribute	The attribute of the parameter: `M` (modifiable - INOUT), `O` (output - OUT) or `I` (input - IN). 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 ***Example 2***. If an interface object has been generated from a subprogram, the attribute is `M` by default, which may need modification. If an interface object has been generated from another interface object, the attribute values specified for the original object are retained. The generated interface object contains a comment that indicates the attribute specified for the parameter: `IN`, `OUT` or `INOUT`.
Type	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. For a description of Natural data formats, see *Format and Length of User-Defined Variables* and *Special Formats* in the section *User-Defined Variables* in the *Programming Guide*.
Length	The length of the parameter or `DYNAMIC`.

Field	Description
	This field does not apply to the following Natural data formats: D (date), G (group), L (logical) and T (time). The Natural data format A is restricted to 1073739357 bytes, the Natural data format B is restricted to 536869678 bytes. DYNAMIC indicates a dynamic parameter and applies to the Natural data formats A and B.
Precision	Only applies to Natural data formats N (numeric) and P (packed). Optional. The precision of the parameter, that is, the number of digits after the decimal point.
Dimension 1/2/3	Only applies to arrays. Optional. The first, second and third dimension of the parameter. An X-array or an X-group array is specified by entering an asterisk (*) for a dimension. See also *Defining X-Arrays and X-Group Arrays*.

This section 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				

83 Generating Multiple Interface Objects

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 direct 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.

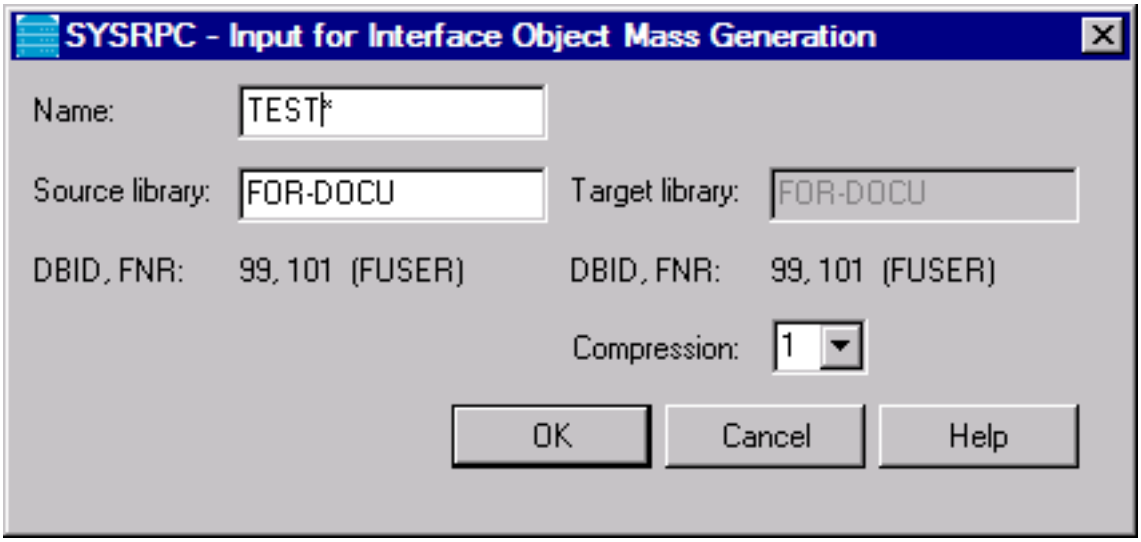
To perform the Interface Object Mass Generation function

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.

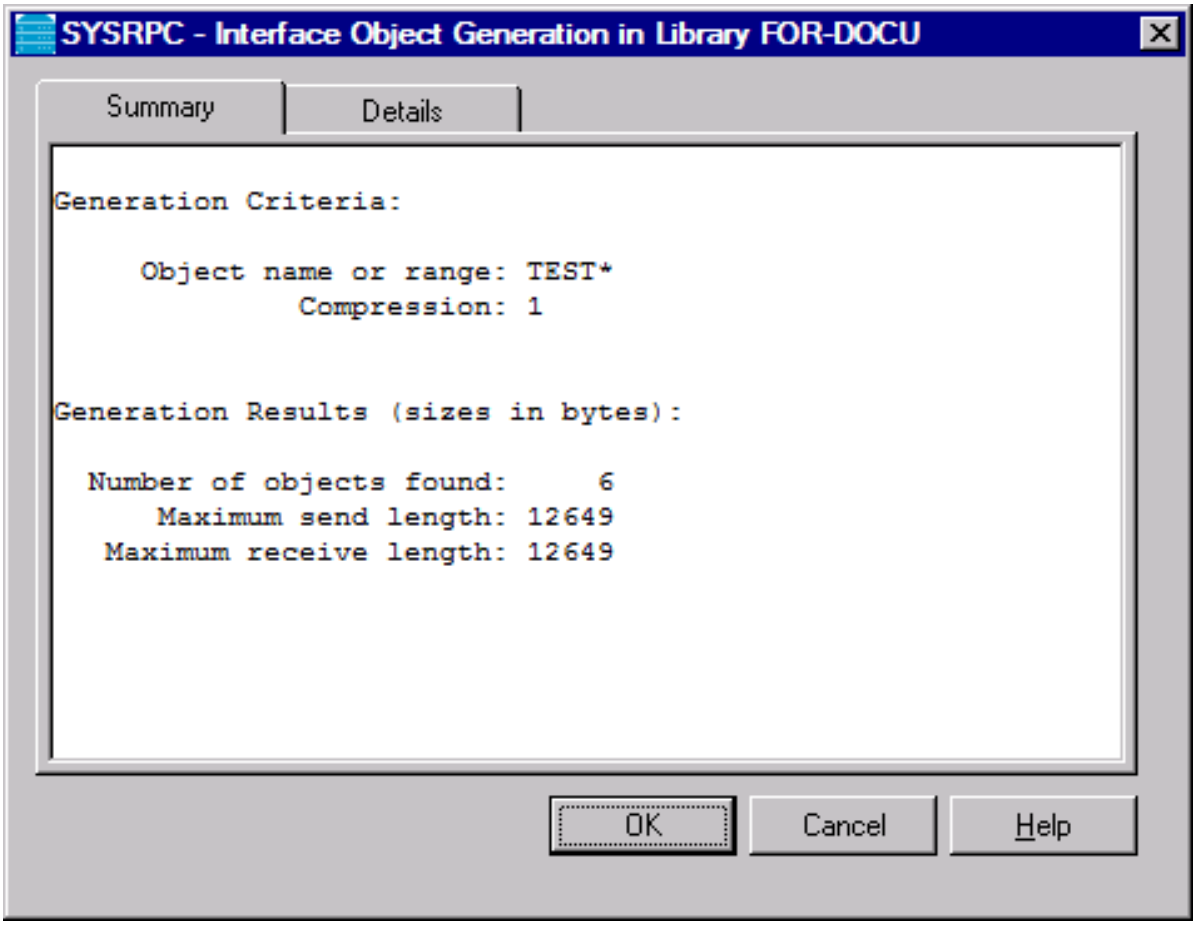
DBID, FNR are non-modifiable text boxes that display the database ID (DBID), the file number (FNR) and the type of Natural file (`FNAT` = system, `FUSER` = user) for the source and target libraries entered.

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 RPC (Remote Procedure Call)* 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:



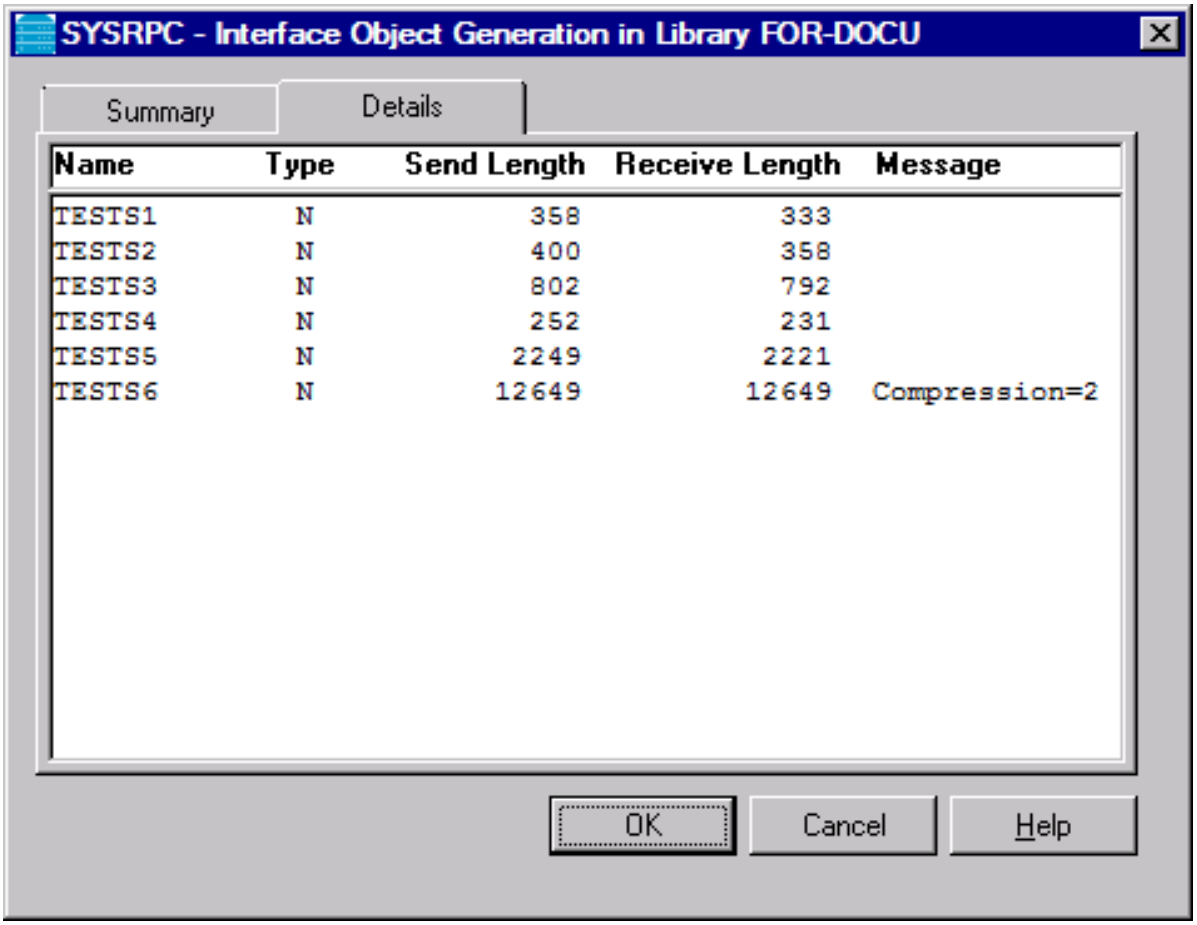
The report is organized in two sections, which contain the following information:

- **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 (sizes in bytes):**
  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 Direct Command

You can enter the `SYSRPC SGMASS` direct command 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.

Syntax of SYSRPC SGMASS

The syntax that applies to the `SYSRPC SGMASS` direct command 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

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.
value	A subprogram with a name equal to *value*.
*value**	All subprograms with names that start with *value*.
value<	All subprograms with names less than or equal to *value*.
value>	All subprograms with names greater than or equal to *value*.

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 RPC (Remote Procedure Call)* documentation.

84 Generating Interface Objects or PDAs from IDL Files

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.

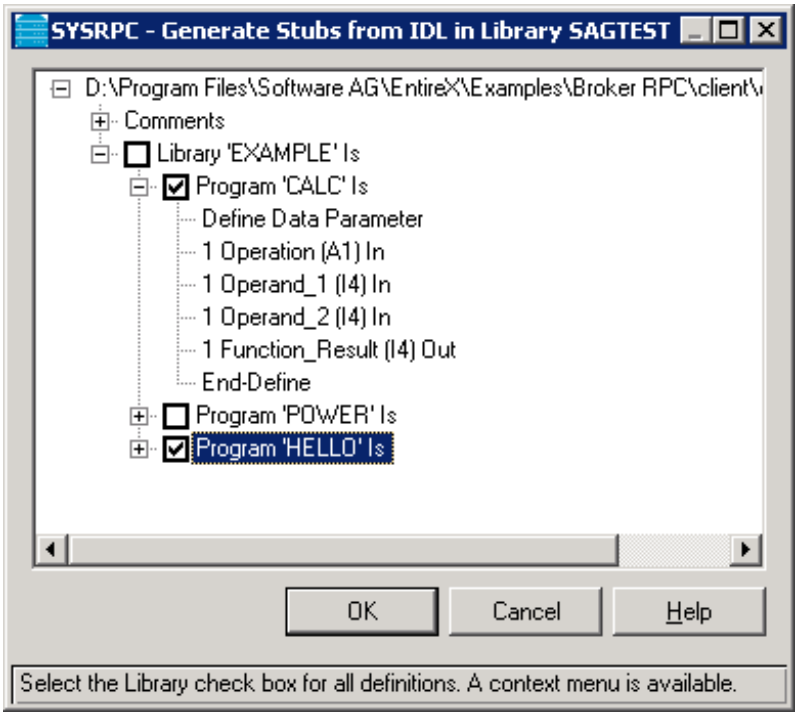
2 In the **Path or file name** text box, enter the complete path to the folder or the file that contains the IDL definitions or choose **Browse** to select a file from a folder.

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.

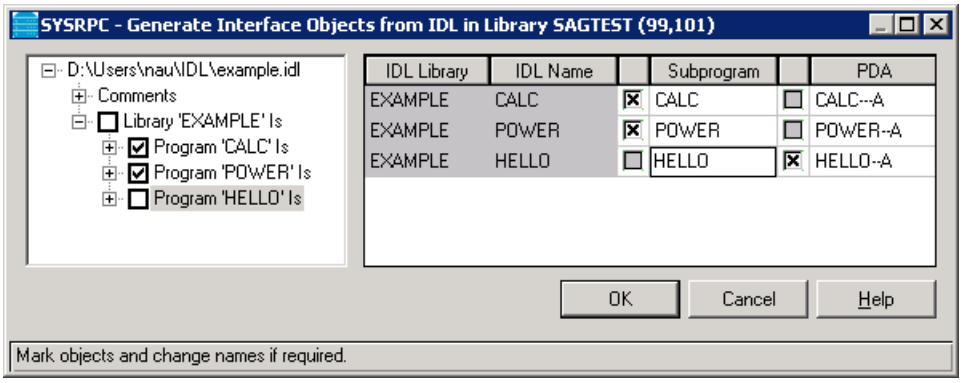
Building a Selection List

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.

85 Calculating Size Requirements

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 `SYSRPC CSMASS` direct command 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** .

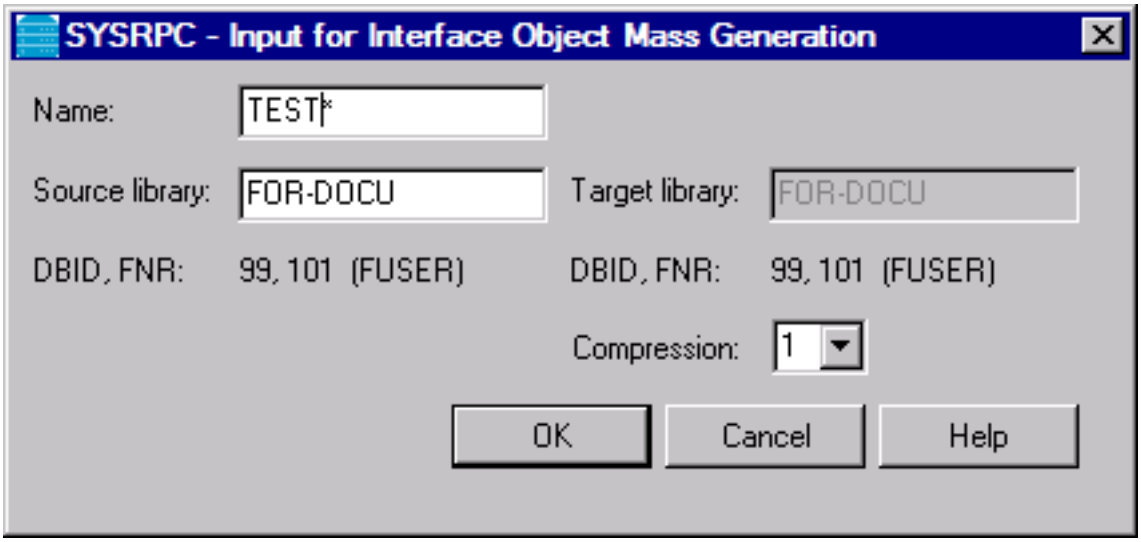
To perform Interface Object Mass Calculation

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.

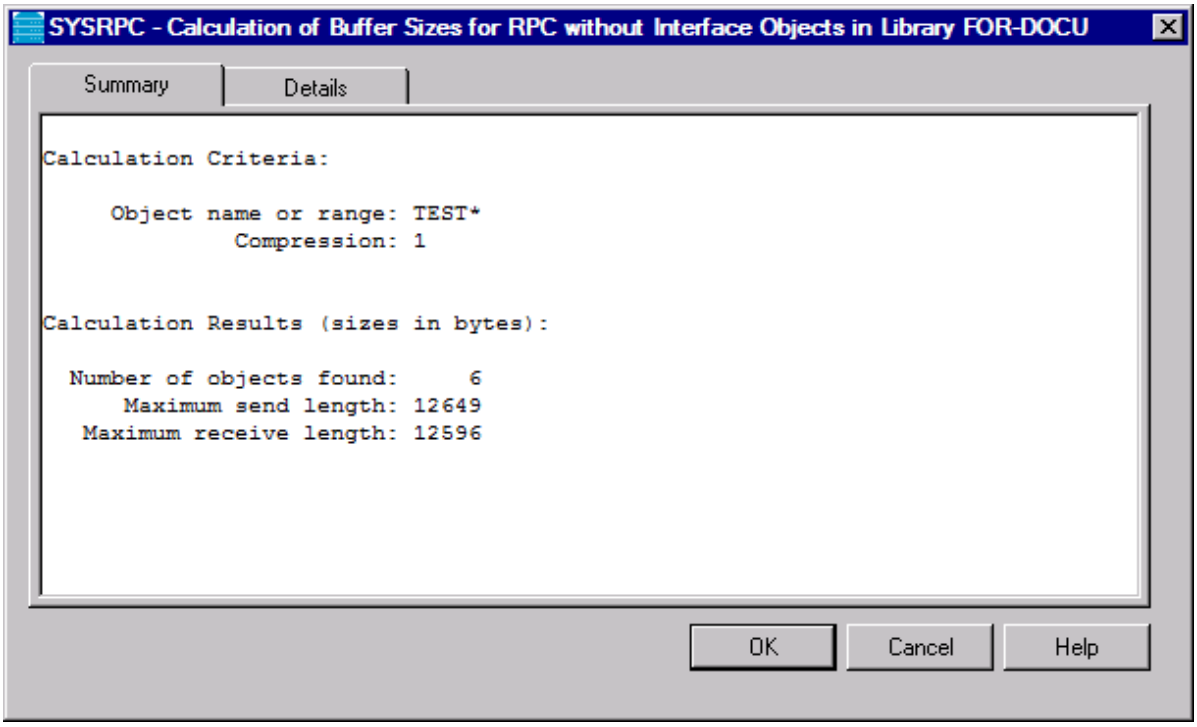
DBID, FNR are read-only text boxes that display the database ID (DBID), the file number (FNR) and the type of Natural file (`FNAT` = system, `FUSER` = user) for the source and target libraries entered.

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 RPC (Remote Procedure Call)* 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:

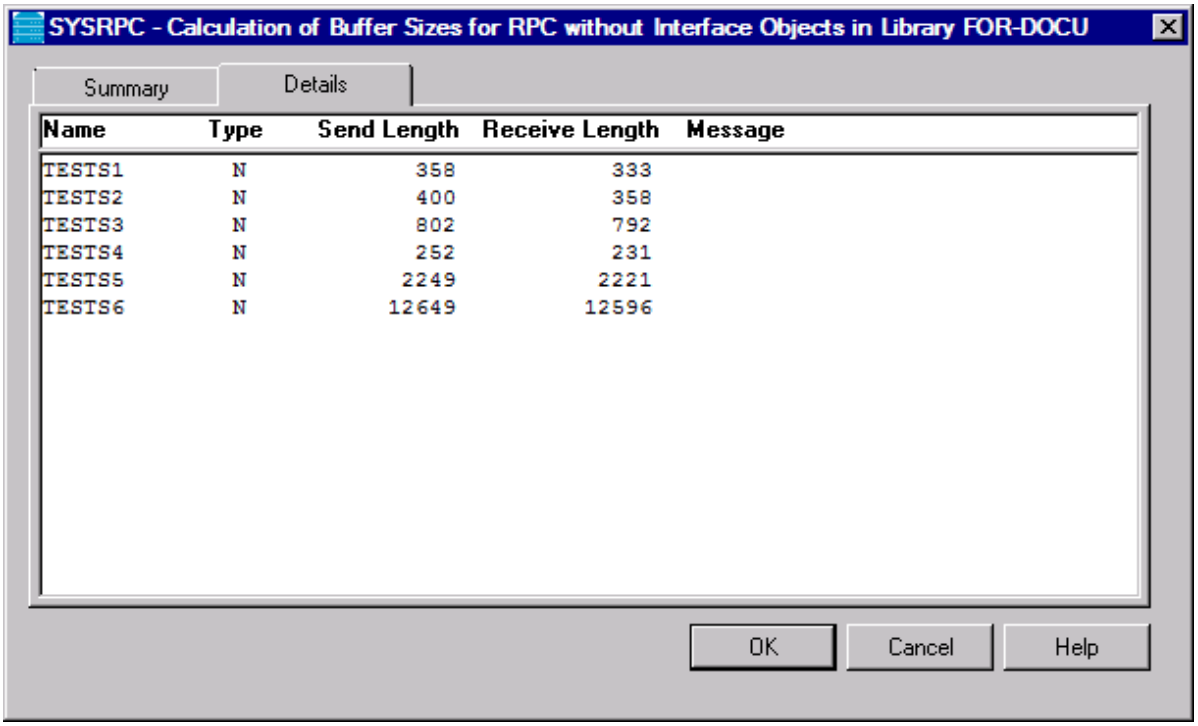


The report is organized in two sections, which contain the following information:

- **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 (sizes in bytes):**
  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:



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 Direct Command

You can enter `SYSRPC CSMASS` direct command 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 `SYSRPC CSMASS` direct command 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.
value	A subprogram with a name equal to *value*.
*value**	All subprograms with names that start with *value*.
value<	All subprograms with names less than or equal to *value*
value>	All subprograms with names greater than or equal to *value*.

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 RPC (Remote Procedure Call)* documentation.

86 Server Command Execution

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.

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 name of the node and the server selected as indicated in the following section.

Pinging an RPC Server

You can ping an RPC server from the standard or extended message view of the **Server Command Execution** screen or by using the `SYSRPC PING` **direct command**.

For information on pinging an RPC server by using the Application Programming Interface USR2073N, see the appropriate *Natural RPC (Remote Procedure Call)* documentation.

The following section provides instructions for pinging an RPC server by using menu functions.

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 8.3.7.0 on WNT-x86.`

This section covers the following topic:

- Using the SYSRPC PING Direct Command

Using the SYSRPC PING Direct Command

You can ping an RPC server by using the `SYSRPC PING` direct command in online or batch mode.

The following command syntax applies:

```
SYSRPC PING { server-name ON broker-name [[PORT] port-number][TRANSPORT {TCP|SSL|NET}] }
            { ALL                                                                      }
```

The symbols used in the syntax diagram are explained in the section *Syntax Symbols* in the *Statements* documentation.

The syntax elements are explained in the following table:

Syntax Element	Format/Length	Description
server-name	A32	Name of an RPC server or a range of names An asterisk (*) selects all names, asterisk notation selects all names that start with the specified value.
broker-name	A32	Name of the EntireX Broker or a range of names An asterisk (*) selects all names, asterisk notation selects all names that start with the specified value.
port-number	N5	Port number of the network address used for the server connection Valid values: 0 to 65535
TRANSPORT	A3	Transport method used by EntireX Broker:

Syntax Element	Format/Length	Description	
		TCP	TCP/IP protocol
		SSL	SSL or TLS
		NET	Entire Net-Work (not supported on UNIX or Windows)
ALL	n/a	Pings all RPC servers defined in the service directory of the current library.	

Terminating an RPC Server

The SYSRPC utility provides the following commands to terminate an RPC server or EntireX Broker service:

Terminate Server:	Terminates a single RPC server task by sending an internal message to the RPC server. If an RPC server is associated with multiple RPC server tasks (including replica on mainframe platforms), you can either terminate each RPC server task separately by using **Terminate Server**, or terminate all RPC server tasks in one go by using the **Terminate EntireX Broker Service** command.
Terminate Server with Sequence Number:	Lists the sequence number(s) for an RPC server registered on an EntireX Broker. Each server task is identified by start date/time, host name, application name and IP address, and can be selected and terminated from this list.
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 *Terminating a Natural RPC Server* described in the *Natural RPC (Remote Procedure Call)* 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 8.3.7.0 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 RPC (Remote Procedure Call)* 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 a single server task with sequence number

1 In the **Service Directory** tree view, select a server node and choose **Terminate Server with Sequence Number** from the context menu.

If required for the logon, a **Terminating EntireX Broker Service** dialog opens.

2 Enter a valid EntireX Broker user ID and password.

Select the **Terminate immediately** check box if you want to immediately terminate all server tasks that are involved in a conversation. If not selected (this is the default setting), all server tasks involved in a conversation remain operational.

Select the **Do not show this dialog again** check box (not selected by default) if you do not want the **Terminating EntireX Broker Service** dialog box to appear repeatedly during the current SYSRPC utilty session.

3 Choose **OK**.

A **SYSRPC Terminate - Information** window opens with a list of server tasks. It provides the following information:

Column	Description
Terminate	Check box to be selected for the terminating the corresponding server task.
Sequence No.	The number assigned to the server task in the sequence in which they are loaded by the EntireX Broker.
Date / Time	The date and time (UTC) when the server was started.
Status	The status of the server. Possible status values are: `idle`, `busy` or `term` (for terminated).
Host Name	The node where the server is hosted.
Application Name	The application name depends on the environment where the server runs. It can be the name of a Natural image (on UNIX), an LPAR (on mainframes) or a CICS started task, for example.
IP Address	The IPv4 or IPv6 address of the node.

You can open a context menu to hide or show additional columns, resize the window, or mark the check boxes of all servers you want to terminate.

4 Select the **Terminate** check box next to the sequence number of each server task to be terminated.

Or:

Choose **Mark all servers** from the context menu to select all server tasks listed in the window.

5 Choose **OK**.

A **Message Display** window opens displaying the name of the nodes, servers and sequence numbers of the terminated server tasks.

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.

87 Listing Servers Registered on EntireX Broker

You can obtain information on RPC servers registered on EntireX Broker by using the `SYSRPC SRVLIST` direct command.

This command corresponds to the **List Natural RPC servers** function of the **Service Directory** context menu.

`SYSRPC SRVLIST` sends a call to EntireX Broker requesting information on RPC servers registered on EntireX Broker with the attributes `SERVER-CLASS=RPC` and `SERVICE=CALLNAT`.

You can execute `SYSRPC SRVLIST` online (issued from a Natural command prompt) or in batch mode.

Note: When you execute this command online, a window prompts you to logon to the EntireX Broker specified in the command.

The following command syntax applies to `SYSRPC SRVLIST`:

```
SYSRPC SRVLIST server-name ON broker-name [[PORT] port-number][TRANSPORT
{TCP|SSL|NET}][USING{HEAD1MAP|object-name}]
```

The symbols used in the syntax diagram are explained in the section *Syntax Symbols* in the *Statements* documentation.

The syntax elements are explained in the following table:

Syntax Element	Format/Length	Description	
server-name	A32	Name of an RPC server or a range of names An asterisk (*) selects all names, asterisk notation selects all names that start with the specified value.	
broker-name	A32	Name of the EntireX Broker or a range of names An asterisk (*) selects all names, asterisk notation selects all names that start with the specified value.	
port-number	N5	Port number of the network address used for the server connection. Valid values: `0` to `65535`	
`TRANSPORT`	A3	Transport method used by EntireX Broker:	
		`TCP`	TCP/IP protocol
		`SSL`	SSL or TLS
		`NET`	Entire Net-Work (not supported on UNIX or Windows)
object-name	A8	Name of the Natural text object used to customize a server report. See also ***Customizing Server Lists***.	

This section covers the following topics:

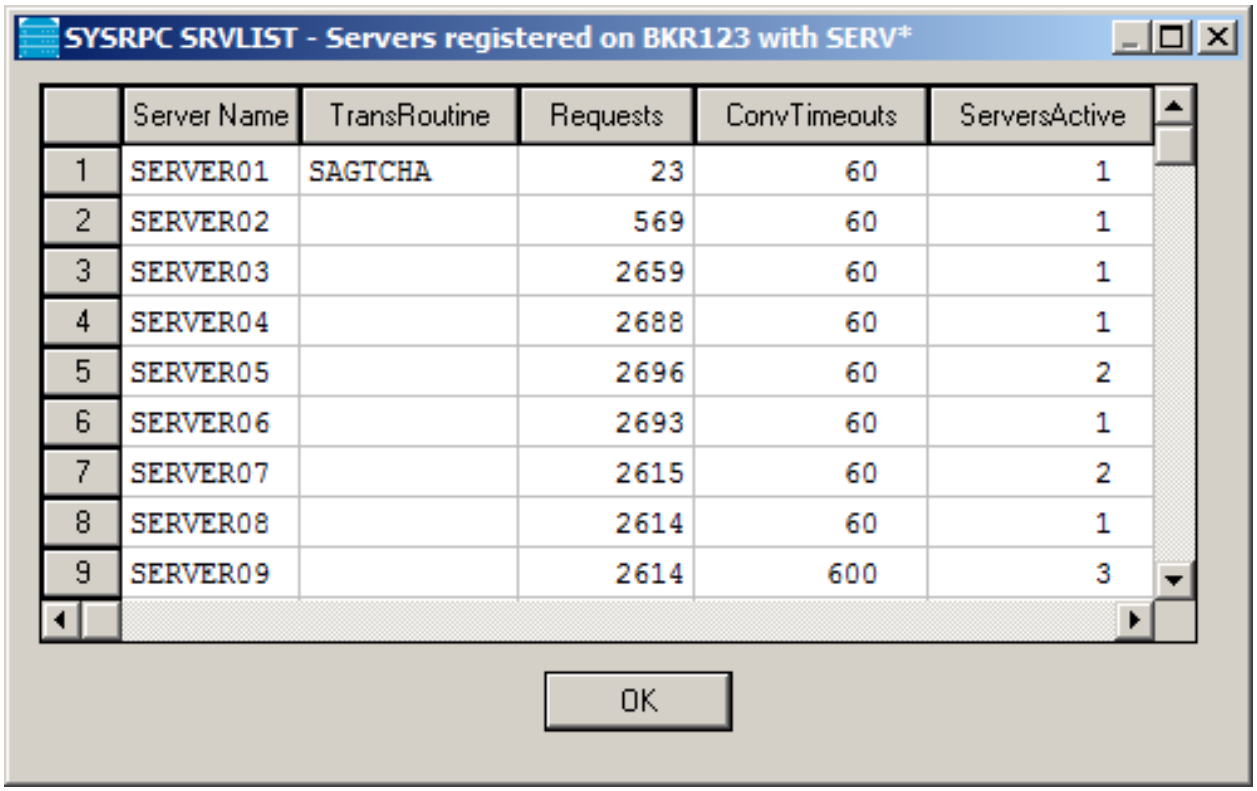
Example of an SYSRPC SRVLIST Direct Command

```
SYSRPC SRVLIST SERV* ON BRK123
```

This command returns data for all servers whose names start with `SERV` on EntireX Broker `BRK123`.

Viewing a Server List

When you execute the `SYSRPC SRVLIST` direct command or corresponding **List Natural RPC servers** context menu function, a **Servers** list window similar to the example below opens:



The columns and column headings on the **Servers** window are described in the `HEAD1MAP` text object. See also *Customizing Server Lists*.

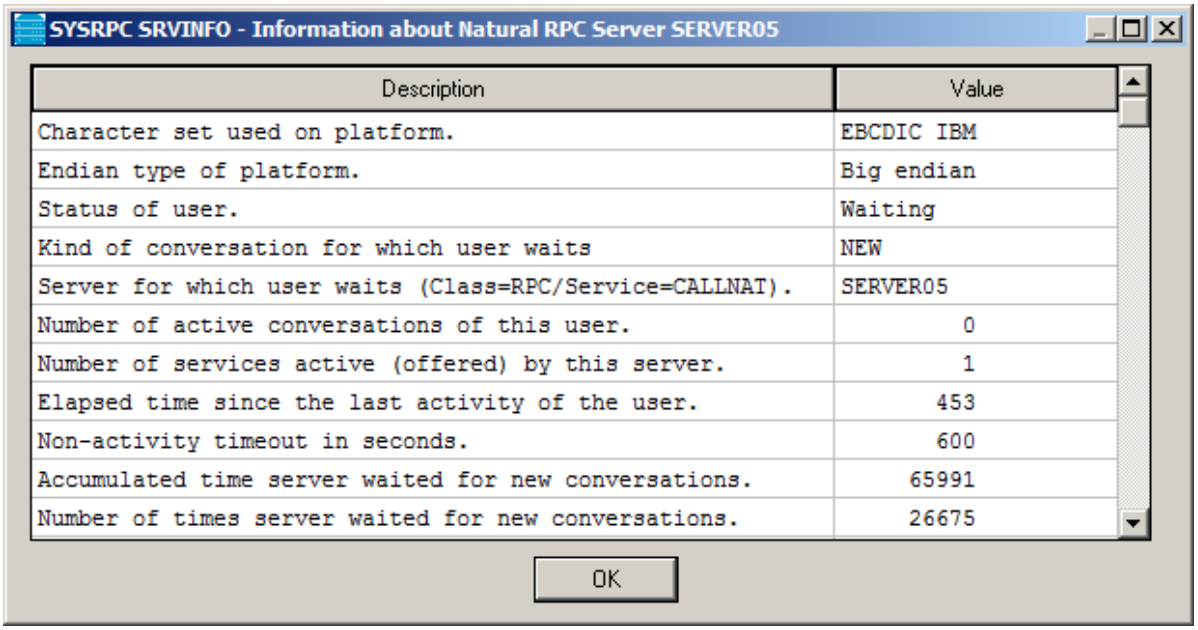
Viewing Additional Server Information

You can display additional information on a specific RPC server.

To display additional information for a single server

- In the **Servers list window**, select the server for which you want to display additional information and choose **Additional information about server** from the context menu.

  An **Information about Natural RPC Server** window similar to the example below opens:



  The screen displays all information EntireX Broker returned for the requested server. See also *Customizing Server Lists*.

Customizing Server Lists

You can rearrange a **list of servers** or a **server information list** as required by using the Natural `HEAD1MAP` or `HEAD2MAP` text object, respectively. `HEAD1MAP` and `HEAD2MAP` are supplied in the SYSRPC system library.

We recommend that you copy `HEAD1MAP` (list of servers) from the SYSRPC library to a user-defined library before you start editing the list. You can then rename the object and reference it in the `SRVLIST` command. You cannot rename `HEAD2MAP` (server information) list).

The text objects to be used must be contained in the current library, the library specified with the profile parameter RPCSDIR (see the *Parameter Reference* documentation), or the SYSRPC system library if the object name HEAD1MAP is used.

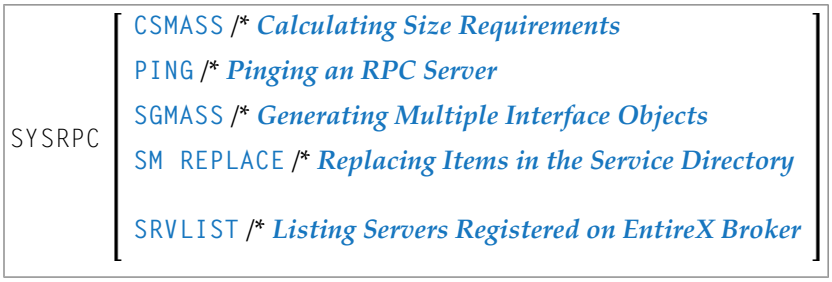
HEAD1MAP and HEAD2MAP contain instructions on how you change a list according to your needs. You can comment out the source code lines for columns and headings not required in your report. You can change code line positions to reorder columns. **Exceptions:**

- For HEAD1MAP: You must not comment out or move the first source line containing the SERVER-NAME field. You must not change the name of a field in the **Field** column.
- For HEAD2MAP: You must not change the name of a field in the **Field** column.

88 Overview of SYSRPC Direct and Batch Commands

The following syntax diagram is an overview of SYSRPC direct commands available online and in batch mode.

The comment next to each command indicates the section where the respective command is described in this chapter.



XVIII Tamino Server Extensions

89 Tamino Server Extensions

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

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 SYSEXSXS. To be able to locate the interface module, you must have defined the library **SYSEXSXS** 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 SYSEXSXS.

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:

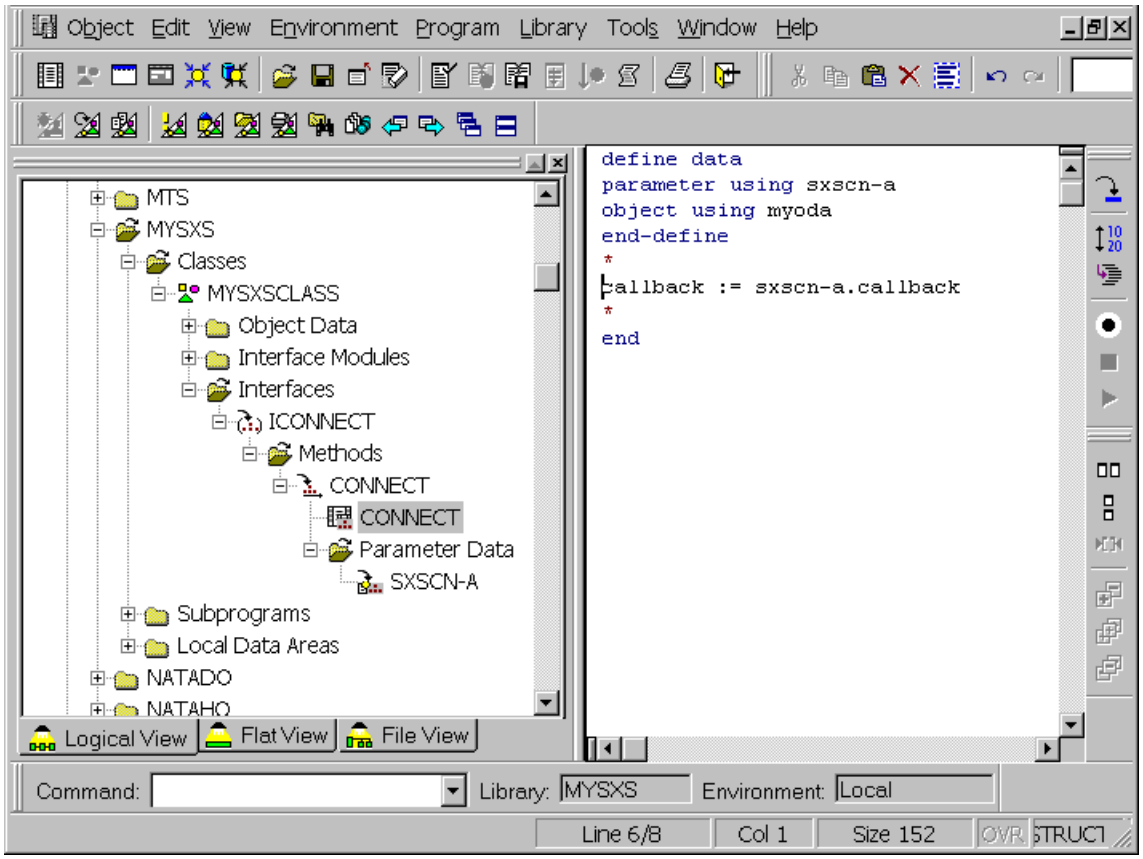
1 Fully expand the branch "Interfaces" of your class in the library workspace.

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.

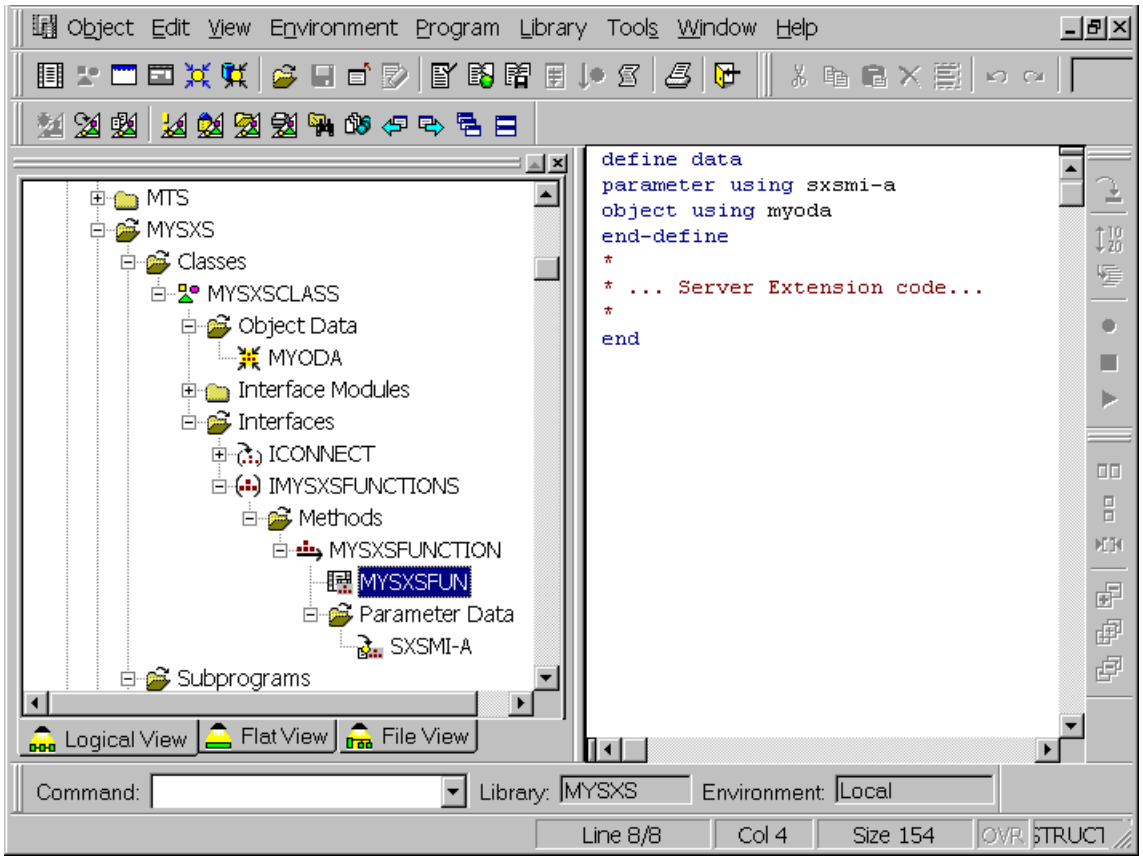
To implement the Server Extension Function:

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 *<install-dir>**Natural**Etc**<comserverid>**<class-name>**<version>*, where *<install-dir>* is the Natural installation directory, *<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 SYSEXSXS.
6. Execute "Load Tamino Data" and select the data file "EmployeeSXSData.xml" from the RES subdirectory of the sample library SYSEXSXS.
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.