

## **Natural Single Point of Development**

February 2010

---

Natural

This document applies to Natural Single Point of Development .

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

Copyright © 2001-2010 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, United States of America, and/or their licensors.

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

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

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

## Table of Contents

1 Natural Single Point of Development .....	1
2 What's New? .....	3
New Natural Development Servers .....	4
Automatic Mapping of Remote Environment .....	4
Drag and Drop between Multiple Remote Environments .....	4
Remote Debugging .....	5
SYSPROD Command .....	5
SYSFILE Command .....	5
SYSAPI and SYSEXT Availability .....	5
XML Toolkit Plug-In with SPoD .....	5
Web Interface Plug-In with SPoD .....	6
Object Handler .....	6
Source Locking .....	6
Unlock from Command Line .....	6
Natural Program-to-Program Communication with SPoD .....	7
Migration to New SPoD Versions .....	7
Changes in the Documentation .....	7
3 Introducing Natural Single Point of Development .....	9
4 What is Natural Single Point of Development? .....	11
Introduction .....	12
The New SPoD .....	12
Powerful Features .....	14
Productivity Advantages and Benefits .....	17
Looking Ahead .....	18
5 Remote Development Interface .....	19
6 SPoD Application Concept .....	23
Conventional Approach .....	24
SPoD Approach .....	24
Base Application .....	25
Compound Application .....	27
Application Workspace .....	28
7 Natural SPoD Architecture .....	29
Development Client .....	31
Development Server .....	31
Add-on Product .....	31
Security .....	31
8 Natural SPoD Frequently Asked Questions .....	33
Which versions of Natural and the related products are required to use the full set of functionality? .....	34
Can Natural objects of previous versions be used in a SPoD environment? .....	34
Do I need Predict? .....	48
Can I use Predict Case in a SPoD environment? .....	34
Can I use Natural Construct in a SPoD environment? .....	35

Can I use Predict Application Control in a SPoD environment? .....	35
What is the difference between the SPoD application concept and the earlier application shell concept in Natural for Windows? .....	35
9 First Steps with Natural Single Point of Development .....	37
10 Starting Natural .....	39
11 Customizing the Natural Studio Window .....	41
Moving and Docking Windows .....	42
Resizing Windows .....	44
Using Different Views .....	44
Displaying Additional Toolbars .....	47
Displaying the Command Line .....	49
12 Local and Remote Environment .....	51
Checking the Environment .....	52
Connecting to a Development Server for the First Time .....	52
Connecting to a Previously Mapped Development Server .....	54
Mapping versus Connecting an Environment .....	56
Logging on to a Library .....	56
13 Issuing Commands .....	59
Context Menus .....	60
Creating User Libraries .....	63
Copying and Moving Objects .....	64
Deleting Objects .....	66
Cataloging Objects .....	67
Displaying the Last Commands .....	69
Listing Objects .....	71
Invoking Terminal Emulation .....	73
14 Handling Programs .....	75
Creating a New Program .....	76
Stowing a Program .....	78
Executing a Program .....	79
Debugging a Program .....	80
15 Locking and Unlocking .....	83
Opening an Object .....	84
Opening the Same Object from Another Session .....	85
Unlocking Objects .....	86
Displaying a List of All Locked Objects .....	87
Moving Folders Containing Locked Objects .....	89
16 Handling Applications .....	91
Prerequisites .....	92
Displaying the Application Workspace .....	92
Creating a Base Application .....	94
Creating a Compound Application .....	97
Linking Objects to a Base Application .....	98
Linking Base Applications to a Compound Application .....	100
Managing Linked Objects .....	101

---





Mapping an Application .....	102
Displaying the Properties of an Application .....	103
17 SPoD Administration Policies and Procedures .....	105
18 Installing Natural Single Point of Development .....	107
Prerequisites .....	108
Installation Procedure .....	108
19 Configuring Natural Single Point of Development .....	111
Configuring Natural for Windows .....	112
Configuring Natural for Mainframes .....	112
Configuring Natural for UNIX .....	113
Configuring Natural for OpenVMS .....	113
Configuring the Natural Development Server .....	113
20 SPoD-Specific Extensions .....	115
User Exits for Natural Development Server on Mainframes .....	116
APIs for the SPoD Utility Protocol .....	117

---

# 1 Natural Single Point of Development

---

This documentation provides information on Natural Single Point of Development (SPoD).

	<b>What's New?</b>	Describes the new features, enhancements and corrections that apply to this release of SPoD.
	<b>Introducing Natural Single Point of Development</b>	What is SPoD? Products belonging to the SPoD architecture. Advantages and benefits.
	<b>First Steps with Natural Single Point of Development</b>	This tutorial provides an introduction to Natural Studio which is part of the SPoD concept. It is intended for developers who are not yet familiar with the Windows version of Natural.
	<b>SPoD Administration Policies and Procedures</b>	Provides information which is required to be able to administer a Natural development environment that is based on the SPoD concept.

For information on the prerequisites for Natural Single Point of Development, see [http://documentation.softwareag.com/natural/spod\\_prereq/prereq.htm](http://documentation.softwareag.com/natural/spod_prereq/prereq.htm).





## 2 What's New?

---

▪ New Natural Development Servers .....	4
▪ Automatic Mapping of Remote Environment .....	4
▪ Drag and Drop between Multiple Remote Environments .....	4
▪ Remote Debugging .....	5
▪ SYSPROD Command .....	5
▪ SYSFILE Command .....	5
▪ SYSAPI and SYSEXT Availability .....	5
▪ XML Toolkit Plug-In with SPoD .....	5
▪ Web Interface Plug-In with SPoD .....	6
▪ Object Handler .....	6
▪ Source Locking .....	6
▪ Unlock from Command Line .....	6
▪ Natural Program-to-Program Communication with SPoD .....	7
▪ Migration to New SPoD Versions .....	7
▪ Changes in the Documentation .....	7

This chapter describes the new features, enhancements and corrections that apply to the release of Version III of Natural Single Point of Development (SPoD).

## New Natural Development Servers

---

The following new development servers are available:

- **Natural Development Server for z/OS (Com-plete)**

The Natural Development Server (NDV) is now available on z/OS under Com-plete. For further information, see the Natural Development Server for z/OS (Com-plete) documentation.

- **Natural Development Server for Windows**

The Natural Development Server (NDV) is now available on Windows. This enables application developers to use one single development environment, that is Natural Studio, to develop Natural applications for all platforms supported by Software AG. Be it a terminal server environment or a local environment on a stand-alone PC, the tools are identical. The only restriction is that Natural dialogs cannot be executed on the NDV for Windows server. Data input and output takes place via the Web I/O Interface. For further information, see the Natural Development Server for Windows documentation.

- **Natural Development Server for OpenVMS**

The Natural Development Server (NDV) is now available on OpenVMS. For further information, see the Natural Development Server for OpenVMS documentation.

## Automatic Mapping of Remote Environment

---

During session start, Natural Studio automatically maps and connects the remote environment which was active when Natural Studio was closed. This reduces the number of steps to be performed and therefore enhances usability.

## Drag and Drop between Multiple Remote Environments

---

It is possible to drag and drop a library from one remote environment into another. This has the functionality of copy instead of move.

## Remote Debugging

---

When using SPoD to work in a remote environment located on a mainframe, UNIX or OpenVMS platform, the new integrated Natural Studio debugger can be used for remote debugging in a SPoD environment, thereby simplifying debugger usage.

The debugger supports debugging of subprograms executed by a remote Natural RPC server.

## SYSPROD Command

---

When using SPoD to work in a remote environment located on a mainframe, UNIX or OpenVMS platform, the redesigned Windows-based graphical user interface can be used in parallel with other utilities and as remote interface under SPoD. In a remote mainframe environment, subcomponents of Natural are displayed.

## SYSFILE Command

---

When using SPoD to work in a remote environment located on a mainframe, UNIX or OpenVMS platform, the enhanced graphical user interface can be used in parallel with other utilities and as remote interface under SPoD.

## SYSAPI and SYSEXT Availability

---

When using SPoD to work in a remote environment located on a mainframe, UNIX or OpenVMS platform, the Windows-based graphical user interface can be used as a remote interface under SPoD to access the SYSAPI and SYSEXT utilities.

## XML Toolkit Plug-In with SPoD

---

The XML Toolkit plug-in has been enhanced to work in a SPoD environment.



**Note:** This feature is not available on mainframe platforms

## Web Interface Plug-In with SPoD

---

The Web Interface plug-in has been enhanced to work in a SPoD environment.



**Note:** This feature is not available on mainframe platforms

## Object Handler

---

When using SPoD to work in a remote environment located on a mainframe, UNIX or OpenVMS platform, you can now select applications maintained in Natural Studio's application workspace for processing and you can select libraries or objects that belong to these applications. The basis for this functionality, which is currently only available via a direct command, has been implemented. The user interface will be adapted in a later release. Please refer to *Application Selection* in the *Object Handler* section of the *Tools and Utilities* documentation on the corresponding platform.

## Source Locking

---

Locking is also supported in mixed environments, that is, when you are editing in a SPoD environment while using one of the Natural editors.



**Note:** The term "mixed environments" currently only refers to an environment using Windows on the client side and a mainframe platform on the server side. An environment using Windows on the client side and a UNIX platform on the server side is not supported.

## Unlock from Command Line

---

The system command `UNLOCK` supports direct command syntax. Thus, it is possible to unlock an object from the command line.



**Note:** In conjunction with a Natural Development Server for Windows, the `UNLOCK` command feature will be available only if a development server file (FDIC) is used, which is optional with this development server.

## Natural Program-to-Program Communication with SPoD

---

Several application programming interfaces are provided to establish a program-to-program communication between a development server program and the Natural Studio client. The APIs use the existing SPoD protocol layer and provide the following functions:

- initialize the communication
- exchange data between the development server program and the Natural Studio client
- perform an interaction
- activate the trace facility and write trace data to a text member or into a log file

Thus customers may extend the Natural Studio and SPoD environments for remote development with their own tools. See *APIs for the SPoD Utility Protocol* for further details.

## Migration to New SPoD Versions

---

For information on the supported versions of Natural for Windows, Mainframes and UNIX and the corresponding versions of Natural Development Server, refer to Empower <https://empower.softwareag.com/>.

## Changes in the Documentation

---

A revised and updated documentation set is available.



# 3

## Introducing Natural Single Point of Development

---

This part covers the following topics:

- **What is Natural Single Point of Development?**
- **Remote Development Interface**
- **SPoD Application Concept**
- **Natural SPoD Architecture**
- **Natural SPoD Frequently Asked Questions**





# 4 What is Natural Single Point of Development?

---

- Introduction ..... 12
- The New SPoD ..... 12
- Powerful Features ..... 14
- Productivity Advantages and Benefits ..... 17
- Looking Ahead ..... 18

This chapter covers the following topics:

## Introduction

---

If you have been developing applications using Natural in a non-graphical environment, you have undoubtedly become familiar with Natural's wide range of capabilities and most likely also with some of its inconveniences and shortcomings. You have probably quite often wished for a more advanced user interface for “your Natural”. With Natural Single Point of Development, your wish has indeed become reality.

Simply speaking, one might say that Natural Single Point of Development is a Windows-based Natural workstation for remote application development on different platforms.

Actually, the Natural Single Point of Development concept covers more than a simple workstation in that it combines the advantages of two disparate worlds:

- the strengths of the character-based Natural products and
- the ease of use provided with Natural for Windows.

In addition, it provides a series of novel features which are explained below.

Natural Single Point of Development is more than wishful thinking - it is indeed *one Natural for all*.

## The New SPoD

---

Natural's Single Point of Development approach is Software AG's response to the increasing demand for a state-of-the-art development workstation for building, testing and maintaining all Natural applications throughout their life cycle and across all supported platforms.

The standard SPoD architecture comprises the following products:

- **Natural for Windows**  
This is the development client and the desktop. The desktop client includes Natural Studio, which is the actual interface where software engineers design applications.
- **Natural Development Server**  
Plug-in for the target platform, available for Mainframe (z/OS, z/VSE, BS2000/OSD) and UNIX, Linux, OpenVMS and Windows. The plug-in resides on each Natural installation in the target environment, enabling it for remote development. The development server maintains a development server file (FDIC) for loading information and application definitions.

- **Natural for the target platform**

Natural at the target platform plus the Natural Development Server plug-in make up the Natural development server.

Optionally:

- **Natural Security**

Both for the development client and the development server, if you wish to secure the local environment or the target environment.

In addition, the following development products and tools can be integrated to make up a full-flanged Natural development environment for increased developer productivity:

- **Predict**

An active data dictionary system which is used to describe information processing systems. It also provides functions to use this information when designing, implementing, using and maintaining the system. See also [Natural X-Ref Evaluation plug-in](#), [Object Description plug-in](#), [Schema Generation plug-in](#).

- **Predict Application Control (PAC)**

A flexible tool for controlling Natural and foreign applications throughout the software life-cycle and for ensuring the integrity of applications in the production environment. PAC facilitates and controls the movement of applications through the life-cycle. When an application is implemented into production, it is protected and audited by Predict Application Audit (PAA). See also [Object Versioning plug-in](#).

- **Natural Construct**

A powerful, easy-to-use, model-based application generator that helps you create and implement high-quality, robust Natural applications. See also [Construct Program Generation plug-in](#).

- **Natural Engineer**

A versatile maintenance and reengineering tool that cuts costs and enhances productivity. It enables you to respond faster to technological developments such as the Web, while fully leveraging prior investment in Natural applications and in proven business logic. See also [Metrics and XRef Viewing plug-in](#).

- **Natural ISPF (Integrated Structured Programming Facility)**

Software AG's application development tool for the building, testing and maintenance of applications throughout their life cycle. See also [Mainframe Navigation plug-in](#).

## Powerful Features

---

Natural's Single Point of Development approach is Software AG's response to the increasing demand for a state-of-the-art development workstation for building, testing and maintaining all Natural applications throughout their life cycle and across all supported platforms. It offers the following powerful features:

- **Client/server architecture enables one single remote development environment for all platforms**

Natural Single Point of Development is based on a client/server concept. On the client side, Windows-based Natural Studio with its modern look and feel, its powerful drag-and-drop copy and paste functionality and its browser-style workspaces offers one single, uniform view for all Natural developers. Its graphical user interface is the basis for a single, uniform working environment for all platforms (mainframe, mid-range and Windows computer systems) supported by Natural.

For more details, refer to [Natural SPoD Architecture](#).

- **Full remote development access after mapping to the target environment**

For remote development, the Windows-based Natural client (which will remain useable for Windows-only application development) can be enabled to interact with a development server that can be located on a single or multiple, homogeneous or heterogeneous mainframe or mid-range computer systems. Once you have mapped your target environment in the client's browser-style application workspace, you have a set of convenient and effective tools at hand that enable you to perform all development tasks directly in the target development environment. You can use the functionality to the extent it is available there. And to prevent concurrent updates, the object under work on the target platform is locked.

For more details, refer to [Remote Development Interface](#).

- **Single, familiar system image of all objects and resources involved in application development**

Natural and non-Natural objects can be accessed and processed with a single user interface. You can generate DCOM components, use RPC and EntireX communications and/or put your applications to the Internet. You can submit and monitor jobs, control job listings and perform dataset maintenance tasks. Operations on all of these different object types, whether they reside on z/OS, z/VSE, BS2000/OSD, UNIX, OpenVMS or Windows, is afforded using a well-known graphical user interface.

For more details, refer to [First Steps with Natural Single Point of Development](#).

- **New application concept giving a logical view to distributed applications**

To meet the requirements of a new, distributed application development scenario, a new-defined application concept has been introduced which provides a logical view of Natural and future objects on the various sites where they are developed and used. On the workstation screen, this

is complemented by an **application workspace** which shows all distributed objects of an application in a tree structure.

For more details, refer to *SPoD Application Concept*.

- **Remote development server file**

To cope with the requirements of the new, distributed application concept, a new central data dictionary file has been introduced. Having the same structure as the well-known Natural system file FDIC, the new system file serves to store the information where the objects linked to an application are stored and which objects are locked.

- **Future-proof by pluggable extras**

Additionally, the following pluggable extras integrating the functionality of the products Predict, Natural Construct and Natural ISPF are available:

- **Natural X-Ref Evaluation plug-in (former XRef GUI Client)**

When the generation of cross-reference data is activated, the Natural compiler provides cross-reference (XRef) data created during CATALOG or STOW commands in the development server file. XRef Evaluation evaluates this data with respect to your environmental settings, for example your logon library. It enables you to retrieve and display this essential development and maintenance information conveniently in Natural Studio.

- **Object Description plug-in**

Using Object Description you can manipulate and retrieve data stored by Predict on the server. Object Description is available as an optional plug-in unit for Natural Studio. Predict must be installed in the remote development environment.

- **Schema Generation plug-in**

External objects can be generated or incorporated with Predict on the server or with Schema Generation in Windows. Both have access to the same data. Schema Generation is available as an optional plug-in unit for Natural Studio. A wizard is provided which enables you to easily generate DDMs as well as schema definitions for Adabas and DB2. The generation of Adabas and (on mainframe platforms) DB2 structures is supported. This includes generation and incorporation of the following: Adabas databases (incorporation only), Adabas files, DB2 storagegroups, DB2 databases, DB2 procedures/functions (generation only), DB2 tablespaces, DB2 tables/views, DDMs for Natural (generation only), and activation of automatic processing rules (generation only). Predict must be installed in the remote development environment.

- **Construct Program Generation plug-in**

Natural programs can be generated with Natural Construct on the server or with Program Generation in Windows. Both have access to the same data. Program Generation is available as an optional plug-in unit for Natural Studio. A wizard is provided which enables you to easily generate Natural programs. Natural Construct must be installed in the remote development environment.

■ **Mainframe Navigation plug-in**

Mainframe Navigation allows you to access and manipulate objects which are stored on a mainframe. These objects include PDS libraries and members, load modules, sequential datasets, as well as system objects such as active jobs, the console, etc. under the z/OS, z/VSE or BS2000/OSD operating system. Mainframe objects are displayed in a tree structure and can be listed, browsed and edited with a Windows-like user interface. In order to use this plug-in, ISPF must be installed on the mainframe.

■ **Metrics and XRef Viewing plug-in**

Metrics and XRef Viewing is a new plug-in which allows you to display industry standard metric information (e.g. Halstead and McCabe) about your Natural Objects. In addition, you can see all referenced objects for a particular Natural object and optionally all data items used by the object. This data is presented in a recursive fashion. The Natural objects are displayed in a tree structure. A Natural Engineer repository which may reside on a mainframe or a PC is required (for the applicable Natural Engineer version, see [http://documentation.software-ag.com/natural/spod\\_prereq/prereq.htm](http://documentation.software-ag.com/natural/spod_prereq/prereq.htm)).

■ **Object Versioning plug-in**

Object Versioning provides the opportunity to bring Natural objects under version control for which applications have been defined with Predict Application Control (PAC) on the mainframe. With Object Versioning, you can perform all versioning-related work directly within the Natural development environment. There is no need to change to the SYSPAC library. Predict Application Control Version 2.4.2 or above is required.

For detailed information, refer to the corresponding plug-in documentation.

Which plug-ins are actually visible and active can be configured on a per user basis using the Plug-in Manager.

■ **Enhanced online help/documentation for remote environments**

For remote environments with a character user interface, Natural Single Point of Development enables application developers and administrators to display context-sensitive help and additional documentation in electronic form using browser style windows within the graphical user interface. This provides users with fast and efficient access to vast amounts of information which formerly required many volumes of printed manuals. Should you need complex information in printed form, you can print out your personal copy of each document that is available online.

In addition, the syntax help in the program editor available in Natural Studio offers full-detail context-sensitive help for the following Natural syntax elements:

- Statements
- System variables
- System functions
- Parameters (for example, the AD parameter)

■ **Natural Studio - a unified consistent interface**

With Natural Studio, software engineers perform all the necessary development tasks remotely, including file manipulation, editing, compiling, debugging, and cross referencing. Developers can manipulate (move, copy) program objects, wherever those objects are located. Natural source files are transparently retrieved from and saved to the target environment, and edited in Natural Studio in Windows. Compiles are initiated from Natural Studio by submitting commands to the target environment. The application executes on the target environment with debugging controlled from Natural Studio.

■ **Local development for the Windows platform within SPoD**

Developing Windows applications within SPoD is simple. There is no need for a separate Windows development environment. Natural Studio's Dialog Wizards and Dialog Editor help users develop complex user interfaces for Windows fat clients.

■ **Remote development on Windows platform**

Starting with Natural Version 6.3.3 for Windows, the Natural Development Server (NDV) is also available on Windows. This enables application developers to use one single development environment, that is Natural Studio, to develop Natural applications for all platforms supported by Software AG. Be it a terminal server environment or a local environment on a stand-alone PC, the tools are identical. The only restriction is that Natural dialogs cannot be executed on the NDV for Windows server.

■ **Web development with Natural's enhanced XML capabilities**

As the XML standard for exchanging data continues to gain popularity, Natural is evolving with the trend. Several Software AG development components, which are included in or are complementary to the SPoD environment, specifically take advantage of the XML standard. These include the DBMS layer, with Tamino XML Server, and the Natural language itself, with XML Toolkit for Natural. In addition, Natural supports XML-supplied XSL/stylesheet-based presentation clients for Web browsers.

## Productivity Advantages and Benefits

---

Independence from separate operating systems or TP monitors reduces the investment in training otherwise required for the different environments and shortens the turnaround time for application development.

This section covers the following topics:

- [Productivity Advantages](#)

- Benefits

### Productivity Advantages

- Common work area for remote development
- Simplified maintenance
- Control over Natural development tasks
- Familiar Windows GUI look and feel
- Support for client/server applications

### Benefits

- Training costs reduced
- Attractive to developers who expect familiar environments with familiar tools
- Lower costs for software development, maintenance, and administration
- Greater job satisfaction for software engineers

## Looking Ahead

---

As it becomes easier to solve complex business problems with these advanced tools and development environments, enterprises and the companies that support their efforts will be generating much more value. Brokered services, shared computing resources, and the formation of development alliances between strategically partnered companies have all gained popularity and profitability over the past few years, and we expect that evolution to continue. New developments and the combination of proven approaches will provide us with faster, more efficient and more secure ways to deal with the world's vast, rapidly expanding network of computing power.

The organizations with superior software assets will certainly provide much more capability to their customers. And, as software continues to evolve as an exchangeable, saleable component for specific solutions and larger systems, these organizations will enjoy significant leverage in the marketplace. With critical leaps in development connectivity like SPoD occurring, we can expect this evolution to continue.

At Software AG, we still consider software development to be in its infancy and we have been at it for more than 30 years. As we connect systems and continue to erase the boundaries between platforms, a whole new wave of creativity and innovation will be unleashed. We are looking forward to both the technical and actual benefits that lie ahead.

To learn more about what Natural can do for you, visit [www.softwareag.com/natural](http://www.softwareag.com/natural).



# 5 Remote Development Interface

---

The primary goal of the Single Point of Development (SPoD) approach is to provide a development interface that enables software engineers to develop Natural applications for any platform using only the Windows-based graphical user environment of Natural Studio.

The essential features are:

- **Remote File Manipulation**

In the Natural Studio views, you can manipulate (for example, move, copy) program objects, regardless of location.

- **Remote Editing**

You can retrieve Natural source files transparently from the target environment, edit them at your workstation and then save them to the target environment. The GUI supported dialogs of the editors provide significant advantages over the character-based editors.

- **Remote Compiling**

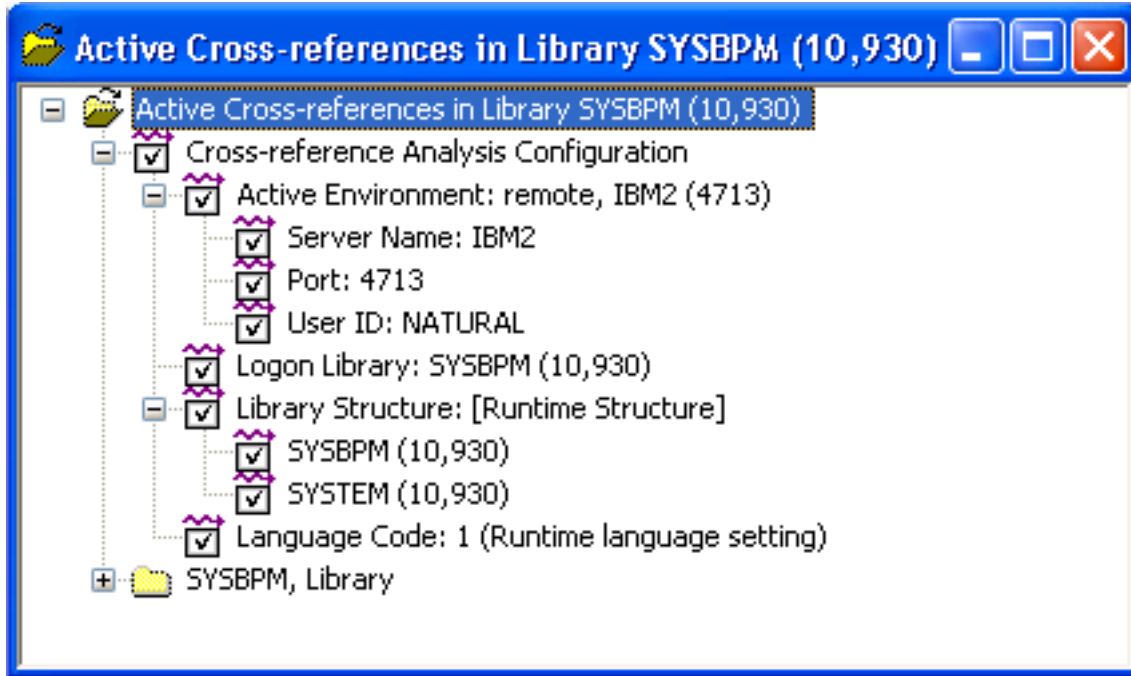
You initiate the compilation process from your workstation by issuing commands to the target environment.

- **Remote Debugging**

The debugger available with Natural Studio can be used to debug applications which execute in a target environment that can be located on the same system or in Natural remote environments.

- **XRef Information**

XRef Evaluation is available as an optional plug-in for Natural Studio. It stores cross-reference information created during `CATALOG` or `STOW` commands in a remote FDIC system file. This facility enables you to retrieve and display this essential development and maintenance information conveniently in Natural Studio. This gives you a comfortable way of listing and navigating through hierarchies of referenced and referencing objects, showing relationships (cross-references) between and within program objects. You need to have Predict installed on the target environment. The figure below shows an XRef example screen.



- **Object Locking**

When you access a remote development server, an object locking mechanism prevents concurrent updates. Locking information is kept in the development server file.

- **Terminal Emulation Window**

Maintenance of character-based applications often involves testing output to terminals. During remote debugging from Natural Studio, a terminal emulation window or a window of the Web I/O Interface appears automatically for that purpose. This feature is also available when you are using a utility with a character user interface, for example:

```

Terminal Emulation
Session Edit View Help

11:04:53          ***** NATURAL SYSPM UTILITY *****          2007-11-06
BPNAME QA42GBP   - Main Menu -                               Type Global Nat
BPPROP OFF                                             Loc DAEF QA42
                                                       Preload QA42GBPL

      Object Functions                                Object Pool Statistics

      L List Objects                                  A Buffer Pool
      D Delete Objects                               C BP Cache
      I Directory Information
      H Hexadecimal Display
      W Write to Work File
      X Display Sorted Extract
      ? Help
      . Exit

      Other Functions

      S Select Buffer Pool
      B Blacklist Maintenance
      P Preload List Maintenance

Code ..  Library ... * _____
          Object .... * _____
          DBID ..... 0 _____ FNR .. 0 _____ Object Pool ... * (B,C,*)

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---
      Help      Exit Last      Flip                                Canc

4AÛ                                                    17,014

```



**Note:** The Development Server for Windows supports only the Web I/O Interface.

The terminal emulation window is not Unicode-enabled. The Web I/O Interface is able to display information which contains Unicode characters. The window which will appear at your site depends on the parameters which have been set on the Natural development server.

In summary, the benefits of remote development and maintenance are:

- Better control over Natural development tasks
- Increased productivity through a powerful graphical work area
- Lower costs for software development, maintenance and administration
- Greater job satisfaction for software engineers



# 6 SPoD Application Concept

---

- Conventional Approach ..... 24
- SPoD Approach ..... 24
- Base Application ..... 25
- Compound Application ..... 27
- Application Workspace ..... 28

This chapter covers the following topics:

## Conventional Approach

---

A conventional Natural application consists of a collection of Natural and non-Natural objects. Together, they form a functional unit which covers the business logic for a particular business problem. An application consists of a set of libraries and their Natural objects. It is possible that one part of the library belongs to one application while another part belongs to another (different) application.

## SPoD Approach

---

With the SPoD approach, the term “Natural application” is introduced. It provides a logical view of Natural and non-Natural objects (such as job control files) as well as DDMs. A Natural application does not contain the objects, but it is a collection of links to Natural objects or “sub-applications” describing where the objects are stored. Natural objects or “sub-applications” can be linked to several applications.

In a SPoD scenario, the following types of applications exist:

- **base application**
- **compound application**

The space where the applications are maintained and displayed at the workstation is called the “**application workspace**”.

The application definitions are stored in the development server file (FDIC). When working with applications which are spread across different development servers, e.g. for building up a compound application, it is mandatory to use a unique development server file for all applications and the application server session.



**Note:** This application concept is supported by Predict.

## Some Arguments for Using the Term Natural Application

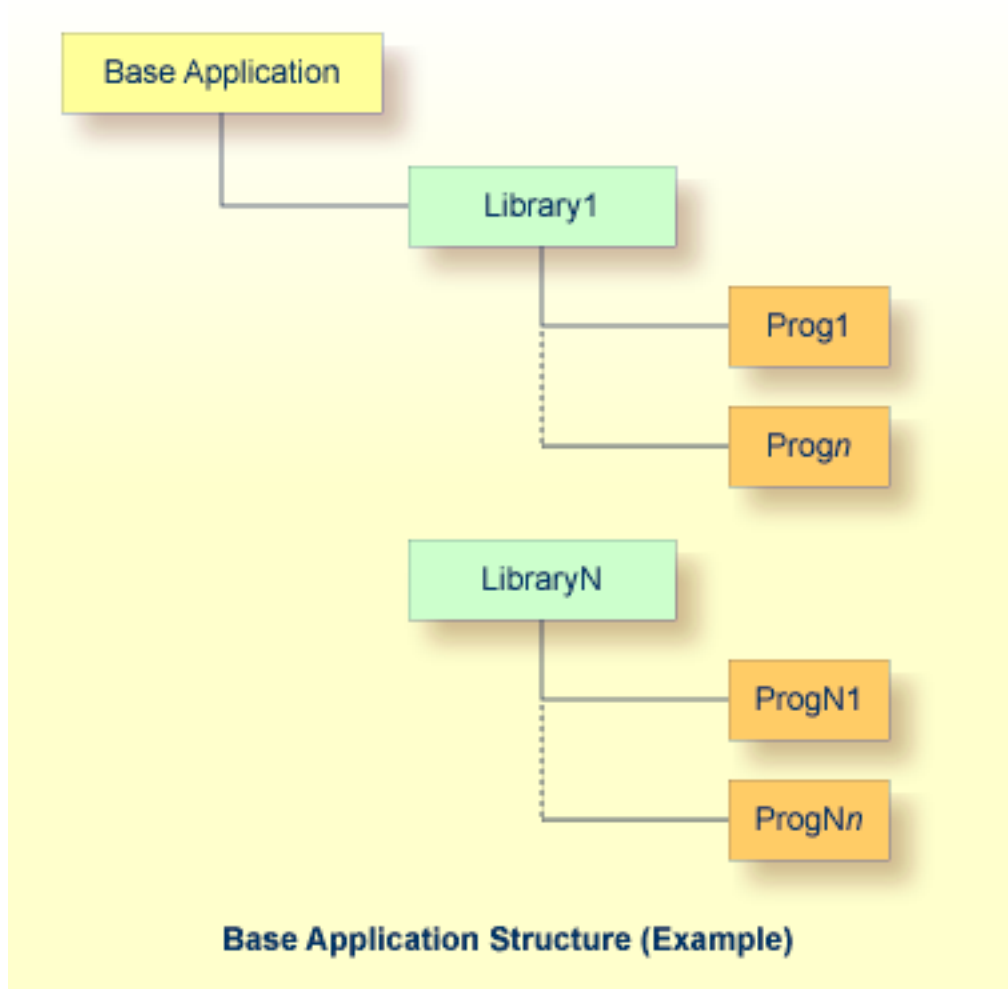
The following arguments may give you an idea why you should use the term “Natural application” and the concept that stands behind it:

- It groups the content of your libraries in a logical order based on the different applications you are maintaining that can be displayed in parallel to the library structure without using an additional tool.
- It focuses your view on the number of objects of a library you intend to work on. You need no longer navigate through all Natural objects located in one library, because you can edit the objects directly from the application workspace.
- It reduces the maintenance effort, because you are able to link all objects of different libraries to the application, especially those located in the steplibs.
- It gives you the opportunity to group an application into different “sub-applications” where each “sub-application” can be assigned to a member of the development team that is responsible for its development or maintenance.
- Thinking of client/server architecture, you are able to group your application into different “sub-applications”, and each “sub-applications” can be stored on a different platform at application runtime.

## Base Application

---

A base application is defined as a set of Natural object links. The associated Natural objects pertain to one specific Natural Development Server and are all located on the same FUSER system file.



The following information is stored for a base application:

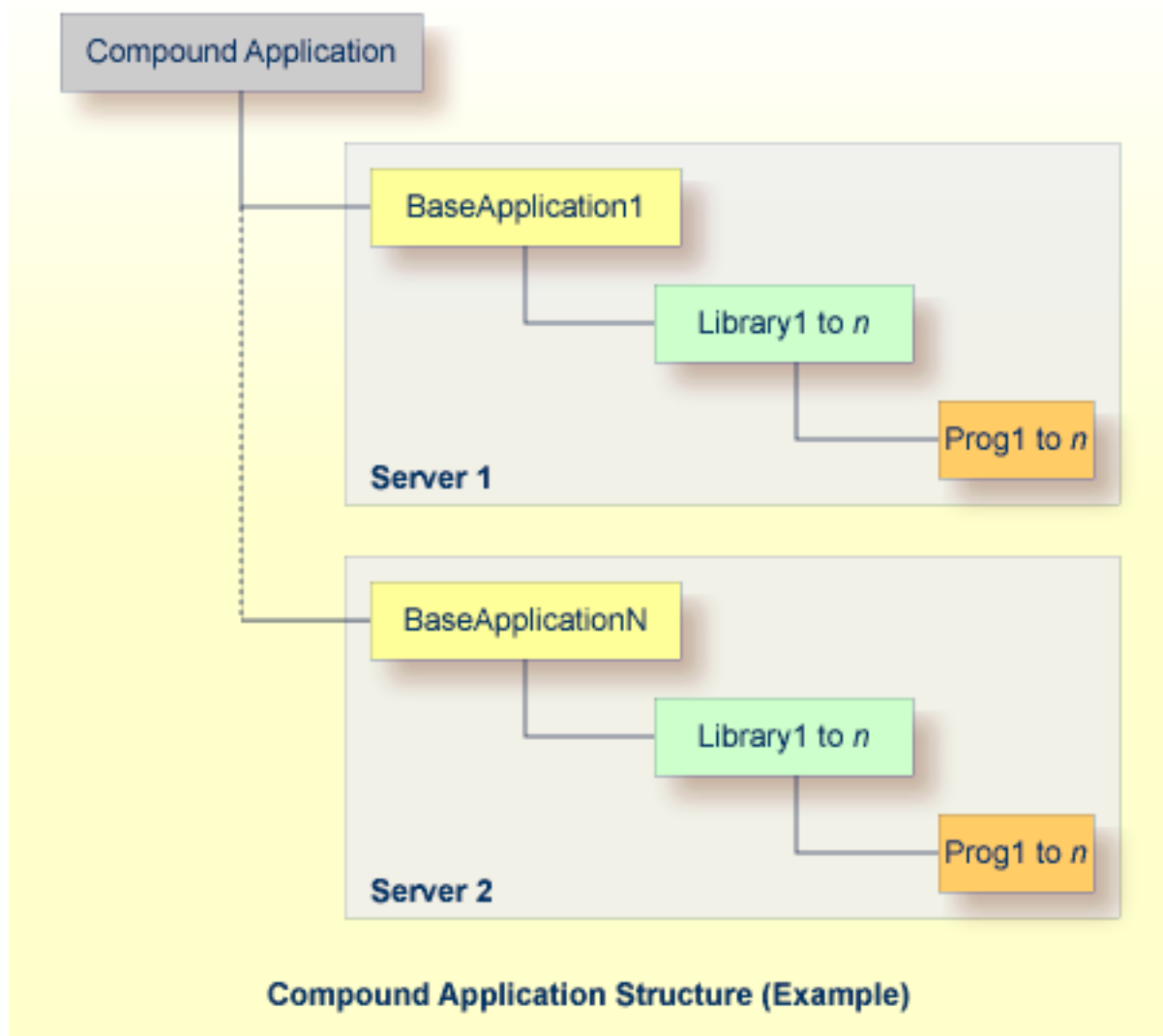
- Name of the application
- Description of the application (textual information only)
- Name and port ID of the Natural development server where the linked objects reside, i.e. the application environment
- Profile for the application environment
- Entry points describing the start objects of your application
- Identification of each object linked



## Compound Application

A compound application enables the application developer to combine several base applications. It is defined as a set of links to these base applications.

The base applications involved in a compound application can be spread across different FUSER system files or different development servers. Each base application may have a different parameter setting.



The following information is stored for a compound application:

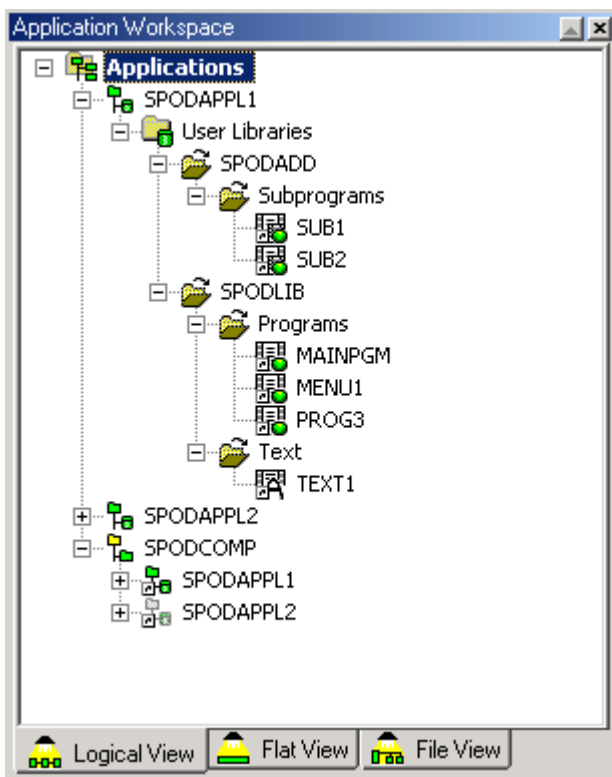
- Name of the application

- Description of the application
- Identification of each base application linked

## Application Workspace

---

The application workspace is an area on the Natural Studio screen (similar to Natural Studio's library workspace) where all known applications and their objects are shown in a view which complements the existing logical, flat and file views and displays a tree structure comprising all program objects belonging to an application, see example below.



# 7 Natural SPoD Architecture

---

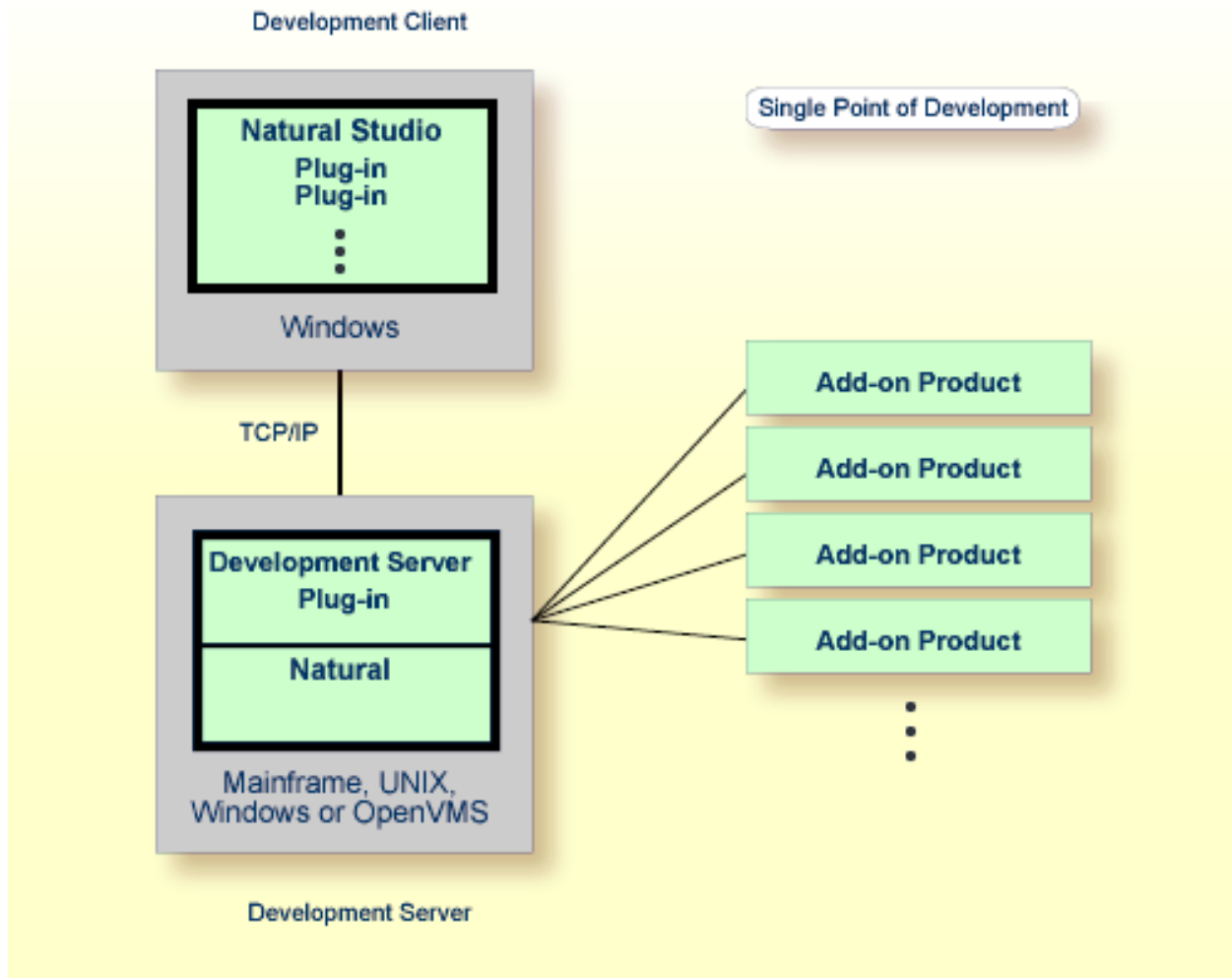
- Development Client ..... 31
- Development Server ..... 31
- Add-on Product ..... 31
- Security ..... 31

This chapter describes the system architecture of Natural's Single Point of Development (SPoD).

It covers the following topics:

SPoD allows centralized application development from a single Windows environment. You can use the features of the Natural Studio (provided with Natural for Windows) to develop and test Natural applications in a remote environment located on a mainframe, UNIX, OpenVMS or Windows platform.

SPoD is based on a client/server concept that allows one single remote development environment for all platforms. The graphic below illustrates this concept and the major components of the SPoD architecture:



## Development Client

---

Natural for Windows serves as the remote development desktop client for the target environment on a mainframe, UNIX, OpenVMS or Windows platform. The desktop client includes Natural Studio, which is the central workstation where users design applications. All Natural-related functions required for remote development can be performed from within Natural Studio.

## Development Server

---

Natural Development Server plug-in allows remote development for the Natural installation in the target environment on a mainframe, UNIX, OpenVMS or Windows platform. Natural on the target platform plus Natural Development Server plug-in constitute the development server.

## Add-on Product

---

Natural Studio provides plug-ins that can be used to integrate one or more Natural add-on products (for example, Predict) into a SPoD environment. The installation of the respective add-on product(s) in the development server environment is a prerequisite for Natural Studio plug-ins.

## Security

---

You can use Natural Security to protect the Natural Development Server environment, and Natural base applications and compound applications. For further information, refer to the *Natural Security* documentation.



# 8

## Natural SPoD Frequently Asked Questions

---

- Which versions of Natural and the related products are required to use the full set of functionality? ..... 34
- Can Natural objects of previous versions be used in a SPoD environment? ..... 34
- Do I need Predict? ..... 48
- Can I use Predict Case in a SPoD environment? ..... 34
- Can I use Natural Construct in a SPoD environment? ..... 35
- Can I use Predict Application Control in a SPoD environment? ..... 35
- What is the difference between the SPoD application concept and the earlier application shell concept in Natural for Windows? ..... 35

This chapter contains frequently asked questions regarding Natural Single Point of Development. Where applicable, reference will be made to relevant problem descriptions/solutions that are maintained in Software AG's Support Information System SAGSIS.

## **Which versions of Natural and the related products are required to use the full set of functionality?**

---

For creating and operating a Natural remote development environment, or for using additional SPoD functionality, the products as described under [http://documentation.softwareag.com/natural/spod\\_prereq/prereq.htm](http://documentation.softwareag.com/natural/spod_prereq/prereq.htm) are required.

## **Can Natural objects of previous versions be used in a SPoD environment?**

---

Yes, Natural objects of previous versions can be used in a SPoD environment without migration.

## **Do I need Predict?**

---

No, although the SPoD approach requires a Development Server File which is a central Natural system file structured like the FDIC file. This does not mean that Predict is a prerequisite for remote development, although Predict supports the SPoD application concept. Predict is needed to use the XRef functionality, the object description and/or the schema generation functionality.

## **Can I use Predict Case in a SPoD environment?**

---

Predict Case (PCA) will not be integrated into the SPoD concept. It can be invoked like any other mainframe Natural application via the terminal emulation feature of the remote development environment. However, Predict Case objects will not appear in the remote development tree view.



## **Can I use Natural Construct in a SPoD environment?**

---

Yes. Natural Construct (CST) for Mainframes and for UNIX are integrated into the SPoD concept. It can be invoked using the Program generation plug-in.

## **Can I use Predict Application Control in a SPoD environment?**

---

Predict Application Control (PAC) is not yet integrated into the SPoD concept. Currently, it can be invoked like any other mainframe Natural application via the terminal emulation feature of the remote development environment. However, PAC objects will not appear in the remote development tree view. A partial integration of PAC functionality is currently under consideration.

## **What is the difference between the SPoD application concept and the earlier application shell concept in Natural for Windows?**

---

The earlier Natural application shell was used with the frame gallery for developing, administering, and monitoring application solutions. Together, they provided an infrastructure in which numerous standards were defined and were implemented in the form of reusable components, such as dialogs and procedures.

The application concept introduced with Natural Single Point of Development is intended to address the structure of customer solutions so as to get a new logical view on the solution which is not limited by the boundaries of the Natural libraries.

---

# 9 First Steps with Natural Single Point of Development

---

This tutorial provides an introduction to Natural Studio which is part of the Single Point of Development (SPoD) concept. It is intended for developers who are not yet familiar with the Windows version of Natural.

This tutorial is not intended to be a comprehensive description of the full range of possibilities provided by Natural Studio. Therefore, explanations are kept to a minimum. For detailed information, see *Using Natural Studio* in the Natural for Windows documentation.

It is recommended that you follow the tutorial in the sequence indicated below.

- **Starting Natural**
- **Customizing the Natural Studio Window**
- **Local and Remote Environment**
- **Issuing Commands**
- **Handling Programs**
- **Locking and Unlocking**
- **Handling Applications**



# 10 Starting Natural

---

This tutorial explains how to work with a development server on a mainframe. It assumes the following:

- Natural for Windows has been installed with setup type “Development Client for Single Point of Development (SPoD)”.
- A development server has been installed on the mainframe. See the corresponding Natural Development Server documentation.
- The dataset `NDV $vr$ s.EXPL` has been loaded with the utility `INPL` on the development server. This dataset contains the sample library `SYSSPODA` that will be used in this tutorial.



**Note:** The terms  $vr$ s or  $v.r.s$  represent the corresponding product version number, where  $v$  represents the version number (one digit),  $r$  the release level (one digit) and  $s$  the system maintenance level (one digit). Example for the above mentioned dataset: `NDV $vr$ s.EXPL`.

- You have a basic understanding of how to use Microsoft Windows.
- You have already read *Introducing Natural Single Point of Development*.

A Natural folder automatically appears in the Programs folder of the Start menu after Natural has been installed. It contains the shortcuts for Natural, including Natural Studio and Help. If specified during installation, several shortcuts are also available on your Windows desktop.

## ▶ To start Natural

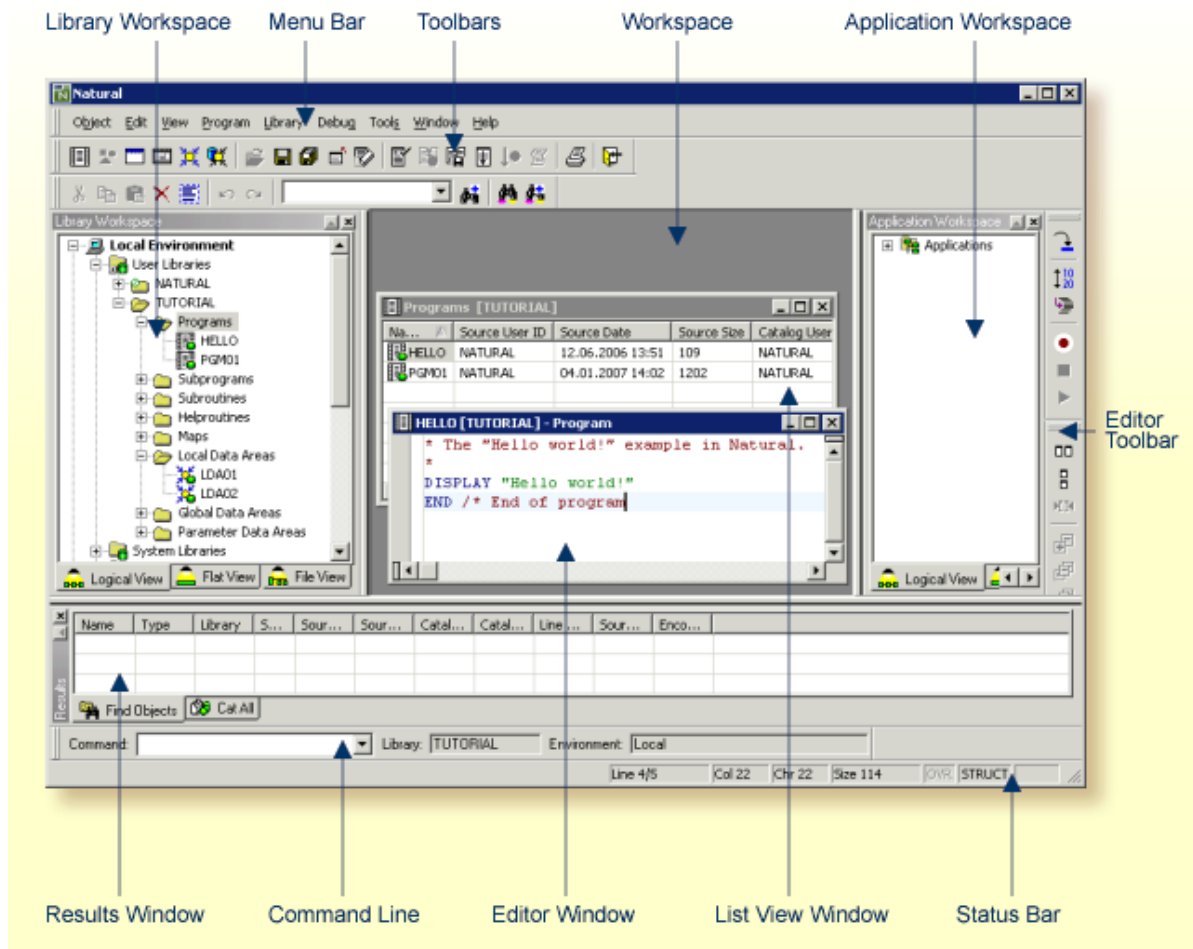
- From the Start menu, choose **Programs > Software AG Natural  $n.n$  > Natural**.

Or:

Use the following shortcut on your Windows desktop.



Natural Studio, the development environment for Natural, appears. When you start Natural for the very first time, Natural Studio shows only your local environment containing the library workspace. When several elements of the Natural Studio window have been activated and you are currently editing a program, the window may look as follows:



The different **environments** are explained later in this tutorial.

You can now proceed with the first exercise: *Customizing the Natural Studio Window*.

# 11 Customizing the Natural Studio Window

---

▪ Moving and Docking Windows .....	42
▪ Resizing Windows .....	44
▪ Using Different Views .....	44
▪ Displaying Additional Toolbars .....	47
▪ Displaying the Command Line .....	49

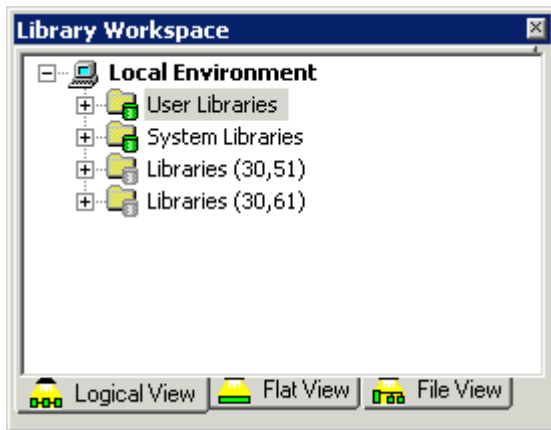
This chapter covers the following topics:

## Moving and Docking Windows

---

Using the mouse, you can drag each dockable window within the Natural Studio window to another position so that it is shown

- at another position within the Natural Studio window, or
- in a window of its own, for example:



You can move the window freely on your screen. You can move it back to the Natural Studio window (for example, back to its original position) so that it is no longer shown in a window. This process is called “docking”.

In the following exercises, you will undock, dock, hide and redisplay the library workspace window.

▶ **To move a docked window so that it is shown in a window of its own (undock)**

- 1 Move the mouse pointer to the title bar of the library workspace window.
- 2 Invoke the context menu by clicking the right mouse button.
- 3 Choose the command **Enable docking** to toggle the status.

Since docking has been disabled (a check mark is no longer shown next to the above command), the library workspace is now shown in a window of its own.

▶ **To move an undocked window back to the Natural Studio window (dock)**

- 1 Move the mouse pointer to the title bar of the undocked window and invoke the context menu.



- 2 Choose the command **Enable docking** to toggle the status.

A check mark should now be shown next to the above command.

- 3 Drag the title bar using the mouse to move the undocked window back to its original position in the Natural Studio window.

While dragging, an outline of the window is shown. The outline indicates the position at which the window can be docked.

- 4 Release the mouse button.

The window is now again docked within the Natural Studio window.



**Note:** You can prevent docking by pressing CTRL while moving the mouse.

#### ▶ To hide the library workspace window

- Choose the following button at the top right of the library workspace window (this can be either a docked or undocked window).



Or:

Move the mouse pointer to the title bar of the window, invoke the context menu and choose **Hide**.

The library workspace window is no longer shown in the Natural Studio window.

#### ▶ To toggle display of the library workspace window

- From the **View** menu, choose **Library Workspace**.

Or:

Press ALT+1.

When the library workspace window is displayed in the Natural Studio window, a check mark is now shown next to the **Library Workspace** command. The window is always restored in the same state it was before it was hidden (either docked or undocked).



**Note:** When you exit Natural Studio, the settings in the **View** menu as well as position and size of the Natural Studio window and its subwindows are stored in the Windows registry. The next time you start Natural Studio, its window is restored as it appeared when you last used it.

## Resizing Windows

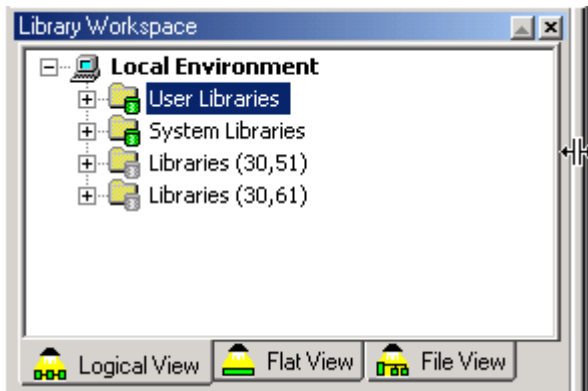
---

You can modify the size of each window within the Natural Studio window.

In this exercise, you will resize the library workspace window which is initially docked within the Natural Studio window.

▶ **To resize a docked window**

- 1 Move the mouse pointer over the right border of the library workspace window until the pointer changes, showing two arrows pointing in opposite directions.



- 2 Click and hold down the mouse button.
- 3 Drag the mouse to make the window larger or smaller.
- 4 When the window has the desired size, release the mouse button.

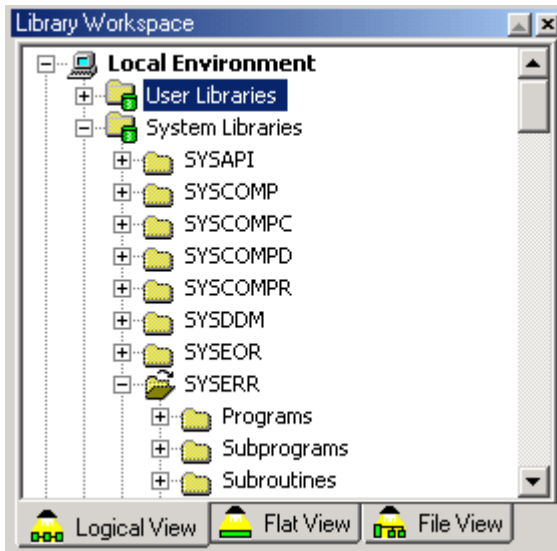
## Using Different Views

---

The library workspace window provides tabs for different views:

### ■ Logical View

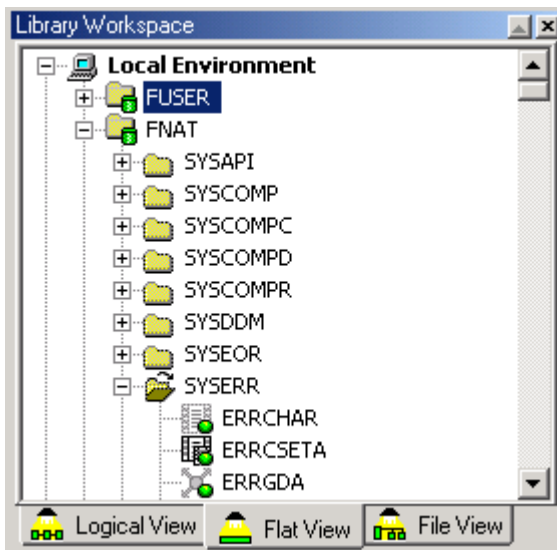
In the logical view, different nodes are provided for user libraries, system libraries and DDMs (FDIC). The objects in a library are grouped into different folders, according to their types.



For example, all programs are shown in a folder called "Programs". Thus, if you want to view the available programs in a library, you must first open the corresponding folder (for example, by clicking the plus sign next to the folder name). This is not required in the flat view.

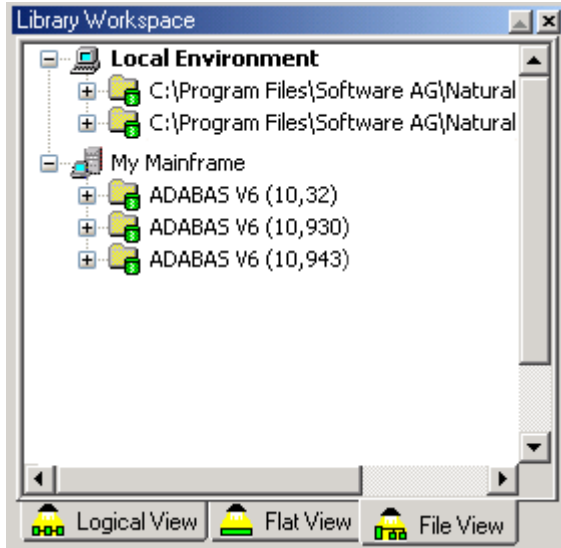
### ■ Flat View

In the flat view, different nodes are provided for the Natural parameters: FUSER, FNAT and FDIC. The objects in a library are displayed without any grouping. However, the object type is always indicated by the icon next to an object.







■ **File View**

In the file view, the name of each mapped remote database is shown with database ID and file number (**mapping** is explained later in this tutorial). Different folders are provided for each library (provided that the corresponding objects exist in this library): "Src" contains all Natural sources, "Gp" contains all generated programs (object code). Folders for other subdirectories (for example, "Res" for resources or "Err" for error codes) may also be available.



In the different views, icons are shown for the Natural objects. For example:

-  This icon is shown for a program.
-  When both source code and a generated program are available for an object, a green dot is displayed on the icon.
-  When only a generated program is available and no source code, the icon is gray.

 **Note:** If you want to see more icons for the different object types, expand the node for `SYSMAIN` in the logical view and then expand the nodes for the different folders.

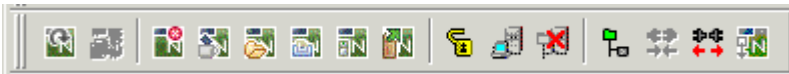
When you resize the library workspace window, it may happen that the window is not wide enough to display all tabs at the bottom. In this case, arrow buttons are provided. To display another view, you can either enlarge the window and select the corresponding tab, or you can use an arrow button to display the corresponding view.




## Displaying Additional Toolbars

Natural Studio provides several toolbars. Only a few of them are initially shown in the Natural Studio window.

In this exercise, you will activate the Tools toolbar which provides buttons for commands that you will use later in this tutorial:



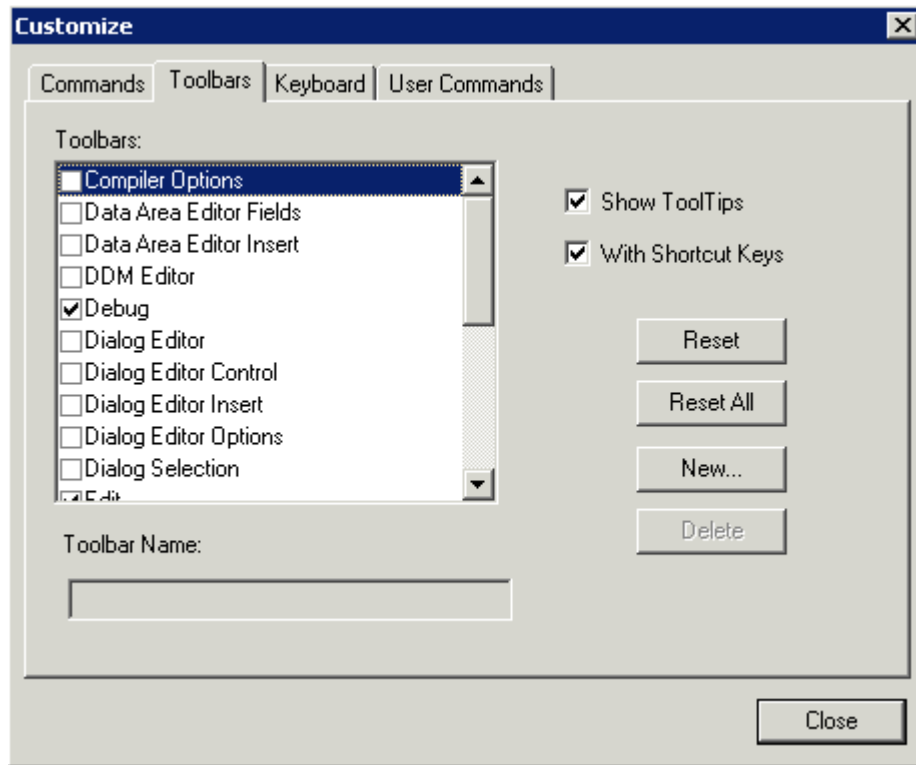
 **Note:** You can undock and dock each toolbar that is shown in the Natural Studio window as described above for the library workspace window.

### ▶ To display the Tools toolbar

- 1 From the **Tools** menu, choose **Customize**.

The **Customize** dialog box appears.

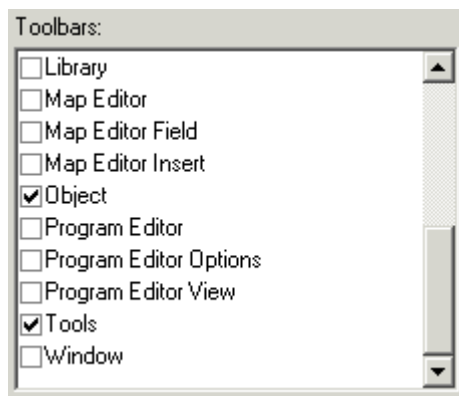
- 2 Choose the **Toolbars** tab.



- 3 To proceed to the bottom of the list box, use the scroll bar or the down arrow.

Initially, the **Tools** check box is not selected. This means that the Tools toolbar is currently not shown in the Natural Studio window.

- 4 Select the **Tools** check box so that a check mark is shown in the box.

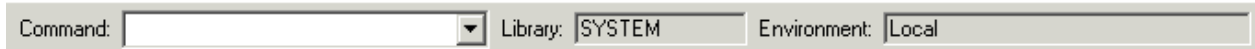


Each toolbar that you activate is immediately shown in the Natural Studio window. It is always shown in its last state (either docked or undocked).

- 5 Choose the **Close** button.

## Displaying the Command Line

You can issue all Natural commands directly from the command line. An [example](#) is given later in this tutorial.



 **Note:** You can undock and dock the command line as described above for the library workspace window.

Initially, the command line is not shown.

### ▶ To toggle command line display

- From the **View** menu, choose **Command Line**.

Or:

Press ALT+3.

When the command line is displayed in the Natural Studio window, a check mark is shown next to the **Command Line** command.

You can now proceed with the next exercise: [Local and Remote Environment](#).





# 12 Local and Remote Environment

---

- Checking the Environment ..... 52
- Connecting to a Development Server for the First Time ..... 52
- Connecting to a Previously Mapped Development Server ..... 54
- Mapping versus Connecting an Environment ..... 56
- Logging on to a Library ..... 56

A Natural development environment contains all application components such as parameter modules, system files and buffer pool.

This chapter covers the following topics:

## Checking the Environment

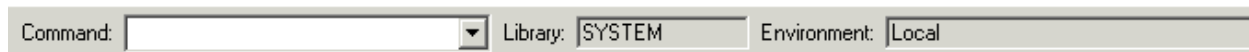
---

SPoD supports the following types of environment:

- local environment on a workstation (this is also the runtime environment of Natural Studio)
- remote environment on a development server

You can check which of these two environments is currently active. The active environment is always indicated in the command line, next to the **Command** drop-down list box. When the command line is not shown, you can display it as described previously in *Displaying the Command Line*.

In the example below, the local environment is active and you are currently logged on the the library SYSTEM.



All commands that you issue are always applied to the active environment. When you edit a Natural object, the corresponding editor is invoked and the object is taken from the active environment. When you execute an object, it is executed in the active environment.

Only one environment can be active at one point in time.


## Connecting to a Development Server for the First Time

---

In order to perform remote development, you have to activate a remote Natural environment. You do this by connecting to the appropriate Natural development server. Each Natural development server provides all remote services (such as access or update) for a specific FUSER system file.

If you want to connect to a development server for the very first time, you have to map it as described below. Once you have connected to a development server, a node for this development server session is automatically shown in the tree the next time you invoke Natural Studio.

If you do not know the name and port number for your development server, ask your administrator before proceeding with the next exercise.

 **Note:** It is possible to map the same development server more than once, for example, if you want to have development server sessions with different session parameters. To switch to another session, you simply select the corresponding node in your library workspace.

▶ **To connect (map) to a development server**

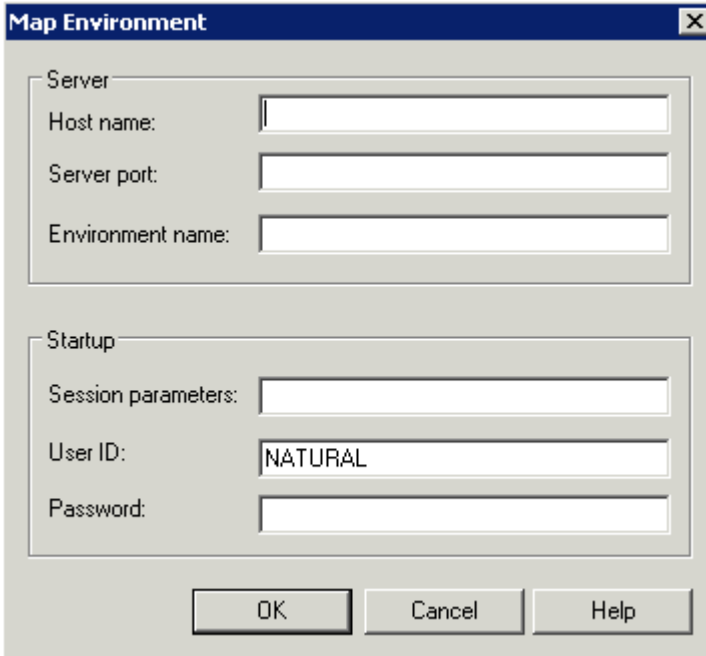
- 1 From the **Tools** menu, choose **Map > Environment**.

Or:

Use the following toolbar button:



The **Map Environment** dialog box appears.



- 2 In the **Host name** text box, enter the name of the development server.
- 3 In the **Server port** text box, enter the TCP/IP port number of the development server.
- 4 Optionally. In the **Environment name** text box, enter the name that is to appear in the tree (for example, "My Mainframe"). If you leave this text box blank, a combination of server name and port number is shown in the tree.
- 5 If dynamic parameters are required for your development server, specify them in the **Session parameters** text box. Otherwise, leave this text box blank.

- 6 Your user ID is automatically provided. If you want to map the development server using a different user ID, specify it in the **User ID** text box. This is useful, for example, when Natural Security is active on the development server and administrator rights are to be used.
- 7 If Natural Security is installed on the development server, specify the required password in the **Password** text box. Otherwise, leave this text box blank.
- 8 Choose the **OK** button.

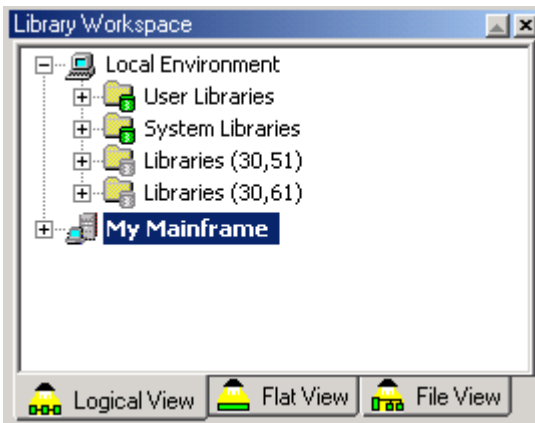
When the connection has been established, all libraries (according to the security profile) for this session are shown in your library workspace. You are automatically logged on to your default library.

The command line now shows the name of the library that is currently selected in the tree and the name of the active environment (i.e. the name you specified for the development server).

## Connecting to a Previously Mapped Development Server

---

Once you have connected to a development server, its name is shown as a node in the tree of your library workspace.



You will now learn how to disconnect a development server and how to connect it again.

### ► To disconnect a connected development server

- 1 In the library workspace, select the node for a connected development server.
- 2 From the **Tools** menu, choose **Disconnect**.

Or:

Invoke the context menu and choose **Disconnect**.

Or:

When the Tools toolbar is shown, choose the following toolbar button.



When a development server has been disconnected, its state is shown as follows in the tree:



### ▶ To connect a disconnected development server

- 1 In the library workspace, select the node for a disconnected development server.
- 2 From the **Tools** menu, choose **Connect**.

Or:

Invoke the context menu and choose **Connect**.

Or:

When the Tools toolbar is shown, choose the following toolbar button.



The **Map Environment** dialog box appears. It shows the information that you have provided when you have mapped the development server.

- 3 If required, specify the Natural Security password in the **Password** text box. Otherwise, leave this text box blank.

You need not specify any other information.

- 4 Choose the **OK** button.

When the connection has been established, all libraries (according to the security profile) for this session are shown in your library workspace. You are automatically logged on to your default library.



**Note:** When you invoke Natural Studio, a **Map Environment** dialog box appears for each development server which was active (that is: connected) when you terminated Natural Studio. You can then decide whether you want to connect to this development server again.

## Mapping versus Connecting an Environment

A remote environment is mapped, if the environment is shown in the tree. A remote environment is connected if an appropriate session on the server exists. The **Map > Environment** command automatically includes a **Connect** command.

It is possible to remove a development server from the tree: Select the server, invoke the context menu and choose the **Unmap** command. The **Unmap** command automatically includes a **Disconnect** command. If the development server is still connected when unmapping it, the running session is shut down. All editor windows that have been opened for this server session are closed.

Connected environments can be disconnected, and disconnected environments can be connected.

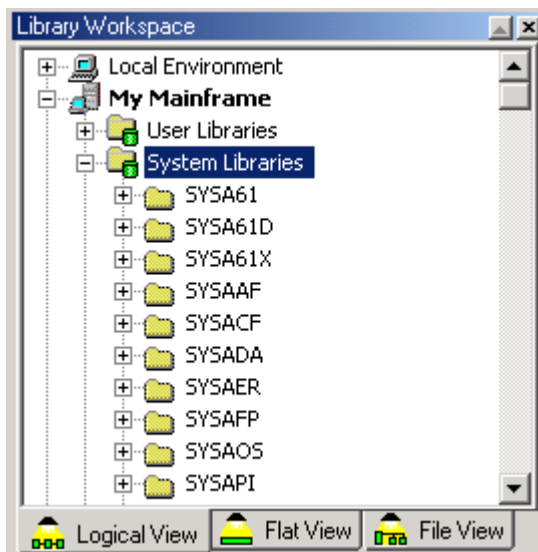
## Logging on to a Library

You will now log on to the library SYSSPODA which contains the objects that will be used in this tutorial.

### ▶ To log on to a library

- 1 Make sure that the logical view is active in your library workspace.
- 2 Under the node for the development server session that you have just mapped, use the plus sign next to the node "System Libraries" to expand the tree.

The system libraries are now listed.



- 3 Scroll down the list until the library SYSSPODA is shown in the tree.
- 4 To log on to the library, simply select the library name SYSSPODA.

The status line at the bottom of the Natural Studio window informs you that this operation has been sent to the server.

- 5 To display the contents of the library, use the plus sign next to the library name SYSSPODA to expand the tree.

You can now proceed with the next exercise: *Issuing Commands*.





# 13 Issuing Commands

---

▪ Context Menus .....	60
▪ Creating User Libraries .....	63
▪ Copying and Moving Objects .....	64
▪ Deleting Objects .....	66
▪ Cataloging Objects .....	67
▪ Displaying the Last Commands .....	69
▪ Listing Objects .....	71
▪ Invoking Terminal Emulation .....	73

Natural Studio commands are often issued via context menus as explained in this section. Several important context menus are shown in this section, and copying and moving via drag-and-drop is explained.

Natural Studio also provides a command line in which you can directly enter Natural system commands. The prerequisite is that a certain logical context is given. For example, the `SAVE` command can only be executed when a source is currently shown in the editor.

A graphical user interface is not provided for all system commands that are available in a remote development environment. When you issue such a command in the command line, terminal emulation will be started in a separate window, showing the corresponding character screen. You can then work in the same way as, for example, on the mainframe.

Certain system commands (for example, the mainframe command `EDT`) are not available in Natural Studio and can therefore not be executed from the command line. See also the section *SPoD-Specific Limitations and Considerations* in your *Natural Development Server* documentation.

For further information on system commands, refer to Natural's *System Commands* documentation.

This chapter covers the following topics:

## Context Menus

---

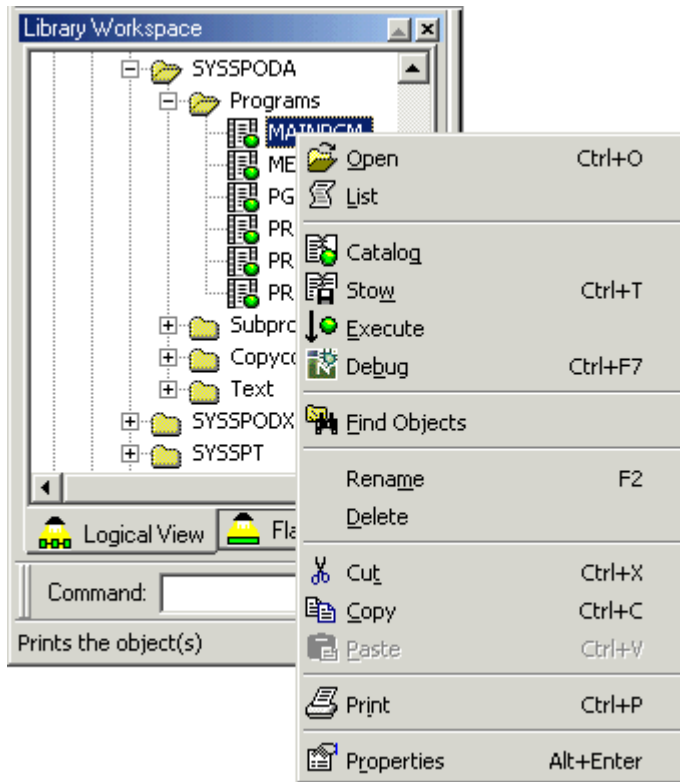
Context menus are invoked using either the *right* mouse button or by pressing `SHIFT+F10`. The commands provided in the context menu depend on the object or the position within the Natural Studio window that has been selected.



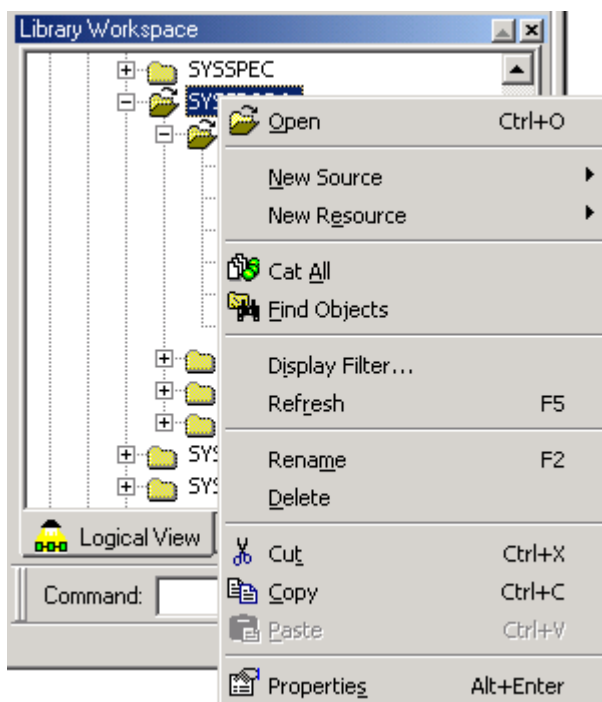
**Note:** The menu bar at the top of the Natural Studio window can be customized. Thus when a menu is not shown in the menu bar, you can still issue the commands that apply to the selected object from the context menu.

When you select an element in a tree and invoke the context menu, all valid commands for this element are shown.

- The following example shows a context menu that has been invoked for a selected a Natural object.

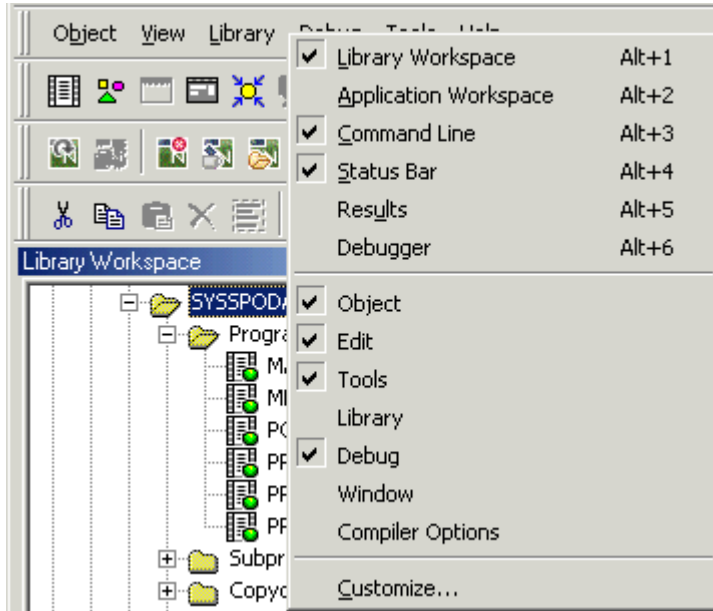


- A different context menu is shown when you select a library. For example:

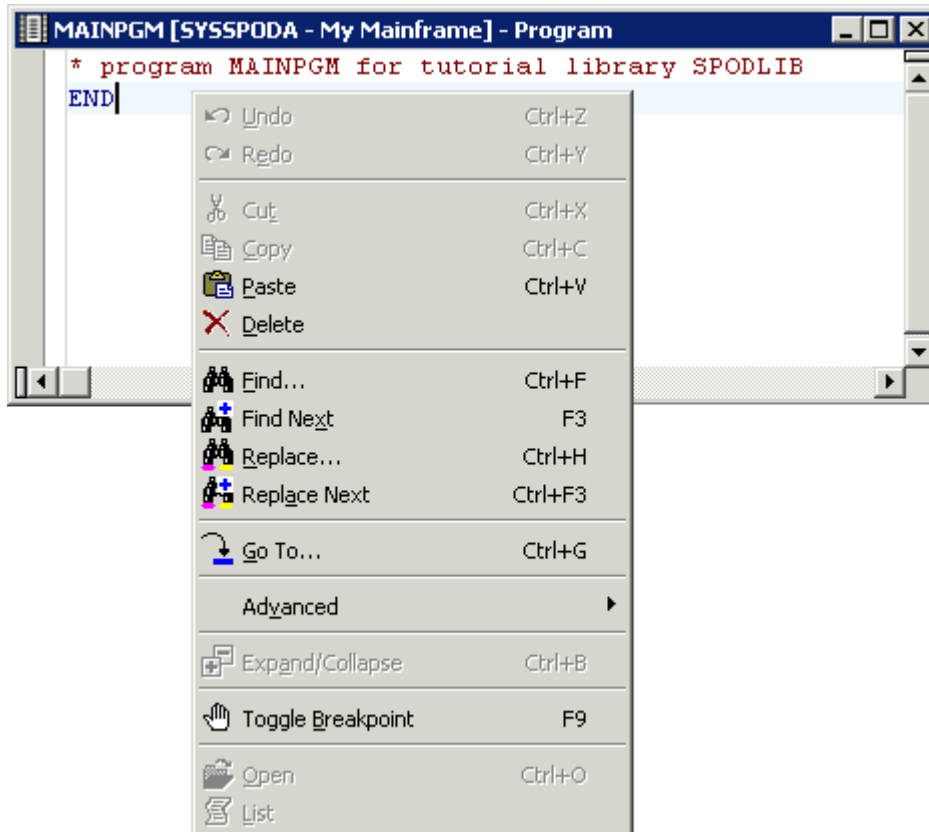



When you click any other position outside the library workspace with the right mouse button, different commands are shown in the context menu.

- The following example shows a context menu that has been invoked by selecting the menu bar or the workspace with the right mouse button:



- A different context menu is shown in a program editor window. For example:



 **Note:** When a toolbar button or shortcut key is available, this information is shown in the context menu. Commands that are dimmed, are currently not available.

## Creating User Libraries

You will now create the following user libraries that will be used later in this tutorial:

- SPODLIB
- SPODADD
- SPODTEST

Make sure to create these libraries in an environment in which a Natural development server has been installed.

▶ **To create the user libraries**

- 1 Depending on the current view, select one of the following (below the node of the development server to which you have previously connected):

Logical view: "User Library"

Flat view: "FUSER"

File view: the name of the database

- 2 From the **Library** menu, choose **New**.

Or:

Select the node, invoke the context menu and choose **New**.

A new library with the default name `USRNEW` is now shown in the tree.



The default name is selected so that you can immediately enter a new name. Any text you enter automatically deletes the selection.

- 3 Specify `SPODLIB` as the name of the library.

- 4 Press `ENTER`.

Or:

Click any other position in the library workspace.

- 5 Create the libraries `SPODADD` and `SPODTEST` as described above.

## Copying and Moving Objects



---

With the following exercises, you will

- copy the contents of the system library `SYSSPODA` to your user library `SPODLIB`,
- move all subprograms from library `SPODLIB` to library `SPODADD`,
- copy the program `PGMCHECK` from library `SPODLIB` to library `SPODTEST`.

These objects will be used later in this tutorial.

Different methods can be used to copy and move objects:

- menu commands (see **Copy** and **Paste** in the first exercise in this section)
- shortcut keys (see CTRL+C and CTRL+V the first exercise in this section)
- toolbar buttons (see  and  in the first exercise in this section)
- drag-and-drop (see the last two exercises in this section)

▶ **To copy the contents of the system library SYSSPODA to the user library SPODLIB**

- 1 Select the system library SYSSPODA.
- 2 Invoke the context menu and choose **Copy**.

Or:

Press CTRL+C.

Or:

Choose the following toolbar button:



You can now paste the contents of the library to your user library.

- 3 Scroll to the user library SPODLIB that you have previously created.
- 4 Select the user library SPODLIB.
- 5 Invoke the context menu and choose **Paste**.

Or:

Press CTRL+V.

Or:

Choose the following toolbar button:



All objects from the system library SYSSPODA are now copied to your user library SPODLIB.

▶ **To move all subprograms from the library SPODLIB to the library SPODADD using drag-and-drop**

- 1 Make sure that the logical view is active.
- 2 Click the plus sign next to the library SPODLIB to expand the tree.

- 3 Click the "Subprograms" node and hold down the mouse button.
- 4 Drag the selected object onto the name of the node SPODADD.
- 5 Release the mouse button.

All subprograms are now moved to the library SPODADD.

▶ **To copy the program PGMCHECK from the library SPODLIB to the library SPODTEST using drag-and-drop**

- 1 Make sure that the flat view is active. Thus you need not open the folder containing all programs which is provided in the logical view.
- 2 Under the node FUSER, click the plus sign next to the library SPODLIB to expand the tree.
- 3 Click the program PGMCHECK and hold down the mouse button.
- 4 Hold down CTRL.
- 5 Drag the selected object onto the name of the node SPODTEST.
- 6 Release the mouse button and then CTRL. The program is now copied to the library SPODTEST.



**Note:** When you move an object, you cut it at its original position and paste it at a new position. To copy an object, press CTRL while dragging. This does not cut the object at its original position.

## Deleting Objects

---

Since you have copied the program PGMCHECK in the previous exercise, it is available in two libraries. You will now delete this program from the library SPODLIB.

▶ **To delete the program PGMCHECK from the library SPODLIB**

This exercise assumes that the flat view is still active.

- 1 Under the node for the library SPODLIB, select the program PGMCHECK.
- 2 Invoke the context menu and choose **Delete**.

A dialog box appears, asking whether you really want to delete the program.

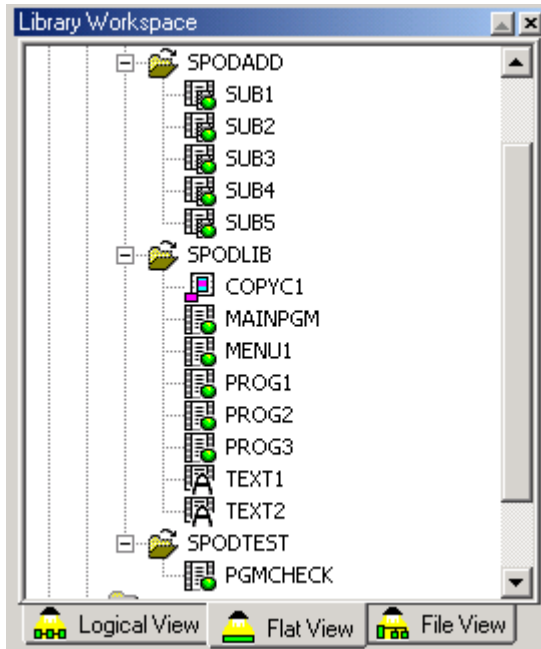
- 3 Choose the **Yes** button to delete the program.



**Note:** You can switch off the display of the delete messages. From the **Tools** menu, choose **Options**. In the resulting dialog box, display the **Workspace** tab, deselect the **Display delete messages** check box and choose the **OK** button.



With all nodes expanded in the flat view, your new libraries should now look as follows:



## Cataloging Objects

CATALL is one of the Natural system commands for which a graphical user interface is provided in Natural Studio.

You will now catalog the objects of the libraries you have just created.

### ▶ To catalog all objects in a library

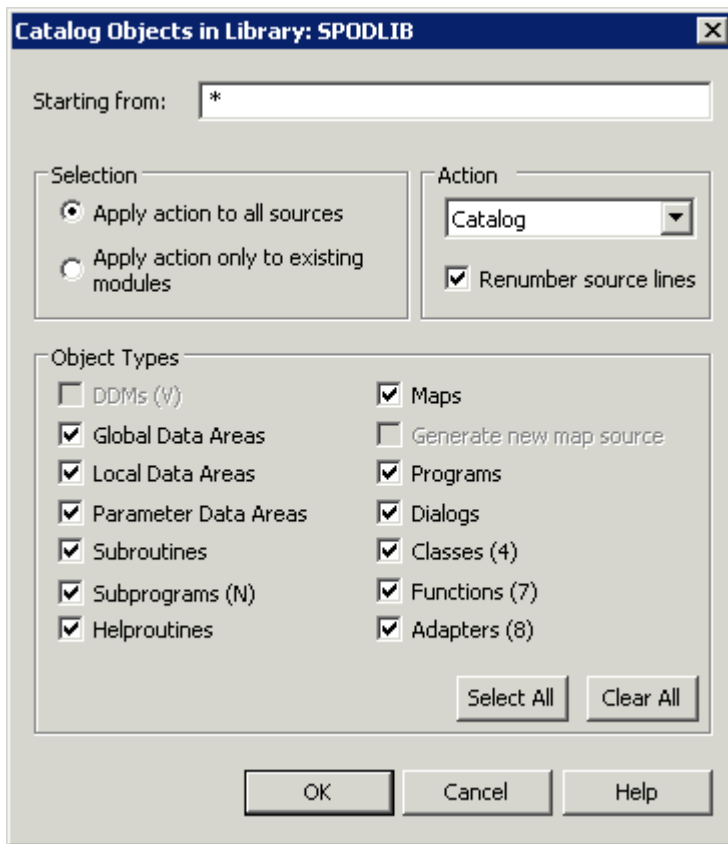
- 1 Select the library SPODLIB.
- 2 From the **Library** menu, choose **Cat All**.

Or:

Invoke the context menu and choose **Cat All**.

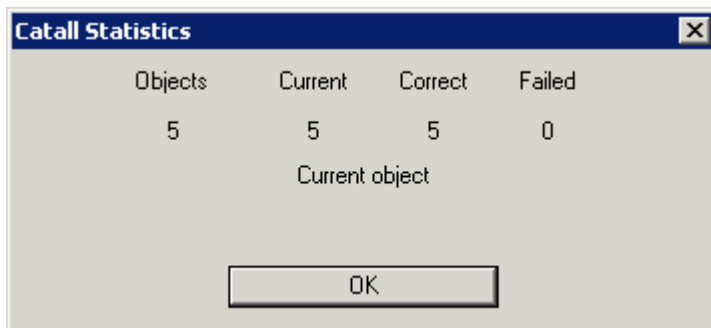
The **Catalog Objects in Library: *libraryname*** dialog box appears.

- 3 Make sure that the option button **Apply action to all sources** is selected. Leave the check boxes in the **Object Types** group box with their default settings.



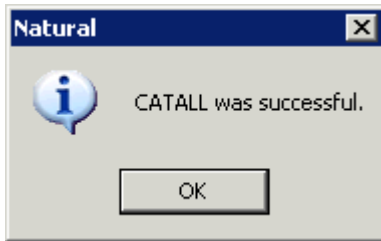
- 4 Choose the **OK** button.

A dialog box appears with statistics about the performed command.



- 5 Choose the **OK** button to close the dialog box.

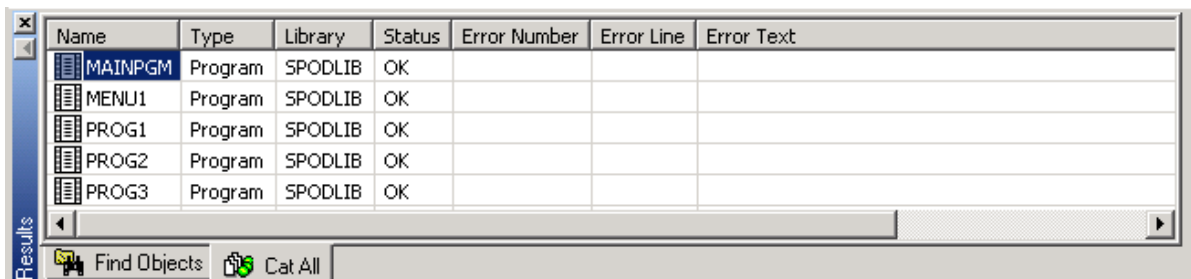
A dialog box appears, informing you that cataloging was successful.



**Note:** You can switch off display of success messages. From the **Tools** menu, choose **Options**. In the resulting dialog box, display the **Workspace** tab, deselect the **Display success messages** check box and choose the **OK** button.

- 6 Choose the **OK** button to close the dialog box.

The results window appears. "OK" in the **Status** column indicates that cataloging was successful for an object.



If an object could not be cataloged, an error number (reason why the object could not be cataloged) and an error line (the line position in source code where the error can be found) is shown in the results window.

- 7 Repeat the above steps for the libraries **SPODADD** and **SPODTEST**.

## Displaying the Last Commands

Natural commands can be executed from the drop-down list box of the command line.

Natural Studio saves each character string you enter in the command line for the current session. The drop-down list box contains your last entries. You can select an entry and press `ENTER` to execute it once more.



When you enter the first character of a command that you have previously entered, the corresponding command is automatically copied to the command line. In this case, you just have to press `ENTER` to execute it.

When the mouse pointer is positioned on the command line, you can invoke a context menu. Using the commands from this context menu, you can, for example, paste a text string into the command line.

The system command `LAST *` is an example of a Natural command for which a menu command is not provided with Natural Studio. In contrast to the commands that you enter in the command line, the `LAST *` command displays all system commands in the order in which they were entered in a dialog box. It considers all Natural commands that were issued during a session: for example, when you select a library to log on to it, or when you execute a program using a menu command, toolbar button or shortcut key. Furthermore, it allows you to select several commands to be executed one after the other (see below).

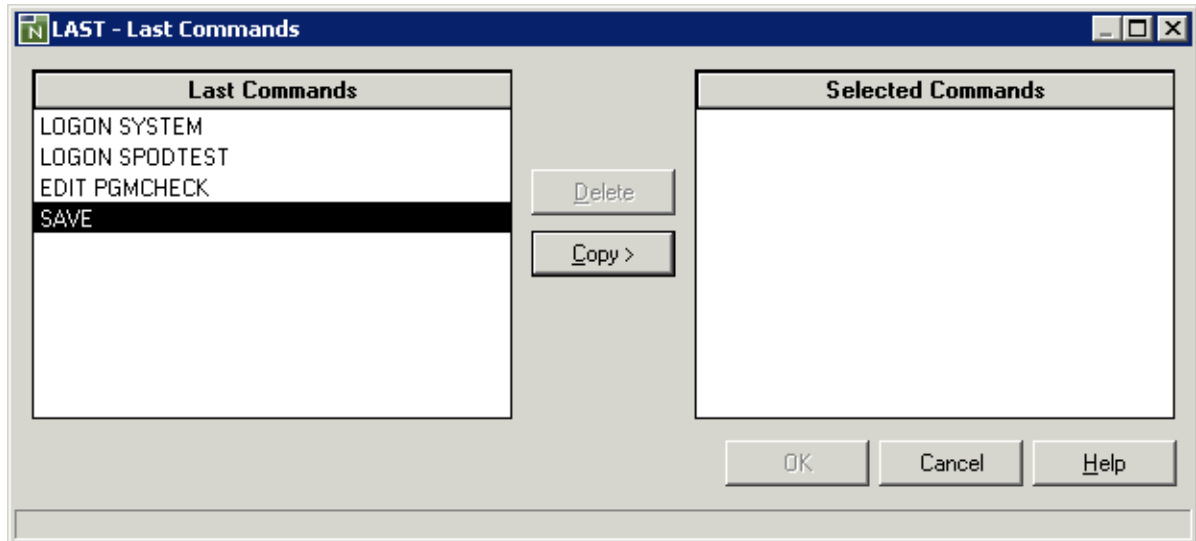
If the command line is not shown in the Natural Studio window, display it as described previously in [Displaying the Command Line](#).

### ▶ To execute the `LAST *` command from the command line

- 1 Enter the following command in the command line and press `ENTER`:

```
LAST *
```

The following dialog box is now shown:



- 2 On the left, select the first command that you want to execute.
- 3 Choose the **Copy** button.

The selected command is now shown on the right.

- 4 Optionally. Modify the command on the right (for example, specify another library name with the LOGON command).
- 5 Repeat the above steps to copy all commands that you want to execute to the right. The commands will be executed in the same sequence in which they appear in the list.
- 6 Choose the **OK** button to execute all selected commands one after the other.

## Listing Objects

You will now list the objects contained in one of the libraries you have previously created. You can do this in two different ways:

- choose **Open** from the context menu, or
- enter the `LIST *` command in the command line

When using the **Open** command, the content of the window depends on the current view. For example, in the logical view, all folders are shown, and in the flat view all objects are shown.

When entering `LIST *` in the command line, all objects of a library are shown. The content of the resulting window is always the same. The current view is not considered. An advantage of the `LIST` command is that you can also specify that only objects starting with a specific letter are shown (e.g. `LIST P*`).

► To list all objects in a library with the **Open** command

- 1 Make sure that the logical view is active for this exercise.
- 2 Select the library `SPODLIB` in the library workspace.
- 3 Invoke the context menu and choose **Open**.

The following window appears displaying a list of folders.

Type	Number	Size
Programs	5	12795
Copycodes	1	71
Text	2	98

- 4 Select the "Programs" folder and press `ENTER` to display its contents.

Or:

Double-click the "Programs" folder.

Name	Source User ID	Source Date	Source Size	Catalog User ID	Catalog Date	Catalog Size	Mode	Encoding
MAINPGM	NATURAL	2006-08-31 15:06	57	NATURAL	2006-08-31 15:08	2784	Structured	
MENU1	NATURAL	2006-08-31 15:07	55	NATURAL	2006-08-31 15:08	2784	Structured	
PROG1	NATURAL	2006-08-31 15:07	73	NATURAL	2006-08-31 15:08	2972	Structured	
PROG2	NATURAL	2006-08-31 15:07	55	NATURAL	2006-08-31 15:08	2784	Structured	
PROG3	NATURAL	2006-08-31 15:07	55	NATURAL	2006-08-31 15:08	2784	Structured	

- 5 To change the display sequence, click the column header **Source Date**.

An arrow is now shown in this column header indicating the current display sequence.

- 6 Click the column header **Source Date** once more.

This toggles between ascending and descending display sequence.

- 7 Close each of these two windows by choosing the standard close button at the top right of a window.

## Invoking Terminal Emulation

---

SYSBPM is a Natural mainframe utility for which a graphical user interface is not provided in Natural Studio. When you invoke this utility, its character-based mainframe screen is shown either in a terminal emulation window or in a window of the Web I/O Interface.

Both the terminal emulation window and the Web I/O Interface are used in a remote development environment to display non-GUI information. This is different from the local environment where the output is shown in an output window.

The terminal emulation window is not Unicode-enabled. The Web I/O Interface is able to display information which contains Unicode characters. The window which will appear at your site depends on the parameters which have been set on the Natural development server.

This section assumes that your remote development environment has been set up to use the terminal emulation window.



**Note:** As long as the terminal emulation window is active, you cannot work with Natural Studio.

### ▶ To invoke the SYSBPM utility in a terminal emulation window

- 1 Enter the following command in the command line:

```
SYSBPM
```

The terminal emulation window appears.

```

Terminal Emulation
Session Edit View Help
11:04:53          ***** NATURAL SYSBPM UTILITY *****          2007-11-06
BPNAME QA42GBP          - Main Menu -          Type Global Nat
BPPROP OFF          Loc DAEF QA42          Preload QA42GBPL

          Object Functions          Object Pool Statistics

L List Objects          A Buffer Pool
D Delete Objects          C BP Cache
I Directory Information
H Hexadecimal Display
W Write to Work File
X Display Sorted Extract
? Help
. Exit

          Other Functions

S Select Buffer Pool
B Blacklist Maintenance
P Preload List Maintenance

Code ..  Library ... * _____
          Object .... * _____
          DBID ..... 0 _____ FMR .. 0 _____ Object Pool ... * (B,C,*)

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---
          Help          Exit Last          Flip          Canc

4AÛ          17,014

```

You can now use the utility in the same way as on the mainframe. Command buttons are provided for frequently-used PF keys. When they are not shown, you can display them as described with the following step.

- 2 From the **View** menu, choose a PF-key set (for example, **PF Keys 1-12**) to display the command buttons for the PF keys.
- 3 To exit terminal emulation, issue the `EXIT` command (either by issuing it in the command line of the mainframe screen or by pressing the corresponding PF key).

You can now proceed with the next exercise: *Handling Programs*.



# 14 Handling Programs

---

- Creating a New Program ..... 76
- Stowing a Program ..... 78
- Executing a Program ..... 79
- Debugging a Program ..... 80

This chapter shows the most important steps for handling programs, including testing and debugging.

For more information, see *Program Editor* in the *Editors* documentation of Natural for Windows.

For detailed information on the debugger and remote debugging, see the *Debugger* documentation of Natural for Windows.

This chapter covers the following topics:

## Creating a New Program

---

You will now create a small program which prompts the user for input and displays this input. You can create this program in any library that you are allowed to use in the remote development environment.

### ▶ To create a program

- 1 Select the name of the library in the remote development environment in which you want to store the new program.
- 2 From the **Object** menu, choose **New > Program**.

Or:

Invoke the context menu and choose **New Source > Program**.

Or:

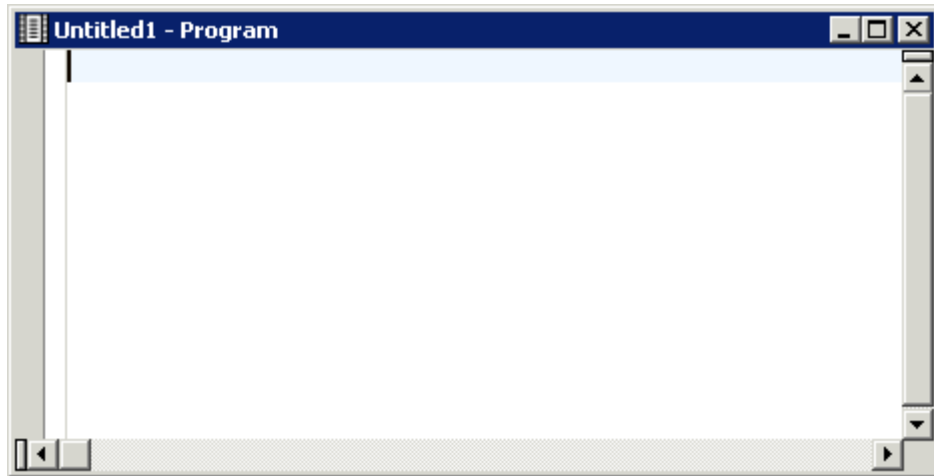
Press CTRL+N.

Or:

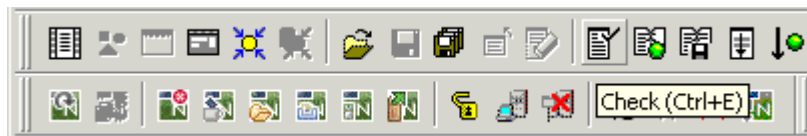
Choose the following toolbar button:



An empty program editor window appears.



The status bar at the bottom of the Natural Studio window now shows additional information (line and column in which the cursor is currently positioned and the current size of the program). When the program editor window is the active window, all toolbar buttons that apply to a program are activated. When you move the mouse over a toolbar, tooltips appear. For example:



- 3 In the program editor, enter the following code:

```
DEFINE DATA LOCAL  
1 A (A5)  
END-DEFINE  
INPUT 'Enter' A  
DISPLAY A  
END
```

## Stowing a Program

---

You will now stow the program you have just created.

Since it is possible to edit several programs at the same time (this is not possible, for example, on the mainframe), you have to make sure that the editor window for the program that you want to stow is the active window. To activate a window, you simply have to select it.

▶ **To stow a program**

- 1 Make sure that the editor window containing your new program is active.
- 2 From the **Object** menu, choose **Stow**.

Or:

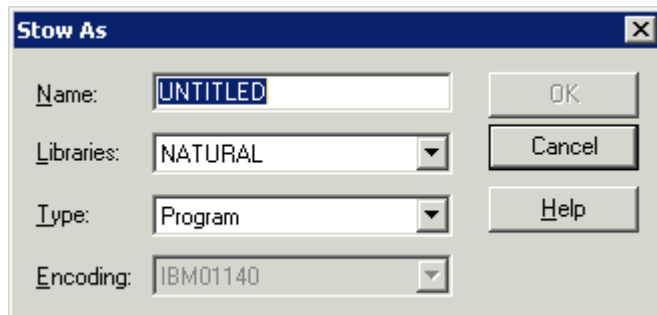
Press CTRL+T.

Or:

Choose the following toolbar button:



The **Stow As** dialog box appears.



- 3 In the **Name** text box, enter "DEMO" as the name of your program.
- 4 To stow the program in the current library, choose the **OK** button.

If you have not switched off the success messages, a dialog box appears, informing you that the stow operation was successful. When this dialog box appears, press ENTER to close it.

The new program is now shown in the library workspace. The title bar of the program editor window now shows the program name, the library in which it is stored, and the environment name of the current development server.

- 5 Close the program editor by choosing the standard close button at the top right of the program editor window.

## Executing a Program

---

You will now execute your DEMO program.

### ▶ To execute a program

- 1 In the library workspace, select the program DEMO.
- 2 Invoke the context menu and choose **Execute**.

Or:

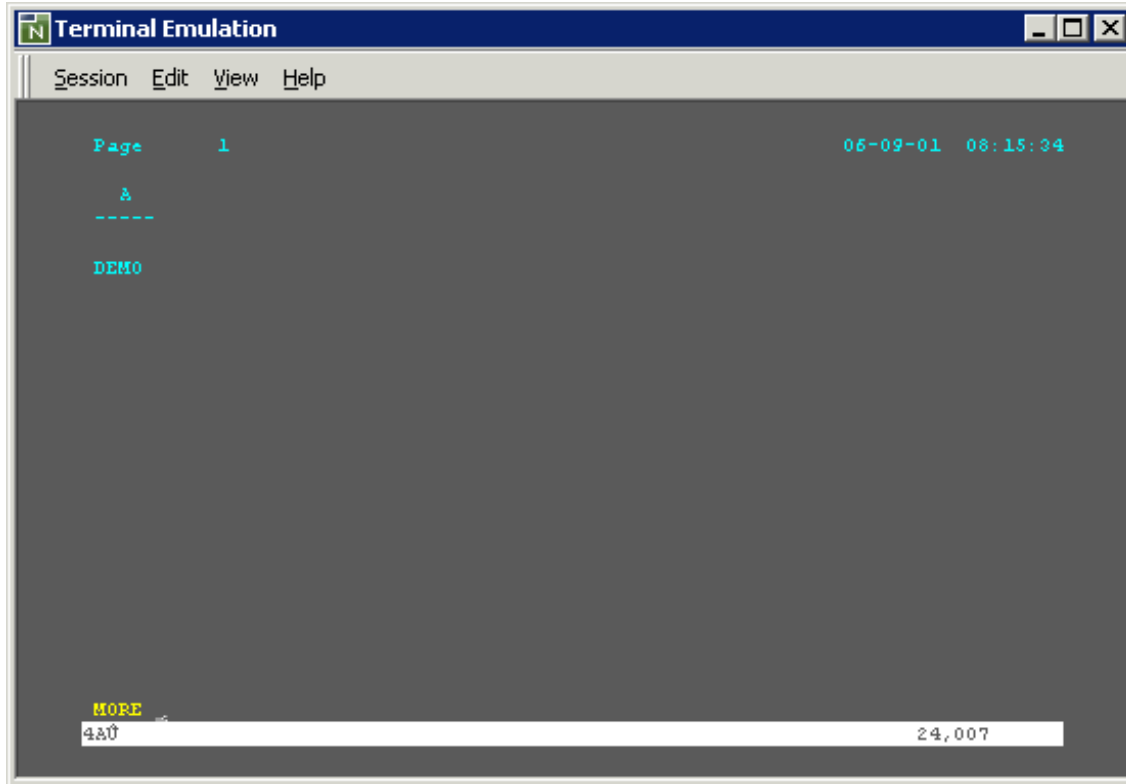
Choose the following toolbar button:



The INPUT statement from the above example program invokes either a terminal emulation window or a window for the Web I/O Interface (depending on the settings on the remote development server; see the description in the [previous](#) section). You are prompted for input.

- 3 Enter the string "Demo" at the **Enter** prompt and press ENTER.

The DISPLAY statement from the above example program now shows the string you have just entered. Example for the terminal emulation window:



- 4 Press ENTER.

The example program is ended and the window in which it was shown is automatically closed.

## Debugging a Program

---

You will now debug your DEMO program.

### ▶ To debug a program

- 1 In the library workspace, select the program DEMO.
- 2 From the **Debug** menu, choose **Start**.

Or:

Invoke the context menu and choose **Debug**.

Or:

Press CTRL+F7.

Or:

Choose the following toolbar button:



When the editor for the selected object has not yet been opened, it is opened now and the debugger is started. The trace position is shown at the first executable source code line (i.e. the line containing the INPUT statement).

```

DEMO [NATURAL - My Mainframe] - Program
DEFINE DATA LOCAL
1 A (A5)
END-DEFINE
INPUT 'Enter' A
DISPLAY A
END
  
```

In addition, the following debugger windows are shown:

- **Variables Window**

This window shows all variables which are available at current state of the program execution.

- **Break- and Watchpoints Window**

This window shows all currently defined breakpoints and watchpoints.

- **Call Stack Window**

This window shows the objects which have been called during the current debugging session in hierarchical order.

3 From the **Debug** menu, choose **Step Into**.

Or:

Press F11.

Or:

Choose the following toolbar button:



The next program step is executed and the trace position is shown at the corresponding source code line. If this source code line invokes or includes a further Natural object, the debugger steps into this object and the trace position is shown at the first executable line.

In case of your program, the `INPUT` statement invokes either a terminal emulation window or a window for the Web I/O Interface (depending on the settings on the remote development server; see the description in the [previous](#) section).

- 4 Enter the string "Demo" at the **Enter** prompt and press `ENTER`.

The window in which you have entered the string is closed. In the editor window, the trace position is now shown in the next executable source code line. In case of your program, this is the line containing the `DISPLAY` statement.

- 5 Choose the **Step Into** command, if required, repeatedly until the output of the `DISPLAY` statement is shown in a window.
- 6 To close the terminal emulation window, press `ENTER`.

The debugger is terminated automatically if the application ends without an error.

- 7 Close the program editor by choosing the standard close button at the top right of the program editor window.

You can now proceed with the next exercise: [Locking and Unlocking](#).



# 15 Locking and Unlocking

---

▪ Opening an Object .....	84
▪ Opening the Same Object from Another Session .....	85
▪ Unlocking Objects .....	86
▪ Displaying a List of All Locked Objects .....	87
▪ Moving Folders Containing Locked Objects .....	89

An object locking mechanism prevents concurrent updates when working with objects that are stored on a remote development server.

The exercises below demonstrate locking for an object that you are currently modifying in the program editor. You will map the same development server once more. When you then try to open the same object from the new session, a locking message is shown.

For more information, see *Object Locking* in the *Remote Development Using SPoD* documentation of Natural for Windows.

This chapter covers the following topics:

## Opening an Object

---

When you open an object, it is locked for all other users until you close it.

You will now display the Natural code of the DEMO program that you have previously created.

► **To open the DEMO program in the logical view**

- In the library workspace, select the program DEMO, invoke the context menu and choose **Open**.

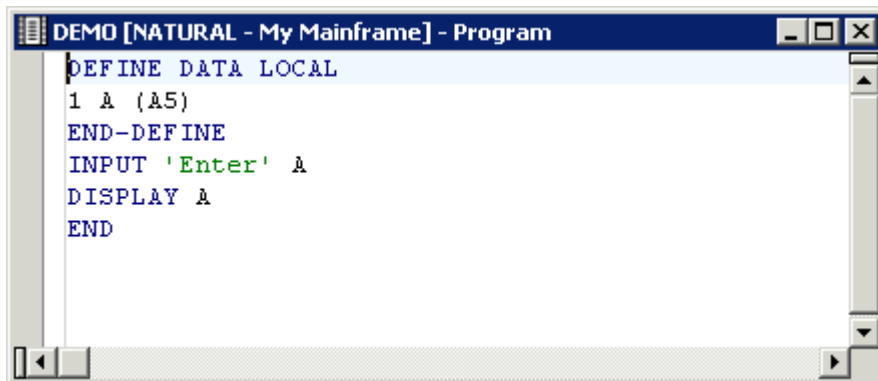
Or:

In the library workspace, select the program DEMO and press CTRL+O.

Or:

Double-click the name of the DEMO program.

The program is now shown in a program editor window.



Do not close this program editor window. Leave it open so that locking can be demonstrated with the next exercise.

## Opening the Same Object from Another Session

---

When you try to open an object that is currently being modified by another user, the corresponding lock message is shown. This message tells you which user is currently modifying the object and the date and time it was locked.

In order to demonstrate locking, you will first map the same development server once more. You will then try to open the DEMO program from the new session.

### ▶ To open the same object from another session

- 1 From the **Tools** menu, choose **Map > Environment**.

Or:

Choose the following toolbar button:



The **Map Environment** dialog box appears.


- 2 Enter all required information as described under *Connecting to a Development Server for the First Time*.
- 3 In the new session, expand the node of the library containing the DEMO program.
- 4 Double-click the name of the DEMO program.

A lock message is now shown in a dialog box. It shows the ID of the user by whom the object was locked, and the date and time when the object was locked.

- 5 Choose the **OK** button to close the dialog box.

## Unlocking Objects

You will now unlock your DEMO program from the session in which it is currently locked (not the session in which it was opened).

 **Note:** When Natural Security is active, it is possible that you cannot use certain commands. Thus, it may be possible that you are not allowed to unlock your own objects.

### ▶ To unlock a locked object

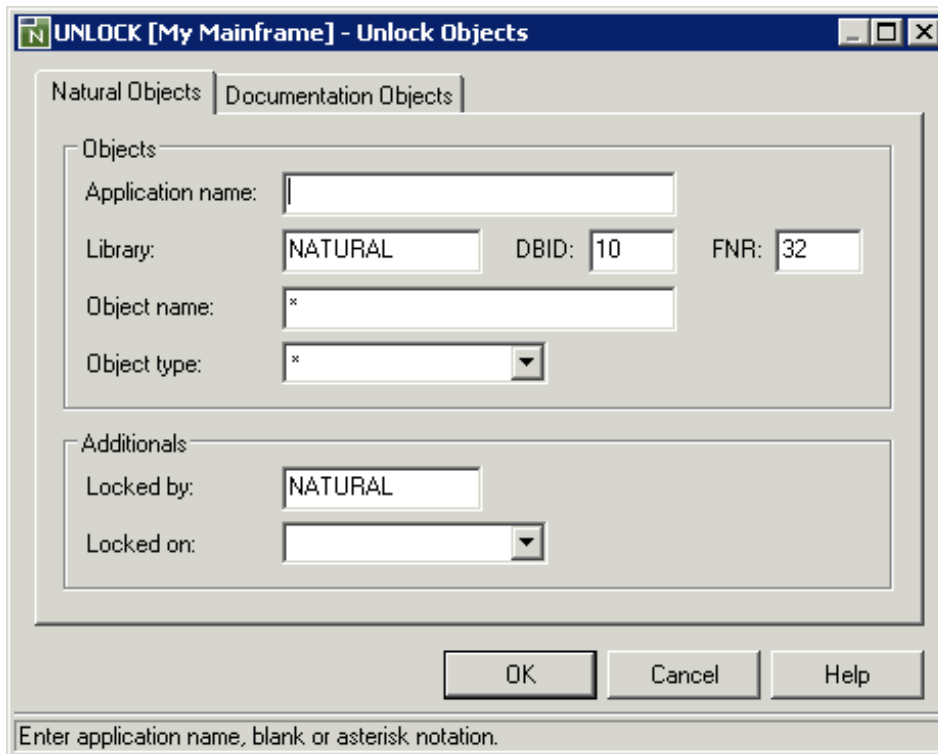
- 1 In library workspace, select the library containing the locked DEMO program.
- 2 From the **Tools** menu, choose **Development Tools > Unlock Objects**.

Or:

Choose the following toolbar button:

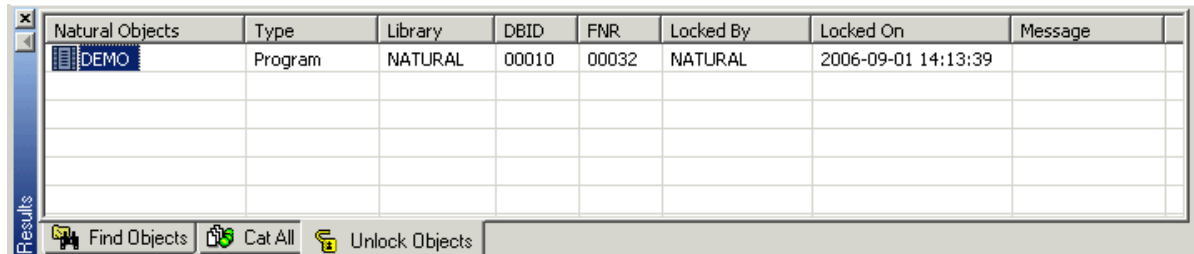


The **Unlock Objects** dialog box appears. The name of the library that is currently selected in library workspace is automatically provided.



- Choose the **OK** button (without entering any information).

When locked objects exist, they are shown on the **Unlock Objects** tab of the results window.



Natural Objects	Type	Library	DBID	FNR	Locked By	Locked On	Message
DEMO	Program	NATURAL	00010	00032	NATURAL	2006-09-01 14:13:39	

- In the results window, select the object to be unlocked.
- Invoke the context menu and choose **Unlock Objects**.

The object is still shown on the **Unlock Objects** tab. The **Message** column, however, indicates that the object has been unlocked.

- To hide the results window, choose the standard close button at the top of the results window.

Or:

From the **View** menu, choose **Results**.

Or:

Press ALT+5.

When the results window is not shown in the Natural Studio window, no check mark is shown in the **View** menu next to the **Results** command.



**Note:** You can **undock and dock** the results window as described previously in this tutorial.

## Displaying a List of All Locked Objects

You can display a list of all locked objects for the currently active development server. This includes the objects of all users in all libraries. You can unlock any object contained in this list.

It is only possible to display and unlock objects from another user in a non-secure environment (i.e. when Natural Security is not active on the development server). In a secure environment, the administrator defines which user locks may be unlocked by other users.

▶ **To display all locked objects**

- 1 From the **Tools** menu, choose **Development Tools > Unlock Objects**.

Or:

Choose the following toolbar button:



The **Unlock Objects** dialog box appears.

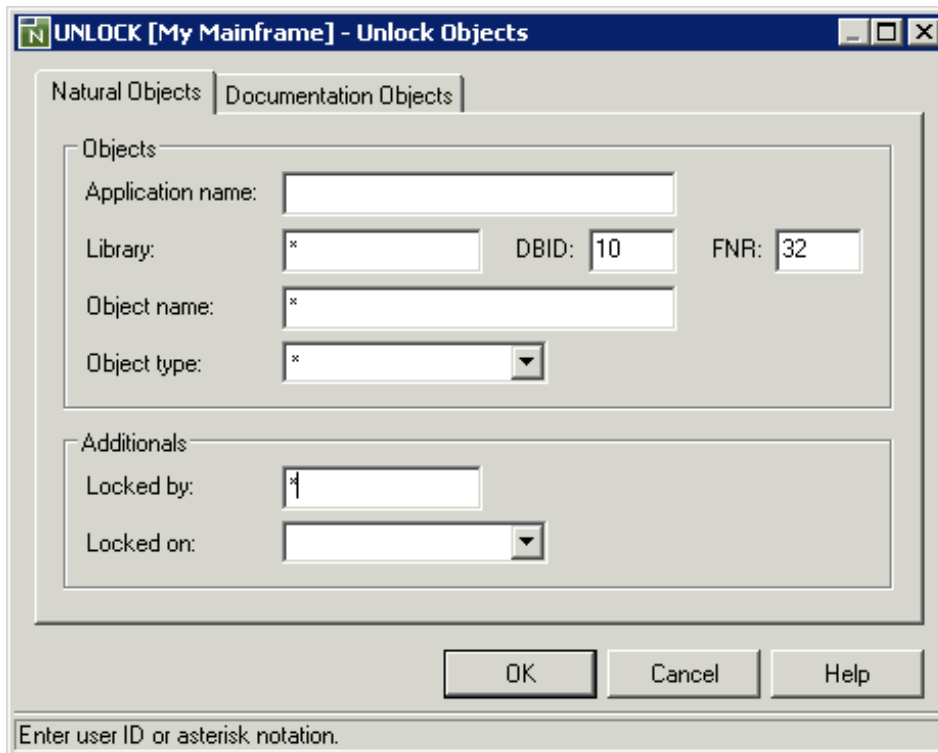
- 2 Enter an asterisk in the **Library** text box.

This will display the locked objects in all libraries.

- 3 Enter an asterisk in the **Locked by** text box.

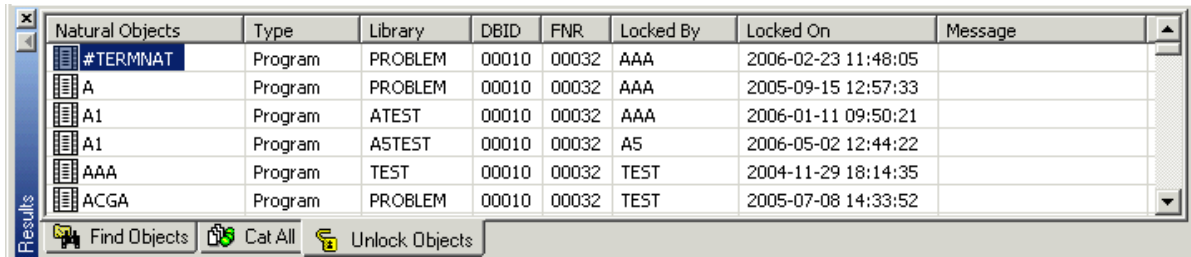
This will display the locked objects of all users.

The dialog box should now look as follows:



- 4 Choose the **OK** button.

The results window is shown again, displaying all locked objects.



Natural Objects	Type	Library	DBID	FNR	Locked By	Locked On	Message
#TERMNAT	Program	PROBLEM	00010	00032	AAA	2006-02-23 11:48:05	
A	Program	PROBLEM	00010	00032	AAA	2005-09-15 12:57:33	
A1	Program	ATEST	00010	00032	AAA	2006-01-11 09:50:21	
A1	Program	ATEST	00010	00032	A5	2006-05-02 12:44:22	
AAA	Program	TEST	00010	00032	TEST	2004-11-29 18:14:35	
ACGA	Program	PROBLEM	00010	00032	TEST	2005-07-08 14:33:52	

- 5 If you want to delete the **Unlock Objects** tab from the results window, select an object name in the first column, invoke the context menu and choose **Delete Tab**.
- 6 Before you continue with the next section, hide the results window as described above.

## Moving Folders Containing Locked Objects

Locked objects cannot be moved.

When you try to move a folder containing locked objects to another folder (for example, via drag-and-drop), a lock message appears.

When you choose the **OK** button to close the dialog box, another dialog box appears, asking whether you want to continue the move.

When you choose the **Yes** button, all objects except the locked objects are moved.

You can now proceed with the next exercise: *Handling Applications*.





# 16 Handling Applications

---

■ Prerequisites .....	92
■ Displaying the Application Workspace .....	92
■ Creating a Base Application .....	94
■ Creating a Compound Application .....	97
■ Linking Objects to a Base Application .....	98
■ Linking Base Applications to a Compound Application .....	100
■ Managing Linked Objects .....	101
■ Mapping an Application .....	102
■ Displaying the Properties of an Application .....	103

An application is a collection of Natural objects and non-Natural objects which build a functional unit from the business point of view.

There are two types of applications:

■ **Base Application**

A set of Natural objects that are stored in the same user system file (FUSER). You can link objects of different libraries to a base application.

■ **Compound Application**

You can link several base applications to a compound application. A compound application can be spread across different hosts.

For detailed information, see *SPoD Application Concept*.



**Note:** The applications that you will create with the exercises below do not run. Their purpose is just to demonstrate the basics for handling applications.

## Prerequisites

---

It is assumed for the following exercises that you have:

- copied the contents of the system library SYSSPODA to your user library SPODLIB,
- moved all subprograms from library SPODLIB to library SPODADD,
- copied the program PGMCHECK from library SPODLIB to library SPODTEST

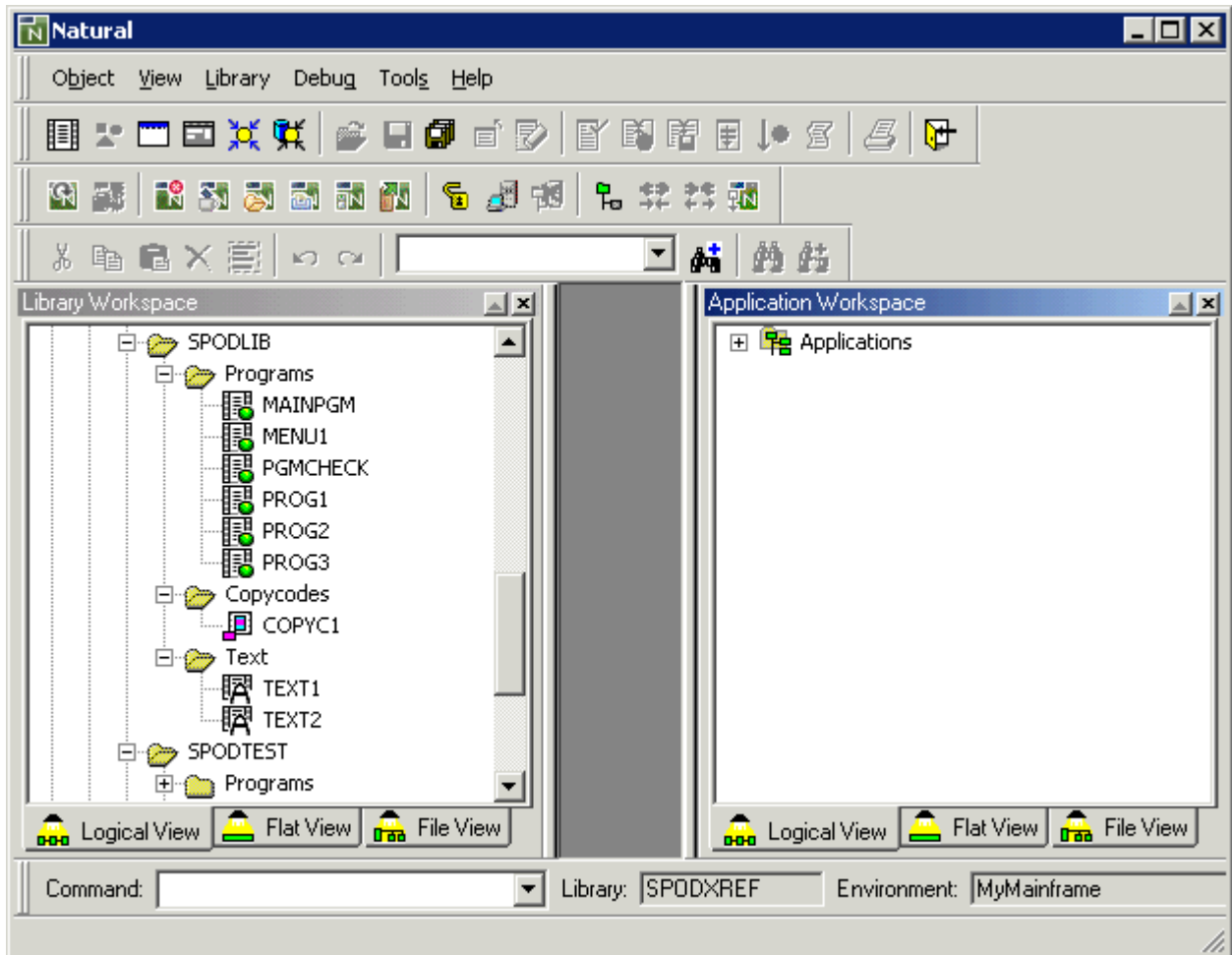
as described **previously** in this tutorial.

## Displaying the Application Workspace

---

The application workspace is the area in which all known applications can be displayed. It provides the same views as the library workspace (logical view, flat view and file view). Your application workspace is initially empty.

When you start Natural Studio for the first time, your application workspace is not shown. The exercise below explains how to display it. It is initially displayed at the right of the Natural Studio window.



▶ **To toggle application workspace display**

- From the **View** menu, choose **Application Workspace**.

Or:

Press ALT+2.

When the application workspace is displayed in the Natural Studio window, a check mark is shown next to the **Application Workspace** command.

## Creating a Base Application

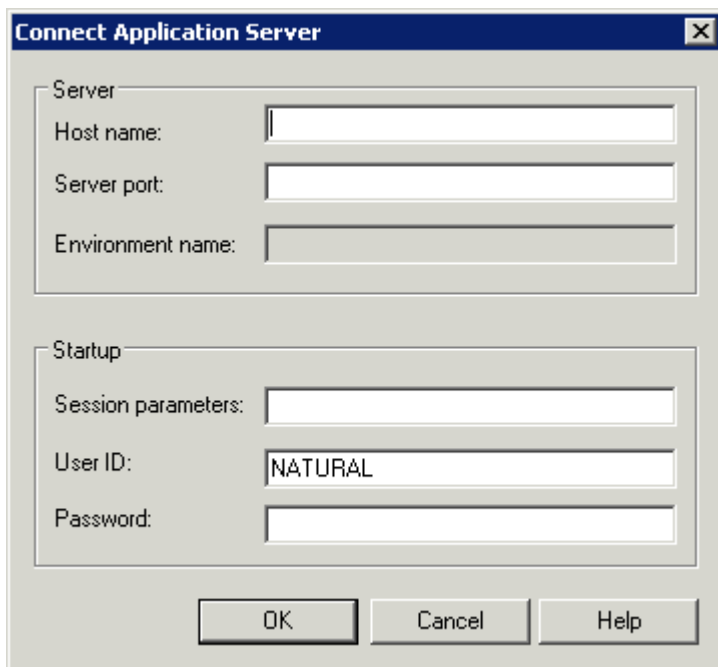
You will now create two base applications with the names `SPODAPPL1` and `SPODAPPL2`. You will create them on the same development server that you have previously mapped in the library workspace.

### ▶ To create a new base application

- 1 In the application workspace, select the "Applications" node.
- 2 Invoke the context menu and choose **New**.

When you work with applications for the first time, the **Connect Application Server** dialog box appears. You have to enter all required information for starting a session on the development server. This session, which is called "application server session", is used to access application data.

When you start Natural Studio the next time and expand the "Applications" node, the application server session is automatically started. The **Connect Application Server** dialog box will appear only if additional information (for example, a password) is required.



The screenshot shows a dialog box titled "Connect Application Server". It has a title bar with a close button (X). The dialog is divided into two sections: "Server" and "Startup".

The "Server" section contains three text boxes:

- Host name: [ ]
- Server port: [ ]
- Environment name: [ ]

The "Startup" section contains three text boxes:

- Session parameters: [ ]
- User ID: [ NATURAL ]
- Password: [ ]

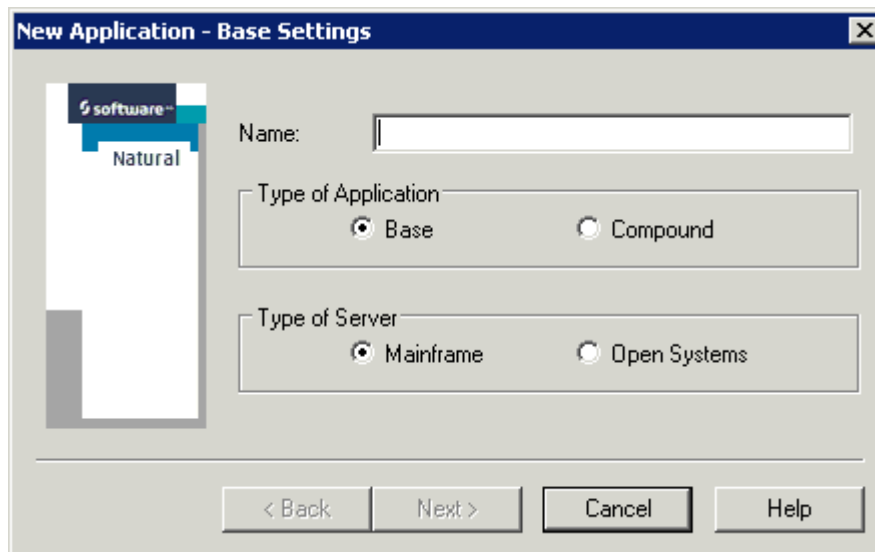
At the bottom of the dialog are three buttons: "OK", "Cancel", and "Help".

- 3 In the **Host name** text box, enter the name of the development server on which the application data are stored.
- 4 In the **Server port** text box, enter the TCP/IP port number of the development server.

- 5 If Natural Security is installed on the development server, specify the required password in the **Password** text box. Otherwise, leave this text box blank.
- 6 Choose the **OK** button.

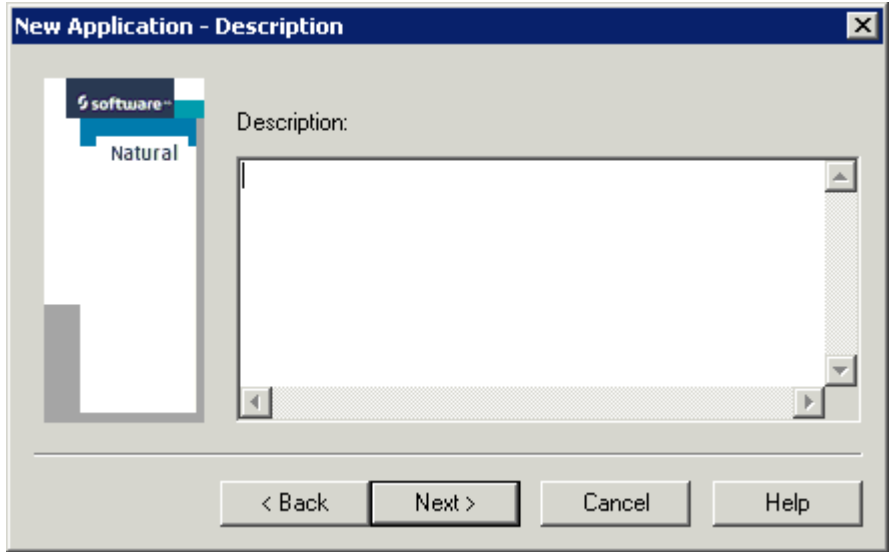
The application server is connected.

Since you have chosen the **New** command from the the context menu, the **New Application - Base Settings** dialog box appears.



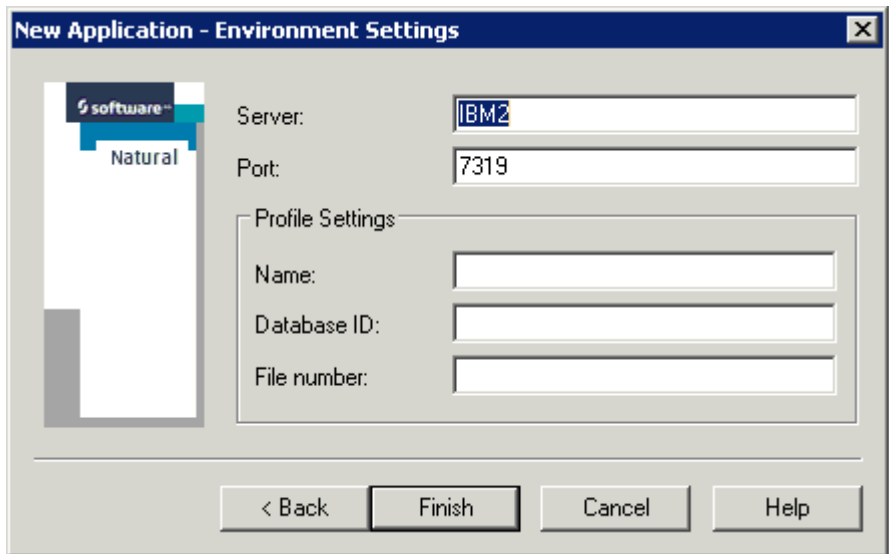
- 7 Enter the name SPODAPPL1 for your first base application.
- 8 Make sure that the **Base** option button is selected.
- 9 Select the option button for the type of server: since this tutorial assumes that you have mapped to a remote mainframe environment, choose the **Mainframe** option button.
- 10 Choose the **Next** button.

The **New Application - Description** dialog box appears.




- 11 Optional. Enter a description for your application. This can be any text.
- 12 Choose the **Next** button.


The **New Application - Environment Settings** dialog box appears.



The name and port number of the application server that you have previously mapped in the **Connect Application Server** dialog box are provided as the default values for the new application. For this tutorial, we will use the default values.


 **Note:** It is also possible to use another development server.

- 13 Optional. Specify the profile settings (name, database ID and file number) to control the session settings as on the mainframe.
- 14 Choose the **Finish** button.

 **Note:** If a password is required, the dialog box in which you have specified the connection information is shown again. Server name, port number and session parameters cannot be changed in this dialog box. They are fixed for an application. If this dialog box appears, specify the password and choose the **OK** button.

The new application is now mapped. It is shown in the application window. Each time you map an application, a new development server session is started for this application.

- 15 Repeat the above steps to create the second base application with the name `SPODAPPL2`. You will later link objects to these two base applications.

 **Note:** Since you have already defined the connection information, the **Connect Application Server** dialog box is not shown for your second application.

## Creating a Compound Application

---

You will now create a compound application with the name `SPODCOMP`.

### ▶ To create a new compound application

- 1 In the application workspace, select the "Applications" node.
- 2 Invoke the context menu and choose **New**.

The **New Application - Base Settings** dialog box appears.

- 3 Enter the name `SPODCOMP` for your compound application.
- 4 This time, select the option button **Compound**.
- 5 Choose the **Next** button.

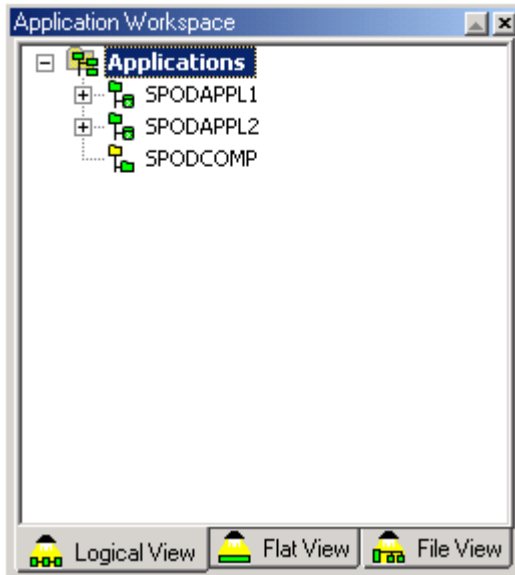
The **New Application - Description** dialog box appears.

- 6 Optionally. Enter a description for your application. This can be any text.
- 7 Choose the **Finish** button.

You will later link your two base applications to this compound application.

 **Note:** If a password is required, a dialog box appears in which you have to specify the password.

Your application workspace should now look as follows:



## Linking Objects to a Base Application

---

You can link any existing Natural object of the attached development server (for example, a program or map) to your application.

With the exercise below, you will link the following objects to your base applications:

	SPODAPPL1	SPODAPPL2
<b>SPODADD</b>	SUB1 SUB2	SUB2
<b>SPODLIB</b>	MAINPGM MENU1 PROG3 TEXT1	MENU1 PROG1 TEXT2
<b>SPODTEST</b>		PGMCHECK

### ▶ To link objects to a base application

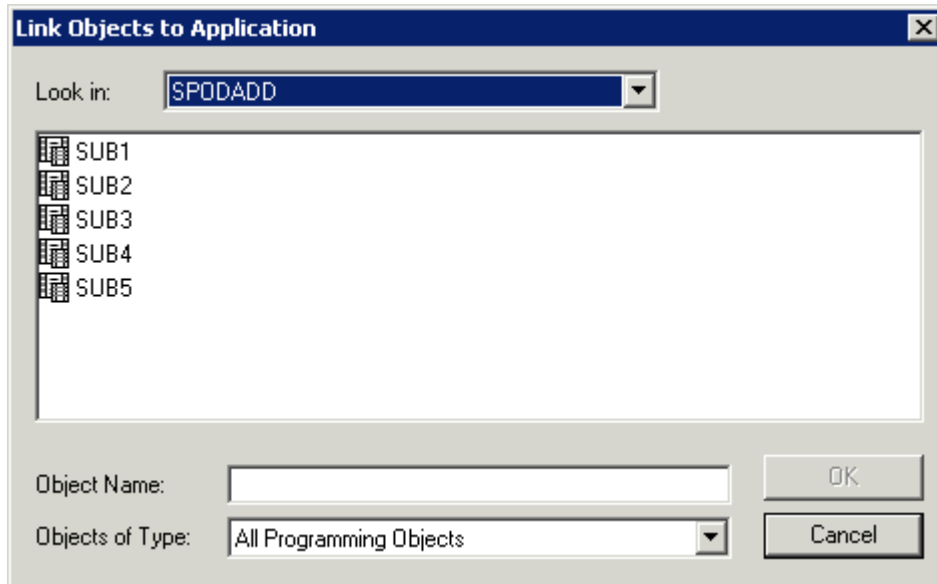
- 1 In the application workspace, select your base application SPODAPPL1.
- 2 Invoke the context menu and choose **Link**.

The **Link Objects to Application** dialog box appears.



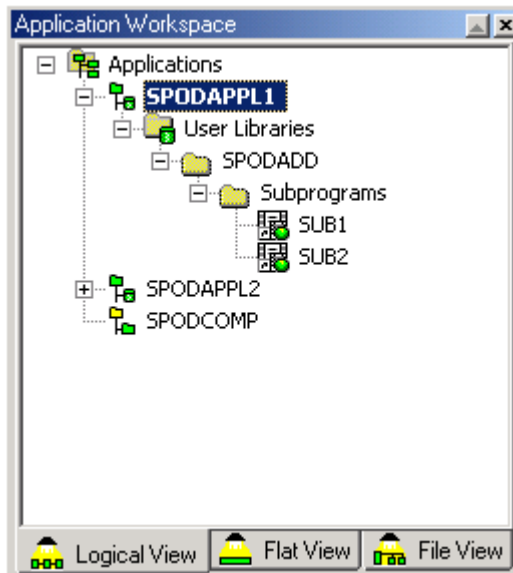
- From the **Look in** drop-down list box, select the library SPODADD.

The content of the selected library is now shown in the dialog box.



- Press and hold down CTRL and select the objects SUB1 and SUB2.
- Choose the **OK** button.

When you expand the nodes for the application SPODAPPL1 in the application workspace, the linked objects are shown.





**Note:** In the logical view, the selected objects are automatically placed in the corresponding folders (for example, a "Subprograms" folder is shown for all subprograms that you have selected).

6 Repeat the above steps to link the remaining objects as indicated in the above table.



**Note:** If you notice that you have linked a wrong object to your base application, you can unlink it. To do so, select the wrong object in the application workspace, invoke the context menu and choose **Unlink**.

## Linking Base Applications to a Compound Application

---

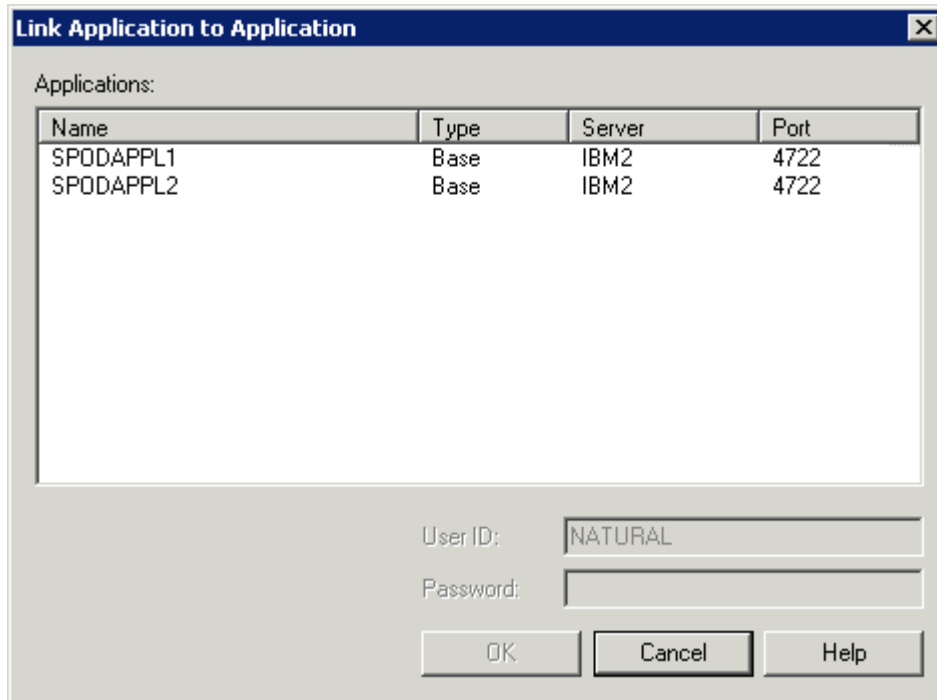
You can link any existing base application on the connected application server to a compound application. You can also link base applications which have different development server settings (that is, the session that is started for the application server may run on a different development server). A compound application thus allows you to combine objects that cannot be combined in the library workspace.


You will now link your two base applications SPODAPPL1 and SPODAPPL2 to your compound application SPODCOMP.

### ▶ To link base applications to a compound application

- 1 In the application workspace, select your compound application SPODCOMP.
- 2 Invoke the context menu and choose **Link**.

The **Link Application to Application** dialog box appears.




- 3 Select your first base application named SPODAPPL1.
  -  **Note:** If a password is required, specify it in the **Password** text box. If you do not specify a required password, an additional dialog box will later prompt you for this information.
- 4 Choose the **OK** button.
- 5 Repeat above steps to link second base application named SPODAPPL2.

## Managing Linked Objects

The objects in the application workspace are just references (or links) to the objects on the development server. They are not copies. For example, when you add a new program it will be visible in both the library workspace and the application workspace.

When you want to modify an object, you can do this either in the library workspace or in the application workspace. When an object is currently being modified by another user, the corresponding lock message is shown. A lock message is also shown when you try to open an object in the library workspace that you are currently modifying in the application workspace (and vice versa).

-  **Note:** Not all commands are available in the application workspace. For example, the commands **Delete** and **Rename** are only available in library workspace.

The following exception applies in the application workspace when cataloging (**CATALL**) the objects in a library: only the objects that have been linked with the application are cataloged (i.e. only the objects that are shown in the application workspace). The objects that are only shown in the library workspace are ignored. To find out which objects have been cataloged for the library that is currently selected in library workspace, issue the `LIST *` command from the command line. You can then check the catalog date in the resulting window.

## Mapping an Application

---

In addition to the applications you create yourself, you can also map applications from other users so that they are shown in the application workspace. You can map all applications that have already been defined on the development server that you have defined in the **Connect Application Server** dialog box.

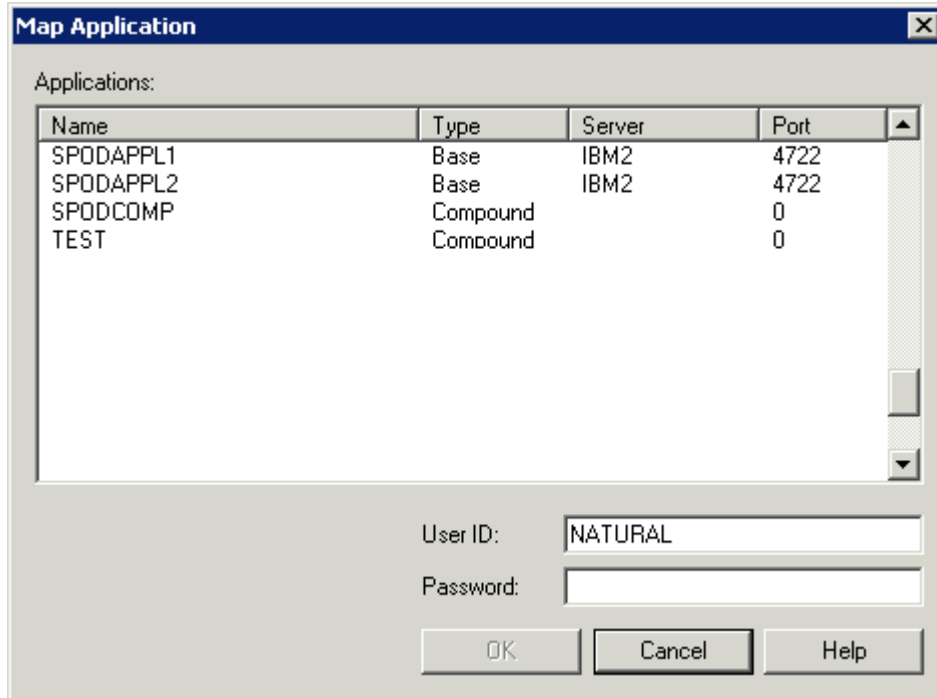
Each time you restart Natural Studio, your application workspace is empty. To display the previously mapped applications, expand the "Applications" node. As a result, all previously mapped applications are mapped again. If a password is required, further dialog boxes may appear in which you have to specify the missing information.

The following exercise explains how to map applications that you have not previously created.

### ▶ To map an application

- 1 In the application workspace, select the "Applications" node.
- 2 Invoke the context menu and choose **Map**.

The **Map Application** dialog box appears providing a list of all defined applications.



- 3 Select the application you want to map.



**Note:** If a password is required, specify it in the **Password** text box. If you do not specify a required password, an additional dialog box will later prompt you for this information.

- 4 Choose the **OK** button.

The mapped application is now shown in the application workspace.

## Displaying the Properties of an Application

If you need information about an application in the application workspace or if you want to change the settings of the application, you can display its properties.

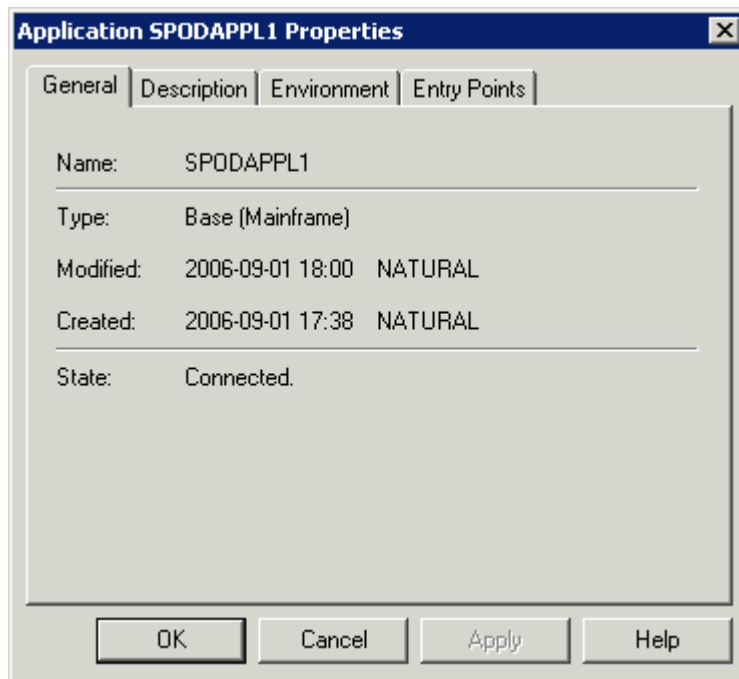
### ▶ To display the properties of an application

- 1 In the application workspace, select the application for which you want to display the properties.
- 2 Invoke the context menu and choose **Properties**.

Or:

Press ALT+ENTER.

A dialog box appears displaying the properties for the selected application. Example for a base application:



- 3 Select the tabs in this dialog box to view the corresponding properties.
- 4 Choose the **OK** button to close the dialog box.

You have now successfully completed this tutorial.

# 17

## SPoD Administration Policies and Procedures

---

This part provides information which is required to be able to administer a Natural development environment that is based on the Natural Single Point of Development (SPoD) approach.

It is primarily aimed at those who are responsible for setting up an effective working environment for Natural application developers.

- **Installing Natural Single Point of Development**
- **Configuring Natural Single Point of Development**
- **SPoD-Specific Extensions**





# 18

## Installing Natural Single Point of Development

---

- Prerequisites ..... 108
- Installation Procedure ..... 108

The following installation guide summarizes the prerequisites and the installation steps that are necessary to create a Natural Single Point of Development environment that uses Natural Studio as a remote development client and the Natural Development Server plug-in as a remote development server for mainframes running under the operating system z/OS (Batch or Complete), z/VSE, BS2000/OSD, Windows, UNIX or OpenVMS.

## Prerequisites

---

### Products Required or Involved

For detailed information on required or involved products, see [http://documentation.software-ag.com/natural/spod\\_prereq/prereq.htm](http://documentation.software-ag.com/natural/spod_prereq/prereq.htm).

### Technical Requirements

If you are working in a wide-area network (WAN) environment, please note that the minimum transmission rate must be 10 Mbit or higher.



**Important:** The speed of a modem or an ISDN connection will be insufficient.

## Installation Procedure

---

Installing the products that make up Natural Single Point of Development comprises the following general installation steps (as applicable):

- [Installing Natural Studio](#)
- [Installing Natural for Mainframes](#)
- [Installing Natural for Windows](#)
- [Installing Natural for UNIX](#)
- [Installing Natural for OpenVMS](#)

- [Installing the Natural Development Server](#)

## Installing Natural Studio

### ■ General Installation

Refer to the *Installation* documentation for Natural for Windows.

### ■ Installing the SPoD-Specific Features of Natural Studio

If you wish to have the SPoD-specific parts of Natural Studio enabled, select the setup type referring to server development (SPoD) in the course of the Natural for Windows installation procedure.

## Installing Natural for Mainframes

Refer to the relevant sections in the *Installation* documentation for Natural for Mainframes:

- *Installing Natural under z/OS*
- *Installing Natural under z/VSE*
- *Installing Natural under BS2000/OSD*

For background information on the automated installation of Software AG products, see also *Concepts of System Maintenance Aid* in the *System Maintenance Aid* documentation.

## Installing Natural for Windows

Refer to the *Installation* documentation for Natural for Windows.

## Installing Natural for UNIX

Refer to the *Installation* documentation for Natural for UNIX.

## Installing Natural for OpenVMS

Refer to the *Installation* documentation for Natural for OpenVMS.

## Installing the Natural Development Server

Refer to the relevant section of the Natural Development Server installation instructions:

- *Installing the Natural Development Server under z/OS and, if CICS is used as TP monitor, Installing the Natural Development Server CICS Adapter*
- *Installing the Natural Development Server under SMARTS on z/VSE and, if CICS is used as TP monitor, Installing the Natural Development Server CICS Adapter*
- *Installing the Natural Development Server under SMARTS on BS2000/OSD*

- *Installing the Natural Development Server under Windows*
- *Installing the Natural Development Server under UNIX*
- *Installing the Natural Development Server under OpenVMS*

### Defining a Natural Development Server File

The Natural Development Server File (FDIC) is used as a central dictionary file for storing Natural applications and the links to objects making up an application. It also holds object locking information. This information is not bound to certain groups of application developers, but has an impact on the entire application development of an enterprise. Therefore, this file should be available only once, to ensure that the application definitions and locking states are kept consistent.

The following information applies to mainframe environments only:

#### ▶ **To ensure the use of one single Natural development file**

- 1 Set the NTDYNP parameter in the default Natural parameter file NATPARM to OFF:
  - Specify NTDYNP OFF, if you want to disallow overwriting of all dynamic parameters
  - or specify NTDYNP OFF, FDIC, if you want to disallow overwriting of the Natural development server file only.
- 2 Disallow any changes to profiles by the end user.

# 19

## Configuring Natural Single Point of Development

---

- Configuring Natural for Windows ..... 112
- Configuring Natural for Mainframes ..... 112
- Configuring Natural for UNIX ..... 113
- Configuring Natural for OpenVMS ..... 113
- Configuring the Natural Development Server ..... 113

This chapter provides an overview of the configuration facilities that exist for the individual products involved in a Natural Single Point of Development environment. References are given to documents where the relevant configuration functions or parameters are described.

See also *Configuring the Natural Development Server* in the corresponding Natural Development Server documentation.

## Configuring Natural for Windows

---

Natural for Windows offers configuration possibilities which are documented in the *Configuration Utility* documentation of Natural for Windows. For details, refer to:

- Local Configuration File
  - *Buffer Pool Assignments*
  - *Installation Assignments*
- Global Configuration File
  - *Database Management System Assignments*
  - *Dictionary Server Assignments*
  - *Printer Profiles*
  - *System Files*

See also the *Parameter Reference* documentation.

## Configuring Natural for Mainframes

---

Information on how to configure Natural in a mainframe environment is primarily contained in the *Operations* and *Parameter Reference* documentation for Natural for Mainframes. The following topics are relevant:

- *Profile Parameter Usage*
- *Natural Parameter Hierarchy*
- *Assignment of Parameter Values*
- *Profile Parameters Grouped by Function*

## Configuring Natural for UNIX

---

Information on how to configure Natural in a UNIX environment is primarily contained in the *Operations* and *Parameter Reference* documentation for Natural for UNIX. The following topics are relevant:

- *Profile Parameters*
- *Natural Buffer Pool*

## Configuring Natural for OpenVMS

---

Information on how to configure Natural in an OpenVMS environment is primarily contained in the *Operations* and *Parameter Reference* documentation for Natural for OpenVMS. The following topics are relevant:

- *Profile Parameters*
- *Natural Buffer Pool*

## Configuring the Natural Development Server

---

For details on how to configure the Natural Development Server, refer to the corresponding platform-specific Natural Development Server documentation.





# 20

## SPoD-Specific Extensions

---

- User Exits for Natural Development Server on Mainframes ..... 116
- APIs for the SPoD Utility Protocol ..... 117

In a SPoD environment, several extensions to existing product functionality are available.

## User Exits for Natural Development Server on Mainframes

---

Natural Single Point of Development provides the following user exits for mainframes:

### NDV-UX01

This exit is invoked before a Natural object or a DDM is edited.

It can be used to reject the editing of certain sources.

The source code of this exit is delivered in the library `SYSLIB` and is named `NDV-SX01`. See note below.

### NDV-UX02

This exit is invoked before a Natural object, a DDM or a user error message is deleted, copied or moved. This also applies when the context menu functions Cut, Copy and Paste are used.

It can be used to reject the further processing of this object, similar to the user exit `MAINEX01` of the `SYSMAIN` utility in Natural for Mainframes.

The source code of this exit is delivered in the library `SYSLIB` and is named `NDV-SX02`. See note below.



**Note:** The sources of these user exit routines are named `NDV-SXnn`, where *nn* denotes the number of the user exit routine.

### Making a User Exit Routine Available

To make a user exit routine available

- Copy the source code from `SYSLIB` into a user library.
- Catalog it under the name `NDV-UXnn`.
- Copy it back into the Natural system library `SYSLIB`.

The name of each user exit source is different from the name of the corresponding cataloged object. This is to guarantee that the object will not be affected if the user exit source is overwritten by an installation update.

For further details, see the source code of the user exit routines `NDV-SXnn` in the Natural system library `SYSLIB`.

## APIs for the SPoD Utility Protocol

---

With the SPoD utility protocol it is possible to invoke a program on the Natural Development Server, send alphanumeric data to the server (up to a maximum of 5000 bytes) and receive alphanumeric data from the server (up to a maximum of 5000 bytes). This works similar to a Natural RPC call. The server program(s) can communicate with the client program(s), i.e. data can be exchanged with the client when the server program is still running.



**Important:** The programs running on the Natural Development Server must not perform any screen I/O operations. This would result in a break down of the server connection.

Two APIs and two example programs are provided in library `SYSEXNDC` of Natural for Windows installed on the client side.

The corresponding APIs and example programs on the server side are provided in the library `SYSEXNDV` of the Natural Development Server.

The APIs can be used together with the SPoD utility protocol.

- [APIs for Natural on the Client](#)
- [APIs for Natural on the Server](#)
- [Example Application](#)

### APIs for Natural on the Client

For the client programs the following APIs and examples are provided in library `SYSEXNDC` of Natural for Windows:

#### NDVC001N

This subprogram returns the status of a connection to the remote Natural Development Server environment.

It can be used to check whether it is possible to connect to a server application.

#### NDVC002N

This subprogram provides access to a connection to the remote Natural Development Server environment.

It invokes a program on the server. The library and program name are part of the data sent to the server.

#### Example NDVC001P

This example explains how to design a user-defined program to call subprogram `NDVC001N`.

It also contains a description of the parameters of subprogram NDVC001N.

### **Example NDVC002P**

This example explains how to design a user-defined program to call subprogram NDVC002N.

It also contains a description of the parameters of subprogram NDVC002N.

### **APIs for Natural on the Server**

On the server, the following APIs and examples are provided in the library SYSEXNDV of the Natural Development Server.

The description of the parameters can be found in the text member A-README in the same library.

#### **NDVS001N**

This subprogram, which must be called when a server program is started, is used for the initialization of the server environment.

It sets the steplib `SYSLIB` and `SYSLIBS` in the user steplib table and checks if the trace function is active.

#### **NDVS002N**

This subprogram should be called at the end of a server program before the `END` or `STOP` instruction for cleanup reasons.

It deletes the steplib `SYSLIB` and `SYSLIBS` from the user steplib table.

#### **NDVS003N**

Natural subprogram that enables you to exchange data with the client program.

#### **NDVS004N**

Natural subprogram that enables you to write trace data into a Natural text member or into the Natural Development Server log file. This may be helpful during program development and for error tracing.



**Note:** On the mainframe, every time a trace message has been written into a Natural text member, an `END TRANSACTION` is performed.

## Example Application

A small example application that illustrates how to use the different APIs is provided with the Natural Development Server.

The client program is provided in the library SYSEXNDC of Natural for Windows.

The server programs are provided in the library SYSEXNDV of the Natural Development Server.

### Client Program in Library SYSEXNDC

For the client side, the Program NDVCMAIN is provided.

It checks whether the remote environment is active and enables you to invoke five different programs on the server side. To activate the program, create a customized user command as described in the section [How to Execute a Program Locally in Natural Studio When a Remote Environment is Active](#).

### Server Programs in Library SYSEXNDVClient

For the server side, the following example programs are provided:

#### ■ NDVSPING

This is an example program for the PING function. It reads the data sent by the client program and simply returns a filled buffer.

#### ■ NDVSINTA

This is an example program for the interaction handling. It

- reads the data sent by the client program;
- returns data to the client;
- waits for an answer from the client;
- depending on the data sent by the client it continues sending data or stops processing.

#### ■ NDVSEMPR

This is an example program that reads the EMPLOYEES file and sends the data to the client. It is an extension of the interaction handling demonstrated in NDVSINTA.

#### ■ NDVSTRCE

This is an example program that enables you to switch the trace facility on or off. The above mentioned example programs all write trace data if the trace facility is activated.

#### ■ NDVSWRLG

This is an example program that writes a message into the Natural Development Server log file.

### **How to Execute a Program Locally in Natural Studio When a Remote Environment is Active**

In order to communicate with a program on the Natural Development Server, a Natural program or dialog must be executed locally in Natural Studio when you are connected to a remote Natural Development Server environment (that is, when your current active library is a library on the Natural Development Server). This can be easily achieved by defining a customized user command in Natural Studio.

In order to execute `NDVCMAN`, your customized user command must contain the following:

```
LOGON SYSEXNDC; NDVCMAN
```

A detailed description on how to create customized user commands is given in the *Natural for Windows* documentation under *User Commands* in the section *Customizing Natural Studio* of the *Using Natural Studio* documentation.