Natural ISPF

Introduction to Natural ISPF

Version 9.2.4

September 2025

ADABAS & NATURAL

This document applies to Natural ISPF Version 9.2.4 and all subsequent releases.

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

Document ID: ISP-INTRO-924-20250929

Table of Contents

Preface

This documentation gives you a short overview of the Natural ISPF features.

What is Natural ISPF?	Describes the purpose and the concept of Natural ISPF, the Integrated Structured Programming Facility.
Scope of Functionality	Illustrates the interaction of Natural ISPF with our other products and the rest of the world.
Natural ISPF Objects	Describes all the object types accessible with Natural ISPF and gives examples of some typical functions that can be executed on them.
User Applications	Describes the incore database, the macro facility and the integration of user applications into Open NSPF.
General Features	Describes special features available in Natural ISPF, such as split screen, multiple sessions, Software AG Editor, versioning and more.

1 About this Documentation

Document Conventions

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

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at **https://documentation.softwareag.com**.

Product Training

You can find helpful product training material on our Learning Portal at **https://learn.softwareag.com**.

Tech Community

You can collaborate with Software GmbH experts on our Tech Community website at **https://techcommunity.softwareag.com**. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software GmbH news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at **https://github.com/softwareag** and **https://hub.docker.com/publishers/softwareag** and discover additional Software GmbH resources.

Product Support

Support for Software GmbH products is provided to licensed customers via our Empower Portal at **https://empower.softwareag.com**. Many services on this portal require that you have an account. If you do not yet have one, you can request it at **https://empower.softwareag.com/register**. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

Data Protection

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

2 What Is Natural ISPF?

Natural ISPF (Integrated Structured Programming Facility) is an application development tool for the building, testing and maintenance of applications throughout their life cycle.

So what? you may ask. In answer, consider the following issues:

- Does the building and testing of applications at your site take place on different operating systems and/or in different TP/DP environments using different interfaces and tools?

  Natural ISPF is independent of your site's operating system(s) and TP/DP environment(s). Users of Natural ISPF can stay in the same working environment, irrespective of the underlying operating environment and TP monitor currently used. CICS, IMS, TSO, openUTM, TIAM, Batch are just some of the more common systems supported by Natural ISPF.
- Would it help you if you had a single, familiar system image of all objects and resources involved in application development?

  Using Natural ISPF, Natural and non-Natural objects can be accessed and processed with a single user interface. You can display and edit text files, JCL, and code held as Natural objects or 3GL programs (Assembler, COBOL, etc.). You can submit and monitor jobs, control job listings, and perform data set maintenance tasks. Operations on all of these different objects types is afforded using standard menus and a single set of commands with consistent syntax.
- How many different editors do your application programmers use in their development work?

  Natural ISPF uses only one editor to list, display and edit all objects (files, members, JCL, job output, Natural programs, etc.). Object lists and the edit environment are presented in a way that is comfortable and largely self-explanatory, and looks especially familiar to programmers who are used to working in time-sharing environments (for example, TSO/ISPF on z/OS).
- Are you having or anticipating resource bottlenecks in your TSO/ISPF environment?

  Possible problems of overhead and resource shortages in growing TSO/ISPF environments is just one example of a burning data processing issue that Natural ISPF addresses. The introduction

of Natural ISPF to your site allows easy migration of data processing methods to other environments, while preserving, even enhancing ISPF capabilities and functionality.

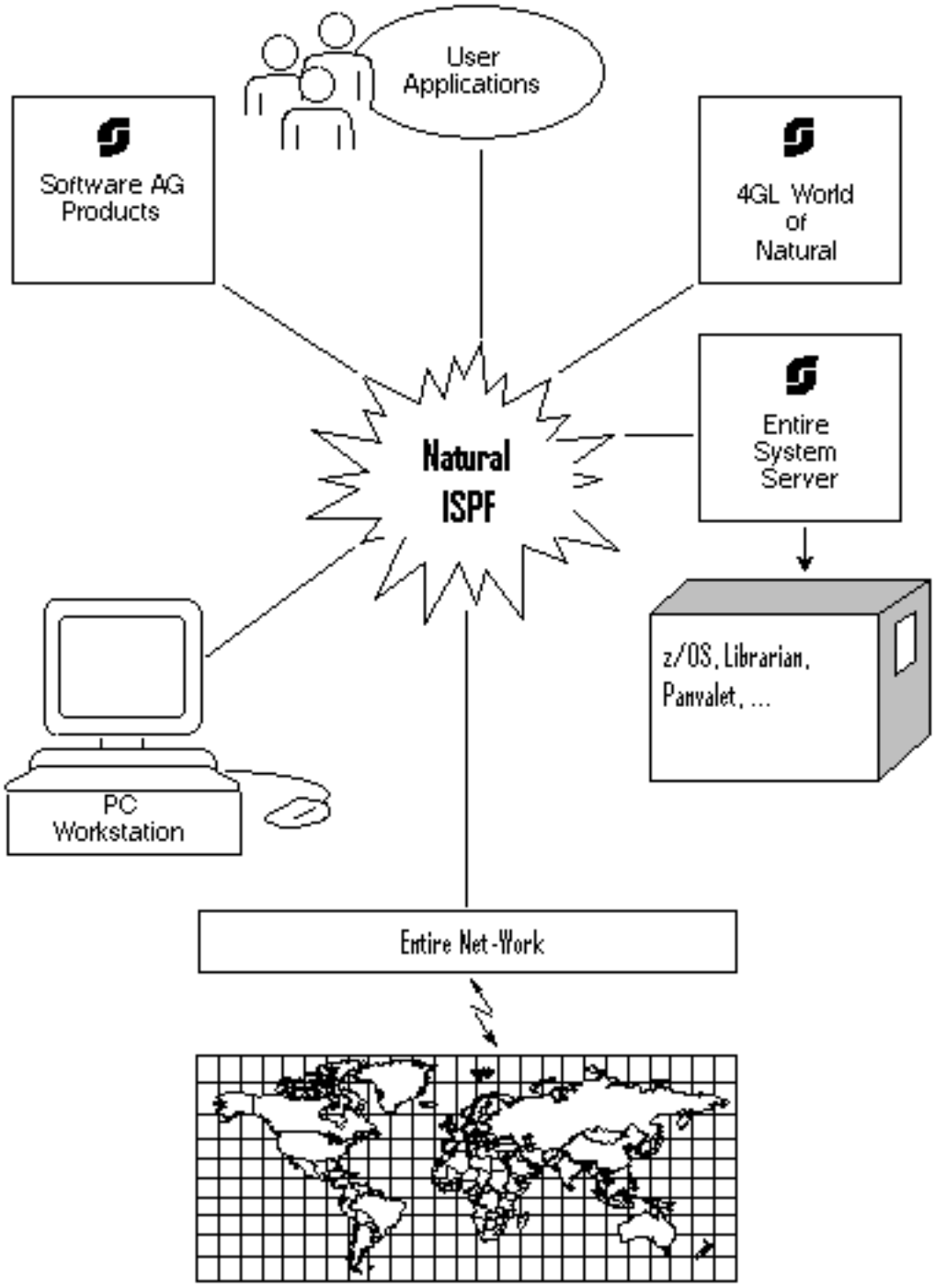
The above are just some of the issues Natural ISPF solves. In addition, Natural ISPF does not clone TSO/ISPF but offers a wide range of additional capabilities and features (outlined in this document) to be a perfect complement for application development with Natural. It includes a server part for the Mainframe Navigator which passes a subset of Natural ISPF functionality to SpoD environments to administer Non-Natural operating system objects. The benefits to be reaped from introducing Natural ISPF to your data processing setup are many. Among the most obvious are:

- **Reduced training costs:**
  The independence from operating systems and TP environments provided by Natural ISPF means reduced investment in training otherwise required for the different environments. Only one working environment is used and one syntax needs to be learned.
- **Increased productivity:**
  Use of a single working environment, together with the flexible, advanced facilities offered by Natural ISPF such as multiple parallel sessions, cross-session operations, automatic generation of skeleton programs, customization of the users' environment to their individual needs and ease of use substantially shortens the turnaround time in application development from specification to completion.
- **Resource savings:**
  With Natural ISPF offering capabilities familiar to TSO/ISPF users in other environments, the migration of data processing methods becomes a smooth operation that involves little or no retraining, yet results in substantial savings on computer resources.
- **Uniform interface for all users:**
  The scope of Natural ISPF functionality, the wide range of accessible objects, and the high level of integration potential with other products make Natural ISPF a powerful tool not only for application developers, but also for system programmers, computer operators and general users. Having a single, uniform view of your data processing resources makes work for these individual groups more convenient and makes communication between them easier and more effective, thus streamlining your whole operation.

3 Scope of Functionality

The Gateway to Your Data Processing World

Natural ISPF is the gateway to your data processing world. This is illustrated by the following graphical representation of the scope of its functionality:


User Applications
Software AG Products
4GL World of Natural
Natural ISPF
Entire System Server
z/OS, Librarian, Panvalet, ...
PC Workstation
Entire Net-Work

User Applications

Natural ISPF offers a high level of integration potential. Existing applications can be integrated under the Natural ISPF user interface. To make the picture complete, Natural ISPF can be customized to the needs of individual users.

Natural

As the perfect companion to Natural, Natural ISPF can be installed in any mainframe Natural environment with no other prerequisites. All Natural commands and functions can be accessed from the Natural ISPF environment with the added benefit of all the advanced features Natural ISPF offers.

Entire System Server

For sites that have Entire System Server (ESY) installed, Natural ISPF provides a comfortable, easy-to-use interface to access a wide range of objects irrespective of their operating environment. Whether it be PDS members, or files held in data management systems such as CA Panvalet or CA Librarian, uniform menus and a single set of commands are available to access each environment.

Entire Net-Work

Natural ISPF is ideally suited to serve a multi-processor environment. From a single location, users can access data residing on different, even diverse computers linked by the transport service Entire Net-Work. Cross-computer processing becomes a reality as, for example, an application implemented on a supported platform in Munich can access data stored on a machine running under z/OS in Chicago and easily transfer it to a system on a different supported platform in Sydney.

PC/Workstation

The PC/Workstation environment can be integrated in Natural ISPF using Entire Connection. Any Natural ISPF object can be downloaded as a PC file, and any PC file can be uploaded and stored as a Natural ISPF object (Natural program, PDS member, etc.).

Other Products

Our other products can be integrated in the Natural ISPF environment. Con-nect documents, Predict long descriptions and cross-reference sets can be handled using the comfortable Natural ISPF interface.

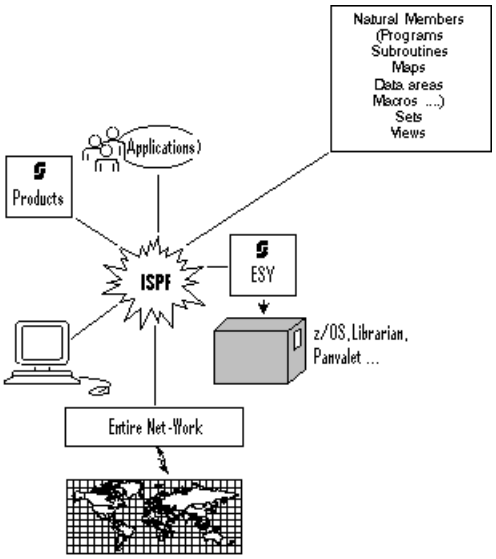
Natural ISPF

Natural ISPF supports work with the wide range of objects mentioned in the above paragraphs with an easy-to-use environment and a number of sophisticated features that include multiple parallel sessions, split-screen, cross-session operations, versioning and an advanced editor.

4 Natural ISPF Objects

This chapter describes all the object types accessible with Natural ISPF and gives examples of some typical functions that can be executed on them.

4 GL World of Natural



- **Natural Members:**
  Natural Members: All types of Natural members (programs, subroutines, maps, data areas, help routines, macros etc.) can be listed, displayed and edited using Natural ISPF. Where the Software AG Editor delivered with Natural ISPF is used, all advanced Editor features are available, including the ability to issue Natural-specific commands such as `STOW`, `CAT`, `RUN`, `CHECK`, `STRUCT`.

  Among other features that can substantially enhance your work with Natural are the Macro Facility, Versioning, and the User Workpool. These are all described in more detail later in this document.

  The following is an example of using the User Workpool while editing a Natural program. You can direct the output of a Natural program to the User Workpool. Running the Natural edit session and the workpool session simultaneously on your screen in split-screen mode, you can edit the program, and immediately see the result of your changes in the output:

```
EDIT-NAT:SYSISPE(EXW1)-Program->Report-Free-43K ----------- >>> Source EXW1 run
 COMMAND===>                                                  SCROLL===> CSR
****** ***************************** top of data ******************************
000010 * DEMO: WORKPOOL
000020 *
000030 DEFINE PRINTER(1) OUTPUT 'WORKPOOL'
000040 *
000050 READ (20) AUTOMOBILES BY MAKE STARTING FROM 'F'
000060    DISPLAY(1) MAKE COLOR MODEL HORSEPOWER
000070             NUMBER-OF-CYLINDERS
000080 END
BROWSE-OUT:EXW1/TYPE=REPORT-1 ---------------------------------- Columns 001 076
 COMMAND===>                                                  SCROLL===> CSR
** ******************************** top of list ********************************
   Page      1                                                  93-12-17  10:05

        MAKE        COLOR           MODEL          HORSE NR
                                                   POWER CYL
   ------------- ---------- ------------------- ----- ---

   FERRARI        BLUE       365 GTB/4             350    6
   FERRARI        WHITE      365 GTB/4             350    6
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right :s
```

Natural programming thus becomes much faster, more convenient, and more efficient, increasing productivity.

- **Sets:**
  If you have used Predict cross referencing to create sets for a Natural library, Natural ISPF allows you to list sets and work on the members in a specified set. The Predict cross-reference menu can be called directly from Natural ISPF: there is no need to leave your working environment to define sets.
- **Views:**
  Natural views defined in your environment can be listed, view definitions can be displayed, and field contents can be queried from the data base. Using Natural ISPF's multi-session and split-screens capabilities, coupled with the comfortable active help windows, this facility allows fast access to data without extra programming, even without interrupting the edit session.

  The following example shows two Natural ISPF sessions in split-screen mode: the session in the top half of the screen contains a Natural edit session. The session in the bottom half of the screen (partly covered by the pop-up windows) shows a list of views. One view has been selected for a query of the database (`B` line command). In the resulting pop-up window, criteria such as start and end values, as well as total records for the database query can be specified. The larger window is then displayed, offering the fields of the view which can be selected for inclusion in the query:

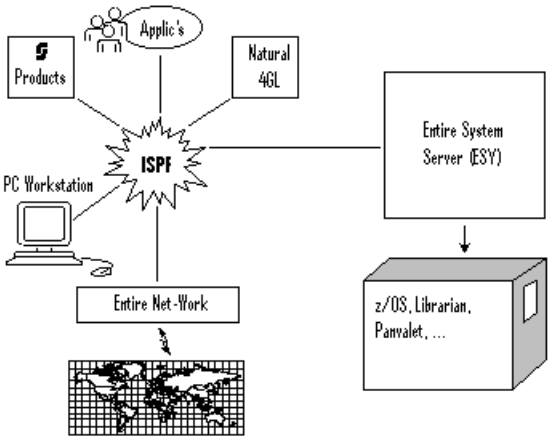
```
EDIT-NAT:SYSISPE(EXW1)-Program->Report-Free-42K --------------- Columns 001 072
 COMMAND===>                                                    SCROLL===> CSR
****** ***************************** top of data ******************************
000010 * DEMO: WORKPOOL
000020 *
000030 DEFINE PRINTER(1) OUTPUT 'WORKPOOL'
000040 *
000050 READ (20) AUTOMOBILES BY MAKE STARTING FROM 'F'
000060    D +-----------------------------------------------+
000070      ! Select fields:                                !
000080 END  !                                               !
LIST-VIW:A* ! _ 1   CAR-DESCRIPTION                         ! - Columns 034 050
 COMM +---- ! X 2 D MAKE                              A 14.0 !-----------------+
   VI ! BRO ! X 2 D MODEL                             A 20.0 !                 !
 b AU ! Vie ! _ 2 D BODY-TYPE                         A 15.0 !                 !
   AU ! Rec !  2 D NUMBER-OF-CYLINDERS               N 2.0  !                 !
** ** ! Sta ! X 2 D HORSEPOWER                        N 3.0  !                 !
      ! End ! _ 2   PISTON-DISPLACEMENT               N 5.0  !                 !
      ! Max ! _ 1   CAR-DETAIL                               !                 !
      ! Pas ! _ 2   WEIGHT                            N 5.0  !                 !
      +---- ! X 2 D COLOR                              A 10.0 !-----------------+
            ! Entr-PF3--PF7--PF8--                           !
Enter-PF1-- ! Down End  Top  Down                           !F10--PF11--PF12---
      Help  +-----------------------------------------------+eft  Right :s
```

When the fields have been selected as shown in the above example screen, the result of the query in the bottom half of the screen looks as follows:

```
EDIT-NAT:SYSISPE(EXW1)-Program->Report-Free-42K --------------- Columns 001 072
 COMMAND===>                                                    SCROLL===> CSR
****** ***************************** top of data ******************************
000010 * DEMO: WORKPOOL
000020 *
000030 DEFINE PRINTER(1) OUTPUT 'WORKPOOL'
000040 *
000050 READ (20) AUTOMOBILES BY MAKE STARTING FROM 'F'
000060    DISPLAY(1) MAKE COLOR MODEL HORSEPOWER
000070             NUMBER-OF-CYLINDERS
000080 END
BROWSE-VIW:AUTOMOBILES ------------------------- Row 0 of 20 - Columns 016 052
 COMMAND===>                                                    SCROLL===> CSR
   MAKE           MODEL                HORSE COLOR
** ******************** top of list *******************
   FERRARI        365 GTB/4             350  BLUE
   FERRARI        365 GTB/4             350  WHITE
   FERRARI        DINO 246 GT           193  WHITE
   FERRARI        DINO 246 GT           193  WHITE
   FERRARI        365 GTB/4             350  WHITE
   FERRARI        DINO 246 GT           193  BLACK
   FERRARI        DINO 246 GT           193  BLACK
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right :s
```

Additionally, when displaying a view definition, a special `GENERATE` command is available that automatically generates data definition statements for a Natural program source.

Objects Accessible with Entire System Server



If Entire System Server is installed at your site, Natural ISPF provides access to a range of operating-specific objects, as well as to objects held by other data management systems. An overview of object types and available functions follows below.

- Data Sets and Libraries
- PDS Members
- System Facilities
- Job Information
- CA Panvalet and CA Librarian Members

Data Sets and Libraries

Maintaining data sets on any supported operating system includes the following functions:

- Allocate data sets;
- List data sets;
- Copy data sets across operating systems;
- Rename data sets;
- Delete data sets;
- Display data set information.
- Edit sequential data sets;
- Browse sequential data sets;
- Print sequential data sets
- Compress data sets (MVS only);

- Catalog/uncatalog data sets (MVS only).

Individual members are selectable for further maintenance from a list of data sets.

PDS Members

Natural ISPF provides facilities to access MVS PDS members. Available functions include the following:

- List members in a data set according to selection criteria such as prefix or character string;
- Edit, browse, rename, print, delete a member;
- Copy a member to a different library; the target library can be of a different type and/or reside on another operating system;
- Submit job control members, check the status of a submitted job and display status information every time you invoke another system screen (for example, you are given the name of the executing step and CPU consumption, and you are notified when the submitted job is in the output queue);
- Rapid editing through the use of inline macros and the `INCLUDE-MACRO` statement, as well as use of a macro model when starting an edit session with a member. See the subsection on the ***Macro Facility*** in the section *User Applications*.
- Retrieve previous versions of edited members. When the user saves a member, this version is stored and kept for later retrieval even after the object is edited and saved again. See the subsection on ***Versioning*** in the section *General Features*.

System Facilities

Operating system-specific administrative and monitoring functions are available in the following areas:

- Display of active jobs: the jobs can be listed according to the name and type selection criteria. The display is refreshed everytime Enter is pressed, and in-memory jobs appear highlighted;
- Display the console screen: all console lines or only those waiting for an operator reply (`WTOR`) are displayed. The console screen is refreshed every time Enter is pressed. The authorized user can issue operator commands from the console screen;
- Issue operator commands; this can even be done for different CPUs and across different operating systems if Entire Net-Work is installed.
- Display of the system log (IBM operating systems);
- Display of system unit information at MVS sites. Users at MVS sites can also display `ENQUEUE` information and make use of IDCAMS services using Natural ISPF menu options.

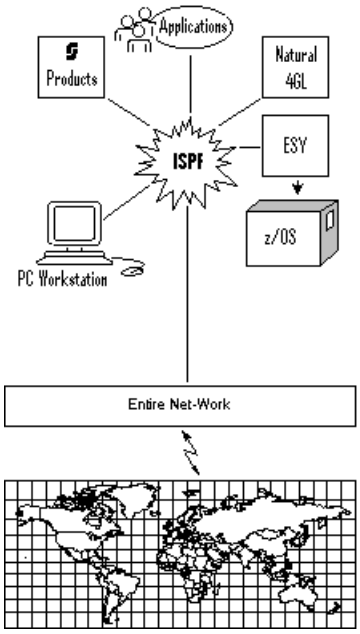
Job Information

The job information facility provides a view of job output data selectable by job name, number and type, as well as queue and spool file type. Alternatively, the required output data can be selected from a list of jobs generated using selection criteria. The user can browse spool files, hold and release them, print them or purge them from the job queue.

CA Panvalet and CA Librarian Members

Natural ISPF provides full transparency to these source management systems. All benefits of integration into the Natural ISPF environment are available, including use of the macro facility.

Multi-CPU Support with Entire Net-Work



With the transport service Entire Net-Work, Natural ISPF can serve multi-processor environments. From a single location and using the same user interface, you can access and manipulate data irrespective of its physical location or operating environment. All data stored anywhere in the computer network looks as if it is available locally. The computer on which the data resides is simply identified by a unique node number.

The following example shows you how you can select the computer in your network on which data you wish to access resides, and how you can perform a cross-computer copy operation irrespective of the underlying operating system.

The screen below shows the entry screen of the Natural ISPF PDS member maintenance facility. Imagine that you know the name of the data set you wish to access (`MBE.COMN.SOURCE`), and that the name of the required member starts with ISP. However, you are not sure on which machine within your network the data set resides.

In order to find the data, you specify the data set name and the known member prefix followed by an asterisk (`ISP*`) in the appropriate fields. You then type a question mark in the `Node` field:

```
------------------------- PDS OBJECTS - ENTRY PANEL --------------------------
COMMAND ===>


   Data Set Name ===> MBE.COMN.SOURCE
   Member        ===> ISP*
   Volume        ===>                        ( If not catalogued     )
   Password      ===>                        ( If password protected )
   Scan for      ===>
   Edit macro    ===>
   Node          ===> ?









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right :s
```

If you now press Enter, a window opens with information concerning all computers in the network. You can select any machine that is active. For example, you can see that the Madrid machine has node number `91` and is available. This machine is selected by entering `2` in the Select field:

```
------------------------- PDS OBJECTS - ENTRY PANEL --------------------------
COMMAND ===>

        +-----------------------------------------------------+
   Data ! Please select                                       !
   Memb !    Node Status   System Info                        !
   Volu !  1 148  ACTIVE   DAEF MVS/ESA  The famous F-machine !ed     )
   Pass !  2 091  ACTIVE   IPO1 MVS/XA   SAG Madrid           !tected )
   Scan !  3 031  ACTIVE        IPO1 MVS/XA   SAG Rome           !
   Edit !    033  inactive               SP3 darmstadt        !
   Node !    193  inactive               SAG Düsseldorf     !
        !    066  inactive               SAG München          !
        !    067  inactive               ZAE Alsbach          !
        !    138  inactive               SAGUK Derby          !
        !  9 206  ACTIVE   SHST MVS/ESA  SAGNA Reston         !
        !                                                     !
        !  Select ==> 2                                       !
        +-----------------------------------------------------+




Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right :s
```

Natural ISPF now searches the Madrid machine, and generates a list of all members starting with ISP in the specified data set.

The required member can now be selected for maintenance operations. Imagine you wish to copy the member from the data set to a MVS data set. The MVS machine has node number `31` (see the screen above). You select the member with the `CP` line command (for `COPY`). This opens a window with all available object types to which the member can be copied (select PDS):

```
 LIST-PDS:MBE.COMN.SOURCE(ISP*) -------------------- Row 0 of 1 - Columns 010 076
 COMMAND===>                                                  SCROLL===> CSR
   MEMBER   +--------------------------+IFIED TIME  SIZE   INIT   TID   ID
** ******** ! ENTER OBJECT              !of list ******************************
cp ISPCON   !   1 N   NATURAL object    !05/06 17:21    150    144       MBE
** ******** !   2 P   PDS member        ! of list ******************************   ↩

            !   3 D   Data set          !
            !   4 MEM PDS member      !
            !   5 D   Data set        !
            !   6 MEM  PDS member       !
            !   7 MEM PDS member       !
            !  Select ==> _7_           !
            +--------------------------+









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right :s
```

Selecting Option 7 from the list, Natural ISPF presents you with a prompt window in which you can enter the PDS characteristics of the member to be copied by overtyping the preset values in the input fields. Note especially that Natural ISPF assumes by default that copy operations are local. You must therefore specify Node Number 31 for the target MVS machine:

```
LIST-PDS:MBE.COMN.SOURCE(ISP*) -----------------  Row 0 of 1 - Columns 010 076
COMMAND===>                                                  SCROLL===> CSR
+-------------------------------------------------------------------------------+
! COPY-PDS:MBE.COMN.SOURCE(ISPCON)                                              !
! To                                                                            !
!  Library   MBE.COMN.SOURCE                                                    !
!  Element   ISPCON                                                             !
!  Type                                                                         !
!  Version                                                                      !
!  Password                                                                     !
!  Node     31                                                                  !
! Replace   NO                                                                  !
! Enter to perform, PF3 to exit.                                                !
+-------------------------------------------------------------------------------+









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right :s
```

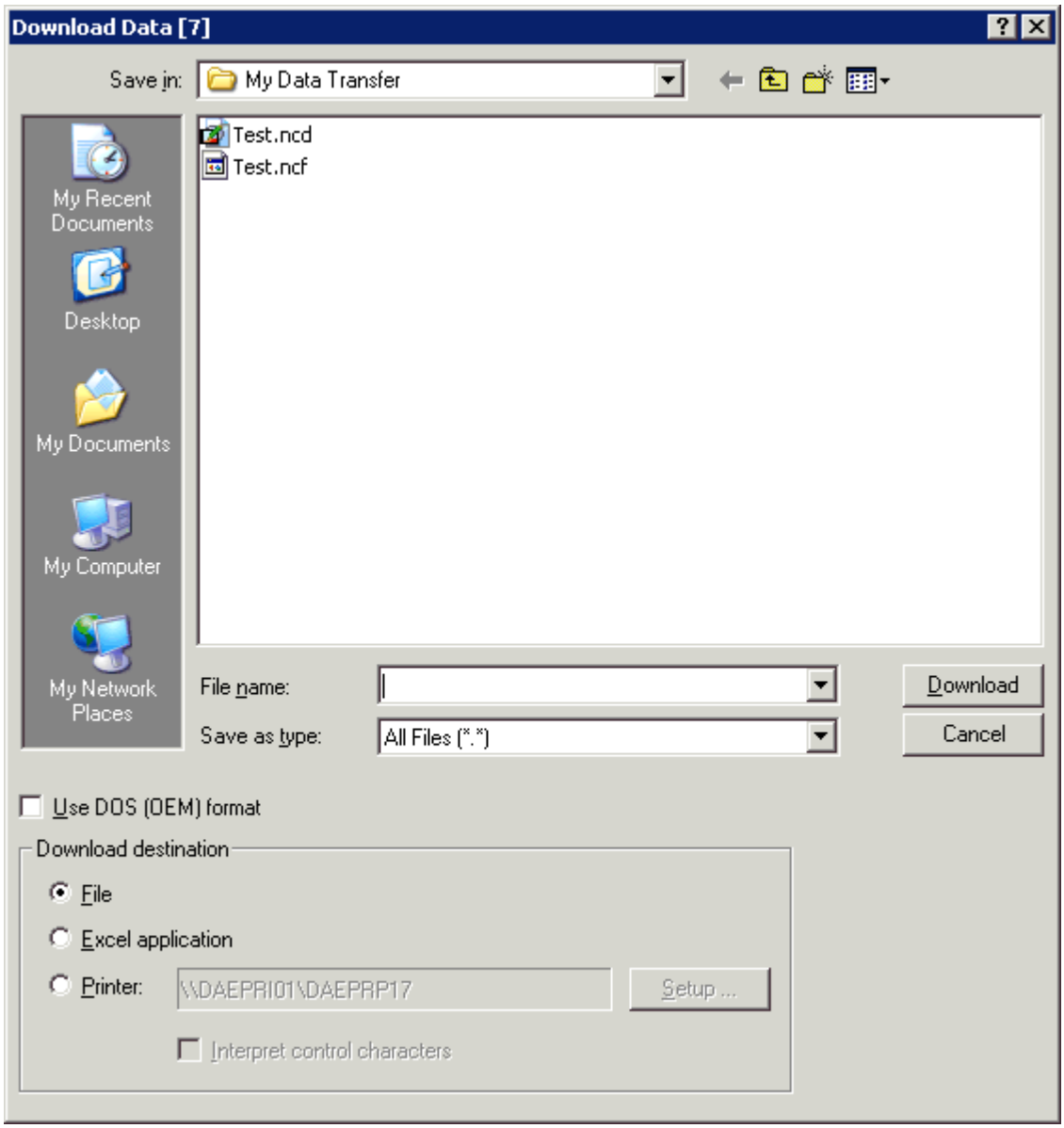
PC / Workstation Integration

If Entire Connection is installed at your site, you can integrate the PC/Workstation environment in Natural ISPF. This means you can download any Natural ISPF object (Natural program or PDS member, etc.) as a PC file, and any PC file can be uploaded and stored as a Natural ISPF object.

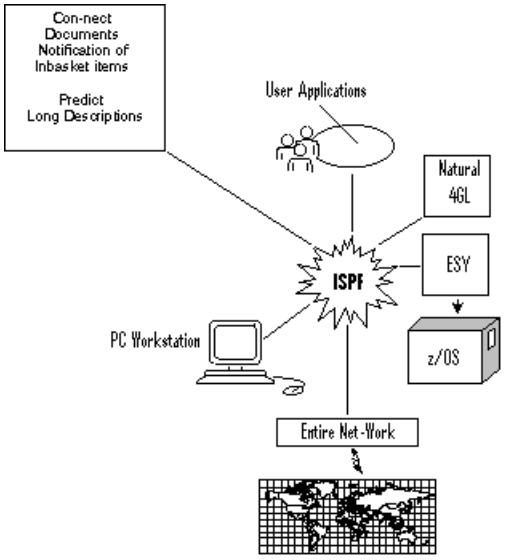
This is done using simple `IMPORT` or `EXPORT` commands from your Natural ISPF session running on your PC terminal emulation with Entire Connection active.

The following example shows a Natural ISPF browse session with a Natural program. The `EXPORT PC` command causes Entire Connection to prompt you for the name of the file under which the program is to be stored on the PC or workstation:

Entire Connection



Objects from Other Products



Objects from our other products can be integrated in Natural ISPF. The interfaces to Con-nect and Predict provide the following functions:

- `EXPORT` any Natural ISPF object (Natural program or PDS member, etc.) to Con-nect and store it as a document and/or send it as a memo to other Con-nect users.
- `IMPORT` a Con-nect document to Natural ISPF and store it as any Natural ISPF object.
- `LIST` Con-nect documents, `BROWSE`, `EDIT`, `DELETE` and `PRINT` them, display information about a document. The advantages of doing this from Natural ISPF are:
  - No need to leave the working environment to handle Con-nect documents
  - Full Editor functionality
  - Use of Natural ISPF features such as multi-session operations, split-screen, flexible layout of lists
- Natural ISPF provides for a procedure that informs you with a message whenever a new item in your Con-nect inbasket is detected: there is no need to keep leaving the Natural ISPF environment to check.
- Predict long descriptions can be displayed and edited directly from Natural ISPF, again using all the advantages of the Editor and Natural ISPF features. The use of Predict sets for Natural objects is described in the section on Natural objects above.

Example Con-nect

The following example screen shows a Natural ISPF session in split-screen mode. The top session contains a list of documents in a Con-nect folder, the bottom session contains an edit session with one of the listed documents:

```
LIST-DOC:MBE(*)/FOLDER=WORK ---------------------- Row 0 of 2 - Columns 045 076
 COMMAND===>                                                  SCROLL===> CSR
   Document                          Int.Number          Created    File
** ******************************** top of list ********************************
   Editor 134                        0001484258          1989-11-24 ISPF
   Textfile                          0006899084          1993-05-06 ISPF
** ***************************** bottom of list ******************************


EDIT-DOC:MBE(Textfile) ---------------------------------------- Columns 001 072
 COMMAND===>                                                  SCROLL===> CSR
****** ***************************** top of data ****************************
000001 This is a text file used to test the NSPF interface to CON-NECT
****** **************************** bottom of data ****************************









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right :s
```

Parallel edit sessions with several Con-nect documents are possible, allowing you to perform operations such as cross-session copying.

Example Predict

The following example screen shows a Natural ISPF session in split-screen mode. The top session contains an edit session with the long description of the Natural program displayed in edit mode in the bottom session.

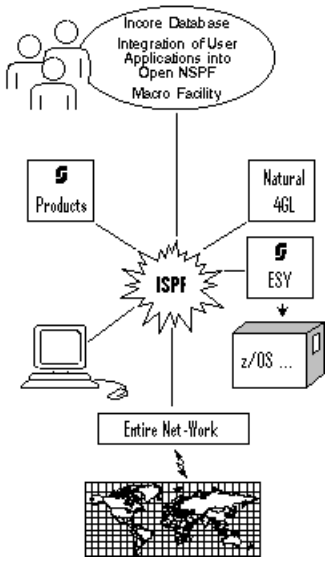
```
EDIT-PRD:DEMO(DEMOREP) ---------------------------------------- Columns 001 072
 COMMAND===>                                                    SCROLL===> CSR
****** ***************************** top of data ******************************
000001 Ref-Number: N-4711
000002 Function  : Report of Automobiles
000003 Descriptor: MAKE
000004 Standard Copycodes to be used:
000005             EXSTARTC
000006             EXEND--C
****** **************************** bottom of data ****************************
EDIT-NAT:DEMO(DEMOREP)-Program->Struct-Free-42K --------------- Columns 001 072
 COMMAND===>                                                    SCROLL===> CSR
000100 *
000110 DEFINE DATA LOCAL
000120 1 AUTOMOBILES-VIEW  VIEW OF AUTOMOBILES
000130   2 CAR-DESCRIPTION
000140     3 MAKE
000150     3 MODEL
000160     3 BODY-TYPE
000170     3 NUMBER-OF-CYLINDERS
000180     3 HORSEPOWER
000190     3 PISTON-DISPLACEMENT
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right :s
```

Again, several edit sessions with programs and their descriptions can be active at the same time, allowing easy cross-checking, editing and cross-session operations.

5 User Applications

Overview

Natural ISPF provides a number of special programming facilities that help you realize powerful and flexible application development at your site with a minimum of effort. The following figure lists the available facilities:



- **Incore Database:**
  Allows you to maintain complex data structures (free format texts, tables, reports, files, etc.) in user memory in a so-called incore file. Data in an incore file can be manipulated using familiar Natural statements and/or by integrating the Software AG Editor in Natural.
- **Integration of User Applications into Open NSPF:**
  The Open NSPF facility allows you to extensively customize your Natural ISPF environment to suit the (changing) working needs of your installation. You can create new objects and relate functions to them, thus making the whole Natural ISPF infrastructure available for these objects. The integration of new objects can be made easy by the ability to modify existing menus and create new ones.
- **Macro Facility:**
  Natural ISPF facilitates rapid editing using a special macro feature: source lines of any kind can be generated automatically, thus making much time-consuming typing in of data redundant.

Incore Database

The Natural ISPF Incore Database facility supports the creation of “incore files” in user memory, allowing you to maintain complex data structures and to perform a wide range of actions on these structures. With the Incore Database, you can use Natural to handle free format texts, reports, files and tables.

Using incore files in application development has several advantages:

- You can integrate Software AG Editor functions in Natural programs, enabling flexible manipulation of your incore files;
- An incore file is a temporary workfile of unlimited space running in unshared memory;
- You can have your personal environment to test and prototype your applications away from your site's database administration activities. After prototyping, no further source changes are required to access shared data in a real database;
- If you wish to keep the contents of an incore file permanently, you can write them to a container file. You can retrieve the incore file later, thus avoiding the need to regenerate the data from your programs.

Defining and Maintaining Incore Files

The structure of an incore file is defined in the same way as any other Natural view. Incore files are not identified by a file number but by an identifier, thus allowing multiple copies of a file created with the same view to be in the database at the same time. Once an incore view has been defined, incore files can be created in any of a number of ways:

- Implicitly using a `STORE` statement in a Natural program
- Explicitly using a `CALLNAT` statement with `ACTION=CREATE`
- Dynamically by writing the output of a Natural program to an incore file with `WRITE` or `DISPLAY`

You can use standard Natural DML statements to retrieve, add, modify and delete records in an incore file. Operations on an incore file as a whole (`EDIT`, `BROWSE`, `DELETE`) can be performed using the `CALLNAT` statement. The `CALLNAT` statement is also used to realize program-controlled editing with Software AG Editor commands for display (for example, `BNDS`, `EXCLUDE`, `INCLUDE`, `CAPS`, `HEX`, etc.), scroll (for example, `FIND`, `BOTTOM`, `TOP`, `UP`, `DOWN`, `LOCATE`, etc.), and text manipulation (`CHANGE`, `DELETE`, `JUSTIFY`, etc.).

Example

The following example program illustrates the implicit creation of an incore file using the STORE statement. A text line is written to the file and the file is then presented to you in an edit session, allowing you to add further text lines:

```
DEFINE DATA LOCAL USING IDBI---L
LOCAL
1 TEXT VIEW OF ISP-IDB-TEXT
  2 LINE
END-DEFINE
*
ASSIGN TEXT.LINE    = 'THIS IS THE FIRST LINE OF TEXT'
STORE  TEXT IDENTIFIER = 'MYTEXT'     /* Create incore text file
*
RESET INCORE-CTL INCORE-DATA          /* Enter data in file
SET CONTROL  'WB'                     /* using an ISPF Editor
SET CONTROL  'L'
SET KEY ALL
ASSIGN ACTION          = 'EDIT'
ASSIGN FILE-IDENTIFIER = 'MYTEXT'
CALLNAT INCORE USING INCORE-CTL INCORE-DATA /* Invoke the Editor as shown below
*
READ TEXT IDENTIFIER = 'MYTEXT'             /* Read file for further processing
.....
END-READ
END
```

Running this program results in the following display:

```
EDIT:-MYTEXT-------------------------------------------------------------------
COMMAND==>
**** ****************************** top of data ******************************
0001 THIS IS THE FIRST LINE OF TEXT
**** ***************************** bottom of data *****************************









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      :I    :CC   QUIT  :D    RFIND RCHAN UP    DOWN  :DD   RIGHT LEFT  CURSO
```

All Editor functions are now available: there is no need to code them into your program. The program does not regain control until the Editor session is terminated with PF3.

Integration of User Applications into Open NSPF

Natural ISPF provides a unified environment that allows you to access various types of external objects (such as Natural programs, PDS members, or jobs) using the same user interface. Objects can be accessed using Natural ISPF menus (examples are entry screens for object types or the administration screen) and/or commands (for example: `EDIT PDS`).

The Open NSPF facility allows you to customize the Natural ISPF environment by:

- integrating your own site-specific objects
- defining new commands or command synonyms relevant to your site
- modifying existing menus or defining new ones

Taking advantage of this facility yields a number of benefits:

- The Natural ISPF user interface can be expanded to encompass your site-specific objects, giving you a truly single system image of all your resources;

- All advantages of Natural ISPF features are available for your integrated objects and applications: split-screen, multi-session operations and cross-session operations are among the most obvious.

Implementing New Objects and Commands

New site-specific objects and new commands are defined to Natural ISPF by adding an object code or a command code to the so-called "Site Control Table". In the same table, new objects can additionally be related to functions (typically, existing Natural ISPF functions).

Implementing the corresponding logic in Natural ISPF is done by writing a subprogram and copying it to the Natural ISPF execution library (`SYSLIB`). Each new object and command has a unique program, which is called by Natural ISPF whenever the object is accessed or the command is issued.

Example New Object

You could integrate management of your personnel file in Natural ISPF by integrating the new object `EMPLOYEE`. You can relate this new Natural ISPF object to functions such as `LIST`, `DELETE`, or `EDIT`. If a function is then invoked for object `EMPLOYEE`, the subprogram linked to the object is executed, handling all object-specific logic.

Example New Command

You could define the new command `MAIL` in the Site Control Table. Each time this command is issued, the corresponding logic is executed, which could consist of a search of your office system and a returned message indicating your mail status (for example, whether you have received new items).

Menu Customization

Menu customization means you can create new menus or modify existing ones. You can also change the default menu, causing a menu of your choice to be displayed when you log on to Natural ISPF.

Within the context of the Open NSPF facility, the ability to customize menus is significant because it allows you to make the integration of and access to new objects and applications at your site much more comfortable and user-friendly. As a prime example, the interfaces to Con-nect and Predict were implemented using the Open NSPF facility.

For another example, having defined `EMPLOYEE` as an object to Natural ISPF and linked it to certain functions (for example, `LIST`, `DELETE`, `EDIT`, ...), you could modify the Natural ISPF main menu, making the maintenance of employees a selectable option. The corresponding `EMPLOYEE` menu could contain parameter fields such as `NAME`, `FIRST NAME`, `SEX`, `CITY` to make a search request to the personnel file as comfortable and flexible as possible:

```
---------------------------- EMPLOYEES - ENTRY PANEL -------------------------
COMMAND ===> LIST


   Name          ===> BA*                     ( Or name pattern for LIST )
   First name    ===> *                       ( Or name pattern for LIST )
   Sex           ===> M                       ( F, M )
   City          ===>
   Records       ===>                         ( Nr. of records to be shown )









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right :s
```

Entering data in the fields with the `LIST` command as shown above will list male employees whose last names start with `BA`. The list could be displayed in Software AG Editor format, provided a suitable incore file is created by the application. All Editor and Natural ISPF features for lists are available.

The power of Open NSPF is limited only by the resources available at your site and the imagination of the programmers.

Macro Facility

Natural ISPF provides a macro feature that allows you to use the Natural language to generate text of any kind. In a process known as macro expansion, text is generated, which can be done by substituting variables, repeating blocks, generating blocks conditionally, even performing screen or file I/Os.

The macro feature is useful when you are creating different sources, all of the same structure but with different content. The macro feature thus supports you in editing programs and other sources, offering a number of benefits when developing applications with Natural ISPF. Among them are:

- Rapid editing: the automatic generation of text lines in any source and any format relieves you of much routine editing work;

- Error elimination: the automatic generation of text lines eliminates a main source of error (typing and syntax errors);
- Reduced disk space requirements, for example for job control: jobs generated using macro expansion need not be held in personal libraries.

Using Macros

Used in Natural programs, the macro feature is an extension of the Natural language, consisting of:

- special processing statements executed when the program is compiled
- variables in source lines substituted by valid values at compilation time

These special processing lines and variables are distinguished in the source by preceding them with the macro character.

As a simple example of macro processing lines and variables, if you define the following macro:

```
0010 § MOVE 'PERSONNEL' TO #FILE-NAME (A32)
0020 § MOVE 'NAME'      TO #KEY       (A32)
0030 READ §#FILE-NAME BY §#KEY
```

the following source line is generated at compilation time:

```
READ PERSONNEL BY NAME
```

By varying the values, a variety of source lines can be created. This makes it possible to access each Natural view at your site from the same “skeleton” program.

However, the macro feature is not restricted to Natural programs and can be used in Natural ISPF in a number of ways, as described below.

- Macro Objects
- Inline Macros
- Edit Macros

Macro Objects

Macro objects are a special type of Natural object and can be accessed and maintained as any other Natural object. They contain macro processing lines and macro variables and can be used to generate any kind of text (for example, documents, Natural sources, 3GL sources, job control). When a macro is executed, it is expanded and the output is written to the user workpool, where it can be further maintained and turned into another source. Macros can also be referenced from other Natural ISPF objects to generate text lines.

Inline Macros

Other Natural ISPF objects (Natural programs, PDS members, etc.) can use macro variables in their source. They can also reference a macro object using the special `INCLUDE-MACRO` *`macro-name`* statement. When the object is compiled (for a Natural object) or submitted (for non-Natural objects), the variables are substituted, the specified macro object is executed, and the generated lines are included in the source.

The generated output of objects that use the macro facility is written to the user workpool, where it can be checked and further handled.

Edit Macros

When starting an edit session with a Natural ISPF object (Natural program, PDS member, etc.), you can specify a macro object to be used as a model for the new edit session. The specified macro is executed and the generated lines written to the new edit session. The lines generated in this way are protected, but you can reserve some places in the macro object at which you can add specific code in the members generated with the edit macro.

A special `REGENERATE` command is available here, which regenerates the macro model, but retains the specific code you have added.

Example Macro

The following is an example macro object. Macro processing lines are identified by the macro character in the first column (in our example, the paragraph sign (§) is used). This example generates JCL lines for an `IEBCOPY` job, allowing flexible selection of members to be copied:

```
000010 § DEFINE DATA LOCAL
000020 § 1 #JOB            (A08)
000030 § 1 #USER           (A08)
000040 § 1 #IN-DSNAME      (A44)
000050 § 1 #IN-VOLSER      (A6)
000060 § 1 #MEMBER         (A8)
000070 § 1 #OUT-DSNAME     (A44)
000080 § 1 #OUT-VOLSER     (A6)
000090 § 1 #DD-VOL         (A20)
000100 § *
000110 § 1 PDS-DIRECTORY-VIEW  VIEW OF PDS-DIRECTORY
000120 §   2 NODE
000130 §   2 DSNAME
000140 §   2 VOLSER
000150 §   2 MEMBER
000160 § END-DEFINE
000170 § INPUT  'COPY MEMBERS  ==>' #MEMBER
000180 §     /  'FROM DSNAME   ==>' #IN-DSNAME   'VOLSER==>' #IN-VOLSER
000190 §     /  'TO   DSNAME   ==>' #OUT-DSNAME  'VOLSER==>' #OUT-VOLSER
000200 § COMPRESS *USER 'IEB' INTO #JOB LEAVING NO SPACE
000210 § MOVE     *USER TO #USER
000220 //§#JOB     JOB  §#USER,CLASS=G,MSGCLASS=X
000230 //COPY   EXEC PGM=IEBCOPY
000240 //SYSPRINT DD SYSOUT=*
000250 § IF #IN-VOLSER NE ' '
000260 §   COMPRESS ',VOL=SER=' #IN-VOLSER INTO #DD-VOL LEAVING NO SPACE
000270 § ELSE
000280 §   RESET #DD-VOL
000290 § END-IF
000300 //I     DD DISP=SHR,DSN=§#IN-DSNAME§#DD-VOL
000310 § IF #OUT-VOLSER NE ' '
000320 §   COMPRESS ',VOL=SER=' #OUT-VOLSER INTO #DD-VOL LEAVING NO SPACE
000330 § ELSE
000340 §   RESET #DD-VOL
000350 § END-IF
000360 //O     DD DISP=SHR,DSN=§#OUT-DSNAME§#DD-VOL
000370 //SYSIN DD *
000380   C I=I,O=O
000390 § IF NOT ( #MEMBER = '*' OR = ' ' )
000400 § FIND PDS-DIRECTORY-VIEW
000410 §     WITH DSNAME = #IN-DSNAME
000420 §      AND VOLSER = #IN-VOLSER
000430 §      AND MEMBER = #MEMBER
000440   S M=§MEMBER
000450 § END-FIND
000460 § END-IF
```

When this macro is `RUN`, macro expansion takes place and you are prompted for the source members and target destination:

```
COPY MEMBERS  ==>
FROM DSNAME   ==>                                                VOLSER==>
TO   DSNAME   ==>                                                VOLSER==>
```

You can make use of generic search criteria (wildcard selection) to select members according to name pattern. The example below selects all members in the source data set that have the dollar sign ($) as third character:

```
COPY MEMBERS  ==> __$*
FROM DSNAME   ==> nispf.qa.cases                                 VOLSER==>
TO   DSNAME   ==> mbe.comn.source                                VOLSER==>
```

When the source members and target data set have been specified, the following JCL is generated and written to the user workpool, where it can be further edited if required and submitted:

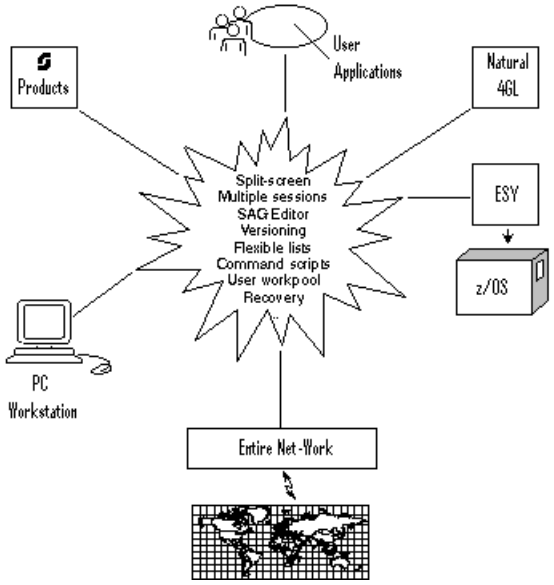
```
000001 //MBEIEB     JOB  MBE,CLASS=G,MSGCLASS=X
000002 //COPY   EXEC PGM=IEBCOPY
000003 //SYSPRINT DD SYSOUT=*
000004 //I      DD DISP=SHR,DSN=NISPF.QA.CASES
000005 //O      DD DISP=SHR,DSN=MBE.COMN.SOURCE
000006 //SYSIN DD *
000007   C I=I,O=O
000008   S M=BF$LS
000009   S M=DA$LS
000010   S M=DJ$LS
000011   S M=DS$AL
000012   S M=DS$IN
000013   S M=DS$LS
000014   S M=MV$LS
000015   S M=NV$LS
000016   S M=PV$LS
```

The `IEBCOPY` job with a valid member list is thus generated automatically, no further manual intervention is required.

6 General Features

Overview

Much of the power of Natural ISPF lies in the wealth of special features it offers to make your work in application development as comfortable as possible. Natural ISPF features are too numerous to elaborate here, and such facilities as user profiling, individual user defaults, abbreviations for long data set names, flexible PF-key assignments, activity tracing and easy administration should these days be taken for granted. However, a few other, particularly interesting features are presented below just to whet your appetite:



Split-Screen

Working with Natural ISPF in split-screen mode means dividing your terminal screen horizontally into two sections using the `SPLIT` command and running a Natural ISPF session in each section. You can change the portion of the terminal screen devoted to each session by moving the cursor to where you wish to split the screen and repeating the `SPLIT` command.

The split-screen feature is useful for easy control of parallel sessions. For example, you could run a Natural program from an edit session in one part of the screen and immediately see the resulting output in a session with the User Workpool in another part of the screen.

If both sessions are edit sessions, cross-session actions are possible. For example, you can move or copy data from one session to the other. A common way to work with Natural ISPF is to run multiple sessions from your terminal with two sessions in split-screen mode. For an example of working in split-screen mode, see the description of Natural members and Views in the section ***Natural ISPF Objects***, as well as in the subsection on *Multiple Sessions* below.

Multiple Sessions

Working in multi-session mode means starting several parallel Natural ISPF sessions. You can control up to 20 active Natural ISPF sessions from your terminal. Sessions can be suspended (put to the back of the other sessions) or resumed (brought to the front) as required.

Typical examples of multi-session operations are copying data from one edit session to another, or editing and running a Natural program in one session and checking the resulting output in another.

The following figure illustrates a terminal screen with Natural ISPF in multi-session mode:

```
--------------------------NATURAL-ISPF-MAIN-MENU-----------------------------
>-------------------------NATURAL-VIEW--ENTRY-PANEL-----------------------------
>>EDIT-PDS:MBE.SYSF.SOURCE(NOJC02)------------------ >>> Versioning is invoked
 COMMAND===>                                                 SCROLL===> CSR
****** ***************************** top of data ****************************
000001 //SNNO2J02 JOB   SN,CLASS=G,MSGCLASS=X,MSGLEVEL=(1,1)
000002 //*
000003 //*        DEMONSTRATION JOB
000004 //*
000005 //IEFBR14  EXEC  PGM=IEFBR14
000006 //SYSPRINT DD    SYSOUT=*
000007 //*
000008 //STEP01   EXEC  PGM=SNABND,PARM='C0004'
000009 //STEPLIB  DD    DSN=NATOP.V110.LOAD,DISP=SHR
000010 //
LIST-NAT:NSPF101------------------------------ Row 11 of 323 - columns 010 076
 COMMAND===>                                                 SCROLL===> CSR
   MEMBER            PGMTYPE     SM S/C VERSION   USERID   DATE     TIME  VV.MM
   ISUO#4            Subprogram  S  S/C 8.2 0001  BLI      20110131 17:46 01.17
   ZDICSET4          Program     S  S/C 8.2 0001  BLI      20110131 16:48 01.01
   BLICATAL          Text           S   8.2 0001  BLI      20110128 11:42
   ISP-TECH          Program     S  S/C 8.2 0001  BLI      20110124 17:57 01.26
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right Curso
```

The screen shows two suspended sessions, one containing the Natural ISPF main menu, the other a session with the Views facility; two session are displayed in split-screen mode, the top session contains an editing session with a PDS member, the bottom session contains a list of Natural objects.

Switching between sessions is easy: just place the cursor on the required session and issue the `POP` command to bring the marked session to the front. This process is made even easier by assigning the `POP` command to a PF-key. A session is then selected by positioning the cursor on it and pressing the PF-key.

Software AG Editor

Natural ISPF includes the Software AG Editor, which is the same as the editor used, for example, by Natural for UNIX. The Editor looks familiar to programmers used to time-sharing environments (for example, TSO/ISPF on z/OS), and use of the Editor requires little or no extra training.

The Editor is used to list, browse and edit all objects accessible via Natural ISPF, meaning that you have one uniform editor to display and handle the whole range of objects at your site, irrespective of any underlying operating systems.

Additionally, the Editor is especially adapted to the Natural ISPF environment and your special needs in application development. This means that the Editor provides object-specific commands such as `CHECK`, `RUN`, `CAT`, `STOW` for Natural objects and `SUBMIT` for job control members.

Together with Natural ISPF features such as multiple sessions, split-screen, and cross-session operations such as data transfer across operating systems, the Editor is a powerful tool in the hands of application developers and system programmers alike.

The following example simply shows an edit session with a macro-type Natural program:

```
EDIT-NAT:NSPFEXAM(MAC-MVS3)-Macro->Struct-Free-42K ------------ Columns 001 072
 COMMAND===>                                                    SCROLL===> CSR
****** ***************************** top of data ****************************
000010 § DEFINE DATA LOCAL
000020 § 1 #JOB            (A08)
000030 § 1 #USER           (A08)
000040 § 1 #IN-DSNAME      (A44)
000050 § 1 #IN-VOLSER      (A6)
000060 § 1 #MEMBER         (A8)
000070 § 1 #OUT-DSNAME     (A44)
000080 § 1 #OUT-VOLSER     (A6)
000090 § 1 #DD-VOL         (A20)
000100 § *
000110 § 1 PDS-DIRECTORY-VIEW  VIEW OF PDS-DIRECTORY
000120 §   2 NODE
000130 §   2 DSNAME
000140 §   2 VOLSER
000150 §   2 MEMBER
000160 § END-DEFINE
000170 § INPUT  'COPY MEMBERS  ==>' #MEMBER
000180 §     /  'FROM DSNAME   ==>' #IN-DSNAME   'VOLSER==>' #IN-VOLSER
000190 §     /  'TO   DSNAME   ==>' #OUT-DSNAME  'VOLSER==>' #OUT-VOLSER
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right :s
```

Versioning

Natural ISPF can keep previous versions of objects after they have been edited and saved (or, in the case of Natural programs, stowed). Product-specific versioning features are supported for data set members kept in CA Librarian.

Previous versions of Natural objects or PDS members are treated as separate objects in Natural ISPF and can be listed, browsed and deleted as any other member. You can retrieve a previous version for further editing by storing it in the object library under a different name. Additionally, specific versions can be held permanently, meaning they are not automatically deleted when the maximum number of versions is reached. A special command is available that allows you to see the difference between any selected previous version and the current version of a member.

The advantages of the Natural ISPF versioning feature to application developers are obvious: the history of applications can be tracked, earlier versions can be reverted to and using other Natural ISPF features such as split-screen and cross-session operations, data can easily be transferred between versions.

The following example shows the effect of the `DIFFERENCE` command issued for a previous version: a message in the prefix area marks those lines that have changed from the selected previous version to the current version. In our example, the message `Old>` marks those lines that have been modified (lines 490 and 530), and the message `New>` marks the line that has been added (line 540):

```
DIFFERENCE-NV:JWO(EXAM)-Ver<-1>-93/12/20-11:04:19 ------------- Columns 011 076
COMMAND===>                                                   SCROLL===> CSR
     000460 *
     000470   DECIDE ON EVERY VALUE OF #FUNCTION
     000480     VALUE 'REPORT', 'BOTH'
     000490       DISPLAY (1) HORSEPOWER MAKE MODEL COLOR
   Old>0490       DISPLAY (1) HORSEPOWER MAKE MODEL COLOR NUMBER-OF-CYLINDERS
     000500     VALUE 'TUNING', 'BOTH'
     000510       IF #NEW NE HORSEPOWER
     000520         MOVE #NEW TO HORSEPOWER
     000530         DISPLAY (2) HORSEPOWER MAKE MODEL COLOR #NEW
   Old>0530         DISPLAY (2) HORSEPOWER MAKE MODEL COLOR NUMBER-OF-CYLINDER
   New>0540         WRITE (2)  'Car is updated'
     000550         UPDATE
     000560         ADD 1 TO #UPD-CNT
     000570         PERFORM ET-LOGIK
     000580       END-IF
     000590     VALUE 'STOP' , 'END'
     000600       STOP
     000610     NONE
     000620       REINPUT WITH TEXT 'INVALID FUNCTION SELECTED'
     000630   END-DECIDE
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right :s
```

Flexible Lists

Before an object can be edited or browsed, you must first locate it. No problem?

How often has it happened that you cannot quite remember the name of the program you need, or that you have forgotten in which library you have stored a certain job? Alternatively, if you have to change the name of, say, a certain parameter and you must change all references to this parameter, would it not be useful to list only those members in the object library that contain the parameter name?

The Software AG Editor integrated in Natural ISPF provides some powerful listing capabilities:

- Libraries, or members in a known library can be listed according to a name pattern using wildcard symbols. For example, specifying `NSPF*` for data set name in a `LIST` command lists all data sets with the prefix NSPF. Other search criteria can be a certain character or string in a certain place of the name, or an object-specific characteristic, such as member type. This allows you to locate objects with a minimum of knowledge of names.
- Using a special scan option, objects can be listed according to a character string they contain. The resulting list indicates the number of occurrences of the string and displays the first occurrence.

The following example shows a list generated using the scan option: all objects in Natural library `NSPFEXAM` starting with the string IDB and containing the string `READ` are listed:

```
LIST-NAT:NSPFEXAM(IDB*)/SC=READ ---------------- Row 0 of 23 - Columns 010 076
 COMMAND===>                                                  SCROLL===> CSR
   MEMBER            PGMTYPE     SM S/C  NUM FIRST FOUND
** ******************************** top of list ******************************
   IDB-DEMO          Program     S  S/C    2 READ INCORE-SEMINAR IDENTIFIER = '
   IDB-HITN          Subprogram  S  S/C    3 * SUBPROGRAM TO READ/WRITE A FILE
   IDB-HITP          Program     S  S/C    5 CALLNAT 'IDB-HITN' 'R'
   IDB-HIT1          Program     S  S/C    1 CALLNAT 'IDB-HITN' 'R'
   IDB-HIT4          Program     S  S/C    6 READ INCORE-MUSIC  IDENTIFIER
   IDB-KEYS          Program     S  S/C    2 READ(100)  EMPLOYEES
   IDB-MOVI          Program     S  S      2 READ MOVIES IDENTIFIER = 'MYMOVIE'
   IDB-STO1          Program     S  S      2 READ IDB-PERSON  IDENTIFIER = 'TAB
   IDB-TABP          Program     S  S/C    2 READ INCORE-SEMINAR IDENTIFIER = '
   IDB-TEXP          Program     S  S      2 READ TEXT IDENTIFIER = 'MYTEXT'
   IDB-TEXT          Program     S  S/C    2 READ TEXT IDENTIFIER = 'MYTEXT'
   IDB-001P          Program     S  S/C    4 READ EMPLOYEES
   IDB-002P          Program     S  S/C    4 READ TEXT IDENTIFIER = 'SAMP1'
   IDB-003P          Program     S  S/C    2 READ(100)  EMPLOYEES
   IDB-004P          Program     S  S/C    5 1 #REPORT-ALREADY-EXISTS(L)
   IDB-006P          Program     S  S/C    5 PERFORM READ-DIRECTORY
   IDB-007P          Program     S  S/C   11 1 READ-FILE VIEW OF READ-FILE
   IDB-008P          Program     S  S/C    6 1 #PERSONNEL-CV-ALREADY-EXIST(L)
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Split End   Suspe Rfind Rchan Up    Down  Swap  Left  Right :s
```

Once a list has been generated, all Editor commands are available that help you find the item you need (`LOCATE` a line, `FIND` a string, `UP`, `DOWN`, `TOP`, `BOTTOM`, `EXCLUDE` lines from display, etc.). Additionally, some commands are available that allow you to customize the list according to your needs:

- With the `SORT` command, you can change the order of objects according to the data in any displayed column. For example, in reverse chronological order according to the `DATE` column or in alphabetical order according to the `USER` column.
- The `LAYOUT` command facilitates an even more powerful rearrangement of listed information. It allows you to select the columns to be displayed, suppress other columns, decide in which order the columns are to be displayed, and define the order of information within the columns. Once the list is customized as you require, you can store the layout, and future lists of this object type will take the defined layout.

Command Scripts

You can write and store a series of Natural ISPF commands in a member of any type (Natural, PDS member, etc.). Such a member is known as a command script. The script can be executed with the `PLAY` function command. The commands are then executed sequentially.

For example, playing a member with the following content from a Natural edit session:

```
CHANGE 'READ' 'FIND' ALL
CHANGE 'FIND-FILE' 'READ-FILE' ALL
STOW
END
```

changes all occurrences of `READ` into `FIND`, then changes all occurrences of `FIND-FILE` back to `READ-FILE` before stowing the program and ending the session.

The example makes clear that the command script feature has many advantages and uses.

- Frequently-used or repetitive command sequences can be kept as a script and executed with minimal editing effort.
- Procedures can be automated by storing them in a script. For example, a script can be specified in your user profile and executed every time you log on to Natural ISPF. Your session is thus tailored to your requirements before the first Natural ISPF screen is even displayed.
- Command scripts can be nested (that is, a `PLAY` command within a command script), allowing maximum flexibility of automated command sequences.
- Command scripts can be generated automatically using the macro facility. This allows for dynamic scripts with variable command parameters that are prompted at execution time.

An executing command script can be interrupted using the `PAUSE` command coded in the script. This allows editing before the script continues. A script which is executed by the `PLAY` command is stored in the User Workpool. When a script is interrupted by a `PAUSE` command or an error, the lines not yet executed are also written to the User Workpool and can be modified.

User Workpool

The User Workpool is an internal pool used as destination for output from a number of different sources. Output can be listed and further maintained in the workpool facility, which is a standard option on the Natural ISPF main menu.

The following objects covered by this document are written to the User Workpool:

- The output of objects that use the macro facility after macro expansion

- The output of any Natural program or Natural utility outside of Natural ISPF that defines the workpool as a printer
- A command script executed by the PLAY command; also, if a command script is interrupted by the PAUSE command or an error, the command lines not yet executed are written to the workpool and can be modified

Output written to the User Workpool can be handled like any other Natural ISPF object, and can be stored permanently by copying to another object type in Natural ISPF.

A typical way of working with Natural ISPF is to have an edit session with a Natural program, with the output of the program in the workpool in another session so that you can see the effect of your editing immediately.

Recovery

Natural ISPF provides a comfortable recovery facility for lost files after an abnormal termination or system crash.

A backup of the file you are editing is written after a certain number of lines have been modified (this number is specified in your personal user profile).

If you then lose files for any reason, Natural ISPF will notify you with a message at your next logon, asking you to list the recovery files. If you issue the RECOVERY command, you are presented with a list of recovery files. You can select any recovery file from the list for EDIT or DELETE.

If more than one file is to be recovered, you can re-edit one file. After saving it, pressing PF3 returns you to the list of recovered files and you can re-edit the next one.