

Natural Engineer

Version 4.3.1

Application Restructuring  
for Mainframes

**Manual Order Number: NEE431-024MFR**

This document applies to Natural Engineer version 4.3.1 and to all subsequent releases.

Specifications contained herein are subject to change, and these changes will be reported in subsequent revisions or editions.

Readers' comments are welcomed. Comments may be addressed to the Documentation Department at the address on the back cover. Internet users may send comments to the following e-mail address:

[document@gensystems.com](mailto:document@gensystems.com)

Copyright © September 2001, Generation Systems Ltd., East Grinstead, UK.

The documentation often refers to numerous hardware and software products by their trade names. In most, if not all cases, these designations are claimed as trademarks or registered trademarks by their respective companies.

# TABLE OF CONTENTS

<b>ABOUT THIS MANUAL.....</b>	<b>1</b>
Purpose of this manual .....	1
Target Audience .....	1
Typographical Conventions used in this manual .....	2
How this manual is organized .....	3
Terminology .....	4
Related Literature .....	7
<b>1. APPLICATION RESTRUCTURING USING OBJECT BUILDER.....</b>	<b>9</b>
Chapter Overview.....	9
Object Builder Overview.....	10
Object Builder Line Range Process.....	15
Object Builder Presentation Process.....	39
Object Builder Technical Process.....	51
<b>INDEX.....</b>	<b>73</b>



# ABOUT THIS MANUAL

## Purpose of this manual

---

This manual contains the Application Restructuring for Natural Engineer version 4.3.1.

It describes the use of Object Builder to restructure Natural applications within Natural Engineer.

The topics covered are:

- How Object Builder works.
- Restructuring a Natural application to separate the presentation logic from the business logic.
- Restructuring a Natural application to separate the database access (technical) logic from the business logic.

## Target Audience

---

The target audience for this manual is intended to be any User of Natural Engineer 4.3.1 at any level of experience.

*Note: This manual should be read in conjunction with the Natural Engineer Application Analysis & Modification manual, which explains the Impact and Modification processes in more detail.*

## Typographical Conventions used in this manual

---

The following conventions are used throughout this manual:

<b>UPPERCASE TIMES</b>	Commands, statements, names of programs and utilities referred to in text paragraphs appear in normal (Times) uppercase.
<b>UPPERCASE BOLD COURIER</b>	In illustrations or examples of commands, items in uppercase bold courier must be typed in as they appear.
< >	Items in angled brackets are placeholders for user-supplied information. For example, if asked to enter <file number>, you must type the number of the required file.
<u>Underlined</u>	Underlined parts of text are hyperlinks to other parts within the online source manual. This manual was written in MS-Word 97 using the "hyperlink" feature.

The following symbols are used for instructions:

⇒	Marks the beginning of an instruction set.
□	Indicates that the instruction set consists of a single step.
1.	Indicates the first of a number of steps.

## How this manual is organized

---

This manual is organized to reflect all the Application Restructuring options of Natural Engineer version 4.3.1 in the following chapters:

<b>Chapter</b>	<b>Contents</b>
1	Describes the Object Builder options, which provide the facility to restructure Natural applications by selecting individual lines of source code, selecting all presentation logic within an application or selecting all the technical logic within an application.

## Natural Engineer Application Restructuring

# Terminology

---

It is assumed that you are familiar with general Natural and mainframe terminology, as well as the terms and concepts relating to MS-Windows environments. This section explains some terms that are specific to the Natural Engineer product.

## Analysis

The Analysis process of Natural Engineer searches application data within the Natural Engineer Repository, according to specified Search Criteria and generates reports on the search results.

## Application

An Application is a library or group of related libraries, which define a complete Application. In Natural Engineer, the Application can have a one-to-one relationship with a single library of the same name, or a library of a different name, as well as related steplibs. The Application refers to all the source code from these libraries, which Natural Engineer loads into the Repository.

## Browser

An Internet Browser such as Microsoft Internet Explorer™ or Netscape™.

## Category

Categories in Natural Engineer specify whether and how a Modification is applied to the Natural code. Valid categories are: Automatic change, Manual change, Reject the default Modification, No change to the data item, and the data item is in Generated Code.

A category is further broken down according to type of change (for example: Keyword, Literal, Data Item, Database Access, Definition).

## Consistency

An option in the Analysis process that causes Natural Engineer to trace an Impact through the code, using left and right argument resolution to identify further code impacted by the code found.

## **Environment**

The Environment process is the means by which Natural Engineer generates a structured view of the application code in the Natural Engineer Repository. This provides application analysis reports and inventory information on the application and is used as the basis for Impact Analysis.

## **Exception**

An Exception is an Item identified as impacted that does not require a Modification. Where there are a few similar Exception Items, they can be treated as Exceptions, and rejected in the Modification review process. Where there are many similar (therefore not Exceptions), consideration should be given to changing the Search Criteria so they are not identified as impacted in the first place.

## **Generated Code**

This is code which has been generated by a Natural code generator, such as Construct, and which is not normally modified directly in the Natural editor.

## **Impact**

An Impact is an instance of a Natural code Item; e.g., data item or statement (a “hit” scored by the Analysis process) that matches the defined Search Criteria used in the Analysis process.

## **Iteration**

An Iteration is one examination cycle of a field identified according to the specified Search Criteria. For example, one Iteration is reading the field right to left. Multiple Iterations are performed when the option of ‘Consistency’ or Multi Search is requested for Analysis, and Natural Engineer performs as many Iterations as necessary to exhaust all possibilities of expressing and tracing the field, and can be limited by a setting in the NATENG.INI file.

## **Library**

A single library of source code, which exists in the Natural system file.

## **Natural Engineer Application Restructuring**

### **Modification**

A Modification is a change suggested or made to an object or data item resulting in the required compliance of that object or data item. Modifications in Natural Engineer are classified according to Category and Type.

### **Presentation Split Process**

The Presentation Split Process is a sub-function of the Object Builder function that removes screen I/O statements from current application objects and places them in generated subprograms.

### **Soft Link**

A Soft Link is where a link between two objects has been defined using an alphanumeric variable rather than a literal constant.

### **Technical Split Process**

The Technical Split Process is a sub-function of the Object Builder function that results in the encapsulation of each database access within the application, into a sub-program so that the application is separated into 'presentation and logic' and 'database access'.

### **Type**

The Type of Modification available, for example: Data Item, Keyword and Literal.

### **TLM**

Text Logic Members are used to contain the code required to support inclusion of common code into the application. An example of this is the code to include into an application before updating a database.

## Related Literature

---

The complete set of Natural Engineer manuals consists of:

**1. Natural Engineer Concepts and Facilities (NEE431-006ALL)**

The Concepts and Facilities manual describes the many application systems problems and solutions offered by Natural Engineer, providing some guidelines and usage that can be applied to Natural applications.

**2. Natural Engineer Release Notes (NEE431-008ALL)**

The Release Notes describe all the information relating to the new features, upgrades to existing functions and documentation updates that have been applied to Natural Engineer 4.3.1.

**3. Natural Engineer Installation Guide (NEE431-010ALL)**

The Installation Guide provides information on how to install Natural Engineer on both PC and mainframe platforms.

**4. Natural Engineer Administration Guide (NEE431-040WIN)**

**Natural Engineer Administration Guide (NEE431-040MFR)**

The Administration Guide provides information on all the various control settings available to control the usage of the different functions within Natural Engineer.

**5. Natural Engineer Application Management (NEE431-020WIN)**

**Natural Engineer Application Management (NEE431-020MFR)**

The Application Management manual describes all the functions required to add Natural applications into the Repository.

**6. Natural Engineer Application Documentation (NEE431-022WIN)**

**Natural Engineer Application Documentation (NEE431-022MFR)**

The Application Documentation manual describes all the available functions to document a Natural application within the Repository. These functions will help enhance / supplement any existing systems documentation such as BSD / CSD / Specifications etc.

## **Natural Engineer Application Restructuring**

### **7. Natural Engineer Application Analysis and Modification (NEE431-023WIN)**

#### **Natural Engineer Application Analysis and Modification (NEE431-023MFR)**

The Application Analysis and Modification manual describes all the available functions to carry out analysis of Natural applications; including basic keyword searches. The modification process is described and detailed to show how it can be applied to modify single selected objects within a Natural application, or the entire Natural application in one single execution.

### **8. Natural Engineer Application Restructuring (NEE431-024WIN)**

#### **Natural Engineer Application Restructuring (NEE431-024MFR)**

The Application Restructuring manual describes the analysis and modification functionality required to carryout some of the more sophisticated functions such as Object Builder.

### **9. Natural Engineer Utilities (NEE431-080WIN)**

#### **Natural Engineer Utilities (NEE431-080MFR)**

The Utilities manual describes all the available utilities found within Natural Engineer and, when and how they should be used.

### **10. Natural Engineer Reporting (NEE431-025ALL)**

The Reporting manual describes each of the reports available in detail, providing report layouts, how to trigger the report and when the report data becomes available. The various report-producing mediums within Natural Engineer are also described.

### **11. Natural Engineer Batch Processing [Mainframes] (NEE431-026MFR)**

The Batch Processing manual describes the various batch jobs (JCL) and their functionality.

# APPLICATION RESTRUCTURING USING OBJECT BUILDER

## Chapter Overview

---

Application Restructuring using Object Builder provides the facilities to restructure an existing Natural application to enable the essential business processing logic to be used in different ways. Examples of restructuring include:

- Creating new objects to execute common processes that are unnecessarily repeated many times over within individual objects.
- Creating new objects from existing monolithic applications to enhance the applications maintainability.
- Separating presentation logic to facilitate uses of different user interfaces, such as GUI or the web.
- Separating database access statements to facilitate use of alternate database architectures.

The Object Builder processes encompasses the use of the Impact Analysis process to identify the source code to be restructured and prepares the necessary triggers to action this.

The Modification process is used to review the Impact results and Modification criteria prior to creating the necessary components. Once the Modification criteria have been reviewed, Modification can be applied and will generate the necessary components.

The topics covered are:

1. [Object Builder Overview](#)
2. [Object Builder Line Range Process](#)
3. [Object Builder Presentation Process](#)
4. [Object Builder Technical Process](#)

## Object Builder Overview

---

The Object Builder process is triggered by using the Impact search keyword 'OBJECT BUILDER'. There are three variants of Object Builder available:

### 1. Object Builder Line Range.

This can be applied for individual objects within an application selecting the source code to be restructured by specifying start and end line ranges.

### 2. Object Builder Presentation.

This is applied against all objects within an application and it will restructure all INPUT statements.

### 3. Object Builder Technical.

This is applied against all objects within an application and it will restructure all Database access statements.

All three options perform in the same way. After the criteria have been specified, Impact Analysis is executed to generate the modification criteria and then Modification is applied to the object(s).

The Object Builder modification will generate new subprogram and Parameter Data Area objects for each individual specification (Line Range, Presentation or Technical), and modify the original object being restructured with the necessary linkage code (CALLNAT).

The Object Builder Technical option generates one further object – a set of Local Data Area objects containing the main file views used within an application. There will be one Local Data Area per view.

Application Restructuring using Object Builder

The following Figure 1-1 illustrates the basic object generation for a simple Object Builder Line Range selection for object XX001P01 from the sample application HOSPITAL.

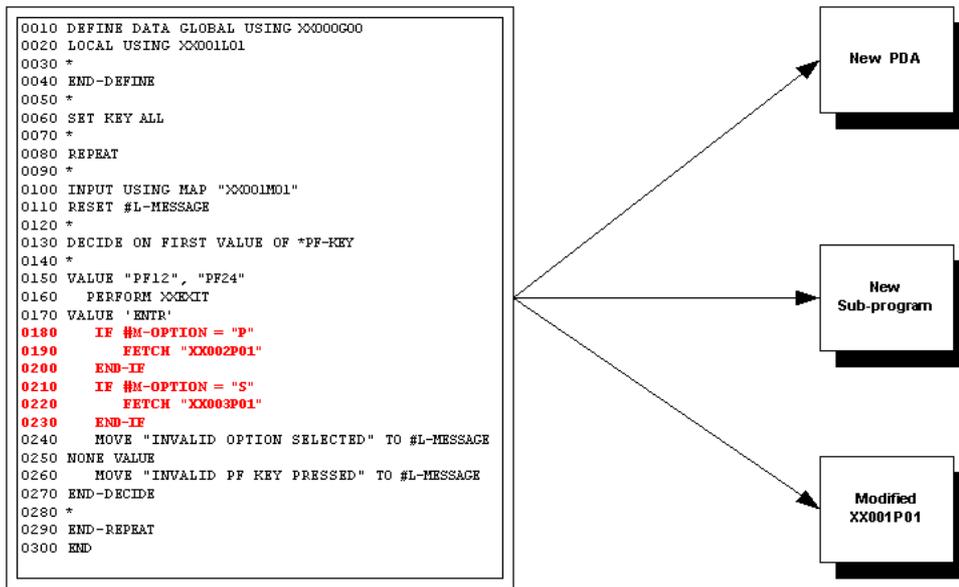


Figure 1-1 The object generation for a simple Object Builder Line Range selection

Note: The same basic object generation also applies to Object Builder Presentation.

# 1

## Natural Engineer Application Restructuring

The following Figure 1-2 illustrates the basic object generation for a simple Object Builder Technical selection for object XX022P01 from the sample application HOSPITAL.

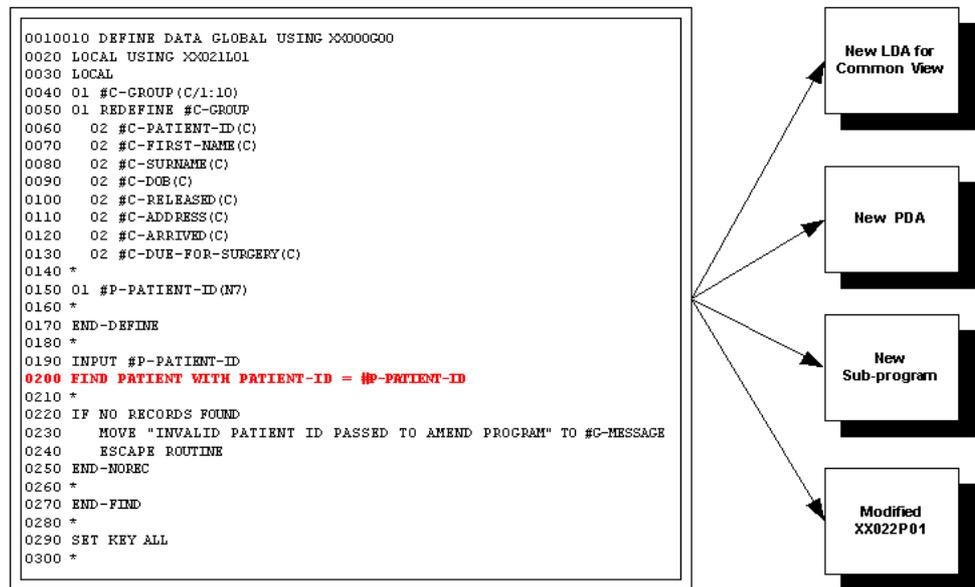


Figure 1-2 The object generation for a simple Object Builder Technical selection

## Object Names used for the Generated Objects

For each object generated by the Object Builder process, the names used for the new objects are based on the template masks defined in the COMPONENT\_OBJECT\_NAME and COMPONENT\_OVERFLOW\_NAME parameters in the ###CINI text member.

The COMPONENT\_OBJECT\_NAME parameter is set to a default value of #####\*% where each mask character is positional in respect of the original object name.

Mask	Description
#	The same character found in the original object name is used. A maximum of 6 are permitted per template mask
*	The object type will be inserted. For a subprogram this will be 'N' and for a Parameter Data Area this will be 'A'. At least 1 must be present within the template mask.
%	A sequential number for the new object name. The range used is 1-9 and then A-Z. At least 1 must be present within the template mask and it cannot be placed at the start of the mask, i.e., %#####* is not permitted.

The following examples illustrate the object names applied for various masks based on the original object name XX021P01:

For mask #####\*%,  
new subprogram name = XX021PN1  
new PDA name = XX021PA1

For mask ###\*%  
new subprogram name = XX0NN1  
new PDA name = XX0AA1

For mask #####\*%%  
new subprogram name = XX021N01  
new PDA name = XX021A01

# 1

## Natural Engineer Application Restructuring

The sequential number assigned to the mask character ‘%’ is determined by the following rules:

5. Check to see if the new object name exists on the base Natural application library.
6. Check to see if the new object name exists on the Modification library.

If either of these two conditions is true, then the next sequence number is assigned. This prevents any duplication of object names by ensuring the new object names are unique within the selected application.

The `COMPONENT_OVERFLOW_NAME` parameter follows the same rules and definitions. It is used when Object Builder Technical process modification is being executed in unattended operation mode using the Task Scheduler. As the parameter name implies, it is used as an overflow when the primary mask provided by `COMPONENT_OBJECT_NAME` is exhausted.

*Note: For more information on the ###CINI text member parameters `COMPONENT_OBJECT_NAME` and `COMPONENT_OVERFLOW_NAME` refer to Chapter 2 in the Natural Engineer Administration Guide for Mainframes manual.*

*Note: For more information on the Task Scheduler refer to Chapter 1 in the Natural Engineer Utilities for Mainframes manual.*

## Object Builder Line Range Process

---

The Object Builder Line Range Process requires a range of source code lines within an object to be selected by specifying the start and end line numbers. Once all the required line ranges have been specified the Impact process is executed to create the necessary impact and modification data. Once the Impact results have been reviewed, the Modification process can be applied.

The Object Builder Line Range process identifies all the source code instances of the line ranges specified, each line range is placed into Object Builder generated subprograms. Any data items associated with the line ranges are placed into Object Builder generated Parameter Data Areas, to support the data transfer between the calling object and the new subprograms.

## Objects generated by the Object Builder Line Range Process

The Object Builder Line Range process will generate the following objects during the Modification execution:

### **1. Generate a subprogram object.**

A new subprogram object is created on the Modification library with the source code for the selected line ranges from the original object copied into it. If any parameters are required, then they will be placed into a new Parameter Data Area object and this will be referenced within the subprogram data definitions.

### **2. Generate a Parameter Data Area (PDA) object.**

A PDA object is only generated, if the Impact process has identified any data that needs to be passed between the calling object and the new subprogram. This will contain all the required data items as defined in the original object.

### **3. Modify the original object to reference the new subprogram object.**

The original object will be modified to comment out the source code for the line range specified. This is replaced with a CALLNAT statement to call the new subprogram and pass any required parameters.

The following statements demonstrate the complexity of the object builder process:

1. If a data item is used in the original object outside of the object builder range THEN it must be added to the generate PDA.
2. If a data item is not used outside of the object builder range THEN don't create a PDA data item.
3. If a data item used inside the object builder range has a value set previously THEN move the value to a new PDA data item.
4. A system variable from the object builder range is used outside of the range THEN create a new PDA data item and move the system value to it in the new object and then change all references in the initial program.
5. The program has an environment setting outside of the object builder range THEN copy all environment settings to the new sub program.
6. The code in the object builder range uses back references THEN change references based on the position of the code in the new sub program.
7. A data item is only used inside the object builder line range THEN create a local data area inside then object and add the data item.

# 1

## Natural Engineer Application Restructuring

### How to specify Line Ranges

The Line Ranges of the source code can be specified in one of two ways:

1. Using search keyword OBJECT BUILDER using the Impact Criteria screen and specifying the line range in the Modification Parameters section.
2. Using the Object Builder Processing option from the Impact Data Preparation sub menu.

### Line Ranges specified using the Impact Criteria Screen

The Object Builder Line Ranges can be specified using the Impact Criteria screen and selecting the search keyword OBJECT BUILDER.

A keyword value is mandatory for this search keyword and must be the object name that contains the source code that the line range applies to.

The line ranges are input using the Modification parameter. This must use the following convention: **nnnn-mmmm,pppppppp,oooooooo**, where:

<b>nnnn</b>	Specifies the start of line range value All 4 characters must be used, for example 10 must be input as 0010.
-	A hyphen (-) is used to separate the start and end line ranges.
<b>mmmm</b>	Specifies the end of line range value All 4 characters must be used, for example 10 must be input as 0010. The value must be equal to or greater than the start of line value.
<b>pppppppp</b>	Specifies an override name for the generated PDA object. For example if object XX021P01 is having Line Range modification applied and the template mask is #####*%, then the default name for the PDA would be XX021PA1. If an override name were input, for example PDA01, then the name of the generated PDA would be PDA01.

*Note: If a PDA override name is used, then a place delimiter comma (,) needs to be inserted prior to the PDA override name.*

**oooooooo** Specifies an override name for the generated subprogram object.

For example if object XX021P01 is having Line Range modification applied and the template mask is #####\*%, then the default name for the subprogram object would be XX021PN1. If an override name were input, for example SUBPGM01, then the name of the generated subprogram would be SUBPGM01.

*Note: If a PDA override name is not present, then two place delimiter commas (,) needs to be inserted prior to the Subprogram override name. If PDA override name is present then a single place delimiter comma (,) needs to be inserted.*

The following examples illustrate some valid Modification parameters based on applying Object Builder line ranges to object XX021P01, with template mask #####\*%:

Modification parameter = 0180-0230  
new subprogram name = XX021PN1  
new PDA name = XX021PA1

Modification parameter = 0180-0230,,  
new subprogram name = XX021PN1  
new PDA name = XX021PA1

Modification parameter = 0180-0230,SUBPGM01  
new subprogram name = SUBPGM01  
new PDA name = XX021PA1

Modification parameter = 0180-0230,PDA01,  
new subprogram name = XX021PN1  
new PDA name = PDA01

Modification parameter = 0180-0230,PDA01,SUBPGM01  
new subprogram name = SUBPGM01  
new PDA name = PDA01

# 1

## Natural Engineer Application Restructuring

The following Figure 1-3 illustrates the Impact Criteria screen showing an Object Builder line range criteria specified.

```

                                     Impact Criteria           Application: HOSPITAL
                                                                Version: 01
Search Keyword...: OBJECT BUILDER_____
Object Name.....: _____
Keyword Value (Object Name)
XX001P01_____

Replace Value....:
_____
                _ Data Item _ Literal

Modification Parms 0180-0230,PDA01,SUBPGM01
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help      Exit      Add                                     Types      Main
```

**Figure 1-3 Impact Criteria screen with Object Builder line range criteria specified**

*Note: For more information on the Impact Criteria screen refer to Chapter 1 in the Natural Engineer Application Analysis & Modification for Mainframes manual.*

## Line Ranges specified using the Object Builder Processing Screen

Specifying Object Builder Line Ranges using the search keyword OBJECT BUILDER requires prior knowledge of the line ranges required, either using a hard copy of an object listing or viewing the object in the Natural editor.

A more interactive way of specifying Object Builder Line Ranges is to use the Object Builder Processing screen, which provides the object source code for you to select the line ranges.

The same functionality provided by the Modification parameters on the Impact Criteria screen is available from the Object Builder Processing screen.

The Object Builder Processing screen is accessed by selecting option 'P' (Impact Data Preparation) from the Analysis Menu screen, and then by selecting option 'O' (Object Builder Processing) from the Impact Data Preparation Sub-menu screen..

For each line range specified, an Impact criteria record is still generated and can be viewed in the Impact Search Criteria Summary screen, and can be identified by option set to 'OEM' in the Opt column.

*Note: For more information on the Impact Search Criteria Summary screen refer to Chapter 1 in the Natural Engineer Application Analysis & Modification for Mainframes manual.*

## 1

## Natural Engineer Application Restructuring

The following Figure 1-4 illustrates the Object Builder Processing screen showing an Object Builder line range criteria specified.

```

- Object Builder Processing - Application: HOSPITAL
Object Name: XX001P01 Line Number....: 0160 Version: 01
Set Start Point: 0180
Set End Point...: 0230 Objects: PDA01,SUBPGM01____
Source Code
-----
0160 VALUE "PF12", "PF24"
0170 PERFORM XXEXIT
0180 VALUE 'ENTR'
0190 IF #M-OPTION = "P"
0200 FETCH "XX002P01"
0210 END-IF
0220 IF #M-OPTION = "S"
0230 FETCH "XX003P01"
0240 END-IF
0250 MOVE "INVALID OPTION SELECTED" TO #L-MESSAGE
0260 NONE VALUE
0270 MOVE "INVALID PF KEY PRESSED" TO #L-MESSAGE
0280 END-DECIDE
0290 *
0300 END-REPEAT
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
Help Exit Range Save Prev Next Top Bot Main

```

**Figure 1-4 Object Builder Processing screen with Object Builder line range criteria specified**

*Note: For more information on the Object Builder Processing screen refer to Chapter 1, section Impact Data Preparation in the Natural Engineer Application Analysis & Modification for Mainframes manual.*

The following Figure 1-5 illustrates the Impact Criteria Summary screen showing an Object Builder line range 'OEM' criteria (normal view).

```

                                     Impact Criteria Summary   Application: HOSPITAL
                                                                Version: 01
Search      Keyword      Search      Replace      Replace
Keyword     Value        Value        Value        TLM
_ OBJECT BUILDER  XX001P01

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  DeleC Exit  GetSa SaveA Add  Prev  Next          W<  W>  Main
    
```

Figure 1-5 Impact Criteria Summary screen with Object Builder line range 'OEM' criteria

# 1

## Natural Engineer Application Restructuring

The following Figure 1-6 illustrates the Impact Criteria Summary screen showing an Object Builder line range 'OEM' criteria (scrolled to the right via 'PF11' (W>)).

```
Replace          Replace Cons Opt Modification
Defn.           Position      Parameters
                OEM 0180-0230,PDA01,SUBPGM01

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
Help  DeleC Exit  GetSa SaveA Add  Prev Next          W<  W>  Main
```

**Figure 1-6 Impact Criteria Summary screen with Object Builder line range 'OEM' criteria**

*Note: For more information on the Impact Criteria Summary screen refer to Chapter 1 in the Natural Engineer Application Analysis & Modification for Mainframes manual.*

## Example of Object Builder Line Ranges

The following example illustrates a simple Object Builder Line Range being applied to object XX001P01 from the sample application HOSPITAL. The start and end line ranges will be specified using the Impact search criteria option.

An application called HOSPITAL has been created with all the objects from the sample application HOSPITAL extracted and loaded into the Repository. The Application Preferences have been specified with the Modification library set to HOSPITAX. The library HOSPITAX is empty. The example begins from the Impact search criteria stage.

**Step 1** Identify the start and end line ranges to be restructured from program object XX001P01. The line ranges to be used will start at line number 0180 and end at line 0230.

These are marked in bold in the following source code listing for object XX001P01.

# 1

## Natural Engineer Application Restructuring

```
0010 DEFINE DATA GLOBAL USING XX000G00
0020 LOCAL USING XX001L01
0030 *
0040 END-DEFINE
0050 *
0060 SET KEY ALL
0070 *
0080 REPEAT
0090 *
0100 INPUT USING MAP "XX001M01"
0110 RESET #L-MESSAGE
0120 *
0130 DECIDE ON FIRST VALUE OF *PF-KEY
0140 *
0150 VALUE "PF12", "PF24"
0160     PERFORM XXEXIT
0170 VALUE 'ENTR'
0180     IF #M-OPTION = "P"
0190         FETCH "XX002P01"
0200     END-IF
0210     IF #M-OPTION = "S"
0220         FETCH "XX003P01"
0230     END-IF
0240     MOVE "INVALID OPTION SELECTED" TO #L-MESSAGE
0250 NONE VALUE
0260     MOVE "INVALID PF KEY PRESSED" TO #L-MESSAGE
0270 END-DECIDE
0280 *
0290 END-REPEAT
0300 END
```

**Step 2** Having identified the start and end line ranges, the Impact search criteria is specified using the Impact Criteria screen. The specification includes; a search keyword of OBJECT BUILDER, a keyword value of XX001P01 (i.e., the object containing the required line range) and Modification parameters of 0180-0230.

*Note: the Modification parameters do not include any PDA or Subprogram object name overrides. These object names will be based on the template masks defined in the COMPONENT\_OBJECT\_NAME parameter, which is set to '#####\*%'.*

The following Figure 1-7 illustrates the Impact Criteria screen after all the details have been specified.

```

                                     Impact Criteria           Application: HOSPITAL
                                                                Version: 01
Search Keyword...: OBJECT BUILDER_____
Object Name.....: _____

Keyword Value (Object Name)
XX001P01_____

Replace Value....:
_____
                _ Data Item _ Literal

Modification Parms 0180-0230_____
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---
      Help      Exit      Add                                     Types      Main
    
```

**Figure 1-7 Impact Criteria screen with line range 0180-0230 specified for object XX001P01**

# 1

## Natural Engineer Application Restructuring

**Step 3** With the Impact search criteria specification complete, the Impact Analysis process is invoked by selecting option Impact Execution from the Analysis menu.

**Step 4** When Impact execution has completed, the Impact results are reviewed using the Impact Element Maintenance option, accessed from the Analysis menu.

The following Figure 1-8 illustrates the Impact Element Categorization screen displaying the Impact results for object XX001P01.

```

                                     - Element Categorization -      Application: HOSPITAL
                                                                    Version: 01
Object: XX001P01                                                    Page: 1
Field : LINE RANGE 0180-0230
Attr  :          Impact Type: G                                Ext. Object:

Search Criteria: Searching All Objects for OBJECT BUILDER XX001P01

Stmt Source Line
0180 VALUE 'ENTR'
0190   IF #M-OPTION = "P"
0200     FETCH "XX002P01"
0210   END-IF
0220   IF #M-OPTION = "S"
0230     FETCH "XX003P01"

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help           Exit                               Prev Next      Ctxt      Main
```

**Figure 1-8 Impact Element Categorization screen displaying the Impact results for object XX001P01**

**Step 5** Having reviewed that the Impact results are correct, the Modification criteria are reviewed using the Modification Element Maintenance option, accessed from the Modification menu.

The following Figure 1-9 illustrates the Modification Element Categorization screen displaying the Modification criteria for object XX001P01.

```

- Element Categorization -      Application: HOSPITAL
                               Version: 01
                               Page: 1
Object: XX001P01
Field : LINE RANGE 0180-0230
Attr  : External Object Name:
Category : A Automatic      Replace Defn: _____ TLM: _____
Type    : G_ O.B. Line Range      Pos: _____
Replace Value: _____
User Comment : _____
TLM Data   : _____
Reason : Data item can be automatically changed
User ID:      Last Update   :
              Execution Date : Not yet modified

Stmt Source Line
0180 VALUE 'ENTR'
0190 IF #M-OPTION = "P"
0200 FETCH "XX002P01"
0210 END-IF
0220 IF #M-OPTION = "S"
0230 FETCH "XX003P01"
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help      Exit      Updt SCrit Prev Next +Parm Ctxt      Main

```

**Figure 1-9 Modification Element Categorization screen displaying the Modification criteria for object XX001P01**

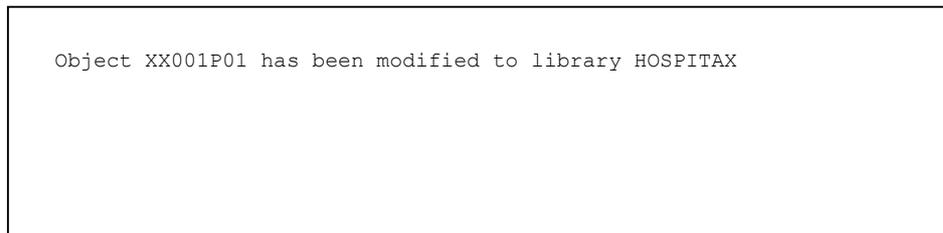
This shows that the line range 0180 to 0230 has a Modification category of A (automatic) and the Modification type is set to O.B. Line Range. We are now ready to apply the modification.

# 1

## Natural Engineer Application Restructuring

**Step 6** The Modification will be applied by using 'PF5' (Mod) from the Modification Object Categorization screen. When the Modification has completed an information window is displayed, detailing that object XX001P01 has been modified to library HOSPITAX.

The following Figure 1-10 illustrates the Modification information window.



**Figure 1-10 Modification information window**

**Step 7** Reviewing the Modification results. To do this we will logon to the Natural library HOSPITAX. This library was empty at the start of this example. It now contains 3 objects:

1. A modified program object: XX001P01.
2. An Object Builder generated Parameter Data Area: XX001PA1.
3. An Object Builder generated Subprogram: XX001PN1

The following Figure 1-11 illustrates the object list for Modification library HOSPITAX.

```

User XGSLXX                - LIST Objects in a Library -                Library HOSPITAX

Cmd  Name          Type          S/C  SM Version  User ID  Date      Time
---  *             *             *    *  *      *         *         *
___  XX001PA1       Parameter     S    S  2.2.08  XGSLXX   2001-09-21 12:49:00
___  XX001PN1       Subprogram    S    S  2.2.08  XGSLXX   2001-09-21 12:49:00
___  XX001P01       Program       S    S  3.1.04  XGSLXX   2001-09-21 12:49:00

                                                    3 Objects found

Top of List.
Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---
      Help  Print Exit          --    -    +    ++    >    Canc
    
```

Figure 1-11 Object list for Modification library HOSPITAX

## 1

**Natural Engineer Application Restructuring**

**Step 8** Review the modified object XX001P01. It can be seen that the source code for the line range selected, has been commented out and replaced by a CALLNAT to the Object Builder generated subprogram XX001PN1. The modified code is shown in bold in the following listing.

```
0010 DEFINE DATA GLOBAL USING XX000G00
0020 LOCAL USING XX001L01
0030 *
0040 END-DEFINE
0050 *
0060 SET KEY ALL
0070 *
0080 REPEAT
0090 *
0100 INPUT USING MAP "XX001M01"
0110 RESET #L-MESSAGE
0120 *
0130 DECIDE ON FIRST VALUE OF *PF-KEY
0140 *
0150 VALUE "PF12", "PF24"
0160 PERFORM XEXIT
0170 VALUE 'ENTR'
0180 /* IF #M-OPTION = "P" /* NEE OLD CODE
0190 /* FETCH "XX002P01" /* NEE OLD CODE
0200 /* END-IF /* NEE OLD CODE
0210 /* IF #M-OPTION = "S" /* NEE OLD CODE
0220 /* FETCH "XX003P01" /* NEE OLD CODE
0230 /* END-IF /* NEE OLD CODE
0240 CALLNAT 'XX001PN1' /* NEE MODIFIED
0250 #M-OPTION /* NEE MODIFIED
0260 MOVE "INVALID OPTION SELECTED" TO #L-MESSAGE
0270 NONE VALUE
0280 MOVE "INVALID PF KEY PRESSED" TO #L-MESSAGE
0290 END-DECIDE
0300 *
0310 END-REPEAT
0320 END
```

**Step 9** Review the Object Builder generated subprogram XX001PN1. This subprogram contains the code that was selected by the line range from object XX001P01.

```
0010 * Subprogram: XX001PN1
0020 *****
0030 * Created by NEE on 2001-09-21 at 12:49:47.8
0040 * Created from XX001P01 from line range 0180-0230
0050 *****
0060 DEFINE DATA
0070 PARAMETER USING XX001PA1
0080 *
0090 END-DEFINE
0100     IF #M-OPTION = "P"
0110         FETCH "XX002P01"
0120     END-IF
0130     IF #M-OPTION = "S"
0140         FETCH "XX003P01"
0150     END-IF
0160 END
```

**Step 10** Review the Object Builder generated PDA XX001PA1. A PDA has been generated because the line range selected for the source code in object XX001P01 contains variables which need to be passed between the calling (XX001P01) and called (XX001PN1) objects. The PDA is used by the Object Builder generated subprogram XX001PN1.

```
0010 DEFINE DATA PARAMETER
0020 1 #M-OPTION(A1)
0030 END-DEFINE
```

**Step 11** To check that the new source codes execute correctly, all objects excluding object XX001P01, from the HOSPITAL library need to be copied to the Modification library HOSPITAX.

All objects can then be re-stowed and the HOSPITAL application invoked by executing object XX001P01.

*Note: For more information on the Impact Analysis and Modification options refer to the Natural Engineer Application Analysis & Modification for Mainframes manual.*

## Object Builder Line Range replacing REINPUT Statements

If an application is being restructured to make use of alternate user interfaces, i.e., all INPUT statement logic is to be separated to make use of client-server type architecture, consideration must be given to the usage of REINPUT statements within the application.

In a client-server environment, such as EntireX, the REINPUT statement will not work. Using the Object Builder Line Range option, a range of lines can be specified that include REINPUT statements, during modification the REINPUT logic is replaced with response code handling. The response code will be returned to the calling object, which can then take the appropriate action.

*Note: This option is a variation on the Object Builder Presentation process, which will apply the same Modifications, but for the whole application. The Object Builder Line Range replacing REINPUT statement can be used if being applied to individual objects only.*

## Example of Object Builder Line Range replacing REINPUT Statement

The following example illustrates a simple object containing REINPUT statements. The example shows the object before applying the Object Builder Line Range replacing REINPUT modification, the object after modification has been applied and the generated objects created by the Object Builder process.

Sample object before modification with intended line ranges marked in bold:

```
0010 DEFINE DATA LOCAL
0020 /*
0030 01 #A          (A10)
0040 /*
0050 END-DEFINE
0060 /*
0070 INPUT USING MAP 'REINPMAP'
0080 /*
0090 IF #A EQ ' '
0100 REINPUT 'INVALID VALUE DETECTED' MARK #A
0110 END-IF
0120 /*
0130 IF #A NE 'B'
0140 REINPUT *1212
0150 END-IF
0160 /*
0170 END
```

To handle the REINPUT statements within this a sample object, Object Builder Line Range is set to start at line 0090 and end at line 0150.

Sample object after modification has been applied (all modifications are in bold):

```

0010 DEFINE DATA LOCAL
0020 /*
0030 01 #A          (A10)
0040 /*
0050 01 #NEE@REINPUT-MSG (A79) /* NEE MODIFIED
0060 01 #NEE@REINPUT-MARK (I4) /* NEE MODIFIED
0070 01 #NEE@REINPUT-REPLY (I4) /* NEE MODIFIED
0080 01 #NEE@REINPUT-MSG# (I4) /* NEE MODIFIED
0090 01 #NEE@REINPUT-GROUP /* NEE MODIFIED
0100 02 #NEE@REINPUT-FIELD (A65/1:50) /* NEE MODIFIED
0110 02 #NEE@REINPUT-FIELD-POS (I04/1:50) /* NEE MODIFIED
0120 END-DEFINE
0130 /*
0140 INPUT USING MAP 'REINPMAP'
0150 /*
0160 /* IF #A EQ ' ' /* NEE OLD CODE
0170 /* REINPUT 'INVALID VALUE DETECTED' MARK #A /* NEE OLD CODE
0180 /* END-IF /* NEE OLD CODE
0190 /* /* /* NEE OLD CODE
0200 /* IF #A NE 'B' /* NEE OLD CODE
0210 /* REINPUT *1212 /* NEE OLD CODE
0220 /* END-IF /* NEE OLD CODE
0230 RESET #NEE@REINPUT-REPLY /* NEE MODIFIED
0240 MOVE '#A' TO #NEE@REINPUT-FIELD(1) /* NEE MODIFIED
0250 ASSIGN #NEE@REINPUT-FIELD-POS(1) = POS(#A) /* NEE MODIFIED
0260 CALLNAT 'REINPPN1' /* NEE MODIFIED
0270         #NEE@REINPUT-MSG /* NEE MODIFIED
0280         #NEE@REINPUT-MARK /* NEE MODIFIED
0290         #NEE@REINPUT-REPLY /* NEE MODIFIED
0300         #NEE@REINPUT-MSG# /* NEE MODIFIED
0310         #NEE@REINPUT-GROUP /* NEE MODIFIED
0320         #A /* NEE MODIFIED
0330 DECIDE ON FIRST VALUE OF #NEE@REINPUT-REPLY /* NEE MODIFIED
0340 VALUE 0 /* NEE MODIFIED
0350 IGNORE /* NEE MODIFIED
0360 VALUE 9999 /* NEE MODIFIED
0370 REINPUT #NEE@REINPUT-MSG MARK #NEE@REINPUT-MARK /* NEE MODIFIED

```

```
0380 VALUE 9998,9997 /* NEE MODIFIED
0390 REINPUT *#NEE@REINPUT-MSG# MARK #NEE@REINPUT-MARK /* NEE MODIFIED
0400 NONE VALUE /* NEE MODIFIED
0410 COMPRESS 'UNKNOWN RESPONSE FROM VALIDATION ROUTINE' /* NEE MODIFIED
0420 #NEE@REINPUT-REPLY INTO #NEE@REINPUT-MSG /* NEE MODIFIED
0430 REINPUT #NEE@REINPUT-MSG /* NEE MODIFIED
0440 END-DECIDE /* NEE MODIFIED
0450 /*
0460 END
```

The sample object has been modified to include a series of parameter data items which will be used to communicate the response codes data between the modified sample object and the Object Builder generated subprogram.

A DECIDE statement has been inserted to determine what action is required for the returned response codes.

The Object Builder process will generate the following PDA and Subprogram objects to support this modification:

1. Parameter Data Area object.

```
0010 DEFINE DATA PARAMETER
0020 1 #NEE@REINPUT-MSG (A79)
0030 1 #NEE@REINPUT-MARK (I4)
0040 1 #NEE@REINPUT-REPLY (I4)
0050 1 #NEE@REINPUT-MSG# (I4)
0060 1 #NEE@REINPUT-GROUP (1:50)
0070 2 #NEE@REINPUT-FIELD (A65)
0080 2 #NEE@REINPUT-FIELD-POS (I4)
0090 1 #A (A10)
0100 END-DEFINE
```

This contains all the required response data, which will be passed between the modified sample object and the generated subprogram.

# 1

## Natural Engineer Application Restructuring

### 2. Subprogram object.

```
0010 * Subprogram: REINPPN1
0020 *****
0030 * Created by NEE on 2001-09-21 at 11:29:37.0
0040 * Created from REINPPGM from line range 0090-0150
0050 *****
0060 DEFINE DATA
0070 PARAMETER USING REINPPA1
0080 *
0090 LOCAL
0100 01 #NEE@REINPUT-INDEX (I04)
0110 *
0120 END-DEFINE
0130 IF #A EQ ' '
0140 MOVE 'INVALID VALUE DETECTED' TO #NEE@REINPUT-MSG
0150 FOR #NEE@REINPUT-INDEX = 1 TO 50
0160 IF #NEE@REINPUT-FIELD(#NEE@REINPUT-INDEX) EQ '#A'
0170 MOVE #NEE@REINPUT-FIELD-POS(#NEE@REINPUT-INDEX) TO
0180 #NEE@REINPUT-MARK
0190 ESCAPE BOTTOM
0200 END-IF
0210 END-FOR
0220 MOVE 9999 TO #NEE@REINPUT-REPLY
0230 ESCAPE ROUTINE
0240 /* REINPUT 'INVALID VALUE DETECTED' MARK #A
0250 END-IF
0260 /*
0270 IF #A NE 'B'
0280 MOVE 1212 TO #NEE@REINPUT-MSG#
0290 MOVE 9998 TO #NEE@REINPUT-REPLY
0300 ESCAPE ROUTINE
0310 /* REINPUT *1212
0320 END-IF
0330 END
```

This contains the processing logic for each of the conditions that were being applied for the old REINPUT statements.

## Object Builder Presentation Process

---

The Object Builder Presentation process will restructure a Natural application in preparation for implementing alternate user interfaces, such as GUI or the Web.

The Object Builder Presentation process does this by identifying all instances of INPUT statements used by objects within an application. These INPUT statements are encapsulated into Object Builder generated subprograms. Any data items associated with the INPUT statements are encapsulated Object Builder generated Parameter Data Areas, to support the data transfer between the calling object and the new subprograms.

The identification of INPUT statements is also known as the ‘Presentation Layer’, given that the main use for an INPUT statement is to utilize a map (internal or external), to either display data or accept data input.

The removal of this presentation layer from the main processing logic within an application creates applications with simpler objects that can be further enhanced to support future user interface platforms that continue to become available, including Web processing. The application will then consist of separate objects for the procedural and presentation logic.

The Object Builder Presentation process is fully automated and requires no user intervention, other than to create the required Impact search criteria. This is further simplified by the supplied Impact search criteria OBJPRES, which contains all the required criteria for the Object Builder Presentation process to use.

Once the Impact search criteria have been specified the Impact process is executed to create the necessary impact and modification data. Once the Impact results have been reviewed, the Modification process can be applied.

*Note: The supplied Object Builder Presentation process Impact search criteria OBJPRES can be accessed by using ‘PF4’ (GetSa) option on the Impact Criteria Summary screen.*

## Object Builder Presentation Impact Search Criteria

The supplied Impact search criteria OBJPRES contains the following criteria:

Search Keyword	Additional Criteria parameters / Comments
<b>OBJECT BUILDER</b>	Keyword Value = ? Modification Parameters = 'PRESENTATION'. This keyword indicates that Object Builder Presentation processing is to be performed for this impact execution.
<b>INPUT</b>	Identifies all INPUT statements.
<b>DEFINE WINDOW</b>	Identifies all associated WINDOW definitions for INPUT WINDOW statements.
<b>DEFINE SUBROUTINE</b>	Identifies subroutine blocks containing INPUT statements.
<b>PERFORM</b>	Identifies PERFORM statements that execute subroutines containing INPUT statements.
<b>REINPUT</b>	Identifies any REINPUT statements associated with a preceding INPUT statement.  <i>For more information see the section <a href="#">Object Builder Line Range replacing REINPUT statements</a>.</i>
<b>END-SUBROUTINE</b>	Identifies subroutine blocks containing INPUT statements.
<b>DATAITEM</b>	Search Value = *PF-KEY  Identifies any statements referencing PF-KEY processing for an INPUT statement.

These combined Impact search criteria will identify all INPUT statement occurrences within an object and any associated data requirements.

*Note: The same could be achieved using the Object Builder Line Ranges option, but this would be a very long laborious exercise in specifying all the required search criteria for each object.*

## Objects generated by the Object Builder Presentation Process

The same sets of objects are generated for the Object Builder Presentation process as for the Object Builder Line Ranges option. These will be:

### 1. Generate a subprogram object.

A new subprogram object is created on the Modification library with the source code for each identified INPUT statement.

### 2. Generate a Parameter Data Area (PDA) object.

A PDA object is only generated, if the Impact process has identified any data that needs to be passed between the calling object and the new subprogram. This will contain all the required data items associated for each INPUT statement.

### 3. Modify the original object to reference the new subprogram object.

The original object will be modified to comment out the source code for each identified INPUT statement. This is replaced with a CALLNAT statement to call the new subprogram and pass any required parameters.

## Example of Object Builder Presentation

The following example illustrates the Object Builder Presentation process against the sample application HOSPITAL.

An application called HOSPITAL has been created with all the objects from the sample application HOSPITAL extracted and loaded into the Repository. The Application Preferences have been specified with the Modification library set to HOSPITAX. The library HOSPITAX is empty. The example begins from the Impact Version stage.

**Step 1** An Impact Version of 01 is created using the Impact Version option from the Analysis menu.



**Step 2** Specify the Impact search criteria using the supplied criteria data OBJPRES. This is done via the Impact Criteria Summary screen, accessed from the Analysis menu. 'PF4' (GetSa) is used and the OBJPRES criteria data is selected.

The following Figure 1-13 illustrates the selection of criteria data OBJPRES.

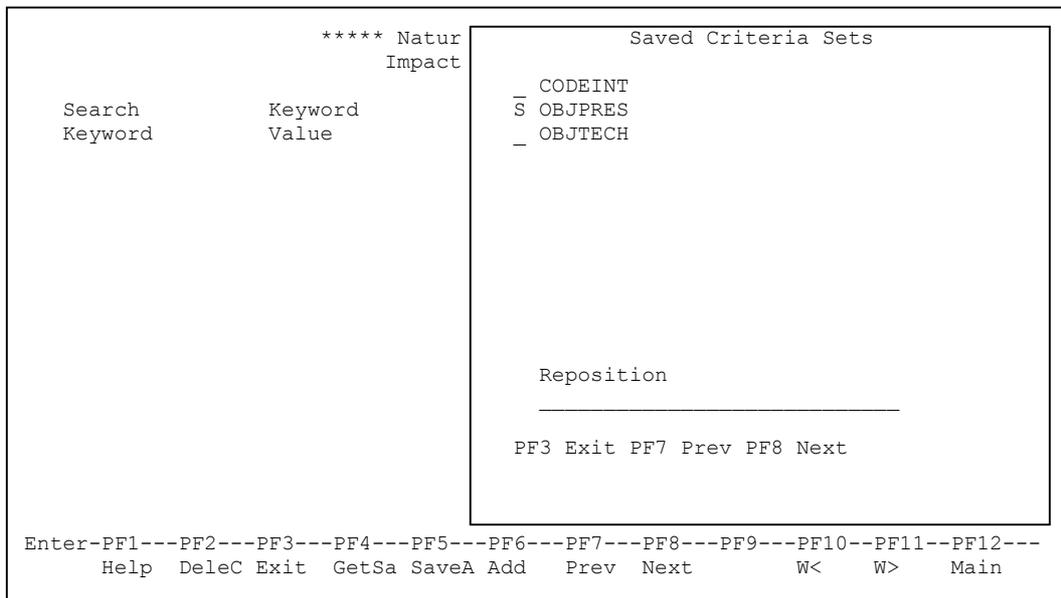


Figure 1-13 Selection of criteria data OBJPRES

Using the 'ENTER' key will read in the OBJPRES details into the Impact Criteria Summary screen.

# 1

## Natural Engineer Application Restructuring

The following Figure 1-14 illustrates the Impact Criteria Summary screen with the OBJPRES criteria loaded.

```
Impact Criteria Summary      Application: HOSPITAL
                               Version: 01
Search      Keyword      Search      Replace      Replace
Keyword     Value        Value        Value        TLM
- OBJECT BUILDER ?
- INPUT
- DEFINE WINDOW
- DEFINE SUBROUTI
- PERFORM
- REINPUT
- END-SUBROUTINE
- DATAITEM                *PF-KEY

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help DeleC Exit  GetSa SaveA Add  Prev Next          W<  W>   Main
Loaded OBJPRES
```

Figure 1-14 Impact Criteria Summary screen with the OBJPRES criteria loaded

**Step 3** With the Impact search criteria specification complete, the Impact Analysis process is invoked by selecting option Impact Execution from the Analysis menu.

**Step 4** When Impact execution has completed, the Impact results are reviewed using the Impact Element Maintenance option, accessed from the Analysis menu. There will be 10 objects impacted from the HOSPITAL application for Object Builder Presentation:

XX001P01; XX002P01; XX021P01; XX022P01; XX023P01; XX024P01; XX025P01; XXCONUPD; XXEXIT and XXVALCC.

The following Figure 1-15 illustrates the Impact Element Categorization screen displaying the Impact results for object XX001P01.

```

- Element Categorization -      Application: HOSPITAL
                                Version: 01
                                Page: 1
Object: XX001P01
Field : PRESENTATION
Attr  :      Impact Type: GK      Ext. Object:

Search Criteria: Searching All Objects for INPUT

Stmt Source Line
0100 INPUT USING MAP "XX001M01"

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12--
      Help      Exit      Prev Next      Ctxt      Main
```

**Figure 1-15 Impact Element Categorization screen displaying the Impact results for object XX001P01**

# 1

## Natural Engineer Application Restructuring

**Step 5** Having reviewed that the Impact results are correct, the Modification criteria are reviewed using the Modification Element Maintenance option, accessed from the Modification menu.

The following Figure 1-16 illustrates the Modification Element Categorization screen displaying the Modification criteria for object XX001P01.

```

- Element Categorization -      Application: HOSPITAL
                                Version: 01
                                Page: 1
Object: XX001P01
Field : PRESENTATION
Attr  :      External Object Name:
Category : A Automatic      Replace Defn: _____ TLM: _____
Type    : GK O.B. Pres LR      Pos: _____
Replace Value: _____
User Comment : _____
TLM Data   : _____
Reason : Data item can be automatically changed
User ID:      Last Update   :
              Execution Date : Not yet modified

Stmt Source Line
0100 INPUT USING MAP "XX001M01"

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help      Exit      Updt SCrit Prev Next +Parm Ctxt      Main
```

**Figure 1-16 Modification Element Categorization screen displaying the Modification criteria for object XX001P01**

This shows that the line 0100 is an INPUT USING MAP statement, which has a Modification category of A (automatic) and the Modification type is set to O.B. Pres LR. We are now ready to apply the modification.

*Note: Each object can be checked by selecting the objects from the Impact Object Categorization screen.*

**Step 6** Modification could be applied by using 'PF5' (Mod) from the Modification Object Categorization screen for each object selected. A quicker option is to use the Execute Modification for All Objects from the Modification menu. This will modify all 10 impacted objects in one single operation. This method invokes the NATRJE Job Submission screen.

The following Figure 1-17 illustrates the NATRJE Job Submission screen for Execute Modification for All Objects.

```

- Job Submission -           Application: HOSPITAL

Job Selection details
-----
Job Selected   : (REMEXE) EXECUTE REMEDY FOR ALL OBJECTS

Job Card details
-----
Job Name      : XGSLXX__
Job Class     : _

Job Control Record details
-----
Control Status :
Last Job Submitted - Job Name :
                  - Opid      :
                  - Step      :
                  - Return Code :

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---
      Help      Exit      Sub  Ref      Rel      Main

```

**Figure 1-17 NATRJE Job Submission screen for Execute Modification for All Objects.**

*Note: For more information on the NATRJE Job Submission screen refer to the Natural Engineer Batch Processing (Mainframes) manual.*

# 1

## Natural Engineer Application Restructuring

**Step 7** Reviewing the Modification results. To do this we will logon to the Natural library HOSPITAX. This library was empty at the start of this example. It now contains 40 objects.

Given that 10 original objects (those identified during impact) will have been copied and modified to CALLNAT the new subprograms, this means that for the HOSPITAL application, the Object Builder Presentation process has generated 30 objects to cater for the restructuring of the presentation layer.

The following Figure 1-18 illustrates the object list for Modification library HOSPITAX.

User XGSLXX		- LIST Objects in a Library -					Library HOSPITAX	
Cmd	Name	Type	S/C	SM	Version	User ID	Date	Time
---	*	*	*	*	*	*	*	*
---	XXCONUA1	Parameter	S	S	2.2.08	XGSLXX	2001-09-21	16:08:39
---	XXCONUA2	Parameter	S	S	2.2.08	XGSLXX	2001-09-21	16:08:39
---	XXCONUN1	Subprogram	S	S	2.2.08	XGSLXX	2001-09-21	16:08:39
---	XXCONUN2	Subprogram	S	S	2.2.08	XGSLXX	2001-09-21	16:08:39
---	XXCONUPD	Subprogram	S	S	3.1.04	XGSLXX	2001-09-21	16:08:39
---	XXEXIT	Subroutine	S	S	3.1.04	XGSLXX	2001-09-21	16:08:40
---	XXEXITN1	Subprogram	S	S	2.2.08	XGSLXX	2001-09-21	16:08:40
---	XX001PA1	Parameter	S	S	2.2.08	XGSLXX	2001-09-21	16:08:40
---	XX001PN1	Subprogram	S	S	2.2.08	XGSLXX	2001-09-21	16:08:40
---	XX001P01	Program	S	S	3.1.04	XGSLXX	2001-09-21	16:08:40
---	XX002PA1	Parameter	S	S	2.2.08	XGSLXX	2001-09-21	16:08:40
---	XX002PN1	Subprogram	S	S	2.2.08	XGSLXX	2001-09-21	16:08:40
---	XX002P01	Program	S	S	3.1.04	XGSLXX	2001-09-21	16:08:40
---	XX021PA1	Parameter	S	S	2.2.08	XGSLXX	2001-09-21	16:08:41
							14 Objects found	
Top of List.								
Command ===>								
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---								
Help Print Exit			-- - + ++		> Canc			

Figure 1-18 Object list for Modification library HOSPITAX

**Step 8** Review the modified object XX001P01. It can be seen that the source code for the INPUT statement at line 0100, has been commented out and replaced by a CALLNAT to the Object Builder generated subprogram XX001PN1. Two parameters for #M-OPTION and #L-MESSAGE are also being passed. The modified code is shown in bold in the following listing.

```
0010 DEFINE DATA GLOBAL USING XX000G00
0020 LOCAL USING XX001L01
0030 *
0040 END-DEFINE
0050 *
0060 SET KEY ALL
0070 *
0080 REPEAT
0090 *
0100 /* INPUT USING MAP "XX001M01" /* NEE OLD CODE
0110 CALLNAT 'XX001PN1' /* NEE MODIFIED
0120         #M-OPTION /* NEE MODIFIED
0130         #L-MESSAGE /* NEE MODIFIED
0140 RESET #L-MESSAGE
0150 *
0160 DECIDE ON FIRST VALUE OF *PF-KEY
0170 *
0180 VALUE "PF12", "PF24"
0190     PERFORM XXEXIT
0200 VALUE 'ENTR'
0210     IF #M-OPTION = "P"
0220         FETCH "XX002P01"
0230     END-IF
0240     IF #M-OPTION = "S"
0250         FETCH "XX003P01"
0260     END-IF
0270     MOVE "INVALID OPTION SELECTED" TO #L-MESSAGE
0280 NONE VALUE
0290     MOVE "INVALID PF KEY PRESSED" TO #L-MESSAGE
0300 END-DECIDE
0310 *
0320 END-REPEAT
0330 END
```

# 1

## Natural Engineer Application Restructuring

**Step 9** Review the Object Builder generated subprogram XX001PN1. This subprogram contains the INPUT statement that was present in object XX001P01.

```
0010 * Subprogram: XX001PN1
0020 *****
0030 * Created by NEE on 2001-09-21 at 16:08:49.1
0040 * Created from XX001P01 from line range 0100-0100
0050 *****
0060 DEFINE DATA
0070 PARAMETER USING XX001PA1
0080 *
0090 END-DEFINE
0100 INPUT USING MAP "XX001M01"
0110 END
```

**Step 10** Review the Object Builder generated PDA XX001PA1. A PDA has been generated because the INPUT statement uses a map 'XX001M01' which contains variables #M-OPTION and #L-MESSAGE. These variables need to be passed between the calling (XX001P01) and called (XX001PN1) objects. The PDA is used by the Object Builder generated subprogram XX001PN1.

```
0010 DEFINE DATA PARAMETER
0020 1 #M-OPTION(A1)
0030 1 #L-MESSAGE(A70)
0040 END-DEFINE
```

**Step 11** To check that the new source codes execute correctly, all objects from the HOSPITAL library, excluding the objects listed below, need to be copied to the Modification library HOSPITAX.

Objects to be excluded from the copy:

XX001P01; XX002P01; XX021P01; XX022P01; XX023P01; XX024P01; XX025P01;  
XXCONUPD; XXEXIT and XXVALCC.

All objects can then be re-stowed and the HOSPITAL application invoked by executing object XX001P01.

*Note: For more information on the Impact Analysis and Modification options refer to the Natural Engineer Application Analysis & Modification for Mainframes manual.*

## Object Builder Technical Process

---

The Object Builder Technical process will restructure a Natural application in preparation for implementing alternate sources of data, for example a different database system or adopting web technologies such as XML.

The Object Builder Technical process does this by identifying all instances of database access statements used by objects within an application, for example FIND, READ, HISTOGRAM, DELETE, STORE and UPDATE. These database access statements are encapsulated into Object Builder generated sub-programs. Any data items associated with the database access statements are encapsulated into Object Builder generated Parameter Data Areas, to support the data transfer between the calling object and the new subprograms.

The identification of database access statements is also known as 'Technical Split', as the process relates to accessing data sources that may reside on alternate platforms, for example client-server type architecture.

The removal of database access statements from the main processing logic within an application creates applications with simpler objects that can be more easily adapted to use alternate data sources and/or platforms such as XML, without having to modify existing procedural code within an application. The application will then consist of separate objects for the procedural and presentation logic.

The Object Builder Technical process is fully automated and requires no user intervention, other than to create the required Impact search criteria. This is further simplified by the supplied Impact search criteria OBJTECH, which contains all the required criteria for the Object Builder Technical process to use.

# 1

## Natural Engineer Application Restructuring

The Object Builder Technical process also utilizes a supplied run-time library NEEDB, which contains standard common structures and routines used by the Object Builder Technical process.

Once the Impact search criteria have been specified the Impact process is executed to create the necessary impact and modification data. Once the Impact results have been reviewed, the Modification process can be applied.

*Note: The supplied Object Builder Technical process Impact search criteria OBJTECH can be accessed by using 'PF4' (GetSa) option on the Impact Criteria Summary screen.*

*Note: The run time library NEEDB will be installed to the Natural FUSER.*

## Object Builder Technical Impact Search Criteria

The supplied Impact search criteria OBJTECH contains the following criteria:

Search Keyword	Additional Criteria parameters / Comments
<b>OBJECT BUILDER</b>	Keyword Value = ? Modification Parameters = 'TECHNICAL'. This keyword indicates that Object Builder Technical processing is to be performed for this impact execution.
<b>DBFILE</b>	Keyword Value = ? Identifies all database file references in the object; this includes READ, GET etc. These are the primary statements to be encapsulated into sub programs.
<b>DBFILE</b>	Keyword Value = ? Search Value = ? Identifies all database field references in the object. These references are required to check for the presence of a high level qualifier, and the removal of labels and statement references.
<b>END-DEFINE</b>	Identifies the end of Data Definition section holding the logical views.
<b>BACKOUT</b>	This is a database related statement that does not have a file reference and is not identified by DBFILE.
<b>END TRANSACTION</b>	This is a database related statement that does not have a file reference and is not identified by DBFILE.
<b>DEFINE SUBROUTINE</b>	Identifies database related statements within subroutine blocks.
<b>END-SUBROUTINE</b>	Identifies database related statements within subroutine blocks.
<b>END-BROWSE</b>	Identifies the end of database processing loops and is replaced with appropriate End-Repeat syntax during Modification.
<b>END-FIND</b>	Identifies the end of database processing loops and is replaced with appropriate END-REPEAT syntax during Modification
<b>END-HISTOGRAM</b>	Identifies the end of database processing loops and is replaced with appropriate END-REPEAT syntax during Modification
<b>END-READ</b>	Identifies the end of database processing loops and is replaced with appropriate END-REPEAT syntax during Modification
<b>LOOP</b>	Identifies the end of database processing loops and is replaced with

<b>Search Keyword</b>	<b>Additional Criteria parameters / Comments</b>
	appropriate END-REPEAT syntax during Modification
<b>IF NO RECORDS</b>	Identifies database related processing statements, which require extra processing during Modification.
<b>END-NOREC</b>	Identifies database related processing statements, which require extra processing during Modification.
<b>DATAITEM</b>	Search Value = *ISN  Database system variable that will need to be replaced with local variables with the same values during Modification.
<b>DATAITEM</b>	Search Value = *COUNTER  Database system variable that will need to be replaced with local variables with the same values during Modification.
<b>DATAITEM</b>	Search Value = *NUMBER  Database system variable that will need to be replaced with local variables with the same values during Modification.
<b>LABEL</b>	Identifies any back references that are used on statements within the database processing loop.
<b>DEFINE DATA LOCAL</b>	Identifies the Data Definition section holding the logical views.
<b>ESCAPE ?</b>	Identifies any ESCAPE statements within a database processing loop that relate to a view variable.
<b>END</b>	Identifies the end of an object.
<b>ENTER</b>	Identifies the use of ENTER within an IF NO RECORDS statement.
<b>AT BREAK OF</b>	Identifies any change of control field statement blocks within a database processing loop, to be replaced with new controlling mechanism to support the encapsulation.
<b>AT END OF DATA</b>	Identifies any specific statement blocks to be executed at the End of the database processing loop, to be replaced with new controlling mechanism to support the encapsulation.
<b>AT START OF DATA</b>	Identifies any specific statement blocks to be executed at the Start of the database processing loop, to be replaced with new controlling mechanism to support the encapsulation.
<b>FOR</b>	Identifies any non-database processing loops, which contain database processing loops within them.
<b>REPEAT</b>	Identifies any non-database processing loops, which contain database processing loops within them.

<b>Search Keyword</b>	<b>Additional Criteria parameters / Comments</b>
<b>READ WORK</b>	Identifies any READ WORK FILE processing loops used to access non-database files, which are not located by DBFILE.
<b>SORT</b>	Identifies any SORT statements relating to database and non-database processing loops.
<b>END-ALL</b>	Identifies any end of SORT processing within a database processing loop.
<b>END-SORT</b>	Identifies any end of SORT processing within a database processing loop.
<b>END-WORK</b>	Identifies any end of WORK FILE processing within a database processing loop.

These combined Impact search criteria will identify all the related database statement occurrences within an object and any associated data requirements.

## Object Builder run-time Library NEEDB

Compile and run-time objects are supplied in a Natural library NEEDB, to support the Stow and execution of the application after Object Builder Technical processing has been applied.

These objects are supplied in source form and can be modified by the user if required, although the individual object names must not be changed.

All the objects found in the Natural library NEEDB must be copied into the Modification library (as specified in the Application Preferences), prior to executing the Object Builder Technical process.

The Natural library NEEDB contains the following objects:

OBJECT	DESCRIPTION
<b>DB@ADBIN</b>	Object Type = Subprogram This sub-program executes the increment of a 64 byte key field by 'X01'.
<b>DB@GCMD</b>	Object Type = Subprogram This sub-program executes the END TRANSACTION and BACKOUT TRANSACTION command for the application, as they are not View/DDM specific.
<b>DB@ACC</b>	Object Type = Copycode Moves data from the database access statement to the structure used in the 'presentation and logic' objects. System variables *NUMBER and *ISN are also moved to local variables.
<b>DB@ACCFN</b>	Object Type = Copycode Resets local variable for NUMBER.
<b>DB@ACCI</b>	Object Type = Copycode Resets local variables for ISN and response code. (Structured Mode version)
<b>DB@ACCIR</b>	Object Type = Copycode Resets local variables for ISN and response code. (Reporting Mode version)
<b>DB@ACCL</b>	Object Type = Copycode End of data check. (Structured Mode version)
<b>DB@ACCLR</b>	Object Type = Copycode End of data check. (Reporting Mode version)
<b>DB@ACCNV</b>	Object Type = Copycode Resets the local variables for ISN, NUMBER and COUNTER.
<b>DB@ACCR</b>	Object Type = Copycode Checks for value of local variables that contain *ISN or *NUMBER, and sets final response code. (Structured Mode version)
<b>DB@ACRRR</b>	Object Type = Copycode Checks for value of local variables that contain *ISN or *NUMBER, and sets final response code. (Reporting Mode version)

<b>OBJECT</b>	<b>DESCRIPTION</b>
<b>DB@DEL</b>	Object Type = Copycode Issues the DELETE command.
<b>DB@GTDL</b>	Object Type = Copycode Part of a DECIDE statement that issues access commands in combination with GET and DELETE commands.
<b>DB@HOLD</b>	Object Type = Copycode Issues the UPDATE command.
<b>DB@STUP</b>	Object Type = Copycode Part of a DECIDE statement that issues access commands in combination with STORE and UPDATE commands.
<b>DB@UPD</b>	Object Type = Copycode Part of a DECIDE statement that issues access commands in combination with DELETE and UPDATE commands. The STORE command is also handled by this routine.
<b>DB@CONST</b>	Object Type = Local Data Area A set of constants used to support each type of database access. For example COMMIT, ROLLBACK, INSERT, STORE, READ etc.
<b>DB@LOCAL</b>	Object Type = Local Data Area Contains local variables for Option and Response Code.
<b>DB@PARMS</b>	Object Type = Parameter Data Area This is used to control the database reads by passing the start ISN, ISN of record read, Adabas response code etc.
<b>DB@PRM01 to DB@PRM40</b>	Object Type = Parameter Data Area Same as DB@PARMS, to be used for any additional database reads.
<b>DB@PRMAC</b>	Object Type = Parameter Data Area Contains the access type required by the calling program. For example READ, FIND, DELETE etc.

## Objects generated by the Object Builder Technical Process

The Object Builder Technical process will generate new subprogram, parameter and local data areas, and common views based on the impacted statements found in the objects within an application. The objects in the supplied Natural library NEEDB, will be utilized to complement the new processing.

### 1. Create specific sub-program for View/DDM and Key

For each database access of FIND, READ and HISTOGRAM in an object, a new sub-program is generated, which contains the database access statement as coded in the original object.

The generated sub-program will also support any GET, DELETE, UPDATE and STORE statements that exist in the original object.

### 2. Generate a structure to replace View

Each view used in the object is replaced with a new structure using the same view name. The new structure is defined in a PDA object.

The structure name has an '@S' appended to the end of the current name. For example PATIENT would be changed to PATIENT@S.

This new PDA allows the transfer of data from the database access statements to the structure used in the presentation and logic objects.

### 3. Generate re-usable consolidated LDA for Views

For each file view used within an application, a new LDA is created using the same database fields.

The Object Builder Technical process will consolidate all views for the same file used within an application, into one single neutral view. This neutral view will be defined in an LDA, with a separate LDA for each view.

These new LDAs are generated using the following naming conventions:

- NVnnnnnn** Used for normal read access to a file. Example: FIND, READ etc. 'nnnnn' is a sequential number assigned to each specific view starting from 000001.
- NVUnnnnn** Used for update access to a file. 'nnnnn' is a sequential number assigned to each specific view starting from 00001.

#### 4. Generate PDA for database access control

For each set of related database access codes within the object, a unique set of control parameters are required to support each database loop process.

A PDA with response-code, ISN, NUMBER, COUNTER and key-value is generated for each.

These control PDAs, are not specific to one object and can be reused across the application. The number of PDAs required depends on the number of database accesses.

#### 5. Modify the original object to reference the new subprogram object

The original object will be modified to comment out the source code for each identified database access statement. This is replaced with a CALLNAT statement to call the new subprogram and pass any required parameters. Additional supporting data definition objects are also added.

# 1

## **Natural Engineer Application Restructuring**

### **Object Builder Technical Process Restrictions**

The Technical Split with Natural Engineer is only possible with structured mode objects or reporting objects that have a DEFINE DATA coded. This is because structures are not possible with RESET type definitions.

Natural 4.1.2 on NT, or Natural 3.1 for mainframe is required, as the STARTING WITH ISN syntax is used within the database access sub-programs.

## Example of Object Builder Technical

The following example illustrates the Object Builder Technical process against the sample application HOSPITAL.

An application called HOSPITAL has been created with all the objects from the sample application HOSPITAL extracted and loaded into the Repository. The Application Preferences have been specified with the Modification library set to HOSPITAX.

The supplied Natural library NEEDB has been copied into library HOSPITAX. These are the only objects present in library HOSPITAX.

The example begins from the Impact Version stage.

**Step 1** An Impact Version of 01 is created using the Impact Version option from the Analysis menu.

The following Figure 1-19 illustrates the Impact Version 01 for application HOSPITAL

Impact Versions				Application: HOSPITAL	
Ver	Date	Time	User ID	Description	Version: 01
_ 01	23/09/2001	15:54	XGSLXX	OBJECT BUILDER TECHNICAL FOR HOSPITAL	

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---  
 Help Exit Add Prev Next Main

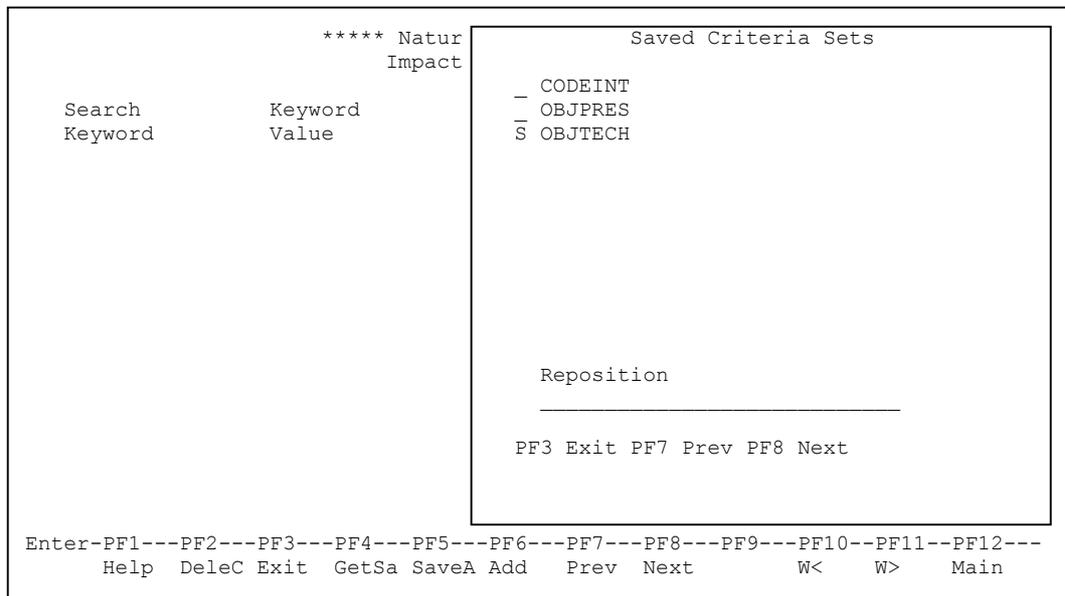
Figure 1-19 Impact Version 01 for application HOSPITAL

# 1

## Natural Engineer Application Restructuring

**Step 2** Specify the Impact search criteria using the supplied criteria data OBJTECH. This is done via the Impact Criteria Summary screen, accessed from the Analysis menu. 'PF4' (GetSa) is used and the OBJTECH criteria data is selected.

The following Figure 1-20 illustrates the selection of criteria data OBJTECH.



**Figure 1-20 Selection of criteria data OBJTECH**

Using the 'ENTER' key will read in the OBJTECH details into the Impact Criteria Summary screen.

The following Figure 1-21 illustrates the Impact Criteria Summary screen with the OBJTECH criteria loaded.

```

                                     Impact Criteria Summary   Application: HOSPITAL
                                                                Version: 01
Search      Keyword      Search      Replace      Replace
Keyword     Value        Value        Value        TLM
- OBJECT BUILDER ?
- DBFILE    ?
- DBFILE    ?           ?
- END-DEFINE
- BACKOUT
- END TRANSACTION
- DEFINE SUBROUTI
- END-SUBROUTINE
- END-BROWSE
- END-FIND
- END-HISTOGRAM
- END-READ
- LOOP
- IF NO RECORDS
- END-NOREC

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help DeleC Exit  GetSa SaveA Add  Prev  Next      W<  W>   Main
Loaded OBJTECH
    
```

Figure 1-21 Impact Criteria Summary screen with the OBJTECH criteria loaded

# 1

## Natural Engineer Application Restructuring

**Step 3** With the Impact search criteria specification complete, the Impact Analysis process is invoked by selecting option Impact Execution from the Analysis menu.

**Step 4** When Impact execution has completed, the Impact results are reviewed using the Impact Element Maintenance option, accessed from the Analysis menu. There will be 5 objects impacted from the HOSPITAL application for Object Builder Technical:

XX021P01; XX022P01; XX023P01; XX025P01 and XXGETID.

The following Figure 1-22 illustrates the Impact Element Categorization screen displaying the Impact results for object XX021P01.

```

                                     - Element Categorization -      Application: HOSPITAL
                                                                    Version: 01
                                                                    Page: 1
Object: XX021P01
Field : PATIENT
Attr  :          Impact Type: GT          Ext. Object:

Search Criteria: Searching All Objects for DBFILE ?

Stmt Source Line
0010 V 1 PATIENT          PATIENT
1240          FIND PATIENT WITH PATIENT-ID = #P-PATIENT-ID
1250          DELETE
1960          STORE PATIENT
2310          FIND PATIENT WITH PATIENT-ID = #P-PATIENT-ID

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help      Exit          Prev Next      Ctxt      Main
```

Figure 1-22 Impact Element Categorization screen displaying the Impact results for object XX021P01

**Step 5** Having reviewed that the Impact results are correct, the Modification criteria are reviewed using the Modification Element Maintenance option, accessed from the Modification menu.

The following Figure 1-23 illustrates the Modification Element Categorization screen displaying the Modification criteria for object XX021P01.

```

- Element Categorization -      Application: HOSPITAL
                                Version: 01
                                Page: 1
Object: XX021P01
Field : PATIENT
Attr  :      External Object Name:
Category : A Automatic      Replace Defn: _____ TLM: _____
Type    : GT O.B. Tech.      Pos: _____
Replace Value: _____
User Comment : _____
TLM Data   : _____
Reason : Data item can be automatically changed
User ID:      Last Update   :
              Execution Date : Not yet modified

Stmt Source Line
0010 V 1 PATIENT                PATIENT
1240     FIND PATIENT WITH PATIENT-ID = #P-PATIENT-ID
1250     DELETE
1960     STORE PATIENT
2310     FIND PATIENT WITH PATIENT-ID = #P-PATIENT-ID

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---
      Help      Exit      Updt  SCrit Prev Next +Parm Ctxt      Main

```

**Figure 1-23 Modification Element Categorization screen displaying the Modification criteria for object XX021P01**

This shows that the line 1240 is a FIND statement, which has a Modification category of A (automatic), and the Modification type is set to O.B. Tech. We are now ready to apply the modification.

*Note: Each object can be checked by selecting the objects from the Impact Object Categorization screen.*

# 1

## Natural Engineer Application Restructuring

**Step 6** Modification for Object Builder Technical process can only be applied by using the Execute Modification for All Objects from the Modification menu. This will modify all 5 impacted objects in one single operation.

The following Figure 1-24 illustrates the NATRJE Job Submission screen for Execute Modification for All Objects.

```

- Job Submission -           Application: HOSPITAL

Job Selection details
-----
Job Selected : (REMEXE) EXECUTE REMEDY FOR ALL OBJECTS

Job Card details
-----
Job Name : XGSLXX__
Job Class : _

Job Control Record details
-----
Control Status :
Last Job Submitted - Job Name :
                    - Opid      :
                    - Step      :
                    - Return Code :

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help      Exit      Sub   Ref              Rel              Main
```

**Figure 1-24 NATRJE Job Submission screen for Execute Modification for All Objects.**

*Note: For more information on the NATRJE Job Submission screen refer to the Natural Engineer Batch Processing (Mainframes) manual.*

**Step 7** Reviewing the Modification results. To do this we will logon to the Natural library HOSPITAX.

Two neutral views have been generated, NV000001 for the read accesses against the PATIENT file and NVU00001 for the update processing against the PATIENT file. New subprograms have been generated to handle the impacted objects, containing all the database accesses statements. New PDA's have been generated to pass data between the modified calling programs and the new subprograms.

The following Figure 1-25 illustrates the object list for Modification library HOSPITAX.

```

User XGSLPN          - LIST Objects in a Library -          Library HOSPITAX

Cmd  Name           Type           S/C   SM Version  User ID   Date       Time
---  *             *             *    * *        *        *         *
___  NVU00001        Local         S     S  2.2.08   XGSLXX   2001-09-23 16:32:11
___  NV000001        Local         S     S  2.2.08   XGSLXX   2001-09-23 16:32:11
___  XXGETID         Subprogram    S     S  3.1.04   XGSLXX   2001-09-23 16:32:11
___  XXGETIN1        Subprogram    S     S  2.2.08   XGSLXX   2001-09-23 16:32:11
___  XX021PA1        Parameter     S     S  2.2.08   XGSLXX   2001-09-23 16:32:12
___  XX021PA2        Parameter     S     S  2.2.08   XGSLXX   2001-09-23 16:32:12
___  XX021PN1        Subprogram    S     S  2.2.08   XGSLXX   2001-09-23 16:32:12
___  XX021PN2        Subprogram    S     S  2.2.08   XGSLXX   2001-09-23 16:32:12
___  XX021P01        Program       S     S  3.1.04   XGSLXX   2001-09-23 16:32:12
___  XX022PN1        Subprogram    S     S  2.2.08   XGSLXX   2001-09-23 16:32:12
___  XX022P01        Program       S     S  3.1.04   XGSLXX   2001-09-23 16:32:12
___  XX023PN1        Subprogram    S     S  2.2.08   XGSLXX   2001-09-23 16:32:12
___  XX023P01        Program       S     S  3.1.04   XGSLXX   2001-09-23 16:32:12
___  XX025PN1        Subprogram    S     S  2.2.08   XGSLXX   2001-09-23 16:32:13
                                         14 Objects found

Top of List.
Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Print Exit          --  -  +  ++  >  Canc
    
```

Figure 1-25 Object list for Modification library HOSPITAX

## 1

## Natural Engineer Application Restructuring

**Step 8** Review the modified object XX022P01. It can be seen that the source code for the FIND statement at line 0220, has been commented out and replaced by a CALLNAT to the Object Builder generated subprogram XX022PN1. The IF NO RECORDS statement at line 0290 has been commented out and replaced by the new parameter driven logic. Two PDAs have been inserted at lines 0170 and 0180 to provide the necessary parameter field definitions. The modified code is shown in bold in the following listing.

```

0010 DEFINE DATA GLOBAL USING XX000G00
0020 LOCAL USING XX021L01
0030 LOCAL
0040 01 #C-GROUP (C/1:10)
0050 01 REDEFINE #C-GROUP
0060 02 #C-PATIENT-ID (C)
0070 02 #C-FIRST-NAME (C)
0080 02 #C-SURNAME (C)
0090 02 #C-DOB (C)
0100 02 #C-RELEASED (C)
0110 02 #C-ADDRESS (C)
0120 02 #C-ARRIVED (C)
0130 02 #C-DUE-FOR-SURGERY (C)
0140 *
0150 01 #P-PATIENT-ID (N7)
0160 *
0170 LOCAL USING DB@PRM01 /* NEE MODIFIED
0180 LOCAL USING DB@CONST /* NEE MODIFIED
0190 END-DEFINE
0200 *
0210 INPUT #P-PATIENT-ID
0220 /* FIND PATIENT WITH PATIENT-ID = #P-PATIENT-ID /* NEE OLD CODE
0230 RESET DB@PRM01 /* NEE MODIFIED
0240 ASSIGN DB@PRM01.#DB@NOREC = TRUE /* NEE MODIFIED
0250 REPEAT /* NEE MODIFIED
0260 CALLNAT 'XX022PN1' #DB@FIND DB@PRM01 PATIENT /* NEE MODIFIED
0270 #P-PATIENT-ID /* NEE MODIFIED
0280 *
0290 /* IF NO RECORDS FOUND /* NEE OLD CODE
0300 IF DB@PRM01.#DB@NUMBER EQ 0 /* NEE MODIFIED
0310 MOVE "INVALID PATIENT ID PASSED TO AMEND PROGRAM" TO #G-MESSAGE
0320 ESCAPE ROUTINE

```

```
0330 /* END-NOREC /* NEE OLD CODE
0340 END-IF /* NEE MODIFIED
0350 *
0360 /* END-FIND /* NEE OLD CODE
0370 IF DB@PRM01.#DB@NUMBER EQ 0 /* NEE MODIFIED
0380 ESCAPE BOTTOM /* NEE MODIFIED
0390 END-IF /* NEE MODIFIED
0400 END-REPEAT /* NEE MODIFIED
0410 *
0420 SET KEY ALL
0430 *
0440 REPEAT
0450 *
0460 INPUT USING MAP "XX022M01"
0470 RESET #G-MESSAGE
0480 *
0490 DECIDE ON FIRST VALUE OF *PF-KEY
0500 *
0510 VALUE "PF12","PF24"
0520     PERFORM XXEXIT
0530 VALUE "ENTR"
0540     IGNORE
0550 NONE VALUE
0560     MOVE "INVALID PF KEY PRESSED" TO #G-MESSAGE
0570 END-DECIDE
0580 *
0590 END-REPEAT
0600 *
0610 END
```

## 1

**Natural Engineer Application Restructuring**

**Step 9** Review the Object Builder generated subprogram XX022PN1. This subprogram contains the FIND statement that was present in object XX022P01 (marked in bold).

```
0010 * Subprogram: XX022PN1
0020 *****
0030 * Created by NEE on 2001-09-23 at 16:32:13.2
0040 * Created from XX022P01 from line range 0200-0200
0050 *****
0060 DEFINE DATA
0070 PARAMETER USING DB@PRMAC
0080 PARAMETER USING DB@PARMS
0090 PARAMETER USING XX021PA1
0100 PARAMETER
0110 01 #P-PATIENT-ID (N7)
0120 LOCAL USING DB@CONST
0130 LOCAL USING DB@LOCAL
0140 LOCAL USING NV000001
0150 LOCAL USING NVU00001
0160 END-DEFINE
0170 DECIDE ON FIRST VALUE OF #DB@ACCESS
0180 VALUE #DB@FIND
0190 INCLUDE DB@ACCI
0200 FIND PATIENT@V1 WITH PATIENT-ID EQ #P-PATIENT-ID
0210 STARTING WITH ISN = #DB@ISNSTART
0220 INCLUDE DB@ACC 'PATIENT@S' 'PATIENT@V1' '*NUMBER'
0230 ESCAPE BOTTOM
0240 END-FIND
0250 INCLUDE DB@ACCR '#DB@ISN'
0260 INCLUDE DB@GTDL 'PATIENT@S' 'PATIENT@V1'
0270 NONE VALUES
0280 IGNORE
0290 END-DECIDE
0300 *
0310 END
```

**Step 10** Two neutral view LDAs will have been generated, one for each view found within the HOSPITAL application. These will be referenced within the Object Builder generated subprograms.

LDA NV000001 contains the definitions for view PATIENT:

```
0010 DEFINE DATA LOCAL
0020 1 PATIENT@V1 VIEW OF PATIENT
0030 2 PATIENT-ID
0040 2 FIRST-NAME
0050 2 SURNAME
0060 2 DOB
0070 2 ADDRESS(1:4)
0080 2 ARRIVED
0090 2 DUE-FOR-SURGERY
0100 2 RELEASED
0110 2 NEXT-ID-FOR-YEAR
0120 2 S2-SURNAME-PATIENT-ID
0130 END-DEFINE
```

LDA NVU00001 contains the definitions used for view PATIENT-UPDATE.

```
0010 DEFINE DATA LOCAL
0020 1 PATIENT@UV1 VIEW OF PATIENT
0030 2 PATIENT-ID
0040 2 FIRST-NAME
0050 2 SURNAME
0060 2 DOB
0070 2 ADDRESS(1:4)
0080 2 ARRIVED
0090 2 DUE-FOR-SURGERY
0100 2 RELEASED
0110 2 NEXT-ID-FOR-YEAR
0120 END-DEFINE
```

# 1

## Natural Engineer Application Restructuring

**Step 11** To check that the new source codes execute correctly, all objects from the HOSPITAL library, excluding the objects listed below, need to be copied to the Modification library HOSPITAX.

Object s to be excluded from the copy:

XX021P01; XX022P01; XX023P01; XX025P01 and XXGETID.

All objects can then be re-stowed and the HOSPITAL application invoked by executing object XX001P01.

*Note: For more information on the Impact Analysis and Modification options refer to the Natural Engineer Application Analysis & Modification for Mainframes manual.*

# INDEX

## O

- Object Builder Line Range Process, 15
  - Example, 25
  - How to specify Line Ranges, 18
  - Objects generated, 16
  - replacing REINPUT statements, 34
- Object Builder Overview, 10
  - Generated Object Names, 13
- Object Builder Presentation Process, 39
  - Example, 41
  - Objects generated, 41
  - OBJPRES, 40
- Object Builder Technical Process, 51
  - Example, 61
  - NEEDB, 55
  - Objects generated, 58
  - OBJTECH, 53
  - Restrictions, 60

