

Natural Business Services

Getting Started with Natural Construct

Version 8.2.2

November 2023

This document applies to Natural Business Services Version 8.2.2 and all subsequent releases.

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

Copyright © 2006-2023 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors.

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

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

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

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third-Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software AG Products / Copyright and Trademark Notices of Software AG Products". These documents are part of the product documentation, located at <http://softwareag.com/licenses> and/or in the root installation directory of the licensed product(s).

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with Software AG.

Document ID: CST-GETSTART-822-20231119

Table of Contents

Preface	v
1 About this Documentation	1
Document Conventions	2
Online Information and Support	2
Data Protection	3
2 Introduction to Natural Construct	5
What is Natural Construct?	6
How Does Natural Construct Work?	8
Generate a Simple Program	13
Creating Specification Documents	20
Utilities Supplied with Natural Construct	22
3 Natural Construct for New Application Development	23
Natural Construct Business Object	24
Add a Business Object	25
Generate a Business Object	28
4 Natural Construct for the Natural Programmer	45
Creating Test Data	46
Testing Subprograms and Help routines	48
Comparing Natural Objects	51
Creating Online Reports	56
Writing Natural Statements	77
5 Troubleshooting and Debugging	83
Specifying Parameters	84
Generating Programs	86
Working in the Editor	88
Viewing Error Messages	88
Using the Natural Debugging Tool	91
6 Getting the Most From Natural Construct	93
Providing Online Help for Your Applications	94
Building New Models	95
Creating JCL	95

Preface

This documentation provides a quick overview of Natural Construct and its many features and capabilities. It is organized under the following headings:

Introduction to Natural Construct

Describes the basic features and functions of the application generation environment. Includes step-by-step instructions to create a simple menu.

Natural Construct for New Application Development

Describes how to develop new applications for use in client/server environments. Contains information about creating business objects that encapsulate business rules and data access needs. Includes step-by-step instructions to create an object maintenance process.

Natural Construct for the Natural Programmer

Describes how you can speed up the development cycle and reduce errors. Intended for Natural programmers who are involved in maintenance programming or want to learn tips to increase productivity. Includes step-by-step instructions to create a multi-panel report supporting many user-requested features. This information will benefit both new and experienced users.

Troubleshooting and Debugging

Provides troubleshooting and debugging information.

Getting the Most From Natural Construct

Describes some of the optional functions of Natural Construct, such as creating help for your applications and building custom models. Also describes the two JCL models and lists several add-on products you can use with Natural Construct.

1

About this Documentation

■ Document Conventions	2
■ Online Information and Support	2
■ Data Protection	3

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Monospace font	Identifies service names and locations in the format <i>folder.subfolder.service</i> , APIs, Java classes, methods, properties.
<i>Italic</i>	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>.

In addition, you can also access the cloud product documentation via <https://www.software-ag.cloud>. Navigate to the desired product and then, depending on your solution, go to “Developer Center”, “User Center” or “Documentation”.

Product Training

You can find helpful product training material on our Learning Portal at <https://knowledge.softwareag.com>.

Tech Community

You can collaborate with Software AG experts on our Tech Community website at <https://tech-community.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 AG 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 AG resources.

Product Support

Support for Software AG 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 Introduction to Natural Construct

■ What is Natural Construct?	6
■ How Does Natural Construct Work?	8
■ Generate a Simple Program	13
■ Creating Specification Documents	20
■ Utilities Supplied with Natural Construct	22

This chapter is intended for programmers who have never used Natural Construct. It introduces new users to the basic features and functions of the generation environment, as well as how Natural Construct works.

To get acquainted with the generation environment, follow the step-by-step instructions to create a simple menu program for an order entry application. If you encounter problems, try one or more of the following suggestions:

- Display the online help as follows:
 - To view the help for a panel, press the help (PF1) key while the cursor is anywhere on the panel except an input field.
 - To view the help for an input field or select a parameter from a pop-up window, press PF1 or type "?" and press Enter when the cursor is in the field.
-  **Note:** If you do not specify a required parameter, or enter an incorrect one, Natural Construct will prompt you for the correct information.
- Check relevant sections of *Natural Construct Generation*.
 - Visit our web site at www.softwareag.com to get timely product information, correspond with customer support representatives, or access related documents.

Although this guide deals mainly with the generation subsystem, Natural Construct contains two other subsystems. For information, see [Help Subsystem](#), and [Administration Subsystem](#).

What is Natural Construct?

Natural Construct is a powerful, easy-to-use, model-based application generator that helps you create and implement high-quality, robust Natural applications. It is built around and uses existing Natural facilities, which means that Natural Construct-generated applications can be deployed on most platforms or distributed across multi-tiered environments.

To Natural's production-oriented development environment and portable execution environment, Natural Construct adds:

- a code generator (at a statement, as well as Natural object level)
- a help maintenance system
- a model maintenance system

These subsystems are described in the following sections.

Generation Subsystem

As a code generator, Natural Construct offers a wide range of reusable program structures, called models. These models allow even inexperienced programmers to generate fully functional applications using high-level, fill-in-the-blank specifications. And Natural Construct uses data defined in Predict, Software AG's data dictionary, to provide the default model specifications, such as local views and field prompts. You can also create your own verification or referential integrity rules in Predict for use with your Natural Construct-generated applications.

Help Subsystem

The help subsystem is designed for non-programmers. It separates the coding of an application from the creation of the online documentation. This separation allows business users or technical writers to write the online help information and programmers to concentrate on programming. In addition, the help subsystem supports multi-lingual help text. For information about this subsystem, see *Natural Construct Help Text*.

Administration Subsystem

The administration and modeling subsystem is used mainly by administrators. It allows them to modify the models supplied in the generation subsystem, create new models, and set security standards. For information about this subsystem, see *Natural Construct Administration and Modeling*.

Benefits of Using Natural Construct

There are many good reasons for using Natural Construct to create Natural applications. For example, you can:

- Reduce development efforts by 50% (as reported by many customers)
- Reduce training time and increase acceptance of new applications, as the generated programs have a consistent interface
- Decrease the development time for complex applications and make fewer mistakes
- Reduce checking and testing requirements; you can concentrate on the customized code
- Reduce maintenance efforts by including business rules within the generated objects
- Easily regenerate applications as requirements change

How Does Natural Construct Work?

Natural Construct is based on the premise that most applications have common functional blocks of code. These blocks of code became object templates, called models, which ensure that the required and expected aspects of each object type are incorporated into the generated object in a standard way.

Rather than re-engineering redundant routines each time you create an object, you can use the Natural Construct models to produce functionality-rich business applications that incorporate state-of-the-art technology.

Natural Construct uses the following information to generate an object:

- The chosen model
- The model specifications supplied by the programmer
- Predict data, as required
- Data coded in the user exits (optional)

For more information about designing or implementing a Natural Construct application, see *Implementation Methodology* or *Design Methodology* in *Natural Construct Generation*.

Choosing a Model

Natural Construct supplies more than 100 models, which are available through the Generation main menu. These models generate various programs, subprograms, help routines, data areas, maps, Natural statements, etc. You can also invoke the statement models from the Natural and User Exit editors.

For client/server environments, Natural Construct provides a series of object maintenance and browse models (Object-Maint and Object-Browse models). In addition, Natural Construct supplies user exit models to assist you in writing user exit code.

Which model you choose depends on what you are doing. For example:

For help:	Try:
Developing client/server applications	Object-Maint and Object-Browse models (for information, see Generate a Business Object).
Remembering Natural syntax	Statement models (for information, see Writing Natural Statements).
Testing Natural objects	Driver or Maint models (for information, see Testing Subprograms and Help routines or Creating Test Data).

For help:	Try:
Writing user exit code	User exit models (for information, see <i>Object-Browse Models</i> in <i>Natural Construct Generation</i>).
Developing conventional applications (character-based interfaces)	Maint model and/or models beginning with Browse or Browse-Select; also the Start, Menu, and Quit models.



Tip: The Browse and Browse-Select models are similar. Both models generate a browse (query) program that allows users to select and process a record, but the Browse-Select model provides more options. A generated browse program performs one pre-determined set of commands on a selected record (such as delete a record or modify a record); a generated browse-select program allows the user to specify which set of commands are performed.

Basically, the models can be grouped into four categories:

- conventional
- client/server
- statement
- user exit

Choose the model that provides the functionality you require. For example, to create the sample menu program later in this documentation, you'll use the Menu model. The following sections describe the most commonly used models in each category.

Conventional Models

These models generate non-distributed programs, subprograms, help routines, data areas, and maps. The models in this category include:

Model	Generates
Startup	A startup program that initializes the application environment and invokes the menu and quit programs. It provides an easy way to set up initial global variables for an application.
Quit	A quit program that releases application resources. It provides a common exit point from anywhere in an application.
Menu	A menu program that controls the flow of work modules in an application.
Browse/Browse-Select	A simple or complex browse program, subprogram, or help routine that queries the database and allows users to browse through and select data for processing.
Maint	An online maintenance program that allows users to select a record from the database and then perform a maintenance function (delete, for example). The program controls all data access and user interface requirements.
Map	An external Natural map.
Batch	A program that runs in batch mode to create a report or update the database.

Model	Generates
Driver	A driver program you can use to test and debug Natural subprograms and help routines.
Shell	A skeleton program you can use as a template when developing Natural programs.

For more information about these models, see the applicable section in *Natural Construct Generation*. For an example of using the Menu model, see [Generate a Simple Program](#).

Client/Server Models

The models in this category generate the components required for a client/server application. They include:

Model	Generates
Object-Maint-Subp	An object subprogram that accesses data, performs updates, and enforces data integrity.
Object-Maint-Dialog	An object dialog program that inputs the object map and invokes the object subprogram.
Object-Browse-Subp	An object browse subprogram and the associated object PDAs, key PDAs, and restricted PDAs.
Object-Browse-Dialog	A character-based dialog that works with an object browse subprogram.
Object-Browse-Dialog-Driver	A driver program or help routine that invokes an object browse dialog.

For more information about the client/server models, see the applicable section in *Natural Construct Generation*. For an example of creating an object maintenance process, see [Generate a Business Object](#).

Statement Models

The statement models generate blocks of code containing one or more Natural statements, which you can use as building blocks when creating user exit code or specialized programs. Using the statement models assists you with the statement syntax and reduces the time required to write recurring code segments and test your code.

Typically, you invoke the statement models from the User Exit and Natural editors using a .g line command and the model name in brackets. For example, enter .g(view) to invoke the View statement model. Enter the .g commands in the same position as other line commands, such as the .i (insert) or .d (delete) command.

A few of the commonly used statement models are:

Model	Generates
Callnat	A Natural CALLNAT statement, which calls a subprogram for execution.
Decide-On	A Natural DECIDE-ON statement, which specifies multiple actions to be performed (depending on the contents of a variable).
Get	A Natural GET statement, which reads a record with a given Adabas ISN (internal sequence number) without initiating a processing loop.
View	The definition of a view, as derived from Predict.

Natural Construct supplies more than 40 statement models. For information about these models, see *Statement Models in Natural Construct Generation*. For an example of using statement models, see [Writing Natural Statements](#).

User Exit Models

These models generate text member modules containing the field layout specifications for an object browse dialog. The models are invoked from the User Exit editor using the User-Exit statement model. The user exit models are:

Model	Generates
Input-Key	Layout specifications for input fields and prompts on the dialog.
Export-Data-Fields	Layout specifications for column headers and fields exported to an ASCII file on a PC.
Report-Data-Fields	Layout specifications for column headers and fields routed to a printer.
Write-Data-Fields	Layout specifications for column headers and fields displayed on a screen.

For more information, see *User Exit Models* and *User-Exit Statement Model* in *Natural Construct Generation*.

Defining the Model Specifications

Each model has at least one specification panel to collect specifications that are common to the code generated by that model. To help you fill in the parameters, most models supply default values for key fields. You need only specify any additional parameters the model requires to generate the appropriate code.

To create a menu program, for example, you must specify the following parameters:

- A menu heading
- The codes a user types to access options listed on the menu
- Descriptions of the options
- The name of the program executed to perform each option

Other parameters allow you to include optional features in your program.



Tip: When creating your specification documents, you can use the fields on the specification panels as a checklist to determine which parameters may be required.

Defining the Predict Data

Predict is a data dictionary system that centrally stores and manages all information about the metadata (data about data) for an organization. Natural Construct can use the metadata to accurately and consistently generate application code. Although a Natural Construct-generated application can be executed without Predict, the code generator relies on this metadata.

The main advantage of using Predict is that you can change the metadata and then regenerate the code to reflect the changes. For example, you can change a relationship between files in Predict, regenerate the code, and the new relationship is automatically incorporated.

Natural Construct uses the Predict metadata that describes:

- File relationships, such as restricted or cascade delete constraints
- File layouts
- Field attributes, such as maximum occurrences, redefinitions, edit masks, validation rules, and headings
- Field descriptions (for online help)



Note: The help subsystem and Natural Construct-generated applications can be implemented in an existing Natural environment without Predict. To do this, you add the standard help-proutine (CDHELPR) to the Natural map for which help is required. For more information, see [Providing Online Help for Your Applications](#).

You can also create Predict metadata by documenting generated Natural objects. This option is available when you generate an object.

For more information about using Predict with Natural Construct, see *Use of Predict in Natural Construct* and *Defining Objects and Relationships in Natural Construct Generation*.

Defining User Exits

When specific customizations are required to fulfill your business needs, you can add additional code in a user exit. User exits are strategic points in the model outline, such as at the start of the program or after input. You choose which user exit you want based on the location of the exit within the code. If you are unsure of the location of a user exit, you can generate and edit the Natural object. The position of each user exit is shown in comments that begin with `** SAG Exit Point` followed by the name of a user exit. For example:

**** SAG Exit Point Change-History**

These pointers indicate where the exit will be located in the generated code.

You enter user exit code in the User Exit editor, which you can access from the last specification panel for a model or when you enter U in the Function field on the Generation main menu. If you forget to choose an exit, you can display a list of available exits by entering "SAMPLE" on the command line in the editor.

You can place any Natural code within a user exit. The user exit code becomes part of the object's source code. If you regenerate the object, changes to the user exit code are always preserved.

Many models supplied with Natural Construct have user exits available. When you're creating an object, select the user exits you require and supply the appropriate code for each user exit. Natural Construct uses both the model specifications and the user exit code to create an object.

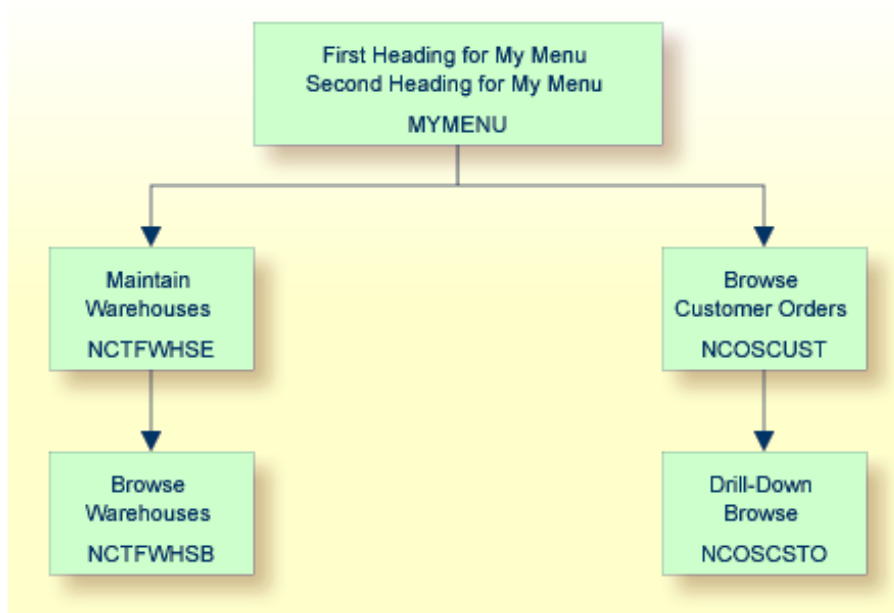
For more information, see *User Exits for the Generation Models* in *Natural Construct Generation*.

Generate a Simple Program

You're now ready to use Natural Construct to generate your first program the main menu for an order entry system. A main menu provides users with a selection of functions available within a system, such as the Generation main menu.

After generating the menu, you'll make a few changes to the specifications and regenerate the program.

The following diagram shows the modules used for the sample menu program:



Before You Begin

The sample menu, as well as other sample programs described in this guide, require access to several modules used in the Natural Construct demo system. The demo system is a fully functional order entry system that was generated using the Natural Construct models (see *Natural Construct Demo System* in *Natural Construct Generation*). Natural Construct supplies the following demo systems:

- SYSCSTDE (Adabas)
- SYSCSTD2 (DB2)
- SYSCSTDV (VSAM)
- SYSCSTD5 (SQL)



Note: Examples in this guide refer to the SYSCSTDE demo system.

Before beginning this procedure, use the Natural SYSMAIN utility to copy the following modules from the demo library into your user library:

Module	Description
MYMAP	Map used by the object maintenance dialog example.
NCGDA	Global data area (GDA) containing global variables. NCGDA is a modified copy of CDGDA, the standard global data area for Natural Construct-generated applications.
NCOSCSTO	Drill-down browse subprogram invoked from NCOSCUST when the user selects an order.
NCOSCUST	Browse program for the Browse Customer Orders option.
NCOSODET	Subprogram used in the Callnat statement model example.

Module	Description
NCTFWHS1	Map used by the browse subprogram.
NCTFWHSB	Browse subprogram invoked from NCTFWHSE when the user enters "B" in the Action field.
NCTFWHSE	Maintenance program for the Maintain Warehouses option.



Note: If you encounter problems trying to access the demo system or files (files beginning with NCST), your environment may not be set up. Contact your system administrator to set up the environment.

Generating the Menu Program

This section describes how to use the Menu model to generate a menu program.



Note: Before beginning the procedure, ensure that the required modules have been copied to the current library. For a list of modules, see [Before You Begin](#).

➤ To generate the menu program

- 1 At the Natural prompt, logon to your Natural Construct library.
- 2 Enter "ncstg". The Generation main menu is displayed:

```

CSGMAIN          N a t u r a l   C o n s t r u c t          CSGMNM0
Oct 22              Generation Main Menu                  1 of 1

      Functions
      -----
      R  Read Specifications
      M  Modify Specifications
      U  User Exit Editor
      G  Generate Source
      T  Test Source
      E  Edit Source
      S  Save Source
      W  Stow Source
      L  List Modules
      C  Clear Edit Buffer
      ?  Help
      .  Return
      -----
Function ..... _  Module ..... _  Panel ..... _
Model ..... _  Type .....
Command ..... _  Library .... MYLIB
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---
      help      quit      opts      lang

```



Tip: Typing "M" on the main menu is the equivalent of Create New.

For more information about the Generation main menu or the Generation subsystem, see *Using the Generation Subsystem*, in *Natural Construct Generation*.

- 3 Specify the following:

M	Function field
MYMENU	Module field
Menu	Model field



Note: If you do not specify a model name, Natural Construct displays a selection window.

- 4 Press Enter. The Standard Parameters panel for the Menu model is displayed. Notice that some of the fields are filled in for you, based on values you specified on the main menu:

```

CUMNMA                               MENU Program                      CU--MA0
Oct 22                               Standard Parameters                1 of 2

Module ..... MYMENU__
System ..... MYLIB_____
Global data area ... CDGDA__
With block ..... _____

Title ..... Menu ..._____
Description ..... This menu ..._____
                        _____
                        _____
                        _____

First heading ..... _____
Second heading ..... _____

Command ..... _
Message numbers .... _
Password ..... _

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11--PF12---
right help  retrn quit                                     right main

```

- 5 Type "First Heading for My Menu" in the First header field.



Note: We aren't using the remaining fields on this panel. If you're curious about what these fields are used for, you can type "?" and press Enter to invoke Natural Construct's online help or see *General Model Specifications* in *Natural Construct Generation*.

- 6 Press Enter. The Additional Parameters panel is displayed:

CUMNMB	MENU Program	CUMNMB0
Oct 22	Additional Parameters	2 of 2
Map layout *		
Code	Functions_____	Program Name
-----	-----	-----
---	-----	----- *
---	-----	----- *
---	-----	----- *
---	-----	----- *
---	-----	----- *
---	-----	----- *
---	-----	----- *
---	-----	----- *
---	-----	----- *
---	-----	----- *
---	-----	----- *
---	-----	----- *
---	-----	----- *
---	-----	----- *

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---
 help retrn quit optns attr left userX main

For this example, leave the Map layout field blank. This allows Natural Construct to create a default (internal) map layout. If you've created an external map layout tailored to your shop standards, you could type the name of that map in this field.

For more information about this panel, press PF1 (help) to invoke the online help or see *Additional Parameters Panel* in *Natural Construct Generation*.

7 Specify the following:

MW	1st Code field
Maintain Warehouses	On line to the right of the 1st Code field
NCTFWHSE	1st Program field
BC	2nd Code field
Browse Customer Orders	On line to the right of the 2nd Code field
NCOSCUST	2nd Program field

These parameters specify the codes the user enters to invoke the corresponding menu functions, the names of the functions, and the names of the programs invoked to perform the function.

- 8 Press Enter. You are returned to the Generation main menu.
- 9 Enter "G" in the Function field. A confirmation message is displayed, indicating that the generation was successful.
- 10 Enter "W" in the Function field. A confirmation message is displayed. Your menu program is now generated and stowed.



Note: For more information about these functions, see *Generate Source Function* and *Stow Generated Source Function in Natural Construct Generation*.

To see the code generated by the Menu model, enter "E" in the Function field.

To see what the menu will look like to users, enter "T" in the Function field. The following menu is displayed:

```
MYMENU          ***** First Heading for My Menu *****
Oct 22                                                  5:18 PM

                Code Functions
                -----
                MW  Maintain Warehouses
                BC  Browse Customer Orders
                ?   Help
                .   Terminate
                -----
Code:  ____

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      help  retrn quit          flip                                main
```

- 11 Press PF2 (retrn). You are returned to the Generation main menu.

You've successfully created a sample menu program. Next, you'll see how easy it is to change a few of the specifications and regenerate the menu program.



Note: If you have any problems creating the menu, see [Troubleshooting and Debugging](#) for possible reasons.

Regenerating the Menu Program

This section describes how to change some of the specifications for the menu and then regenerate the program. The procedure begins at the Generation main menu.

➤ To change the specifications and regenerate the menu program

- 1 Enter "M" in the Function field.



Note: "MYMENU" and "Menu" should be displayed in the Module and Model fields.

The Standard Parameters panel is displayed, showing the specifications you entered when you created the menu.

- 2 Enter "Second Heading for My Menu" in the Second header field. The Additional Parameters panel is displayed.
- 3 Select the word "Functions" and type "Options". On the generated menu, Options will be displayed instead of Functions.
- 4 Enclose the words "Maintain" and "Browse" within angle brackets (for example, <Maintain>Warehouses and <Browse>Customer Orders).

Angle brackets are the default intensify characters for Natural Construct. They indicate that the enclosed text is highlighted on the generated menu screen.



Tip: Notice that the intensify characters are placed in the space between words; they do not take up a physical space.

- 5 Press Enter. You are returned to the Generation main menu.
- 6 Enter "G" in the Function field. Natural Construct regenerates the menu program, incorporating your changes.
- 7 Enter "W" in the Function field. Your menu is now regenerated and stowed.
- 8 Enter "T" in the Function field. The following menu is displayed:

```
MYMENU          ***** First Heading for My Menu *****
Oct 22          - Second Heading for My Menu -                      5:21 PM

                Code Options
                -----
                MC  Maintain Warehouses
                BW  Browse Customer Orders
                ?   Help
                .   Terminate
                -----
Code:  ____

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      help  retrn quit          flip                                main
```

You've successfully changed the specifications and regenerated the menu program. Now that you're familiar with the generation subsystem, see [Natural Construct for New Application Development](#) for information about creating a Natural Construct business object.

Creating Specification Documents

A common task an analyst performs is writing the specifications for a new program. Natural Construct can help create a list of questions you can use to create these documents.

- The first question is whether the program will be used in a client/server environment or as a stand-alone program. This answer restricts the range of models you should use.
- The next question is whether the program will query or maintain data, which further restricts the model selection.

Once you've selected a model, you can use the fields on the model specification panels to determine what parameters may be required. For example, the following figure shows the first specification panel for the Menu model:

CUMNMA	MENU Program	CU - MA0
Oct 22	Standard Parameters	1 of 2
Module MYMENU__		
System MYLIB_____		
Global data area ... CDGDA__		
With block _____		
Title Menu ..._____		
Description This menu ..._____		

First heading _____		
Second heading _____		
Command _		
Message numbers _		
Password _		
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---		
right help retn quit		right main

Use the fields on this panel to determine what information is required to create the menu program. For example:

- What is the name of the menu program? This question is important if you must adhere to standardized naming conventions.
- What will it do?
- What is the first heading on the menu screen?
- What is the second heading?
- Will the menu allow users to access the Natural command line?
- Will the menu be implemented in different languages?
- Will the menu require security restrictions?

Press Enter to display the Additional Parameters panel and use the fields on that panel to determine other questions you need to answer. Once you have the answers to all these questions, you can easily write the specification document for the menu.

Utilities Supplied with Natural Construct

Natural Construct supplies many utilities to aid in the application development process. These utilities are described in different Natural Construct guides, depending on their function. The following table lists each guide and some of the utilities it describes:

Guide	Describes utilities to:
<i>Natural Construct Administration and Modeling</i>	Transfer data across dissimilar platforms Print a list of all code frames Compare two source modules Compare modules in two libraries Convert Natural Construct components to uppercase
<i>Natural Construct Generation</i>	Transfer data across dissimilar platforms Export models to a work file Import models from a work file Insert all INCLUDE statements into generated code (used for debugging) Submit JCL Convert the contents of the source buffer to uppercase Regenerate multiple modules (either online or in batch) Trace and debug Natural modules
<i>Natural Construct Help Text</i>	Print a copy of help text members Unload members to a work file Load members from a work or PC file to the help text file Save the contents of the source buffer to the help text file

3

Natural Construct for New Application Development

■ Natural Construct Business Object	24
■ Add a Business Object	25
■ Generate a Business Object	28

This chapter is intended for programmers who are creating new applications that require the benefits and functionality of object-oriented development for client/server applications. It describes a Natural Construct business object and how to define the business object in Predict. The last section in this chapter contains a step-by-step example of generating an object maintenance process.

Natural Construct Business Object

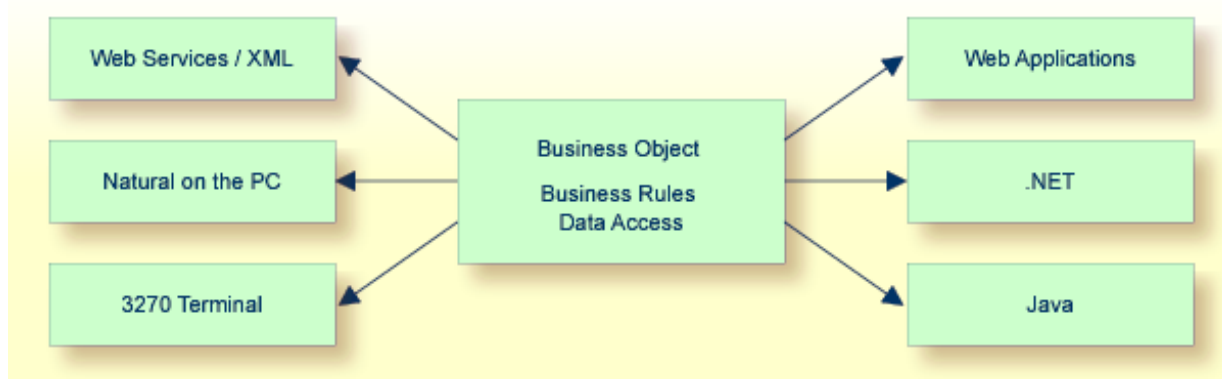
A business object is a group of entities that work together to process a single business function, such as adding an order. When you create a maintenance program using conventional design, the program contains the logic to access and validate data, as well as the user interface code. When you create the program using object-oriented design, the user interface is isolated from the rest of the program code.

The user interface component (dialog) invokes actions (methods) that are implemented by the data access component (data object). Because these components are separate, data is passed back and forth between the dialog and the data object via a communication component. The combination of these components produces a Natural Construct business object.

There are many advantages to separating data access from user interface, such as:

- Data, logic, and business rules are embedded in one central location.
- Data objects are reusable. For example, you can use the same data object with an Accounts Payable dialog and a Product Distribution dialog.
- The data object can be stored on a file server and the dialog can reside on another platform.

The following diagram shows how a Natural Construct business object encapsulates business rules and data access requirements for use with different interfaces:



To create an object transaction, you must create the modules for the following data object and communication components:

Component	Module Required	Purpose
Data object		Builds the data access to a data structure.
	Object subprogram	Contains all the integrity checks, edit checks, and updates to the database.
Communication		Passes messages and data from the dialog to the data object.
	Parameter data area (object PDA)	Contains the database fields that comprise the data object.
	Restricted PDA	Contains variables for update logic that checks for intervening updates in the data object. It is passed with the object PDA.

Natural Construct provides a series of models to generate the modules for an object transaction. The following table lists each module and which model to use:

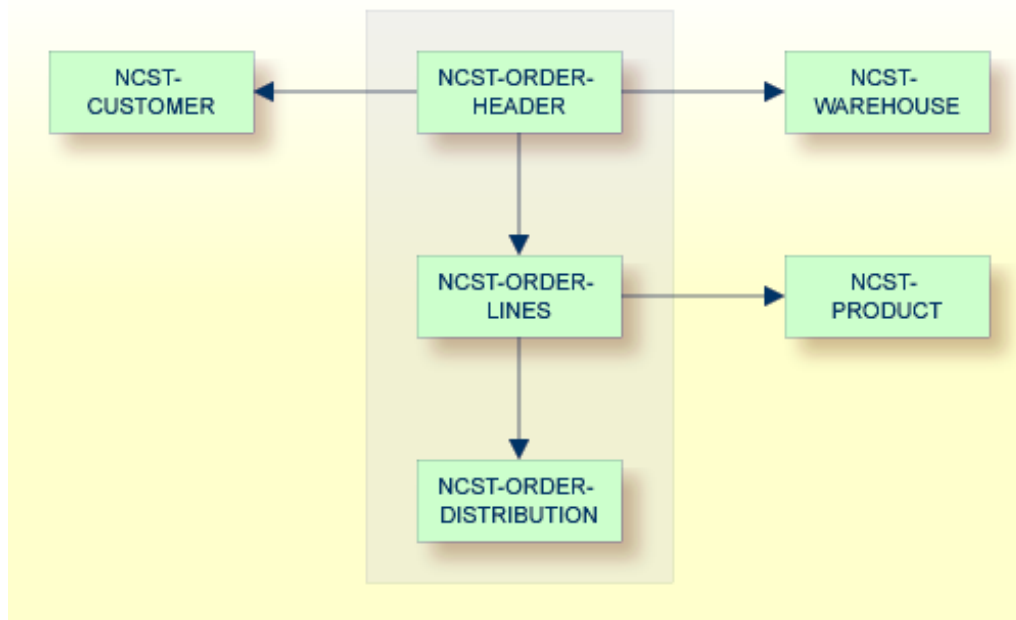
Subcomponent	Model
PDA	Object-Maint-PDA
Restricted PDA	Object-Maint-PDA-R
Object subprogram	Object-Maint-Subp
Map	Map
Object maintenance dialog program	Object-Maint-Dialog
Object maintenance dialog subprogram	Object-Maint-Dialog-Subp



Note: In addition to the Object-Maint models, Natural Construct provides a series of Object-Browse models to generate an object query process. For information, see *Object-Browse Models* in *Natural Construct Generation*.

Add a Business Object

Once you decide to use a business object, the next step is to look at the relationships between the files. Certain relationships must exist before you can use the Object-Maint models effectively. The following diagram shows the files used for a maintenance function in the demo system supplied with Natural Construct:



The files in the shaded area represent the business object, Order. These files have intra-object relationships. The NCST-CUSTOMER, NCST-WAREHOUSE, and NCST-PRODUCT files are not part of the Order object, but they do have inter-object relationships with the NCST-ORDER files.

The metadata stored in Predict provides Natural Construct with the information it needs to create data areas and generate a working program. When generating the components of an object transaction, Natural Construct not only uses the file information in Predict to build views, but it also builds referential integrity and verification rules based on the relationship metadata between the files.

When coding a maintenance program that maintains two or more files, you need to understand the file relationships and decide whether a modification of the business object (for example, the adding, deleting, or modifying of an order) will affect the secondary and/or tertiary files.

When you use the Object-Maint models to generate the program, you only specify the primary file; Natural Construct uses the Predict metadata to identify all other files that belong to the business object and all the maintenance logic required to access the data. To ensure data integrity, the primary file must contain a unique primary key field (either a descriptor or a superdescriptor) and a timestamp field.

For more information about the file views and relationships in Predict, see *Object-Maint Models* in *Natural Construct Generation*.

File Relationships Within Objects

To include files within a business object, you must verify that an intra-object relationship exists between the files and that it has been documented in Predict as type "N" (Natural Construct). Intra-object relationships relate the files within an object, since they define the object within the overall file structure.

Files have intra-object relationships if:

- They have an upper cardinality of (1:1) and a lower cardinality of (0 or 1:N).
- Always behave in a cascading delete fashion.
- Each contain a key that relates the parent file to the child file.

NCST-ORDER-HEADER and NCST-ORDER-LINES meet these requirements since:

- Every order header must have at least one order line.
- When an order is deleted, the order lines should also be deleted.
- NCST-ORDER-LINES has a key called ORDER-LINE-KEY that is made up of ORDER-NUMBER (a key field from NCST-ORDER-HEADER) and LINE-NUMBER (the suffix portion of the key).

Once you determine that intra-object relationships exist, define the relationship in Predict. All files used to define the relationships form a single object. From a Predict point of view, the object has already been created since Predict only describes the application's data needs and technical details required to implement the object.

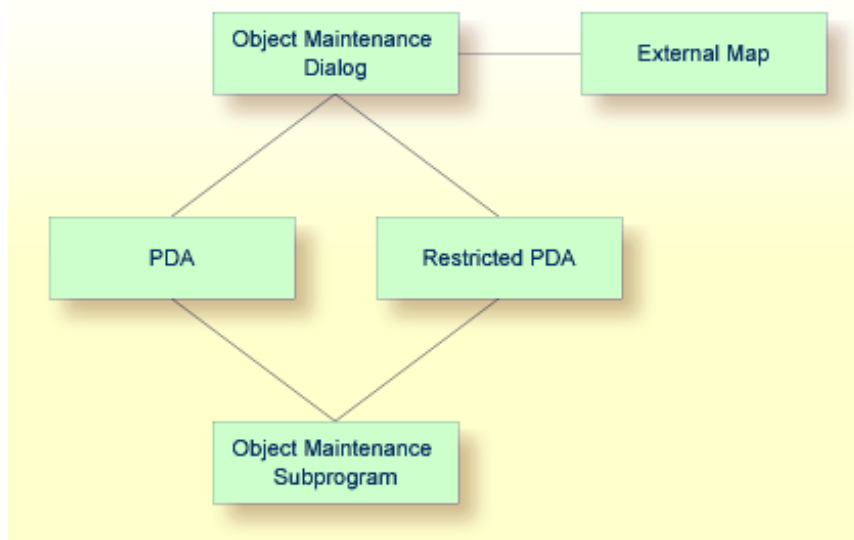
File Relationships Between Objects

The cardinality between files is identified by inter-object relationships. All inter-object relationships with a business object must have a restricted delete and update constraint type. For example, in the above diagram, the NCST-CUSTOMER, NCST-WAREHOUSE, and NCST-PRODUCT files must have a restricted delete and update constraint type. You would not want actions (such as delete) to affect these files.

Application Design

After the files have been documented in Predict, you're ready to create the business object from a Natural Construct point of view. For the object maintenance process example, intra-object relationships exist between the NCST-ORDER-HEADER, NCST-ORDER-LINES, and NCST-ORDER-DISTRIBUTION files and inter-object relationships exist between the Order object and the NCST-CUSTOMER, NCST-WAREHOUSE, and NCST-PRODUCT files.

The following diagram shows the modules required to create an object maintenance process:



Generate a Business Object

In this section, you'll generate a business object to maintain customer orders. It is assumed that the required demo modules and Predict relationships are installed.



Note: Before beginning the procedure, ensure that the required modules have been copied to the current library. For a list of modules, see [Before You Begin](#).

➤ To create an object maintenance process

- 1 Generate the object maintenance subprogram, which automatically generates the object PDA and restricted object PDA.
- 2 Generate the external map.
- 3 Generate the object maintenance dialog.

The components must be generated in this order as the specifications for the object maintenance dialog require the names of an existing map, object maintenance subprogram, object PDA, and restricted object PDA.

In the following sections, you'll generate the object maintenance subprogram and dialog for an order maintenance function. The external map is one of the demo system modules you copied into your library in [Introduction to Natural Construct](#).

- [Generating the Object Maintenance Subprogram](#)
- [Supplied External Map](#)
- [Generating the Object Maintenance Dialog](#)

- Adding Business Rules and Validation Edits
- Executing the Program

Generating the Object Maintenance Subprogram

➤ **To generate the object maintenance subprogram**

- 1 Enter "ncstg" at the Natural prompt. The Generation main menu is displayed.
- 2 Enter "C" in the Function field. This clears the edit buffer.
- 3 Specify the following:

M	Function field
MYORDSUB	Module field

- 4 Press Enter. Because you didn't specify a model name, Natural Construct displays the Select Model window.
- 5 Enter "O" in the Model field. This repositions the list of models to model names beginning with "O".
- 6 Select "Object-Maint-Subp".
- 7 Press Enter. The Standard Parameters panel is displayed:

CUOBMA

Oct 22

OBJECT-MAINT-SUBP Subprogram

Standard Parameters

CUOBMA0

1 of 2

Module

MYORDSUB

System

MYLIB_____

Title

Object_____

Description

This subprogram is used to perform object maintenance_____

for..._____

Message numbers

_

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11--PF12---

right help retrn quit

right main

8 Specify the following:

Order Maintenance Subprog	Title field
... the Order business object	Description field

9 Press Enter. The Additional Parameters panel is displayed:

```

CUOBMB                      OBJECT-MAINT-SUBP Subprogram          CUOBMB0
Oct 22                      Additional Parameters                2 of 2

Predict view ..... *
Primary key ..... *
Hold field ..... *

Object description .....

Object PDA ..... MYORMSA_ *      X      Generate   Source   Object
Restricted PDA ..... MYORMSR_ *    X
Object name ..... MYORMSA_____

Next action prefix ..... _
Log file suffix .....
Trace relationships ..... _

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
main  help  retrn quit                                left  userX main

```

10 Specify the following:

NCST-ORDER-HEADER	Predict view field
ORDER-NUMBER	Primary key field
ORDER-TIMESTAMP	Hold field



Tip: As indicated by the asterisk (*), these fields have selection help available. To display the selection window, either type "?" and press Enter or press PF1 (help) when the cursor is in the field.

11 Type "Order" in the Object description field. Application messages will be displayed as "Order has been added" or "Order has been updated".



Note: Because the object PDA and restricted PDA do not exist, Natural Construct will automatically name and generate these modules. It determines their names based on

the object maintenance subprogram name. For information, see *Object-Maint Models* in *Natural Construct Generation*.

- 12 Type "Order" over "MYORMSA" displayed in the Object name field.
- 13 Press Enter. You are returned to the Generation main menu.
- 14 Enter "G" in the Function field. A confirmation message is displayed, indicating that the subprogram has been generated. For more information, see *Generate Source Function* in *Natural Construct Generation*.
- 15 Enter "W" in the Function field. A confirmation message is displayed, indicating that the subprogram has been stowed. For more information, see *Stow Generated Source Function* in *Natural Construct Generation*.

Next, you'll list the objects in your library to see how many modules have been created.

- 16 Enter "L" in the Function field. The Select Module window is displayed:

CSGLIST	Natural Construct	CSGLIST0
Dec 03	Select Module	1 of 1
Module	Model	Title
-----	-----	-----
MYORDSUB	OBJECT-MAINT-SUBP	Order Maintenance Subprog
MYORMSA	OBJECT-MAINT-PDA	
MYORMSR	OBJECT-MAINT-PDA-R	
SUPRDREW	VB-CLIENT-SERVER-SUPER-MODEL	Super Spec for...
VB-DREW	VB-CLIENT-SERVER-SUPER-MODEL	Super Spec for...
VEHLBCPV	VB-BROWSE-OBJECT	VB Browse Object ...
VEHLBSO	OBJECT-BROWSE-SUBP	Object Browse ...
VEHLBSP	SUBPROGRAM-PROXY	Subprogram proxy for...
VEHLMCDV	VB-MAINT-DIALOG	Object Dialog...
VEHLMCPV	VB-MAINT-OBJECT	Visual Basic Maint Object
VEHLMSA	OBJECT-MAINT-PDA	
Module ...	Model	
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11		
help retrn bkwr frwr		

Notice that three modules were generated: the object maintenance subprogram, the object PDA and the restricted object PDA.

- 17 Press PF2 (retrn). You are returned to the Generation main menu. Now, you'll regenerate the subprogram and specify trace options.
- 18 Enter "M" in the Function field. The Standard Parameters panel is displayed.
- 19 Press Enter. The Additional Parameters panel is displayed:

```

CUOBMB          OBJECT-MAINT-SUBP Subprogram          CUOBMB0
Dec 03          Additional Parameters                2 of 2

Predict view ..... NCST-ORDER-HEADER_____ *
Primary key ..... ORDER-NUMBER_____ *
Hold field ..... ORDER-TIMESTAMP_____ *

Object description ..... Order_____

Object PDA ..... MYORMSA_ *          _      MYLIB      MYLIB
Restricted PDA ..... MYORMSR_ *      _      MYLIB      MYLIB
Object name ..... Order_____

Next action prefix ..... _
Log file suffix ..... _____
Trace relationships ..... _

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
main  help  retn quit                                left  userX main

```

Notice that the parameter data areas are not being generated by default.

- 20 Mark the Trace relationships field.
- 21 Press Enter. You are returned to the Generation main menu.
- 22 Enter "G" in the Function field. Because you marked the Trace relationships field, Natural Construct displays each Predict relationship and indicates whether it has been accepted as an intra- or inter-object relationship.

For example, Natural Construct displays the following message for the NCST-ORDER-HAS-LINES relationship between the NCST-ORDER-HEADER and NCST-ORDER-LINES files:

```

CUOBGPR4                      Natural Construct                      CUOPGPRO
Dec 03                        NCST-ORDER-HEADER Relationship Trace      1 of 1

Relationship ..... NCST-ORDER-HAS-LINES
Type ..... N

Cardinality ..... 1 : CN

File 1                        Minimum      Average      Maximum
File ..... NCST-ORDER-HEADER          1          1.00          1
Field .... ORDER-NUMBER

File 2
File ..... NCST-ORDER-LINES              5.00          30
Field .... ORDER-LINE-KEY
Constraint attribute
Update Type .. N
Delete Type .. C
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF1
      help  retrn
Relationship accepted as part of object MYORMSA

```

This indicates an intra-object relationship between the files and the object is maintained.

23 Press Enter to display the next relationship.

24 Press Enter again. The NCST-PRODUCT-ORDER-LINES relationship is displayed:

```

CUOBGPR4                      Natural Construct                      CUOPGPRO
Dec 03                        NCST-ORDER-LINES Relationship Trace      1 of 1

Relationship ..... NCST-PRODUCT-ORDER-LINES
Type ..... N

Cardinality ..... 1 : CN

File 1                        Minimum      Average      Maximum
File ..... NCST-PRODUCT                1          1.00          1
Field .... PRODUCT-ID

File 2
File ..... NCST-ORDER-LINES              100.00
Field .... ORDER-PRODUCT-ID
Constraint attribute
Update Type .. R
Delete Type .. R
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF1
      help  retrn
Accepted as object Update Type constraint against foreign file

```

This indicates an inter-object relationship between the files and Natural Construct will check to see if the product exists before adding an order line to the order.

- 25 Continue pressing Enter until all relationships have been displayed. When generation is completed, a confirmation message is displayed.
- 26 Enter "W" in the Function field. A confirmation message is displayed. Your object maintenance subprogram is now regenerated and stowed.



Tip: If your module will not stow, ensure that you changed the object name to Order on the Additional Parameters panel.

You've successfully created the object maintenance subprogram. To see the code generated by the Object-Maint-Subp model, enter "E" in the Function field.



Note: If you have any problems creating the subprogram, see [Troubleshooting and Debugging](#) for possible reasons.

For more information about the Object-Maint-Subp model, see *Object-Maint-Subp Model* in *Natural Construct Generation*.

Supplied External Map

The external map is called "MYMAP" and is one of the modules copied into your library from the demo library. This map was created using the Map model. For more information, see *Map Model* in *Natural Construct Generation*.

Natural Construct provides many variables you can use on your maps. For example, you can use the following variables to define scroll regions on a map:

Variable	Purpose
#LIN (P3)	Output field that identifies the array occurrences. The upper bounds value for this variable must match the upper bounds value for the largest array.
#NEXT-ARRAYn (P3)	Modifiable field that allows the user to specify which occurrence appears at the top of the display area.
#ARRAYn (N7)	Starting index for all the fields from the secondary file.

For a complete list of the variables you can use on a maintenance map, see *Variables You Can Use with Object-Maint-Dialog Model Maps* in *Natural Construct Generation*.

Generating the Object Maintenance Dialog

➤ To generate the object maintenance dialog

- 1 Enter "C" in the Function field on the Generation main menu. This clears the specifications from the previous generation.
- 2 Specify the following:

M	Function field
MYORDDIA	Module field
Object-Maint-Dialog	Model field

- 3 Press Enter. The Standard Parameters panel is displayed:

```

CUOMMA                      OBJECT-MAINT-DIALOG Program          CU--MA0
Oct 22                      Standard Parameters                    1 of 4

Module ..... MYORDDIA
System ..... MYLIB_____
Global data area ... CDGDA___
With block ..... _____

Title ..... Object Dialog..._____
Description ..... This program is used to maintain the..._____
_____
_____
_____

First heading ..... _____
Second heading ..... _____

Command ..... _
Message numbers .... _
Password ..... _

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
right help  retrn quit                                     right main

```

- 4 Specify the following:

NCGDA	Global data area field
Order Maintenance Dialog	Title field
... Order business object	Description field
Order Entry System	First header field
Maintain Customer Orders	Second header field

- 5 Press Enter. The Additional Parameters panel is displayed:

```
CUOMMB          OBJECT-MAINT-DIALOG Program          CUOMMB0
Oct 22              Additional Parameters              2 of 4

Object maint subprogram .. _____ *

#ACTION field length ..... 1  Add ..... X  Browse ... _____ *
                                Clear .... X  Display .. X
                                Modify ... X  Next ..... X
                                Purge .... X  Former ... _

Window support ..... _
Push-button support ..... _
Mark cursor field ..... _____

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      help  retrn quit          windw          left  right main
```

- 6 Select "MYORDSUB" from the Object maint subprogram field. This is the name of the object maintenance subprogram you created.



Note: By default, the Add, Clear, Display, Modify, Next, and Purge actions are selected. These are the actions the program will allow.

- 7 Type "2" over "1" displayed in the #ACTION field length field. You change the length because the action field on MYMAP is two characters long.
- 8 Mark the Push-button support field. This allows users to select actions by cursor on the generated screen.
- 9 Press Enter. The Scroll Region Parameters panel is displayed:

```

CUOMMC                                OBJECT-MAINT-DIALOG Program          CUOMMC0
Oct 22                                Scroll Region Parameters              3 of 4

Horizontal panels ..... 1

>> 1 Input using map ..... *

Scrollable Regions          1          2          3          4
Total occurrences .....      _____
Screen occurrences .....      _____
Starting from ..... #ARRAY1 #ARRAY2 #ARRAY3 #ARRAY4
Scroll with panel .....      -          -          -          -

Top left ..... Line .....      _____
                          Column .....      _____
Bottom right .. Line .....      _____
                          Column .....      _____

Depth occurrences .....      _____

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      help  retrn quit          deflt          bkwrđ frwrđ          left  right main

```

- 10 Select "MYMAP" from the Input using map field.
- 11 Press PF5 (deflt). Natural Construct reads the map and retrieves information about the arrays and scroll regions:

```

CUOMMC                                OBJECT-MAINT-DIALOG Program          CUOMMC0
Oct 22                                Scroll Region Parameters              3 of 4

Horizontal panels ..... 1

>> 1 Input using map ..... MYMAP *

Scrollable Regions          1          2          3          4
Total occurrences ..... 30____ 10____ 20____ ____
Screen occurrences ..... 1____ 4____ 2____ ____
Starting from ..... #ARRAY1 #ARRAY2 #ARRAY3 #ARRAY4
Scroll with panel ..... -      -      -      -

Top left ..... Line ..... 10_   10_   16_   ____
                        Column ..... 2_   45_   2_   ____
Bottom right .. Line ..... 15_   15_   18_   ____
                        Column ..... 70_   70_   64_   ____

Depth occurrences ..... ____

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      help  retrn quit      deflt      bkwrđ frwrđ      left  right main
Scroll parameters read from MYMAP map successfully

```

- 12 Press Enter. The Related File Parameters panel is displayed. As Natural Construct does not require access to other files, you don't need to specify information on this panel.
- 13 Press Enter again. You are returned to the Generation main menu.
- 14 Enter "G" in the Function field.
- 15 Enter "W" in the Function field. Your maintenance dialog is now generated and stowed.

To see the code generated by the Object-Maint-Dialog model, enter "E" in the Function field.
To see what the panel will look like to users, enter "T" in the Function field:

```

Add      Clear      Display      Modify      Next      Purge

MYORDDIA          ***** Order Entry System *****          MYMAP
Oct 25,01          - Maintain Customer Orders -          9:00 PM
Action.....: ____
Order Number.....: _____ Invoice Number.....: _____
Customer Number..: _____
Warehouse ID.....: ____
Order Date.....: _____ Order Amount:
1_ ----- Product Information ----- 1_ Distribution Information
1 *Product....: _____ /\          Account      Amount
  Quantity...: _____          1 _____ /\
  Cost/Unit..: _____          2 _____
  Total.....: _____          3 _____
  Description: _____          4 _____ \
1_ Delivery Instructions (Scroll right for full screen)
1 _____ /\
2 _____ \

```

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
 confm help retrn quit flip pref bkwrд frwrд main

- 16 Press PF2 (retrn). You are returned to the Generation main menu.

You've successfully created the object maintenance dialog, but there are no business rules or validation edits. You can add this logic in the UPDATE-EDITS user exit for the object maintenance subprogram (the MYORDSUB module). The following section describes how to do this.



Note: If you have any problems creating the dialog, see [Troubleshooting and Debugging](#) for possible reasons.

For more information, see *Object-Maint-Dialog Model* in *Natural Construct Generation*.

Adding Business Rules and Validation Edits

» To add business rules and validation edits

- 1 Enter "C" in the Function field on the Generation main menu.
- 2 Specify the following:

R	Function field
MYORDSUB	Module field

- 3 Press Enter. Natural Construct reads the object maintenance subprogram into the edit buffer.
- 4 Enter "U" in the Function field. Because you marked the Trace relationships field when you created MYORDSUB, the relationship windows are displayed. Continue pressing Enter until all relationships have been shown and the User Exits panel is displayed:

CSGSAMPL	NATURAL CONSTRUCT				CSGSM0
Oct 22	User Exits				1 of 1
	User Exit	Exists	Sample	Required Conditional	

_	CHANGE-HISTORY		Subprogram		
_	PARAMETER-DATA		Example		
_	EXTENDED-RI-VIEWS				
_	LOCAL-DATA		Example		
_	START-OF-PROGRAM		Example		
_	SELECT-STATEMENT		Subprogram		X
_	USER-DEFINED-FUNCTIONS		Example		
_	BEFORE-ET-PROCESSING		Example		
_	AFTER-ET-PROCESSING		Example		
_	END-OF-PROGRAM		Example		
_	AFTER-GET		Example		
_	AFTER-INIT		Example		
_	UPDATE-EDITS		Subprogram		
_	DELETE-EDITS		Subprogram		
_	AFTER-GET-EDITS		Subprogram		
_	EXTENDED-RI-CHECKS		Subprogram		
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---					
frwr help retn quit bkwr frwr					

- 5 Mark the UPDATE-EDITS user exit.
- 6 Press Enter twice. The UPDATE-EDITS Build Subroutine window is displayed:

CUOBGS	Natural Construct	CUOBGS0
Oct 22	UPDATE-EDITS Build Subroutine	1 of 1
File name		

— NCST-ORDER-HEADER		
— NCST-ORDER-LINES		
— NCST-ORDER-DISTRIBUTION		
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---		
help retn		

Natural Construct provides validation subroutines that execute edit checks at different points during the processing of an entity.

- 7 Mark NCST-ORDER-HEADER.
- 8 Press Enter. The default code is displayed in the User Exit editor:

```

0010 DEFINE EXIT UPDATE-EDITS
0020 *
0030 * Specify subroutines of the form V0-entity-name (or V1- or V2-) used
0040 * to validate or manipulate data before an UPDATE/STORE operation which
0050 * move data from the object to the entity-name record. Data manipulation
0060 * must be made against the record. If any reference to the object is
0070 * required because the relevant fields have different names on the
0080 * object and the file, appropriate indices (#L2),(#L2,#L3),(#L2,#L3,#L4)
0090 * should be used. The V2- routine is performed for an entity after all
0100 * of its children's processing while the V0-/V1- are invoked before
0110 * its children's, and before/after its PREDICT rules respectively.
0120 *
0130 *****
0140 DEFINE SUBROUTINE V0-NCST-ORDER-HEADER
0150 *****
0160 *
0170 * Insert code which is necessary to validate/manipulate the data from
0180 * NCST-ORDER-HEADER record before an update/store operation,
0190 * before its children's processing and before its PREDICT rules if any.
0200 *
0210 DECIDE FOR FIRST CONDITION
0220   WHEN NCST-ORDER-HEADER.ORDER-NUMBER EQ 0
0230     ASSIGN MSG-INFO.##MSG = ' :1: is required'
0240     ASSIGN MSG-INFO.##MSG-DATA(1) = 'Order Number'
0250     ASSIGN MSG-INFO.##ERROR-FIELD = 'ORDER-NUMBER'
0260   WHEN NCST-ORDER-HEADER.ORDER-AMOUNT EQ 0
0270     ASSIGN MSG-INFO.##MSG = ' :1: is required'
0280     ASSIGN MSG-INFO.##MSG-DATA(1) = 'Order Amount'
0290     ASSIGN MSG-INFO.##ERROR-FIELD = 'ORDER-AMOUNT'
0300   WHEN NCST-ORDER-HEADER.ORDER-DATE EQ 0
0310     ASSIGN MSG-INFO.##MSG = ' :1: is required'
0320     ASSIGN MSG-INFO.##MSG-DATA(1) = 'Order Date'
0330     ASSIGN MSG-INFO.##ERROR-FIELD = 'ORDER-DATE'
0340   WHEN NCST-ORDER-HEADER.ORDER-CUSTOMER-NUMBER EQ 0
0350     ASSIGN MSG-INFO.##MSG = ' :1: is required'
0360     ASSIGN MSG-INFO.##MSG-DATA(1) = 'Customer Number'
0370     ASSIGN MSG-INFO.##ERROR-FIELD = 'ORDER-CUSTOMER-NUMBER'
0380   WHEN NCST-ORDER-HEADER.ORDER-WAREHOUSE-ID EQ ' '
0390     ASSIGN MSG-INFO.##MSG = ' :1: is required'
0400     ASSIGN MSG-INFO.##MSG-DATA(1) = 'Warehouse ID'
0410     ASSIGN MSG-INFO.##ERROR-FIELD = 'ORDER-WAREHOUSE-ID'
0420   WHEN NCST-ORDER-HEADER.INVOICE-NUMBER EQ 0
0430     ASSIGN MSG-INFO.##MSG = ' :1: is required'
0440     ASSIGN MSG-INFO.##MSG-DATA(1) = 'Invoice Number'
0450     ASSIGN MSG-INFO.##ERROR-FIELD = 'INVOICE-NUMBER'

```

```
0460 WHEN NCST-ORDER-HEADER.DELIVERY-INSTRUCTIONS(*) EQ ' '
0470     ASSIGN MSG-INFO.##MSG = ' :1: is required'
0480     ASSIGN MSG-INFO.##MSG-DATA(1) = 'Delivery Instructions'
0490     ASSIGN MSG-INFO.##ERROR-FIELD = 'DELIVERY-INSTRUCTIONS'
0500     ASSIGN MSG-INFO.##ERROR-FIELD-INDEX1 = 1
0510 WHEN ANY
0520     ESCAPE ROUTINE
0530 WHEN NONE
0540     IGNORE
0550 END-DECIDE
0560 END-SUBROUTINE /* V0-NCST-ORDER-HEADER
0570 *
0580 *****
0590 DEFINE SUBROUTINE V1-NCST-ORDER-HEADER
0600 *****
0610 *
0620 * Insert code which is necessary to validate/manipulate the data from
0630 * NCST-ORDER-HEADER record before an update/store operation,
0640 * before its children's processing and after its PREDICT rules if any.
0650 *
0660     ESCAPE ROUTINE
0670 END-SUBROUTINE /* V1-NCST-ORDER-HEADER
0680 *
0690 *****
0700 DEFINE SUBROUTINE V2-NCST-ORDER-HEADER
0710 *****
0720 *
0730 * Insert code which is necessary to validate/manipulate the data from
0740 * NCST-ORDER-HEADER record before an update/store operation
0750 * after all of its children have been processed.
0760 *
0770     ESCAPE ROUTINE
0780 END-SUBROUTINE /* V2-NCST-ORDER-HEADER
0790 END-EXIT
```

For this example, you need only ensure that the order number and amount are not zero.

- 9 In the DEFINE SUBROUTINE V0-NCST-ORDER-HEADER section, delete the conditions for the order date, order customer number, order warehouse ID, invoice number, and delivery instructions (lines 300 to 500 of the default code).
- 10 Type "." on the command line in the User Exit editor.
- 11 Press Enter. You are returned to the Generation main menu.
- 12 Enter "G" in the Function field.
- 13 Enter "W" in the Function field. Your object maintenance subprogram is now generated and stowed.

You've successfully added business rules and validation edits to your object maintenance subprogram. The following section describes how to execute the Maintain Customer Orders function you've created.

Executing the Program

Up to this point, you've executed programs using the Test Source function on the Generation main menu. You can also execute Natural Construct-generated programs as you do any Natural program from the command line.

➤ To execute the Maintain Customer Orders function

- 1 Type "MYORDDIA" on the Natural command line.
- 2 Press Enter. The Maintain Customer Orders panel is displayed.
- 3 Move the cursor to Next.
- 4 Press Enter. The order information for the first record is displayed:

```

Add      Clear      Display  Modify  Next      Purge

MYORDDIA          ***** Order Entry System *****          MYMAP
Oct 25           - Maintain Customer Orders -          9:25 PM
Action.....: ____
Order Number.....: 122____ Invoice Number.....: 999____
Customer Number..: 22266
Warehouse ID.....: 544
Order Date.....: 99/08/12 Order Amount: 10880.00
1_ ----- Product Information -----      1_ Distribution Information
1 *Product.....: 187361          /\          Account      Amount
  Quantity...: 444____          1 _____ /\
  Cost/Unit..: 20.00            2 _____
  Total.....: 8880.00          3 _____
  Description: Fish Sticks      \/          4 _____ \/
1_ Delivery Instructions (Scroll right for full screen)
1                                     /\
2                                     \/

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
confm help retrn quit          flip pref bkwrđ frwrđ          main

```

You've successfully executed the Maintain Customer Orders function from the Natural command line. To test the edits, try adding an order with an order number or amount of zero and see if an error message is displayed.



Note: If you have any problems executing the program, see [Troubleshooting and Debugging](#) for possible reasons.

4

Natural Construct for the Natural Programmer

■ Creating Test Data	46
■ Testing Subprograms and Helprouines	48
■ Comparing Natural Objects	51
■ Creating Online Reports	56
■ Writing Natural Statements	77

This chapter contains information about using Natural Construct in the Natural development environment. It is intended for Natural programmers who are not familiar with the product and want to know how it can help them develop Natural applications. This chapter shows how to take advantage of Natural Construct as a development aid without creating entire applications. The information is especially useful for maintenance programmers.

Creating Test Data

When developing Natural programs, you often require data to test your programs. Did you know you can use the Maint model to generate a maintenance program for a file associated with your program? This allows you to quickly add data to test new Natural programs. For example, you can generate a maintenance program for the NCST-CUSTOMER file to supply data to test a new customer browse program.

For a more detailed description of the Maint model, see *Maint Model* in *Natural Construct Generation*.

➤ To generate a maintenance program

- 1 Enter "ncstg" at the Natural Next prompt. The Generation main menu is displayed.
- 2 Specify the following:

M	Function field
MYMAINT	Module field
Maint	Model field



Note: If you do not specify a model name, Natural Construct displays a selection window.

- 3 Press Enter. The Standard Parameters panel is displayed:

```

CUFMMA                                MAINT Program                                CU--MA0
Oct 23                                Standard Parameters                                1 of 3

Module ..... MYMAINT_
System ..... MYLIB_____
Global data area ... CDGDA__
With block ..... _____

Title ..... Maintain..._____
Description ..... This program is used to maintain the...._____
                                     _____
                                     _____
                                     _____

First heading ..... _____
Second heading ..... _____

Command ..... _
Message numbers .... _
Password ..... _

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
right help  retrn quit                                right main

```

- 4 Enter "Test Maintenance Program" in the First header field. The Additional Parameters panel is displayed:

```

CUFMMB                                MAINT Program                                CUFMMB0
Oct 24                                Additional Parameters                                2 of 3

Predict view ..... *
  Natural (DDM) .....
Primary key ..... *
Record description .....
Log file name ..... *
  Natural (DDM) .....
Input using map ..... *
Minimum key value .....
Maximum key value .....
Single prompt ..... _
Multiple prompts ..... _
Mark cursor field .....

#ACTION field length ..... 1  Add ..... X  Browse ... _____ *
                                Clear .... X  Display .. X
                                Modify ... X  Next ..... X
                                Purge .... X  Recall ... _  Former ... _

Push-button support ..... _
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
right help  retrn quit                                left  right main

```

- 5 Select the following:
 - "NCST-CUSTOMER" from the Predict view field
 - "CUSTOMER-NUMBER" from the Primary key field
- 6 Press Enter. The Secondary File Parameters panel is displayed.
- 7 Press Enter again. You are returned to the Generation main menu.
- 8 Enter "G" in the Function field. When generation is completed, a confirmation message is displayed.

Enter "W" in the Function field. Another confirmation message is displayed. You can now use the maintenance program to enter test data for the browse program.



Note: For more information, see *Generate Source Function* and *Stow Generated Source Function* in *Natural Construct Generation*.

To see the code generated by the Maint model, enter "E" in the Function field.

Testing Subprograms and Help routines

To test Natural subprograms and help routines, you require a program to call the modules. Use the Driver model to create a driver program to test and debug subprograms and help routines.

The Driver model determines the parameters in a subprogram, defines the parameters in a simple driver program, and generates an input screen. This allows you to easily test the module without the complex logic of programs that typically call the subprogram. The Driver model also supplies headings and PF-keys in the language of your choice.



Tip: Using a driver program can eliminate the complexity of debugging a subprogram that runs in a GUI, client/server environment.

In the following example, you'll create a driver program to test the NCOSODET browse subprogram, which displays details about order lines.



Note: Before beginning the procedure, ensure that the required modules have been copied to the current library. For a list of modules, see [Before You Begin](#).

➤ To generate a driver program

- 1 Enter "ncstg" at the Natural Next prompt. The Generation main menu is displayed.
- 2 Specify the following:

M	Function field
MYDRIVER	Module field
Driver	Model field

- 3 Press Enter. The Driver Program window is displayed:

CUDRMA	DRIVER Program	CUDRMA0						
Dec 04	Standard Parameters	1 of 1						
Module MYDRIVER System MYLIB_____								
Title Driver Program_____ Description Driver program for ..._____ _____ _____ _____								
<table border="0" style="width: 100%;"> <tr> <td style="width: 40%;"></td> <td style="width: 20%; text-align: center;">Source</td> <td style="width: 40%; text-align: center;">Object</td> </tr> <tr> <td>Called Module _____ *</td> <td></td> <td></td> </tr> </table>				Source	Object	Called Module _____ *		
	Source	Object						
Called Module _____ *								
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12--- main help retrn quit userX main								

- 4 Select "NCOSODET" from the Called Module field.
- 5 Press Enter. The names of the library containing the source and object code for the NCOSODET subprogram are displayed.
- 6 Press Enter again. You are returned to the Generation main menu.
- 7 Enter "G" in the Function field. A confirmation window is displayed, indicating that generation was successful.
- 8 Enter "W" in the Function field. The driver program is stowed.



Note: You cannot regenerate a driver program.

- 9 Enter "T" in the Function field. The Driver Program window is displayed:

```
MYDRIVER                      Natural Construct
Dec 04                        Driver Program                      1 of 1

Subprogram ..... NCOSODET

Input Parameters ... #PDA-KEY ____0 SELECTED-FUNCTION ____ ##COMMAN
____ ##MAIN ____
##QUIT ____ ##MSG
##MSG-NR ____0 ##MSG-DATA ____
____ ##RETURN-
__ ##ERROR-FIELD ____ ##ERROR-FIELD-INDEX1 ____0
##ERROR-FIELD-INDEX2 ____0 ##ERROR-FIELD-INDEX3 ____0 ##USER ____
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF1
      help      quit      bkwrđ frwrđ      right left
```

10 Enter "111" in the #PDA-KEY field. The NCOSODET subprogram is processed for order number 111:

```
NCOSODET ***** ORDER SUBSYSTEM *****
Dec 4,01 - ORDER DETAIL - 1 more >

Ln Prod Id Quantity Unit Cost Total Cost
--
1 199210 5 100.00 500.00
2 111111 2 555.00 1110.00
3 745977 1 7.00 7.00
4 777208 20 55.00 1100.00
      *** End of Data ***

Line Order Number: 000111 Line Number: __
Direct command...: ____
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8
      help retrn quit flip bkwrđ frw
```

 **Tip:** When you enter "111" in the #PDA-KEY field, ensure that you remove the "0".

For a description of the Driver model, see *Driver Model* in *Natural Construct Generation*.

Comparing Natural Objects

Natural Construct supplies various utilities to help you develop Natural applications. This section describes two of these utilities:

- CSGCMPL utility, which you can use to compare a range of Natural objects in one library to the same objects in another library
- CSGCMPS utility, which you can use to compare the code in two programs



Note: For information about other utilities, see [Utilities Supplied with Natural Construct](#).

Comparing Objects in Different Libraries

➤ To compare the Natural objects in different libraries

- 1 Enter "CSGCMPL" at the Next prompt. The Source Range Compare Facility is displayed:

```
Source Range Compare Facility
                                Library  Database  File  Dominant
                                =====  =====  =====  =====
Old library.....             _____  ____196   ____4   X
New Library.....             _____  ____196   ____4   _
Program range.....            _____ thru _____
Summary only.....            _
Only report if different..    _
Ignore comment lines.....    _
Ignore trailing comments..    _
Ignore leading spaces.....    _
Only compare object types    _____ (ACGHLMNPST)
```

- 2 Specify the following:

MYLIB	Library field for the Old library
SYSCSTDE	Library field for the New library
A*	1st portion of the Program range field
Z*	2nd portion of the Program range field

The dominant library (MYLIB) identifies which objects are compared. Only objects that exist in the MYLIB and SYSCSTDE libraries are compared. Objects that only exist in the SYSCSTDE library are not included in the results.



Notes:

1. The "Old" and "New" designation is for identification purposes only; it does not indicate older or newer versions of the libraries.
 2. If desired, you can limit the comparison by specifying one or more options listed in this window. For more information, see *CSGCMPL Utility* in *Natural Construct Administration and Modeling*.
- 3 Press Enter. The Library Compare Facility window is displayed:

```
Library Compare Facility                      Page 1
Dec 04,01                                     08:03 PM

Program                                     Matching
Name      lines      lines      lines      Comments
-----
Input parameters...

                                Library Database File Dominant
                                =====
Old library..... MYLIB          196          4      X
New Library..... SYSCSTDE       196          4
Program range..... A*           thru Z*
Summary only.....
Only report if different..
Ignore comment lines.....
Ignore trailing comments..
Ignore leading spaces.....
Compare all types
```

- 4 Press Enter again. The results of the comparison are displayed:

```

Library: MYLIB      **** N A T U R A L   C O N S T R U C T ****      CSGCMPS1
Object.: MYBROWSE      - SOURCE COMPARE FACILITY -                      Dec 04,01
Page....: 1                                           08:05 PM

      Line                Source Text
-----
= 10  **SAG GENERATOR: BROWSE                                VERSION: n.n.n
- 20  **SAG TITLE: Summary browse Ords
- 30  **SAG SYSTEM: SACADB
- 40  **SAG GDA: NCGDA
- 50  **SAG DESCS(1): This program is used to browse the ... Orders
- 60  **SAG HEADER1: Order Entry System
- 70  **SAG HEADER2: Order display by order number
+ 20  **SAG TITLE: Summary Browse for Orders
+ 30  **SAG SYSTEM: SYSCSTDE
+ 40  **SAG GDA: NCGDA
+ 50  **SAG DESCS(1): This program will display all the orders in order
+ 60  **SAG DESCS(2): number sequence.
+ 70  **SAG HEADER1: Order Entry System
+ 80  **SAG HEADER2: Order Display by Order Number
= 80  **SAG DIRECT-COMMAND-PROCESS:
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      retrn quit  print      top

```

Notice the signs displayed to the left of the line numbers. These signs indicate whether the corresponding line is the same in both libraries (=), contains less characters (), or contains more characters (+).

- 5 Continue pressing Enter until the results for all modules have been displayed. The following window is displayed:

```

Library Compare Facility                      Page 55
Dec 04,01                                           08:07 PM

Program  MYLIB    SYSCSTDE Matching
Name     lines    lines    lines    Comments
-----
NCTFWHS1      89      89      89    Exact match
NYORDSUB     100       0       0
NYORMSA       39       0       0
NYORMSR       12       0       0
OFACTORY     100       0       0
SUPRDREW      25       0       0
VB-DREW       65       0       0
VEHLBCPV     100       0       0
VEHLBKEY      13       0       0
VEHLBPRI       8       0       0
VEHLBROW      17       0       0
VEHLBSO      100       0       0
VEHLBSP      100       0       0
VEHLMCDV     100       0       0
VEHLMCPV     100       0       0

```

6 Press Enter. The Source Range Compare Facility window is redisplayed:

```
Source Range Compare Facility
      Library  Database  File  Dominant
=====
Old library..... MYLIB___  ___196  ___4    X
New Library..... SYSCSTDE  ___196  ___4    -
Program range..... A*_____ thru Z*_____
Summary only..... -
Only report if different.. -
Ignore comment lines..... -
Ignore trailing comments.. -
Ignore leading spaces..... -
Only compare object types _____ (ACGHLMPST)
4 members match; 41 members don't match
```

The message at the bottom of the window indicates that 4 members match and 41 do not.

Comparing Two Natural Objects

➤ To compare the Natural objects in different libraries


1 Enter "CSGCMPS" at the Next prompt. The Compare Criteria window is displayed:

```
Compare Criteria
      Library  Object  Database  File  or Source Area
=====
Old version ==> SYSCSTDE  _____  ___196  ___4    -
New version ==> SYSCSTDE  _____  ___196  ___4    -

Options...
  Ignore comment lines..... -
  Ignore trailing comments.. -
  Ignore leading spaces..... -
  Summary only..... -
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10-
                               quit
```

2 Specify the following:

MYLIB	Library field for the Old version
MYBROWSE	Object field for the Old version
MYBROWSE	Object field for the New version

 **Note:** The "Old" and "New" designation is for identification purposes only; it does not indicate older or newer versions of the objects.

- 3 Press Enter. The Source Compare Facility window is displayed:

```

Library: MYLIB          **** N A T U R A L   C O N S T R U C T ****          CSGCMPS1
Object.: MYBROWSE      - SOURCE COMPARE FACILITY -                      Dec 04,01
Page....: 1                                                    08:16 PM

  Line                Source Text
-----
= 10  **SAG GENERATOR: BROWSE                                VERSION: n.n.n
- 20  **SAG TITLE: Summary browse Ords
- 30  **SAG SYSTEM: SACADB
- 40  **SAG GDA: NCGDA
- 50  **SAG DESCS(1): This program is used to browse the ... Orders
- 60  **SAG HEADER1: Order Entry System
- 70  **SAG HEADER2: Order display by order number
+ 20  **SAG TITLE: Summary Browse for Orders
+ 30  **SAG SYSTEM: SYSCSTDE
+ 40  **SAG GDA: NCGDA
+ 50  **SAG DESCS(1): This program will display all the orders in order
+ 60  **SAG DESCS(2): number sequence.
+ 70  **SAG HEADER1: Order Entry System
+ 80  **SAG HEADER2: Order Display by Order Number
= 80  **SAG DIRECT-COMMAND-PROCESS:
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---
              quit  print          top                      <      >

```

The signs displayed to the left of the line numbers indicate whether the corresponding line is the same in both libraries (=), contains less characters (), or contains more characters (+).

- 4 Continue pressing Enter until the last line is displayed. The Comparison Results window is displayed:

```

Comparison Results
Number of lines in lib MYLIB    ...:   775
Number of lines in lib SYSCSTDE ...:   755
Number of matching source lines ...:   724

```

The results indicate that the MYBROWSE object in MYLIB has 20 more lines than the same object in the demo system, and that 724 source lines match in both objects.

For more information about the CSGCMPS utility, see *CSGCMPS Utility* in *Natural Construct Administration and Modeling*.

Creating Online Reports

Part of your job as a programmer is to create online reports. Did you know you can use the Browse model to quickly create multi-panel reports? Although you can easily write a browse program yourself, it's not always easy to include some of the functions users want, such as:

- Scrolling up and down through data, as well as left and right
- Printing the report
- Using wildcard characters to specify starting points, such as < or >
- Drilling down to display additional details

When you use the Browse model to create reports, you can provide all these functions. In addition, you can regenerate the report if the database changes (from DB2 to Adabas, for example) and the model will supply the appropriate database code.



Note: If the report produces large volumes of output, use the Batch model. For more information, see *Batch Model* in *Natural Construct Generation*.

To build fields for display, Natural Construct defaults the display value to the specified browse key. This functionality is ideal for generating quick browse routines to check the records in a file. For example, you can specify Customer-Name as the key field and the generated program will browse the Customer file by customer name.

If you want to display information about more than one field, you can use the WRITE-FIELDS user exit. The code in this exit defines additional logic to write information to each screen based on the contents of the #PANEL variable.

Another powerful feature of the Browse model is the PROCESS-SELECTED-RECORD user exit. You can use this exit to specify processing performed on a selected record based on the record's unique key. For more information about the user exits, see *WRITE-FIELDS* and *PROCESS-SELECTED-RECORD* in *Natural Construct Generation*.

The following sections describe how to use the Browse model to create different reports. For more detailed information about the Browse model, see *Browse Models* in *Natural Construct Generation*.

- [Creating a Report for a Key Field](#)
- [Creating a Report for Multiple Fields](#)
- [Creating a Drill-Down Report](#)

■ Creating a Drill-Down Report Using Different Keys

Creating a Report for a Key Field



Note: Before beginning the procedure, ensure that the required modules have been copied to the current library. For a list of modules, see *Before You Begin*.

➤ To create a report for a key field

- 1 Enter "ncstg" at the Natural prompt. The Generation main menu is displayed.
- 2 Specify the following:

M	Function field
MYBROWSE	Module field
Browse	Model field

- 3 Press Enter. The Standard Parameters panel is displayed:

```

CUSCMA                               BROWSE Program                      CU--MA0
Oct 23                               Standard Parameters                  1 of 4

Module ..... MYBROWSE
System ..... MYLIB_____
Global data area ... CDGDA___
With block ..... _____

Title ..... Browse ..._____
Description ..... This program is used to browse the ..._____
_____
_____
_____

First heading ..... _____
Second heading ..... _____

Command ..... _
Message numbers .... _
Password ..... _

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
right help  retn quit                                     right main

```

- 4 Specify the following:

NCGDA	Global data area field
Summary Browse for Orders	Title field
This program will display all the orders in order number sequence	Description field
Order Entry System	First header field
Order Display by Order Number	Second header field



Note: For more information about this panel, see General Model Specifications, Natural Construct Generation.

- 5 Press Enter. The Additional Parameters panel is displayed:

```

CUSCMB                                BROWSE Program                                CUSCMB0
Oct 23                                Additional Parameters                                2 of 4

Predict view ..... *
  Natural (DDM) .....
  Program view .....
Primary key ..... *

Horizontal panels ..... 1_
Backward scroll pages .... 10

Input using map ..... *
Reserved input lines ..... _
Minimum key value .....
Maximum key value .....
Single prompt ..... _
Multiple prompts ..... _

Wildcard support ..... _      Export data support ..... _
Hardcopy support ..... _      Window support ..... _

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---
      help  retrn quit          windw                                left  right main

```

- 6 Select the following:

- "NCST-ORDER-HEADER" from the Predict view field
- "ORDER-NUMBER" from the Primary key field

The report will browse the NCST-ORDER-HEADER file by order number.

- 7 Mark the following:

- Wildcard support field

Marking this field enables wildcard processing in the generated program. Users can enter wildcard characters, such as < or *, to limit the query.

■ Window support field

This enables the output to be displayed in a window, instead of on a full screen.

- 8 Press Enter. The Window Parameters window is displayed:

```
CU--DWM          Natural Construct          CU--DWM0
Oct 23           Window Parameters          1 of 1

      Size ..... Height ..... ____
                Width ..... ____

      Position ..... Line ..... ____
                  Column ..... ____

      Frame OFF .... _
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---
      help retrn quit test
```



Note: You can also display this window by pressing PF5 (windw).

- 9 Specify the following:

24	Height field
10	Line field
10	Column field

- 10 Press Enter twice. The Additional INPUT Parameters panel is displayed.

- 11 Press Enter. The Restriction Parameters panel is displayed.



Note: For more information about these panels, see *Additional INPUT Parameters Panel* and *Restriction Parameters Panel* in *Natural Construct Generation*.

- 12 Press Enter. You are returned to the Generation main menu.

- 13 Enter "G" in the Function field.

- 14 Enter "W" in the Function field. Your browse program is now generated and stowed.

To see the code generated by the Browse model, enter "E" in the Function field.



Tip: To browse the orders by customer number, you can select Customer-Number as the primary key.

To see what the window will look like to users, enter "T" in the Function field. The following window is displayed:

```
MYBROWSE          ***** Order Entry System *****
Oct 25            - Order Display by Order Number -      9:47 PM

Order Number
-----

Order Number: _____
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---
      help  retrn quit          flip          bkwrд frwrд
```

To display the order numbers, press Enter or PF8 (frwrд).

- 15 Press PF2 (retrn). You are returned to the Generation main menu.

You've successfully created a report, but it only displays details about one field. To add the logic to display details about additional fields, you must select and define the WRITE-FIELDS user exit. The following section describes how to do this.

Creating a Report for Multiple Fields

You can create a report for more than one field by selecting and defining the WRITE-FIELDS user exit. This user exit helps you create the WRITE or DISPLAY statements required to display the additional fields on the report.

If you regenerate the report, the user exit code is not touched. You must select and define the user exit again and re-create the user exit code.



Tip: Because this user exit is invoked after each record in the primary file is read, you can also include the code to perform calculations if desired.

➤ To create a report for more than one field

- 1 Enter "U" in the Function field. The first User Exit panel is displayed:

CSGSAMPL		NATURAL CONSTRUCT		CSGSM0	
Oct 23		User Exits		1 of 1	
User Exit		Exists	Sample	Required	Conditional
-----				-----	
—	CHANGE-HISTORY		Subprogram		
—	PARAMETER-DATA		Example		X
—	HE-PARAMETER-INDEXES		Example		X
—	BUILD-REPORT-LOCAL-VARS				
—	LOCAL-DATA		Subprogram		
—	START-OF-PROGRAM		Example		
—	SELECT-STATEMENT		Subprogram		X
—	AFTER-READ		Example		
—	REJECT-AFTER-MAX-KEY-CHECK				
—	EXPORT-DATA		Subprogram		X
—	WRITE-FIELDS		Subprogram		
—	TOP-OF-PAGE		Example		
—	BEFORE-INPUT		Example		
—	BEFORE-STANDARD-KEY-CHECK		Example		
—	AFTER-INPUT		Example		
—	HARDCOPY-EDITS		Example		X
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11--PF12---					
frwr help retrn quit			bkwr frwr		



Tip: If this panel is not displayed, enter "SAMPLE" on the command line.

- 2 Mark the WRITE-FIELDS user exit.
- 3 Press Enter twice. The WRITE-FIELDS Data Parameters window is displayed:

CSUSELFV		NATURAL CONSTRUCT				CSUSELF0	
Oct 23		WRITE-FIELDS Data Parameters				1 of 1	
Label	Type	PREDICT Views or Data Areas				Select	All
1	View	NCST-ORDER-HEADER_____	*	—	—		
2		_____	*	—	—		
3		_____	*	—	—		
4		_____	*	—	—		
5		_____	*	—	—		
6		_____	*	—	—		
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11							
help retrn							

This panel determines whether you want to use all the fields in the file or only a subset. Based on your previous specifications, Natural Construct supplies the name of the file you're using. NCST-ORDER-HEADER is displayed in the first Predict Views or Data Areas field.

- 4 Mark the Select field following NCST-ORDER-HEADER. This indicates that you want to select the fields.

- 5 Press Enter. The Select NCST-ORDER-HEADER Field window is displayed:

CSGSELF0		Natural Construct			CSGSELF0
Oct 23		Select NCST-ORDER-HEADER Field			1 of 1
Type	Level	Field name	Format	Length	Desc
	1	_ ORDER-NUMBER	N	6.0	D
	1	_ ORDER-AMOUNT	P	13.2	
	1	_ ORDER-DATE	N	8.0	
	1	_ ORDER-CUSTOMER-NUMBER	N	5.0	D
	1	_ ORDER-WAREHOUSE-ID	A	3.0	D
	1	_ INVOICE-NUMBER	N	6.0	
	1	_ ORDER-TIMESTAMP	T		
MC	1	_ DELIVERY-INSTRUCTIONS	A	60.0	N
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--					
help retrn					

This window shows the fields in the NCST-ORDER-HEADER file.

- 6 Mark the following fields:

- ORDER-NUMBER
- ORDER-AMOUNT
- ORDER-DATE
- ORDER-CUSTOMER-NUMBER
- ORDER-WAREHOUSE-ID
- INVOICE-NUMBER

- 7 Press Enter. The first BROWSE Program Build Report panel is displayed:

CSUBLDRP		Natural Construct					CSUBLD10
Oct 23		BROWSE Program Build Report					1 of 2
1_	Panel(*)	Order	Label	Field name	Format	AL=	
>>	-----	-----	-----	-----	-----	-----	-----
1	___	___	1 *	ORDER-NUMBER_____	N 6.0__	_____	
2	___	___	1 *	ORDER-AMOUNT_____	P 13.2__	_____	
3	___	___	1 *	ORDER-DATE_____	N 8.0__	_____	
4	___	___	1 *	ORDER-CUSTOMER-NUMBER_____	N 5.0__	_____	
5	___	___	1 *	ORDER-WAREHOUSE-ID_____	A 3.0__	_____	
6	___	___	1 *	INVOICE-NUMBER_____	N 6.0__	_____	
7	___	___	- *	_____	- _____	_____	
8	___	___	- *	_____	- _____	_____	
9	___	___	- *	_____	- _____	_____	
10	___	___	- *	_____	- _____	_____	
11	___	___	- *	_____	- _____	_____	
12	___	___	- *	_____	- _____	_____	
13	___	___	- *	_____	- _____	_____	
14	___	___	- *	_____	- _____	_____	
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---							
help retrn deflt test gen selfd frwrд bkwrд reord right proto							

This panel displays the fields selected for the report and allows you to define where and how the fields are displayed. The second panel (displayed when you press PF11) allows you to change the default values for the selected fields, such as the field prompt name or the field length and format.



Note: For more information, see the online help or *Build Report Panel 1* and *Build Report Panel 2* in *Natural Construct Generation*.

- 8 Press PF12 (proto). Natural Construct fills in a panel number and order (sequence) for each field:

CSUBLDRP		Natural Construct					CSUBLD10
Oct 25		BROWSE Program Build Report					1 of 2
1_	Panel(*)	Order	Label	Field name	Format	AL=	
>> -----							
1	01	010	1 *	ORDER-NUMBER_____	N 6.0__	_____	
2	01	020	1 *	ORDER-AMOUNT_____	P 13.2__	_____	
3	01	030	1 *	ORDER-DATE_____	N 8.0__	_____	
4	01	040	1 *	ORDER-CUSTOMER-NUMBER_____	N 5.0__	_____	
5	01	050	1 *	ORDER-WAREHOUSE-ID_____	A 3.0__	_____	
6	02	010	1 *	INVOICE-NUMBER_____	N 6.0__	_____	
7	---	---	- *	_____	- _____	_____	
8	---	---	- *	_____	- _____	_____	
9	---	---	- *	_____	- _____	_____	
10	---	---	- *	_____	- _____	_____	
11	---	---	- *	_____	- _____	_____	
12	---	---	- *	_____	- _____	_____	
13	---	---	- *	_____	- _____	_____	
14	---	---	- *	_____	- _____	_____	

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
 help retrn deflt test gen selfd frwr bkwrd reord right proto



Note: When ordering fields on the reports, Natural Construct uses 10, 20, 30, etc., instead of 1, 2, 3, etc. This allows you to easily insert fields later if necessary. For example, to insert a new field in the second position, you can type 15.

- 9 Press PF4 (test). The Test Display window is displayed:

```

CSGBLDTE                                BROWSE Program
Oct 25                                  Test Display

Panel .. 01 -----
Order                                Customer Warehouse
Number    Order Amount    Order Date    Number    ID
-----
999999 999999999999999999 99999999 99999  XXX

Panel .. 02 -----

Invoice Number
-----

          999999

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---
      test test test test test test test test test test

```

Notice that the Invoice Number field is displayed on the second panel by itself. This will make it difficult for a user to determine what order the invoice is associated with. Also, the first panel appears to be crowded so you may want to remove one of the fields from the first panel and place it on the second panel.

- 10 Press Enter. You are returned to the Build Report panel.
- 11 Specify the following:

* (asterisk)	Panel field preceding ORDER-NUMBER
2	Panel field preceding ORDER-WAREHOUSE-ID
30	Order field for ORDER-WAREHOUSE-ID

The order number will appear as the first field on all panels of the report and the warehouse ID will now appear as the third field on the second panel.

- 12 Press PF9 (reord). Natural Construct reorders the fields on the panels.
- 13 Press PF4 (test) again. The Test Display window is displayed:

```

CSGBLDTE                                BROWSE Program
Oct 25                                Test Display

Panel .. 01 -----
Order                                Customer
Number    Order Amount    Order Date    Number
-----
999999 999999999999999999 99999999 99999

Panel .. 02 -----
Order                                Warehouse
Number Invoice Number    ID
-----
999999 999999 XXX

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---
      test test test test test test test test test

```

- 14 Press Enter. You are returned to the Build Report panel.
- 15 Press PF5 (gen). The code is generated into the User Exit editor:

```

0010 DEFINE EXIT WRITE-FIELDS
0020 DECIDE ON FIRST VALUE #PANEL
0030   VALUE 01
0040     WRITE(0)
0050       NCST-ORDER-HEADER.ORDER-NUMBER
0060       NCST-ORDER-HEADER.ORDER-AMOUNT
0070       2X NCST-ORDER-HEADER.ORDER-DATE
0080       3X NCST-ORDER-HEADER.ORDER-CUSTOMER-NUMBER
0090   VALUE 02
0100     WRITE(0)
0110       NCST-ORDER-HEADER.ORDER-NUMBER
0120       5X NCST-ORDER-HEADER.INVOICE-NUMBER
0130       8X NCST-ORDER-HEADER.ORDER-WAREHOUSE-ID
0140   NONE
0150     WRITE(0) 23X 'No fields to display on this panel'
0160 END-DECIDE
0170 END-EXIT
0180 DEFINE EXIT TOP-OF-PAGE
0190 **
0200 DECIDE ON FIRST VALUE #PANEL
0210   VALUE 01
0220     WRITE(0)
0230       'Order' (I)
0240       30X 'Customer' (I)

```



```

0250     WRITE(0)
0260     'Number' (I)
0270     3X 'Order' (I)
0280     1X 'Amount' (I)
0290     3X 'Order' (I)
0300     1X 'Date' (I)
0310     2X 'Number' (I)
0320     WRITE(0)
0330     '-----' (I)
0340     1X '-----' (I)
0350     1X '-----' (I)
0360     1X '-----' (I)
0370     VALUE 02
0380     WRITE(0)
0390     'Order' (I)
0400     17X 'Warehouse' (I)
0410     WRITE(0)
0420     'Number' (I)
0430     1X 'Invoice' (I)
0440     1X 'Number' (I)
0450     4X 'ID' (I)
0460     WRITE(0)
0470     '-----' (I)
0480     1X '-----' (I)
0490     1X '-----' (I)
0500     NONE
0510     WRITE(0) 23X 'No fields to display on this panel'
0520     END-DECIDE
0530     SKIP(0) 1
0540     END-EXIT

```

If there's more than one panel, Natural Construct automatically generates the required WRITE statements and provides default column headings in the TOP-OF-PAGE user exit (see the example above).



Note: If there is only one panel, Natural Construct generates a DISPLAY statement.

- 16 Enter "." on the command line. You are returned to the Generation main menu.
- 17 Enter "G" in the Function field.
- 18 Enter "W" in the Function field. Your browse program is now generated and stowed.
- 19 Enter "T" in the Function field. The following panel is displayed:

```

MYBROWSE          ***** Order Entry System *****
Oct 25            - Order Display by Order Number -      1 more >

Order              Customer
Number  Order Amount  Order Date  Number
-----

```

Order Number: _____

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---

help retn quit flip bkwr frwr

20 Press Enter. The order data is displayed. Use the PF-keys to scroll left and right, up and down.

21 Press PF11 (right). The next panel of the report is displayed:

```

MYBROWSE          ***** Order Entry System *****
< 1 more         - Order Display by Order Number -      10:00 PM

Order              Warehouse
Number Invoice Number  ID
-----

```

323232	332323	222
323456	123	544
332222	10294	111
333333	231201	111
422222	190123	111

*** End of Data ***

Order Number: _____

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---

help retn quit flip bkwr frwr

22 Press PF2 (retn). You are returned to the Generation main menu.

You've successfully created an online report with additional fields. If desired, you can also create a drill-down report to display details about a selected record. See the following section, [Creating a Drill-Down Report](#), for information.



Note: If you have any problems creating the report, see [Troubleshooting and Debugging](#) for possible reasons.

Creating a Drill-Down Report

You can create a drill-down report that displays additional details about a record by selecting and defining the PROCESS-SELECTED-RECORD user exit. The code in this exit calls a browse subprogram when a user selects a record displayed on a report.

➤ To create a drill-down report

- 1 Enter "U" in the Function field. The User Exit editor is displayed.
- 2 Enter "SAMPLE" on the command line. The first User Exit panel for the Browse model is displayed.
- 3 Press Enter again. The second User Exit panel is displayed:

CSGSAMPL	NATURAL CONSTRUCT			CSGSM0
Oct 25	User Exits			1 of 1
	User Exit	Exists	Sample	Required Conditional
—	HARDCOPY-TERMINATING-PROCESS		Example	X
—	PROCESS-SELECTED-RECORD		Subprogram	X
—	END-OF-PROGRAM		Example	
—	ASSIGN-PREFIX-VALUE		Subprogram	X
—	SET-PF-KEYS		Example	
—	MISCELLANEOUS-SUBROUTINES		Example	
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---				
help retrn quit bkwr frwr				

- 4 Mark the PROCESS-SELECTED-RECORD user exit.

- 5 Press Enter. The User Exit editor is displayed, showing the default code generated for the PROCESS-SELECTED-RECORD user exit:

```
Module ..... MYBROWSE
Title ..... Summary Browse for Orders
>
> + ABS: _ X-Y: _ S 69 L 52
Bot ....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
0520 END-DECIDE
0530 SKIP(0) 1
0540 END-EXIT
0550 DEFINE EXIT PROCESS-SELECTED-RECORD
0560 *
0570 * Processing to be performed after a record has been selected.
0580     IF *PF-KEY = 'ENTR' THEN
0590         /*
0600         /* Process #SELECTED-KEY record here.
0610         IGNORE
0620     ELSE
0630         /* PF8 will just reposition the selected value to the top of
0640         /* the screen OR you may wish to use PF8 to perform different
0650         /* OR you may wish to use PF8 to perform
0660         /* record processing.
0670         IGNORE
0680     END-IF
0690 END-EXIT
```

The Natural Construct-defined variable, #SELECTED-KEY, identifies the selected record. In this case, the order number is the selected key.

- 6 Change the default code as follows:

```

Module ..... MYBROWSE
Title ..... Summary Browse for Orders
>
> + ABS: _ X-Y: _ S   69 L   52
Bot  ....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
0520 END-DECIDE
0530 SKIP(0) 1
0540 END-EXIT
0550 DEFINE EXIT PROCESS-SELECTED-RECORD
0560 *
0570 * Processing to be performed after a record has been selected.
0580     IF *PF-KEY = 'ENTR' THEN
0590     CALLNAT 'NCOSODET'
0600         #SELECTED-KEY
0610         'select fun' /* Note NCOSODET has a 10-character parm
0620                        /* field in case a function needs to be passed.
0630                        /* In this example we don't need it, but we must
0640                        /* pass something.
0650         DIALOG-INFO
0660         MSG-INFO
0670         PASS
0680     END-IF
0690 END-EXIT

```

Notice that the syntax calls the NCOSODET module, which displays order lines for a selected order.

- 7 Enter "." on the command line. You are returned to the Generation main menu.
- 8 Enter "G" in the Function field.
- 9 Enter "W" in the Function field. Your browse program is now generated and stowed.
- 10 Enter "T" in the Function field. The following panel is displayed:

```

MYBROWSE      ***** Order Entry System *****
Oct 25        - Order Display by Order Number -      1 more >

Order          Customer
Number      Order Amount  Order Date  Number
-----

```

Order Number: _____

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---

help retrn quit flip bkwr frwr

- 11 Press Enter to display the orders.
- 12 Select an order. The order lines for the selected order are displayed in a window:

```

NCOSODET      ***** ORDER SUBSYSTEM *****
Oct 25        - ORDER DETAIL -      1 more >

Ln Prod Id Quantity  Unit Cost  Total Cost
-----
1 654070          40    100.00    4000.00
2 222222          100    50.00    5000.00
      *** End of Data ***

Line Order Number: 234234 Line Number: __
Direct command...: _____
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8
      help retrn quit flip bkwr frw

```

- 13 Press PF2 (retrn) twice. You are returned to the Generation main menu.

You've successfully created a drill-down report.

The following section describes the NCOSODET subprogram used in this example.



Note: If you have any problems creating the report, see [Troubleshooting and Debugging](#) for possible reasons.

NCOSODET Subprogram

NCOSODET is a subprogram that displays additional details about a selected record. To learn more about the module, look at the information specified on the specification panels and in the user exits.

➤ To see the specifications and user exits for NCOSODET

- 1 Enter "ncstg" at the Natural Next prompt. The Generation main menu is displayed.
- 2 Specify the following:

R	Function field
NCOSODET	Module field

- 3 Press Enter. Natural Construct reads the module into the edit buffer and displays a confirmation message.
- 4 Enter "M" in the Function field. The Standard Parameters panel is displayed:

```

CUSCMA                      BROWSE-SUBP Subprogram          CU--MA0
Oct 23                      Standard Parameters              1 of 5

Module ..... NCOSODET
System ..... MYLIB_____
Global data area ... _____
With block ..... _____

Title ..... Order Header Detail_____
Description ..... This subprogram will display all the lines associated__
                  with the selected order entry._____
                  _____
                  _____

First heading ..... ORDER SUBSYSTEM_____
Second heading ..... ORDER DETAIL_____

Command ..... X
Message numbers .... _
Password ..... _

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
right help  retrn quit                                right main

```

As indicated by the title, the NCOSODET module is a browse subprogram. Because it is a subprogram, and not a program, a GDA is not required.

- 5 Press Enter. The Additional Parameters panel is displayed.

- 6 Press Enter again. The Additional INPUT Parameters panel is displayed. These two panels are similar to the panels for a browse program.
- 7 Press Enter again. The Specific Parameters panel for a browse subprogram is displayed:

```
CUSCMF                      BROWSE-SUBP Subprogram          CUSCMF0
Oct 23                      Specific Parameters              4 of 5

Parameter Override
Note: Format and Length defaults to ORDER-LINE-KEY key
Field Name ..... #PDA-KEY
Natural format ..... N 6.0__ *
```

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
right help retrn quit left right main

When the key for the browse subprogram differs from the key for the calling program, you specify the format and length of the passed key (#SELECTED-KEY) on this panel. Since the format and length defaults to the ORDER-LINE-KEY key, which contains the order number and order line sequence number, the format and length has been changed to that of the ORDER-NUMBER key (N6).



Note: If the key for the browse subprogram is the same size as that of the calling program, you do not need to specify the format and length on this panel.

- 8 Press Enter. The Restriction Parameters panel is displayed:


```
CUSCMG                                BROWSE-SUBP Subprogram                                CUSCMG0
Oct 23                                Restriction Parameters                                5 of 5

Prefix Options
Restrict browse with prefix .. X

Number of characters (bytes) . 6__
Number of components ..... __

Protect prefix ..... X
Suppress prefix ..... _

Field Name ..... _____

Prefix Helproutine Specifications
Component      Helproutine      Parameter Name
  1              *              _____
  2              *              _____

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
main  help  retrn quit                                left  userX main
```

Because the default key contains the order number and the line sequence number, the first 6 digits (which represent the order number) must be protected. This allows the calling program to determine which order is browsed; the user cannot change the order number.

- 9 Press Enter. You are returned to the Generation main menu.

At this point, you can read the MYBROWSE module back into the editor and re-test your report. Drill down to the order lines and see if you can change the order number.

The following section describes how to create a drill-down report when the calling program and the subprogram use different keys.

Creating a Drill-Down Report Using Different Keys

This section describes the NCOSCSTO module, which browses the NCST-ORDER-HEADER file by customer date. When a user selects a customer, details about the customer's orders are displayed.

To see how this is done, read the NCOSCSTO module into the editor and display the user exits:

```
0010 DEFINE EXIT LOCAL-DATA
0020   LOCAL
0030   01 ORDER-DETAIL VIEW OF NCST-ORDER-HEADER
0040     02 ORDER-NUMBER
0050 END-EXIT /* LOCAL-DATA
0060 DEFINE EXIT WRITE-FIELDS
0070   DISPLAY
0080     1X 'Order Date'(I) NCST-ORDER-HEADER.ORDER-DATE
0090       (AD=I EM=9999/99/99)
0100     'Order No'(I) NCST-ORDER-HEADER.ORDER-NUMBER
0110       (LC=^ )
0120     'Invoice No'(I) NCST-ORDER-HEADER.INVOICE-NUMBER
0130       (LC=^^ )
0140     'WH ID'(I) NCST-ORDER-HEADER.ORDER-WAREHOUSE-ID
0150       (LC=^ )
0160     'Order Amount'(I) NCST-ORDER-HEADER.ORDER-AMOUNT
0170 END-EXIT
0180 DEFINE EXIT BEFORE-INPUT
0190 *
0200 * Processing to be performed just before the input statement.
0210 /* Change standard message to indicate that selection can be done
0220 /* ONLY by positioning the cursor (not entering key value since
0230 /* input is protected.
0240 ASSIGN MSG-INFO.##MSG = 'Position cursor to select.'
0250 END-EXIT
0260 DEFINE EXIT PROCESS-SELECTED-RECORD
0270 *
0280 * Processing to be performed after a record has been selected.
0290 IF *PF-KEY = 'ENTR' THEN
0300   GET ORDER-DETAIL #SELECTED-ISN
0310   RESET MSG-INFO
0320   CALLNAT 'NCOSODET'
0330     ORDER-DETAIL.ORDER-NUMBER
0340     CDSELPDA.SELECTED-FUNCTION
0350     DIALOG-INFO
0360     MSG-INFO
0370     PASS
0380 END-IF
0390 END-EXIT /* PROCESS-SELECTED-RECORD
```

Notice that the LOCAL-DATA user exit has been defined. Because the NCOSCSTO module browses by customer date, the customer date is available in the #SELECTED-KEY variable.

When a user drills down to display additional details, the module must have access to the NCST-ORDER-HEADER file to display the order details. To do this with Adabas, you can use the #SELECTED-ISN variable to retrieve the order header file and find the order number.

Also notice the GET statement in the PROCESS-SELECTED-RECORD user exit. When a user selects a record, the order header file is retrieved and the order number is passed to the NCOSODET module.

Writing Natural Statements

Another way Natural Construct can save you time and effort is by using the statement models. These models generate individual Natural statements or functions (blocks of code). Some of the benefits of using the statement models are:

- Reduces your dependency on the Natural documentation to recall syntax
- Reduces the number of keystrokes required to code Natural statements, since keywords are generated automatically
- Produces statement code that:
 - has a consistent indentation scheme
 - programmatically performs tedious calculations
 - accesses your system files to retrieve Predict views, etc.

The following sections describe some of the ways you can use the statement models to save time and effort.

- [Generating a Predict View](#)
- [Generating a DECIDE-ON Statement](#)
- [Generating a CALLNAT Statement](#)
- [Repeating a Sequence With Variations](#)

For more information, see *Statement Models* in *Natural Construct Generation*.



Notes:

1. Each statement model has a two-character alias you can use to invoke the model. For example, the alias for the View model is VW. The model alias is used in the statement model examples.
2. Before beginning the procedures, ensure that the required modules have been copied to the current library. For a list of modules, see [Before You Begin](#).

Generating a Predict View

You can use the View statement model to generate a Predict view. The following sample procedure generates a view for the NCST-CUSTOMER file.

➤ To generate a Predict view for the NCST-CUSTOMER file

- 1 Enter "e p" at the Next prompt. The Natural editor is displayed.
- 2 Press the Tab key twice. The cursor should be in the first position of the first line in the editor. This is where you enter the .g line command.

- 3 Enter ".g". Because you didn't specify a model name, the Select Model window is displayed:

```

CSGDLIST          Natural Construct          CSGDLSTO
Oct 25             Select Model              1 of 1

      Model                      Type
      -----
      ADD                        Statement
      ARBITRARY-FIELD-PROCESSING Statement
      AT-TOP-OF-PAGE             Statement
      CALLNAT                    Statement
      CLOSE                      Statement
      COMPRESS                   Statement
      DECIDE-FOR                  Statement
      DECIDE-ON                   Statement
      DEFINE-PRINTER              Statement
      DEFINE-SUBROUTINE           Statement
      DEFINE-VARIABLE             Statement
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---
      help  retrn                bkwrdr frwrdr
Model .. _____ All models ... _

```

- 4 Enter "V" in the Model field. The list of models is repositioned to model names beginning with "V".
- 5 Select the View model. The VIEW Function window is displayed:

```

                        VIEW Function

Predict view ..... *
Program view .....

C* variables ..... X
Redefined components ..... X
Superdescriptor fields ..... _
Natural format specifications .. _

```

- 6 Select "NCST-CUSTOMER" from the Predict view name field.
- 7 Press Enter. The following view is generated:

```

0010 /*
0020 /*
0030 01 NCST-CUSTOMER VIEW OF NCST-CUSTOMER
0040 02 CUSTOMER-NUMBER
0050 02 BUSINESS-NAME
0060 02 PHONE-NUMBER
0070 02 MAILING-ADDRESS
0080 03 M-STREET
0090 03 M-CITY

```

```

0100      03 M-PROVINCE
0110      03 M-POSTAL-CODE
0120      02 SHIPPING-ADDRESS
0130      03 S-STREET
0140      03 S-CITY
0150      03 S-PROVINCE
0160      03 S-POSTAL-CODE
0170      02 CONTACT
0180      02 CREDIT-RATING
0190      02 CREDIT-LIMIT
0200      02 DISCOUNT-PERCENTAGE
0210      02 CUSTOMER-WAREHOUSE-ID
0220      02 CUSTOMER-TIMESTAMP

```

Generating a DECIDE-ON Statement

When you use the statement models to generate Natural statements, you don't need to remember the statement syntax.

➤ To generate a DECIDE-ON statement

- 1 Enter ".g(do)" at the beginning of a line in the Natural editor. The following window is displayed:

```

DECIDE ON FIRST X EVERY _ VALUE OF #NUMBER_____
>> 1 VALUE 1_____
    ASSIGN ..._____
    _____
    _____
    _____

    ANY VALUE
    _____
    _____

    ALL VALUE
    _____
    _____

    NONE VALUE
    IGNORE_____
    _____

END-DECIDE

```

- 2 Specify the following:

*PF-KEY	VALUE OF field
'PF1'	1 VALUE field
PERFORM ROUTINE-UPD	Over "ASSIGN"

- 3 Press Enter. The following code is generated:

```
0010 /*
0020 /*
0030 DECIDE ON FIRST *PF-KEY
0040     VALUE 'PF1'
0050     PERFORM ROUTINE-UPD
0060     NONE
0070     IGNORE
0080 END-DECIDE
```

Generating a CALLNAT Statement

When using subprograms, you may want to generate a CALLNAT statement to supply the parameters. You can do this easily using the Callnat statement model. For example, you can generate a CALLNAT statement in the PROCESS-SELECTED-RECORD user exit to pass the selected key to an object PDA. This PDA then passes the data to an object maintenance subprogram for processing.

The following example shows how to use the CALLNAT statement within the PROCESS-SELECTED-RECORD user exit to supply the parameters for MYORDSUB (described in [Generating the Object Maintenance Subprogram](#)).



Note: The source code for the specified subprogram must exist in the current library or steplib.

➤ To generate a CALLNAT statement for the MYORDSUB subprogram

- 1 Enter ".g(cn)" at the beginning of a line in the Natural editor. The following window is displayed:

```
CALLNAT _____ *
```



Note: For more information, see *Using the Callnat Statement Window in Natural Construct Generation*.

- 2 Select "MYORDSUB" from the Callnat field.
- 3 Press Enter. The following code is generated:

```

0010 /*
0020 CALLNAT 'MYORDSUB'
0030         ORDER
0040         MYORMSA-ID
0050         MYORMSR
0060         CDAOBJ2
0070         MSG-INFO

```

Notice that all the parameters from the subprogram are added to the CALLNAT statement. Although you may need to change the names to correspond with the program names, this list helps you determine which parameters are required.

Repeating a Sequence With Variations

Not all statement models generate Natural statements. The Sequence model generates a block of code that repeats a given line, with some variations, as many times as specified. The changes are applied to each variation.

➤ To generate a repeating sequence with variations

- 1 Enter ".g(sq)" at the beginning of a line in the Natural editor. The following window is displayed:

```

Repeat 3__
> Assign initial-value = my-array(&1&:&2&)_____
> _____
> _____
> _____
> _____
> _____
      &n& FROM STEP
      &1& 001  1__
      &2& 001  1__
      &3& ____  ____
      &4& ____  ____
      &5& ____  ____
      &6& ____  ____
      &7& ____  ____
      &8& ____  ____
      &9& ____  ____

```

- 2 Specify the following:

7	Repeat field
WRITE '&1&'	Over "Assign initial-value ..."
002	FROM field for &1&
003	STEP field for &1&

3 Press Enter. The following code is generated:

```
0010 /*
0020 /*
0030 WRITE 'LINE 2'
0040 WRITE 'LINE 5'
0050 WRITE 'LINE 8'
0060 WRITE 'LINE 11'
0070 WRITE 'LINE 14'
0080 WRITE 'LINE 17'
0090 WRITE 'LINE 20'
```


5

Troubleshooting and Debugging

■ Specifying Parameters	84
■ Generating Programs	86
■ Working in the Editor	88
■ Viewing Error Messages	88
■ Using the Natural Debugging Tool	91

If you encounter problems while using Natural Construct, there are a few things you can try before contacting your Natural Construct administrator. This chapter describes some of the methods you can use to troubleshoot and/or debug your applications.



Note: Many of these methods are applicable to any Natural program, not just Natural Construct-generated programs.

Natural Construct provides several utilities you can use to trace and debug Natural modules. For information about these and other utilities, see *Supplied Generation Utilities* in *Natural Construct Generation*.

Specifying Parameters

This section contains suggestions you can try if you encounter problems when specifying the model parameters.

Check the Online Help

Before contacting your administrator, check the online help available from each model specification panel. For example, many required fields have active help available. If you press the help PF-key when the cursor is in the field, a window is displayed listing valid entries for that field.

For more information about using the online help, see *Getting Online Help* in *Natural Construct Generation*.

Look at the Demo System

Another suggestion is to look at the demo system supplied with Natural Construct. The demo system is a fully functional order entry application generated using Natural Construct.

There are two ways you can use the demo system for troubleshooting your programs. You can:

- display the model specification panels to see what values were entered for a similar program in the demo system
- look at the generated code to see how a similar program is structured
- invoke the demo system to see how the programs run and interact

The following sections describe each of these options.

Display the Model Specification Panels

➤ To display the model specification panels for a program in the demo system

- 1 Logon to the demo (SYSCSTDE, for example) library.
- 2 Enter "ncstg" at the Natural Next prompt. The Generation main menu is displayed.
- 3 Enter "L" in the Function field. The Select Module window is displayed:

CSGLIST	Natural Construct	CSGLIST0
Oct 22	Select Module	1 of 1
Module	Model	Title
-----	-----	-----
CUSBRSUB	OBJECT-BROWSE-SUBP	Object Browse ...
CUSBR020	OBJECT-BROWSE-STATIC	Object Browse Static ...
CUSTN	OBJECT-MAINT-SUBP	Customer Maintenance
EMPLBR	OBJECT-BROWSE-SUBP	Employee Browse Object
MENU	STARTUP	Demo System Startup Pgm
MYMENU	MENU	Menu ...
MYORDSUB	OBJECT-MAINT-SUBP	Order Maintenance Subprog
MYORMSA	OBJECT-MAINT-PDA	
MYORMSR	OBJECT-MAINT-PDA-R	
NCCFCSTB	BROWSE-SUBP	Browse Customer File
NCCFCUST	OBJECT-MAINT-DIALOG	Customer Maintenance
Module ...	Model	
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11		
help retrn bkwrdr frwrdr		
Position cursor or enter screen value to select		

This window lists the modules contained in the demo library. It shows the module names, the type of model used to generate the modules, and a brief description of each module.



Note: For more information about this window, see *List Generated Modules Function* in *Natural Construct Generation*.

- 4 Move the cursor to the line containing the module for which you want to see specifications.
- 5 Press Enter. You are returned to the Generation main menu.

The name of the module you selected is displayed in the Module field and the name of the model used to generate the module is displayed in the Model field.

- 6 Enter "M" in the Function field. The first specification panel for the selected module is displayed. Continue pressing Enter to display all specification panels for the module. Press Enter again to return to the main menu.
- 7 Enter "U" in the Function field. The user exits for the selected module are displayed.

Display the Generated Code

➤ To display the code generated for a program in the demo system

- 1 Logon to the demo library (SYSCSTDE, for example).
- 2 Enter "ncstg" at the Natural Next prompt. The Generation main menu is displayed.
- 3 Enter "L" in the Function field. The Select Module window is displayed.
- 4 Move the cursor to the line containing the module for which you want to see generated code.
- 5 Press Enter. You are returned to the Generation main menu.
- 6 Enter "E" in the Function field. The code generated for the selected module is displayed.

Invoke the Demo System

➤ To display the model specification panels for a program in the demo system

- 1 Logon to the demo library.
- 2 Enter "menu" at the Natural Next prompt. The main menu for the demo system is displayed.

Generating Programs

This section contains troubleshooting suggestions you can try if you encounter problems when generating programs.

Display the Embedded Statements

When you generate a module from the Generation main menu, a status window is displayed showing the progress of the generation. If a problem occurs, you can optionally regenerate the module and display the embedded statements that perform the generation processing. You can then determine where in the problem occurred.

➤ To display the embedded statements

- 1 Prior to generating the module, press PF5 (optns) on the Generation main menu. The Optional Parameters window is displayed:

```

CSGOPTS          Natural Construct          CSGOPTS0
Oct 25           Optional Parameters         1 of 1

  Status window ..... X
                Step ..... _
                Text ..... _
  Embedded statements ..... _
  Condition codes ..... _
  Post-generation modifications _

  Specifications only ..... _
  Document in PREDICT ..... _
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9-
      help  retrn quit

```

- 2 Mark the Embedded statements field.
- 3 Press Enter.
- 4 Enter "G" in the Function field. The status window is displayed, showing the progress of generation.

When generation is complete, enter "E" in the Function field. The embedded statements are displayed in the generated code.

The following example shows the first 25 lines of code generated by the Object-Maint-Dialog model. An embedded statement is displayed on line 0040:

```

0010 **SAG GENERATOR: OBJECT-MAINT-DIALOG          VERSION: n.n.n
0020 **SAG TITLE: Order Maintenance Dialog
0030 **SAG SYSTEM: MYLIB
0040 **> Trace ..... --> SAVE CUOMS
0050 **SAG GDA: CDGDA
0070 **SAG DESCS(1): This program is used to maintain the order business
0080 **SAG DESCS(2): object.
0090 **SAG HEADER1: Order Entry System
0100 **SAG HEADER2: Maintain Customer Orders
0110 **SAG DIRECT-COMMAND-PROCESS:
0140 **SAG OBJECT-NAME: MYORDSUB
0150 **SAG ACTIONS: 0101010101010000
0160 **SAG ACTION-LENGTH: 2
0190 **SAG ACTIONS-AS-PUSH-BUTTONS: X
0200 **SAG MAX-WINDOWS: 1
0210 **SAG MAP-NAME(1): MYMAP
0220 **SAG UPPER-BOUNDS(1,1): 00030
0230 **SAG UPPER-BOUNDS(1,2): 00010
0240 **SAG UPPER-BOUNDS(1,3): 00020
0250 **SAG SCREEN-OCCURS(1,1): 001

```

After determining where the error occurred, contact your system administrator.



Note: The problem may have occurred in a code frame, a block of code that performs a specific function. The Natural Construct models use code frames, as well as subprograms, in the generation process. For example, the C--BAN9 code frame supplies the standard banner that appears at the top of all generated modules.

Working in the Editor

This section contains troubleshooting suggestions you can try if you encounter problems while working in the Natural editor.

Determine the Last Command Issued

While working in the editor, you may not always remember which command was issued when an error occurred. If this happens, you can display the name of the last command by typing an asterisk (*) on the command line and pressing Enter.

To display a list of the last 10 (or less) commands issued, type "LAST *" and press Enter:

```

                LAST
-  NPC CSGMENU
-  NPC CSGMAIN
-  NPC      CSGMAIN
-  RENUMBER
-  STOW
-  NPC      CSGMAIN
-  RENUMBER
-  STOW
-  ? NAT0247
    
```



Note: This functionality is only available in the Natural editor (not the User Exit editor or Help Text editor).

Viewing Error Messages

This section contains information about viewing Natural error messages.

Determine The Last Error

Sometimes you require the number of the last Natural error that occurred. One method of determining the error number is to enter "TECH" at the Next prompt.



Note: You can also issue the TECH command on the command line in the Natural editor.

A series of three windows is displayed:

```

23:05:21                                01-10-22

User ..... SACAPR
Library ..... MYLIB

Version / SM Level ... 2.3 / 0004
Startup Transaction ..
NATURAL SECURITY ..... Yes
Operating System ..... MVS/ESA
Oper. Sys. Version ... SP6.0.5
Machine Class ..... MAINFRAME
Hardware ..... 7060
TP Monitor ..... COMPLETE
Device Type ..... COLOR
Terminal ID ..... 1      3

Last Command ..... STOW

```

This window shows information about the current settings, such as the user and library names, the operating system, and the terminal ID. It also lists the name of the last command issued. Press Enter to display the second window:

```

23:05:53                                01-10-22

Last Error
Error Number ..... 247

Error Line ..... 1950
Object ..... MYORDDIA
Object Type ..... Program
Level ..... 0
Library .....
DBID/FNR .....

Error Class ..... System
Error Type ..... Syntax
Error Time ..... 2001-10-22 23:05:01

Error Transaction ..

```

This window shows the number of the last error, as well as the error line, class, type, and time. In the example above, 82 is displayed in the Error Number field. Since error numbers consist of four digits, the actual error number is 0082.

Press Enter again to display the last window:

23:06:26			01-10-22		
Steplib	DBID	FNR	Object	Type	Level
-----	-----	-----	-----	-----	-----
SYSTEM	196	4			
SYSTEM	196	3			

This window shows information about the steplib chain, database ID, and file number.



Note: The steplib chain indicates the order in which libraries are searched in the current library. The current library is searched first, then the first steplib, second steplib, etc., up to the eighth steplib (if present). The SYSTEM library is searched last.

Once you know the error number, you can use the method described in the following section to view the error message.

Display a Natural Error Message

Once you know the number of a Natural error, you can display the associated error message text.

> To display the message text

- 1 Type "?" and the four-digit error number at the Next prompt (? 0247, for this example).



Note: You can also issue this command on the command line in the Natural editor.

- 2 Press Enter. The following window is displayed:


```

23:08:00          ***** NATURAL HELP UTILITY *****          01-10-22
Library MYLIB      - NATURAL System Message NAT0247 -          Page 1

    Error in automatic parameter :1: for map.

Tx *** Short Text ***

    Error in automatic parameter ... for map.

Ex *** Explanation ***

    A parameter which is included for a constant input map or write map
    without explicit parameter specification is invalid or undefined.

Ac *** Recommended Action ***

    Correct error in program.

```

After reading the text, you can determine what to do next.

Using the Natural Debugging Tool

To help debug your Natural programs, you can also use the Natural debugging tool. This tool allows you to place break points and watch points at strategic locations in your code.

The procedure to run this utility differs depending on the platform. For more information, see the Natural documentation for your platform.

6

Getting the Most From Natural Construct

■ Providing Online Help for Your Applications	94
■ Building New Models	95
■ Creating JCL	95

This chapter discusses additional capabilities of Natural Construct, such as providing online help information for your applications, building new models, and creating JCL for DOS or OS environments. It also lists add-on products available for use with Natural Construct.

Providing Online Help for Your Applications

You can provide either passive or active online help for all your applications not just those generated using Natural Construct.



Note: For more information about providing help for your applications, see *Natural Construct Help Text*.

The following sections describe each type of online help.

Passive Online Help

Passive help displays a window containing text that describes what to do on a panel or which parameters are required for a field. Within this help window, you can also include hotlinks to other help text members. Hotlinks appear as bolded text within angle brackets; when the user places the cursor over this text and presses Enter, another help window is displayed.

You create passive help in the Help Text subsystem. For example, you can create a help text member that describes the options available on a panel (panel-level help) or lists valid values for a field (field-level help). Using the Help Text subsystem allows you to:

- Create multilingual help text for your applications
- Separate technical writing requirements from the programming

➤ To enable passive help

- 1 Create the help text members in the Help Text subsystem.
- 2 Enter the following as the HE (help) parameter for the panel or field on the associated map:

```
'CD-HELPR',=
```

CD-HELPR is a subprogram supplied with Natural Construct. When a user invokes help on a panel or field, it locates and displays the help member based on the internal name of the panel or field.

Active Online Help

Active online help invokes a browse window listing valid values for a field and allows the user to select a value.

➤ To create the active online help

- 1 Generate the browse help routine module.
- 2 Enter the help routine name as the HE (help) parameter for the field on the associated map.

Building New Models

In addition to the Help Text subsystem, Natural Construct provides the Administration subsystem. This subsystem helps you create your own models or modify the ones shipped with Natural Construct. You must have the appropriate security rights to access this subsystem. Typically, only Natural Construct administrators can access this subsystem.



Note: For more information about creating or modifying models, see *Natural Construct Administration and Modeling*. This guide describes the external programs, subprograms, and help routines used in the model creation process, the supplied utilities, and how to implement security and multilingual support.

Creating JCL

For DOS and OS mainframe environments, Natural Construct supplies models to create a JCL (job control language) member used to execute Natural in batch mode. The models generate the JCL stream into a Natural text object, which you can use as required.

The JCL models are:

Model	Purpose
JCL-DOS-Natbatch	Generates JCL for the DOS environment.
JCL-OS-Natbatch	Generates JCL for the OS environment.

As with all other Natural Construct models, the JCL models are available through the Generation main menu. For more information, see *JCL Models (Mainframe)* in *Natural Construct Generation*.

