USER EXITS FOR THE GENERATION MODELS

This document describes the user exits available for models supplied with the Construct Program Generation Plug-In.

The following topics are covered:

Introduction

User exits are positions within a Natural Construct-generated module where you can insert customized or specialized processing. Changes to the user exit code are always preserved upon subsequent regeneration of the module.

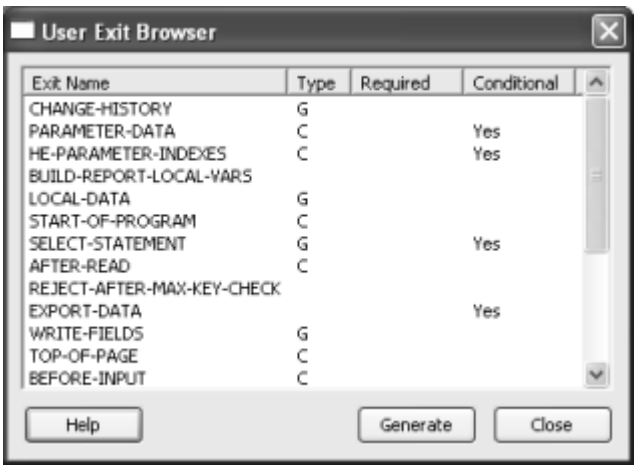
Natural Construct models provide a wide variety of user exits. The exits vary depending on the type of module you are generating. Some exits contain sample code or subprograms, while others generate the DEFINE EXIT and END-EXIT lines only — you provide the actual code. You can modify any user exit code generated into the edit buffer.

If you require code to be inserted in the generated module where no user exit currently exists, have your Natural Construct administrator recommend a suitable exit or add a new exit to the model.

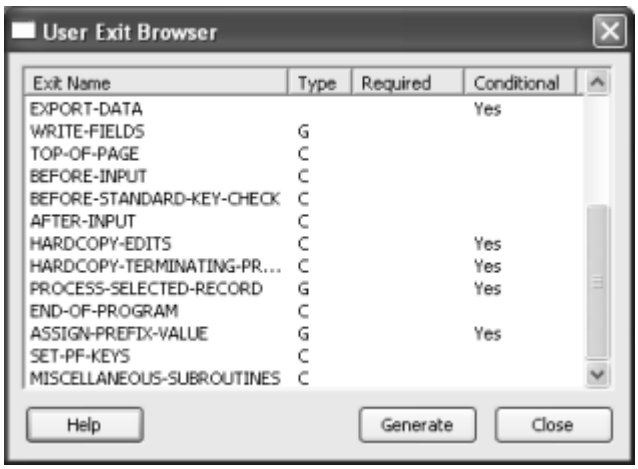
For more information on user exits, see **Generating User Exits**.

Selecting a User Exit

Use the User Exit Browser to display a list of user exits for the module you are generating. The following example shows the User Exit Browser for the Browse model:



User Exit Browser — Part 1



User Exit Browser — Part 2

The fields in the User Exit Browser are:

Field	Description
Exit Name	Name of each user exit available for this model. If a user exit is required and is not conditional (its existence is not based on condition codes in the code frames), it is selected by default.
Type	Type of user exit.
Required	If the corresponding user exit must be specified, Yes is displayed. If the user exit is optional, this field is blank.
Conditional	If the corresponding user exit is conditional (its existence is based on condition codes in the code frames), Yes is displayed. If the user exit is not conditional, this field is blank.

Defining User Exits

The code specified within a user exit depends on the type of module being generated and the type of user exit you are using. However, all user exits have the following format:

```
DEFINE EXIT user-exit-name

  user exit code

END-EXIT user-exit-name
```

Note: Do not insert comments or Natural code on the DEFINE EXIT and END-EXIT lines.

Note: If multiple user exits are generated with the same name, Natural Construct merges them into a single user exit within the generated module.

Supplied User Exits

The following sections describe the user exits available for the supplied models. The user exits are listed in alphabetical order. For many exits, one or more examples are also included.

ADD-ACTION-PROCESSING

The code in this exit specifies additional processing performed for the Add action (in addition to the standard Natural Construct-generated processing).

ADDITIONAL-ACTIONS-PROCESSING

The code in this exit processes any additional actions that were defined within the SELECT-ADDITIONAL-ACTIONS user exit. For more information, see **SELECT-ADDITIONAL-ACTIONS**, page 26.

ADDITIONAL-INITIALIZATIONS

This user exit generates the framework for any additional initializations to be performed in the INITIALIZATIONS subroutine.

Example of code in the ADDITIONAL-INITIALIZATIONS user exit

```
** SAG DEFINE EXIT ADDITIONAL-INITIALIZATIONS
* Assign parameters for help routine CD-HELPR
MOVE  'CU'  TO #MAJOR-COMPONENT
MOVE *PROGRAM   TO  #MINOR-COMPONENT
**SAG END-EXIT
END-SUBROUTINE /* INITIALIZATIONS
```

ADDITIONAL-TRANSLATE-MAP

The code in this exit defines the processing performed to translate field headings and prompts on the generated map.

Note: If the generated module supports cursor translation, you must define this exit.

ADDITIONAL-TRANSLATE-TEXT

The code in this exit defines the processing performed to translate field headings and prompts in the local data area.

Note: If the generated module supports cursor translation, you must define this exit.

ADDITIONAL-TRANSLATIONS

This exit generates the framework for translation parameters that are not defined in the translation local data areas (LDAs) for the generated module.

Natural Construct translates prompts and headings whenever they are called within a module. This exit allows you to specify additional translations at the appropriate position within the module, depending on the operation mode (Modify mode, Generate mode, Batch mode, for example).

ADJUST-OBJECT-ID-IN-MSG

The code in this exit overrides the default name that refers to an object in messages (concatenation of object description and ID).

AFTER-BROWSE-CALLNAT

The code in this exit defines any special processing performed after calling the browse subprogram. Use this exit with the Object-Maint-Dialog model.

Tip: If you encounter an “unclaimed user exit” error message when generating a dialog that uses this user exit, ensure that you specified a browse subprogram name on the Additional Parameters panel for the Object-Maint-Dialog model.

AFTER-CALLNAT-SUBPROGRAMS

The code in this exit defines any additional processing performed after the Natural subprograms are called.

AFTER-ET-PROCESSING

The code in this exit defines any special processing performed after issuing an END TRANSACTION (ET) statement.

AFTER-GET

The code in this exit is executed after a new object is retrieved from the database. Use this exit to manipulate information in the object PDA, prior to returning the PDA to the calling module.

AFTER-GET-EDITS

The code in this exit converts or computes data after a GET operation moves the data from an entity record to an object. This exit must contain a series of subroutines named A-*entity-name*, where *entity-name* is a valid entity within the object. After an entity is retrieved from the database, the A-*entity-name* subroutine is performed once for each occurrence of the entity.

Since the values in the record buffer are copied to the object PDA using the MOVE BY NAME statement after the A-*entity-name* subroutine is performed, data manipulation in the subroutine should be made against the record buffer of the corresponding entity. Only manipulate directly against an object if the relevant fields have different names in the object PDA and the file.

Object-Maint-Subp Model

The Object-Maint-Subp model generates the #L2, #L3, and #L4 index variables for accessing the occurrences of the second, third, and fourth level entities within the object PDA. These indices must not be modified by the user exit code. However, the user exit code can access the object PDA fields by referencing the appropriate index tuples corresponding to the second, third, or fourth level entities (#L2, or #L2,#L3, or #L2,#L3,#L4, for example).

If these subroutines encounter invalid data within the object, they should assign values to the variables in the CDPDA-M parameter data area to terminate the process.

Example of using the subroutines in the AFTER-GET-EDITS user exit

The following INVOICE object contains values from the INVOICE-HEADER entity:

```
0010 DEFINE-EXIT AFTER-GET-EDITS
0020   DEFINE SUBROUTINE A-INVOICE-HEADER
0030    MOVE EDITED INVOICE-HEADER.INVOICE-DATE TO #DATE-VARIABLE
0040       (EM=YYYYMMDD)
0050    MOVE EDITED #DATE-VARIABLE (EM=LLL' 'DD','YY) TO
0060      INVOICE.INVOICE-DATE-EXTERNAL
0070   END-SUBROUTINE /* A-INVOICE-HEADER
0080 END-EXIT AFTER-GET-EDITS
```

AFTER-INIT

The code in this exit is executed immediately after the object PDA variables are reset. You can use this exit to assign default initial values to the object fields.

AFTER-INPUT

The code in this exit is executed immediately after each input panel is displayed and the standard keys and direct commands are processed (AT END OF PAGE section). You can use this exit to define validity edits for user-defined fields or to add non-standard PF-key processing to a module.

For example, when you add a non-standard PF-key, you should set the #SCROLLING variable to TRUE so the generated module does not trap the PF-key as invalid. After processing the non-standard key, include the PERFORM NEW-SCREEN code to return to the main panel (main INPUT statement) for the module.

Note: If you do not include the PERFORM NEW-SCREEN code and continue with normal execution after processing this exit, an Invalid PF-key error message is displayed.

Example of user exit code for the Browse models

```
0010 DEFINE EXIT AFTER-INPUT
0020 *
0030 * Processing to be performed immediately after the exit checks,
0040 * after input.
0050 IF NOT (#OPTION = ' ' OR = 'M' OR = 'S' OR = 'C') THEN
0060   REINPUT 'Valid options are "M", "S" or "C" or blank'
0070   MARK *#OPTION ALARM
0080 END-IF
0090 END-EXIT AFTER-INPUT
```

Example of user exit code for the Object-Maint-Dialog model

```
0010 DEFINE EXIT AFTER-INPUT
0020   /*
0030   /* Compute total for current product line
0040  COMPUTE ORDER.TOTAL-COST(#ARRAY1) = ORDER.QUANTITY(#ARRAY1) *
0050                                      ORDER.UNIT-COST(#ARRAY1)
0060 END-EXIT AFTER-INPUT
```

AFTER-LOOKUP-SUBROUTINES

The code in this exit defines all subroutines specified in the Perform subroutine field for the Object-Maint-Dialog model. Before each subroutine is executed, the #LOOKUP-STATUS variable is assigned the value Found, Not Found, or Null (depending on the result of the lookup).

If a subroutine is defined for an array field within the object PDA, you can access the current occurrence of the array using the following indexes:

Dimension of Array	Index
1	#I1
2	#I1, #I2
3	#I1, #I2, #I3

Example of user exit code for the Object-Maint-Dialog model

```
0010 DEFINE EXIT AFTER-LOOKUP-SUBROUTINES
0020 *
0030 ************************************************************
0040 DEFINE SUBROUTINE PROCESS-PRODUCT
0050 ************************************************************
0060 *
0070   DECIDE ON FIRST VALUE #LOOKUP-STATUS
0080    VALUE 'FOUND'
0090      ASSIGN ORDER.LINE-DESCRIPTION(#I1) =
0100              NCST-PRODUCT.PRODUCT-DESCRIPTION
0110      ASSIGN ORDER.UNIT-COST(#I1) =
0120             NCST-PRODUCT.PRODUCT-UNIT-COST
0130      COMPUTE ORDER.TOTAL-COST(#I1) = ORDER.QUANTITY(#I1) *
0140                           ORDER.UNIT-COST(#I1)
0150    VALUE 'NOT FOUND'
0160    COMPRESS 'Product' ORDER.ORDER-PRODUCT-ID(#I1) 'not found'
0170             TO MSG-INFO.##MSG
0180      RESET ORDER.LINE-DESCRIPTION(#I1) ORDER.QUANTITY(#I1)
0190            ORDER.UNIT-COST(#I1) ORDER.TOTAL-COST(#I1)
0200     VALUE 'NULL'
0210       RESET ORDER.NCST-ORDER-HAS-LINES(#I1)
0220     NONE IGNORE
0230   END-DECIDE
0240 END-SUBROUTINE
0250 END-EXIT AFTER-LOOKUP-SUBROUTINES
```

AFTER-OBJECT-CALL

The code in this exit is executed after the object subprogram is called. It allows extra processing upon returning from the object subprogram (for example, you can override the MSG-INFO.##ERROR-FIELD or MSG-INFO.##MSG variables).

AFTER-PROCESS-ACTIONS

The code in this exit defines the processing performed after actions are processed.

AFTER-READ

The code in this exit is executed immediately following the retrieval of a view occurrence (after a READ statement). You can use this exit to join multiple files and maintain a view containing more than one secondary file, for example.

Note: The REJECT-AFTER-MAX-KEY-CHECK user exit is similar to this exit, except it is generated after a minimum or maximum key value is rejected. For information, see **REJECT-AFTER-MAX-KEY-CHECK**, page 25.

AFTER-ROW-ASSIGNMENT

The code in this exit is executed immediately after the view contents have been copied to the output object (row) PDA. You can alter the contents of the PDA passed back to the caller by using #ROW-INDEX to locate an item in the PDA.

Example of user exit code for the Object-Browse-Subp model

```
0140 DEFINE EXIT AFTER-ROW-ASSIGNMENT
0150 **
0160 ** Compute the derived column ANNUAL-VACATION-DAYS for current row
0170 COMPUTE EMPROW.ANNUAL-VACATION-DAYS(#ROW-INDEX)=
0180   EMPROW.LEAVE-TAKEN(#ROW-INDEX)+EMPROW.LEAVE-DUE(#ROW-INDEX)
0190 END-EXIT
```

Note: Use the #ROW-INDEX variable to reference the last assigned occurrence of the object row PDA within the AFTER-ROW-ASSIGNMENT user exit.

AFTER-SCREEN-CLEAR

The code in this exit is executed immediately after the CLEAR subroutine is performed. You can use this exit to reset the derived fields assigned when the AFTER-READ or BEFORE-INPUT user exit subroutines are performed.

ASSIGN-PREFIX-VALUE

The code in this exit assigns the value for a browse prefix.

BEFORE-BROWSE-CALLNAT

The code in this exit defines any special processing performed before calling the browse subprogram. Use this exit with the Object-Maint-Dialog model.

Tip: If you encounter an "unclaimed user exit" error message when generating a dialog that uses this user exit, ensure that you specified a browse subprogram name on the Additional Parameters panel for the Object-Maint-Dialog model.

BEFORE-CALLNAT-SUBPROGRAMS

The code in this exit defines any additional processing performed before the Natural subprograms are called.

BEFORE-CHECK-ERROR

The code in this exit defines the processing performed when an error condition is encountered within a generated module. Because an error condition will bypass the END-OF-PROGRAM user exit, use this exit if processing is required before leaving the program when an error condition occurs.

Example of code in the BEFORE-CHECK-ERROR user exit

```
1320 **SAG DEFINE EXIT BEFORE-CHECK-ERROR
1330 *
1340 * Use this user exit for specific error checking
1350   IF CSASTD.RETURN-CODE = CSLRCODE.#INTERRUPT(*)
1360     ASSIGN CU--PDA.#PDA-PHASE = #SAVE-PHASE
1370   END-IF
1380  **SAG END-EXIT
```

BEFORE-CHECK-PFKEYS

The code in this exit defines any additional processing performed before the PF-keys are checked.

BEFORE-CONFIRMATION

The code in this exit defines the processing performed before invoking the INPUT statement to confirm termination of the program.

BEFORE-ET

The code in this exit is executed before the END OF TRANSACTION statement is processed. You can use this exit to perform any integrity checking that cannot be done until the END OF TRANSACTION statement is processed.

BEFORE-ET-PROCESSING

The code in this exit defines any special processing performed before issuing an END TRANSACTION (ET) statement.

BEFORE-FETCH

The code in this exit is executed before the main menu program is FETCHed.

BEFORE-INPUT

The code in this exit is executed immediately before the INPUT statement is processed in the AT END OF PAGE section. You can use this exit to lookup a code table (to display a description, as well as a code value), to issue SET CONTROL statements, or to capture or default map variables prior to displaying each panel.

Example of user exit code for the Browse-Select-Subp model

```
0010 DEFINE EXIT BEFORE-INPUT
0020 *
0030 * Processing to be performed before the INPUT statement.
0040 * Change standard message to indicate selection can be done ONLY
0050 * by positioning the cursor (not entering key value since input is
0060 * protected).
0070   ASSIGN MSG-INFO.##MSG = 'Position cursor to select.'
0080 END-EXIT BEFORE-INPUT
```

Example of user exit code for the Menu model

```
0010 DEFINE EXIT BEFORE-INPUT
0020 *
0030 * Processing to be performed before each INPUT statement.
0040   SET CONTROL 'WB'   /* Restore window size to physical screen size.
0050 END-EXIT BEFORE-INPUT
```

Example of user exit code for the Object-Maint-Dialog model

```
0010 DEFINE EXIT BEFORE-INPUT
0020 *
0030 * If order lines were scrolled, set distributions array to 1
0040   IF #LAST-ARRAY1 NE #ARRAY1 THEN
0050     ASSIGN #ARRAY2 = #NEXT-ARRAY2 = #CURR-INDEX(#PANEL,2) = 1
0060   END-IF
0070   ASSIGN #LAST-ARRAY1 = #ARRAY1
0080   /*
0090   /* Update total for the order
0100   COMPUTE ORDER.ORDER-AMOUNT = 0 + ORDER.TOTAL-COST(*)
0110 END-EXIT BEFORE-INPUT
```

BEFORE-OBJECT-CALL

The code in this exit defines the processing performed before an object is called.

BEFORE-PROCESS-ACTIONS

The code in this exit defines the processing performed before actions are processed.

BEFORE-PROCESSING-MENU-CODES

The code in this exit is executed before a menu code value is processed.

BEFORE-RESUMING-PROCESSING

The code in this exit is performed when a user cancels a quit (termination) request.

BEFORE-ROW-ASSIGNMENT

The code in this exit is executed immediately prior to copying the file view to the object (Row) PDA. To reject this record, execute an ESCAPE ROUTINE statement.

Example of user exit code for the Object-Browse-Subp model

```
0010 DEFINE EXIT BEFORE-ROW-ASSIGNMENT
0020 *
0030 * Don't return employee records if they have zero in the leave-taken
0040 * and leave due fields.
0050 IF EMPLOYEES.LEAVE-TAKEN = 0 AND EMPLOYEES.LEAVE-DUE = 0 THEN
0060   ESCAPE ROUTINE
0070 END-IF
0080 **
0090 ** Reclassify divorced and widowed people as single
0100 IF EMPLOYEES.MARITAL-STATUS = 'D' OR = 'W' THEN
0110   ASSIGN EMPLOYEES.MARITAL-STATUS = 'S'
0120 END-IF
0130 END-EXIT
```

Note: Avoid rejecting a significant percentage of records, or a large number of consecutive records, by executing the ESCAPE ROUTINE statement within the BEFORE-ROW-ASSIGNMENT user exit, due to the negative impact this will have on performance.

BEFORE-STANDARD-KEY-CHECK

The code in this user exit checks any additional PF-keys defined for the modify subprogram, or prepares for standard PF-key validations.

Example of code in the BEFORE-STANDARD-KEY-CHECK user exit

```
DEFINE EXIT BEFORE-STANDARD-KEY-CHECK
*
* Use this user exit to check additional PF-keys or prepare for the
* standard PF-key check.
END-EXIT BEFORE-STANDARD-KEY-CHECK
```

BROWSE-ACTION-PROCESSING

The code in this exit specifies additional processing performed for the Browse action (in addition to the standard Natural Construct-generated processing).

Tip: If you encounter an “unclaimed user exit” error message when generating a dialog that uses this user exit, ensure that you specified a browse subprogram name on the Additional Parameters panel for the Object-Maint-Dialog model.

BUILD-REPORT-LOCAL-VARS

This user exit contains control variables or indices (indexes). It is generated in conjunction with the WRITE-FIELDS user exit to write arrays (or MUs/PEs) to a report.

CHANGE-HISTORY

This user exit keeps a record of changes to the generated module. It generates comment lines indicating the date, the user ID of the user who created or modified the module, and a description of any change.

Example of code in the CHANGE-HISTORY user exit

```
DEFINE EXIT CHANGE-HISTORY
* Changed on Aug 27,02 by DEVPR for release ____
* >
* >
* >
END-EXIT CHANGE-HISTORY
```

CLEAR-ACTION-PROCESSING

The code in this exit specifies additional processing performed for the Clear action (in addition to the standard Natural Construct-generated processing).

COPY-ACTION-PROCESSING

The code in this exit specifies additional processing performed for the Copy action (in addition to the standard Natural Construct-generated processing).

DEFINE-REPORT-PRINTER

The code in this exit defines the name or pathname for the printer to which reports are routed.

DEFINE-TRANSLATION-HEADERS

The code in this exit defines the panel headings used by the generated module to support cursor translation.

Note: If the generated module supports cursor translation, you must define this exit.

DELETE-EDITS

The code in this exit validates the contents of an object record before the record is deleted. This exit must contain a series of subroutines named D-*entity-name*, where *entity-name* is a valid entity within the object. The subroutine is performed once for each occurrence of the corresponding entity, immediately before the entity is deleted.

Because object values are copied to the corresponding file view using the MOVE BY NAME statement before the D subroutines are performed, data validation in the D subroutine should be made against the record buffer of the file view. Only refer directly to the object when the relevant fields have different names in the object PDA and the file.

Object-Maint-Subp Model

The Object-Maint-Subp model generates the #L2, #L3, and #L4 index variables to access the occurrences of the second, third, and fourth level entities within the object PDA. These indices must not be modified by the user exit code. However, the user exit code can access the object PDA fields by referencing the appropriate index tuples that correspond to the second, third, or fourth level entities (#L2, or #L2,#L3, or #L2,#L3,#L4). If subroutines encounter invalid data within an object, they should assign values to the variables in the CDPDA-M parameter data area to terminate the process.

Example of using the subroutines in the DELETE-EDITS user exit

```
0010 DEFINE-EXIT DELETE-EDITS
0020           DEFINE SUBROUTINE D-INS-POLICY
0030             IF INS-POLICY.EXPIRY-DATE GE *DATX THEN
0040               ASSIGN MSG-INFO.##MSG = 'Cannot delete active policy'
0050                ASSIGN MSG-INFO.##ERROR-FIELD = 'EXPIRY-DATE'
0060                 ESCAPE ROUTINE
0070             END-IF
0080           END-SUBROUTINE
0090 END-EXIT DELETE-EDITS
```

Note: The subroutines in this exit should assign the name of the object PDA field in error to the MSG-INFO.##ERROR-FIELD variable and a message to the MSG-INFO.##MSG or MSG-INFO.##MSG-NR variable.

DISPLAY-ACTION-PROCESSING

The code in this exit specifies additional processing performed for the Display action (in addition to the standard Natural Construct-generated processing).

END-OF-PROGRAM

The code in this exit is executed once before the module is terminated. You can use this exit for any cleanup required (such as assigning a termination message or resetting windows, for example) before exiting the module.

If an error condition occurs, this user exit will not be executed. Use the BEFORE-CHECK-ERROR user exit if processing is required before leaving the program.

Tip: You can assign the current key value to a global variable in this exit, so it is carried into other modules that use the same key.

Example of user exit code for the Object-Maint-Subp model

```
0010 DEFINE EXIT END-OF-PROGRAM
0020 FOR #I = 1 TO 3
0030 * Strip Ncst off of file name references in messages.
0040  IF MSG-INFO.##MSG-DATA(#I) = MASK('Ncst ') THEN
0050   RESET MSG-INFO.##MSG-DATA-CHAR(#I,1:4)
0060   MOVE LEFT MSG-INFO.##MSG-DATA(#I) TO MSG-INFO.##MSG-DATA(#I)
0070  END-IF
0080 END-FOR
0090 END-EXIT END-OF-PROGRAM
```

EXPORT-COLUMN-HEADERS

The code in this exit defines the processing performed to export selected column headings to a PC file.

EXPORT-DATA

The code in this exit defines parameters for export to a work file. The export parameters can be the same or different from the browse parameters.

If you mark the Export data support field and do not select this exit, only the primary key data is available for export. If you mark the field and select this exit, a series of windows is displayed to select the fields for export.

Note: If you are using Entire Connection, you can export the work file directly to a PC text file, which you can then import into any PC spreadsheet.

EXPORT-DATA-FIELDS

The code in this exit defines the processing performed to export selected fields to a PC file.

EXTENDED-RI-CHECKS

The code in this exit performs further validations after a referential integrity check. For example, after relationship checking verifies that a record exists for the customer number specified in the order header, you can check that the customer is flagged as active and is in good credit standing. To verify these concerns, this exit must contain a V-*relationship-name* subroutine, where *relationship-name* is a valid relationship used by an object to check for referential integrity.

This exit contains the V-*relationship-name* validation routine, which is executed during the pre-editing phase after the Predict automatic rules are checked and after the V1-*entity-name* subroutine for the current entity has been executed.

The V-*relationship-name* subroutine is invoked from the appropriate E-*entity-name* routine or, in the case of the root entity, from the EDIT-OBJECT subroutine.

Note: For more information on the *V-relationship-name* subroutine, see *Natural Construct Generation*.

EXTENDED-RI-VIEWS

This user exit contains the views required by the validation routines defined in the EXTENDED-RI-CHECKS user exit. It is generated automatically when you select the EXTENDED-RI-CHECKS user exit and specify fields to be included in the referential integrity check views.

EXTEND-SELECTION-TABLE

Use this exit to assign user-defined values to the selection table for the Browse-Select models.

FINAL-PROCESSING

The code in this exit defines the processing performed immediately before leaving the module.

FORMER-ACTION-PROCESSING

The code in this exit specifies additional processing performed for the Former action (in addition to the standard Natural Construct-generated processing).

HARDCOPY-EDITS

The code in this exit is executed after the INPUT statement is executed for the hardcopy screen options. You can use this exit to specify edit checks on the values specified for hardcopy device, lines per page, and page size.

HARDCOPY-TERMINATING-PROCESS

The code in this exit is executed after selected records are printed. You can use this exit to specify any processing required when terminating the hardcopy function.

HE-PARAMETER-INDEXES

The code in this exit defines additional parameters to receive indices for a multi-dimensional key (passed parameter). For example, if you are generating a browse helproutine or subprogram that is attached to an array field and you need to know the occurrence for which help was requested, you can define the index values for each dimension within this exit.

Example of user exit code for the Browse-Select model

```
0010 DEFINE EXIT HE-PARAMETER-INDEXES
0020 01  #HE-FLD-INDEX1 (I2)     /* Index for first dimension
0030   02  #HE-FLD-INDEX2 (I2)   /* Index for second dimension
0040   02  #HE-FLD-INDEX3 (I2)   /* Index for third dimension
0050 END-EXIT HE-PARAMETER-INDEXES
```

INPUT-KEY

The code in this exit defines the input key used by a generated module to INPUT selected fields.

LOCAL-DATA

The code in this exit defines additional local variables that are used in conjunction with other user exits.

Example of user exit code for the LOCAL-DATA user exit

```
0010 DEFINE EXIT LOCAL-DATA
0020   LOCAL
0030   01 #CITY-PROVINCE(A50)
0040   01 NCST-CUSTOMER VIEW OF NCST-CUSTOMER
0050     02 CUSTOMER-NUMBER
0060     02 BUSINESS-NAME
0070     02 PHONE-NUMBER
0080     02 SHIPPING-ADDRESS
0090       03 S-STREET
0100       03 S-CITY
0110       03 S-PROVINCE
0120       03 S-POSTAL-CODE
0130     02 CONTACT
0140     02 CREDIT-RATING
0150     02 CREDIT-LIMIT
0160 END-EXIT LOCAL-DATA
```

Example of user exit code for the Browse-Select model

```
0010 DEFINE EXIT LOCAL-DATA
0020   01 NCSTDB2-CUSTOMER-PROGRAM-VIEW VIEW OF NCSTDB2-CUSTOMER
0030   02 CUSTOMER_NUMBER
0040   02 BUSINESS_NAME
0050   02 PHONE_NUMBER
0060   02 M_STREET
0070   02 M_CITY
0075   02 N@M_CITY
0080   02 REDEFINE N@M_CITY
0090       03 FILLER-90(A1)
0100       03 N#M_CITY(L)
0110   02 M_PROVINCE
0120   02 M_POSTAL CODE
0130   02 S_STREET
0140   02 S_CITY
0150   02 S_PROVINCE
0160   02 S_POSTAL CODE
0170   02 CONTACT
0180   02 PROVINCE
0190   02 M_CITY
0200   02 N@M_CITY
0210   02 REDEFINE N@M_CITY
0220       03 FILLER-90(A1)
0230       03 N#M_CITY(L)
0240   02 M_PROVINCE
0250   02 M_POSTAL CODE
0260   02 S_STREET
0270   02 S_CITY
0280   02 S_PROVINCE
0290   02 S_POSTAL CODE
0300   02 CONTACT
0310   02 CREDIT_RATING
0320   02 CREDIT_LIMIT
0330   02 DISCOUNT_PERCENTAG
0340   02 CUSTOMER_WAREHOUSE
0350   02 LOG_COUNTER
0360 END-EXIT LOCAL-DATA
```

MISCELLANEOUS-SUBROUTINES

The code in this exit defines any subroutines invoked within your user exit code. It is placed immediately before the END statement in the generated module.

Example of code in the MISCELLANEOUS-SUBROUTINES user exit

```
DEFINE EXIT MISCELLANEOUS-SUBROUTINES
**
************************************************************************
DEFINE SUBROUTINE some-subroutine
************************************************************************
**
  ESCAPE ROUTINE IMMEDIATE
END-SUBROUTINE /* some-subroutine
END-EXIT MISCELLANEOUS-SUBROUTINES
```

MODIFY-ACTION-PROCESSING

The code in this exit specifies additional processing performed for the Modify action (in addition to the standard Natural Construct-generated processing).

NEXT-ACTION-PROCESSING

The code in this exit specifies additional processing performed for the Next action (in addition to the standard Natural Construct-generated processing).

PARAMETER-DATA

This user exit defines any additional parameters used in conjunction with other programs.

Example of code in the PARAMETER-DATA user exit

```
DEFINE EXIT PARAMETER-DATA
** PARAMETER USING PDAname
** PARAMETER
**  01 #Additional-parameter1
**  01 #Additional-parameter2
END-EXIT PARAMETER-DATA
```

PROCESS-SELECTED-RECORD

Browse Models

The code in this exit is performed when a record is selected (either by using the cursor or by entering a key value for the record while the record is displayed). You can use this exit to specify processing performed on the selected record.

The #SELECTED-KEY variable contains the key value for the selected record. The #SELECTED-ISN variable contains the ISNs (Internal Sequence Numbers) of the selected record (for Adabas files/user views). In non-Adabas files, the #SELECTED-UQ variable contains the value of the unique field for the selected record. The #SELECTED-UQ variable is generated only if a field is marked as unique in Predict. For more information, see *Natural Construct Generation*.

Example of user exit code for the Browse-Subp model

```
0010 DEFINE EXIT PROCESS-SELECTED-RECORD
0020 **
0030 ** Processing to be performed after a record has been
0040 ** selected.
0050 IF *PF-KEY = 'ENTR' THEN
0060   ASSIGN #PDA-KEY = #SELECTED-KEY    /* Return selected value.
0070 ** Action field is used if this is a maintenance browse
0080 ** subprogram.
0090   ASSIGN CDSELPDA.SELECTED-FUNCTION = 'D'
0100 ** Display the selected record.
0110   ESCAPE BOTTOM(PROG.) IMMEDIATE
0120 END-IF
0130 END-EXIT PROCESS-SELECTED-RECORD
```

Browse-Select Models

The code in this exit is performed once for each record marked with a valid action. If any actions are selected, this user exit is required. Define the processing for a selected record based on the action code.

The #ACTION variable contains the action code entered by a user. The #SELECTED-KEY variable contains the key value for the selected record.

For Adabas files/user views, the #SELECTED-ISN variable contains the ISNs (Internal Sequence Numbers) for the selected record. For non-Adabas files, the #SELECTED-UQ variable contains the value of the unique field for the selected record. The #SELECTED-UQ variable is only generated if a field is marked as unique in Predict. For more information, see *Natural Construct Generation.*

Example of user exit code for the Browse-Select model

```
0010 DEFINE EXIT PROCESS-SELECTED-RECORD
0020 * This user exit is invoked once for each record that is
0030 * marked with a valid action.
0040 * The following variables are set prior to invoking this
0050   CALLNAT 'NCOSELN'        /* CALLNAT the object subprogram to
0060            #SELECTED-KEY   /* process a record. The record's
0070            #ACTION         /* key is stored in #SELECTED-KEY.
0080            DIALOG-INFO     /* Apply action indicated in
0090            MSG-INFO        /* Action field.
0100            PASS
0110 END-EXIT PROCESS-SELECTED-RECORD
```

Example of user exit code for the Browse-Select-Subp model

```
0010 DEFINE EXIT PROCESS-SELECTED-RECORD
0020 *
0030 * Processing to be performed after a record has been selected.
0040 IF *PF-KEY = 'ENTR' THEN
0050   GET ORDER-DETAIL #SELECTED-ISN
0060   RESET MSG-INFO
0070   CALLNAT 'NCOSODET'
0080           ORDER-DETAIL.ORDER-NUMBER
0090           CDSELPDA.SELECTED-FUNCTION
0100           DIALOG-INFO
0110           MSG-INFO
0120           PASS
0130 END-IF
0140 END-EXIT PROCESS-SELECTED-RECORD
```

PURGE-ACTION-PROCESSING

The code in this exit specifies additional processing performed for the Purge action (in addition to the standard Natural Construct-generated processing).

REINPUT-SCREEN

If #ERROR is set to TRUE, the code in this exit is executed automatically after the UPDATE-EDITS user exit is executed and the maps and scrolling lines are repositioned. For more information, see **UPDATE-EDITS**, page 29.

If the UPDATE-EDITS positioning technique is used, you can use the code in this exit to test the value of #ERROR-NR and issue the corresponding REINPUT (using the MARK field-name safely, since the positioning mechanism has guaranteed that the current map displays the field in error).

Object-Maint-Dialog Model

If an additional field is on a map that is not part of the object, include a REINPUT statement for edit checks on this field. The actual edit checks are included in another user exit, depending on the type of edit check being performed. (User exits that might include the actual edit checks include the AFTER-INPUT, AFTER-GET, and ACTION-PROCESSING user exits, such as ADD-ACTION-PROCESSING, BROWSE-ACTION-PROCESSING, etc.)

REJECT-AFTER-MAX-KEY-CHECK

The code in this exit defines processing performed after the minimum or maximum key value is rejected in a browse or browse-select program. It provides more efficient code when processing minimum and maximum key values.

Note: The AFTER-READ user exit is similar to this exit, except it is generated before the minimum or maximum key value is checked. For information, see **AFTER-READ**, page 10.

REPORT-COLUMN-HEADERS

The code in this exit defines the column headings displayed on a printed report.

REPORT-DATA-FIELDS

The code in this exit defines the processing performed to route selected fields to a printer.

REPORT-HEADERS

The code in this exit defines the field headings displayed on a printed report.

SCREEN-HEADERS

The code in this exit defines the panel headings displayed on a generated panel.

SELECT-ADDITIONAL-ACTIONS

The code in this exit defines any additional action codes to appear as valid actions in the actions list. You must add the action code to the CDACTA parameter data area and to the #CODES table in CDACT to be valid.

Note: For information on adding actions, see *Natural Construct Generation*.

The code in this exit has the following format:

```
ASSIGN #valid-action(*) = 'X'
```

where *valid-action* is defined in CDACTA and CDACT.

This user exit works with the ADDITIONAL-ACTIONS-PROCESSING user exit. For information, see **ADDITIONAL-ACTIONS-PROCESSING**, page 5.

SELECT-STATEMENT

The code in this exit generates a SELECT statement for SQL files. Select statements are used by the Browse models. Although the Browse models generate a generic SELECT statement for all SQL-defined files, this may not be the most efficient statement for a particular SQL file. For example, a DB2-defined file may require a certain SELECT syntax, whereas an Oracle-defined file may require a different syntax. Either let Natural Construct generate the generic SELECT statement as before (within the generated code), or generate it within the SELECT-STATEMENT user exit (where statement modifications can be made).

Example of user exit code for the SELECT-STATEMENT user exit

```
0010 DEFINE EXIT SELECT-STATEMENT
0020     SELECT *
0030       INTO VIEW NCSTDB2-CUSTOMER
0040       FROM NCSTDB2-CUSTOMER
0050       WHERE (CUSTOMER_NUMBER >= #START.CUSTOMER_NUMBER)
0060       ORDER BY CUSTOMER_NUMBER
0070 END-EXIT
```

SET-PF-KEYS

The code in this exit is executed before the PF-keys are set and allows the addition of non-standard PF-keys to a browse program. Define the additional PF-keys in the CD-KEYLDA local data area.

Note: For information on adding PF-keys, see *Natural Construct Generation.*

Example of user exit code to add a non-standard PF-key

```
0010 DEFINE EXIT SET-PF-KEYS
0020 /*
0030 /* set non-standard PF-key
0040 RESET INITIAL CDKEYLDA.#YOUR-KEY
0050 END-EXIT SET-PF-KEYS
```

SPECIAL-CODE-PROCESSING

The code in this exit defines the processing performed for each menu code (if entering a code on the generated menu does not FETCH a program).

START-OF-PROGRAM

Code in this exit is executed once at the beginning of the generated module after all standard initial values are assigned. You can use this exit to do any initial set-up required, such as initializing input values from globals, setting window or page sizes, or capturing security information for the restricted data area.

Tip: Use this user exit to set or reset variables or call routines that must be executed before the normal processing in a subprogram proxy.

Browse and Browse-Select Models

For the Browse and Browse-Select series of models, you can use this exit to indicate whether a browse module reads files in ascending (from 1 to 999999 or A to Z), descending (from 999999 to 1 or Z to A), or user-defined sequence. The user-defined sequence allows users to read files in either sequence as desired.

Note: If you do not include the START-OF-PROGRAM user exit, the default is ascending sequence.

For an example of using this exit, refer to the NCOSEL module in the Natural Construct demo system.

When you mark this user exit and press Enter, the following code is displayed:

```
DEFINE EXIT START-OF-PROGRAM
*
* Processing to be performed once at the start of the program.
* Adjust the #SEQUENCE variable to get ascending, descending sequence.
* For example, uncomment these lines:
* MOVE 'D'    TO #SEQUENCE
* MOVE (AD=I) TO #SEQUENCE-CV
END-EXIT
```

START-OF-PROGRAM User Exit for a Browse Module

Modify this code as follows:

Sequence:	**Action:**
Ascending only	Do not include this user exit or modify the code.
Descending only	Remove the comment indicator from line 0060: `MOVE 'D' TO #SEQUENCE`
Ascending by default, but can be changed by the user	Remove the comment indicator from line 0070: `MOVE (AD=I) TO #SEQUENCE-CV`
Descending by default, but can be changed by the user	Remove the comment indicators from line 0060 and line 0070.

TOP-OF-PAGE

The code in this exit is executed whenever a TOP OF PAGE condition occurs. You can use this user exit to print panel and column headings, for example.

TRANSLATE-COLUMN-HEADERS

The code in this exit defines the processing performed to allow cursor translation of the column headings.

Note: If the generated module supports cursor translation, you must define this exit.

TRANSLATE-INPUT-KEY

The code in this exit defines the input key used by the generated module to support cursor translation.

Note: If the generated module supports cursor translation, you must define this exit.

TRANSLATE-SCREEN-HEADERS

The code in this exit defines the processing performed to support cursor translation of the panel headings.

Note: If the generated module supports cursor translation, you must define this exit.

UPDATE-EDITS

The code in this exit performs edit checks before adding, updating, or purging a record. The exit contains validation subroutines that execute edit checks at different points during the processing of an entity. You can create subroutines for each entity in an object.

The following table describes the validation subroutines available within this exit:

Subroutine	Description
V0-*entity-name*	Executed during the pre-editing phase before the Predict automatic rules are checked and the children of the current entity are processed.
V1-*entity-name*	Executed during the pre-editing phase after the Predict automatic rules are checked and before the children of the current entity are processed.
V2-*entity-name*	Executed during the post-editing phase after the Predict automatic rules are checked and all children of the current entity are processed.

Natural Construct assumes that primary fields are always on the current panel (on each map) and that secondary fields may require repositioning before displaying a message and marking the field in error.

Normally, a REINPUT statement displays an error message. However, when multiple input maps are used, an error may be detected when the field in error is not on the current panel. Issuing a REINPUT statement may cause the error message to be displayed for that field. To avoid this, assign a number to #ERROR-NR that represents the error, assign #PANEL to the desired map number (with 1 being the leftmost), assign #ERROR=TRUE, and perform the NEW-SCREEN subroutine. The generated module repositions the maps and invokes the REINPUT-SCREEN user exit (which can issue the error message).

Maint Model

If a maintenance program uses a secondary file or has multiple-valued fields (MUs) or periodic groups (PEs), you can reposition the scrolling area on a panel to a particular set of occurrences. To do this, you assign a value to #LINE before performing the NEW-SCREEN subroutine. In addition to repositioning the maps, the program repositions the scrolling area so the desired occurrence is displayed.

Example of user exit code for the Maint model

```
0010 DEFINE EXIT UPDATE-EDITS
0020 *
0030 IF #ACTION = #PURGE(*) THEN
0040   /*
0050   /* Perform purge edits.
0060   ESCAPE ROUTINE IMMEDIATE /* Skip Add/Modify edits
0070 END-IF
0080 *
0090 * Add/Modify edits
0100 DECIDE FOR FIRST CONDITION
0110   WHEN NCST-PRODUCT.PRODUCT-DESCRIPTION EQ ' '
0120     REINPUT 'Description is required'
0130       MARK *NCST-PRODUCT.PRODUCT-DESCRIPTION ALARM
0140   WHEN NCST-PRODUCT.PRODUCT-REORDER-POINT EQ 0
0150     REINPUT 'Reorder Point is required'
0160       MARK *NCST-PRODUCT.PRODUCT-REORDER-POINT ALARM
0170   WHEN NONE IGNORE
0180 END-DECIDE
0190 END-EXIT UPDATE-EDITS
```

Object-Maint Models

The code in this exit validates or manipulates the contents of an object. Before an entity is updated or stored, the subroutines in this exit are performed once, in the following order, for each occurrence of an entity:

1 The V0-*entity-name* subroutine is performed before the Predict rules associated with the fields of the current entity are processed. It retrieves data for validation. If an error occurs, this subroutine terminates the process immediately.

2 The V1-*entity-name* subroutine is performed after the Predict rules are processed. If no Predict rules are processed, you require either a V0-entity-name or V1-entity-name subroutine.

3 The V2-*entity-name* subroutine is performed after all child records are processed. This subroutine is required when child attributes affect the parent attributes. For example, each child record contributes an amount to the total value of the parent record (this contribution can be done in the V2-*child-entity-name* subroutine). After all child records are processed, the V2-*entity-name* subroutine of the current entity is invoked to validate the total value.

Because of the pre-order and post-order execution of the V0, V1, and V2 subroutines, the entities at the lowest level (entities without a child) do not need a V2 subroutine. They contribute to the parent through their V0 and V1 subroutines. At each node in the object hierarchy tree, the record buffers of the current entity and its parent (and ancestors) are available. Thus, the data validation or manipulation in the V0, V1, or V2 subroutine should be made against the record buffers for the corresponding views.

Prior to performing the V*n* subroutines, the object field values are copied to the corresponding file view using the MOVE BY NAME statement. Only refer directly to the object when the relevant fields have different names in the object PDA and file.

The Object-Maint-Subp model generates the #L2, #L3, and #L4 index variables to access the occurrences of the second, third, and fourth level entities within the object PDA. These indices must not be modified by the user exit code. However, user exit code can access the object PDA fields by referencing the appropriate index tuples corresponding to the second, third, or fourth level entities (for example, #L2, or #L2,#L3, or #L2,#L3,#L4).

Note: The subroutines in the UPDATE-EDITS user exit should assign the name of the object PDA field in error to the MSG-INFO.##ERROR-FIELD variable and a message to the MSG-INFO.##MSG or MSG-INFO.##MSG-NR variable (defined in the CDPDA-M parameter data area).

Example of using the subroutines in the UPDATE-EDITS user exit

Consider the Invoice object. It is created from INVOICE-HEADER, which has many INVOICE-GROUPs, each of which has many INVOICE-LINE records. The three files contain the respective fields: HEADER-AMOUNT, GROUP-AMOUNT, and LINE-AMOUNT. You can implement the amount accumulation from the bottom level to the top level using the subroutines on the following page.

```
0010 DEFINE-EXIT UPDATE-EDITS
0020   DEFINE SUBROUTINE V-INVOICE-HEADER
0030     IF INVOICE-HEADER.INVOICE-AUTHORIZED = 'N'
0040       ASSIGN MSG-INFO.##MSG = 'Invoice is not authorized'
0050       ASSIGN MSG-INFO.##ERROR-FIELD = 'INVOICE-AUTHORIZED'
0060       ESCAPE ROUTINE
0070     END-IF
0080     /*
0090     /* Reset the total amount before going to the invoice groups
0100     /* and lines.
0110     RESET INVOICE-HEADER.HEADER-AMOUNT
0120     /*
0130     /* Convert the displayed date of the object to the internal
0140    /* date to be stored in the file. Reference to the object field
0150    /* is required because the date field has different name on the
0160     /* object and the file
0170     MOVE EDITED INVOICE.INVOICE-DATE-EXTERNAL TO #DATE-VARIABLE
0180       (EM=LLL' 'DD','YY)
0190     MOVE EDITED #DATE-VARIABLE (EM=YYYYMMDD) TO
0200       INVOICE-HEADER.INVOICE-DATE
0210   END-SUBROUTINE /* V-INVOICE-HEADER
0220   /*
0230   DEFINE SUBROUTINE V2-INVOICE-HEADER
0240     /*
0250     /* Validate the total amount accumulated from the groups
0260     IF INVOICE-HEADER.HEADER-AMOUNT <= 0
0270       ASSIGN MSG-INFO.##MSG = 'Invalid total invoice amount::1:'
0280      ASSIGN MSG-INFO.##MSG-DATA(1) = INVOICE-HEADER.HEADER-AMOUNT
0290       ESCAPE ROUTINE
0300     END-IF
0310   END-SUBROUTINE /* V2-INVOICE-HEADER
0320   /*
0330   DEFINE SUBROUTINE V-INVOICE-GROUP
0340     /*
0350     /* Reset the group amount before going to the lines.
0360     RESET INVOICE-GROUP.GROUP-AMOUNT
0370   END-SUBROUTINE /* V-INVOICE-GROUP
0380   DEFINE SUBROUTINE V2-INVOICE-GROUP
0390     /*
0400     /* Validate the group amount accumulated from the lines
0410     IF INVOICE-GROUP.GROUP-AMOUNT <= 0
0420       ASSIGN MSG-INFO.##MSG = 'Invalid group amount::1:'
0430       ASSIGN MSG-INFO.##MSG-DATA(1) = INVOICE-GROUP.GROUP-AMOUNT
0440       ASSIGN MSG-INFO.##ERROR-FIELD = 'GROUP-AMOUNT'
0450       ESCAPE ROUTINE
0460     END-IF
0470     /*
0480     /* Accumulate the group amount to the total amount
0490   ADD INVOICE-GROUP.GROUP-AMOUNT TO INVOICE-HEADER.HEADER-AMOUNT
0500   END-SUBROUTINE /* V2-INVOICE-GROUP
0510   /*
0520   DEFINE SUBROUTINE V-INVOICE-LINE
0530     /*
0540     /* Accumulate the line amount to the group amount
0550     ADD INVOICE-LINE.LINE-AMOUNT TO
0560       INVOICE-GROUP.GROUP-AMOUNT
0570   END-SUBROUTINE /* V-INVOICE-LINE
0580 END-EXIT UPDATE-EDITS
```

Example of user exit code for the Object-Maint-Subp model

```
0130 DEFINE EXIT UPDATE-EDITS
0140 ************************************************************
0150 DEFINE SUBROUTINE V0-NCST-ORDER-HEADER
0160 ************************************************************
0170 *
0180   IF NCST-ORDER-HEADER.INVOICE-NUMBER = 0
0190     ASSIGN MSG-INFO.##MSG = 'Invoice number is required.'
0200     ASSIGN MSG-INFO.##ERROR-FIELD = 'INVOICE-NUMBER'
0210     ESCAPE ROUTINE
0220   END-IF
0230   /* Initialize order date when creating a new order
0240   IF CDAOBJ.#FUNCTION = 'STORE'
0250     /* Update date in file
0260     ASSIGN NCST-ORDER-HEADER.ORDER-DATE = *DATN
0270   END-IF
0280   /*
0290   /* Reset total order amount before going to the order lines
0300   /* to obtain their contribution.
0310   RESET NCST-ORDER-HEADER.ORDER-AMOUNT
0320 END-SUBROUTINE
0330 *
0340 ************************************************************
0350 DEFINE SUBROUTINE V2-NCST-ORDER-HEADER
0360 ************************************************************
0370 *
0380 * Validate total order amount accumulated from the order lines
0390   IF NCST-ORDER-HEADER.ORDER-AMOUNT LT 1000.0 THEN
0400     ASSIGN MSG-INFO.##MSG = 'Order Amount not less than 1000'
0410     ASSIGN MSG-INFO.##ERROR-FIELD = 'ORDER-AMOUNT'
0420   END-IF
0430 END-SUBROUTINE
0440 *
0450 ************************************************************
0460 DEFINE SUBROUTINE V0-NCST-ORDER-LINES
0470 ************************************************************
0480 *
0490 * If no product is ordered, terminate the process without
0500 * going to the lower level (order distribution).
0510   IF NCST-ORDER-LINES.QUANTITY LE 0
0520     ASSIGN MSG-INFO.##MSG = 'One product must be ordered'
0530     ASSIGN MSG-INFO.##ERROR-FIELD = 'QUANTITY'
0540     ESCAPE ROUTINE
0550   END-IF
0560 *
0570 * Update the order line amount
0580   COMPUTE NCST-ORDER-LINES.TOTAL-COST =
0590   NCST-ORDER-LINES.QUANTITY*NCST-ORDER-LINES.UNIT-COST
0600   /*
0610   /* Accumulate the line amount to the total order amount
0620 ADD NCST-ORDER-LINES.TOTAL-COST TO NCST-ORDER-HEADER.ORDER-AMOUNT
0630 END-SUBROUTINE
0640 END-EXIT UPDATE-EDITS
```

USER-DEFINED-FUNCTIONS

The code in this exit defines the processing performed for user-defined functions processed against an object. You can use this exit to specify user-defined action codes, for example.

The format for this user exit is:

```
VALUE 'function-name'
processing
```

Note: For information on adding actions, see *Natural Construct Generation.*

USER-DEFINED-METHODS

The code in this exit defines the methods available to the generated module.

WRITE-COLUMN-HEADERS

The code in this exit defines the column headings displayed on a generated panel.

WRITE-DATA-FIELDS

The code in this exit defines the processing performed to display selected fields on a generated panel.

WRITE-FIELDS

The code in this exit defines additional logic to write information based on the contents of the #PANEL variable. 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 a browse program using the customer name as the key field; by default, Natural Construct will generate a browse program that browses the Customer file by customer name.

Note: This user exit is not currently available.

Example of user exit code for the Browse-Subp model

```
0010 DEFINE EXIT WRITE-FIELDS
0020 DISPLAY 'Warehouse'(I) NCST-WAREHOUSE.WAREHOUSE-ID(AD=I)
0030         'Description'(I) NCST-WAREHOUSE.WAREHOUSE-DESCRIPT
0040 END-EXIT WRITE-FIELDS
```

Example of user exit code for the Browse-Select-Subp model

```
0010 DEFINE EXIT WRITE-FIELDS
0020     WRITE(0)
0030       3X NCST-ORDER-HEADER.ORDER-NUMBER
0040       6X NCST-ORDER-HEADER.ORDER-DATE
0050       6X NCST-ORDER-HEADER.ORDER-AMOUNT
0060 END-EXIT WRITE-FIELDS
```

Example of user exit code for the Browse model

```
0010 DEFINE EXIT WRITE-FIELDS
0020     DISPLAY(0)
0030       'Cust/No' NCST-CUSTOMER.CUSTOMER-NUMBER (AD=I)
0040       '/Business Name' NCST-CUSTOMER.BUSINESS-NAME
0050       '/Phone No' NCST-CUSTOMER.PHONE-NUMBER
0060       'WHS/ID' NCST-CUSTOMER.CUSTOMER-WAREHOUSE-ID
0070       '/Contact' NCST-CUSTOMER.CONTACT
0080         (AL=25)
0090 *
0100 * Display additional data upon request.
0110 DECIDE ON FIRST VALUE #OPTION
0120   VALUE 'M' /* Display Mailing Address
0130     WRITE 06X 'Mailing Address:'(I) NCST-CUSTOMER.M-STREET
0140     WRITE 23X NCST-CUSTOMER.M-CITY
0150     WRITE 23X NCST-CUSTOMER.M-PROVINCE
0160     WRITE 23X NCST-CUSTOMER.M-POSTAL-CODE
0170     WRITE ' '
0180   VALUE 'S' /* Display Shipping Address
0190     WRITE 06X 'Shipping Address:'(I) NCST-CUSTOMER.S-STREET
0200     WRITE 24X NCST-CUSTOMER.S-CITY
0210     WRITE 24X NCST-CUSTOMER.S-PROVINCE
0220     WRITE 24X NCST-CUSTOMER.S-POSTAL-CODE
0230     WRITE ' '
0240   VALUE 'C'
0250    WRITE 06X 'Credit Rating..:'(I) NCST-CUSTOMER.CREDIT-RATING
0260     WRITE 06X 'Credit Limit...:'(I)
0270               NCST-CUSTOMER.CREDIT-LIMIT(AD=L SG=OFF)
0280     WRITE 06X 'Disc. Percentage:'(I)
0290               NCST-CUSTOMER.DISCOUNT-PERCENTAGE(EM=Z9.99'%')
0300     WRITE ' '
0310   ANY
0320     IF *LINE-COUNT > 15
0330       /*
0340       /* Trigger new page if there is not enough space for
0350       /* next record's additional data.
0360       NEWPAGE
0370     END-IF
0380   NONE IGNORE
0390 END-DECIDE
0400 END-EXIT WRITE-FIELDS
```

Example of user exit code for the Browse-Select model

```
0010 DEFINE EXIT WRITE-FIELDS
0020 WRITE 9T NCST-ORDER-HEADER.ORDER-NUMBER (AD=I)
0030       18T NCST-ORDER-HEADER.ORDER-CUSTOMER-NUMBER
0040       30T NCST-ORDER-HEADER.ORDER-WAREHOUSE-ID
0050       39T NCST-ORDER-HEADER.INVOICE-NUMBER
0060       47T NCST-ORDER-HEADER.ORDER-DATE(EM=99'/'99'/'99)
0070       57T NCST-ORDER-HEADER.ORDER-AMOUNT
0080 IF DETAIL THEN
0090   FIND-CUST.
0100   FIND(1) NCST-CUSTOMER WITH CUSTOMER-NUMBER =
0110        NCST-ORDER-HEADER.ORDER-CUSTOMER-NUMBER
0120     DECIDE FOR FIRST CONDITION
0130       WHEN NCST-CUSTOMER.S-CITY NE ' ' AND
0140            NCST-CUSTOMER.S-PROVINCE NE ' '
0150        COMPRESS NCST-CUSTOMER.S-CITY ',' TO #CITY-PROVINCE
0160    LEAVING NO
0170         COMPRESS #CITY-PROVINCE NCST-CUSTOMER.S-PROVINCE
0180              TO #CITY-PROVINCE
0190       WHEN NCST-CUSTOMER.S-CITY NE ' '
0200         ASSIGN #CITY-PROVINCE = NCST-CUSTOMER.S-CITY
0210       WHEN NCST-CUSTOMER.S-PROVINCE NE ' '
0220         ASSIGN #CITY-PROVINCE = NCST-CUSTOMER.S-PROVINCE
0230       WHEN NONE
0240         ASSIGN #CITY-PROVINCE = '** Unknown Province **'
0250     END-DECIDE
0260     WRITE 18T 'Customer Name:'(I) NCST-CUSTOMER.BUSINESS-NAME
0270               'Contact:'(I) NCST-CUSTOMER.CONTACT
0280     WRITE 18T 'Phone        :'(I) NCST-CUSTOMER.PHONE-NUMBER
0290     WRITE 18T 'Address      :'(I) NCST-CUSTOMER.S-STREET
0300     PRINT 33T #CITY-PROVINCE
0310     WRITE 18T '              ' NCST-CUSTOMER.S-POSTAL-CODE
0320     /*
0330     /* Try to put record and its details on the same panel.
0340     IF *LINE-COUNT GE 14 THEN
0350       NEWPAGE
0360     ELSE
0370       SKIP 1
0380     END-IF
0390   END-FIND
0400   /*
0410   /* Try to put the record and its detail on the same panel.
0420   IF *NUMBER(FIND-CUST.) NE 1 AND *LINE-COUNT GE 14 THEN
0430     NEWPAGE
0440   END-IF
0450 END-IF
0460 **
0470 END-EXIT WRITE-FIELDS
```

Multi-line Browse-Select Programs

Programs generated using the Browse-Select models usually display one row per record on the screen, although these models allow many lines to be written for each record. By default, the Action field is only activated for the first row in each new record. The following example of a selection screen shows this default behavior:

```
NCCSCUS1         ***** MULTI-LINE BROWSE SELECT *****
Aug 26                 - SELECTION ON FIRST LINE ONLY -                9:52 AM

Action  Cust          Business Name          Phone Number
       Number
------ ------ ----------------------------- ------------

  __        1  Software AG       (CANADA)     519-622-0889
               Mailing Addr : 151 Savage Drive Cambridge Ontario N1T 1S6
               Shipping Addr: P.O. BOX 9 Cambridge Ontario N1T 1S6
  __        2  JOURNEYMEN FABRICATING         519-234-6422
               Mailing Addr : 230 LONGWOOD ST. Kitchener Ontario N3H 2S6
               Shipping Addr: 230 LONGWOOD ST. Kitchener Ontario N3H 2S6
  __      511  Software AG   US               519-624-5623
               Mailing Addr : 1 Bay Street TORONTO ONTARIO B4B 3B3
               Shipping Addr: 100 YOUNG STREET STN A TORONTO ONTARIO B4B 3B3
  __    10001  Journeymen Fabricating         519-234-6422
               Mailing Addr : RR3 Kitchener Ontario N3H 2S5
               Shipping Addr: RR3 Kitchener Ontario N3H 2S5
  __    10002  Les Rivers Custom Fabricating  519-623-6850
Customer Number: _____
Direct command...: ____________________________________________________________
 Copy       DEtail     DIsplay    Modify     Purge      (PF5=flip)
```

Multi-Line Browse-Select Program Example

To change the default, write user exit code to change the default values for the control variables used by the generated program.

Example of changing defaults for the Action field

The following example removes the Action field from the first record line and applies it to subsequent record lines only. Notice that the UPDATE-SELECTION-TABLE subroutine is performed after each line is written:

```
DEFINE EXIT START-OF-PROGRAM
  /*
  /* Override the default attributes for the first and subsequent
  /* display lines.
  ASSIGN #ATTR-LINE1 = (AD=NP) /* No selection allowed for first row
  ASSIGN #ATTR-REST = (AD=I)   /* Allow selection for subsequent rows
END-EXIT
DEFINE EXIT WRITE-FIELDS
  DISPLAY 'Action' #ACTION(AD=N)
          'Cust/Number' NCST-CUSTOMER.CUSTOMER-NUMBER
          NCST-CUSTOMER.BUSINESS-NAME
          NCST-CUSTOMER.PHONE-NUMBER
  PERFORM UPDATE-SELECTION-TABLE
  PRINT(AD=I) 14X 'Mailing Addr :' NCST-CUSTOMER.MAILING-ADDRESS
  PERFORM UPDATE-SELECTION-TABLE
  PRINT(AD=I) 14X 'Shipping Addr:' NCST-CUSTOMER.SHIPPING-ADDRESS
END-EXIT
```

To indicate the record line for which the action applies, use the #SELECTED-LINE variable in the PROCESS-SELECTED-RECORD user exit. For more information, see **PROCESS-SELECTED-RECORD**, page 23.

The following example of the selection screen shows the new default behavior:

```
NCCSCUS2        ***** MULTI-LINE BROWSE SELECT *****
Aug 26                - SELECTION ON DETAIL LINES ONLY -              11:22 AM

Action  Cust          Business Name          Phone Number
       Number
------ ------ ------------------------------ ------------

          1  Software AG       (CANADA)     519-622-0889
  __         Mailing Addr : 151 Savage Drive Cambridge Ontario N1T 1S6
  __         Shipping Addr: P.O. BOX 9 Cambridge Ontario N1T 1S6
          2  JOURNEYMEN FABRICATING         519-234-6422
  __         Mailing Addr : 230 LONGWOOD ST. Kitchener Ontario N3H 2S6
  __         Shipping Addr: 230 LONGWOOD ST. Kitchener Ontario N3H 2S6
        511  Software AG   US               519-624-5623
  __         Mailing Addr : 1 Bay Street TORONTO ONTARIO B4B 3B3
  __         Shipping Addr: 100 YOUNG STREET STN A TORONTO ONTARIO B4B 3B3
      10001  Journeymen Fabricating         519-234-6422
  __         Mailing Addr : RR3 Kitchener Ontario N3H 2S5
  __         Shipping Addr: RR3 Kitchener Ontario N3H 2S5
      10002  Les Rivers Custom Fabricating  519-623-6850
Customer Number: _____
Direct command...: _____________________________________________________________
 Copy       DEtail     DIsplay    Modify     Purge      (PF5=flip)
```

Multi-Line Browse-Select Program Example — After Changing Defaults

For additional examples of multi-line browse-select programs, refer to the Customer subsystem in the Natural Construct demo system.