User Exits for the Administration Models

This section describes the user exits supplied for the Natural Construct administration models. The administration models generate the model subprograms used by all models.

This section covers the following topics:

- What are User Exits?
- Supplied User Exits

What are User Exits?

User exits insert customized or specialized processing into a model subprogram, which is preserved when the module is regenerated. Natural Construct provides a wide variety of user exits for the administration models. The exits vary depending on the type of subprogram generated. Some exits contain sample code or subroutines, while others generate the DEFINE EXIT…END-EXIT lines only — and you provide the code.

You can modify any user exit code generated into the edit buffer. If multiple user exits are generated with the same name, Natural Construct merges them into a single exit.

User exits are provided for the following administration models:

- CST-Clear
- CST-Read
- CST-Save
- CST-Modify and CST-Modify-332
- CST-Pregen
- CST-Postgen
- CST-Frame
- CST-Document
- CST-Validate
- CST-Stream
- CST-Shell

Reuse User Exit Code

If you specify a new model name on the Generation main menu (M function) and the source buffer contains code, you can retain the code and use it with the model you are creating. This functionality saves time and effort when creating models that use the same code.

If the source buffer contains code when you specify a new model name, the following window is displayed:

```
                              Natural Construct            CSGNEW0
                              CLEAR Source Area


             Mark if you wish to clear the source area           _
```

To retain the code in the source buffer for use with the new module:

- Press Enter.

  The first specification panel for the new model is displayed and Natural Construct retains the user exit code for use with the new module.

To clear the code in the source buffer (and not save it for the new module):

1. Select Mark if you wish to clear the source area.
2. Press Enter.

   The source buffer is cleared and the first specification panel for the specified model is displayed.

Invoke the User Exit Editor

You can invoke the User Exit editor from the Generation main menu or from the last specification panel for a model that supports user exits.

- To invoke the User Exit editor from the Generation main menu, see *User Exit Editor*, *Natural Construct Generation*.
- To invoke the User Exit editor from the model specification panels, press PF11 (userX) on the last specification panel for a model that supports user exits.

If user exits are defined for the specified module, the existing user exit code is displayed in the User Exit editor. If no exits are defined, a list of the exits available for that model is displayed.

Tip:
To select additional exits, enter "SAMPLE" at the > prompt.

Note:
The SAMPLE command is performed automatically when you invoke the User Exit editor and no user exits are defined for the specified module.

The User Exits panel is similar for all models. The following example shows the User Exits panel for the CST-Clear model:

```
CSGSAMPL                     CST-Clear Subprogram                    CSGSM0
Aug 17                            User Exits                         1 of 1

               User Exits            Exists    Sample   Required Conditional
 --------------------------------   --------  ---------- -------- ----------
 _  CHANGE-HISTORY                            Subprogram
 _  PARAMETER-DATA
 _  LOCAL-DATA
 _  PROVIDE-DEFAULT-VALUES                    Subprogram
 _  BEFORE-CHECK-ERROR                         Example
 _  ADDITIONAL-INITIALIZATIONS                 Example
 _  END-OF-PROGRAM
```

The fields on this panel are:

Field	**Description**
User Exits	Names of the user exits available for this model. If a user exit is required and not conditional (its existence is not based on condition codes in the code frames), it is marked by default.
Exists	Indicates whether the corresponding user exit is defined. ● If the exit exists, this field is marked. ● If the exit does not exist, this field is blank.
Sample	Indicates the contents of the user exit. ● If the exit is empty (contains DEFINE EXIT … END-EXIT lines), this field is blank. ● If the exit contains a subprogram, "Subprogram" is displayed. ● If the exit contains sample code, "Example" is displayed.
Required	Indicates whether the user exit is required. ● If the exit must be specified, "X" is displayed. ● If the exit is optional, this field is blank.
Conditional	Indicates whether the user exit is conditional (its existence is based on condition codes in the code frames). ● If the exit is conditional, "X" is displayed. ● If the exit is optional, this field is blank.

To select a user exit displayed on the User Exits panel:

1. Type "X" in the input field to the left of each user exit you want to use.
2. Press Enter.

The selected user exits are displayed in the User Exit editor.

Note:
Fully qualify all references to database fields with the file name.

Tip:
You can also define user exits in the User Exit editor without using the SAMPLE command.

Define User Exits

The code specified within a user exit depends on the type of module generated and the user exit used. However, all Natural Construct user exits have the following format:

```
0010 DEFINE EXIT user-exit-name
0020   user exit code
0030 END-EXIT user-exit-name
```

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

Supplied User Exits

This section describes the user exits supplied for the Natural Construct administration models. The user exits are listed in alphabetical order. For many exits, one or more examples are included.

The supplied user exits are:

- ADDITIONAL-INITIALIZATIONS
- ADDITIONAL-SUBSTITUTION-VALUES
- ADDITIONAL-TRANSLATIONS
- AFTER-INPUT
- AFTER-INVOKE-SUBPANELS
- ASSIGN-DERIVED-VALUES
- BEFORE-CHECK-ERROR
- BEFORE-INPUT
- BEFORE-INVOKE-SUBPANELS
- BEFORE-REINPUT-MESSAGE
- BEFORE-STANDARD-KEY-CHECK
- CHANGE-HISTORY
- DESCRIBE-INPUTS

- END-OF-PROGRAM
- GENERATE-CODE
- GENERATE-SUBROUTINES
- GENERATE-VALIDATIONS
- INPUT-ADDITIONAL-PARAMETERS
- INPUT-SCREEN
- LOCAL-DATA
- MISCELLANEOUS-SUBROUTINES
- MISCELLANEOUS-VARIABLES
- PARAMETER-DATA
- PF-KEYS
- PROCESS-SPECIAL-KEYS
- PROVIDE-DEFAULT-VALUES
- SAVE-PARAMETERS
- SET-CONDITION-CODES
- START-OF-PROGRAM
- SUBSTITUTION-VALUES
- VALIDATE-DATA

ADDITIONAL-INITIALIZATIONS

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

Example of Code

```
** 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-SUBSTITUTION-VALUES

This user exit is used in combination with the LOCAL-DATA user exit. It generates STACK statements for code frame parameters that do not have a corresponding variable in the model PDA.

Example of Code

```
DEFINE EXIT ADDITIONAL-SUBSTITUTION-VALUES
*
* Substitution for frame parameters not defined in model PDA
STACK TOP DATA FORMATTED ’&CENTERED-HEADER1’
                          #CENTERED-HEADER1
STACK TOP DATA FORMATTED ’&CENTERED-HEADER2’
                          #CENTERED-HEADER2
STACK TOP DATA FORMATTED ’&DATE-EM’
                          #DATE-EM
STACK TOP DATA FORMATTED ’&EOD-TABT’
                          #EOD-TABT
STACK TOP DATA FORMATTED ’&EXPORT-DELIMITER’
                          #EXPORT-DELIMITER
STACK TOP DATA FORMATTED ’&GT-LT’
                          #GT-LT
STACK TOP DATA FORMATTED ’&HEAD1-LEN’
                          #HEAD1-LEN
STACK TOP DATA FORMATTED ’&HEAD2-LEN’
                          #HEAD2-LEN
STACK TOP DATA FORMATTED ’&INPUT-LINES’
                          #INPUT-LINES
STACK TOP DATA FORMATTED ’&KEY-PREFIX’
                          #KEY-PREFIX
STACK TOP DATA FORMATTED ’&LT-GT’
                          #LT-GT
STACK TOP DATA FORMATTED ’&PARM-NAT-FORMAT’
                          #PARM-NAT-FORMAT
STACK TOP DATA FORMATTED ’&PREFIX-NAT-FORMAT’
                          #PREFIX-NAT-FORMAT
STACK TOP DATA FORMATTED ’&SEL-TBL-SIZE’
                          #SEL-TBL-SIZE
STACK TOP DATA FORMATTED ’&TIME-EM’
                          #TIME-EM
STACK TOP DATA FORMATTED ’&UQ’
                          #UQ
STACK TOP DATA FORMATTED ’&UQ-FOUND’
                          #UQ-FOUND
STACK TOP DATA FORMATTED ’&VALUE-UQ’
                          #VALUE-UQ
STACK TOP DATA FORMATTED ’&VAR-UQ’
                          #VAR-UQ
STACK TOP DATA FORMATTED ’&VIEW-LDA’
                          #VIEW-LDA
STACK TOP DATA FORMATTED ’&WINDOW-WIDTH’
                          #WINDOW-WIDTH
STACK TOP DATA FORMATTED ’&WITH-BLOCK’
                          #WITH-BLOCK
END-EXIT ADDITIONAL-SUBSTITUTION-VALUES
```

ADDITIONAL-TRANSLATIONS

This user exit generates the framework for additional translations performed in the GET-PROMPT-TEXT subroutine.

Example of Code

```
**SAG DEFINE EXIT ADDITIONAL-TRANSLATIONS
*
 IF #FIRST-TRANSLATION OR CU--PDA.#PDA-PHASE = CSLPHASE.#TRANSLATE
    THEN
    PERFORM SET-MODIFY-HEADER3
    /*
    /* Set completed message
    RESET CNAMSG.INPUT-OUTPUTS
    ASSIGN CNAMSG.MSG-DATA(1) = #PDA-FRAME-PARM
    ASSIGN CNAMSG.MSG = CUBASRPL.#RETURN-MESSAGE
    PERFORM GET-MESSAGE-TEXT
    ASSIGN CUBASRPL.#RETURN-MESSAGE = CNAMSG.MSG
    RESET CNAMSG.INPUT-OUTPUTS
    /*
    /* Assign available keys
    ASSIGN CU--PDA.#PDA-AVAILABLE1-NAME = #AVAILABLE1-NAME
    ASSIGN CU--PDA.#PDA-AVAILABLE2-NAME = #AVAILABLE2-NAME
    ASSIGN CU--PDA.#PDA-AVAILABLE3-NAME = #AVAILABLE3-NAME
    RESET #FIRST-TRANSLATION
    /*
    /* Override pfkey settings
    RESET #LOCAL-PFKEYS-REQUIRED
    /*
    /* Set all PF-keys named off
    INCLUDE CU--SOFF
    /*
    /* Set Help and Return keys
    SET KEY DYNAMIC CU--PDA.#PDA-PF-HELP = HELP
            NAMED CU--PDA.#PDA-HELP-NAME
    SET KEY DYNAMIC CU--PDA.#PDA-PF-RETURN
            NAMED CU--PDA.#PDA-RETURN-NAME
    END-IF
    **SAG END-EXIT
```

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
- Add non-standard PF-key processing to a module

For example, when you add a non-standard PF-key, you can 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. If you do not include the PERFORM NEW-SCREEN code and continue with execution after processing this exit, an Invalid PF-key message is displayed.

Example of Code

```
**SAG DEFINE EXIT AFTER-INPUT
  IF #FORMAT-HELP
    RESET #FORMAT-HELP
    ASSIGN  CU--FHL.#TEXT-REQUIRED = TRUE
    PERFORM GET-PROMPT-TEXT
    INPUT WINDOW = ’FRMT’ USING MAP ’CU--FH0’
    ASSIGN CSAMARK.ERROR-POS = POS(#PDAX-VARIABLE-FORMAT)
    ESCAPE BOTTOM (NEW-SCREEN.)
  END-IF
*
**SAG END-EXIT
```

AFTER-INVOKE-SUBPANELS

This user exit generates the framework for any processing performed after subpanels are invoked.

Example of Code

```
DEFINE EXIT AFTER-INVOKE-SUBPANELS
  PERFORM SET-MORE-INDICATORS
END-EXIT
```

ASSIGN-DERIVED-VALUES

This user exit generates initialization statements for all #PDA variables in the model PDA. The variables are assigned null default values. You can modify the generated code as desired.

Tip:
If you add specification parameters to the model PDA, you can get sample statements for the new parameters by regenerating this exit. Regeneration adds the new variables, but does not modify code from the previous generation.

Example of Code

```
DEFINE EXIT ASSIGN-DERIVED-VALUES
*
* Initialize ’#PDA-’ parameters in PDA.
  ASSIGN #PDA-FIELD-TYPE = ’ ’
  ASSIGN #PDA-FIELD-REDEFINED = FALSE
  ASSIGN #PDA-LEVEL-NUMBER = 0
  ASSIGN #PDA-FIELD-FORMAT = ’ ’
  ASSIGN #PDA-FIELD-LENGTH = 0
  ASSIGN #PDA-UNITS = 0
  ASSIGN #PDA-DECIMALS = 0
  ASSIGN #PDA-FROM-INDEX(*) = 0
  ASSIGN #PDA-THRU-INDEX(*) = 0
  ASSIGN #PDA-FIELD-RANK = 0
  ASSIGN #PDA-FILE-CODE = 0
  ASSIGN #PDA-MAX-LINES = 0
  ASSIGN #PDA-WFRAME = ’ ’
  ASSIGN #PDA-WLENGTH = ’ ’
  ASSIGN #PDA-WCOLUMN = ’ ’
  ASSIGN #PDA-WBASE = ’ ’
END-EXIT ASSIGN-DERIVED-VALUES
```

BEFORE-CHECK-ERROR

This user exit generates the framework for any processing performed before a standard error check. When an error condition occurs, the END-OF-PROGRAM user exit is bypassed. If a model subprogram requires processing before leaving the program, use this user exit to specify the processing.

Example of Code

```
**SAG DEFINE EXIT BEFORE-CHECK-ERROR
*
* Use this user exit for specific error checking
  IF CSASTD.RETURN-CODE = CSLRCODE.#INTERRUPT(*)
    ASSIGN C--PDA.#PDA-PHASE = #SAVE-PHASE
  END-IF
 **SAG END-EXIT
```

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:

- Look up a code table (to display a description, as well as a code value)
- Issue SET CONTROL statements
- Capture or default map variables prior to displaying each panel

Example of Code

```
DEFINE EXIT BEFORE-INPUT
*
* Assign external value
  FOR #I = 1 TO 7
    IF #PDAX-BACKGROUND-COLOUR = #CD(#I) THEN
      ASSIGN #REVERSED-CD(#I) = TRUE
      ESCAPE BOTTOM
    END-IF
  END-FOR
END-EXIT
```

BEFORE-INVOKE-SUBPANELS

This user exit generates the framework for any processing performed before subpanels are invoked.

Example of Code

```
DEFINE EXIT BEFORE-INVOKE-SUBPANELS
  IF CU--PDA.#PDA-PHASE NE CSLPHASE.#TRANSLATE THEN
    PERFORM VALIDATE-FILE-INFO
  END-IF
END-EXIT
```

BEFORE-REINPUT-MESSAGE

The code in this user exit allows you to interrogate the message codes and override the display logic for the generated messages. For example, if the logic specifies that a message is ignored, you can display the message. If the logic specifies that the program is interrupted, you can terminate the program.

Example of Code

```
END-SUBROUTINE /* INPUT-SCREEN
*
* DEFINE SUBROUTINE REINPUT-MESSAGE
*
**SAG DEFINE EXIT BEFORE-REINPUT-MESSAGE
   IF CSASTD.RETURN-CODE = CSLRCODE.#COMMUNICATION THEN
      ESCAPE BOTTOM(PROG.) IMMEDIATE
   END-IF
**SAG END-EXIT
   DECIDE FOR FIRST CONDITION
      WHEN CSASTD.RETURN-CODE = CSLRCODE.#CONTINUE(*)
             IGNORE
      WHEN CSASTD.RETURN-CODE = CSLRCODE.#INTERRUPT(*)
                  ESCAPE BOTTOM(NEW-SCREEN)
       WHEN NONE
             IGNORE
END-DECIDE
```

BEFORE-STANDARD-KEY-CHECK

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

Example of Code

```
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
```

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

```
DEFINE EXIT CHANGE-HISTORY
* Changed on Aug 17,07 by SAG for release ____
* >
* >
* >
END-EXIT CHANGE-HISTORY
```

DESCRIBE-INPUTS

This user exit contains statements that document specification parameter values (#PDAX variables) in the model PDA. For example, if you are documenting a menu program, this user exit contains the menu function codes and descriptions.

Example of Code

```
DEFINE EXIT DESCRIBE-INPUTS
*
* Enter other model parameters to be documented.
* Use WRITE statements of the following format:
*     WRITE(SRC) NOTITLE LDA.#Variable-name #PDAX-variable-name
END-EXIT DESCRIBE-INPUTS
```

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) before exiting the module. You can also use this exit to assign the current key value to a global variable so it is carried into other modules that use the same key.

Note:
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.

Example of Code

```
3310 **SAG DEFINE EXIT END-OF-PROGRAM
3320 *
3330 * Actions to be performed before program exit.
3340   IF #PDAX-GDA NE ’ ’ AND #PDA-PHASE = ’M’ THEN
3350     ASSIGN CNAEXIST.#OBJECT-SOURCE = ’O’
3360     ASSIGN CNAEXIST.#LIBRARY-NAME = *LIBRARY-ID
3370     ASSIGN CNAEXIST.#INCLUDE-STEPLIB-SEARCH = TRUE
3380     ASSIGN CNAEXIST.#OBJECT-NAME   = #PDAX-GDA
3390     CALLNAT ’CNUEXIST’ CNAEXIST
3400                        CSASTD
3410     PERFORM CHECK-ERROR
3420     IF NOT CNAEXIST.#OBJECT-EXISTS THEN
3430       ASSIGN CNAMSG.RETURN-CODE = CSLRCODE.#WARNING
3440       ASSIGN CNAMSG.MSG-DATA(1) = CU--MAL.#GDA
3450       ASSIGN CNAMSG.MSG-DATA(2) = #PDAX-GDA
3460       INCLUDE CU--GMSG ’2128’
3470         ’’’:1::2::3:not in current library or STEPLIBs’’’
3480     END-IF
3490   END-IF
3500 **SAG END-EXIT
```

GENERATE-CODE

This user exit generates the framework for any code generated by a model subprogram.

Example of Code

```
DEFINE EXIT GENERATE-CODE
*
  RESET CSASELFV CSASELFV.GENERAL-INFORMATION
                 CSASELFV.FIELD-SPECIFICATION(*)
  MOVE CUMPPDA.#PDAX-VIEW-LPDA-STRUCT-NAME(*) TO
                                    CSASELFV.#VIEW-LPDA-STRUCT-NAME(*)
  MOVE CUMPPDA.#PDAX-FIELD-NAME(*) TO CSASELFV.FIELD-NAME(*)
  MOVE CUMPPDA.#PDAX-FIELD-FORMAT(*) TO CSASELFV.FIELD-FORMAT(*)
  MOVE CUMPPDA.#PDAX-FIELD-LENGTH(*) TO CSASELFV.FIELD-LENGTH(*)
  FOR #I = 1 TO #MAX-FLDS
    MOVE CUMPPDA.#PDAX-MAX-OCCURS(#I) TO
                                       CSASELFV.FIELD-OCCURRENCES(#I,1)
  END-FOR
  MOVE CUMPPDA.#PDAX-STRUCTURE-NUMBER(*) TO
                                       CSASELFV.#STRUCTURE-NUMBER(*)
  MOVE CUMPPDA.#PDAX-FIELD-PROMPT-OR-TEXT(*) TO
                                       CSASELFV.FIELD-HEADINGS(*)
  ASSIGN CSASELFV.#ARRAY-RANK-SELECTED = 1
  CALLNAT 'CSUSELFV' CSASELFV
                     CU--PDA
                     CSASTD
  ASSIGN CSASTD.ERROR-FIELD-INDEX1 = CSASELFV.#ERROR-FIELD-INDEX
  PERFORM CHECK-ERROR
  RESET CSASTD.ERROR-FIELD-INDEX1
  MOVE CSASELFV.FIELD-NAME(*) TO CUMPPDA.#PDAX-FIELD-NAME(*)
  MOVE CSASELFV.FIELD-FORMAT(*) TO CUMPPDA.#PDAX-FIELD-FORMAT(*)
  MOVE CSASELFV.FIELD-LENGTH(*) TO CUMPPDA.#PDAX-FIELD-LENGTH(*)
  MOVE CSASELFV.#STRUCTURE-NUMBER(*) TO
                                  CUMPPDA.#PDAX-STRUCTURE-NUMBER(*)
  MOVE CSASELFV.FIELD-HEADINGS(*) TO
                                  CUMPPDA.#PDAX-FIELD-PROMPT-OR-TEXT(*)
  MOVE CSASELFV.#VIEW-LPDA-STRUCT-NAME(*) TO
                                  CUMPPDA.#PDAX-VIEW-LPDA-STRUCT-NAME(*)
  FOR #I = 1 TO #MAX-FLDS
    MOVE CSASELFV.FIELD-OCCURRENCES(#I,1)
      TO CUMPPDA.#PDAX-MAX-OCCURS(#I)
    EXAMINE CUMPPDA.#PDAX-FIELD-PROMPT-OR-TEXT(#I) FOR '/'
      REPLACE WITH ' '
  END-FOR
END-EXIT GENERATE-CODE
```

GENERATE-SUBROUTINES

This user exit generates the framework for validations performed by the model validation subprogram. It is used in conjunction with the GENERATE-VALIDATIONS user exit and is available for modules generated using the CST-Validate model.

Code validations as subroutines in this user exit. For each #PDAX-FIELD-NAME field you want to validate, create a subroutine called V-*field-name* to perform the validations. Whenever a validation error is found, the V-*field-name* subroutine must:

- Assign `CSASTD.RETURN-CODE = 'E'`
- Assign the error message in CSASTD.MSG

- Perform an ESCAPE-ROUTINE to bypass subsequent checks

Tip:
To retrieve SYSERR messages, use the CU--VERR copycode.

Tip:
To return a warning message, rather than an error, use the CU--VWAR copycode.

References

- For more information about coding validations, see CST-Validate Model.
- For information about validating array fields, see Validate Array Fields.

GENERATE-VALIDATIONS

This user exit generates the framework for validations performed by the model validation subprogram. It is used in conjunction with the GENERATE-SUBROUTINES user exit and is available for modules generated using the CST-Validate model.

Note:
For more information, see CST-Validate Model.

INPUT-ADDITIONAL-PARAMETERS

This user exit contains an INPUT statement to read parameters that are not automatically included in a read subprogram.

Example of Code

```
DEFINE EXIT INPUT-ADDITIONAL-PARAMETERS
*
* Input all other parameters..
*
*   /* Input parameter SAMPLE
*   WHEN #LINE = ’SAMPLE:’
*     INPUT CXMYPDA.#PDAX-SAMPLE
END-EXIT INPUT-ADDITIONAL-PARAMETERS
```

INPUT-SCREEN

This user exit generates code to input screens (maps) for a maintenance subprogram.

Example of Code

```
DEFINE EXIT INPUT-SCREEN
IF CSASTD.RETURN-CODE = CSLERROR.#OK OR = CSLERROR.#WARNING
  INPUT WITH TEXT CSASTD.MSG
    MARK POSITION CSAMARK.ERROR-COLUMN IN CSAMARK.ERROR-POS
    USING MAP ’map’
ELSE
  INPUT WITH TEXT CSASTD.MSG
    MARK POSITION CSAMARK.ERROR-COLUMN IN CSAMARK.ERROR-POS
    ALARM
    USING MAP ’map’
END-IF
END-EXIT INPUT-SCREEN
```

LOCAL-DATA

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

Example of Code

```
0480 **SAG DEFINE EXIT LOCAL-DATA
0490     01 #HELPR(A8) INIT<’CD-HELPR’>
0500   LOCAL USING CNAEXIST
0510 **SAG END-EXIT
```

MISCELLANEOUS-SUBROUTINES

This user exit generates the framework for any additional subroutines used by a maintenance subprogram.

Example of Code

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

MISCELLANEOUS-VARIABLES

This user exit generates code to write the field and prompt values to Predict. To generate the correct code, translation LDAs must adhere to the following naming standards:

Field	Prompt
#PDA-GEN-PROGRAM	CUMNMAL.#GEN-PROGRAM
#PDAX-TITLE	CUMNMAL.#TITLE

Example of Code

```
0010 DEFINE EXIT MISCELLANEOUS-VARIABLES
0020 *****************************************************************
0030 DEFINE SUBROUTINE MISCELLANEOUS
0040 *****************************************************************
0050 *
0060   WRITE(SRC) NOTITLE 20T CU--DOCL.#MISC-SPECIFICATIONS
0070   WRITE(SRC) NOTITLE     CU--PDA.#PDA-UNDERSCORE-LINE (AL=70)
0080   WRITE(SRC) NOTITLE ’ ’
0090 END-SUBROUTINE /* MISCELLANEOUS
0100 END-EXIT
```

PARAMETER-DATA

This user exit generates the framework to process any additional parameters used in conjunction with other programs.

Example of Code

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

PF-KEYS

This user exit documents information about PF-keys supported by a generated subprogram to the Predict data dictionary.

To document information about PF-keys:

1. Select the PF-KEYS user exit.
2. Press Enter.

   A window is displayed, in which you can specify the supported PF-keys. Descriptions of the keys are added to Predict.

Example of Code

```
* Translate pfkey functions
  PERFORM GET-CDKEYFL-TEXT
*
* Write pfkey names and functions
  PRINT(SRC) NOTITLE / 20T CU--DOCL.#PFKEY-SUPPORT
    / ' '
    / 3T CU--DOCL.#PFKEY 14T CU--DOCL.#FUNCTION
    / 3T CU--PDA.#PDA-UNDERSCORE-LINE (AL=10)
         CU--PDA.#PDA-UNDERSCORE-LINE (AL=60)
    / 3T CDKEYLDA.#KEY-NAME(2)
     14T CDKEYFL.#KEY-FUNCTION(2)
END-SUBROUTINE /* PF-KEYS
END-EXIT
DEFINE EXIT PF-KEYS
*****************************************************************
DEFINE SUBROUTINE PF-KEYS
*****************************************************************
*
* Translate pfkey names
  INCLUDE CU--DOC
*
* Translate pfkey functions
  PERFORM GET-CDKEYFL-TEXT
*
* Write pfkey names and functions
  PRINT(SRC) NOTITLE / 20T CU--DOCL.#PFKEY-SUPPORT
    / ' '
    / 3T CU--DOCL.#PFKEY 14T CU--DOCL.#FUNCTION
    / 3T CU--PDA.#PDA-UNDERSCORE-LINE (AL=10)
```

```
0380          CU--PDA.#PDA-UNDERSCORE-LINE (AL=60)
0390     / 3T CDKEYLDA.#KEY-NAME(3)
0400      14T CDKEYFL.#KEY-FUNCTION(3)
0410 END-SUBROUTINE /* PF-KEYS
0420 END-EXIT
```

PROCESS-SPECIAL-KEYS

This user exit is required for the CST-Modify-332 model if the maintenance subprogram supports special PF-keys (all keys other than Enter and help, return, quit, right, and left PF-keys).

Note:
Define the special PF-keys on the Maintain Subprograms panel. For information, see Maintain Subprograms Function.

After defining the keys and generating the model, this exit contains code you can use as a starting point for processing the keys.

Example of Code

```
DEFINE EXIT PROCESS-SPECIAL-KEYS
  ASSIGN #PF-KEY = *PF-KEY
  DECIDE ON FIRST VALUE OF *PF-KEY
    VALUE #PF-*0039
      /*
      /* Perform *0039 processing
      ASSIGN CSASTD.MSG = '*0039 processing completed successfully'
      ESCAPE TOP
    NONE VALUE
      IF *PF-KEY NE 'ENTR'
        REINPUT 'Invalid key:1:entered',#PF-KEY
      END-IF
  END-DECIDE
END-EXIT PROCESS-SPECIAL-KEYS
```

PROVIDE-DEFAULT-VALUES

This user exit provides a list of default values for model parameters. If desired, it can also supply values for other parameters you want to initialize. Natural Construct provides default values for the #PDAX variables in the model PDA.

Tip:
To specify default values for additional parameters in a model PDA, regenerate this user exit. This adds the new variables but does not modify the code from the previous generation.

Example of Code

```
DEFINE EXIT PROVIDE-DEFAULT-VALUES
  ASSIGN CXMNPDA.#PDAX-DESCS(*) = ' '
  ASSIGN CXMNPDA.#PDAX-USE-MSG-NR = FALSE
  ASSIGN CXMNPDA.#PDAX-PDA = ' '
  ASSIGN CXMNPDA.#PDAX-FILE-NAME = ' '
  ASSIGN CXMNPDA.#PDAX-FIELD-NAME = ' '
  ASSIGN CXMNPDA.#PDAX-MAP-NAME = ' '
  ASSIGN CXMNPDA.#PDAX-LINES-PER-SCREEN = 0
  ASSIGN CXMNPDA.#PDAX-WINDOW-BASE = ' '
  ASSIGN CXMNPDA.#PDAX-WINDOW-BASE-LINE = 0
  ASSIGN CXMNPDA.#PDAX-WINDOW-BASE-COLUMN = 0
  ASSIGN CXMNPDA.#PDAX-WINDOW-SIZE = ' '
  ASSIGN CXMNPDA.#PDAX-WINDOW-LINE-LENGTH = 0
  ASSIGN CXMNPDA.#PDAX-WINDOW-COLUMN-LENGTH = 0
  ASSIGN CXMNPDA.#PDAX-WINDOW-FRAME = FALSE
END-EXIT PROVIDE-DEFAULT-VALUES
```

SAVE-PARAMETERS

This user exit is required for the CST-Save model. It generates a WRITE statement for each specification parameter (#PDAX variable) in the model PDA. Elements of array variables are written individually, including the number of array occurrences. The WRITE statement has the following format:

`WRITE(SRC) NOTITLE '=' #PDAX-`*`variable-name`*

Natural Construct transforms these lines as follows:

`**SAG `*`variable name`*`: `*`variable contents`*

and writes them at the beginning of Natural Construct-generated modules.

Tip:
If you add specification parameters to a model PDA, regenerate this user exit to generate the WRITE statements for the new parameters. Regeneration adds the new variables, but does not modify code from the previous generation.

Example of Code

```
DEFINE EXIT SAVE-PARAMETERS
FOR #I = 1 TO 4
  IF #PDAX-DESCS(#I) NE ' ' THEN
    COMPRESS '#PDAX-DESCS(' #I '):' INTO #TEXT
    LEAVING NO
    PRINT(SRC) NOTITLE #TEXT #PDAX-DESCS(#I)
  END-IF
END-FOR
WRITE(SRC) NOTITLE '=' #PDAX-USE-MSG-NR
  / '=' #PDAX-PDA
  / '=' #PDAX-FILE-NAME
  / '=' #PDAX-FIELD-NAME
  / '=' #PDAX-MAP-NAME
  / '=' #PDAX-LINES-PER-SCREEN
  / '=' #PDAX-WINDOW-BASE
  / '=' #PDAX-WINDOW-BASE-LINE
  / '=' #PDAX-WINDOW-BASE-COLUMN
  / '=' #PDAX-WINDOW-SIZE
  / '=' #PDAX-WINDOW-LINE-LENGTH
  / '=' #PDAX-WINDOW-COLUMN-LENGTH
  / '=' #PDAX-WINDOW-FRAME
END-EXIT SAVE-PARAMETERS
```

SET-CONDITION-CODES

This user exit is required for the CST-Pregen model. It generates initialization statements for all conditions (#PDAC variables) in the model PDA. You can modify the generated code as desired.

A condition is set to True when a variable corresponding to the condition exists in the model PDA and has a non-null value. The variables and conditions are linked through their names; the #PDAX-name variable corresponds to the #PDAC-*name* or #PDAC-*name*-SPECIFIED condition.

For example, if the model PDA contains:

- #PDAX-USE-MSG-NR(L) variable
- #PDAC-USE-MSG-NR(L) condition

This user exit generates the following code:

```
WHEN #PDAX-USE-MSG-NR NE FALSE
   #PDAC-USE-MSG-NR = TRUE
```

If the model PDA contains:

- #PDAX-GDA(A8) variable
- #PDAC-GDA-SPECIFIED(L) condition

This user exit generates the following code:

```
WHEN #PDAX-GDA NE ’ ’
   #PDAC-GDA-SPECIFIED = TRUE
```

The WHEN clause is blank for all conditions that have no corresponding variable in the model PDA.

Code for the conditions currently existing in this user exit is not generated. When you regenerate this user exit, only the code for new conditions (that were added to the model PDA since the previous generation) is added.

Example of Code

```
DEFINE EXIT SET-CONDITION-CODES
*
* Set conditions in PDA.
  DECIDE FOR EVERY CONDITION
    WHEN #PDAX-USE-MSG-NR NE FALSE
      ASSIGN #PDAC-USE-MSG-NR = TRUE
    WHEN #PDAX-FILE-NAME NE ’ ’
      ASSIGN #PDAC-FILE-NAME-SPECIFIED = TRUE
    WHEN #PDAX-FIELD-NAME NE ’ ’
      ASSIGN #PDAC-FIELD-NAME-SPECIFIED = TRUE
    WHEN #PDAX-PDA NE ’ ’
      ASSIGN #PDAC-PDA-SPECIFIED = TRUE
    WHEN NONE
       IGNORE
  END-DECIDE
END-EXIT
```

START-OF-PROGRAM

The code in this user exit is executed once at the beginning of the generated subprogram after all standard initial values are assigned. You can use this exit to do any initial setup required. For example:

- Initialize input values from globals
- Set window or page sizes
- Capture security information for a restricted data area

SUBSTITUTION-VALUES

This user exit is used by the CST-Postgen model, which generates the post-generation subprogram for a model. The post-generation subprogram generates STACK statements for substitution variables in the model PDA. To generate STACK statements for any substitution variables that are not in the model PDA, select the SUBSTITUTION-VALUES or ADDITIONAL-SUBSTITUTION-VALUES user exit (see below for a comparison).

If you select the SUBSTITUTION-VALUES user exit, STACK statements for all substitution variables are generated in the exit — those in the model PDA, as well as any additional variables. You can modify these variables as desired.

Which user exit you select depends on whether you want the model to stack substitution parameters in the code frame or in a user exit, thereby overriding the default substitution parameter handling.

- If you use the SUBSTITUTION-VALUES user exit, you must code all substitution values in the exit since default code will not be generated.
- If you use the ADDITIONAL-SUBSTITUTION-VALUES user exit (or no user exit), the model automatically stacks any model PDA variables that match the &SUBSTITUTION values in the code frame. For example:

  ```
  STACK TOP DATA FORMATTED ’&PRIME-FILE’ #PDAX-PRIME-FILE
  ```

Note:
Use either the SUBSTITUTION-VALUES user exit or the ADDITIONAL-SUBSTITUTION-VALUES user exit, but not both.

VALIDATE-DATA

The code in this user exit performs edit checks on each parameter on a maintenance map (specified in the Map name field on the Standard Parameters panel). This section contains examples of user exit code for the CST-Modify and CST-Modify-332 model. The CST-Modify model supports dynamic multilingual specification panels and messages using SYSERR references and substitution variables. The code generated in this exit contains SYSERR numbers and substitution values.

Example of Code for CST-Modify Model

```
0010 DEFINE EXIT VALIDATE-DATA
0020   DECIDE FOR EVERY CONDITION
0030     WHEN #HEADER1 = ’ ’
0040       ASSIGN CNAMSG.MSG-DATA(1) = #HEADER1
0050       INCLUDE CU--RMSG ’2001’
0060       ’’’:1::2::3:is required’’’
0070       ’#HEADER1’
0080     WHEN #HEADER2 = ’ ’
0090       ASSIGN CNAMSG.MSG-DATA(1) = #HEADER2
0100       INCLUDE CU--RMSG ’2001’
0110       ’’’:1::2::3:is required’’’
```

```
      ’#HEADER2’
    WHEN #PDA-GEN-PROGRAM = ’ ’
      ASSIGN CNAMSG.MSG-DATA(1) = #GEN-PROGRAM
      INCLUDE CU--RMSG ’2001’
      ’’’:1::2::3:is required’’’
      ’#PDA-GEN-PROGRAM’
    WHEN #PDA-SYSTEM = ’ ’
      ASSIGN CNAMSG.MSG-DATA(1) = #SYSTEM
      INCLUDE CU--RMSG ’2001’
      ’’’:1::2::3:is required’’’
      ’#PDA-SYSTEM’
    WHEN #PDA-TITLE = ’ ’
      ASSIGN CNAMSG.MSG-DATA(1) = #TITLE
      INCLUDE CU--RMSG ’2001’
      ’’’:1::2::3:is required’’’
      ’#PDA-TITLE’
    WHEN CUBAPDA.#PDAX-DESCS = ’ ’
      ASSIGN CNAMSG.MSG-DATA(1) = #DESCS
      INCLUDE CU--RMSG ’2001’
      ’’’:1::2::3:is required’’’
      ’CUBAPDA.#PDAX-DESCS’
    WHEN CUBAPDA.#PDAX-GDA = ’ ’
      ASSIGN CNAMSG.MSG-DATA(1) = #GDA
      INCLUDE CU--RMSG ’2001’
      ’’’:1::2::3:is required’’’
      ’CUBAPDA.#PDAX-GDA’
    WHEN CUBAPDA.#PDAX-GDA-BLOCK = ’ ’
      ASSIGN CNAMSG.MSG-DATA(1) = #GDA-BLOCK
      INCLUDE CU--RMSG ’2001’
      ’’’:1::2::3:is required’’’
      ’CUBAPDA.#PDAX-GDA-BLOCK’
    WHEN CUBAMAL.#DESCRIPTION = ’ ’
      ASSIGN CNAMSG.MSG-DATA(1) = #DESCRIPTION
      INCLUDE CU--RMSG ’2001’
      ’’’:1::2::3:is required’’’
      ’CUBAMAL.#DESCRIPTION’
    WHEN CUBAMAL.#GDA = ’ ’
      ASSIGN CNAMSG.MSG-DATA(1) = #GDA
      INCLUDE CU--RMSG ’2001’
      ’’’:1::2::3:is required’’’
      ’CUBAMAL.#GDA’
    WHEN CUBAMAL.#GDA-BLOCK = ’ ’
      ASSIGN CNAMSG.MSG-DATA(1) = #GDA-BLOCK
      INCLUDE CU--RMSG ’2001’
      ’’’:1::2::3:is required’’’
      ’CUBAMAL.#GDA-BLOCK’
    WHEN CUBAMAL.#GEN-PROGRAM = ’ ’
      ASSIGN CNAMSG.MSG-DATA(1) = #GEN-PROGRAM
      INCLUDE CU--RMSG ’2001’
      ’’’:1::2::3:is required’’’
      ’CUBAMAL.#GEN-PROGRAM’
    WHEN CUBAMAL.#SYSTEM = ’ ’
      ASSIGN CNAMSG.MSG-DATA(1) = #SYSTEM
      INCLUDE CU--RMSG ’2001’
      ’’’:1::2::3:is required’’’
      ’CUBAMAL.#SYSTEM’
    WHEN CUBAMAL.#TITLE = ’ ’
      ASSIGN CNAMSG.MSG-DATA(1) = #TITLE
      INCLUDE CU--RMSG ’2001’
      ’’’:1::2::3:is required’’’
      ’CUBAMAL.#TITLE’
END-EXIT
```

Example of Code for CST-Modify-332 Model

```
DEFINE EXIT VALIDATE-DATA
*
* Edit checks on map parameters.
  DECIDE FOR EVERY CONDITION
    WHEN #HEADER1 = ’ ’
      REINPUT ’Header1 is required’
      MARK *#HEADER1 ALARM
    WHEN #HEADER2 = ’ ’
      REINPUT ’Header2 is required’
      MARK *#HEADER2 ALARM
    WHEN CDDIALDA.#PROGRAM = ’ ’
      REINPUT ’Program is required’
      MARK *CDDIALDA.#PROGRAM ALARM
    WHEN CDGETDCA.#DIRECT-COMMAND = ’ ’
      REINPUT ’Direct Command is required’
      MARK *CDGETDCA.#DIRECT-COMMAND ALARM
    WHEN NONE IGNORE
  END-DECIDE
END-EXIT VALIDATE-DATA
```

The basic structure of this user exit is supplied in the above format. You can edit the supplied code as required.

Tip:
If you add specification parameters to the model PDA, you can generate sample statements for the new parameters by regenerating this user exit. Regeneration adds the new variables, but does not modify code from the previous generation.