Modifying the Supplied Models

This section describes how to modify the models supplied by Natural Construct. In most cases, the existing model can be customized by modifying the code frames associated with the model or the copycode members used in the generated modules. In some cases, the generated code may need to be modified by the subprograms in the model code frames (identified by the CU prefix).

This section covers the following topics:

- Introduction
- Change the Supplied Models
- Example of Modifying a Model
- Use Steplibs to Modify Models

Introduction

The source code for all CU-prefixed subprograms is supplied with Natural Construct. To reduce dependencies between Predict and the Natural Construct models, all models use external subprograms to access the Predict data dictionary (they do not access Predict directly).

Do not modify the supplied model subprograms, as changes to these subprograms may have to be reapplied with each new release of Natural Construct. If you want to modify supplied subprograms, copy the supplied subprogram and use a CX prefix (rather than the CU prefix) to name it.

Additionally, do not modify the supplied code frames. All supplied code frames end with a suffix value of 9 (for example, CMNA9) and updated Natural Construct code frames end with a suffix of 8. To create a custom code frame, copy and rename the supplied code frame with a lower suffix value (for example, CMNA7) and modify the new code frame. Natural Construct searches for and uses the code frame with the lowest suffix value when the program is generated. Document all changes so they can be reapplied to subsequent versions of Natural Construct. For more information, see Maintain Models Function.

Note:
If changes are confined to model subprograms or copycode members used in modules generated by the model, use the multiple steplib feature to customize the model. For information, see Use Steplibs to Modify Models.

Change the Supplied Models

Typically, the Natural Construct administrator makes changes to the generation models. Before a modified model is available for general use, it should be thoroughly tested. The following sections explain how to modify the supplied model code frames, subprograms, and copycode, as well as how to modify the external data areas and subprograms used by the generation models:

- Modify Code Frames

- Modify the Model Subprograms
- Modify Copycode (CC*) and External Data Areas and Subprograms (CD*)

Modify Code Frames

Do not modify the supplied code frames. Instead, make a copy the code frames you want to customize and modify the copy. Keep the original code frames so they can be referred to if problems arise. Changes to code frames take effect immediately after the code frame is saved.

Note:
Document all modifications to the code frames so changes can be reapplied to new versions of Natural Construct.

To modify a code frame:

1. Copy the code frame and use an X prefix to name the copy. For example, the CFEXAM9 code frame becomes XFEXAM9.

   Tip:
   Rather than copying and renaming individual model components, you can create standard, development, and production versions of all system files. Use the CSFUNLD and CSFLOAD utilities to move code frames between files. For information, see Import and Export Utilities.

2. Copy the model that uses the modified code frame and give the copy a different name.

   For example, the Menu model becomes Menu2.

3. Invoke the model copy to test changes to your code frame.

   For example, you can invoke Menu2 model to test the modified code frame without interrupting the use of the Menu model.

4. Change the X prefix back to a C and change the 9 in the last position of the code frame name to a lesser number (from 1 to 7).

   For example, the XFEXAM9 code frame becomes CFEXAM7. Natural Construct always uses the code frame ending with the lesser number.

   Note:
   Do not use the number 8 in the last position of the code frame name. Number 8 is reserved for future changes to the supplied code frames (should they be issued). For more information about modifying code frames, see Step 5: Create Code Frame(s) and Define the Model.

Modify the Model Subprograms

Because the production copies of the model subprograms are invoked from the SYSLIBS library, you can modify and test the model subprograms within the SYSCST library without affecting existing users of the model. To invoke Natural Construct from the SYSCST library (instead of the SYSTEM library), use the CSTG command (not NCSTG).

To invoke Natural Construct from the SYSCST library (instead of the SYSTEM library):

- Enter "CSTG" at the Natural prompt (instead of NCSTG).

To modify a supplied model subprogram (prefixed by CU):

1. Copy the subprogram and change the CU prefix to CX.
2. Copy the corresponding model and refer the copy to the new CX subprogram.

   Note:
   Use the CSUTEST utility to test the model subprograms individually. For more information, see Test the Model Subprograms.

3. After testing the model subprograms in the SYSCST library, copy the modified modules to the SYSLIBS library in the FNAT system file.

   Tip:
   If you change the condition codes in the model PDA, copy the object code for the model PDA into the SYSLIBS library as well.

   Note:
   If Natural Construct is invoked from a steplib, you do not have to rename the supplied subprograms during modification and testing. Instead, copy the subprogram to a test library or other higher level steplib. Once tested, you can copy the modules to the steplib reserved by all development libraries for modifying the supplied modules.

Modify Copycode (CC*) and External Data Areas and Subprograms (CD*)

If you modify any of the CC or CD-prefixed supplied modules and want to apply the changes:

- To programs generated in all libraries, copy the modified modules to the SYSTEM library.
- To one application, copy the modified modules to the corresponding application library.

If you modify the CC or CD-prefixed modules and assign a new name to the modified modules, reference the new name in the Natural Construct standard models. For example, if you modify CCSTDKEY and name the new module MYSTDKEY, refer the Natural Construct standard models to MYSTDKEY instead of CCSTDKEY.

The supplied CSXCNAME user exit subprogram (in the SYSCSTX library) allows users to substitute their own symbols or names for the default values generated into a Natural Construct object (CC* copycode and CD* routines, for example). If this subprogram exists in the SYSLIBS library, it is invoked immediately before the post-generation subprogram for the current model.

The main function of the CSXCNAME subprogram is to place a list of substitution symbols and values on the Natural stack. For example, if you enter the following code in CSXCNAME:

```
STACK TOP DATA FORMATTED 'CCSTDKEY' 'MYSTDKEY'
```

Natural Construct scans for CCSTDKEY and replaces it with MYSTDKEY.

Example of Modifying a Model

This section describes how to modify the maintenance model (Maint). The modifications include the option to generate depth scrolling capabilities, in addition to the current up-down and left-right scrolling. This capability allows a user to scroll a three-dimensional array using the PF4 and PF5 keys. Additionally, the user can name these keys on the second specification panel.

To modify a model:

1. Determine what modifications are required by manually applying the changes to a maintenance program generated by the model.

   The modified program is the prototype. To identify which code frames, PDA, and subprograms to modify, invoke the Maintain Models panel and display information for the Maint model. For information, see Maintain Models Function.

2. Modify the parameter data area (PDA) as follows:

   - Copy the PDA and change the CU prefix to CX.
   - Add a #PDAC-DEPTH-KEYS logical variable to the end of the redefinition of #PDA-CONDITION-CODES.
   - Add a #PDAX-DEPTH-KEYS logical variable to the end of the redefinition of #PDA-USER-AREA.
   - Add two A5 fields (#PDAX-DEPTH-IN and #PDAX-DEPTH-OUT, for example).
   - Stow the modified PDA in the SYSCST library.

   Note:
   If you are executing the steplib version of Natural Construct, move the model PDA to a lower level steplib and make the changes without renaming the object.

3. Modify the second maintenance map and subprogram as follows:

   Tip:
   The subprogram name is displayed in the top left corner of the panel; the map name is displayed in the top right corner of the panel.

   - Copy the current versions and change the CU prefix to CX.
   - Add the #PDAX-DEPTH-KEYS, #PDAX-DEPTH-IN, and #PDAX-DEPTH-OUT fields to the new map. For example:

     ```
     Include Depth Keys: _ (Named: _____ and _____)
     ```

   - Stow the new map and subprogram.

   Note:
   Validation edits (ensuring the keys are named if they are included, for example) can be initiated on the map or within the invoking subprogram.

4. Modify the code frames as follows:

   - Identify the code frames to modify.

     The easiest way to do this is by selecting the Options field when generating a program using the Maint model. When the Status window is displayed, select the Embedded statements option. The generated program will then contain comments showing where each code block originated.

   - Copy the code frames and change the C prefixes to X.

   - Modify the X code frames in the DEPTH-KEYS condition. You can name the keys using substitution parameters assigned in the post-generation subprogram. For example:

     ```
     DEPTH-KEYS                                                  1
     SET KEY CDKEYLDA.#DEPTH-IN-KEY NAMED “&DEPTH-IN’            "
     SET KEY CDKEYLDA.#DEPTH-OUT-KEY NAMED “&DEPTH-OUT’          "
     ```

   - Save the code frame.

   - Make a copy of the model and have the copied model refer to the X copies.

   Note:
   Add the new PF-keys to CDKEYLDA. For information, see *Adding a New PF-Key*, *Natural Construct Generation*.

5. Modify the model subprograms as follows:

   - Make copies of the subprograms and name the copies using an X prefix (or use a steplib).

   - Modify the clear subprogram to initialize the new parameters. For example:

     ```
     RESET #PDAX-DEPTH-KEYS
     ASSIGN #PDAX-DEPTH-IN = ’front’
     ASSIGN #PDAX-DEPTH-OUT = ’back’
     ```

   - Modify the pre-generation subprogram to assign the #PDAC-DEPTH-KEYS logical condition variable to True if the user marks the #PDAX-DEPTH-KEYS field.

   - Modify the post-generation subprogram to assign the names of the depth keys. For example:

     ```
     IF #PDAC-DEPTH-KEYS THEN
       STACK TOP DATA FORMATTED ’&DEPTH-IN’ #PDAX-DEPTH-IN
       STACK TOP DATA FORMATTED ’&DEPTH-OUT’ #PDAX-DEPTH-OUT
     END-IF
     ```

   - Modify the save subprogram to write the new parameters. For example:

     ```
     IF #PDAC-DEPTH-KEYS THEN
       WRITE(SRC) NOTITLE ’=’ #PDAX-DEPTH-KEYS
       WRITE(SRC) NOTITLE ’=’ #PDAX-DEPTH-IN
       WRITE(SRC) NOTITLE ’=’ #PDAX-DEPTH-OUT
     END-IF
     ```

   - Modify the read subprogram to accept the new parameters. For example:

```
WHEN #LINE = 'DEPTH-KEYS:'
  INPUT #PDAX-DEPTH-KEYS
WHEN #LINE = 'DEPTH-IN:'
  INPUT #PDAX-DEPTH-IN
WHEN #LINE = 'DEPTH-OUT:'
  INPUT #PDAX-DEPTH-OUT
```

6. Test the modified model in the SYSCST library (using the CSTG command).

   You can also test individual components of the model using the CSUTEST program or debug the model using the trace options available through the Generation main menu. For more information, see Test the Model Subprograms.

7. Migrate the modified model as follows:

   - Copy the modules for the modified subprograms and PDA from the SYSCST library to the SYSLIBS library.
   - Modify the model definition record (Maintain Models panel) to refer to the modified code frame. For information, see Maintain Models Function.

8. Document all modifications to the model.

Use Steplibs to Modify Models

Using Natural Security, you can define up to eight steplibs for each Natural Construct library. The searching order is the current library (*LIBRARY), the first steplib (if present), the second steplib (if present), …, the eighth steplib (if present), and then the SYSTEM library.

If you store the executing Natural Construct modules in a steplib, you can store your modified model subprograms or copycode in a higher level steplib, effectively overriding any supplied Natural Construct modules with the same names and types. In this way, users access your modified models and the supplied models remain untouched.

When you invoke Natural Construct from a steplib, use the CSTG command (as in the SYSCST library) — not the NCSTG command. The NCSTG command always invokes the copy of Natural Construct that is stored in the SYSLIBS library and bypasses the steplibs.

Tip:
To use the NCSTG command, you can write an NCSTG program to fetch CSTG in the application library.

Because SYSCST is available in a steplib, this method can regulate access to the Administration subsystem. As the Natural Construct administrator, you can use the security routines in the SYSCSTX library to control access to this subsystem.

The following example describes how to use the steplib method to eliminate direct command processing in Natural Construct-generated programs. Direct command processing is triggered by the #PDAX-DIRECT-COMMAND-PROCESS variable on the CU—MA0 map. You can remove the field that contains this variable from the CU—MA0 map and move the modified map into a steplib at a higher level than the SYSCST library.

To use steplibs to modify a model (assuming that APPL is the application library):

1. Define the steplibs to APPL in the following order: NODIRECT, SYSCST, and SYSTEM from Natural Security.

   NODIRECT is a new library and SYSCST and SYSTEM are steplibs of this new library.

2. Copy the CU—MA0 map from the SYSCST library to the NODIRECT library.

3. Edit the CU—MA0 map in the NODIRECT library.

   Delete the text `Mark to include Direct Command Processing` and define the field containing the #PDAX-DIRECT-COMMAND-PROCESS variable as non-display.

4. Stow the modified CU—MA0 map.

5. If you deleted the field that contains the #PDAX-DIRECT-COMMAND-PROCESS variable, copy all the modules that use the CU—MA0 map in the SYSCST library to the NODIRECT library and catalog them.

   Because SYSCST and SYSTEM are steplibs of NODIRECT, these modules can be cataloged in the NODIRECT library.

Note:
If you use the steplib version of Natural Construct for batch regeneration, use the CSTBGEN command instead of the NCSTBGEN command.

Invoke Natural Construct From a Steplib

To invoke Natural Construct from a steplib:

1. Define the SYSCST and SYSLIBS libraries as steplibs of all development libraries requiring Natural Construct.

2. Define a higher level steplib where modules can be stored that override the supplied objects.

3. Add a module called NCSTG to the new steplib and code it as follows:

   ```
   FETCH ’CSTG’
   END
   ```

Tip:
If extensive code frame changes are required, consider installing a second copy of the Natural Construct system file. You can then make changes to code frames directly, without having to make a copy of individual frames and/or modules. You can use the compare facilities supplied with Natural Construct to compare modified models and code frames with the originals. For information about the compare facilities, see Compare Menu Function.