Tips and Techniques

This section provides helpful tips and techniques you can reference when using the Transform-Browse model. The following topics are covered:

- Access Maps and Menus from Original Browse Module
- Transform More Than One Browse Module
- Transform in a Secure Environment
- Display a Variable Number of Lines per Record
- Display Correct Number of Rows
- Display Direct Command Line on a Screen
- Use Edit Masks
- Stop Screens from Advancing Data
- Check for Data Access Code in the PROCESS-SELECTED-RECORD User Exit
- Reference Field Names
- Use Wildcard Characters for Numeric Fields
- Coordinate Data Areas for Browse Program Modules
- Modify Transformation Data
- Change Error Message When No Records Match Selection
- Use the Find Objects Window to Scan a Library (Natural Plug-in Only)

Access Maps and Menus from Original Browse Module

To ensure that the transformed object browse modules have access to other required modules, such as maps or menus used by the original browse module, include the name of the current library in the steplib chain for the transformed module library.

Transform More Than One Browse Module

When using the Transform-Browse model in the Generation subsystem to transform more than one browse module:

- Ensure specifications are cleared between transformations. If not, the module names from the first transformation may be inadvertently used to name the modules for the second transformation.

- Since naming conventions require modules to exist, do not save the specifications for the first transformation without generating the object modules. If you do, the Transform-Browse model may overwrite the specifications. For example, assume the first transformation created the name BR01PN2 for the object browse subprogram and then saved the specifications. If the module is not generated and moved to the transformed module library, the Transform-Browse model is unaware of its existence and may create the same name for the object browse subprogram for the second transformation. During generation, the Transform-Browse model will successfully transform the first browse module but will display a message indicating that the object browse subprogram already exists when transforming the second browse module. If you mark yes to replace the subprogram, the object browse subprogram for the first transformation will be overwritten.

Note:
If you use the Transform Browse wizard, the specifications will not be overwritten because you cannot save the specifications without generating the modules.

Transform in a Secure Environment

When using the Transform-Browse model in a secure environment, a SYSMAIN error may occur. If this happens, try setting the RUNSIZE value to 40.

Display a Variable Number of Lines per Record

If a browse module can have more than one line per record, but the number of lines per record can vary based on user input, developer intervention is required.

The number of lines per record is specified when the transformation is initially performed (see Specification Parameters). Since this number can vary based on user input, specify the smallest number of lines per record. This will create enough space for the most number of rows.

After transforming the browse module, add code to the AFTER-INPUT user exit for the object browse dialog module to change the number of rows requested based on what the user selects. For an example of this functionality, transform the NCCSCUST module in the Demo application and specify one line per record. Next, edit the AFTER-INPUT user exit for the object browse dialog module and specify what processing to perform. This user exit contains the following sample code:

```
*
* If this Browse is transformed to an object browse specifying how
* the requested rows for each screen is important.  To solve this
* uncomment the following lines
* IF #OPTION = 'M' OR = 'S' OR= 'C' THEN
*   CDBRPDA.ROWS-REQUESTED := 2
* ELSE
*   CDBRPDA.ROWS-REQUESTED := 12
* END-IF
*
* Processing to be performed just after the exit checks, after input.
IF NOT (#OPTION = ' ' OR = 'M' OR = 'S' OR = 'C') THEN
  REINPUT 'Valid options are "M", "S", "C", or blank'
  MARK *#OPTION ALARM
END-IF
```

Display Correct Number of Rows

After transforming a browse module, you may encounter the following interface problems:

- Too many rows on a screen

  If a screen has too many rows (i.e., a row is overwritten by the input prompt), modify the specifications on the Standard Parameters panel for the Transform-Browse model, add more lines for the field headings (by default, the Transform-Browse model reserves two lines for field headings), and regenerate the model. Before regenerating, move the original browse module to the current library and set the Replace option to overwrite the modules in the transformed module library.

  Note:
  As the input prompt may overwrite one of the data rows, this problem may not be apparent until runtime. If this is the case, the input prompt may inadvertently change input values while trying to select the hidden row.

- Too few rows on a screen

  If a screen has too few rows, modify the specifications on the Standard Parameters panel for the Transform-Browse model, lower the number of lines reserved for the field headings, and regenerate the model. If the Transform-Browse model reserves two lines for field headings and only one is used, there will be a blank(s) in front of the input prompt and the direct command line will be missing (if the generated module supports direct command processing).

Display Direct Command Line on a Screen

If the direct command is not being displayed on the transformed object browse dialog, ensure the correct number of field heading lines have been specified (see Too few rows on a screen above). If this is not the problem, ensure the dialog specifications include the following line:

```
**SAG INTERNATIONAL-PARMS: F01CSTAPPL CSTAPPL  FF
```

The second last letter indicates whether direct command processing is enabled. F means False (direct command code will not be generated) and T means True (direct command code will be generated).

Use Edit Masks

Since the object browse dialog module does not have access to the Natural views, edit masks are not automatically derived from DDMs. Unless the edit masks have been hard coded in user exits, they will not be included in the transformed code. If edit masks are required, you must manually add them to the object browse dialog.

Stop Screens from Advancing Data

References to `SET CONTROL Q, N, or K0` statements in the original browse module may inadvertently advance screens. By default, the browse module does not populate the first screen; the Transform-Browse model sets the POPULATE-FIRST-SCREEN specification to False. When this happens, the following changes are made to the generated object browse dialog code:

- ```
  01 #FORWARD(L) INIT<FALSE>           /* Forward scrolling
  ```
- ```
  CDBRPDA.ACTUAL-ROWS-RETURNED := 1
  ```

Using this solution, items for WRITE statements may be derived too early. To solve this problem, determine whether the derived code is in the correct position. We recommend that derived values go into the object browse subprogram and that you create additional PDAs for them. If not, the derived values will be lost when dialogs are replaced with web pages or Web services.

If you determine that the derived values should stay in the dialog, they should be wrapped in an IF statement to avoid being processed with the first screen. For example:

```
IF FIRST-TIME NE " " THEN
  DECIDE ON FIRST VALUE OF WORK2D.TRANSFORM-IMPACT(#ROW)
  VALUE 'E'
    #ERROR-TRANSLATION := #E
  VALUE 'S'
    #ERROR-TRANSLATION := #S
  NONE
    #ERROR-TRANSLATION := #T
  END-DECIDE
END-IF
```

Check for Data Access Code in the PROCESS-SELECTED-RECORD User Exit

There is a high probability that the PROCESS-SELECTED-RECORD user exit contains data access code. Because the object browse dialog is replaced by Web services or pages, check for this code when web enabling to ensure that the code is not lost.

Reference Field Names

To ensure proper referencing, field names must be fully qualified. For example, use NCST-CUSTOMER.CUSTOMER-NUMBER, not just CUSTOMER-NUMBER. Natural Engineer can do this for you.

Use Wildcard Characters for Numeric Fields

For the MOVE-BY-NAME functionality to work correctly, variables generated for the #INPUT statement in the original browse module must match those in the key PDA. In anticipation of wildcard support, the #INPUT statement in the browse module defines the variable as alphanumeric. This feature is not available for numeric fields in the object browse subprogram. In addition, any references to redefined numeric values within the #INPUT statement will not allow the module to be compiled. If the browse module uses the #NUM-inputComponent syntax, it will be converted to inputComponent in the user exit code (because #NUM- is no longer available).

Coordinate Data Areas for Browse Program Modules

After transforming a browse program module, you must ensure that the parameter data area used by the object browse subprogram (CDPDA-D, for example) contains the same fields as the global data area used by the transformed browse program module (CDGDA, for example).

Natural Construct assumes that the following level one structures in the GDA will always be available (see CDGDA for an example):

- DIALOG-INFO
- MSG-INFO
- PASS

As the object browse subprogram has no access to the GDA, Natural Construct supplies the following PDAs:

- CDPDA-D (containing the DIALOG-INFO structure)
- CDPDA-M (containing the MSG-INFO structure)
- CDPDA-P (containing the PASS structure)

This allows data to be easily moved from the global data area for the browse program module to the PDAs for the object browse subprogram. If you have customized your version of the CDGDA global data area, ensure that the fields in each level one structure in CDGDA match those in the CDPDA-P parameter data area for the object browse subprogram (for example, the PASS structure).

For an example of coordinating data areas after customizations, refer to the browse modules in the SYSCSTDE library. These modules use:

- A customized global data area, called NCGDA, which has a different level 1 PASS variable from the standard GDA (called CDGDA)
- A customized copy of the CDPDA-P parameter data area, which is different from the standard PDA found in the SYSTEM library

If you create another browse module in the SYSCSTDE library that uses CDGDA, it will work correctly because no PDA is required. But if you transform the browse module, the generated object browse subprogram will be compiled with the customized copy of the CDPDA-P data area in SYSCSTDE. As this PDA reflects the NCGDA data area, a NAT0935 error (conflicting number of parameters) occurs when the object browse dialog driver program is executed and tries to pass the level 1 PASS variable (as the driver program uses CDGDA).

Modify Transformation Data

The CUTRLDA local data area contains basic transformation data that is not related to specifications. For example, it handles situations where the user exit functionality and variables are similar in the different models but do not have the same names. CUTRLDA contains a list of user exit names, the object name for each exit (if it is different), and the name of the Object model(s) to which the user exit will be transferred. In addition, CUTRLDA contains an array of all variables in the browse module and which variables they

should be mapped to in the transformed object modules. You can modify this LDA to reflect your site requirements, if necessary.

Note:
If you modify CUTRLDA, you must recompile the CUTRPR, CUTRPR1, and CUTRVAL subprograms for the Transform-Browse model. Make all changes in the SYSCST library and then use the Natural SYSMAIN utility to copy the object code to the SYSLIBS library.

Change Error Message When No Records Match Selection

If no records match a selection for the object browse subprogram, an 8004 error message is displayed (to be consistent with the original browse module). You can replace this message with 8074 (the default error message number for an object browse subprogram) by modifying the CBDBD09 code frame.

Use the Find Objects Window to Scan a Library (Natural Plug-in Only)

If you use the Transform Browse wizard in the Natural Business Services Natural plug-in, you can use the **Find Objects** window to scan a library, determine which modules can be transformed, and display them in the editor. For example, you can scan for the following line:

```
**SAG GENERATOR: BROWSE
```

Note:
The example above will also find modules generated by the Browse-Subp, Browse-Select, and Browse-Select-Subp models.