

Natural Business Services

Natural Construct Transform-Browse Model

Version 5.3.1

February 2010

This document applies to Natural Business Services Version 5.3.1.

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

Copyright © 2006-2010 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, United States of America, and/or their licensors.

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

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

This software may include portions of third-party products. For third-party copyright notices and license terms, please refer to "License Texts, Copyright Notices and Disclaimers of Third-Party Products". This document is part of the product documentation, located at <http://documentation.softwareag.com/legal/> and/or in the root installation directory of the licensed product(s).


Table of Contents

1 Natural Construct Transform-Browse Model	1
2 Introduction	3
3 Tips and Techniques	5
Access Maps and Menus from Original Browse Module	6
Transform More Than One Browse Module	6
Transform in a Secure Environment	7
Display a Variable Number of Lines per Record	7
Display Correct Number of Rows	8
Display Direct Command Line on a Screen	8
Use Edit Masks	9
Stop Screens from Advancing Data	9
Check for Data Access Code in the PROCESS-SELECTED-RECORD User Exit	10
Reference Field Names	10
Use Wildcard Characters for Numeric Fields	10
Coordinate Data Areas for Browse Program Modules	10
Modify Transformation Data	11
Change Error Message When No Records Match Selection	12
Use the Find Objects Window to Scan a Library (Natural Plug-in Only)	12
4 Specification Parameters	13
Verify Transformation Specifications Panel	14
Standard Parameters Panel	22
5 Technical Information	25
Naming Conventions for Generated Object Browse Modules	26
Coding Conventions for Browse Keys	27
PF-Key Styles	28
Action Styles	29





1 Natural Construct Transform-Browse Model

Natural Construct Transform-Browse Model describes the Transform-Browse model, which is called the Transform Browse wizard in the Natural Business Services Natural or Eclipse plug-in. This model transforms an existing browse module (generated by the Browse, Browse-Select, Browse-Select-Subp, or Browse-Subp model) into object browse modules (generated by the Object-Browse series of models). The object browse modules retain the functionality of the original browse module and can be used with a terminal screen.

Using Natural Business Services, you can create business services from the transformed object browse subprogram and then use the Visual Studio add-in or Eclipse plug-in to create Web services or client proxy classes.

 **Note:** The Transform-Browse model does not transform helproutine modules (modules generated by the Browse-Helpr or Browse-Select-Helpr models).

Natural Construct Transform-Browse Model covers the following topics:

 Introduction	Provides an overview of the Transform-Browse model (Transform Browse wizard). It contains information on naming conventions for the generated object browse modules, coding conventions for key fields used by the browse, and PF-key and action styles for transformed modules.
 Tips and Techniques	Provides tips and techniques when using the Transform-Browse model (Transform Browse wizard).
 Specification Parameters	Describes the Verify Transformation Specifications panel for the Transform Browse wizard in the Natural Business Services Natural and Eclipse plug-ins, as well as the Standard Parameters panel for the Transform-Browse model.
 Technical Information	Provides internal information on the Transform-Browse model.

2 Introduction

The Transform-Browse model reads the specifications for a browse module (module generated by a Browse, Browse-Subp, Browse-Select, or Browse-Select-Subp model) and generates object browse modules for use in a client/server environment. As the modules generated by Browse models contain a combination of UI, data, and business logic, they cannot be used in a client/server environment. The Object-Browse models, however, generate separate modules for these layers. The Transform-Browse model uses the following models to generate the object browse modules:

- Object-Browse-Subp (generates the module containing data and business logic)



Note: This model also generates the object PDA, key PDA, and restricted PDA.

- Object-Browse-Dialog (generates the module containing the user interface information)
- Object-Browse-Dialog-Driver (generates a driver program for the dialog module)



Note: The driver program maintains the same interface into the object browse module as was used for the browse module. You do not need to modify modules that used the original browse module.

To preserve the original browse module, the Transform-Browse model does the following:

1. Reads the data for the browse module.
2. Moves the browse module to the archived module library.
3. Generates the specifications for the object modules.
4. Generates all the object modules in the current library.
5. Moves the object browse modules to the transformed module library.

By default, the archived module library name is B + the first seven characters in the current library name and the transformed module library name is T + the first seven characters in the current

library name. These standards are implemented in the WTRNAME module and validated in the CUTRVAL module and can be changed if desired.



Note: If Natural Security is being used, you must have access to both libraries (there is a validation check to ensure you have access to both).

3

Tips and Techniques

- Access Maps and Menus from Original Browse Module 6
- Transform More Than One Browse Module 6
- Transform in a Secure Environment 7
- Display a Variable Number of Lines per Record 7
- Display Correct Number of Rows 8
- Display Direct Command Line on a Screen 8
- Use Edit Masks 9
- Stop Screens from Advancing Data 9
- Check for Data Access Code in the PROCESS-SELECTED-RECORD User Exit 10
- Reference Field Names 10
- Use Wildcard Characters for Numeric Fields 10
- Coordinate Data Areas for Browse Program Modules 10
- Modify Transformation Data 11
- Change Error Message When No Records Match Selection 12
- Use the Find Objects Window to Scan a Library (Natural Plug-in Only) 12

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

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: FO1CSTAPPL 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.

4 Specification Parameters

- Verify Transformation Specifications Panel 14
- Standard Parameters Panel 22

You can use the Transform-Browse model:

- on the server using the NCSTG command
- Natural Business Services Natural plug-in
- Natural Business Services Eclipse plug-in



Note: Although you can generate the Transform-Browse model on the server, we recommend that you use the Transform Browse wizard in one of the Natural Business Services plug-ins.

The Transform Browse wizard has one specification panel: **Verify Transformation Specifications**. The Transform-Browse model has one specification panel: Standard Parameters.

Verify Transformation Specifications Panel

This section describes the **Verify Transformation Specifications** panel for the Transform Browse wizard for the following Natural Business Services (NBS) plug-ins:

- [Eclipse Plug-in](#)
- [Natural Plug-in](#)



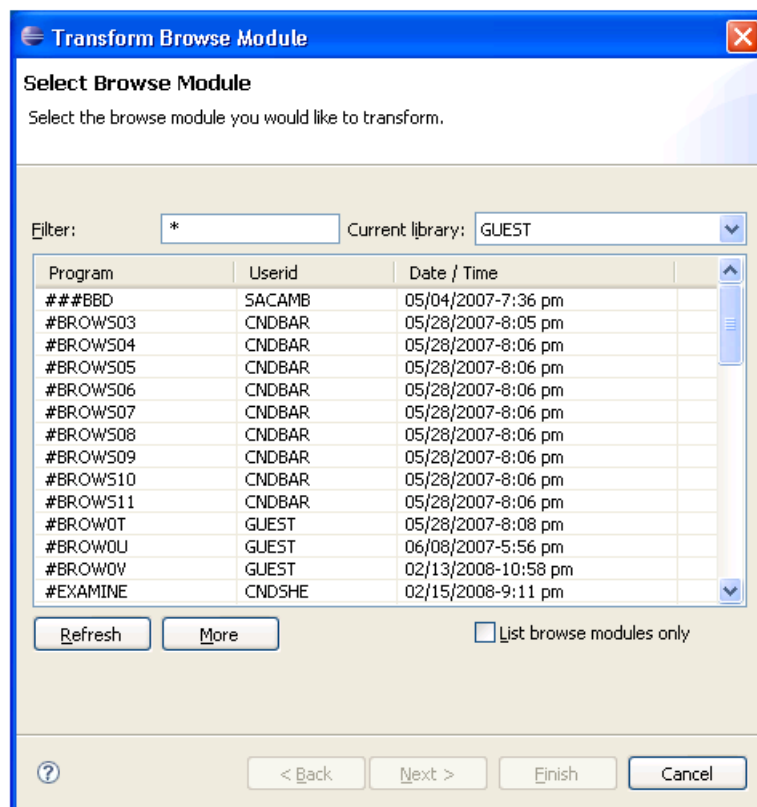
Note: Before transforming a module for the first time, read [Tips and Techniques](#) for helpful information.

Eclipse Plug-in

▶ **To transform a browse module in the NBS Eclipse plug-in:**

- 1 Open the context menu for the SPoD connection in the **NBS Repositories** view.
- 2 Select **Transform browse module**.

The **Select Browse Module** panel is displayed. For example:



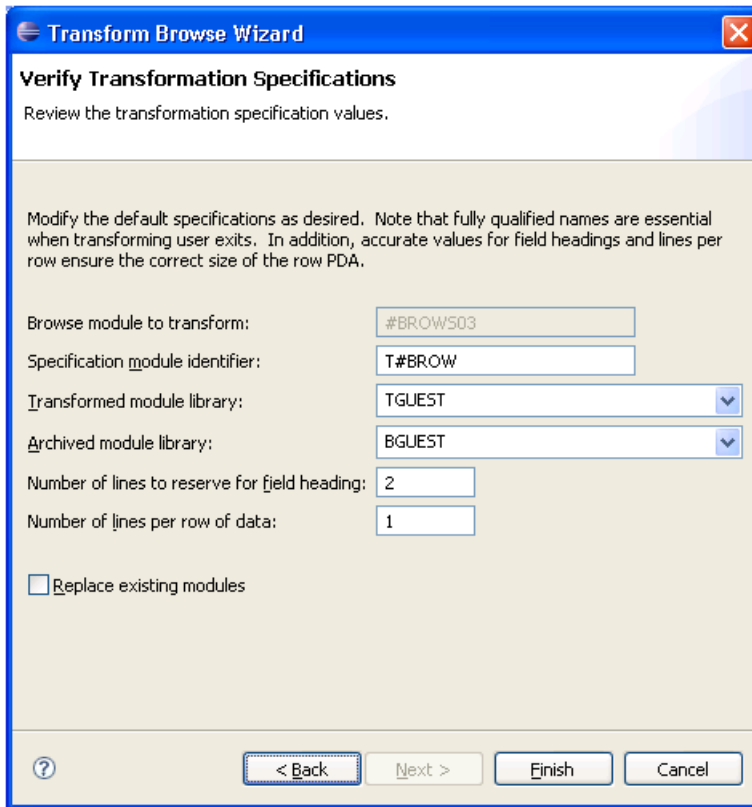
This panel lists the first 50 modules in the library indicated.

- If there are more than 50 modules, select **More** to list the additional modules.
- If you only want to display browse modules, select **List browse modules only** and then select **Refresh**.

3 Select the browse module you want to transform.

4 Select **Next**.

The **Verify Transformation Specifications** panel is displayed. For example:



This panel displays the default specification values for the transformation process. The input fields on this panel are:

Field	Description
Specification module identifier	Name of the transform module you are creating.
Transformed module library	Name of the library in which the transformed object browse modules are moved. This name is B + the first seven characters in the current library name.
Archived module library	Name of the library in which the original browse module is moved before the transformation process begins. By default, this name is T + the first seven characters in the current library name.
Number of lines to reserve for field heading	Number of lines reserved for field (column) headings. The default is two lines. Tip: Ensure that the correct number of lines is entered in this field. Although the transformation will not fail if the number is inaccurate, problems will become apparent at runtime. For example, if too many data rows are available for a screen, the input line and a data line can overlap at runtime. For more information, see Display Correct Number of Rows .

Field	Description
Number of lines per row of data	Number of lines reserved per row of data. The default is one line. Tip: 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, do not change the default for this field and define the AFTER-INPUT user exit for the Object-Browse-Dialog model. For information, see Display a Variable Number of Lines per Record .
Replace existing modules	If this option is selected, modules with the same name are replaced in the libraries shown.

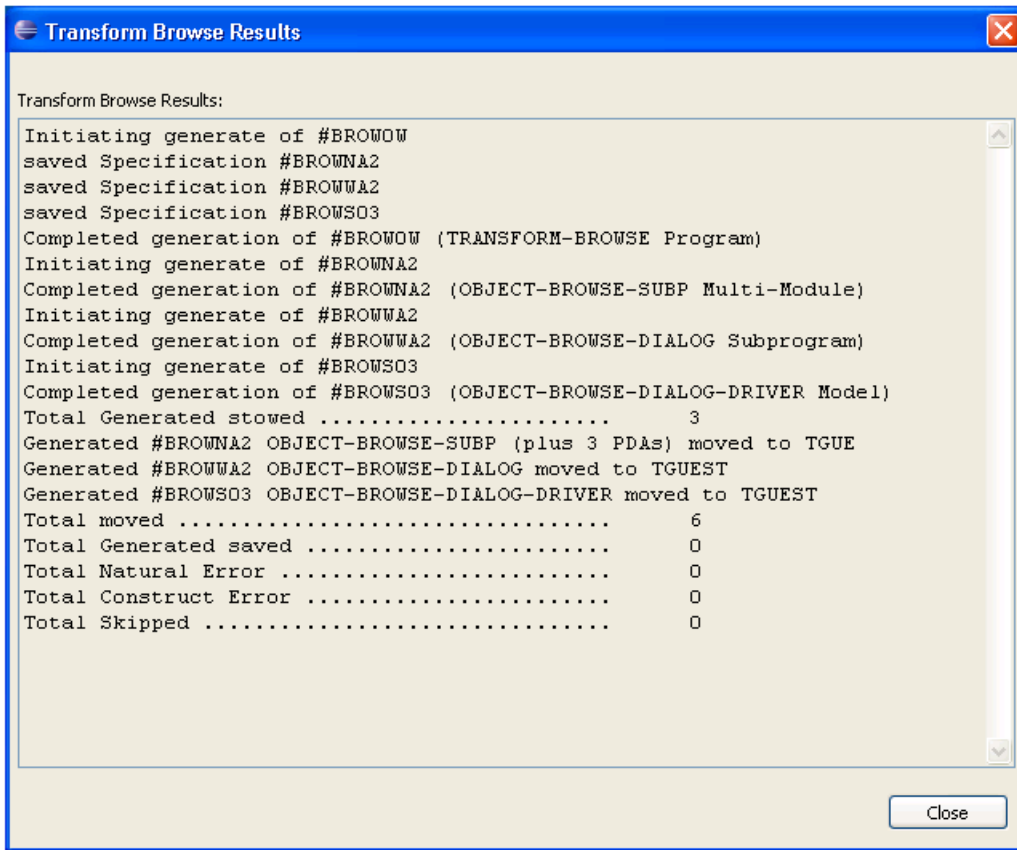
- 5 Verify the specifications that will be used for the transformation.

If you change these values, ensure the following guidelines are met:

- Use fully qualified names to ensure that user exits are transformed successfully.
- Specify accurate values for field headings and lines per row to ensure that the row PDA is the correct size.

- 6 Select **Finish** to transform the browse module.

The **Transform Browse Results** window is displayed. For example:



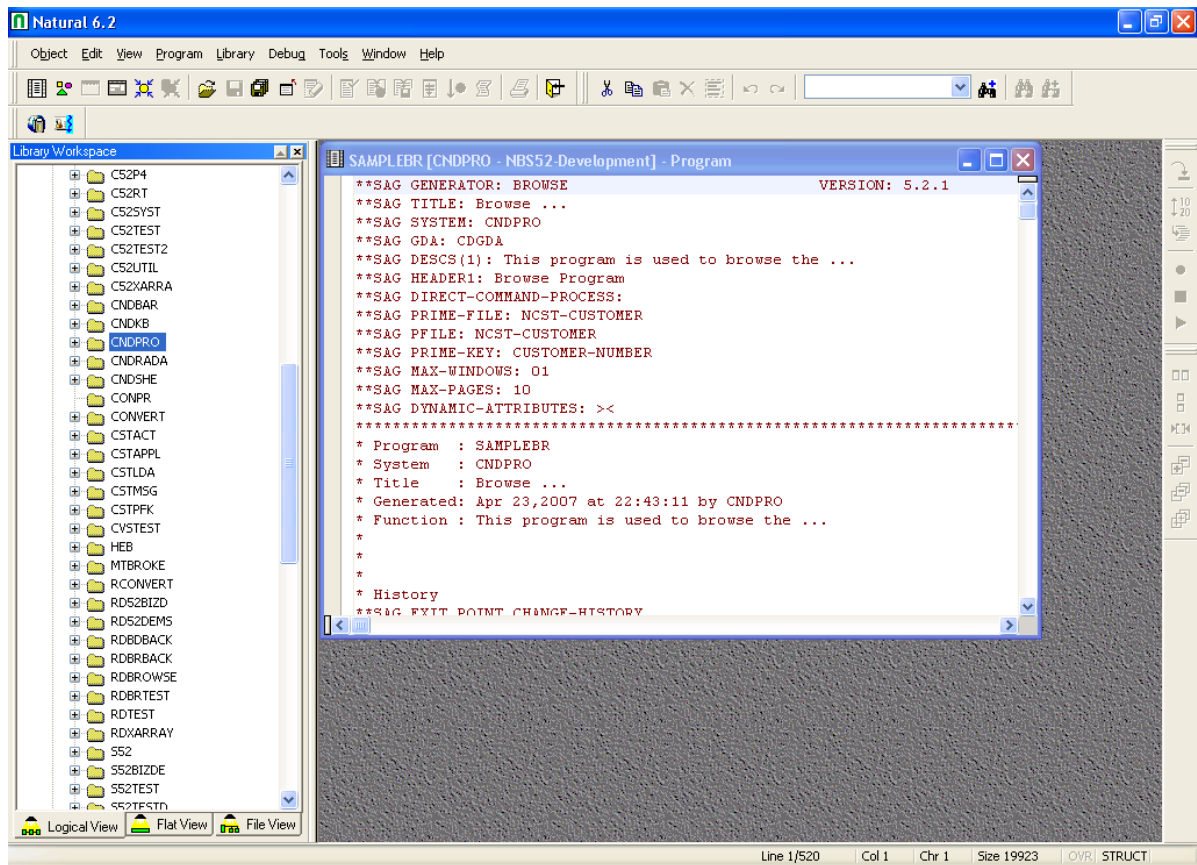
- 7 Review the results of the transformation.
- 8 Select **Close** to close the window.

Natural Plug-in

► To transform a browse module in the NBS Natural plug-in:

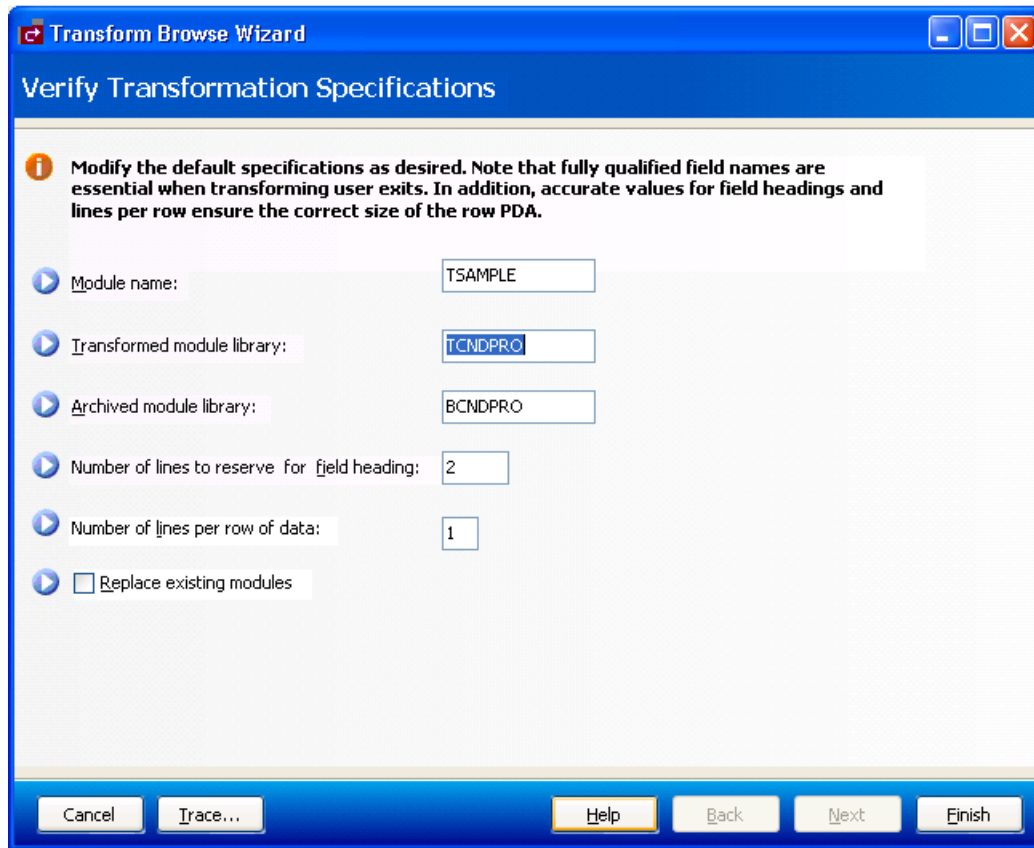
- 1 Display the browse module in the program editor (for information, see the Natural for Windows documentation).

For example:



- 2 Open the context menu for the module.
- 3 Select **Transform Browse**.

The **Verify Transformation Specifications** panel is displayed. For example:



This panel displays the default specification values for the transformation process. The fields on this panel are:

Field	Description
Module name	Name of the transform module you are creating.
Transformed module library	Name of the library in which the transformed object browse modules are moved. This name is B + the first seven characters in the current library name.
Archived module library	Name of the library in which the original browse module is moved before the transformation process begins. By default, this name is T + the first seven characters in the current library name.
Number of lines to reserve for field heading	Number of lines reserved for field (column) headings. The default is two lines. Tip: Ensure that the correct number of lines is entered in this field. Although the transformation will not fail if the number is inaccurate, problems will become apparent at runtime. For example, if too many data rows are available for a screen, the input line and a data line can overlap at runtime. For more information, see Display Correct Number of Rows .

Field	Description
Number of lines per row of data	Number of lines reserved per row of data. The default is one line. Tip: 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, do not change the default for this field and define the AFTER-INPUT user exit for the Object-Browse-Dialog model. For information, see Display a Variable Number of Lines per Record .
Replace existing modules	If this option is selected, modules with the same name are replaced in the libraries shown.

- 4 Verify the specifications that will be used for the transformation.

If you change these values, ensure the following guidelines are met:

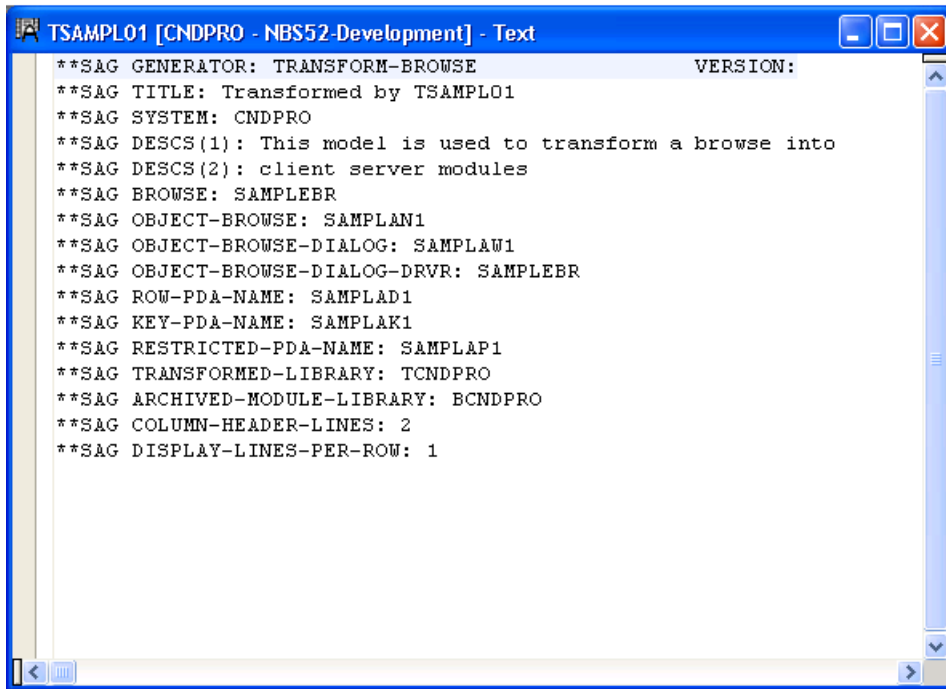
- Use fully qualified names to ensure that user exits are transformed successfully.
- Specify accurate values for field headings and lines per row to ensure that the row PDA is the correct size.

- 5 Select **Finish**.

The **Generate Status** window is displayed. If any of the transformed modules cannot be compiled, the module with the problem is displayed.

- 6 Select **OK**.

The results of the transformation are displayed in the editor. For example:



This window displays the names of the modules generated by the Transform Browse wizard and other specification values. You can invoke the Transform Browse wizard from the context menu for the module and change the specifications (if required). If the transformed modules can be successfully compiled, they are moved to the transformed module library and you can select the next browse module for transformation.

Standard Parameters Panel

CUTRMA	TRANSFORM-BROWSE Program	CUTRMA0
Nov 07	Standard Parameters	1 of 1
Module	ACUSTN__	
System	CXT341S_____	
Title	Transform browse module__	
Description	This model is used to transform a browse into_____	
	client server modules_____	

Browse	_____ *	Field heading lines. 2
Transformed module library. _____		Replace existing modules .. _
Archived module library ... _____		Display lines per row 1

```

Object browse subp ..... _____
Object browse dialog ..... _____
Object PDA ..... _____
Key PDA ..... _____
Restricted PDA ..... _____
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
main help retrn quit                                     main

```

The fields in the upper portion of this panel are similar for all models. For a description of these fields, see Common Fields on the Standard Parameters Panel.

The fields in the lower portion of this panel are:

Field	Description
Browse	Name of the browse module you want to transform into object browse modules. Field-level help is available to select a module.
Field heading lines	Number of lines reserved for field (column) headings. The default is two lines. Tip: Ensure that the correct number of lines is entered in this field. Although the transformation will not fail if the number is inaccurate, problems will become apparent at runtime. For example, if too many data rows are available for a screen, the input line and a data line can overlap at runtime. For more information, see Display Correct Number of Rows .
Transformed module library	Name of the library in which the transformed object browse modules are moved. By default, this name is B + the first seven characters in the current library name.
Replace existing modules	Indicates whether to replace modules with the same name in the libraries shown.
Archived module library	Name of the library in which the original browse module is moved before the transformation process begins. By default, this name is T + the first seven characters in the current library name.
Display lines per row	Number of lines reserved per row of data. The default is one line. Tip: 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, do not change the default for this field and define the AFTER-INPUT user exit for the Object-Browse-Dialog model. For information, see Display a Variable Number of Lines per Record .
Object browse subp	Name of the transformed object browse subprogram module.
Object browse dialog	Name of the transformed object browse dialog module.
Object PDA	Name of the transformed object browse parameter data area module. The object PDA contains one field for each field defined in the specified Predict view. These fields are defined within a 1:V structure so the object browse subprogram can support an arbitrary number of return rows.
Key PDA	Name of the transformed object browse key (search) parameter data area module. The key PDA defines the union of all fields that are components of a logical key.

Field	Description
	Additionally, the generated key PDA contains a field that can be used to begin the browse at a specific record.
Restricted PDA	Name of the transformed object browse restricted parameter data area module. The restricted PDA stores data, such as the last sort key, the last starting value, the last row returned, etc. so that the next set of consecutive records is returned to the caller. The contents of this data area should not be altered by the calling module.

After selecting the browse module and entering your specifications on this panel, press Enter to return to the Generation main menu and generate the modules.

5 Technical Information

- Naming Conventions for Generated Object Browse Modules 26
- Coding Conventions for Browse Keys 27
- PF-Key Styles 28
- Action Styles 29

The Transform-Browse model retrieves the specifications and user exits for a browse module and transforms this information into the specifications for the object browse modules within the program editor. The model takes advantage of the multi-model feature of Natural Construct to:

- Parse the contents of the editor to determine the specifications for each model
- Save the specifications to individual Natural modules
- Copy the information to the NCST-WORK view of the Natural Construct LFILE
- Read the NCST-WORK file to determine which Natural modules to read into the editor
- Generate and stow the modules
- Move the stowed modules to the transformed module library (using the reporting feature)



Note: The Transform-Browse model does not support user exit processing because it only generates text specifications for the object browse modules. The model copies the user exit specifications for the browse module into user exits for the appropriate object browse module.



Tip: If the browse module is a program, run the program to determine how many lines are used for the field headings. By default, the Transform-Browse model assumes that field headings use two lines. If this is not accurate, it will affect the number of rows displayed on the screen after transformation. For more information, see [Display Correct Number of Rows](#).

This section covers the following topics:

Naming Conventions for Generated Object Browse Modules

To name the generated object browse modules, the Transform-Browse model first determines whether the module names have been specified on the Standard Parameters panel. If these names are blank, the model uses the first five characters in the name of the original browse module + a suffix + a number from 1 to 99 to make the name unique in both the current library and transformed module library (for example, if BINSN already exists in the transformed module library then BINSN1 is used).

The following table lists each module generated by the Transform-Browse model and the default suffix used in the generated module name:

Module Name	Suffix
Object browse subprogram	N
Object browse dialog	W
Object browse dialog driver	X
Object PDA	D
Key PDA	K
Restricted PDA	P

To see the naming conventions for your site, logon to the SYSCST library and execute the CTENAME driver program.



Note: For information on changing these conventions, see *Modify/Test the Naming Conventions for Natural Objects*.

Coding Conventions for Browse Keys

To implement Browse model functionality in the Object-Browse models, the Transform-Browse model creates internal specifications that are only accessible if the browse module has been transformed. This functionality includes:

- [Ascending and Descending Sort Order](#)
- [Key Prefixes](#)
- [Minimum and Maximum Key Values](#)

Ascending and Descending Sort Order

To handle ascending and descending sort order, the object browse subprogram is generated with two logical keys: *A-keyname* and *D-keyname*. Next, an object browse dialog is generated to use either the *A-keyname* or *D-keyname* key, depending on which option a user requests. From the user's perspective, the functionality is the same as was in the browse module.

Key Prefixes

To handle the prefix concept for browse key components, the object browse subprogram specifies the leading components in the CDBRPDA parameter data area. In addition, it handles the prefix bytes by specifying a starting with value. All of this is automatically handled by the object browse dialog based on the information passed in from the browse specifications.

Minimum and Maximum Key Values

To establish a logical start and/or end of file for a browse subprogram, you must specify minimum and/or maximum key values. The minimum (starting value for the browse) and maximum (ending value for the browse) key values create a subset of records within the file. The program will not browse before or after these values.

The only way to use minimum and/or maximum key values in an object browse subprogram is via the Transform-Browse model as this option is only available for the first two keys (where the first one is a physical key in logical ascending order and the second one is a physical key in logical descending order). In addition, the object browse subprogram only generates code for the input criteria for the first and second key when minimum and/or maximum key values are specified in the original browse module.



Note: If these options were not specified in the original browse module, the object browse subprogram generates the input criteria for all keys.

To allow greater flexibility when using minimum and/or maximum key values for additional keys, the READ-INPUT-CRITERIA user exit is available for the Object-Browse-Subprogram model. You can use this exit to write code similar to the first and second key for any additional keys. For information, see *READ-INPUT-CRITERIA, Natural Construct Generation*.

PF-Key Styles

By default, a generated object browse dialog module and browse module use different styles of PF-keys. To solve this problem, the Transform-Browse model generates the object browse dialog module with the browse style of PF-keys.

During transformation, the Transform-Browse model creates the following specification for the dialog:

```
**SAG USE-BROWSE-PFKEYS: X
```



Note: If the object browse dialog module uses the browse-style PF-keys, the PF6 (pfkey) option is removed from the Standard Parameters panel (accessed using the NCSTG command) for the dialog. If the module does not use browse-style PF-keys, PF6 (pfkey) is available on the panel.

For information on changing the default style of PF-keys for an object browse dialog module, see the *Change the Default PF-key Style* section in Object-Browse-Dialog Model.

Action Styles

By default, a transformed object browse dialog module has one of the following action styles:

- no actions

If a browse module was transformed, the dialog has no actions.

- browse-select style actions

If a browse-select module was transformed, the dialog uses the browse-select style actions (instead of the object browse dialog-style actions).

