OPENTEXT™

TEAM DEVELOPER™

Localizing & Customizing SQLWindows Applications

Product Version 7.0

Team Developer™: Localizing & Customizing SQLWindows Applications

Open Text Corporation

275 Frank Tompa Drive, Waterloo, Ontario, Canada, N2L 0A1
Tel: +1-519-888-7111 Toll Free Canada/USA: 1-800-499-6544 International: +800-4996-5440
Fax: +1-519-888-0677
Support: http://support.opentext.com
For more information, visit https://www.opentext.com

Table of Contents

Preface

Localizing and Customizing SQLWindows Applications describes what Object Nationalizer™ can do and how you can use it. In this preface, you find information about:

- Who should read this book
- What is in this book
- Notation conventions

Who should read this book

Localizing and Customizing SQLWindows Applications is written for anyone who is localizing (translating) a SQLWindows application into another language, or modifying the appearance (but not the logic) of a SQLWindows application.

Note: You do not have to be a programmer to use Object Nationalizer.

What is in this book

This book describes what Object Nationalizer can do and how you use it.

- To get an overview of Object Nationalizer, read *Chapter 1, Introduction*.
- To learn how to install, start, and use Object Nationalizer, read *Chapter 2, Using Object Nationalizer*.
- To see some examples of how to use Object Nationalizer, read *Chapter 3, Examples*.
- To look up the function of a menu item, refer to *Chapter , Menu Reference*.

Summary of chapters

This book is organized as follows:

1	**Introduction**	Gives an overview of Object Nationalizer.
2	**Using Object Nationalizer**	Describes how to install and start Object Nationalizer, and how to use it to localize an application or customize its user interface.
3	**Examples**	Provides short examples of how to do certain tasks with Object Nationalizer.
A	**Menu Reference**	Describes the menus and the menu items available in Object Nationalizer.

Notation conventions

The table below show the notation conventions that this book uses.

Notation	Explanation
You	A translator or programmer who reads this book
User	The end-user of the applications that you (the programmer) write or that you (the translator) localize.
bold	Menu items, push buttons, and field names (things that you select)
`Call SqlCommit`	SQLWindows or C language code example
q	Start of a procedure
SQL.INI MAPDLL.EXE	Program names and file names
Precaution	**Warning:**
Vital information	**Important:**
Supplemental information	**Note:**
Alt+1	Keystroke sequence. A plus sign between key names means to press and hold down the first key while you press the second key
TRUE FALSE	Boolean constants defined internally in SQLWindows: **Constant** **Value** **Meaning** TRUE 1 Successful, on, set FALSE 0 Unsuccessful, off, clear

Chapter 1. Introduction

This chapter describes what Object Nationalizer can do for you.

Use Object Nationalizer to localize (translate) Team Developer applications from one language to another without changing the application source code; all you need is the executable file (and dynalib files, if any) that the programmer builds using Team Developer.

You can also use Object Nationalizer to modify the appearance of an application's user interface. For example, you can resize the various components of the user interface (such as push buttons, background text, list boxes, and so on) to accommodate translations that are too large to fit on the existing components.

How Object Nationalizer works

The executable code of a Team Developer application contains not only the logic of the application, but also attributes known as *resources*. These attributes are merely cosmetic in nature—changing them does not change the logic of your application.

You use Object Nationalizer to extract, view, and edit the resources in an application. It is because Object Nationalizer can extract resources directly from an executable or dynalib that you can modify the appearance and language of an application without changing (or even having access to) the application source code (the outline).

About resources

Resources are attributes associated with the visual elements of an application that affect only the appearance of those elements.

For example, the background color or font used in a data field affects only the appearance of the data field—these attributes are resources. This is different from the type of a data field (numeric, string, and so on), which is tied to the logic of the program—the type of a data field is not a resource.

Other typical application resources are window titles, menu items, status bar text, messages (when defined as string constants), table column titles, and labels for such things as push buttons, radio buttons, and check boxes. (Read the book *Developing with SQLWindows* for more information about resources.)

Important: Object Nationalizer allows you to change only the resources in an application; you cannot change anything that can affect the logic of the application.

What you can do with resources

Using Object Nationalizer, you can:

- View resources.
- Translate text resources.
- Change the resources that affect the size or position of components in the user interface.
- Save resources to a file in binary format.
- Save resources to a file in human-readable format.
- Apply resources to an executable file or dynalib.

Localizing applications

In the typical scenario for translating a Team Developer application, a translator is set up to use Object Nationalizer, then given a copy of the executable file and dynalibs to translate.

The translator opens the application files and extracts the resources into the Object Nationalizer editor/designer. Using the editor/designer, the translator translates the application text and resizes components as needed. Finally, the translator applies the changed resources back into the application.

When you now run the application, it displays text in the target language.

Note: The Object Nationalizer editor/designer can display translated text for Western European languages. To view translations into other languages (such as Japanese, Hebrew, Russian, and so on), you must obtain a version of Object Nationalizer that has been adapted to display the characters for that language.

Modifying the user interface

Even if you are not translating an application, you can use Object Nationalizer to modify an application by changing its appearance.

In general, you can use the Object Nationalizer editor/designer to move and resize components, change picture formats for data fields, show components that were hidden or hide components that were visible, replace icons and graphics, and so on.

By using Object Nationalizer to make these changes (rather than SQLWindows), you guarantee that you do not inadvertently change the logic of the application.

Files you can localize

You can use Object Nationalizer to localize executable (.exe) files or dynalib (.apd) files built using SQLWindows.

Important: Read the release notes to find out which version of Team Developer you must use to create the .exe or .apd file that Object Nationalizer can process.

When you open one of these files, Object Nationalizer extracts the resources into the editor/designer where you can localize and modify them.

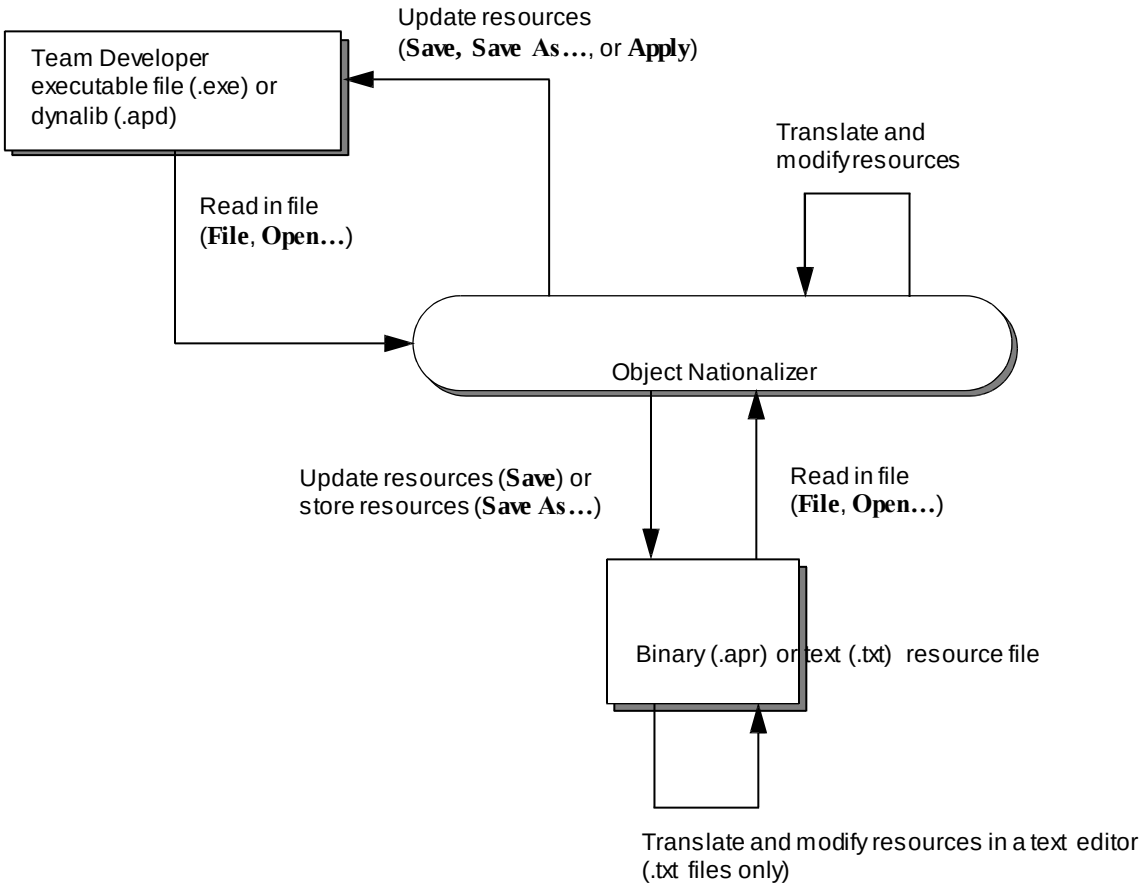
Once you have made your changes to the application's resources, you can update the executable file or dynalib with the modified resources, save as a new executable or dynalib with the modified resources, or save the modified resources into a resource

file. The resource file can be in either binary (.apr file) or human-readable (.txt file) format.

Important: Before programmers build the .exe or .apd files that are to be modified with Object Nationalizer, they must check the Enable Resource Editing box in the Build Settings dialog box when they build the application with SQLWindows.

Overview of the development process

The diagram below shows you the overall flow as you might use Object Nationalizer to localize or modify an application.



The role of resource files

After you open a SQLWindows application with Object Nationalizer, you modify its resources in one of two ways:

- You work directly with the application's resources while you have the application open.
- You save a copy of just the resources in a resource file, then close the application and open the resource file to modify the resources.

The rest of this section summarizes how you typically modify resources in either of these two ways. It also tells you how you can use resource files to make it easier to localize subsequent versions of an application and to implement dynamic binding of resources to an application.

Modifying resources with the application open

1. The programmer builds the first version of the application to produce an executable (.exe file) and perhaps one or more dynalibs (.apd files).
2. The translator opens the application executable or dynalib with Object Nationalizer.
3. Using the Object Nationalizer editor/designer, the translator localizes the textual portions of the resources into the target language, resizes resources, and replaces icons and graphics as needed.
4. The translator saves the changes made to the resources in one of two ways: either back into the original executable or dynalib (using **File**, **Save**) or to a copy of the executable or dynalib file, but with the changed resources and with a different file name (using **File**, **Save As…**).

Modifying resources in a resource file

1. The programmer builds the application to produce an executable (.exe file) and perhaps one or more dynalibs (.apd files).
2. The programmer opens the application executable or dynalib with Object Nationalizer, saves the resources in binary format (.apr file), then gives the file to a translator.
3. The translator opens the .apr file with Object Nationalizer.
4. The translator localizes the textual portions of the resources into the target language, resizes resources, and replaces icons and graphics as needed.
5. The translator saves the changes to the .apr file and returns the file to the programmer.

6. The programmer opens the .apr file in Object Nationalizer and applies the resources in the .apr file to the executable or dynalib.

Note: The steps a programmer performs and the steps a translator performs in the localization process is, of course, determined by how your organization decides to divide up these tasks. The descriptions above are assumed to be typical of many organizations.

Making repeat localizations easier

If you save your localizations in a resource file, then apply the translated resources to the application, you can keep the resource file to reuse later.

One common reason for reusing a resource file is to make localizing new versions of the same application easier. For example, suppose you have localized version 1.0 of your application into several different languages, and you have saved the localizations in resource files. Later, you create a new version (1.1) of the same application. When you begin localizing version 1.1, you can start off by applying the existing resource files to the new version.

All the resources that still exist in the new version of the application will be localized appropriately. All resource that were deleted in the new version will be ignored when you apply the resource file. All you will have left to localize are any new resources that were added to the new version of the application.

Once you have applied the old resource file to the new application, load the application into Object Nationalizer. Translate the remaining resources (the new ones), then save the resources into a resource file. This new resource file will then have all of the translated resources for the new version of the application.

Applying localizations at runtime

You can use resource files to delay the applying of resources until the application actually runs.

In this scenario, you create multiple .apr files (with different names), each containing the translations for a single language. The advantage of this approach is that you can then deploy one version of the application along with all of the localized resource files you create. When the application starts up, it needs to detect the language environment in which it is running, then load the resources from the appropriate resource file before continuing.

Thus, instead of distributing different .exe files to your users (one for each language), you distribute a single .exe file along with several .apr files (one for each language).

There are two requirements for this to work:

- The translator must save the localized resources in .apr files without applying the resources back into the application.
- The application programmer must include a small amount of special code to detect the language environment in which the application is running, then load the resource file with the translations for the appropriate language.

What the programmer must do

To apply resources at runtime, the programmer must call the new function SalResourceSet in the On SAM_Startup section of the application outline.

The syntax of this function is:

```
bOk = SalResourceSet(strResourceFile, nReserved)
```

This function takes two parameters: a string parameter with the pathname of the resource file to load, and a number parameter that must be set to zero (this parameter is reserved for future use).

SalResourceSet returns TRUE if it succeeds and FALSE if it fails.

When called, SalResourceSet loads the resources from the specified resource file and modifies the application's resources stored in memory (RAM). The application files on disk (.exe and .apd) are not modified.

Here is an example of the code the programmer might write in the SQLWindows outline:

```
Application Description: Dynamic Localization Example  Libraries
  Global Declarations Window Defaults Formats
              External Functions
      Library name: Kernel32.dll  Function:
        GetUserDefaultLangID
          Description: Export Ordinal: 0
          Returns
            Number: WORD Parameters
    Constants System
        !LANG_ constants defined in winres.h  Number: LANG_English =
        0x09
        Number: LANG_French = 0x0C
```

```
  User Resources Variables
  Number: nPrimaryLangID Internal Functions
Named Menus
Class Definitions Application Actions
  On SAM_AppStartup
    Set nPrimaryLangID = GetUserDefaultLangID() & 0xFF  Select Case
    nPrimaryLangID
      Case LANG_French
        Call SalResourceSet('french.apr', 0)  Break
      Case LANG_English
        Call SalResourceSet('english.apr', 0)  Break
```

The constants used to designate various languages come from the Windows header file `winres.h`.

This example assumes that the application is being deployed with two resource files: french.apr and english.apr. (The resource file names are arbitrary—you can name them whatever you like).

Testing the localized application

Once you have applied the translated and modified resources to the application, you need to test the results. Run the application and view every window and dialog box to make sure everything that should be was translated correctly and is readable. You should also verify that all text which should not be translated has not been.

Important: Programmers should bear in mind that, if an application that calls SalResourceSet is run inside of Team Developer, the application outline stored in memory is modified to reflect the resources loaded by that function. If you then save the application, you overwrite the original resources.

Target language platform

You need to do localization and testing on a suitably configured system. Such a system must have all the appropriate hardware (such as a language-specific keyboard and video display). It should also be running the appropriately localized version of Windows. Finally, configure the Regional Settings under Control Panel appropriately for the target language and country.

Translating without Object Nationalizer

It is possible for you to have a translator without access to Object Nationalizer localize the text in an application. To do this, you save the resources in human- readable form (a .txt file), then give the .txt file to the translator to edit using just a text editor.

This approach has serious drawbacks, however. The translator must know exactly which parts of the file to translate and must be extremely careful not to translate those portions that must be left alone. If the translator changes any part of the .txt file that he or she should not, Object Nationalizer will not be able to use the file to apply the translations to the application.

To help make modifying the .txt file a little safer, Object Nationalizer delimits those portions of the file that can be localized with curly braces ("{" and "}"). To translate a string, leave the curly braces intact and translate everything in between them. Be aware, however, that translating a .txt file is still potentially error-prone even with the help of the curly brace delimiters.

Note: A string can span multiple lines—everything after the opening curly brace is part of the string up to the closing curly brace. To denote a literal closing curly brace in a string, you must escape it with a backslash by entering "\}".

Another serious problem with translating .txt files is that the translator cannot check to see if the components on which translated text will appear are large enough to show all of the translated text.

For these reasons it is highly desirable to always use Object Nationalizer to translate the text in a Team Developer application.

Chapter 2. Using Object Nationalizer

This chapter describes how to install, start, and use Object Nationalizer.

Getting ready

This section briefly describes what you should know or do before you can actually begin localizing a Team Developer application using Object Nationalizer.

System requirements. The hardware and software system requirements for installing Object Nationalizer are the same as for Team Developer.

Installing Object Nationalizer. Please read the release notes to learn how to install Object Nationalizer.

Preparing to use Object Nationalizer. Object Nationalizer has a user interface which is very similar to that used in Team Developer. You will find it much easier to use Object Nationalizer if you first familiarize yourself with the Team Developer interface by doing the tutorials presented in the Team Developer book *Introducing Team Developer*.

Starting Object Nationalizer. To start Object Nationalizer, click the **Start** button, then select **Programs**, **Gupta**, **Team Developer 6.2**, **Object Nationalizer 6.2**.

Opening a file. Once you have started Object Nationalizer, open the file to localize (.exe or .apd file) or the resource file in which you saved an application's resources (.apr or .txt file) by selecting **File**, **Open**, then select the file.

Important: If resource editing was not enabled when a .exe or .apd file was built, Object Nationalizer refuses to open the file and displays an error message. To learn how to build a .exe or .apd file with resource editing enabled, read *Creating the sample executables* on page *3-2*.

Automating Object Nationalizer with command-line arguments

In some development environments it may be useful to run Object Nationalizer with command-line arguments, so that the interface opens with some steps already accomplished. This is not a true batch-mode solution, since there are still some manual steps required within the interface to finish your work. But a command line with an argument can accomplish the tasks described in "Starting Object Nationalizer" and "Opening a file", above.

Object Nationalizer accepts a single command-line argument. That argument contains the name of a file. It can be any file which would be permitted in the File, Open task of the interface. Here is an example of a command line:

```
cni62.exe mychanges.apr
```

In this example, the .APR file would be opened, and since an .APR file also specifies the .EXE or .APD file upon which it is based, that application would also be

available. You could then use the interface to make further changes, or to apply the changes to an application as described below in *Saving your changes*.

Localizing text resources

You can localize (translate) the text portions of an application's interface in either of two ways: by translating the application text directly or by modifying resources graphically. This section describes both methods.

Translating the application text

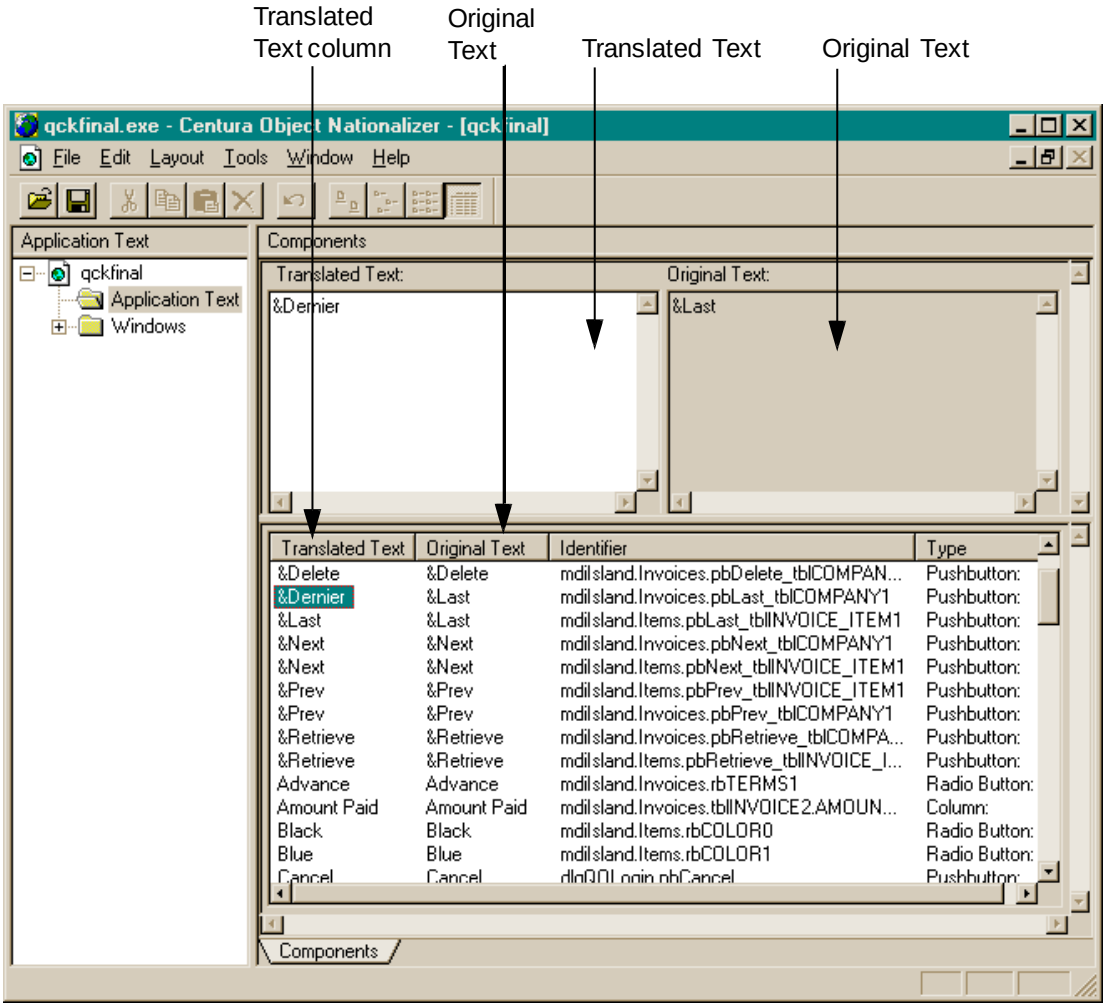
Object Nationalizer provides an editor you can use to just translate the textual elements of an application, leaving everything else in the application unchanged. Translators can use this editor to do fast, bulk editing of the strings in the application into the target language.

Using the Object Nationalizer editor

1. Select the Application Text node in the explorer tree (displayed in the left pane of the Object Nationalizer window).
2. Select an item in the Translated Text column in the bottom half of the right pane. This automatically places the same text in the Translated Text: pane (located in the upper half of the right pane).

   The list is ordered alphabetically according to the strings in the Translated Text column. To find a specific string, scroll through the list, or select **Edit**, **Find**.
3. Click in either the Translated Text column or the Translated Text window pane and translate the text.
4. Repeat the preceding two steps until done.

This is how things look once you have translated text (in this case, "Last" to "Dernier" in French). The translation appears in both the Translated Text pane and in the Translated Text column of the lower pane; the original text appears in the Original Text pane and the Original Text column of the lower pane.



To do the same translation multiple times, select **Edit**, **Replace**. Put the original text in the Find What field and its translation in the Change To field. Click **Find Next** to locate the next occurrence of the original text, then click **Change** to translate it. To translate all occurrences of the original text in one operation, click **Change All** instead of **Change**.

Warning: If you click **Change All** and then discover you have changed too much, you cannot undo this operation. If you are not sure it is safe to click **Change All**, alternate between clicking **Change** and **Find Next** so you can control the operation one change at a time. If you find text that you do not want to change, click **Find Next** again instead of clicking **Change** to continue searching for the text in the Find What field.

About the right window pane

When you select **Application Text** in the explorer tree, the right pane displays a single tab labeled **Components**. The bottom half of the window pane is divided into four columns:

Translated Text—This is the text as you have modified it most recently.

Original Text—This is the text extracted from the file you opened.

Identifier—This is the SQLWindows identifier of the component with which the text is associated.

Type—This is the type of user interface element with which the original text is associated. Examples of values that appear here are Pushbutton, Radio Button, Background Text, and Form Window.

Modifying resources graphically

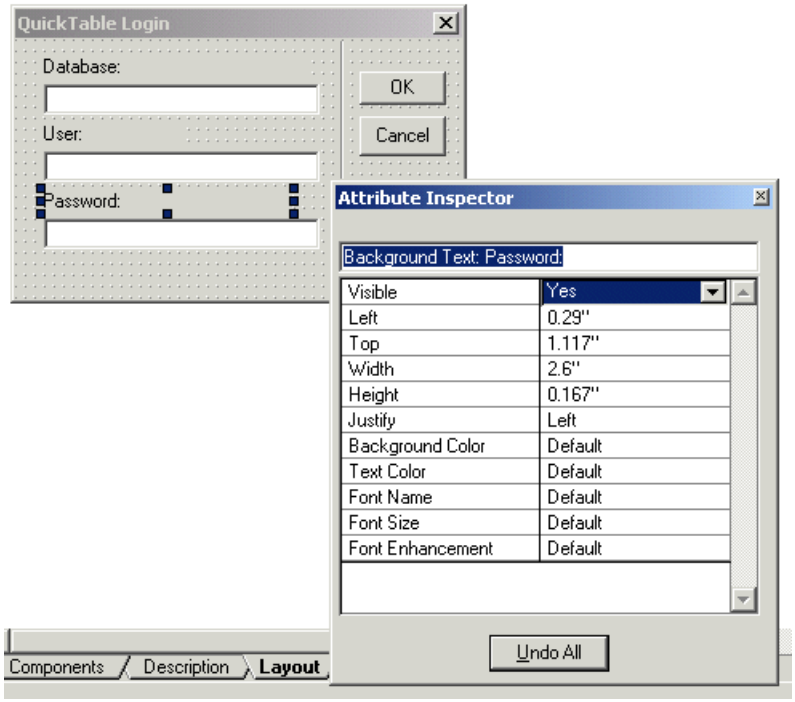
You can use Object Nationalizer to view and modify an application graphically. You display a dialog box or window much as it will appear at runtime, then use the built-in tools to modify the appearance of the dialog box or window (including translating the visible text).

To modify the resources in an application graphically, select a dialog box, MDI window, form window, or table window in the explorer tree. Select the **Layout** tab at the bottom of the right pane to see the dialog box or window as it was designed by the application developer.

For example, to graphically modify the QuickTable Login dialog box of the Island Outfitters sample application (located in the Samples directory of your Team Developer installation):

1. Expand the explorer tree to show the dlgQOLogin dialog box component.
2. Select the dlgQOLogin component in the explorer tree, then select the Layout tab in the right window pane.
3. In the right window pane, right-click on the Password background text and choose Properties from the context menu. The Attribute Inspector appears, with a text box in which you can replace the existing object title (**Password:**) with the desired translation.

The following screen shot shows what you see after performing these steps.



4. Type the translation in the text box, then click the close icon on the Attribute Inspector window. The background text resource changes from **Password:** to the translation you entered.

Another way to change the same text graphically is:

1. Expand the explorer tree to show the dlgQOLogin dialog box component, then select it.
2. Select the Password: background text in the right pane, pause without moving the mouse, then click again to make the text editable.
3. Translate the text, then press **Enter**.

A third way to change the same text graphically is:

1. Expand the explorer tree to show the dlgQOLogin dialog box component.
2. Expand the dlgQOLogin component, then expand the Child Windows component underneath it.

3. Select the Password: background text component in the explorer tree, pause without moving the mouse, then click again to make the text editable.

4. Translate the text, then press **Enter**.

Resizing a component

Once you translate the text, you may discover that the component on which the text appears is too small to show all of the translation. To enlarge the component, position the mouse cursor over one of the dark grab handles so that the cursor changes to a double-headed arrow. Hold down the left mouse button, drag the double-headed arrow in the direction you want to expand the component, then release the mouse button when you finish.

Saving your changes

Once you have finished translating text and modifying resources, save your changes. You can do one of three things:

- Update the original executable (.exe file) or dynalib (.apd file) with **File**, **Save**.
- Create a new executable or dynalib containing the updated resources with **File**, **Save As…**. To do this, you select either .exe or .apd from the Save as type combo box in the Save As dialog box.
- Save the changes in a resource file with **File**, **Save As…**. In this case you must choose the format of the resource file from the Save as type combo box: binary (.apr file) or human-readable (.txt file).

Note: Each copy of an executable, dynalib, or resource file should contain the translations for just one language .

Comparing the two methods

You can localize the actual text in an application faster by using the Object Nationalizer editor to do the translation. With this method you can also make global changes to text using **Edit**, **Replace…**.

However, you need to use the graphical method to verify that translated text is not cut off because a component is too small to show all of it. You also need to use the graphical method to enlarge a component when it does cut off translated text so that all the text can be displayed, or to change other resources (such as the component's font, color, and so on).

Translating messages and other constants

You must use the Object Nationalizer editor to translate the string constants (such as message text) in an application. You cannot translate these graphically because these strings are not components of a dialog box or window. Also, you can edit only constants that the application developer has placed in the User Constants section of the outline (not the System Constants section).

Applying resource file information

If you save resource changes in a resource file and you choose not to bind resources dynamically (read *Applying localizations at runtime* on page *1-6* for information on the dynamic approach), you must eventually apply those changes to an executable (.exe) or dynalib (.apd) file. You can do this in either of two ways:

- Open the resource file that has the modifications you want with **File**, **Open**, click **File**, **Apply**, then select the executable or dynalib you want to apply the resources to.
- Open the executable or dynalib you want to modify with **File**, **Open**, click **File**, **Apply**, then select the resource file that has the modifications you want to apply.

Object Nationalizer verifies that the resource file you select was derived from the executable or dynalib you are applying it to. If the verification succeeds, the resources in the resource file are applied to the executable or dynalib. If the verification fails, Object Nationalizer displays an error message.

Working in the right window pane

The right pane of the application window displays information related to the component selected in the explorer tree (in the left pane). This information is displayed on one of three tabs:

Components—This tab displays the components that are included in the component selected in the explorer tree. For each component you see its name, its type (for example, Background Text), the name of the class of which the component is an instance (if any), and the file with which the component is associated (if any).

When you expand the Child Windows component in the explorer tree, the Components tab in the right pane lists all of the components (child windows) in that component.

Description—This tab displays the textual information about either the entire application (if you select the top of the explorer tree) or one of its components (if you select a dialog box, MDI window, table window, or form window in the explorer tree).

To view the description of the entire application, select the **Application** node in the explorer tree, then select the **Description** tab. To view the description of a component in the application, select that component in the explorer tree, then select the **Description** tab.

You can also edit this text if you want to. You can use the description to provide information about what a component is, how it is used, how it has been changed over time, which components should not be translated, and anything else you feel is important to record.

Note: Users who run the application will never see this description text; you do not have to translate descriptions to localize the application.

Layout—This tab lets you graphically alter a dialog box, MDI window, table window, or form window. This tab is available only if you select a dialog box, MDI window, table window, or form window in the left pane.

Having multiple views into the application

You can have more than one view of a single application at the same time. Each view of the application is presented in its own window (and its own window title) inside the Object Nationalizer main window. You can have different views of the same component, and you can view and modify more than one component at the same time.

For example, you may want to view both the **Layout** tab and the **Components** tab for a given component at the same time. As another example, you may want to view the **Layout** tab for two different components. In either case, you need to have multiple windows open inside Object Nationalizer at the same time.

Opening a new window inside Object Nationalizer

1. Display the context menu for a component by right-clicking the component in the explorer tree.
2. Select **Open View** from the context menu.

Using more than one window

When you select **Open View**, Object Nationalizer creates a new window containing the tabbed information for the component. If there is only one tab of information for the component (such as Application Text), the same information about the component appears in both windows. If there is more than one tab of information, each window shows a different tab. For example, in the case of a dialog box, form window, table window, or MDI window, you could see the Layout tab in one window and the Components tab in another.

Once you have created a new window, you can locate another component in the explorer tree, display its context menu, then select **Open View** to create yet another window. In this way you can, for example, see the layout for two or more dialog boxes and windows at the same time.

Note: Only the window that appears when you first open a file (the application window) shows the explorer tree for the entire application. Each additional window that you open displays just that portion of the tree starting with the selected component. When the window first opens up, you cannot see this reduced tree—to view it, drag the left inner edge of the window to the right.

How to view multiple windows

Once you have two or more windows open at the same time, you can view these windows in one of three different ways:

- One window at a time.

  Maximize the window you want to view. The window fills the entire viewing area inside of Object Nationalizer. To view the next window, either minimize or close the current window, then maximize the next window.

- All the windows at once.

  Select **Window**, **Tile Horizontally** or **Window**, **Tile Vertically**. All of the windows (suitably reduced in size) are displayed either left-to-right or top-to- bottom in the Object Nationalizer window.

- All the window titles at once, but the information from just one window.

  Select **Window**, **Cascade**. All the windows are stacked like a hand of playing cards. The contents of the topmost window are completely visible; the remaining windows are mostly hidden behind the topmost window, with only their title bars visible. To bring another window to the front of the stack, click the title bar for that window.

You can close all the open windows at once by selecting **Window**, **Close All**.

Another way to select a window is to find the name of the window at the bottom of the **Window** menu. If the name of the window is listed, click the name to display the window.

Note: A checkmark always appears next to the window that is currently topmost.

Multiple views are for one application at a time

While you can have more than one view into a single application's resources at the same time, you cannot have more than one executable, dynalib, or resource file open at the same time.

To open another file when you already have a file open, save the changes you have made by selecting **Save** (or **Save As…**, along with a file name), select **Open…**, then specify the file to open.

Using the toolbars

Object Nationalizer has two toolbars to help you work faster: the Standard toolbar and the Arrange toolbar.

Standard toolbar—Use this toolbar to: open or save a file; cut, copy, and paste text; clear the clipboard; undo an operation; select large icons, small icons, or the list format; and show details.

Open Save Cut Copy Paste Clear Undo Large Small List Details

Arrange toolbar—Use this toolbar to arrange the components of a dialog box or window in a systematic way.

Grid Align Align Align Align Space Space Vert Horiz Same Same Same

To use the Arrange toolbar, select the components to be arranged, then click the appropriate toolbar button.

Use the Arrange toolbar buttons to:

- Turn on the grid.
- Align the left edges of the selected components.
- Align the right edges of the selected components.
- Align the top edges of the selected components.

- Align the bottom edges of the selected components.
- Space the components evenly on the horizontal axis.
- Space the components evenly on the vertical axis.
- Align the centers of the selected components horizontally.
- Align the centers of the selected components vertically.
- Make the selected components the same width.
- Make the selected components the same height.
- Make the selected components both the same width and the same height.

When Object Nationalizer first starts, the Standard toolbar is displayed and the Arrange toolbar is hidden.

To change whether a toolbar is visible or hidden, select **Tools**, **Toolbars** to display the Toolbars dialog box. Check the box next to the name of the toolbar to make it visible; clear the box to hide it.

Spacing and sizing components

Use the Layout menu to control the spacing and size of components on a dialog box or window. The menu items to use are:

Align Edges. Use this to line up the edges or centers of two or more components. You can align the top, left, right, or bottom edges, or you can align the vertical or horizontal centers.

Select the components to align, then select **Layout**, **Align Edges**. From the submenu select the kind of alignment you want.

Space Evenly. Use this to separate components evenly. You can space the components either across (left-to-right) or down (top-to-bottom).

Select the components to align, then select **Layout**, **Space Evenly**. From the submenu select the kind of spacing you want.

Arranging components with precision

You can arrange components on a window or dialog box in exactly the locations you want by using the grid. The grid consists of regularly spaced raised dimples on the background of a window or dialog box.

To use the grid, make sure it is enabled by seeing if the **Grid** menu item (on the **Layout** menu) is checked. If it is not checked, click the **Grid** menu item.

Once you have enabled the grid, you can align components to the nearest grid point. Select the component, then select **Layout**, **Align to Grid**.

Configuring the grid

To configure the grid, select **Tools**, **Preferences**. On the Grid tab, you can:

- Set the width and height of the grid.
- Show or hide the grid.
- Make the grid either active or inactive.

When you configure the grid, the settings stay in effect until you change them. You can open a different file or even exit from and restart Object Nationalizer without losing your current grid configuration settings.

Setting the width and height of the grid

When you set the width, you determine the horizontal spacing between grid lines; when you set the height, you determine the vertical spacing between the grid lines. The unit of horizontal spacing (width) is 1/2 the font size being used for text in the window or dialog box; the unit of vertical spacing (height) is 1/4 the font size.

For example, if the font size used in the window or dialog box is 12 points (1/6 inch), set the width to 12 and the height to 24 to create a square grid with 1 inch between all the grid points.

Showing and hiding the grid

To show the grid, check the **Visible** box; to hide the grid, clear the **Visible** box.

Making the grid active or inactive

To make the grid active, check the **Active** box; to make the grid inactive, clear the **Active** box.

Make the grid active if you want to ensure that all components are aligned with a grid point. When you move or resize a component on a window or dialog box with the grid active, Object Nationalizer automatically moves ("snaps") the component to the nearest grid point. (This has the same effect as selecting a component, then selecting **Layout**, **Align to Grid**.)

Note: The grid does not have to be visible to be active.

Verifying your settings

To see the effects of your settings while keeping the Preferences dialog box open, click **Apply**. Once you are satisfied with the settings, click **Close**.

Viewing obscured components

When you rearrange the components on a form or dialog box, you can partially or totally obscure another component. To see the obscured component, select the component that is in the way, then select **Layout**, **Send to Back**; this places the component behind any other component that is being obscured.

Use this method if the component you can see completely hides another component that you want to view.

If you can see only part of a component and you want to see all of it, you can select the component that is in the way and send it to the back, or you can select the partially visible component and select **Layout**, **Bring to Front**; this puts the selected component in front of the component that was obscuring it.

Scrolling in a dialog box or window

Some of the components in a dialog box or window may lie outside the boundaries of the window as displayed in the right pane on the **Layout** tab. To scroll beyond the boundaries of the window, you need to use two extra scroll bars called "design scroll bars."

To display design scroll bars, enable the **Show Design Scroll Bars** menu item in the **Layout** menu; to hide design scroll bars, disable **Show Design Scroll Bars**.

To see if **Show Design Scroll Bars** is enabled, select **Layout**, **Show Design Scroll**. If the menu item is checked, it is enabled; if it is not checked, it is disabled. To change it from disabled to enabled, or from enabled to disabled, click the menu item.

Showing sample text

To show sample text in the data fields, multiline fields, and table window columns defined in a window or dialog box, display the **Layout** menu and make sure the **Show Sample Text** menu item is enabled (checked). If it is not checked, click **Show Sample Text** to enable this feature.

What Object Nationalizer displays depends on the data type associated with the data field, multiline field, or table window column. For Number and Date/Time data types, Object Nationalizer displays data using the format specified in the Attribute Inspector. For String and Long String data types, Object Nationalizer formats the sample data using the input mask (if there is one).

For more information about the formatting of displayed data and the validation of input data with input masks, read the SQLWindows book *Developing with Team Developer, Chapter 9, Formatting and Validating*.

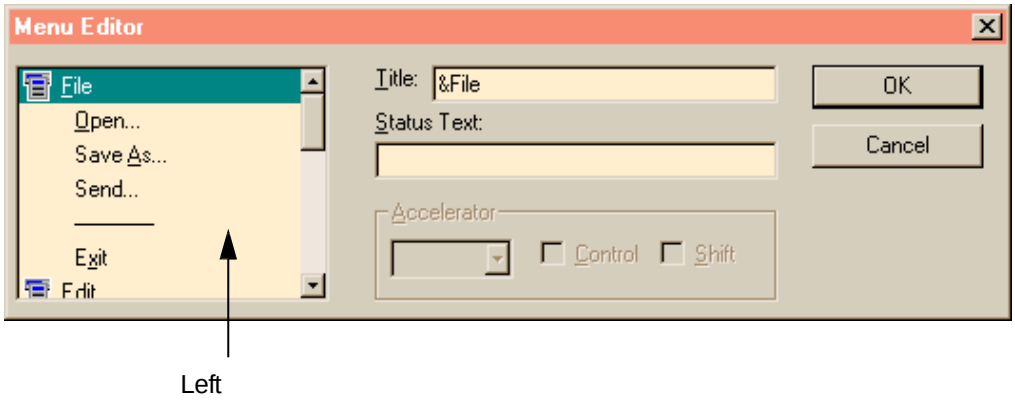
Localizing menus

You can localize the menus and status bar text in an application using either one of two editors:

- The Application Text editor. Menu items and status text appear in the Original Text column of the editor. To learn how to use this editor, read *Translating the application text* on page *2-3*.
- The Menu Editor. The rest of this section tells you how to use this editor.

To display the Menu Editor, use the right mouse button to select a component in the explorer tree that has menus, then select **Menu Editor…**. The three kinds of components that can have menus are form windows, top-level table windows, and MDI windows.

The Menu Editor looks like this:



Select the menu item to be translated in the left pane of the dialog box. (In the screen shot above, the **File** menu item is highlighted.) Next, click in the Title field and translate the existing text. Then click in the Status Text field and translate the text there. Repeat these steps for each menu item in the left pane, then click **OK**.

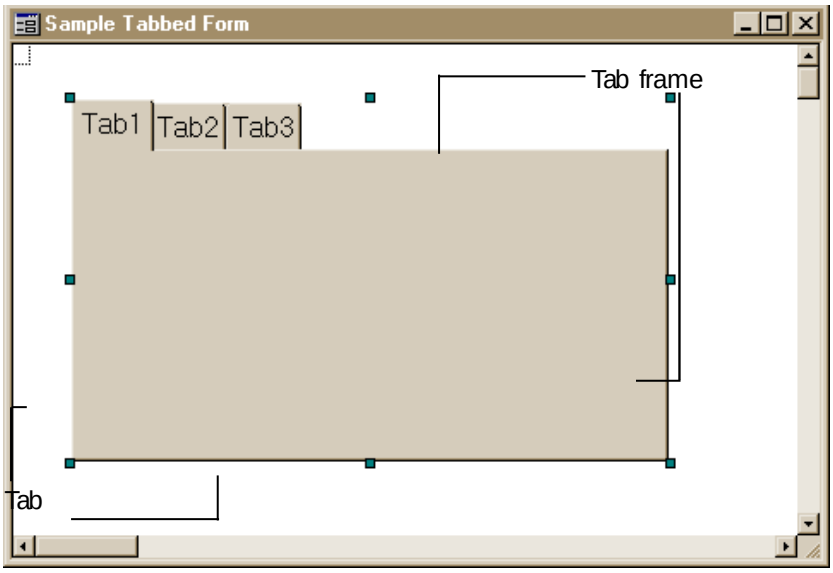
You can also rearrange the menu items in the left pane. To move a menu item, click on it, then drag it to its new location in the list of menu items and release.

The ampersand (&) in the Title text indicates that the following letter is the keyboard shortcut key for that menu item. When translating a menu item, you need to determine which letter in the translated text (if any) should be the shortcut key and precede that letter with an ampersand.

To actually see the menus that a window has, you can preview the window—read *Previewing a window or dialog box* on page *2-17* for more information.

Localizing QuickTabs

To localize or modify the tab labels created using QuickTabs on a form or dialog box, select the form or dialog box in the explorer tree, then click the Layout tab in the right pane to view the form or dialog box. Right-click on the tab frame (the outline of the tab—see the picture below) to display the context menu, click **Properties…**, then click **Tab Frame Properties…** to display the Tab Frame Properties dialog box.



In the Tab Frame Properties dialog box, select the Tabs tab, click any of the entries in the Tab Label column, then replace the existing text with the appropriate translation. Repeat for all the entries in the Tab Label column, then click **OK**.

Previewing a window or dialog box

After you have made changes to a window or dialog box, you can check your work and see how it will appear at runtime by selecting **Layout**, **Preview Window**. By previewing a window you can also see the menus (if any) associated with the window.

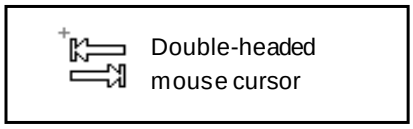
If you see that you need to make changes, close the preview of the window or dialog box by selecting **File**, **Close** from the Preview Window tools window, then return to the Layout tab in the right pane and modify the window or dialog box appropriately. When you have finished, select **Layout**, **Preview Window** again to see the results.

Changing the tab order

Every dialog box, form window, and MDI window defines the order in which components get the current focus when the user presses the **Tab** key. (The component listed first in the tab order has the focus when the dialog box or window is first displayed.) You can change this order with Object Nationalizer.

To view the tab order, select **Layout**, **Tab Order**. On the window or dialog box you see a number next to each of the components that are editable or can receive the focus. The component that first gets the focus when the window or dialog is displayed is labeled 1; the component that next gets the focus when the user presses the Tab key is labeled 2; and so on.

To change the tab order, set the tab sequence number that you want to assign to a component in the Tab Order dialog box. (You can enter a number directly, or you can increase and decrease the number displayed by clicking on the up and down arrows.) Move the double-headed mouse cursor (shown below) to the component to which you want to assign the tab sequence number you just set and click the left mouse button.



Changing a component's tab sequence number triggers a swap of tab sequence numbers between the component selected and one of the other components in the window or dialog box. The selected component is assigned the tab sequence number you set; the component that had that number is given the number originally assigned to the selected component.

For example, assume components A, B, and C have been assigned the tab sequence numbers 1, 2, and 3, respectively. Set the tab sequence number in the Tab Order dialog box to 2, then select component C. This action changes the tab sequence

numbering for A, B, C to 1, 3, 2. Component C gets tab sequence number 2 and gives up sequence number 3. B, which had sequence number 2, can't keep that number anymore now that C has it, so it gets sequence number 3 (the number that C is "releasing"). Only B and C are affected by this exchange of sequence numbers; the tab sequence number for component A remains 1.

Continue the process of reassigning tab sequence numbers until you are done, then click **Close**.

Showing hidden components

Some of the components in a dialog box or window may be marked as Hidden. A programmer hides components if they need to perform some behind-the-scenes action in the application, but the users of the application are not supposed to be able to see and manipulate those components.

To see hidden components when you view the components in any dialog box or window, enable the **Show Hidden Windows** menu item (in the **Layout** menu); to not see hidden components, disable **Show Hidden Windows**.

To see if **Show Hidden Windows** is enabled, select **Layout**, **Show Hidden Windows**. If the menu item is checked, it is enabled; if it is not checked, it is disabled. To change the setting, click the menu item.

Once you can see a hidden component, you can modify it just like any other component.

Important: When you enable Show Hidden Windows, you can see an application's hidden components in Object Nationalizer, but those components still keep the Hidden property enabled.

Localizing day and month names

Whenever a Team Developer application runs, it requires a number of files to support its execution. One of these files is the dynamic link library (DLL) named GTLSI62.DLL. This DLL contains string resources for the names of the days of the week (Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday) and the months of the year (January, February, … , December). It also contains string resources for the corresponding abbreviated names: Mon, … , Sun and Jan, … , Dec.

To have a Team Developer application display the days of the week and months of the year in a language other than English, you must translate the string resources for day names and month names in GTLSI62.DLL. You must then install this translated file along with the translated .exe file on the end users' systems before they run the translated application.

To extract and modify these resources, first make a copy of GTLSI62.DLL and store it in another directory. (You can find this DLL in the Deploy subdirectory of your Team Developer installation.)

Important: Do not modify the resources in the original GTLSI62.DLL. Also, do not change the name of the copied file—it must also be named GTLSI62.DLL.

Open the copy of GTLSI62.DLL using a suitable resource editor (such as the one that comes with Microsoft Visual C++) and translate the day and month names. You cannot use Object Nationalizer to edit this DLL; Object Nationalizer can only edit the resources in a .exe or .apd file generated by Team Developer.

Chapter 3. Examples

This chapter provides several example snippets of localizing Team Developer applications using Object Nationalizer. The examples use some of the sample applications that are available as part of the Team Developer installation.

Creating the sample executables

This chapter describes how to localize parts of several sample SQLWindows applications into standard French.

The sample applications used in this chapter are: qckfinal.app, company.app, and stddlg.app. All of these files are located in the Samples subdirectory of the Team Developer installation directory.

To do the examples, you must have the executable (.exe) versions of the sample applications. Moreover, these .exe files must have been built with resource editing enabled.

Creating the sample executables

To create the executable (.exe) sample files yourself, you need access to SQLWindows. Alternatively, you can have a SQLWindows programmer create them for you.

1. Open the source (.app) file for the sample application you want to make into an executable file.

   Start SQLWindows. Select **File**, **Open**. In the Open dialog box, make sure the Files of type combo box shows Normal (*.app). Navigate to the Samples subdirectory of the Team Developer installation directory, select the sample application you want, then click **Open**.

2. Set the target of the build and enable resource editing.

   Select **Project**, **Build Settings…** to display the Build Settings dialog box. Select the Executable radio button under Select build target, check the Enable Resource Editing checkbox, then click **OK**.

   If you are shown a message box saying that the .exe file to be generated already exists and asking you if you want to replace it, click **Yes**.

3. Save the file by selecting **File**, **Save**.

4. Build the .exe file.

   Click the **Project** menu, then click the menu item that says **Build***path*/*application name***.exe** (where *path* is the pathname of the directory where the .exe will be built, and *application name* is the name of the application file you opened).

5. Verify that one of the lines in the Build Configuration field of the Build Information window says "Resource Editing is Enabled". If it does, click **OK** in the Build Information dialog box; if it does not, click **Cancel** and go back to step 2.

6. Finish up and quit SQLWindows.

   Click **OK** in the message box saying that the build was successful, select **File**, **Exit**, then click **No** in the Save Changes dialog box.

Translating a dialog box

In this section you translate the dialog box from the sample application qckfinal. The SQLWindows source file is qckfinal.app; the executable file is qckfinal.exe.

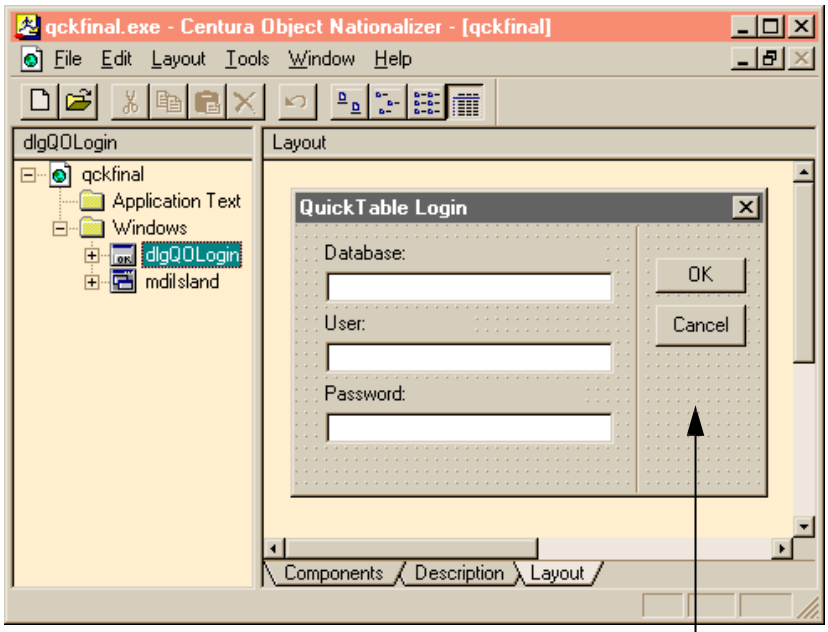
Translating a dialog box

1. Start Object Nationalizer, then open the file qckfinal.exe. In the left window pane, expand the tree node labeled qckfinal, then expand the node labeled Windows.

The tree now shows you that there are two top-level windows in this application: the dialog box dlgQOLogin and the MDI window mdiIsland.

2. Select the dlgQOLogin node, then click the Layout tab in the right pane.

You now see the QuickTable Login dialog box as it was created by the application programmer.



An area on the dialog box not occupied by ~~any of~~ the components of the dialog box.

3. Position the mouse cursor on the dialog box at a location not occupied by a component of the dialog box. (The figure above shows you one place where you can position the cursor.) Right-click at that location, then choose Properties, to display the Attribute Inspector for the dialog box.

4. Position the cursor over the Object Title property, enter the text "Ouverture de Session QuickTable", then close Attribute Inspector. The French translation for the title of the dialog box now appears in the title bar.

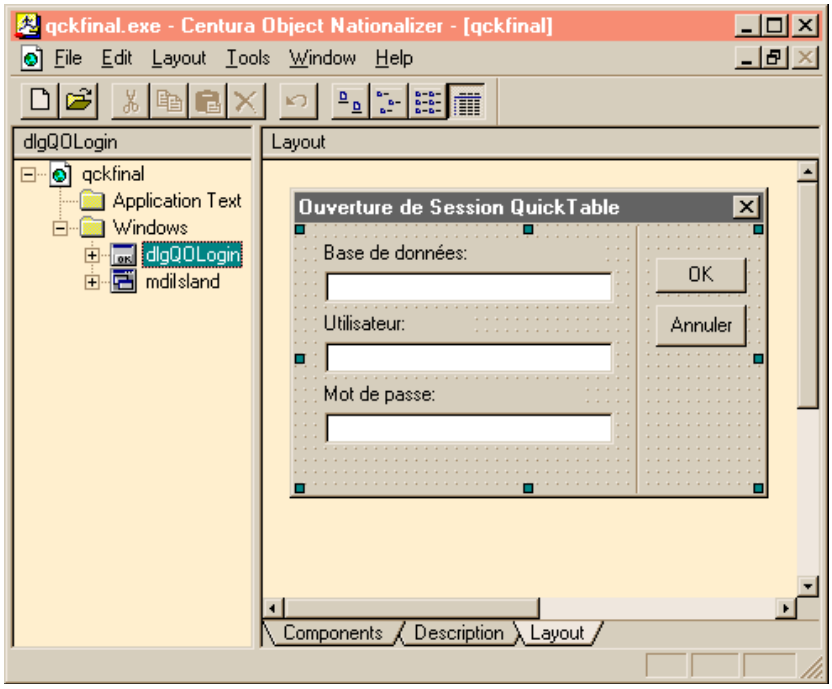
To translate the background text item Database, you proceed in the same way: position the mouse cursor over the text, right-click it , choose Properties to see the Attribute Inspector, move the cursor to the Object Title property, type in the text box "Base de données", then close Attribute Inspector.

Note: You don't have to close Attribute Inspector each time. If you have several changes to do, it is faster to simply open Attribute Inspector for the first change and, after doing that change, left-click on the next item to be changed. When you do, your first change is shown in the layout

window, and the properties for the second item are displayed in Attribute Inspector. You can continue this process of changes and left-clicks as long as you like.

Translate the two other background text items (User and Password) and the two push buttons (OK and Cancel) just as you translated the background text Database: use right click and the Attribute Inspector, and replace the existing Object Title with the translated text.

The dialog box now looks like this:



Translating tabs

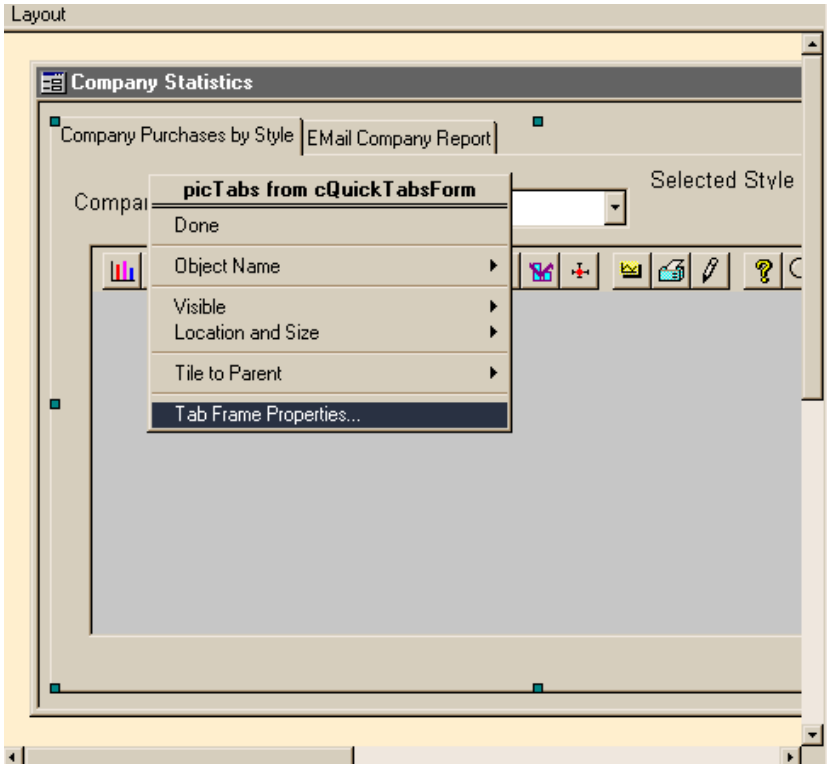
In this section you translate the tab labels that appear on the tabbed form window in the sample application company. The SQLWindows source file is company.app; the executable file is company.exe. You may need to build company.exe if it does not already exist.

Translating tabs

1. Open the sample file company.exe.

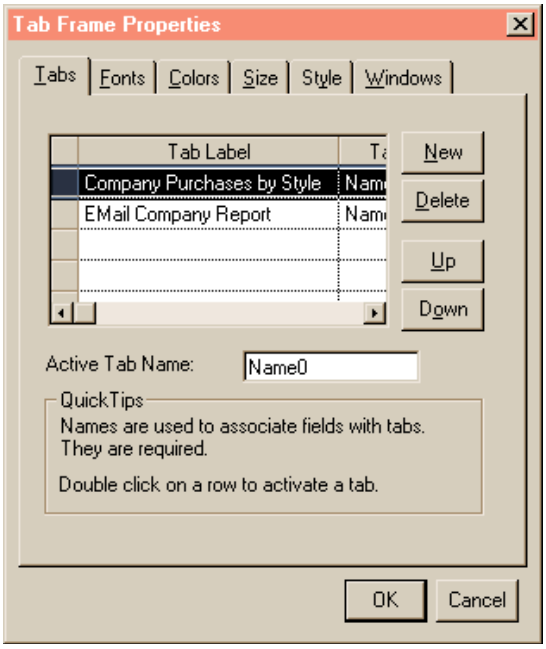
2. Expand the explorer tree on the left until you can select the form frmStat (one of the child windows of the mdiMain window), then display the Layout tab on the right.

3. Right-click the Company Purchases by Style tab to display the Edit menu, then select the **Tab Frame Properties…** menu item (as shown below).



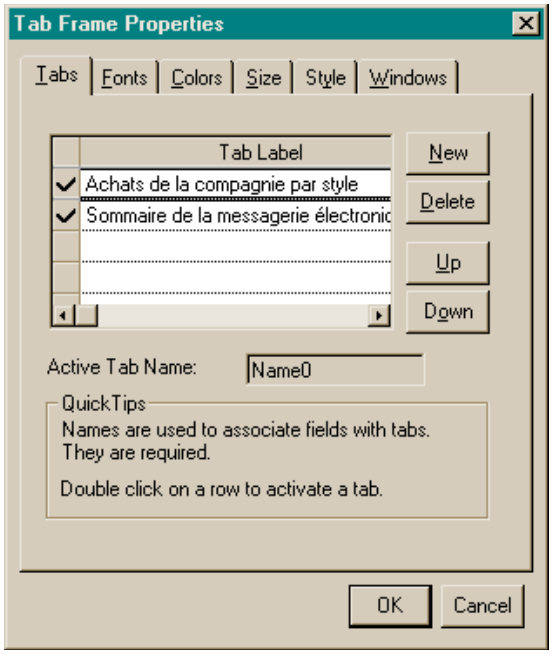
You now see the Tab Frame Properties dialog box.

4. Select the Tabs tab if it is not already on top. Drag the divider between the Tab Label column and the Tab Name column to the right to show the entire tab name "Company Purchases by Style."



5. Click in the first cell (with the text "Company Purchases by Style"), then change the text to "Achats de la compagnie par style." Click in the cell with the text

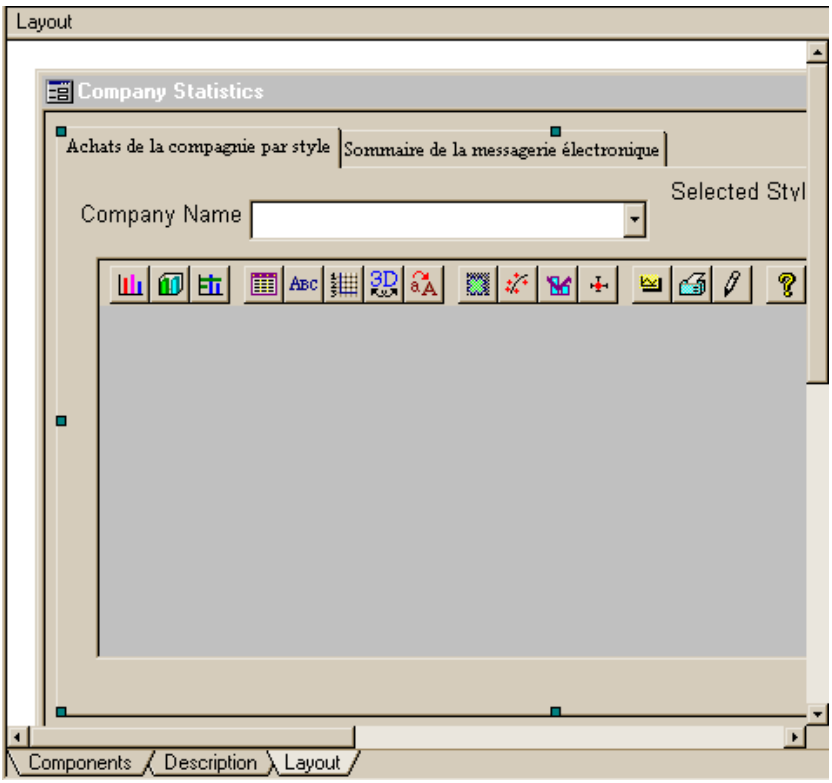
"EMail Company Report" and change the text to "Sommaire de la messagerie électronique."



6. Now that you have localized the tab labels, you can go further and customize the appearance of the labels as well. Click the Fonts tab, click Use Custom Font, then select Times New Roman from the Name drop-down list.

7. Click **OK**.

You have both translated the tab text and customized the appearance of the text (as shown below).



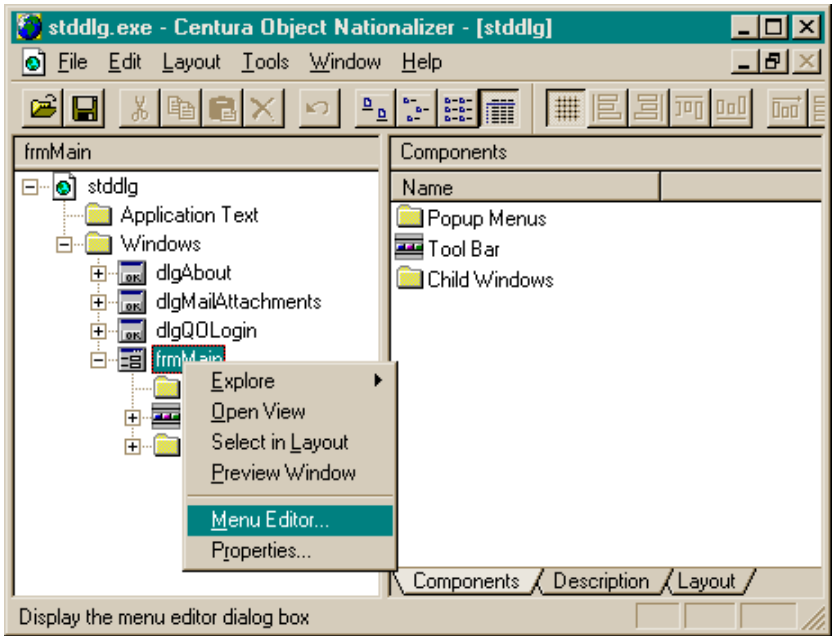
Translating menus

In this section you translate one of the menus in the sample application stddlg. The SQLWindows source file is stddlg.app; the executable file is stddlg.exe.

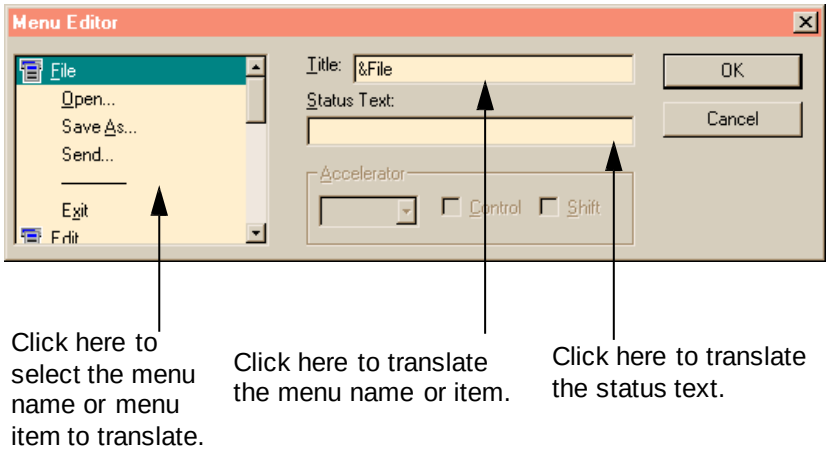
Translating menus

1. Open the sample file stddlg.exe.
2. Expand the explorer tree on the left until you can select the component frmMain.

3. Right-click frmMain, then select **Menu Editor…** from the context menu (as shown below):

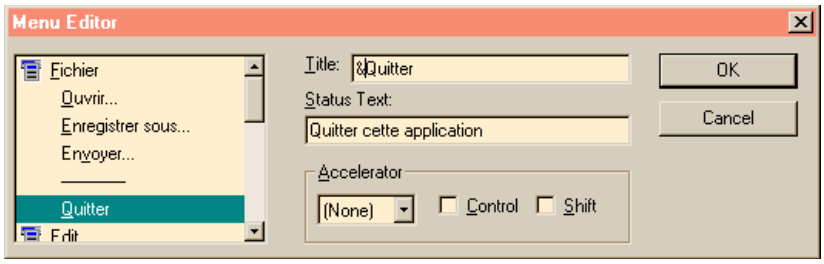


You now see the Menu Editor dialog box:



4. Click in the Title field and replace &File with &Fichier.

5. Select the menu item Open… in the scrollable pane on the left, then click in the Title text field to translate it to &Ouvrir….
6. Translate Save As… to Enregistrer sous…, Send… to Envoyer…. and Exit to Quitter. You end up with this:



To localize all the menus, you would continue scrolling down the left window pane, selecting and translating each one until you had done them all. At the end, you click **OK** to finish.

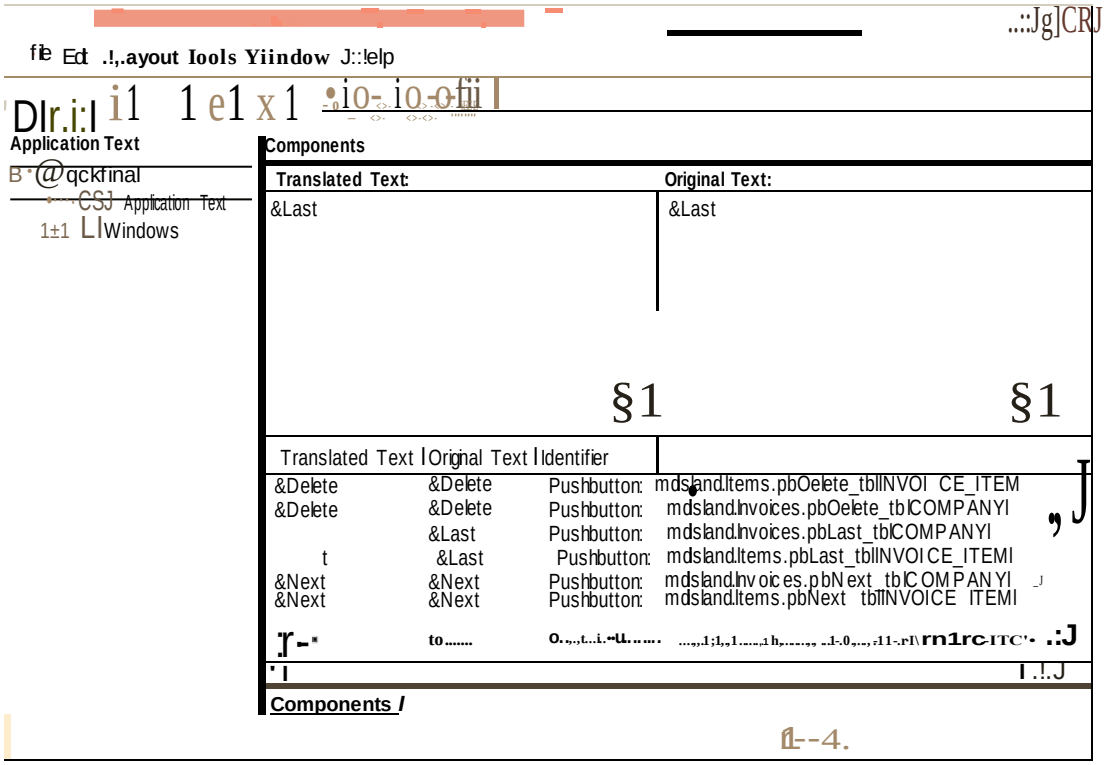
Translating application text

In this section you translate a piece of application text in the sample application qckfinal.app. With this approach you translate the strings without seeing the layout of the components that the strings are associated with.

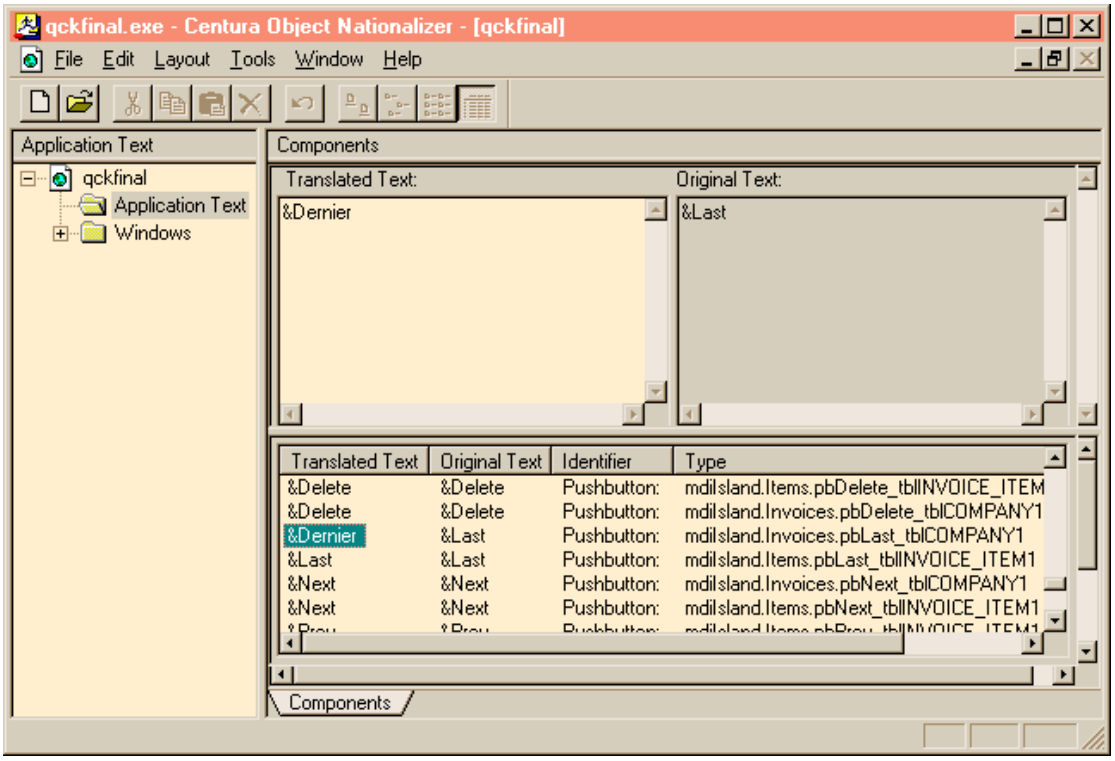
Translating application text

1. Open the sample file qckfinal.exe.
2. Expand the explorer tree on the left and select the item Application text. The right- hand pane displays the strings in the application.
3. Search for the string "Last" by selecting **Edit**, **Find…**, entering "Last" in the Find What text box, then clicking **Find Next**. Object Nationalizer displays the first occurrence of the text in the Translated Text column and highlights it.

4. Click on the text to make it editable (as shown below).



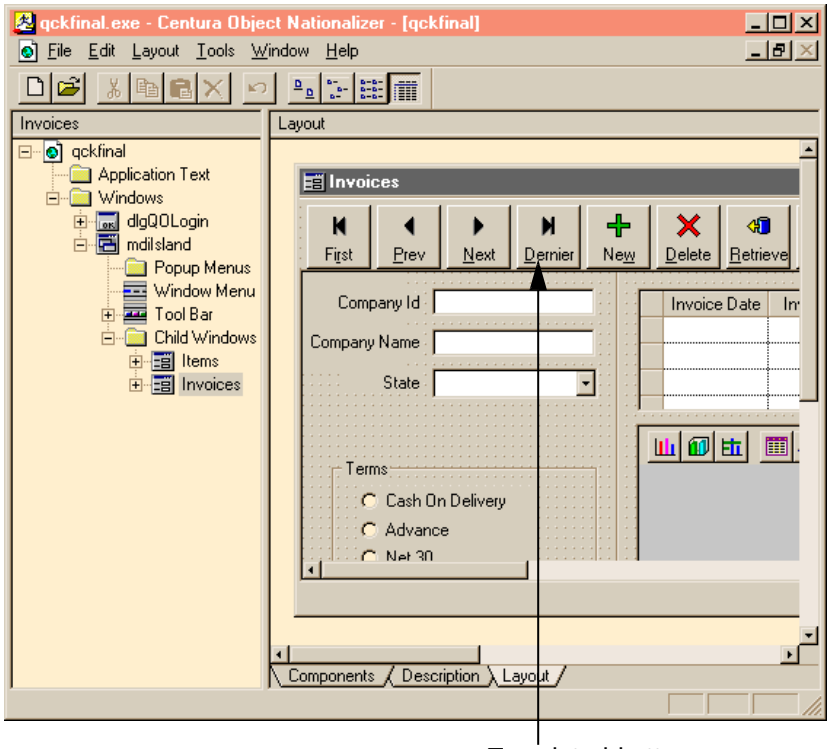
5. Type in the text "&Dernier" and press **Enter**.



The translation appears both in the Translated Text column where you typed it in and the Translated Text window pane (above the Translated Text column).

6. You can see how well the translation fits by viewing the form on which it appears. The first part of the identifier (mdiIsland.Invoices) tells you that the push button you just translated appears on the Invoices form in the mdiIsland window. Expand the Windows component in the explorer tree until you see the Invoices form, select it, then display the Layout tab in the right-hand window pane.

You can see in the next screen shot the translation on the push button. You can verify also that, in this particular case, the button does not need to be enlarged to display all of the translated text.



Translated button

Appendix A. Menu Reference

This appendix lists the menus and menu items available in Object Nationalizer.

File Menu

Menu Item	Description
Open	Opens an executable (.exe), a dynalib (.apd), a binary resource file (.apr), or a text resource file (.txt)
Save	Updates the contents of the file you opened with all the changes you have made so far.
Save As…	Saves the current values of the resources into a resource file, or creates a new executable or dynalib with the current values of the resources. You are prompted to choose the type of file to create and to name the file.
Apply…	Updates the resources in an executable (.exe) or dynalib (.apd) file. *See the note below.*
Exit	Closes down Object Nationalizer.

This menu also displays a list of the most recently opened applications between the **Print** and **Exit** menu items.

Note: When you select **Apply**, you are prompted for different information depending on the kind of file you currently have open. If you have a .exe or .apd file open, you are prompted to specify which resource file to use to update the executable (.exe) or dynalib (.apd) file; if you have a resource file open, you are prompted to specify which .exe or .apd file to update (using the information in the resource file you currently have open).

Edit Menu

Menu Item	Description
Undo	Undoes the last change you made to a resource. **Note:** Once you undo a step, you cannot redo it. Also, you can undo only one step, not multiple steps.
Cut	Moves the selected component or text to the Clipboard.
Copy	Copies the selected component or text to the Clipboard.
Paste	Copies the contents of the Clipboard to the active window at the cursor location.
Delete	Discards the selected component or text.
Find…	Finds the specified application text.
Find Again	Repeats the most recent search.
Replace…	Replaces the specified application text with the replacement text.
Properties…	Displays the Properties context menu for the currently selected component.

Layout Menu

Menu Item	Description
Preview Window	Displays the window or dialog box currently visible on the Layout tab as it would appear in the running application.
Bring to Front	Places the selected component in front of all other components that share at least some of the same area on a dialog box or window.
Send to Back	Places the selected component behind all other components that share at least some of the same area on a dialog box or window.
Align to Grid	Aligns the selected components with the points of the grid.
Align Edges	Aligns the edges or centers of two or more selected components. (The edge of the first component selected is used to align the edges of the other components.)
Space Evenly	Makes the spacing between selected components uniform.
Make Same Size	Makes the width, height, or both width and height of all the selected components identical. The width or height of the first component selected is used to set the width or height of the remaining components.)
Grid	Enables or disables aligning (snapping) components to the grid.
Tab Order…	Displays the Tab Order dialog box, which is used to view and modify the tab order of components in a window or dialog box.
Show Sample Text	Fills data fields, multiline fields, and table window columns with sample data according to the component's assigned format or input mask (depending on the data type of the component).
Show Design Scroll Bars	Shows or hides scroll bars that let you view parts of a window or dialog box that may not be visible when the application runs.
Show Hidden Windows	Respects or overrides the Visible property associated with components. If Show Hidden Windows is enabled, all components in the application are visible in the editor/ designer, even components whose Visible property has been set to "no".

Tools Menu

Menu Item	Description
Toolbars…	Displays a dialog box on which you select which toolbars (standard and alignment) are to be visible.
Attribute Inspector	Accelerator: Alt-3. Displays the Attribute Inspector dialog, which allows you to modify the attributes of the object that is currently selected.
Preferences…	Displays the Preferences dialog box. Use this dialog to customize the grid.

Windows Menu

Menu Item	Description
Cascade	Displays all open child MDI windows in a stepladder fashion, with only the topmost window completely visible.
Tile Horizontally	Displays all open child MDI windows from left to right.
Tile Vertically	Displays all open child MDI windows from top to bottom.
Close All	Closes all child MDI windows.

At the bottom of the Windows menu is a list of the windows that are currently open. A checkmark appears in front of the window that currently has the focus.

You can use this list to change the focus to another window. If the windows are being displayed in cascade mode, the window you select is brought to the top of the stack of windows.

Help Menu

Menu Item	Description
About Object Nationalizer …	Displays the version of Object Nationalizer.

Index

U

V