

# Building National Language Support Systems with Designer/2000

Stig Brandt

Product Manager/Oracle Denmark A/S

## This paper will cover issues as

This paper focuses on how to build Forms4.5 templates and extend the use of Designer/2000 to achieve almost 100% generated NLS\*Supporting systems. The paper will discuss:

- The approach of building NLS Systems with Designer/2000
- How to implement NLS Support on the Client Side versus Server Side
- How to build the Template to support NLS
- A demo of NLS \* Support System

Implementing NLS using Designer/2000 can be done in different ways, this paper will give ideas and hints how to implement all the objects that have to be translated and still obtain almost 100% generation using Oracle Forms Generator.

- What objects to be translated?
- How to setup Designer/2000
- The datamodel to support NLS
- The Template Form and Report
- Client/Server implementation

## What objects to be translated ?

### Textitems

For the textitems you have to consider what you want to translate, because it can give maintenance problems, and there are performance issues regarding the amount of data to move across the network. But in order to get a total picture of the NLS you have to translate the following for the textitem object:

- Prompts
- Hinttext
- Help on the item

Unfortunately it is not possible to set hinttext at runtime in Developer/2000, we therefore decided not to implement hinttext on the items. But a workaround could be to use when-new-item-instance trigger and write a message('Hint...');

### MenuItems

On the menu items you want to translate the:

- Label
- Accelerator keys

MenuItems are created as domains, so the menustructures is stored in the Cg\_ref\_code table, with meaning and value exactly matching the itemname in the menu.

Accelerator key are defined in Oracle\*Terminal, and the only way to translate the label to set labels in the terminal definition to null or a blank and then put the label at the end of the function label. Though it will give alignment problems, because of the proportional font.

ex. Create Record - Shift + F1

### Toolbar Icons

The toolbar contains bubble help. The bubble help is contained as a label on the button. To maintain this functionality we created a domain in Designer/2000 "Toolbar", with the toolbar items as valid values and the label as the meaning.

ex.      *BlockName*                      *Button*  
         *QMS\$TOOLBAR*                      *Save*

*Domain:*                      *QMS\$TOOLBAR*  
*Values:* *Save*  
                                 *and so on...*

To get the Domain values for the TOOLBAR domain, associate it to the cg\_ref\_code\_all table and then run the utility, you can then deassociate the domain again after the utility has been executed. Or as we did wrote a routine that extracted the values from the MetaModel View CI\_DOMAINS, CI\_ATTRIBUTE\_VALUES.

At forms - startup we set the toolbar buttons label in accordance to the chosen language.

### GUI-Items

GUI Items such as Listbox, Radiogroups, Checkbox's, those three groups are handled different because Radio Groups and ListBoxes contain values and Labels stored in the CG\_REF\_CODE table, and the Checkbox is only one label which is created as the prompt.

- Listbox

For the listbox we have to look at recordgroup to be able to get the right value for the language, therefore it will be implemented in the CG\_REF\_CODE table. The implementation of the CG\_REF\_CODE view

will be described in the section “Datamodel to support NLS”

- Radiogroups

For radiogroups we have to be able to change the label on each radiobutton and that we do with the Forms-builtin `SET_RADIO_BUTTON_PROPERTY` (item\_id, button\_name, LABEL, value) in the template. The label was set at forms startup, to every radiogroup and the actual label was retrieved from the `CG_REF_CODES` view through the Module/Domain table.

- Checkbox

For checkboxes we have to be able to change the label on the checkbox and therefore we have to store the label on each language in the item - tables. And set the label with `SET_ITEM_PROPERTY` (item\_id, LABEL, value) in the template. The label is retrieved from the Module/Domain table by a function in the template.

- Buttons

Buttons can be treated the same way as textitems, except that there is no help defined for the button, therefore for a button we translated: `SET_ITEM_PROPERTY` (item\_id, LABEL, value) in the template. The label is retrieved from Module/Domain table by a function in the template.

### Alert-boxes

On the alert boxes we want to translate the buttons:

- Alert\_Button1
- Alert\_Button2
- Alert\_Button3

This is implemented in the template form using the form built-in `SET_ALERT_BUTTON_PROPERTY`(alert\_name, button, property, value); Alerts were created as domains as well as the toolbar items, where each alert had these buttons as valid values.

### List of Values

On the list of Values we want to translate the title, and therefore use the built-in `Set_lov_property`('LOV\_NAME',Title,'Title'); This is done in forms startup.

To be able to get the right names of the LOV, we have to associate the LOV-Name to the Module, that is done in the Module/Domain Table. The List of Values Title was not solved at the customer's site, but the above approach will work.

### Message Handling

The feedback to the user is divided into two areas,

such as the FRM messages, and then there is the Designer/2000 generated messages on the Primary Key, Foreign Key, Check Constraints and in validation, those messages are recorded in each language outside Designer/2000 in a message table.

The Designer/2000 hard-coded generated messages are trapped in the on-error and on-message trigger and translated to the standard message handling system.

Default Oracle Generated messages were translated using the NLS-textfiles for the actual customer.

### Window-Titles

Each module consists of one or more window(s) and the challenge is to get the right title in the right language showing in the right window.

Each window title is retrieved from the Module/Domain table.

Window Titles were implemented as a domain, with number as value, the window was then associated to the module through the Module/Domain Table. And the relation is numbered to be able to recognize which window the title is associated.

Ex.      Domain:      *CG\$WINDOW*  
         Value:    *1*

*The module is associated to CG\$WINDOW, 1 as sequence 1, that means that the title is for CG\$WINDOW\_1.*

### How to Set up Designer/2000

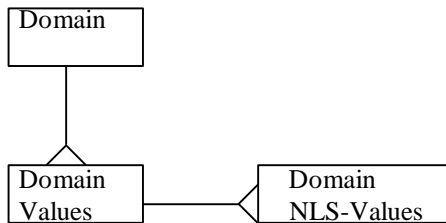
It is important to know how to use Designer/2000 to be able to get most out of the NLS system. The issue here is whether to use user extensibilities to be able to store your NLS-informations in the repository or to implement it as add-on tables to the repository. In the project we implemented it outside Designer/2000 in separate tables, but used the information from the repository to transfer them as initial values to the tables.

### Domains

We used the DOMAIN definition to record objects as:

- Toolbar Items
- Menu Items
- Window Titles
- List of Values
- Buttons
- Radio Groups
- Checkbox Items

## Structure of the Domain



example 1:

Domain: *Employee Job*  
 Value: *Clerk*  
 NLS-Values: *Danish Assistant*  
               *English Clerk*

Domain values are implemented in the table CG\_REF\_CODE and has the unique key RV\_DOMAIN, RV\_VALUE, what we did was to have one value and many “meanings”, namely a meaning for each language.

We could have implemented this by using the user-extensibilitet features and create a NLS\_DOMAIN, which should be a one-to-many for the domain, but it is not possible to create a “List of Values” for an User-Element.

For every attribute which have a limited amount of valid values, it is a pre-requisitet that you use the domain definition and set the flag on at the SoftLov, to get the domain implemented as a record group. For any value you choose to implement it with the “meaning”, Pop-list(meaning), Radiogroup(Meaning), Checkbox(Meaning).

## Description/Help text

Description at the analysis level is converted to helptext at design level, so now we have to decide how to record the descriptions, because we want helptext implemented in the different languages. And the helptext can either be implemented in MS\*Help or CG\_FORM\_HELP if we want to have the Forms\*Generator to generate the help code. CG\_FORM\_HELP gives us the same upportunity as with the Domain implementation.

MS\*Help is possible to generate with 1.3 release, and if you want to be able to swift between language helpfiles you have to change the standard procedure that follows Designer/2000 - CGHP\$CALL\_MS\_HELP, to change the call, depending on the chosen language.

The actual customer in this project used Oracle\*Book and created the Key-Help trigger as a Post-Generation operation.

## Utilities in Designer/2000

- Get CI\_Domain, CI\_attributeValues from Designer/2000 to CG\_REF\_CODES\_ORG.
- Create Initial Table/Columns as Domain=“ITEMS” and values equal distinct Column name

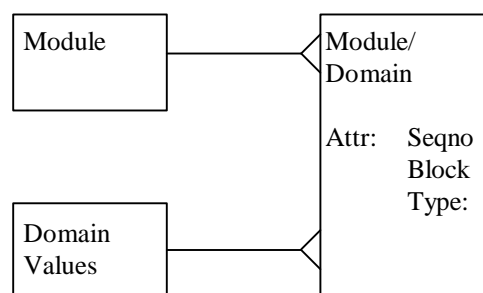
## The datamodel to support NLS

One prompt can be used in many modules  
 One Button Label can be used in many modules  
 MenuItems and Toolbar Items is used once  
 Window Titles can be used in many modules  
 Structure of CG\_REF\_CODE tables  
 To be able to get NLS on Listboxes, we reversed engineered the original CG\_REF\_CODE table, created two new tables called CG\_REF\_CODE\_ORG (Containing all the original values), we also created another one CG\_REF\_CODE\_ALL (Containing all the values translated to each language, extended with a LANGUAGE\_CODE column.  
 In addition we had to create a view to get the generated forms to work against the original CG\_REF\_CODE and the view definition is as follows:

```

CREATE OR REPLACE VIEW qms330_ref_codes
(
  rv_low_value
,rv_domain
,rv_high_value
,rv_abbreviation
,rv_meaning
,rv_type
)
AS SELECT   QRCA.RV_LOW_VALUE
,QRCA.RV_DOMAIN
,QRCA.RV_HIGH_VALUE
,QRCA.RV_ABBREVIATION
,QRCA.RV_MEANING
,QRCA.RV_TYPE
FROM
  qms330_ref_codes_all qrca,
  qms_user_prefs upr
where qrca.rv_lang_code = upr.lang_code
and   upr.user_name = user;
  
```

## Module/Domain table.



## The Template Form and Report

The template form was build using the Designer/2000 template package strategy, extended with functionality to maintain the display items. For each prompt we create a display item in the template form with the naming convention scr\_1, scr\_2...scr\_n placed on a control block, the post-generation job, was then to place the display items together with the actual item that is using the prompt, a exercise of about 5-10 minutes per form.

### Utilities in the template form

- Set Toolbar labels
- Set Radiogroup Button Labels
- Set Checkbox Labels
- Set Prompts in the Display Item
- Set Alert Box Button Labels
- Set LOV Titles
- Set menu Item Labels
- Set window Titles
- Set Button Labels

The display items was then set at startup time, with a routine that retrieved the data from the

The same approach was used in report, we had several display-items where the source was a parameter, the parameter was then filled at runtime.

## Client/Server implementation

To implement this functionality on the server side will of course give some network traffic, but surprising enough the startup of the forms were not magnificient different if the NLS hasn't been used. But the way this was implemented using the domain table structure, it is quite easy for each language to make a language file and put it on the client and instead of using the server to retrieve the data use the text\_io package forms built-in to read the file information.

## Author

Stig Brandt  
Product Manager (Tools)  
Oracle Denmark A/S  
Lautrupbjerg 2-6  
DK-2750 Ballerup  
Oracle mail: stbrandt.dk  
Email : stbrandt @dk.oracle.com