# **Multilingual Application Design**

- About this Guide
- Introduction
- Setting Up LANSA to Run Multilingual Applications
- Multilingual Variables
- The Data Dictionary and Multilingual Applications
- Creating Multilingual Application Programs
- Tips and Techniques
- Special Facilities

Edition Date July 17, 2014

© LANSA

#### **About this Guide**

- This guide provides a description of how LANSA multilingual support is implemented and how to obtain the best results when using it.
- You should read this guide after you have become familiar with the design and implementation of non-multilingual applications.
- By itself, this guide will not tell you all you need to know about designing and implementing systems. You will find related information in the following guides:
  - LANSA for iSeries User Guide
  - LANSA Technical Reference Guide
  - Deploying Visual LANSA Applications
  - LANSA Application Design Guide
  - LANSA Application Deployment Tool

### 1. Introduction

- 1.1 What is Multilingual Support?
- 1.2 When is it necessary to use Multilingual Support?
- 1.3 The Developer's Environment and the End User's Environment
- 1.4 The Difference Between National Language and Multilingual Support
- 1.5 Major Language Groups
- 1.6 Internationalization with Unicode
- 1.7 Restrictions and Limitations

### 1.1 What is Multilingual Support?

There are two components to Multilingual Support in any LANSA system:

- The Developer's Environment defines the set of languages available to be used in the Development Environment. Currently the Developer's Environment is support in English, French and Japanese.
- The End User's Environment defines the set of languages a delivered application is available to be executed in. In this environment Multilingual Support facilitates the execution of an application in one or more of the following:
  - Multiple languages
  - Bi-directional languages (IBM i 5250 applications only)
  - Languages which use a Double Byte Character Set (DBCS)

#### Also See

- 1.2 When is it necessary to use Multilingual Support?
- 1.3 The Developer's Environment and the End User's Environment

### 1.2 When is it necessary to use Multilingual Support?

Multilingual Support is set up at a partition level within a LANSA system.

It is strongly recommended that you use multilingual support for all partitions, even if you only intend to execute your application in a single language.

Aside from this recommendation, there are a variety of situations where it is necessary to set up Multilingual support including:

- You want to produce applications that use a double byte character set (DBCS) (also referred to as ideographic characters (IGC)) such as Chinese, Japanese or Korean.
- You want to produce applications that can run in more than one language.
- You want to produce applications that use a bi-directional language such as Hebrew or Arabic. Note that bi-directional languages are only available for 5250 applications using RDML code.

For applications in bi-directional or DBCS languages you must use multilingual support, regardless of whether or not the resulting applications are truly multilingual (that is, able to operate in more than one language).

**Note:** The ability to use national language capabilities in a normal LANSA system (without using multilingual support) only applies to languages that use the Latin alphabet (or a derivation from it).

All new partitions created on the IBM i on Japanese and French machines are automatically created as multilingual. All Visual LANSA systems are automatically created as multilingual.

# 1.3 The Developer's Environment and the End User's Environment

Before beginning to work with the multilingual facilities supported by LANSA, you must understand the difference between multilingual support for a developer and support for the end user.

The interface to LANSA used a developer can be installed in English, French and Japanese only. This is referred to as the Development Language. During the LANSA installation process you can nominate the set of Development Languages you want installed.

Applications created in LANSA using these languages can be designed and implemented to be fully multilingual and execute in a large number of languages. The LANSA development environment includes a variety of objects and features that enable you to create multilingual applications for your end users.

For example, you might be a Swiss computer programmer set the task of developing an order entry system.

When defining this application to LANSA and referring to technical LANSA documentation, you will be working in English.

However, a user of your completed order entry system can elect to execute it in, say, French, German or Italian, provided the appropriate translations have been incorporated into the database.

# 1.4 The Difference Between National Language and Multilingual Support

Very early versions of LANSA had basic national language support for all languages (other than bi-directional and IGC/DBCS languages).

This meant that it was relatively easy to set up a LANSA application to run exclusively in, for example, Swedish.

However, that's all the application could do, run in Swedish.

With full multilingual support it is possible to create an application so that at execution time you can elect to run the application in any of your supported application languages. For example you may have defined your application to run Swedish, French, Italian, German or English.

In fact, two users of the same program can sit side by side, with one running the application in French and the other in Italian.

In summary, a national language application can "speak" in one language, a multilingual application can "speak" in many different languages.

### 1.5 Major Language Groups

LANSA multilingual support differentiates between 3 major language groups because of hardware and operating system restrictions.

The 3 major language groups are:

LRTB Left to Right, Top to Bottom Languages.

Most languages derived from the Latin alphabet fall into this group. Examples include English, German, Swedish, French, etc.

DBCS Ideographic Languages.

Languages in this group use an ideographic form and use the Double Byte Character Set (DBCS) format. Examples include Chinese, Japanese and Korean.

RLTB Right to Left, Top to Bottom Languages.

Languages in this group are referred to as bi-directional languages as well. Examples include Hebrew and Arabic.

Bi-directional languages are supported in 5250 applications using server-side RDML but are not supported in Visual LANSA.

#### 1.6 Internationalization with Unicode

Unicode is a universal text encoding standard. Each character has a unique number, which allows every language to be handled with no concern of data loss when the data is moved between different environments.

LANSA provides the following features utilizing Unicode:

- All multilingual text in the Visual LANSA Editor is displayed using Unicode. You can see and edit English, French, Japanese, Chinese, and so on exactly as it will appear in the target application when running in a particular language.
- In an RDMLX partition, NChar and NVarChar field types allow data to be handled as Unicode throughout an RDMLX application, in the User Interface and in the database. This means an application can allow data from multiple languages to be displayed and edited, and saved in the same database file. When the data is retrieved from the database, no characters will be lost.
- Existing fields can be changed to NChar and NVarChar field types to enable your existing application to support world-wide multilingual customer data (such as names and addresses) as well as language-specific descriptions for code tables and many other situations.
- *Load Other File* creates NChar and NVarChar field types to support external database columns of Unicode native types.
- WAMs are always output in the Unicode encoding UTF-8.

#### 1.7 Restrictions and Limitations

The following restrictions and limitations apply to the use of the multilingual support facilities:

- The multilingual facilities can only be activated in partitions that use the SAA/CUA standards option.
- When defining a new partition (or modifying an existing partition) to support multilingual applications a "default" language must be nominated. The partition will not be considered multilingual until this language has been defined.
- NChar and NVarChar fields are only supported on RDMLX objects in an RDMLX partition. They may not be displayed on \*WEBEVENT functions.
- It is virtually impossible to fully develop a DBCS application on a machine that does not have a DBCS primary language and DBCS capable devices attached.
- For applications in bi-directional or DBCS languages you must use multilingual support, regardless of whether or not the resulting applications are truly multilingual (that is, able to operate in more than one language).
- The following table indicates dynamic language switching capabilities between the 3 main language groups:

<b>Compiled Language</b>	LRTB	DBCS	RLTB
LRTB	Yes	Yes *2*4	Yes *1
DBCS *2	Yes *3	Yes	Yes *1*3
RLTB	Yes *1	Yes *1*2*4	Yes

<sup>\*1</sup> indicates that some screen panels may appear strange in a dynamic switch between the languages. Recompilation would be recommended.

<sup>\*2</sup> indicates that the machine must have OS/400 or CPF DBCS support for the language being used as a primary or secondary language.

<sup>\*3</sup> indicates that the display device should be DBCS capable or the operating system will issue an inquiry message indicating the device is non-DBCS. Recompilation is recommended if non-DBCS capable display devices are to

be used.

\*4 indicates that recompilation is recommended if DBCS capable display devices are to be used.

Refer also to Handling Multilingual Text.

### 2. Setting Up LANSA to Run Multilingual Applications

- 2.1 Creating or Modifying a Partition
- 2.2 Specifying a Partition's Multilingual Attributes
- 2.3 System Message File Handling
- 2.4 Considerations for Exporting/Importing Multilingual Applications

### 2.1 Creating or Modifying a Partition

Partition Maintenance includes the options "Multilingual support" and "SAA/CUA standards". Both of these options must be set to indicate that the partition is required to support multilingual development.

If you wish to make use of the Unicode field types NChar and NVarChar, you also need to be aware of the options *Enable partition for full RDMLX* and *Available field types*. The partition must be enabled for RDMLX and the types you wish to use must be allowed.

On an IBM i master system, when you create a partition and choose the option to copy system fields from another partition, only the non-multilingual field information will be copied as no languages have yet been defined for this partition. You should later use the export and import facility of LANSA to set up any multilingual information.

After creating a partition or modifying a partition to be multilingual you are required to define the languages you want to work with in this partition.

**Note:** Until a default language has been defined for a multilingual partition, the partition will not be considered to be multilingual.

#### Also See

2.2 Specifying a Partition's Multilingual Attributes

Enable Existing Partitions for Full RDMLX in the Visual LANSA Administrators Guide.

For an IBM i Master: Available field types refer to If you Enable a Partition for Full RDMLX in the *LANSA* for iSeries User Guide.

For no IBM i Master: Available field types refer to Field Type Enabled in Partition in the *Visual LANSA Administrators Guide*.

### 2.2 Specifying a Partition's Multilingual Attributes

It is recommended that all partitions are defined as multilingual.

To provide multilingual support within a partition you must define the languages you want to work with in the partition.

**In** Visual LANSA, a new partition is automatically defined as multilingual and a default language is specified for the partition.

**In an IBM i** the partition must changed to be multilingual and an appropriate default language (and any other languages) must be defined before the partition can be used.

#### LANSA on an IBM i

For an IBM i Master system you change the new partition after it has been created using the **Housekeeping** option *Change partition being worked with*. After selecting the partition you wish to work with, use the **Housekeeping** option *Work with partition multilingual attributes* to set up or review the multilingual attributes of the partition.

DC@P400501 Multilingual Partition Languages

**Multilingual Partition: XXX** 

Select language to review/change or use Add key to add a new language

```
Sel Language Description Default

_ ENG English YES

_ DEU German

_ TCHI Traditional Chinese

_ HEB Hebrew
```

+

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

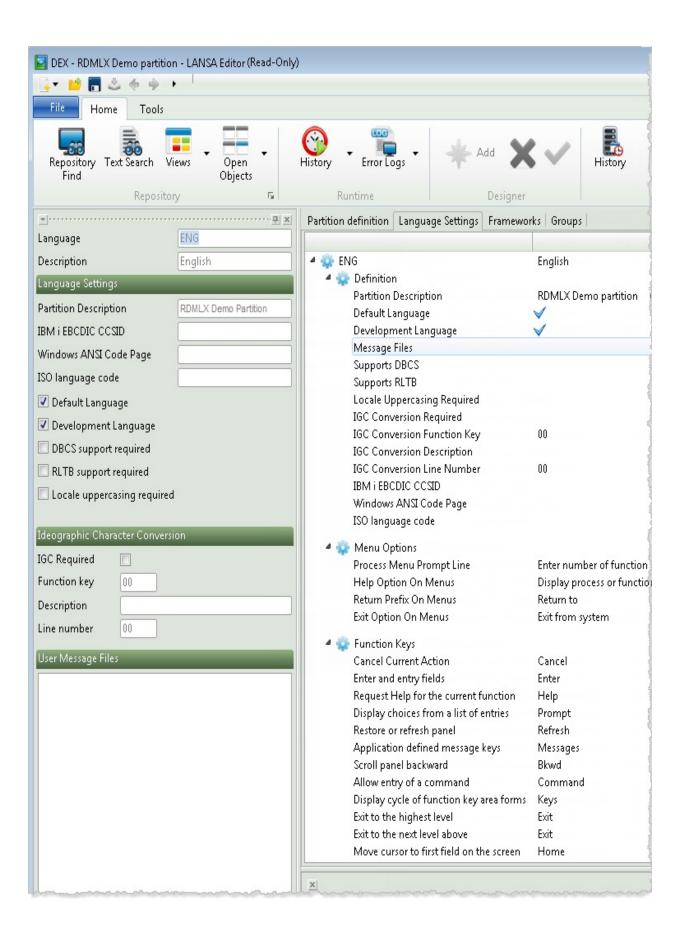
DC@P400502 Display Multilingual Partition Language **Multilingual Partition . . . . : XXX** Language . . . . . . . . . ENG Language description . . . . . . English DBCS support required ..... NO YES, NO IGCCNV required ..... NO YES, NO Function key / description . . . . / Line number . . . . . . . . . . . . RLTB support required ..... NO YES, NO Default language . . . . . . YES YES, NO Development language . . . . . . NO YES, NO PC Locale uppercasing required . . . NO YES, NO Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Change Fnn=Delete

Select the language and options you need for your partition. Advice for each option is in this guide.

#### **Visual LANSA**

In a Visual LANSA Slave or Independent System, after creating a partition or changing a partition to be multilingual you define the supported languages using

the *Language Settings* tab on the Partition definition.



Advice for each option is in this guide.

You can define any number of languages (from any of the language groups) that are to be supported within the partition. The definition of a language includes these options:

Partition identifier	Language settings	Ideographic character conversion
2.2.1 Language Code	2.2.3 Partition Description	2.2.12 IGC required
2.2.2 Language	2.2.4 IBM i EBCDIC	2.2.13 IGC Function Key
Description	CCSID	2.2.14 IGC Description
	2.2.5 Windows ANSI code page	2.2.15 IGC Line Number
	2.2.6 ISO language code	
	2.2.7 Default Language	
	2.2.8 Development	
	Language	
	2.2.9 DBCS support required	
	2.2.10 RLTB support required	
	2.2.11 Locale uppercasing required	

- 2.3.3 User Message Files
- 2.2.17 Menu Options
- 2.2.18 Function Keys

### 2.2.1 Language Code

One of the language codes for this partition. Code used must be unique in this partition.

A maximum of 4 characters.

The following codes for Languages are recommended, but not enforced.

Code	Language
CES	Czech
DEU	German
DUT	Dutch
ENG	English
FIN	Finnish
FRA	French
GRE	Greek
HEB	Hebrew (RLTB)
ITL	Italian
JPN	Japanese (DBCS)
NOR	Norwegian
POR	Portuguese
SCHI	Simplified Chinese (DBCS)
SWE	Swedish
SRFA	Swiss French
TCHI	Traditional Chinese (DBCS)
TUR	Turkish

# 2.2.2 Language Description

Provide the country name for this language.

### 2.2.3 Partition Description

A Partition description for the language. Enter a description that will assist developers to understand the purpose of the partition.

#### 2.2.4 IBM i EBCDIC CCSID

When text is moved from the client to the server and vice versa it must be translated from one system's code page to the other. Specify the CCSID to be used to translate text to the correct IBM i format for the selected language. If the code page is not entered then the default CCSID for the language will be used by the LANSA system.

### 2.2.5 Windows ANSI code page

When text is moved from the client to the server and vice versa it must be translated from one system's code page to the other. Specify the code page to be used to translate text to the correct Windows format for the selected language. If the code page is not entered then the default code page for the language will be used by the LANSA system.

### 2.2.6 ISO language code

The ISO 639-1 two-letter language code. It can be optionally qualified by an ISO 3166-1 two-letter country code. The language code or qualified language-country code must be a valid registered code, for example: en (English), en-US (English-United States) and en-GB (English-United Kingdom). This field is case sensitive. Values must be entered in the correct case (for example: en-US).

The ISO language code is commonly used in Web applications and when interfacing with external libraries that support localization.

### 2.2.7 Default Language

Select this option for the language that is to be the default. One of the languages defined must be nominated as the default language. The default language is an important language and has the following implications:

- When an application is invoked without nominating the required language, the default language will be used.
- Initial data dictionary, database and application development must be done in the default language. For instance, if you specify a field description via the data modeling extended file definition facilities, it is assumed that the description is in the default language.

Important: Until a default language has been defined for a multilingual partition, the partition will not be considered to be multilingual.

### 2.2.8 Development Language

This option indicates the language(s) available to be used in the development environment.

## 2.2.9 DBCS support required

Indicates that this language requires Double Byte Character Support.

## 2.2.10 RLTB support required

Indicates that this language requires Right to Left, Top to Bottom support.

### 2.2.11 Locale uppercasing required

In a client/server configuration, this option is obtained from an IBM i server and is set during system initialisation on the PC.

The option is only used with fields that do not have the attribute LC (Lowercase entry allowed) specified. Unless LC is specified, a field's input values in the range a to z will be converted to upper case (that is A to Z).

On the IBM i this option is set to:

**YES** if characters outside the a to z range must also be converted to upper case, for example, from  $\grave{a}$  to  $\grave{A}$ 

**NO**, if special characters are to remain as lower case while those in the a to z range are converted to upper case. This could be the result: RESUMé.

For information regarding field attributes, refer to Input Attributes in the Technical Reference Guide.

### 2.2.12 IGC required

This option is only relevant if you are using an IBM i application server. Indicates if IdeoGraphic Character conversion is required for applications generated under this language in this partition via the DDS keyword IGCCNV.

### 2.2.13 IGC Function Key

This option is only relevant if IGC conversion is required and is only implemented on IBM i.

Indicate the Function Key to be associated with Ideographic character support. Select a function key number in the range of 1 to 24 that does not conflict with any other CUA assigned function key in this partition.

### 2.2.14 IGC Description

This option is only relevant if IGC conversion is required and is only implemented on IBM i.

Indicate the Description to be used for the function key associated with Ideographic character support.

### 2.2.15 IGC Line Number

This option is only relevant if IGC conversion is required.

Indicate the line number where the function key associated with Ideographic character support should be displayed.

• Exactly the same technique is used by LANSA for its own internal messages. This is described in 2.3 System Message File Handling.

#### 2.2.16 User Message Files

Enter the Message File name for messages you wish to be displayed when an internal or end user makes an error that is specific to your system or an application being developed. These message files are used to execute OVRMSGF commands when entering applications defined within this partition.

The message file name must be a short file name, rather than a long one and will contain, as a suffix, the language for the file.

You may specify up to 10 User Message Files for each language.

The same number of User Message Files specified for the default language must be specified for each language in your partition. That is, message file "pairing" is mandatory.

For example, if a partition is set up to run three languages, French, Italian and Traditional Chinese, with French as the nominated default language (and thus the development language) you would create the following message file "pairings":

Language	User Message File 1	<b>User Message File 2</b>
----------	---------------------	----------------------------

French USR01FRA USR02FRA

Italian USR01ITL USR02ITL

Traditional Chinese USR01TCHI USR02TCHI

USR01FRA contains a set of user messages in French.

USR01ITL contains exactly the same set of messages in Italian.

USR01TCHI contains exactly the same set of messages in Traditional Chinese (DBCS).

The same pattern applies to USR02xxxx.

For futher information about System and User message files refer to 2.3 System Message File Handling.

### 2.2.17 Menu Options

Provide translations in the current language for the menu options which will appear on process menus.

### 2.2.18 Function Keys

Provide translations in the current language for the function key short form descriptions which will appear on process and function screen interfaces.

### 2.3 System Message File Handling

LANSA is supplied with a message file called DC@M01 which resides in the LANSA program library.

In most cases, but not all, the messages in DC@M01 are for application developers only.

The equivalent files in French (message file DC@M01FRA) and Japanese (message file DC@M01JPN) are also supplied.

The DC@01xxx message files may be replaced when Program Changes are applied or a new version of LANSA is installed. It is therefore recommended that you do not change the messages in these base files but you can use them as a basis for your own message files as described in 2.3.3 User Message Files.

#### Also see

- 2.3.1 LANSA-Issued Messages
- 2.3.2 MLTnnnn messages
- 2.3.4 User Message File Process Example

Add Message Files to PC Export Lists in the LANSA for iSeries User Guide.

### 2.3.1 LANSA-Issued Messages

Most of the messages in the message files are only issued in situations that make them visible to application developers. However, some of these messages are issued in situations that make them visible to end users of applications created by LANSA.

For example, if an end user displays a process menu and attempts to select an option with an incorrectly positioned screen cursor, message DCM0546 is issued. In English it looks like this:

"Screen cursor not positioned on same line as a valid function"

In non-multilingual partitions, all messages are issued from message file DC@M01 and appear only in English (unless changed to another language by directly changing the DC@M01 message file).

In a multilingual partition, a message file override command is issued from DC@M01 to DC@M01xxxx, where "xxxx" is the 4 character identifier of the language being used.

Thus when message DCM0546 is issued, message file DC@M01xxxx is searched first.

If the message is found in DC@M01xxxx, it is issued from there. If it is not found, it is issued from message file DC@M01 (in English, if the original file hasn't been altered).

If the LANSA environment has been left, message file language selection does not apply and the messages will always come from file DC@M01.

To provide messages in all your supported languages, check that the message file DC@M01xxxx exists (where xxxx is the 4 character language identifier) for each language you will be using and ensure that it has the correct owner (the same as the LANSA system owner).

## In summary

- Fully translated versions of the DC@M01xxxx message files are shipped with LANSA for English, French and Japanese.
- The list of messages that LANSA issues that are "end user visible" is in file DC@M01ENG (English). This message file may be found in any LANSA system.
- In an IBM i environment, to copy messages from one message file to another (for example, from DC@M01ENG to DC@M01TCHI):

- a. Use the merge message file commands. Initially, both versions of the message will be in English.
- b. Tranlate the copied messages into the desired language. In an IBM i environment use the operating system change message commands.
- DO NOT put user defined messages in a DC@M01xxxx message file. Instead go to 2.3.3 User Message Files.

## 2.3.2 MLTnnnn messages

MLTnnnn messages are special messages that contain text used on LANSA screen panels.

The following table contains the rules for translating some of these MLT messages within the multilingual message files DC@M01xxxx. These messages should always be translated for the required language.

The Max Length is shown in bytes including SO/SI characters.

Mltnnnn Message No.	Where Used	Max Len	Comments
MLT0001	Help support (Index Panel Title)	40	Center
MLT0002	Help support (instruction - cursor6 blank characters )	74	Left align
MLT0003	Help support (instruction - number)	74	Left align
MLT0004	Help support (word for process)	10	Left align
MLT0005	Help support (word for function)	10	Left align
MLT0006	Help support (word for field)	10	Left align
MLT0007	Help support (scrolling - up)	20	Left align
MLT0008	Help support (auto.generated text)	26	Left align
MLT0009	Help support (auto.generated text)	26	Left align
MLT0010	Help support (auto.generated text)	26	Left align
MLT0011	Help support (page numbering)	6	Left align
MLT0012	Action bar processing (unavailable choice)	74	Left align
MLT0014	Help support (scrolling - up/down)	20	Left align
MLT0015	Help support (scrolling - down)	20	Left align
MLT0016	MESSAGE TYPE(*WINDOW) reply string	10	Left align

MLT0017	Editing on a DBCS capable machine	60	Left align
MLT0018	Pop-up windows (Messages top)	23	Left align
MLT0019	Pop-up windows (Messages bottom)	26	Left align
MLT0020	Graphical User Interface (Help text)	77	Left align
MLT0021	Graphical User Interface (Help text)	77	Left align after 6 blank chars.

If the use or meaning of any of these text strings is unclear, refer to the original version of the message in DC@M01 or DC@M01ENG.

#### 2.3.3 User Message Files

You may specify up to ten User Message Files for each language. These message files are used to execute OVRMSGF commands when entering applications defined within this partition.

#### Points to note:

- The ten pairs of message files allow messages to be split up into different message files by application.
- The same number of User Message Files specified for the default language must be specified for each language in your partition. That is, message file "pairing" is mandatory.

For example, if a partition is set up to run two languages, French and Italian, with French as the nominated default language (and thus the development language). You would use the following message file "pairings":

#### Language User Message File 1 User Message File 2

French USR01FRA USR02FRA

Italian USR01ITL USR02ITL

USR01FRA contains a set of user messages in French.

USR01ITL contains exactly the same set of messages in Italian.

To create User Message files in a client server environment with IBM i as the server:

- a. Create the files on the IBM i as described in 2.2 Specifying a Partition's Multilingual Attributes
- b. Export them to the Visual LANSA system as described in Add Message Files to PC Export Lists in the LANSA for iSeries User Guide.

If you are exporting from an IBM i to Visual LANSA all languages are selected by default. If you are exporting from an IBM i to an IBM i, you can select the language to be exported.

To create User Message files in a Visual LANSA independent Visual LANSA system:

a. Create User Message files as described in 2.2 Specifying a Partition's Multilingual Attributes (you will need to scroll down the page).

#### **Maintaining Message Files**

For LANSA on both the IBM i and PC using Visual LANSA, the content for your message file is usually specified when you are setting some rules, such as for a range of values (refer to the *LANSA for iSeries Guide* or the *Visual LANSA User Guide* for more information), however you can add or amend rules using the message maintenance facilities.

#### **Using Visual LANSA**

- 1. Define the message file name as described in 2.2 Specifying a Partition's Multilingual Attributes.
- 2. Select *Message Files* from the *Tools* menu.
- 3. Adjust the text or add new messages, as described in Message File Maintenance in the Visual LANSA Administrator's Guide.

#### Using IBM i

- 1. Define the message file name as described in 2.2 Specifying a Partition's Multilingual Attributes.
- 2. Add, amend or delete messages using the ADDMSGC, CHGMSCD and DLMSGD commands that are described in *LANSA for iSeries User Guide*.

#### Also see

- 2.3.4 User Message File Process Example
- 2.4 Considerations for Exporting/Importing Multilingual Applications

## 2.3.4 User Message File Process Example

A developer may define a validation check in the LANSA data dictionary and indicate that if an error occurs, the error message number is USR4567 from message file USR01FRA is to be displayed.

The French message file is used because French is the default language for the partition, and therefore the development language.

What will happen when an application running in French encounters the error? Message USR4567 from message file USR01FRA is issued and will appear on the bottom of the screen in French.

If, for example, the same LANSA system is running on an Italian-language PC, when message USR4567 is issued for message file USR01FRA, the operating system looks in USR01ITL first. If message UST4567 is found in USR01ITL, it will be displayed from that file in Italian. If it can't be found then it will be displayed in French from USR01FRA. (The equivalent would occur with USR02ITL and USR02FRA.)

# 2.4 Considerations for Exporting/Importing Multilingual Applications

When setting up a partition for multilingual support, you should consider the following when exporting and importing between partitions:

- Material from a multilingual partition cannot be imported into a non-multilingual partition.
- When exporting from a multilingual partition you can indicate whether the export run should be in multilingual format.
- If you are exporting from an IBM i to an IBM i you can choose a language.
- If you are exporting from an IBM i to Visual LANSA all languages are selected by default.
- Material from a non-multilingual partition can be imported into a multilingual partition. You must nominate which of the languages defined in the multilingual partition is to be the default language for the imported data so that it can be correctly stored in the data dictionary. If not specified, the partition default language will be used as the default.
- Enabling ENG (English) in any multilingual partition is a good idea, because some LANSA development facilities are shipped in English only.

#### Also see

Add Message Files to PC Export Lists in the LANSA for iSeries User Guide.

### 3. Multilingual Variables

A multilingual variable is essentially a character string whose contents vary with the language being used. Multilingual variables can only be defined and used in multilingual enabled partitions.

Multilingual variables are defined at the system (rather than partition) level and are identified by the prefix \*MTXT.

Generally you can use a multilingual variable name throughout the LANSA system. Some examples of this include:

- Within validation checks.
- As default values for fields in the repository.
- Just about anywhere in RDML or RDMLX code.

#### Also see

- 3.1 Defining Multilingual Variables
- 3.2 Where Can Multilingual Variables Be Used?
- 3.3 Standard System Variables useful for Multilingual Development

## 3.1 Defining Multilingual Variables

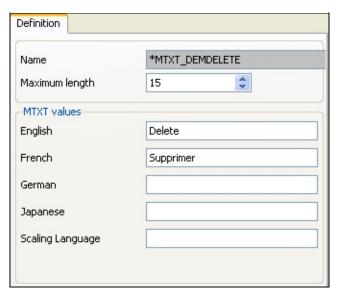
When you create a Multilingual Variable the default prefix of \*MTXT is provided but you need to supply a suffix which complies with your site naming standards.

Multilingual Variables are defined with a maximum length of 78 characters. Ensure the length specified is as long as the longest translation for the multilingual variable in any language defined for the partition.

The description entered when creating a multilingual variable is used as the current language description for the variable.



To provide a description for the other partition languages, you need to open the multilingual variable in the LANSA Editor and modify the definition as required.



If a description is not provided for any of the partition languages, the value supplied for the default language will be used when the multilingual variable is referred to in an application.

**Note:** In the example above, the maximum length is longer than either of the translations for English or French.

# 3.2 Where Can Multilingual Variables Be Used?

Multilingual variables can be used just about anywhere within LANSA. The following examples give an idea of some of the areas.

The following multilingual variables are used in the examples:

Name	Description
*MTXTYES	Multilingual version of "YES"
*MTXTNO	Multilingual version of "NO"
*MTXT12345	Multilingual version of "Value specified must be YES or NO"
*MTXT12346	Multilingual version of "Customer type must be A or B"
*MTXTINST01	Multilingual version of "Input Customer Number and press Enter"
*MTXTREPT07	Multilingual version of "CUSTOMER SALES HISTORY REPORT"

**Note:** Site naming standards for \*MTXT variables are recommended.

# 3.2.1 Data Dictionary

For example, if a field #YESNO was defined in the repository with a default value of \*MTXTYES, then by default whenever the field #YESNO is displayed on a screen interface it will contain the equivalent value of "YES" for the current language.

# 3.2.2 File Definition

A file level validation check confirms that a field called #CUSTYPE is an "A" or a "B" and issues textual error message \*MTXT12346.

# 3.2.3 RDML and RDMLX Programming

#Com\_owner.caption := \*MtxtCust01.Substitute(#Givename #Surname
#Limit.AsString)

In the above example, \*MtxtCust01 is a multilingual variable containing the following:

"&1 &2 has a limit of \$&3"

#Givename, #Surname and #Limit have values of Veronica, Brown and 2000 respectively.

The result at runtime would be a "Veronica Brown has a limit of \$2000"

# 3.3 Standard System Variables useful for Multilingual Development

These additional normal system variables aid in multilingual system development.

Name	Description
*LANGUAGE	Code of current language being used. Set to NONE when not running in a multilingual partition.
*LANGUAGE_DESC	Current language description
*LANGUAGE_IGC	Contains a "Y" if the current language requires DBCS/IGC support. Contains "N" in all other cases.
*LANGUAGE_LRTB	Contains a "Y" if the current language requires left to right support. Contains "N" in all other cases.  Note: LRTB languages are not supported in Visual LANSA
*LANGUAGE_RLTB	Contains a "Y" if the current language requires right to left support. Contains "N" in all other cases.  Note: RLTB languages are not supported in Visual LANSA

# 4. The Repository and Multilingual Applications

- 4.1 Specify a Field's Multilingual Attributes
- 4.2 Specify a File's Multilingual Attributes
- 4.4 Multilingual Validation Rules and Error Messages
- 4.5 Multilingual Database Design Considerations

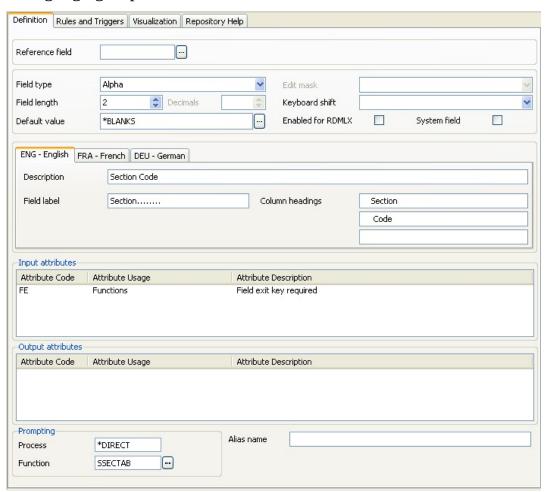
# 4.1 Specify a Field's Multilingual Attributes

When you create a field in the repository the field label, description and column heading for all partition languages default to the value entered for the current language.

The input attributes and output attributes for each partition language are defaulted based on the field type created.

After the field has been created the field definition can be reviewed and appropriate multilingual attributes can be added for each partition language.

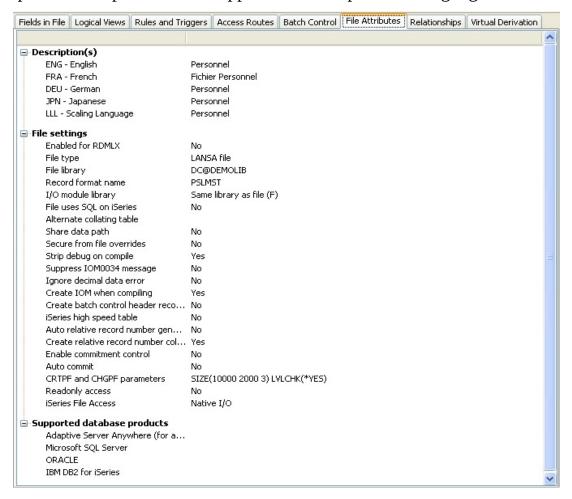
• If any of the languages is a DBCS language, then DBCS data can be specified for this language group.



### 4.2 Specify a File's Multilingual Attributes

When you create a file in the repository the physical file name defaults to the value entered on the create dialog for all partition languages. The description can be customized for each language by modifying the file attributes after the file has been created.

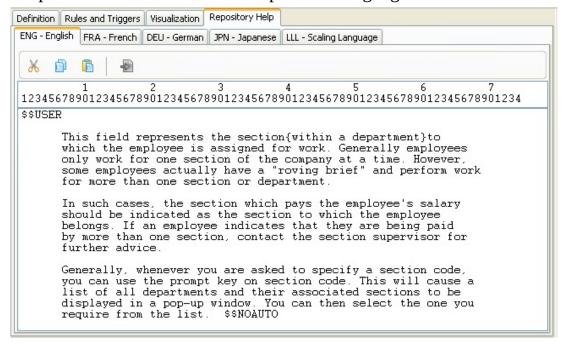
- If any of the languages is a DBCS language then DBCS data can be specified for this language group.
- Similarly, when any logical views are added to the file definition an appropriate description can be supplied for each partition language.



# 4.3 Multilingual Help Text

In multilingual partitions, the HELP text facility allows the entry of multilingual help text for fields, processes, functions, forms and reusable parts.

Help text can be entered for each partition language.



### 4.4 Multilingual Validation Rules and Error Messages

The following points should be considered when specifying field or file level rule and triggers:

## Error Message Text

When specifying multilingual error message text, use multilingual variables to specify the text. This way the textual content of the message will be determined at execution time and thus issued in the correct language.

## Error Message Numbers and Message Files

When specifying error messages by message number and message file, use only a message file that has been defined at the partition level as a user defined message file for the default development language. This will ensure the correct multilingual message file will actually be used at execution time, ensuring that the message appears in the correct language.

Refer to The Handling of System Messages Issued by LANSA for details of how this feature works.

## **List of Values Check**

Where a field can have a value that is language dependent, use multilingual variables in the list of values rather than literals.

### Range of Values Check

Where a field can have a value that is language dependent, use multilingual variables in the ranges of values rather than literals.

### Code File Lookup

Multilingual variables can be used as key fields in a file look-up check.

## **Simple Condition Check**

Multilingual variables can be used in an expression as replacements for alphanumeric literals.

### Call a Program Check

Multilingual variables can be used as parameters to dictionary checking programs.

### **Date Checking**

There does not appear to be any requirements for multilingual support in date

and date range checking validation rules.

### 4.5 Multilingual Database Design Considerations

Some important design considerations apply to developing truly multilingual applications. Consider the following simple sequence of screen panels representing a product inquiry:

**Specify Product Number** 

Key in Product Number and press Enter.

Product Number . . . . . . 64786

#### **Product Details**

Product Number . . . . : 64786

Description . . . . . 1 kg plastic bag of cane sugar

Short Description . . . : SUGAR 1KG

Price ..... 2.45

Stock on Hand ....: 7685

Initially the RDML program could be coded like this:

GROUP\_BY NAME(#PANEL001) FIELDS(#PRODNO #PRODES #PROSD! BEGIN LOOP

REQUEST FIELD((#PBINST01 \*OUTPUT \*NOID) #PRODNO)

FETCH FIELDS(#PANEL001) FROM\_FILE(PRODMST) WITH\_KEY(#PRODMST)

IF\_STATUS IS\_NOT(\*OK)

MESSAGE MSGTXT(\*MTXT12347)

**ELSE** 

DISPLAY FIELDS(#PANEL001)

#### ENDIF END LOOP

By setting up the correct multilingual attributes for this program you can instantly achieve the following variations in this program. Refer to Specifying a Field's Multilingual Attributes for details.

#### **Execute the application in French**

Specification du Numero de Produit

Tapez le Numero de Produit puis faites Enter

Numero de Produit ..... 64786

#### Details du Produit

Numero de Produit . . . : 64786

Description . . . . . : 1 kg plastic bag of cane sugar

Description Abregee ...: SUGAR 1KG

Prix . . . . . . . . . 2.45

Stock Disponible . . . : 7685

## **Execute the application in German**

Produkt Nummer Angeben

Geben Sie die Produkt Nummer ein und drucken Sie Eingabe

Produkt Nummer . . . . . . 64786

#### Produkt Detail

Produkt Nummer . . . . : 64786

Seschreibung . . . . . : 1 kg plastic bag of cane sugar

Kurzbeschreibung . . . . : SUGAR 1KG

Preis . . . . . . . : 2.45

Verfugbare Menge . . . . : 7685

#### Notice what's wrong?

The product description and short description shown are still in English.

This simple example illustrates a very important point.

LANSA can get all the details it knows about out on the screen in the correct language, but because this is intended to be a completely multilingual program, it should have stored the product details in two separate files:

- One for the non-language dependent product details (like price and stock on hand);
- The other for the language dependent product details (like the long and short descriptions).

If the program was coded like this (making use of the system variable \*LANGUAGE to determine the language):

GROUP\_BY NAME(#PANEL001) FIELDS(#PRODNO #PRODES #PROSD

#### **BEGIN\_LOOP**

REQUEST FIELD((#PBINST01 \*OUTPUT \*NOID) #PRODNO) FETCH FIELDS(#PANEL001) FROM\_FILE(PRODMST) WITH\_KEY(#PROPUTED) FETCH FIELDS(#PANEL001) FROM\_FILE(PRODLNG) WITH\_KEY(\*LAI)

IF\_STATUS IS\_NOT(\*OK)

MESSAGE MSGTXT(\*MTXT12347)

**ELSE** 

DISPLAY FIELDS(#PANEL001)

**ENDIF** 

#### END\_LOOP

The result would be a true multilingual system where side by side users could be running the same program in completely different languages.

#### **Execute the application in French**

Specification du Numero de Produit

Tapez le Numero de Produit puis faites Enter

Numero de Produit ..... 64786

#### Details du Produit

Numero de Produit . . . : 64786

Description . . . . . : 1 kg sac plastique de sucre

Description Abregee ...: SUCRE 1KG

Prix . . . . . . . . 2.45

Stock Disponible . . . : 7685

## **Execute the application in German**

Produkt Nummer Angeben

Geben Sie die Produkt Nummer ein und drucken Sie Eingabe

Produkt Nummer . . . . . . 64786

## Produkt Detail

Produkt Nummer . . . . : 64786

Seschreibung . . . . . : 1 kg Plastiksack mit Rohzucker

Kurzbeschreibung . . . . : ZUCKER 1KG

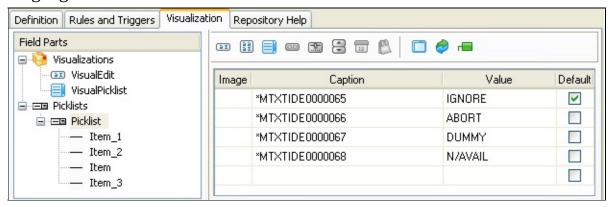
Preis ..... 2.45

Verfugbare Menge . . . . : 7685

# 4.6 Multilingual Field Design Considerations

#### **Using Field Visualizations**

When defining a field visualization with any associated text values, ensure you use multilingual variables to represent the field correctly in all the partition languages.



# 5. Creating Multilingual Applications using Functions

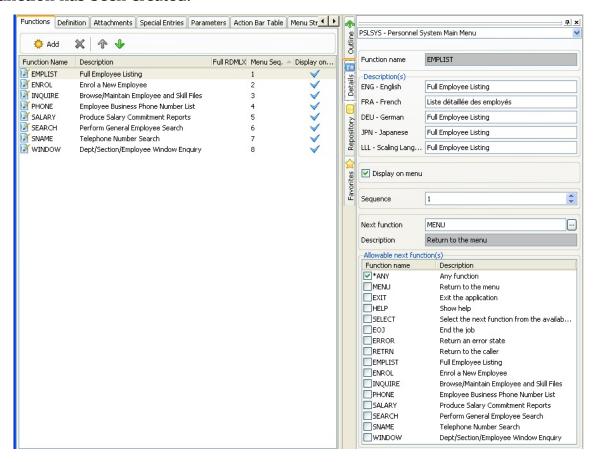
- 5.1 Specifying a Process or Function's Multilingual Attributes
- 5.2 Specifying Special Entries on a Process Menu
- 5.3 Specifying Process Parameters
- 5.2 Specifying Special Entries on a Process Menu
- 5.5 Multilingual Process or Function Design Considerations

## 5.1 Specifying a Process or Function's Multilingual Attributes

When you create a process, the process name defaults to the value entered on the create dialog for all partition languages. The process description can be customized for each language by modifying the process attributes after the process has been created.

• If any of the languages is a DBCS language then DBCS data can be specified for this language group.

Similarly, when any functions are created within a process, the function description entered on the create dialog is used a default description for all partition languages. The function description can be customized for each language by modifying the function details against the process where the function has been created.



# 5.2 Specifying Special Entries on a Process Menu

Any special entries created on against a Process definition can have a description entered for each partition language.

• If any of the languages is a DBCS language then DBCS data can be specified for this language group.

# **5.3 Specifying Process Parameters**

Process parameter descriptions are not language specific as they should never appear in an end user environment and are considered to be developer testing tool.

Process parameters cannot be defined in DBCS languages.

# **5.4 Specifying process Action Bars**

Any action bar or pull down menu options defined against a Process definition can have a description entered for each partition language.

• If any of the languages is a DBCS language then DBCS data can be specified for this language group.

### 5.5 Multilingual Process or Function Design Considerations

The following basic guidelines apply to creating and maintaining multilingual applications:

#### **Database Design Considerations**

Understand the difference between an application system that is designed to run in different national languages and one that is designed to run in true multilingual mode.

For instance, a software company in France might develop a system that they can sell to Germans (to run in only German) and also to Italians (to run in only Italian). This is a system designed to run in different national languages.

However, a Swiss company may wish to design a system where the end user can, at execution time, elect to run the system in French, German or Italian. This is a true multilingual system.

You will need to use multilingual support in both of these situations, but the type of system you require will make a large difference to the database that you design to support it.

#### **DBCS Considerations**

It is almost impossible to use a DBCS language on a machine that does not have DBCS support installed.

The best approach is to implement the system in all other required languages and port it to a machine that does have DBCS support. Then add the DBCS language to the partition definition and enter all dictionary, multilingual variable, file, process, function and component definition DBCS information before recompiling the application.

Note that an application compiled when the current language is a DBCS language may only execute on a machine with DBCS support.

## **RDML and RDMLX Programming Considerations**

The following considerations and guidelines apply to RDML and RDMLX development using multilingual attributes:

 Text should never be placed on screen panels. Use a field that has a multilingual variable as its default or current value.
 For example:

"Key Customer Number and press Enter."

is a classical piece of text from a screen panel. In a multilingual application

this text should be associated with a multilingual variables value:

DEFINE FIELD(#PANI782) TYPE(\*CHAR) LENGTH(76)
OUTPUT\_ATR(PBPI) DEFAULT(\*MTXTPANI782)
DISPLAY FIELDS((#PANI782 \*NOID \*OUTPUT) etc, etc ....)

- Text should never be placed on reports. Use the same approach as recommended for screen panels.
- Never issue text messages. Use \*MTXT variables or messages from user message files that are overridden by LANSA at execution time.
- Code description should be displayed using \*MTXT (including field visualizations) or values retrieved from an language specific file.
- Code values are usually NOT required to be multilingual but their associated description usual is required to be multilingual. Consequently code value can be used in RDML or RDMLX code without consideration of language.
- Be as flexible as possible. For example, the following code allows YES or NO to be specified in any language. It is assumed that field #YESNO is defined in the data dictionary with multilingual labels, column headings and \*MTXTYES as a default value.

```
REQUEST FIELD(#YESNO)

BEGINCHECK

VALUECHECK FIELD(#YESNO) WITH_LIST(*MTXTYES *MTXTI MSGTXT(*MTXT12345)

ENDCHECK

IF (#YESNO = *MTXTYES)

SUBMIT ... whatever ...

ENDIF
```

• If you are writing applications using processes and function for languages which typically have longer phrasing, such as German and Dutch, consider using the description (\*DESC) rather than labels (\*LABEL) to identify information on screens or panels.

#### 6. Tips and Techniques

We recommend that you read these Tips and Techniques before you start the development of your multilingual applications:

- 6.1 Use a "Scaling" or "Test Card" Language
- 6.2 Aligning Column Headings in Multiple Languages
- 6.3 Development of Extended Naming Standards
- 6.4 Use the Design View Language
- 6.5 If the Wrong Language (or No Language) Appears
- 6.6 Multilingual Action Bars
- 6.7 Translation Tables Used by Host Monitor and Export to PC
- 6.8 Supporting International Date Formats
- 6.9 Microsoft Windows Multilingual Operating Environment Guidelines
- 6.10 Multilingual Text Handling

# 6.1 Use a "Scaling" or "Test Card" Language

A useful technique in the development of standards and practices for multilingual applications development, is the use of a "scaling" or "test card" language.

To do this, you just invent a "new" language.

The recommended identifier is LLL and the description should be something like "Scaling / Testcard Language".

The LLL language sets the standards and scaling sizes for all other languages.

For example, when you put a new field in the dictionary you would also specify its description, label and column headings in the "LLL" language as rows of "LLLLLLL" or "XXXXXXXXXXX"s.

This information establishes the application designer's scaling limits for the maximum lengths of descriptions, labels and column headings.

All other language's details, whether filled in now or later, must conform to the scaling limits established by the designer in the scaling or test card language.

It may also be useful to set up the environment for application developers so that they actually do all initial development in the scaling language.

This may have several benefits for the developers:

- They are continually reminded that they are working on a multilingual application system.
- They are effectively prevented from performing development in just one specific language, helping to stop any natural bias towards a specific language from being reflected in their work. They develop and perform basic functional and layout tests in the scaling language. Only after this work is complete do they elect to see the application running in a "real" language or languages.
- They continually see the scaling limits set by the designer when laying out screen panels and reports. This again helps to prevent any natural language bias creeping into their work and aids in understanding why screen panels and reports have been laid out in a possibly foreign manner.

#### **6.2 Aligning Column Headings in Multiple Languages**

When reports and screen panels are designed in multilingual applications, "width" details for the column headings, labels, descriptions, etc are established from the widest available multilingual definition.

In bi-directional languages the width is calculated from the right, in all other languages the width is calculated from the left (including shift-out/shift-in characters for DBCS languages).

The need to make space for the "widest" language on screen panels and reports can have a bearing on how multilingual details such as column headings are recorded into the dictionary.

Consider an alphanumeric field of length 3 that has a multilingual column heading that is, for example, 20 characters wide in German.

Ignoring the fact that using a 20 character wide column heading for a 3 character wide field is fairly unlikely and violates the guidelines in the *LANSA Application Design Guide*, consider if it was placed onto a report, the space used would be like this (where GG..GG is the German column heading and XXX the printable field value):

#### GGGGGGGGGGGGGGG XXX

Now if the system also allowed English and Hebrew, and both of these column headings only used 5 characters, the run time change to either of these languages would produce a result like this:

English .... EEEEE XXX

Hebrew ..... HHHHHH

Both of these results would look really odd on the report.

The solution is to remember that column headings should generally be centered within the space used by the "widest" language entry.

The use of a scaling or test card language can aid in doing this.

If these column headings were all centered in the 20 characters, the results

would be like this (which are far more acceptable):

German ..... GGGGGGGGGGGGGGGG

XXX

English .... EEEEE

XXX

Hebrew ..... HHHHH

XXX

#### 6.3 Development of Extended Naming Standards

It is strongly recommended that naming standards cater for some special objects associated with developing multilingual applications.

#### **Multilingual Variables (\*MTXT variables)**

A naming standard for multilingual variables should be established.

A possible standard (which should be refined by you) is:

\*MTXTttttnnn

#### where:

- *tttt* indicates the type/usage of the variable with values like:
  - REPT Report Title
  - REPI Report Instruction
  - PANT Panel Title
  - PANI Panel Instruction
  - MSGT Message Text
  - and so on.
- *nnn* is an identifying sequence number.

#### **Panel and Report Literals**

Multilingual systems tend to have a large number of constants or literals defined as fields in the dictionary.

These are generally used for placing multilingual constants on screen panels and reports.

They usually have a default value that is established from a multilingual variable.

A possible standard (which should be refined by you) is:

ttttnnn

#### where:

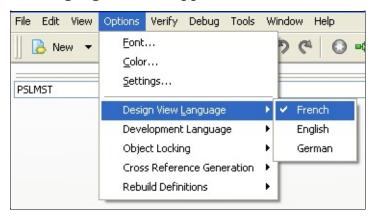
- *tttt* indicates the type/usage of the variable with values like:
  - REPT Report Title
  - REPI Report Instruction
  - PANT Panel Title
  - PANI Panel Instruction

- MSGT Message Text
- etc, etc.
- *nnn* is an identifying sequence number.

This standard would mean that multilingual variable \*MTXTREPI007 would be defined as field called REPI007 in the data dictionary with a default value of \*MTXTREPI007.

#### 6.4 Use the Design View Language

To review any interfaces designed for your application in all the supported languages use the Design View Language from the Visual LANSA options menu to change the design view. This will set the lagnauge displayed in the design view tab as required so you can ensure all screen interacts accommodate the text values in all languages to be supported.



#### 6.5 If the Wrong Language (or No Language) Appears

Most multilingual information is "locked in" at compile time. This is necessary to ensure good performance in multilingual applications.

If you are testing an application and details appear in the wrong language, or do not appear at all, you should always do the following before reporting the problem to your product vendor:

- Check all fields, files, processes, function and components involved have multilingual details.
- Check all \*MTXT variables involved have multilingual details.
- Check all predefined messages involved have multilingual details. Remember that all text not defined in a specific language context is assumed to be in the Current Partition language. This includes such things as string literals in RDML, filenames such as DC@M01 and validation error messages. For this text to be interpreted correctly on the Master system, all processes, functions and components must be compiled when logged on with the correct Current Partition language. For example, if your text is in French, your Message File name is DCàM01 you should compile with French as your Current Partition language.
- Recompile all processes, functions and components involved.
- If problem still exists, contact your product vendor.

# **6.6 Multilingual Action Bars**

Multilingual Action Bar style processes can be slow when running in interpretive mode.

This is generally not a problem while testing applications, but it is strongly recommended that all multilingual action bar processes be compiled before being used in production systems.

#### 6.7 Translation Tables Used by Host Monitor and Export to PC

If you are developing your application using Visual LANSA with a IBM i Master Repository, you will need to transfer data from one platform to another. In IBM i Master system the definitions required to run the Host Monitor and to export using the PC export can be created using the LANSA REQUEST(PCMAINT), or they can be created automatically by performing System Initialization or Partition Initialization from Visual LANSA.

The names of the translation tables used to convert data to and from EBCDIC and ASCII are included in these definitions. These tables are only used by export to PC and the Host Monitor and NOT by LANSA Open or LANSA/SuperServer which have their own tables.

Refer to Creating PC Definitions and Changing PC Definitions in the *LANSA* for iSeries User Guide for further details.

#### **6.8 Supporting International Date Formats**

The client's date format will be automatically passed to the server. If the date formats are different (e.g. MDY versus DMY), the server will automatically return data in the client's format.

The client's format can be changed from this default by specifying the x\_run parameter DATF=. You will find information about this parameter in the Standard X\_RUN Parameters in the *Technical Reference Guide*.

If the client and server date formats are different, such as between USA and UK clients, date format validation rules specifying exact formats will fail. For example, DDMMYY may be returned as MMDDYY. Where clients will be using different date formats, the date format SYSFMT is recommended.

# 6.9 Microsoft Windows Multilingual Operating Environment Guidelines

When developing or executing a multilingual application in LANSA you are required to have a Client PC with an Operating System running under the language you wish to develop or execute your application.

The reason for this is that you need a keyboard that can produce all the characters for the language and you need the Windows System Codepage to match the language you are entering.

The Visual LANSA IDE allows all languages to be entered on a machine of any language, provided operating system support for that language has been installed.

When executing an application, note that only data that matches the Windows System Codepage can be displayed correctly, unless NCharNChar and NVarChar fields are being used. If a database contains Chinese and English data in one NChar column of different rows, both languages will display correctly on a form. However, Multilingual Variables will only display correctly if the LANG= parameter is appropriate for the language the PC is using.

For data stored in Alpha, String or Char fields, some languages CAN co-exist. These are languages that share the same codepage. For example, English and French both use codepage 1252. Thus French and English data in one String column of different rows will BOTH be able to be displayed on BOTH French and English PCs.

But in general, data needs to either all be in the same language, it needs to be keyed by language, or it needs to be stored in an NChar or NVarChar.

Also note that there are a set of characters that are invariant on all Windows codepages. These WILL display correctly in all languages. The code range is 0x20 to 0x7F. This contains principally the English alphabet, numbers and punctuation. It may suit the design of an application to limit data entry to these characters, and just have the Field Descriptions, Multilingual text variables, etc displayed in a language to match the operating system. Limiting the data entry could be enforced using LANSA Repository rules.

#### 6.10 Multilingual Text Handling

The text processing described in this section is used by LANSA when:

- checking in/out multilingual text in all LANSA object types (V11 SP4 and later).
- when exporting to Windows.
- when compiling text in all objects on Windows and RDMLX objects on IBM i under LANSA 11.0 and later.

#### **Recommendations:**

- 1. On the IBM i master, ensure that the CCSID associated with each partition language matches the intended Windows execution environment. See table in 6.10.1 Does LANSA know the correct CCSID to use for Multilingual text? for the LANSA defaults.
- 2. Don't use literals in RDML/RDMLX code. Use Multilingual Variables (MTXTs) to ensure VARIANT characters are correctly converted to the CCSID that will be used at execution time. Refer to 6.10.7 Literals in RDML for further details.
- 3. There is a third recommendation that is particularly useful for new installs and users. That is to ensure that the language codes used are the ones listed in the table in 6.10.1 Does LANSA know the correct CCSID to use for Multilingual text? or if not listed, they are the IBM provided language codes. LANSA looks up the CCSID using the language code as part of the algorithm described in 6.10.1 Does LANSA know the correct CCSID to use for Multilingual text?

# 6.10.1 Does LANSA know the correct CCSID to use for Multilingual text?

Partition multilingual languages are processed in the *Work With Partition Multilingual Attributes* menu item. The two values, iSeries EBCDIC CCSID and Windows ANSI code page are stored in F60CHR in DC@F60.

iSeries EBCEIC CCSID
Windows ANSI code page

Any future references to F60CHR (or X60CHR) in this section is referring to the IBM i EBCDIC CCSID and Windows ANSI code page values specified for a language.

You are responsible for ensuring the IBM i F60CHR values and windows X60CHR values are valid (and up-to-date on the PC if changed on the IBM i). LANSA uses the default values listed in the table following when X60CHR has a blank value.

DBCS languages must be set to the correct code for the language, otherwise DBCS translation will not be performed correctly by Host Monitor. For example, Korean MUST use KOR.

All languages that are in the following table should use the listed code otherwise the defaulting of the IBM i CCSID and Windows codepage will not be correct. If a language is not listed, then X60CHR must be specified. Amongst other things this affects Host Monitor, multilingual text and source code multilingual text.

F60CHR in DC@F60 is set using *Multilingual Partition SAA / CUA Defaults*. The two fields are *iSeries EBCDIC CCSID* and *Windows ANSI codepage*. For further information, see Review or Change a Partition's Multilingual Attributes in the *LANSA for iSeries User Guide*.

If a default value is needed on Windows, it is calculated by first by looking up the language in ccsid.txt. If the language is not found there, the current operating system codepage is used.

This table is formatted as follows:

LANSA Language	IBM I Language	IBM i CCSID	Windows Codepage
ENG	ENU	37	1252
ARA	ARA	420	1256
CAT	CAT	284	1252
CES	CSY	870	1250
CZE	CSY	870	1250
DAN	DAN	277	1252
DEU	DEU	273	1252
DUT	NLD	37	1252
ELL	ELL	875	1253
ESP	ESP	284	1252
FIN	FIN	278	1252
FRA	FRA	297	1252
FRB	FRB	500	1252
FRC	FRC	500	1252
GRE	ELL	875	1253
HEB	HEB	424	1255
ITL	ITA	280	1252
JPN	JPN	5026	932
KOR	KOR	933	949
NOR	NOR	277	1252
POR	PTG	37	1252
SCHI	CHS	935	936
SFRA	FRS	500	1252
SWE	SVE	278	1252

TCHI	CHT	937	950
THA	THA	9030	874
TRK	TRK	1026	1254
TUR	TRK	1026	1254

**On IBM i** the fallback position is the same table as for Windows, using the IBM i CCSID. But prior to that an algorithm is used to obtain a best default CCSID if possible. The algorithm is currently as follows and may be changed at any time by LANSA.

```
static void GetCCSID( char * pvchLanguage, UTC_CodePage * pCodePage )
 Get the job CCSID
 /* Check if the Job CCSID overides the default */
 if ( JobCCSID == 500 || JobCCSID == 1148 )
   if (language is one of FRB, FRC, DEU, DUT, FIN, ITL, NOR, POR, SWE
     Set codepage to the Job CCSID (500 or 1148)
  }
 if (language is ENG)
   if ( JobCCSID == 285 || JobCCSID == 1140 || JobCCSID == 1146 ||
      JobCCSID == 500 || JobCCSID == 1148 )
     Set codepage to the Job CCSID
   else
     Set codepage to 037
```

```
else if (language is FRA)
 if ( JobCCSID == 1147 || JobCCSID == 500 || JobCCSID == 1148 )
   Set codepage to the Job CCSID
 else
   Set codepage to 297
else if (language is ESP)
 if ( JobCCSID == 1145 || JobCCSID == 500 || JobCCSID == 1148 )
   Set codepage to the Job CCSID
 else
   Set codepage to 284
else if (language is JPN)
 /* This is a workaround for QLGRTVDC which returns 5026 unless the
   QTQ_DEFAULT_CCSID environment variable is explicitly set to 5035 *
 if (JobCCSID == 5035)
   Set codepage to the Job CCSID
if (codepage has been set)
 return;
/* Convert LANSA Lang Id -> ISeries Lang Id */
```

```
if (language code found in table)
   Set codepage to the CCSID in the table.
   Set iSeries language code to the value in the table (i.e. map from LANSA
  }
else
 Set iSeries language code to the LANSA language code.
if (iSeries language code is a valid IBM language code)
 /* QLGRTVDC only works on systems >= V5R2 */
 P8601_GetOSReleaseNumber(&releaseNo, &chRetCode);
 if (release >= V5R2 use IBM language code to obtain the "Input CCSID")
        Set codepage to the Input CCSID
 /* else the default will be used as set above when language code found in *
 /* Note that if the language code is NOT found in the table and the OS400
 /* release is BEFORE V5R2 then the codepage is not set. This implies that
 /* in this case an override MUST be provided in F60CHR.
else
 /* Unrecognised language requested. Use default JobCCSID. No distinction
   is required at the moment between LLL (special language code) and
   other unrecognised language codes */
 if (JobCCSID != 0)
   Set codepage to the Job CCSID
  }
  else
   Set codepage to 037
 if (language is not LLL)
```

```
{     if ( message not yet issued for this language )
     {
        Issue log message
        "Language code not recognised: %s. Using default Job CCSID %.
        vchISeriesLang,
        *pCodePage );
    }
}
```

### **6.10.2 Changing F60CHR defaults**

The F60CHR defaults are changed using the *Work with Partition Multilingual Attributes*. Choose the language and the second page allows the IBM i EBCDIC CCSID to be entered. It is critical that this truly reflects how you use languages on your system.

For example, if English and French are both run in CCSID 037 (English US), then the French default CCSID needs to be set to 037. If this is not done, 297 will be used. If 297 is used, the French data will be incorrect on Windows.

#### 6.10.3 Non-multilingual Issues

Non-multilingual partitions can only work when the system codepage on Windows is matched to the default job CCSID of the logged in user on IBM i. For example Japanese Windows codepage 932 with Japanese IBM i CCSID 5035 will work but English Windows codepage 1252 with Japanese IBM i CCSID 5035 will not work.

The reason for this is there is no override field X60CHR to specify the codepage/CCSID to use so LANSA can only use the current or system codepage/CCSID.

#### **6.10.4 LANSA Backward Compatibility**

You cannot export data from IBM i Version 11 SP4 and above directly to an earlier Windows Version11 system.

If you need to do this:

- 1. Move the data from the IBM i V11 SP4 and above to an IBM i V11 SP3 system
- 2. Export the data from the V11 SP3 IBM i to the Windows V11 SP3.

The following message (from 479 in x\_txt<lang>.dat) is displayed if you attempt to go directly from IBM i V11 SP4 to Windows V11 SP3 or earlier:



#### 6.10.5 Orphan Language Record Handling

Orphan language records are created when a language is removed from a partition after that language has been added to a LANSA object. When the language is removed from the partition, the language text for all LANSA objects is left in place. Thus orphan records are created.

LANSA has never had any special handling of orphaned language text records and this has not changed. Please note:

- No data is lost if the deleted language is added back to the partition immediately before any LANSA object creation or modification is performed. This is purely a matter of what other work is being performed on the LANSA system with LANSA providing no system features to control what is occurring. If a LANSA object is created or modified before the language is added back in, data may be compromised.
- IBM i export and check out only send LANSA object information for languages that exist in DC@F60. Thus orphaned language text is not sent.
- PC Export , using the EXPORT\_OBJECTS Built-In Function or the Deployment Tool, exports all language text including the orphaned records.
- PC Import only imports language text which has will have a matching language in LX\_F60 by the end of the import. If an object already exists then it is deleted first, including orphaned language text, and thus the object will not have orphaned language text after the import.
- The LANSA Editor does not display the orphaned language text nor is it deleted when an object is changed.
- PC Check In sends all language text that exists and IBM i Check In takes whatever it is given, but the IBM i user interface only shows text for languages that exist in DC@F60.
- When a LANSA object is deleted, the orphaned language text is deleted.
- LANSA reorg on IBM i removes orphaned language text.

We have no special handling for orphaned language records. It is not a feature of LANSA to maintain these records, that is, IBM i Export does not export them. Therefore we have no need to work out the correct CCSID for these texts when for example, IBM i Check In receives them. The current behavior is that the default language's CCSID will be used.

#### 6.10.6 Help Text

Help text multilingual support has a special issue because it contains keywords that the IBM i must recognize so that the help text can be displayed correctly. These include \$\$HI=, \$\$TECH and the attribute flags such as '{' and '}'.

The IBM i program must recognize these keywords and attributes in the help text as the code point of these characters in CCSID 037. This is because the IBM i program has been written in CCSID 037.

The help text also contains the actual text that the end-user sees. This text will be in the specific language.

For example the help text for a field can contain English text in English, Japanese text in Japanese and French text in French. But the same IBM i program must understand the keywords and attributes in all these three languages.

For check Out, the special formatting characters ( $%\{@\sim\}|$ ) and the directive prefix (\$\$) are first converted to the CCSID of the help text. The complete help text including directives and formatting can then be converted to UTF-8 by the IBM i check out processing.

For check in, the complete help text for a language is first converted from UTF-8 to the CCSID of the language. This text is then scanned and the formatting characters and directives replaced by the CCSID 37 equivalents. This text can then be written to the IBM i help files. This text will contain the directives and formatting that can be interpreted by the IBM i program which displays help text.

Where specific formatting characters are defined, for example, \$\$HI= , the standard replacement is not done.

The replacement characters for the formatting and directive are held on IBM i file DC@FHR. The information is held by target CCSID. It contains the CCSID 37 character and the equivalent in the target CCSID. Characters held are "\$% {@~}\".

#### 6.10.7 Literals in RDML

Don't use literals in RDML/RDMLX code. Use Multilingual Variables (MTXTs) so that VARIANT characters are correctly converted to the CCSID that will be used at execution time. For example, @ is a variant character, therefore it should be in an MTXT.

- If a literal is used in RDML, it is converted using the CCSID derived from the language displayed in the LANSA Editor's Design View at the time the object was compiled. If it does not match the Job CCSID, errors will occur.
- If an MTXT is used in RDML, each language instance of the MTXT is converted using the CCSID derived from the language of the MTXT instance.

Clearly, an issue will not be present if the literal in question is invariant in the languages used by the application. But let's say that the literal contains the @ character. If the application currently used US English (037) and Canadian French (500) then the @character is invariant. But, if French French (297) was added to the application, the @ character would now be variant. If this was compiled with US English as the Design View Language, then French French would not display the @ correctly.

# 7. Special Facilities

- 7.1 Multilingual Built-In Functions
- 7.2 The SREV Field Attribute
- 7.3 The \$\$RIGHT Help Text Keyword

# 7.1 Multilingual Built-In Functions

Some multilingual applications require this Built-In Function:

Name Description

REVERSE Reverse order of data in an alphanumeric field.

New multilingual Built-In Functions will be developed, particularly to handle DBCS data.

#### 7.2 The SREV Field Attribute

The SREV field attribute is provided to assist the collating sequence of bidirectional data stored in database files.

It indicates that the field's contents should be reversed as it crosses the interface between an RDML application program and a database I/O module or \*DBOPTIMIZE generated inline I/O logic.

This means that everywhere the field is referenced at the RDML (or programmer) level it is in a reversed format to what it is at the I/O level or the database level.

The following should be noted about using this attribute:

- Use attribute SREV as either an input attribute or an output attribute at the data dictionary level only. Use of this attribute in any other context, such as RDML DEFINE or OVERRIDE commands is meaningless. The SREV attribute only applies to a field in one place: at the database I/O interface level.
- Take care when changing an existing field from, or to, an SREV field. Data already stored in database files may conflict with this definition and require (re)alignment before it can be effectively used.
- Since SREV applies at the I/O level, it can have a bearing on other operations performed at the I/O level. Such operations must be aware that a field is in reversed format as soon as it enters the I/O level, and remains in reversed format until it leaves the I/O level.

Operations that may be affected by this consideration include:

- File and dictionary level validation checks may need to reference the field in reversed format.
- User defined virtual code.
- Avoid giving fields that have the SREV attribute default values that are directional in nature. A field's default value has a dual role at the RDML level and at the I/O level. Where a field's default value has a directional content, there will almost certainly be a conflict that may produce strange or unexpected results.
- The SREV attribute can only be applied to alphanumeric fields.

# 7.3 The \$\$RIGHT Help Text Keyword

Use of this keyword in help text details indicates that all help text should be right aligned just before it is displayed, regardless of how it is input by the user.