

Getting Started



MULTILIZERTM

with software localization

using MultilizerTM

Innoview Multilizer™

Getting Started With Internationalizing Software

First Edition, February 1998, Draft

Copyright 1997-98 Innoview Data Technologies Oy. All rights reserved.

Multilizer™ is a trademark of Innoview Data Technologies, Ltd.
Delphi is a registered trademark of Borland International, Inc.
C++Builder is a trademark of Borland International, Inc.

Program license agreement

Innoview Multilizer™ for Delphi and C++Builder

1. **DEFINITIONS.** Software means the enclosed Innoview Multilizer for Delphi and C++Builder software program and related documentation. Application Program means a Delphi program designed to accomplish a specific purpose or perform a specific task for the end user. The parties to the agreements (Agreement) are: Innoview Data Technologies Oy (Innoview) and the purchaser (you as a Licensee).
2. **LICENSE.** Innoview agrees to grant to you and you accept a non-exclusive, non-transferable license to use Software to the extent for which a license fee has been paid. You may not sublicense, rent, distribute, lease or otherwise assign your rights in the Software. The license is not a sale of the Software and Innoview retains full title to the software recorded on the media contained in the Software. Innoview reserves all rights not expressly granted to you in this License. Innoview grants to you the right to use one copy of the Software specifically for the purposes described in this Agreement. Except as otherwise expressly approved in writing by Innoview, you may not use the Software at the same time on more computers or computer terminals than the number of authorized copies of this Software that you have purchased.
3. **TERM.** This License is effective from the date you open this package or use the Software, and shall remain in effect until terminated. You may terminate this License by destroying all complete and partial copies of the Software in your possession. The license shall terminate immediately if you fail to comply with any of its terms, in which case you will certify to Innoview in writing that, to the best of your knowledge, the original and all copies or partial copies of the Software have been destroyed or returned to Innoview.
4. **RIGHTS IN THE SOFTWARE.** You acknowledge that the Software and any copies of it, regardless of the form or media in which the original or copies may exist, are the sole and exclusive property of Innoview, and you further acknowledge that the Software, including the code, logic and structure, constitute valuable trade secret rights that belong to Innoview. You agree to secure and protect the Software consistent with the maintenance of Innoview's rights in it, as set forth in this License. By accepting this License, you do not become the owner of the Software; Innoview retains full title to the software recorded on the media contained in the Software.
5. **COPIES AND OTHER RESTRICTIONS.** The Software is protected by copyright law and international treaty provisions. Notwithstanding the copyright, the Software contains trade secrets and confidential information of Innoview. You agree not to disclose or otherwise make available any part of the Software to any third party. You may make copies in machine-readable form of the computer program which is part of the Software, provided that the copies are used only for back-up or archival purposes, that no more than two complete or partial copies exist at any time, and that the copies contain the original copyright notice. You may make unlimited copy of the Application Software and deliver them to the end user with no additional fees or royalties payable to innoview.

6. **DISCLAIMER OF WARRANTY.** The Software is provided 'as is' without warranty of any kind, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Further, Innoview does not warrant, guarantee, or make any representations regarding the use, or the results of the use, of the Software or its documentation in terms of correctness, accuracy, reliability, currentness, or otherwise. The entire risk as to the results and performance of the software is assumed by you. The above are the only warranties of any kind; no oral or written information or advice given by Innoview or its employees shall create a warranty or in any way increase the scope of this warranty, and you may not rely on any such information or advice.
7. **LIMITATION OF LIABILITY.** You assume full and sole responsibility for any use you make of the Software and you bear the entire risk of any error in its functioning. You agree that regardless of the cause of any error or the form of any claim, your sole remedy and Innoview's sole obligation shall be governed by this agreement, and in no event shall Innoview's liability exceed the amount you paid for the software alleged to have given rise to the liability. Innoview shall not be liable for any direct, indirect, consequential, or incidental damages (including damages for loss of business profits, business interruption, loss of business information, and the like) arising out of the use or inability to use the software, even if Innoview has been advised of the possibility of such damages. Because some states do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitation may not apply to you.
8. **U.S. GOVERNMENT RESTRICTED RIGHTS.** The Software is provided with RESTRICTED RIGHTS. Use duplication or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(2) of the commercial computer Software – Restricted Rights Clause at FAR 52.227-19 and in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFAR 252.227-7013. Contractor/manufacturer is Innoview Data Technologies Oy, Eteläranta 14, 00130 Helsinki, Finland.
9. **GENERAL.** This license, including the Disclaimer of Warranty and Limited Warranty, merges all prior written and oral communications regarding the Software and sets forth the entire agreement between you and Innoview, unless otherwise stated in a separate written agreement. This License shall be construed in accordance with the internal laws of Finland and all disputes shall have exclusive venue in the state court in Helsinki, Finland. If any term of this License shall be found invalid, the term shall be modified or omitted to the extent necessary, and the remainder of the License shall continue in full effect.

YOU ACKNOWLEDGE THAT YOU HAVE READ THIS AGREEMENT, UNDERSTAND IT AND AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS. YOU FURTHER AGREE THAT IT IS THE COMPLETE AND EXCLUSIVE STATEMENT OF THE AGREEMENT BETWEEN US WHICH SUPERSEDES ANY PROPOSAL OF PRIOR AGREEMENT, ORAL OR WRITTEN, AND ANY OTHER COMMUNICATIONS BETWEEN US RELATING TO THE SUBJECT MATTER OF THIS AGREEMENT.

Table of Contents

| | | |
|----------|--|-----------|
| 1 | USING THIS MANUAL | 7 |
| | PREFACE | 7 |
| | CONVENTIONS USED IN THIS BOOK..... | 8 |
| | <i>Text types</i> | 8 |
| | <i>Symbols</i> | 8 |
| | INSTALLATION | 10 |
| | GETTING HELP | 12 |
| | <i>On-line help</i> | 12 |
| | <i>Internet</i> | 12 |
| | <i>Registration</i> | 12 |
| | <i>Technical Support</i> | 13 |
| 2 | OVERVIEW | 15 |
| | LANGUAGE SUPPORT | 16 |
| 3 | BASIC CONCEPTS..... | 17 |
| | INTERNATIONALIZATION | 17 |
| | LOCALIZATION..... | 18 |
| 4 | COPING WITH DIFFERENT LANGUAGES | 20 |
| | CHARACTER SETS | 21 |
| | <i>Single byte character sets</i> | 21 |
| | <i>Multi or double byte character sets</i> | 23 |
| | <i>Bi-directional character sets</i> | 23 |
| | TEXT INPUT | 25 |
| | <i>Western languages</i> | 25 |
| | <i>Far Eastern languages</i> | 25 |
| | <i>Middle Eastern languages</i> | 25 |
| 5 | COUNTRY SPECIFIC ITEMS | 27 |
| 6 | MORE ABOUT LOCALIZATION | 28 |
| 7 | HOW DOES MULTILIZER WORK? | 29 |
| | DICTIONARY..... | 30 |
| | <i>Language Manager</i> | 30 |
| | <i>Dictionary components</i> | 31 |
| | TRANSLATOR..... | 32 |

CONCERNS IN BUILDING A DICTIONARY 33

Coping with words with several meanings..... 33

Native language concerns 35

APPENDIX A GLOSSARY..... 36

1

Using this manual

Preface

The purpose of this Getting Started manual is to familiarize you with Innoview Multilizer™ and the concepts and technics that are behind software internationalization.

This manual gives a short overview of the world's languages, which is a central issue in software internationalization. Furthermore, the different scripts are discussed. Later these issues are described in the context of developing software.

After the introduction to software internationalization related concepts, the Multilizer technology is introduced to the reader. It gives an overview of the technical solutions, which make it flexible and scalable, even when working with multilingual – or multicultural – software.

This chapter give important information on the following issues:

- Text style and symbol conventions used in this book
- Installation of Innoview Multilizer into your computer
- Where to get support.

At the end of this manual, there is a glossary of terms and concepts used in this book.

Conventions used in this book

Text types

The following typographical conventions have been used in this book.

Names of windows, menu options and keybuttons are printed in **bold Sans serif**.

Texts for figures and references to chapters and sections in this guide are shown in *italic Sans serif*.

Programming language related items are shown in the following manner.

- Names of components, component properties, procedures, functions are shown in **bold monospaced font**.
- Code listings and URLs are shown in `monospaced font`.

Symbols



This symbol indicates that the information given applies to Standard Edition only. In the front of a header it applies to the whole chapter, otherwise it applies to the current paragraph.



This symbol indicates that the information given applies to Professional Edition only. In the front of a header it applies to the whole chapter, otherwise it applies to the current paragraph.



MORE INFO

More info symbol is used, when there is additional information available either in the appendices of this document, in Multilizer online help or on Multilizer web-pages at: <http://www.innoview-data.com>



NOTE!

The note symbol is used, in order to give emphasis for certain tasks or issues, which are of big importance in the current topic.



The text marked with the Tip symbol gives useful hints, which may simplify tasks described in the current chapter.



The Warning symbol is used, whenever there may be a possibility to lose data or experience other kind of damage. The normal context for this symbol indicates that, you may lose your translation data if you proceed.



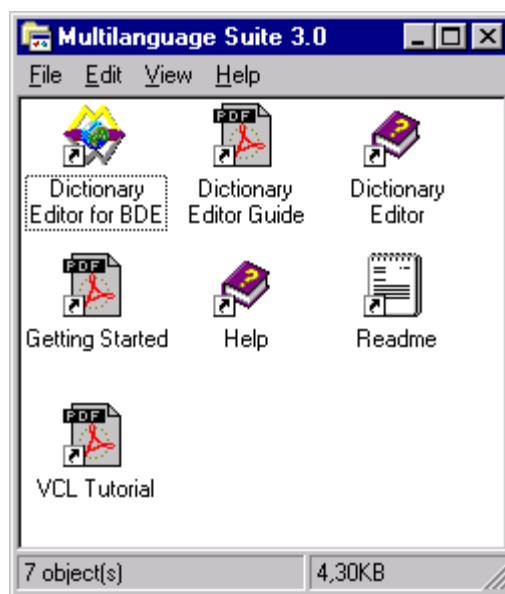
This symbol is used in issues describing different character sets. Character sets are one central issue to be taken in consideration, when internationalizing software.

Installation

To install Innoview Multilizer:

1. Insert the installation CD in the CD-ROM drive.
2. Run the setup program (MULTI.EXE).
3. Follow the instructions that appear on the screen.

A new program icon folder named 'Multilizer 3.0' (you can specify a different folder during setup, if desired) is created and the following program and document icons are put in it (the content of the folder may vary, depending of the installation options chosen in the setup):



Program folder created in Multilizer setup.

Help

The online help file. From this online help, you find information on component use. The help covers everything from the component installation to object hierarchy. It provides the user with examples and hints on using the components.

Language Manager

Language Manager online help.

**Language Manager
for BDE (or ODBC)**

Executes Language Manager utility, which is used for creating, maintaining and editing dictionaries.

Language Manager 3.0. is a 32-bit Windows application. To use Innoview Language Manager, you must have one of the following:

- Microsoft® Windows NT version 3.51 or later
or
Microsoft® Windows 95

Readme

A text file that describes changes to the documentation and new features added after the manual was printed. Please read the file before using Multilizer.



After you have installed the software, you have to add the Multilizer components to the IDE of your compiler. In addition, the online help may be linked to the help system. See more on the online help topic 'Getting started'.

Getting help

On-line help

This manual is a getting started manual. It does not contain any reference information about Multilizer. For complete instructions on using Innoview Multilizer, see the online help.

Internet

You can also get more information and tips from our Internet site.

<http://www.innoview-data.com/multilanguage/>

Currently all the WWW files are located in Finland. In the future we will duplicate downloadable files to North America and Asia. Check the current status from our Internet site.

Registration

Although the Multilizer package contains the registration card, we strongly suggest that you register the software at our web site:

<http://www.innoview-data.com/multilanguage/register.htm>

This will enable your personal account in our Internet server. Using the account you can download new versions and bug fixes. You can also join to mailing list that keeps you updated about Innoview Multilizer and multilingual globalization technology.

You can download the newest Multilizer version at (your have to register your software first at our Internet site)

<http://www.innoview-data.com/support/default.htm>

Technical Support

Useful web- addresses

If you have a question about Innoview Multilizer component, first look in online help. Check it from the FAQ-center:

<http://www.innoview-data.com/multilanguage/faq.htm>

and the bug list (you have to register your software first at our Internet site):

<http://www.innoview-data.com/support/default.htm>

If you still cannot find the answer, contact Innoview Product Support by email:

multilanguage@innoview-data.com

Sending your program source to tech. support

Try to create a simple program that demonstrates your case. ZIP the source code files (*.PAS, *.DFM and *.DPR) and the dictionary files (*.DEP, *.TXT or *.DB or *.DBF) into a single zipped file and email it to our technical support.

Please, include the serial number of your Multilizer, compiler version used and the operation system (version/language). This will greatly speed up the solving of your case. If you have downloaded Multilizer updates, please remember to inform us your current version of it (You find it on the first line of readme.txt)

Any source code we obtain through bug reports are handled strictly confidential.

Other ways to contact us

If you do not have email access you can contact us by fax, phone or mail:

Innoview Data Technologies Oy
Etelaranta 14 A 6
FIN-00130 Helsinki
Finland

Phone: +358 9 4762 0550 (GMT +2.00)

Fax: +358 9 4762 0555

Web: www.innoview-data.com

2

Overview

Innoview Multilizer provides an easy and flexible way to localize your applications. Multilizer goes beyond localization – it makes your programs multilingual. A multilingual application supports multiple languages and user can switch the language on the fly.

Multilizer uses unique *dictionary-translator-architecture* where the actual layout of the form and translation table are kept separate. It makes it possible to support multiple languages with one resource (using just the standard .DFM files). It is also possible to reshape forms without breaking the translation any way and add new languages without using Delphi or C++Builder.

There are two different editions of Multilizer: *standard* and *professional*. This manual covers the both editions. Whenever there is a difference in the features depending on the edition, it is marked either with the Pro symbol or with the Std symbol. To see the symbols, cf. *Previous chapter*.

Both Multilizer editions work using the same technology – the difference is based on supported languages.

Language support

The standard edition supports all languages based on European character sets. These are:

- Latin character set. e.g. English, French, German, Spanish, Swedish, Finnish, Turkish.
- Cyrillic character set, e.g., Russian and Ukrainian.
- Greek character set, e.g. Greek.



The professional edition adds support for all languages based on non-European character sets. In addition to the European character sets, Professional Edition supports:

- multibyte languages (Japanese, Korean, Chinese).
- bi-directional languages (Arabic, Hebrew and Farsi). Bi-directional languages as generally written from right to left, numbers are written from left to right. This is supported in 32-bit version only.

With Multilizer Professional you can create *single world-wide binary*. The standard edition makes it possible to create *single European wide binary*.

Ultimately, the support for languages depends of the OS support for language specific features.

3

Basic concepts

This chapter explains some key issues about software localization and globalization. For terms used in this chapter, cf. *Appendix A Glossary*.

Internationalization

In software internationalization the hard-coded country dependent information is removed. In practice this means, that no language specific information, currencies, dates, times etc. should be inside the program code.

Software internationalization is the first phase, which have to be done, in order to make the software apt for the target country. Depending of the type of the software and how it ahs been programmed, this phase may be very time consuming.

Multilizer™ is a tool for localization. However, the *dictionary-translator architecture* and the components make it possible to save a lot of time in this phase, if Multilizer is used from the beginning of the software development. Using the Multilizer components, country specific items are never hard-coded. This feature saves you a lot of time.

Localization

After the software internationalization, country specific items have to be added. The features to be added are called locale data. In addition the software has to be prepared to support the target country's character set and language.

Localized program

Localization means converting a program to a local market. The simplest case is to convert all the strings to the local language. In most cases you also have to make slight modification to the program to meet local standards and culture. A localized program can only support one language.

For example the native language might be English. If you decide to localize the program to German, French, and Spanish you will have four different versions of the program - each supporting one language.

Single worldwide source

Single worldwide source means that you have only one version of your source code. You either compile or include new resources for each localized version.

Traditional localization techniques use localization and single worldwide sources. The methods for making the internationalization and localization may vary a lot. The time spent on these phases may become the most decisive factor in software delivery dates, as well. Traditional localization techniques may take several months to complete and during it, the source should be freezed in the current version.

With Multilizer, you can continue developing the software even when localization takes place.

Single worldwide binary

A multilingual application takes localization one step further, in that a single-file application supports multiple languages, including the capability of switching languages at run-time. Having only one set of source files to maintain, obviously makes the development process more cost-effective.

Single worldwide binary means that there is only one version of your program. This version can handle all of the world's languages.

Innoview Multilizer makes it possible to create a single worldwide binary.

4

Coping with different languages

One main task in software localization is to make the software work in the target country's language and character set.

Originally, first computers were designed to work with character sets including only those characters needed in English. When computers became commercially available everywhere in the world, there arose a need for including target country's characters into the character set as well.

To be able to handle a specific language in the computer, it must be possible to handle its character set. There must be methods for inputting, storing and outputting characters. Thus, localizing applications involves

- processing of character sets and
- accommodation of the application's I/O methods for current language.

Following chapters introduce the different types of scripts to the reader. Later, most important items of these will be discussed in the context of today's information technology.

Character sets

All languages in the world can be divided into three basic groups, depending on the type of character set they use. The groups are:

- Single byte character sets.
- Multi or double byte character sets.
- Bi-directional character sets.

This division is rather a technical one, but as a matter of fact, it reflects three major cultures as well. The groups mentioned above could be rewritten in the following way:

- European character sets.
- Far Eastern character sets.
- Middle Eastern character sets.

Single byte character sets

Single byte character sets (SBCS), sometimes called single byte left to right, contain a maximum of 256 characters. Each character is stored in one byte. The text is written from left to right. Latin (e.g. English, German, French, Spanish, Finnish, Swedish), Greek and Cyrillic (e.g. Russian, Ukrainian) character sets are single byte.

These character sets evolved in Europe, and they started from the old Hellenistic culture. It is very possible, that the scripts came into Europe from Mesopotamia over Near East, though.

Greek

The ancient Greek character set was derived from Linear B, which was similar in structure to the Japanese. Later, due to the Dorian invasions, a script based on North Semitic model was adapted. As modern Middle Eastern scripts, this was written from right to left.

Later, such inventions as the vowels a, e, i, o, u made it the most successful and the most practically useful of the world's scripts on that time.

Modern Greek script has undergone only a few changes since the classical Greek period. Most changes are phonological.

Latin

Old Greek script expanded over Southern Italia and came into Roman hands. The alphabet was simplified. Through the influence of the Roman Empire, Latin alphabet came into use the whole Western civilization. On the base of latin script, the modern European character sets are based.

Thus, these alphabets are very denominative for European culture(s) and for those countries, where the Europeans established their colonies. The strong cultural expansion from the XVth Century on has exported – especially Latin alphabet – all over the world. In South and North America, Africa and Australia, the Latin alphabet is almost the only one in use.

All of these character sets have implemented accentuation marks to denote special sounds, i.e, the characters are based on a base character, and the phonetical difference is marked with an accent. In addition, some additional characters have been taken in use.

E.g, in German, 'ß' is used to denote 'ss', when it is in a certain place. In addition, in African languages characters like |, ||, ‡, ≠ are used to mark non-pulmonic sounds (clicks) not used in European languages.

From the Middle Ages some ligatures survived: French œ, Danish æ. The most known is however the ampersand '&', a ligature of the Latin word 'et' (*and*).

In addition, languages using different character sets and scripts are often transliterated in literature and school books into latin character set. For transliteration additional diacritics are attached to base characters.

Multi or double byte character sets

Multi byte character sets (MBCS), sometimes called double byte (DBCS) or Far East, contain more than 256 characters or idioms. Each character is stored either in one byte or two bytes. The text is written mainly from left to right top to bottom but sometimes from top to bottom left to right. Chinese, Japanese and Korean character sets are multi byte.



Only the professional edition of Multilizer supports multi byte languages.

Bi-directional character sets

Bi-directional character set (BiDi), sometimes called single byte bi-directional, contain a maximum of 256 characters. Each character is stored in one byte. The text is written mainly from right to left but sometimes from left to right. Arabic and Hebrew character sets are bi-directional.

The ancient Arabic script was a derivative of the Nabataean consonantal script, which was used two thousand years ago. Later, it was influenced by Mesopotamian Kufic scripts. From the eleventh century onwards, a flowing cursive style was developed, and it became the Arabic script to be commonly used. This script underlies most contemporary type-fonts.

Arabic script is used for a number of important languages: Persian, Urdu, Pashto, Baluchi, Kurdish, Lahnda, Kashmiri, Sindhi and Uighur.

Since the phonological inventories between these languages may differ a lot from that of Arabic, the script has had to be augmented and adapted to meet the new demands made upon it. In some languages, as e.g. Sindhi, certain Arabic letters are adapted to denote multiple sounds. On the other hand, in some languages there are a lot of redundant letters – in Persian there are four Arabic letters pronounced exactly in the same way.

Arabic script has been used in many other languages. However, many of them have abandoned Arabic script for Latin. Among them, there are languages like Indonesian (Malay), Hausa, Somali, Sundanese, Swahili and Turkish. Partially this is explained by the

last 400 years' aggressive expansion of the European cultures. In addition Turks wanted to modernize the country in the beginning of the 20th century and they took the Latin alphabet in use. Several Caucasian languages, e.g. Chechen, Kabardian, Lak, Avar, Lezgi used for a while latin alphabet. Due to the expansion of Russian power, cyrillic alphabet was implemented.

Arabic script is used nowadays on a widespread area from the Atlantic coast in Morocco to India. It is very probable, that these areas remain as users of this script. This area has been politically quite unstable, but an econimcal growth is to be expected, thus bringing the information technology to these countries. This will open a need for supporting Arabic script in the software.



The 32-bit professional edition of Multilizer supports bi-directional languages.

Text Input

Western languages

Western alphabets are based on writing characters. There is normally a very restricted number of characters by which all the words in a western language can be formed. Therefore most characters can be inputted by one character stroke in the keyboard.

Different Western languages use keyboard layouts that differ slightly from each other. This is due to the input of some language specific characters (Cf. *previous chapter*).

Multilizer Language Manager is able to change the keyboard layout to match the current language being edited.

Far Eastern languages

Far Eastern languages need a specific way for inputting ideographs. There might be thousand of characters, which should be inputted. For that reason, Far Eastern Windows language Editions whip with Input Method Editors (IME).

Using the IME, users can compose each character in one of several ways: by radical, by phonetic representation or by the character's numeric code-page index.

Multilizer Language Manager uses the IME provided by Far Eastern Windows language Editions for inputting those languages.

Middle Eastern languages

Middle Eastern languages have a restricted number of characters in their alphabets, like Western languages. The problem of inputting characters in these languages lies in the following reasons.

- text is written from right to left, numbers from left to right

- Arabic characters obtain different forms depending on where they are located in the word (initial character, in the middle of the word, final character or isolated character).

Arabic language editions of Windows are needed for inputting Middle Eastern characters in Language Manager.



For more information on editing a specific language in Language Manager, cf. Language Manager documentation.

5

Country specific items

Besides of just making the software work in different languages, it must be finetuned to work with country specific standards, as well. Country specific standards are later referred as locale data.

In fact locale information tells, what language is spoken in the locale specified. Therefore the language can be considered as a subset of the locale.



In Language Manager a specific locale is referred using the language and the country's name in parenthesis. Thus, selecting a locale like e.g, **French (Canada)**, means that the country is Canada and the language is French.

6

More about localization

The following list contains various sources where to get more information about localization:

- Developing International Software for Windows 95 and Windows NT, Nadine Kano, Microsoft Press, 1995. This is an excellent book about developing international software.
- WIN32 API documentation

7

How does Multilizer work?

Innoview Multilizer is based on translation-dictionary mechanism. There is a Dictionary containing the translations and a translator, which translates the program strings on the fly.

Multilizer dictionary-based translation solution keeps the language maintenance separate from programming, which makes big projects easier to maintain. In addition, the developer can make the decision, where to store the translation data used by the software: He can choose between standard text files, Unicode, database table, dictionary server, binary file etc.

The Dictionary is edited and maintained with Language Manager. The Dictionary is implemented in the programming phase as a Dictionary component. Translator component does the actual translation. For any multilingual program you develop, you have to be familiar with:

- **Language Manager**
- **Dictionary component(s)**
- **Translator component(s)**

Following chapter describes in brief the issues mentioned above.

Multilizer does not translate your words.

It is important to understand that Multilizer is not a tool that automatically translates words or phrases into another language. Instead it provides a mechanism that uses precompiled translator tables, dictionaries, to make the program multilingual. It is your (or your translator's) job to translate the strings of your program. Language Manager makes this translation as easy as possible.

Dictionary

The underlying concept behind Multilizer is to iterate each child component of the form, and translate their string type properties into another language. The translation is made by finding the corresponding string from the lookup table and replacing that string with the new one.

Language Manager

The lookup table mentioned above is called the translation table, and contains strings for each language that the application uses. In addition, the dictionary contains a language table and a locale table.

Language Manager is used for creating and maintenance of all the tables in the dictionary. It is a Windows utility, which makes the dictionary maintenance as comfortable as possible. In addition you are able to maintain all the locale-specific information apart from your software. This makes the development work both clearer and the work can be shared among several persons.

For additional information of Language Manager, cf. *Language Manager specific documentation*.

Language table

In addition to the translation table, Multilizer (and the Operating System) requires information about the current language. To provide this information, each dictionary contains another table called the language table, which contains information about the languages in the dictionary.

For every column in the translation table there must be a row in the language table (in the same order). The first row of the language table is the first column of the translation table and so on. The language table contains the native name of the language and the name in English. The table also contains the primary language id and the sub ids and the default font of the language.

Locale table

Simply switching between languages is not always the only requirement of a multilingual program. The program may have to adapt to some local customs and standards such as date and time formats, currency and measurement system.

To provide this facility, Multilizer uses locale data. Depending on the OS, Multilizer uses either the locale OS locale information or, if it alternatively uses a third table called the locale table. The locale table is generated with Language Manager's *New Project Wizard*.

With Multilizer you ensure that your localized program follow the locale support provided by the OS. This ensures the maximal compatibility. In addition, you are able to specify locales not supported by the OS, thus giving you the ability to localize your software for countries/locales not supported by the OS.

Depending on the target countries and the platform of your software, you may need the locale table or not.

Dictionary components

Once you have created a dictionary with Language Manager, you have to attach the information in it into the software. This is done by using a dictionary component.

There are different kind of dictionary components, each one of them implementing the data in the dictionary in different way into the software. By using an appropriate dictionary component, you can implement the dictionary as an external text file, binary file, data base table, etc.

Different implementations of dictionary gives you the freedom to choose the best possible way for your project. Thus, you are able to optimize your localized program according to the characteristics of your project.

For comprehensive information on the dictionary components, cf. Multilizer on-line help.

Translator

The translator is the part of Multilizer architecture, which does the translations. It uses the dictionaries for making the application either multilingual or adds support for a specific locale.

The translator is implemented as a component in Multilizer. It iterates each child component of the form, and translates the string type properties into another language. Setting the Targets and Restrictions properties you can specify, which strings to translate in your application.

The translation is made by finding the corresponding string from the lookup table and replacing that string with the new one. This is made by assigning a Dictionary component to the Translator component.

Together the Dictionary components and Translator components work as the powerful core for your multilingual or localized applications. Using them, you easily get started with localization. Though, the used architecture provides scalability and flexibility to the software development.

Concerns in building a dictionary

As mentioned in the previous chapter, the translator components translate the targets on run time. The language used in software development is called native language. When running the localized/multilingual program in English, the translator components translate from Native to English. When run in French, Native is translated into French.

For a software developer, using English as native language, at first it sounds strange that Multilizer translates from English 'Native' to English. However, in order to make the architecture and functionality of Multilizer as simple as possible, the same translation is applied for every target language.

Furthermore, in localizing software there are certain issues, which make the use of 'Native' language very convenient. Next chapter explains one reason for using 'Native' language.

Coping with words with several meanings

A problem arises when the same native word is used to convey two different meanings, depending on the context in which it is placed. For example, the word 'firm' can mean a 'company', or 'hard'. In other languages, separate words may be used to convey the two meanings, but because the dictionary index key must be unique it is not possible to specify the same word twice.

| <u>English</u> | Finnish | Swedish |
|----------------|---------|---------|
| ... | | |
| firm | yritys | företag |
| firm | luja | stark |
| ... | | |

This is not possible in the database dictionary (remember that the native column must have a primary index). In the text dictionary the syntax is legal but the second 'firm' entry is never used.

There are two solutions for ambiguous words. Either use a fake native language, or insert a space characters after the word.

Using a fake language as the native language

With this technique, the native language is a fake one. For example it can be English whenever an unambiguous word is used but in the case of an ambiguous word the English word must be altered slightly to make it unique.

| <u>Fake</u> | English | Finnish | Swedish |
|-------------|---------|---------|---------|
| ... | | | |
| firm | firm | yritys | företag |
| firm1 | firm | luja | stark |
| ... | | | |

The first 'firm' is unchanged, but the second 'firm' is actually 'firm1'. Use 'firm1' in the source of the application whenever you want to refer to firm meaning hard, solid, etc.

Use LANG_FAKE as the primary language id of the row containing the fake language in the language table of the dictionary.

We recommend the use of this technique in every dictionary.

Using space characters

Another solution is add space character(s) to the native word or sentence. This extra space makes the distinction between the two meanings of the word.

| <u>English</u> | Finnish | Swedish |
|----------------|---------|---------|
| ... | | |
| "firm" | yritys | företag |
| "firm " | luja | stark |
| ... | | |

The later 'firm' contains one space as the last character.

We do not recommend this technique except in the cases that the size of the dictionary is explicitly the key issue.



Language Manager makes it easy to find multiple appearances of Native language strings from the source code: pointing on the left side of the you see map info, i.e., the information where the strings are located in the source code. This helps in finding the ambiguous words in the program source.

Native language concerns

Native language is the key field in the translation table. Whenever an item is translated in the program, there is performed a search in the translation table. The search mechanism is optimized for maximal speed, which disallows certain characters in the Native language.

If the localized software works on one code page only, then there is no restrictions in Native language's content. Otherwise, the Native language should use standard 7-bit Ascii characters only: they are located in the same place on every code page.



Language Manager marks with red colour those Native language strings, which contain characters outside the allowed ones.

Appendix A Glossary

This glossary describes common terminology in software localization.

Code pages

Code page is a code array that maps the integer code to the character of the character set. The first 128 items of every code page contains the ASCII characters. The remaining items depend on the character set.

Windows 95 and Windows NT support multiple code pages but only one is active each time. This active code page is bound on the language version of your operation system. It can not be changed.

Globalization

Globalization is the compound of tools and methods, which are applied to software, in order to make it work globally. Thus, the globalized software works with the appropriate features of each of the target countries.

Globalization can be seen as the sum of internationalization and localization: when globalizaing software, it first has to be internationalized and after that localized. A properly developed software targeted for many locales must go through both the internationalization and localization.

Internationalization

Internationalization is the first phase in making software for global markets. Software internationalization means the act of removing country and locale specific features hardcoded in the software's source code.

However, internationalization tends to be used in the meaning of globalize software.

Language id's

Primary language id

Primary language id specifies the language (e.g. English, Finnish, Japanese, etc.).

Multilizer uses standard Win32 NLSAPI primary language ids (LANG_XXX). The constants can be found from the windows.pas (32 bit) or `ivmulti.pas` (16 bit) file.

Sub language id

Sub language id specifies the country of the language (e.g. English in United States, English in Great Britain, English in Canada, etc.).

Multilizer uses standard Win32 NLSAPI sub language ids (SUBLANG_XXX). The constants can be found from the windows.pas (32 bit) or `ivmulti.pas` (16 bit) file.

Locales

Locale is a combination of primary (language) and sub language (country) ids. In other word it is a language spoken in a country (e.g. German spoken in Austria). There can be multiple locales for one language. For example, English (United Kingdom), English (United States), English (Canada), etc.

Localization

Localization is the act of applying country specific features into the program. The name is derivative of locale, which is a OS-specified set of items of the target country's denominative features.

Software localization aims to make the software reflect the target country's cultural features, in order to make the software customer-friendly.

Some countries need several localized versions of the software. For instance in Canada, there is a need to produce both English and French locale versions of the software.