ORACLE White Paper

Building Multilingual Applications with Developer/2000

July 1997

Building Multilingual Applications with Developer/2000

July 1997

Author: Nick Zaveri

Building Multilingual Applications With Developer/2000

TABLE OF CONTENTS

Translation Builder Release 3.0

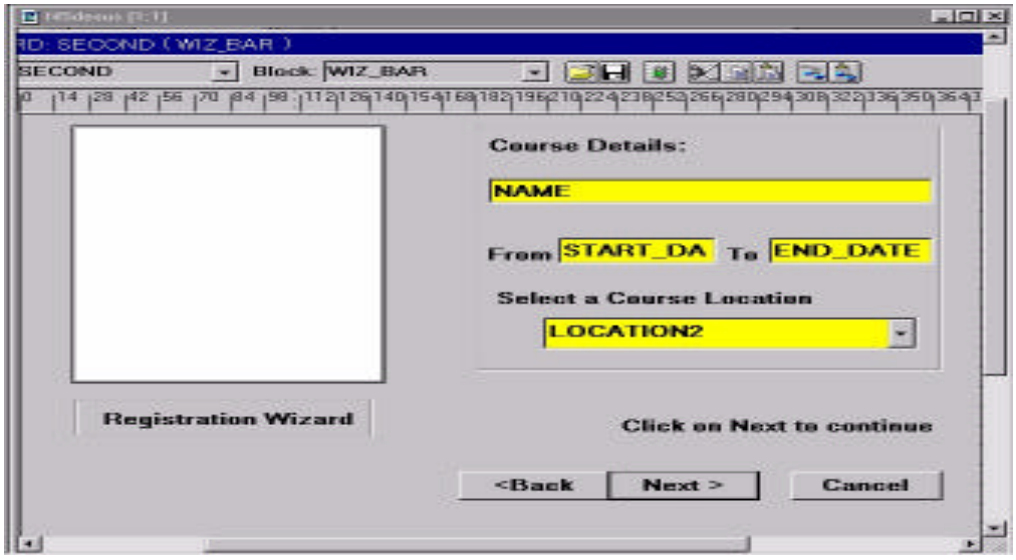
Synopsis

Overview

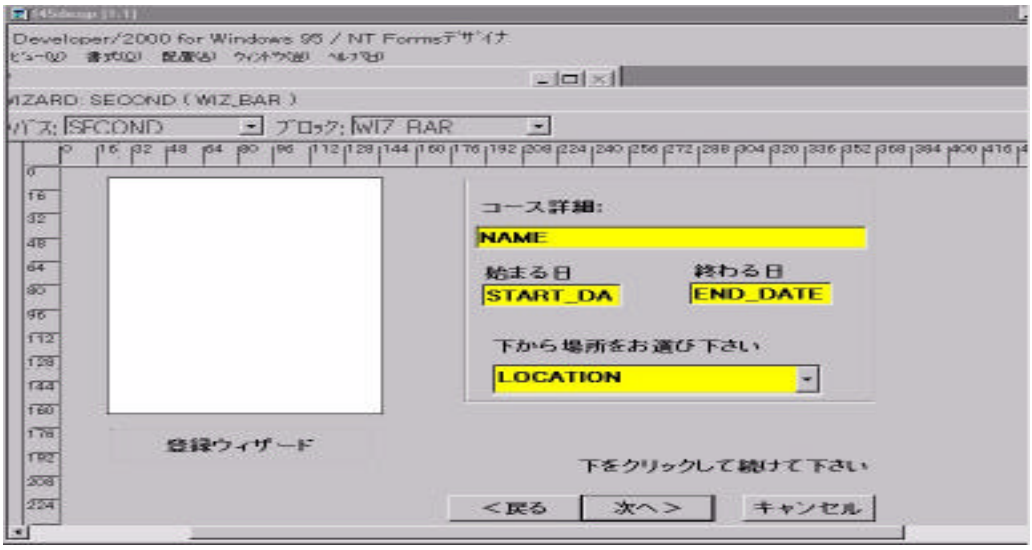
This document describes the functional specifications for the Developer /2000 - Translation Builder Version 3.0. Translation Builder is a powerful and user friendly tool which supports and manages translations of text extracted from Oracle and Non-Oracle resources.

The target user of Translation Builder is an Oracle's customers who has a need to localize applications developed using one of Oracle's application development tools such as Developer/2000.

The primary goal of Translation Builder 3.0 is to support and manage translations of text extracted from Developer / 2000 applications. The tool will also support translations of Non-Oracle resources files such as Microsoft Windows (.rc) and HTML files. Translation Builder 3.0 provides a user friendly and cost effective solution to translate a given application and will help the customers to make a simultaneous release of native and multilingual applications.



Native Application



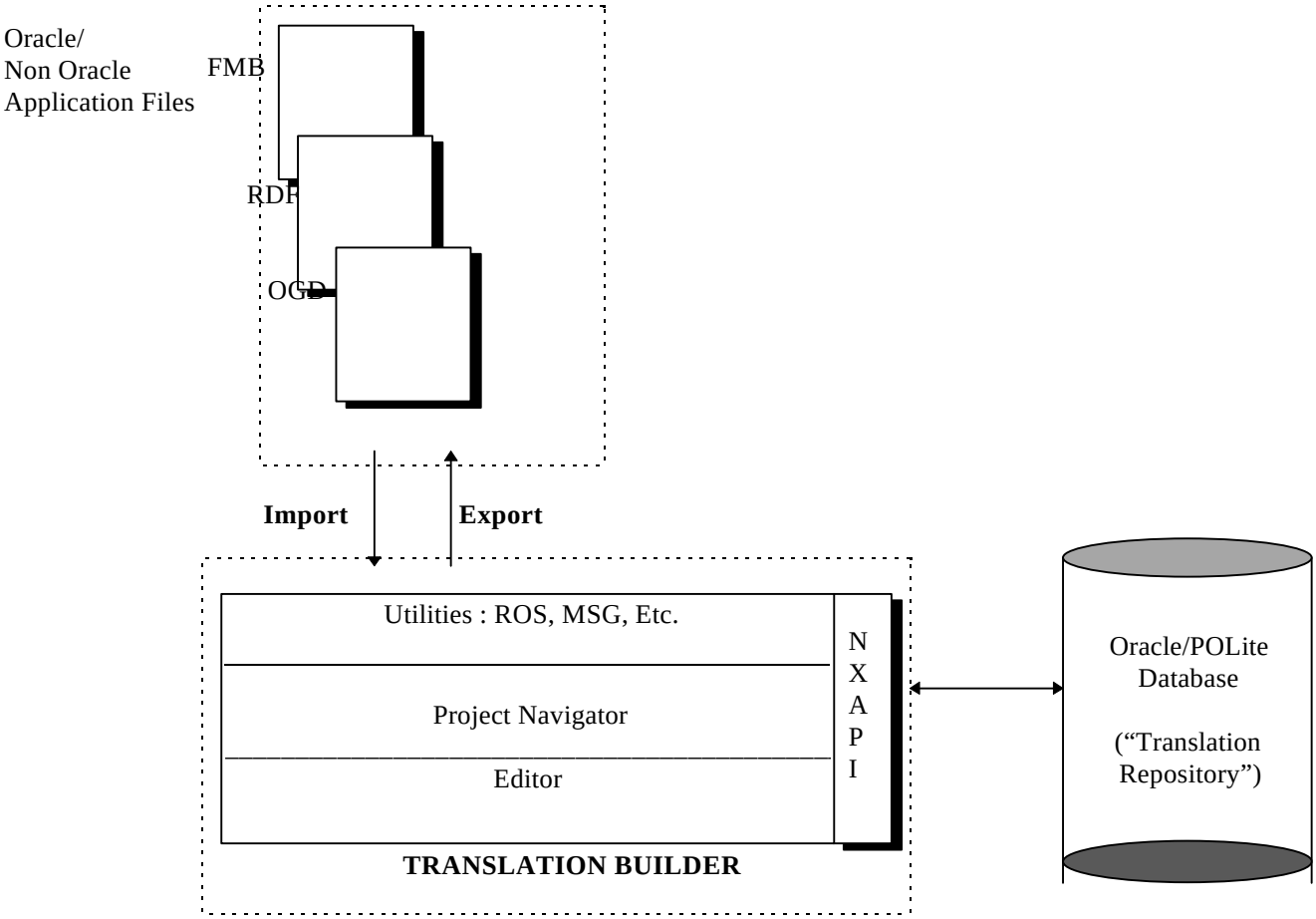
Localized Application Using Translation Builder

Requirements

The architecture of Translation Builder 3.0 is based on following requirements:

1. Eliminate dependency of the Database character set.
2. Use a single repository mechanism for Translation Management.
3. Allow users to translate a base module into multiple languages.
4. Provide a single user interface for any resource file being translated.
5. Provide preview capability to check the appearance of translated data.
6. Move translated text to and from the application resources easily.
7. Support glossaries and management of earlier translation to help the translator expedite the translation process by suggesting partial or complete translations.

Product Architecture



As shown in the figure above, strings are extracted from resource files and imported to the translation repository by utilities (one for each resource type). The project navigator manages the overall process and ensures integrity of the data. From the project navigator, the user launches the editor to translate a given set of strings, adding the translation into the repository. Once translation is complete, the strings are extracted from the repository and exported to the resources. If the resource type supports multiple languages, the translations are added back to the original resource. Otherwise, a copy of the resource is created, but with the base strings replaced by the translations.

The most important aspect of the Architecture are as follows:

1. The user interface is independent of the Database character set. For example, if a user needs to translate a given module from English to Chinese, it is not required that the data base support the Chinese character set.

2. Modules can be translated from any language to any language. Once a given language translation is complete, that translation can be used as a base for further translations. For example, consider a module in English that needs to be translated to both Traditional and Simplified Chinese. Once the module has been translated to the first dialect, the subsequent translation will be much quicker (and less expensive) than having to retranslate from English.

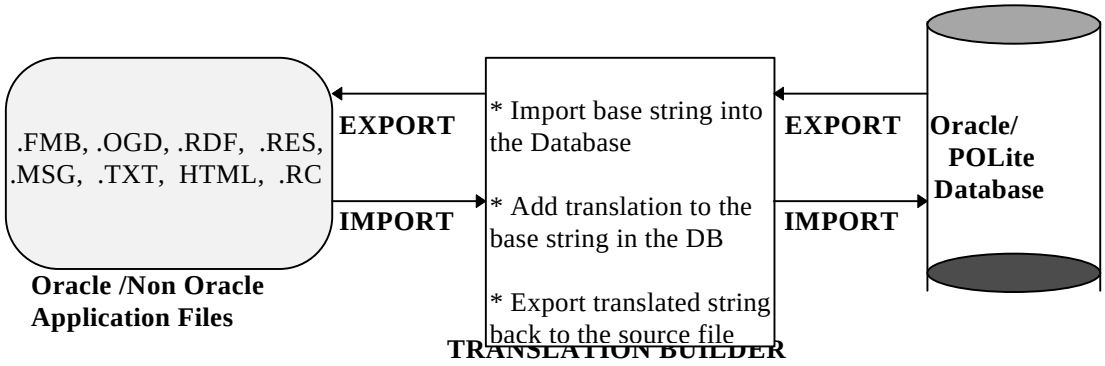
3. Users can change fonts to any language for display purpose, independent of the local operating system, if the fonts are available.

4. Translation Manager is available on the Microsoft Windows 95 and NT platforms. The Oracle RDBMS is available on multiple platforms including Microsoft Windows and UNIX. This allows operation of Translation Builder in both client/server and stand alone configurations. The translations themselves are portable across multiple platforms - you can use Translation Builder on Windows 95 to translate resources or messages for UNIX.

5. Translation Builder can use Personal Oracle Lite ("POLite) as a local repository, as well as remote Oracle 7 repository. The POLite repository can contain entire translation projects, or as a single module to be translated. The POLite files can be compressed, sent via electronic mail or downloaded from the web. The Project Navigator provides simple drag and drop transfer of modules to and from POLite database. This facilitates remote translation remote translation, while still ensuring integrity of the data being translated, since it stored in a database, not a flat file format.

Translation Process

The following diagram provides a top level view of the main tasks associated with translating a given module.



From the diagram above, the steps to translate a module are:

1. Import String: Translatable strings are extracted from Oracle or Non-Oracle resources and imported into the Translation Repository.

2. Add Translation: "Adding a translation" is the process of indicating which language(s) a given module will be translated to. When this is done, users have the option of using available glossaries and prior versions of the module ("Translation Memory") to pre-translate any exact matches found in the repository. While adding the translation, users can check the " Use Attached Glossaries" box to replace the base string with an exact match of the targeted language.

3. Translate: The project navigator presents and intuitive explorer-style view of available projects and modules. The translation editor is a side-by view of base and translated strings, and includes on-the-fly glossary lookup for the

string being translated. The translation made here become part of the Translation Memory and are available for use in future translations.

Translation Editor Window

4. Preview: Perform appearance checking and tune if necessary to preserve acceptable look and feel of the translated application.

5. Export String: The translated strings are extracted from the repository and merged back to the original resource (or a copy of it) by the export utility.

Translation Management

Overview

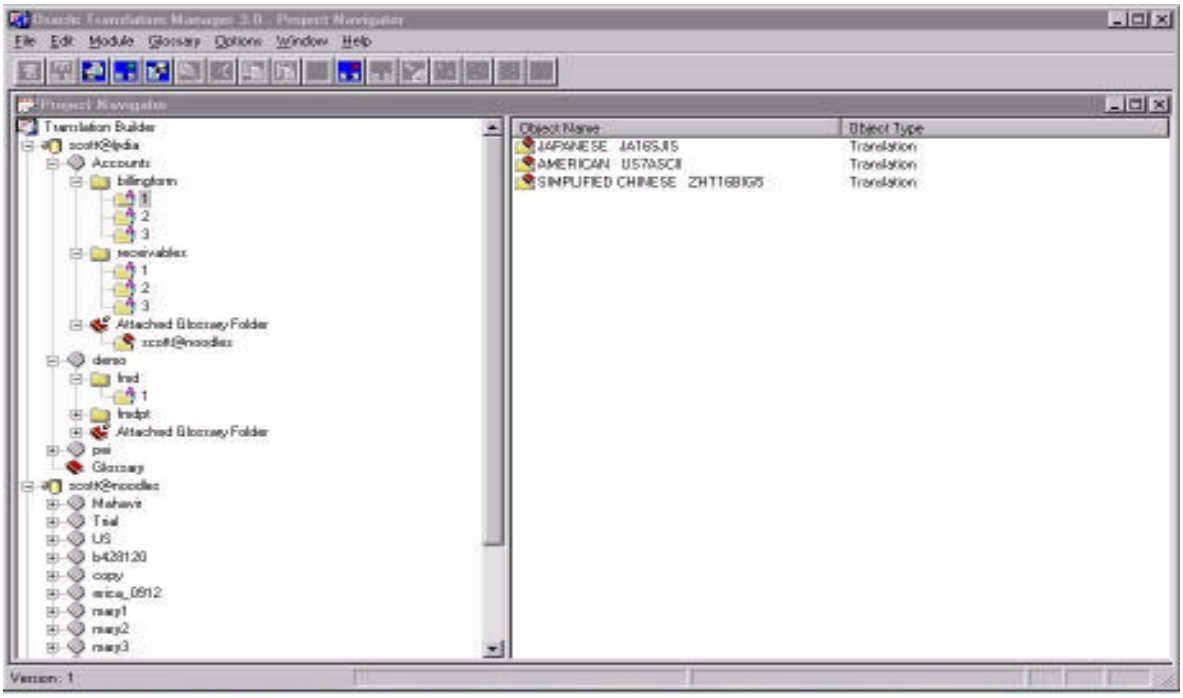
1. Translation Manager organizes projects and glossaries, manages base and translation modules/multiple versions and allows reuse of earlier translations.

2. Extraction and Insertion utilities (“filter”) are associated with different file types, and are provided for all Oracle Developer/2000 file types and some common industry types.

3. Supports bulk loading and pre-translation from a command line interface..

4. Maintain translation integrity at all times by using either a remote Oracle7 repository, or a local Personal Oracle Lite data file.

Project, Module, Version, and Language Relationship



Project Navigator

Translation of user interface text often involves hundreds of files. Organizing and tracking these files is a tedious and laborious process, which can lead to costly errors such as misplaced files or incorrect translations. To make the task of organizing and tracking translation projects efficient, Translation Builder uses an Oracle database to provide powerful project management facilities and interfaces with Developer/2000's project builder component.

As shown in the figure above, users can create projects in the database (projects like accounts, demo etc.). Each project contains modules (billingforms, receivables) imported from the source. Users can identify specific version of each module. Each version can then be translated into multiple languages. Version 1 of billingform is translated into Japanese and Simplified Chinese from the native English Language. The Project Navigator will make sure that the above relationship remains undisturbed during the complete translation process. The translation process can be comprehensively monitored through stages of import, translation, update, review and export.

Translation Memory to Reuse Translations

Contract translators typically charge by the word, hence the ability to reuse prior translation provide a tremendous cost savings. Moreover reusing translations also enables faster delivery of localized applications.

One of the key features of Translation Builder is that previous translations are stored in the translation repository. The set of previous translations and glossaries is collectively known as "Translation Memory". Upgrades and changes to an application does not require that the translation process be restarted from the beginning. Each new translations adds to the translation memory, making future translations easier and quicker to accomplish. Version control of loaded text and translations is also provided by the translation memory.

Translation Builder can compare two different versions of application definition files and determine which strings have been changed and which have been added. If an earlier version of the resource has already been translated using Translation Builder, the tool will then use the previous translation as a starting point to search for matches. The tool will also carry out incremental search and will search the entire translation memory to find an exact match if necessary.

Languages Supported by Translation Builder

Arabic	Danish	Greek	Lithuanian	Slovak
Bengali	Egyptian	Hebrew	Malay	Slovenian
Brazilian Portuguese	English	Hungarian	Mexican Spanish	Spanish
Bulgarian	Estonian	Icelandic	Norwegian	Swedish
Canadian French	Finnish	Italian	Polish	Thai
Catalan	French	Japanese	Portuguese	Traditional Chinese
Croatian	German	Korean	Romanian	Turkish
Czech	German Din	Latvian	Simplified Chinese	Vietnamese

Integration with Developer/2000

In future releases, Translation Builder will be more tightly integrated with Developer/2000 in the following ways:

- Users will be able to invoke Developer/2000 designer components as an advance previewers to perform appearance checking and adjustments.

- Developer/2000 will be able to use Translation Builder's glossary support to perform or revise translations without invoking the translation manager.
- Users will be able to translate strings inline within the Developer/2000 Development Environment. These translations will be preserved when the module is imported to the translation.

Long Term Road Map

Longer term plans may include the following, depending on the users feedback:

Bundled Glossaries

- For a variety languages.
- For a variety of vertical and horizontal application disciplines (Science, Medicine, Engineering, Manufacturing and Human Resources, etc.)

Additional Filters

- Oracle will include filters for document types based on demand, for example .pdf and Oracle Power Objects format.
- API's for 3rd parties to add filters for their types.

Web Enablement

- Java front end to a remote Translation Builder engine and repository. This allows remote work by translators equipped only with a web browser, while preserving central management of entire translation process.
- " On-the-fly" Translation Cartridge for the NCA. This would allow a document such as an HTML page to be translated in against a pre-created translation memory before presentation to the users. This would remove the need for multi-lingual data or knowledge in web server applications, or the need to build multiple, translated copies of the same page.
- Web enabled automated Translation services "Send your file and we'll translate it based on our glossaries"

Show Context of Strings Within Resources

One of the most significant challenges in translating a user interface outside a GUI builder is knowing the context in which a string is used. Understanding the complexity of the relationship of menu option descriptions or the

structures of screen displays, defined in an application being translated, is critical as this determines what the actual translation will be. While the preview capability will assist in properly placing and sizing strings, having to constantly switch back and forth between preview and the Translation Builder editor, reduces translation efficiency. Translation Builder has a capability to record context information for strings imported to the repository. In future release the Translation Builder filter utilities will be enhanced to capture context information from the resource files. Not only will translators work more efficiently knowing that a given string is a prompt, as opposed to an error message or a menu item, but automatic reuse of translation memory will become more effective, giving greater weight to strings of matching types.

Conclusion

In today's world markets, success depends on the ability to communicate quickly and efficiently in multiple languages. Oracle Translation Manager enhances the ability to communicate to the world by reducing the time, cost and efforts to produce high quality translated applications and simultaneously release native and localized applications.