

Migrating from Oracle Translation Builder to Oracle9i Forms and Oracle9i Reports TranslationHub

An Oracle White Paper
September 2002

Migrating from Oracle Translation Builder to Oracle9i Forms and Oracle9i Reports TranslationHub

Introduction.....	3
Introduction to TranslationHub.....	4
Projects in TranslationHub.....	5
The migration path to TranslationHub	6
A few words about TranslationHub and databases	7
Before you start	7
UI-based migration	8
Script-driven migration.....	9
TranslationHub command-line executable	9
TranslationHub list of commands	10
New Project.....	10
OpenProject	11
Update	12
Merge.....	13
Rename Project.....	13
Delete Project.....	14
Import	14
Migration Example	16
Capturing translation strings stored in Oracle Reports	16
Converting Reports using the command line.....	17
Converting Reports using the graphical UI.....	18
Sample migration	18
Capturing translation strings stored in Forms	21
Sample migration script	21
Migration results.....	23
Testing the migration success.....	25
Capturing Strings from Forms 4.5 modules	26
FAQ.....	31

Migrating from Oracle Translation Builder to Oracle9i Forms and Oracle9i Reports TranslationHub

For an overview of TranslationHub, see the [Oracle9i Forms - TranslationHub Overview](#) whitepaper

INTRODUCTION

TranslationHub is the new translation tool for Forms Developer and Reports Developer* modules in Oracle9i Developer Suite (Oracle9iDS) Release 2. TranslationHub is not a standalone product but a subcomponent of Oracle9i Forms and Reports.

Before TranslationHub there was Oracle Translation Manager (OTM) and Oracle Translation Builder (OTB) for building multilanguage Forms and Reports applications. OTM and OTB are no longer shipped within Oracle 9iDS.

Oracle9i Forms and Reports modules are not supported for direct upgrade from release versions prior to 6i. If you have applications in versions earlier than 6i, you will need to bring them to a 6i level first, before continuing the upgrade to 9i. The TranslationHub base release is supported with Oracle9i Forms and Reports modules only. Forms6i and Reports6i support will be added in a future patch set.

Before starting, it is important to mention that Forms Developer, the Oracle9i Forms builder component, is contained in Oracle9iDS Release 2, while the Web runtime environment, Forms Services, is contained in the Oracle9i Application Server (Oracle9iAS) Release 2.

Similarly, Reports Developer, the Oracle9i Reports building environment, is contained in Oracle9iDS Release 2, while Reports Services is contained in the Oracle9iAS Release 2.

The intended audience for this paper is customers that currently use Oracle Translation Builder or Oracle Translation Manager to create multilanguage Forms and Reports applications. This whitepaper is not intended to be a tutorial on creating and deploying multilanguage Forms and Reports applications.

Whenever the term *migration* is used throughout this document, it is always to be seen in the context of Oracle Translation Builder and Oracle Translation Manager to TranslationHub migration. This document doesn't describe the upgrade path

* TranslationHub translates Oracle9i Reports paper layouts only. The new Oracle9i ReportsWeb layout is not supported in TranslationHub. Paper is the only layout option in Reports 6i and Reports 2.5. If you are migrating a Developer 6i-built application to Oracle9i Forms and Oracle9i Reports, your reports will use paper layouts.

for application modules of previous Forms and Reports releases to Oracle9i Forms and Oracle9i Reports.

Additionally, the word *migration* is used only to describe the process of capturing existing translation strings created with Oracle Translation Builder or Oracle Translation Manager and importing them into TranslationHub.

INTRODUCTION TO TRANSLATIONHUB

TranslationHub was derived from software used internally at Oracle to translate commercial products like Oracle Forms, Oracle Reports, the Oracle database, and Oracle Applications 11i into more than 24 languages.

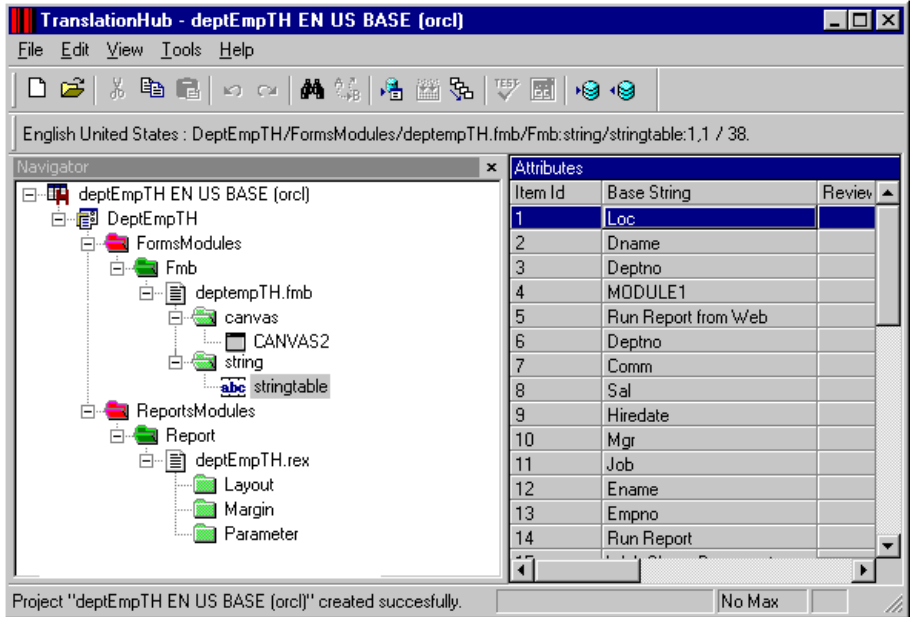


Figure 1: Oracle TranslationHub hierarchical structure

The hierarchical tree structure in TranslationHub provides more detailed information about the application modules and their translatable strings than Oracle Translation Builder did.

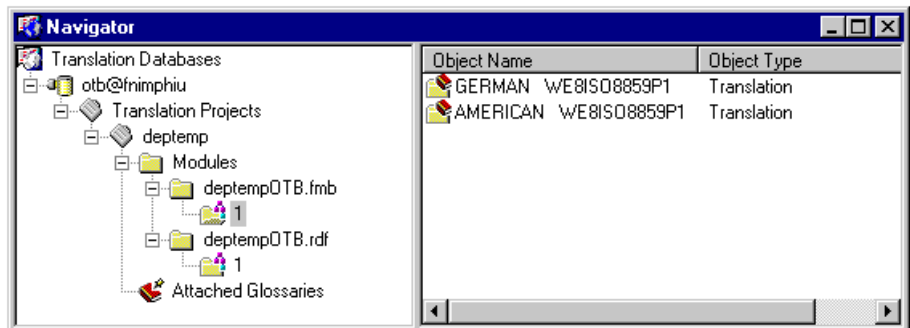


Figure 2: Oracle Translation Builder hierarchical structure

Unlike Oracle Translation Builder, Oracle TranslationHub uses the concept of projects when working with translation strings. In TranslationHub you create one project for each application and language, where the development language project is referred to as the base project.

Related Forms and Reports application modules can be viewed in one project, allowing you to check translation consistence across modules. As with Oracle Translation Builder, the translation strings in TranslationHub are saved in an Oracle database.

Projects in TranslationHub

Translations in TranslationHub are stored in projects. A project can be either a base project, containing the language strings of the base development modules for a Forms and Reports application, or a translation project, containing the translated strings for an application.

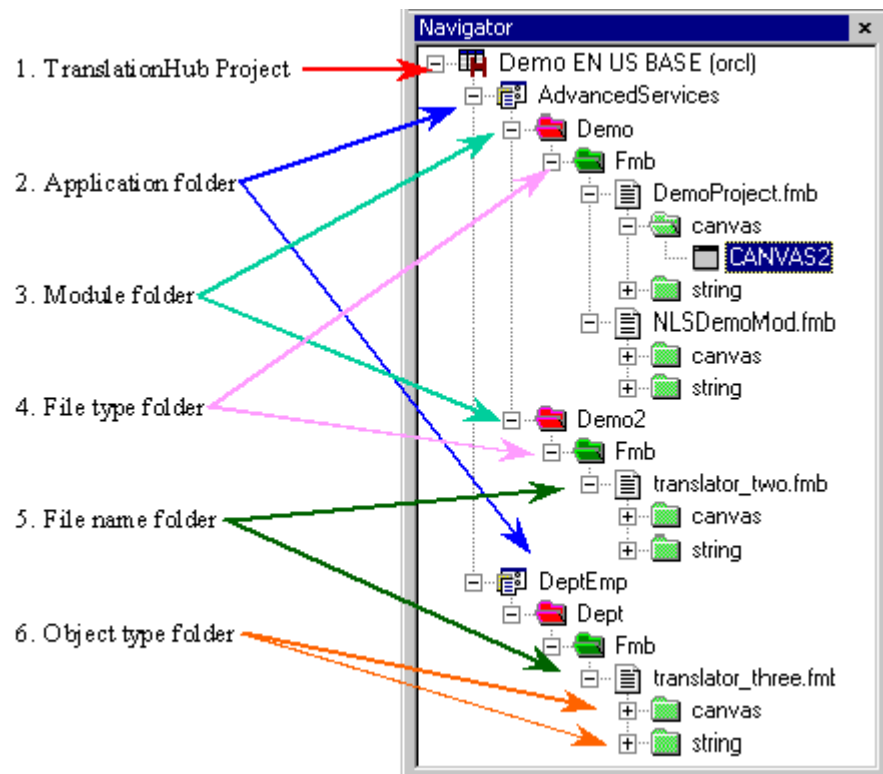


Figure 3: TranslationHub project folder structure

- **TranslationHub project** — A new translation in TranslationHub starts with a base translation project.

The project shown in the image above is a base project for English strings in an American territory. The name of the base translation project in this example is chosen as “Demo.”

The translation projects are derived from the base translation project. The name for a German translation project of the “Demo” base project shown in the image above would be “Demo DE DE TRANS (orcl).”

There is one important restriction to keep in mind when you name a TranslationHub project, which is that the name must be within the WE8ISO8859P1 character set. The recommendation, though, is to be even more conservative and to use names conforming to the US7ASCII standard.

- **Application folders** — An application folder is a logical container for translatable strings contained in Forms and Reports modules. Application folders are optional and can be used to organize translation strings. If you translate a Human Resource application, for instance, two possible application folders could be “Employment” and “Careers.”
- **Module folders** — Module folders are also optional and can be used to further organize translation strings within application folders. If there are logical application sections within the “Employment” application folder described above, these sections can be represented in TranslationHub using module folders.

TranslationHub can compare strings stored in two or more module folders, ensuring consistent translations.

- **File type folders** — There are three possible file type folders in TranslationHub: “Fmb”, “Mmb” and “Report”. Fmb type folders contain strings imported from Forms binary modules (fmb), Mmb type folders contain strings imported from Forms menu modules (mmb), and Report type folders contain strings imported from Reports rex file formats.
- **File name folders** — File name folders represent the names of the Forms and Reports application modules from which the translation strings are read. A file type folder can contain many file name folders.
- **Object type folders** — Object type folders contain the translatable strings. Object type folders are created for each Canvas in a Forms application and for the string table containing all those strings not displayed on a Canvas. For Reports modules, an object type folder is created for the Reports layout margin and header and for custom parameters.

If no names are specified for the application and module folders when an application module is imported, then default names will be assigned.

THE MIGRATION PATH TO TRANSLATIONHUB

To migrate from OTB to TranslationHub, you need to import strings from translated Forms or Reports modules into TranslationHub.

A few words about TranslationHub and databases

TranslationHub uses the Oracle database to store the language strings captured from the Forms and Reports application modules. TranslationHub is only supported with the Oracle8*i* and Oracle9*i* database releases.

TranslationHub uses the WE8ISO8859P1 character set. Problems will occur if your database uses another character set because characters are interpreted differently between character sets.

If your database does not use WE8ISO8859P1, you must create another database instance for the TranslationHub repository.

There is no specific database tuning required to work with the TranslationHub repository. For that reason, database tuning considerations remain beyond the scope of this paper.

Before you start

Unlike OTB and OTM, where all translation strings are exported back into the application module they were imported from, TranslationHub creates a new copy of the original application module for each translation language. This leads to an increase in the number of application modules, which requires some organizing:

- Create a language subdirectory under the directory containing the application fmb and rex files. If, for example, an application directory “CustomerOrder” exists, containing several Forms and Reports modules, and the application is translated into German, French, and Italian, then the following subdirectory structures should be created for TranslationHub: “CustomerOrder\de”, “CustomerOrder\fr” and “CustomerOrder\it”.
- If your application modules constantly change, it is recommended that you keep the base development module as a single point of truth. All changes would then go in this base module first, before translated versions are created.
- Make sure that all application dependency files (PLLs, for example, or Object Libraries) are accessible when importing Forms and Reports files into TranslationHub.

If you kept the absolute path when attaching libraries to your application modules, this path must be accessible to TranslationHub. If the library path was removed from the attachment, the library files need to be in the same directory as the Forms or Reports modules.

Plan on using a source control mechanism like Oracle SCM to track the state of individual Forms modules. Oracle SCM also helps you to determine the differences between any given module and earlier versions of that module. Before using TranslationHub to translate your Oracle9*i* Forms and Reports application modules, make sure that you have backed up your application files.

UI-based migration

User interface–driven migration is suitable for small applications with a small number of Forms and Reports modules. It is not that the UI cannot handle large applications, but the time required to do so may be prohibitive. The following steps need to be processed to capture OTB/OTM translated strings, stored in Forms and Reports modules, into TranslationHub:

For information on how to create rex file version of your Reports modules, refer to the Reports product documentation or to the Reports online help.

This topic also is briefly described later in this document.

1. If you have not yet converted Oracle Reports modules into the rex format, do this first.
2. Create a TranslationHub base project for the application files to capture. Define the project language as the language used when the Forms and Reports application was built.
3. Import Forms and Reports modules into the base project.
4. Create and update the translation projects with the strings from the base project. Use the “Tools | Update Projects” menu option to create and update multiple projects at the same time.

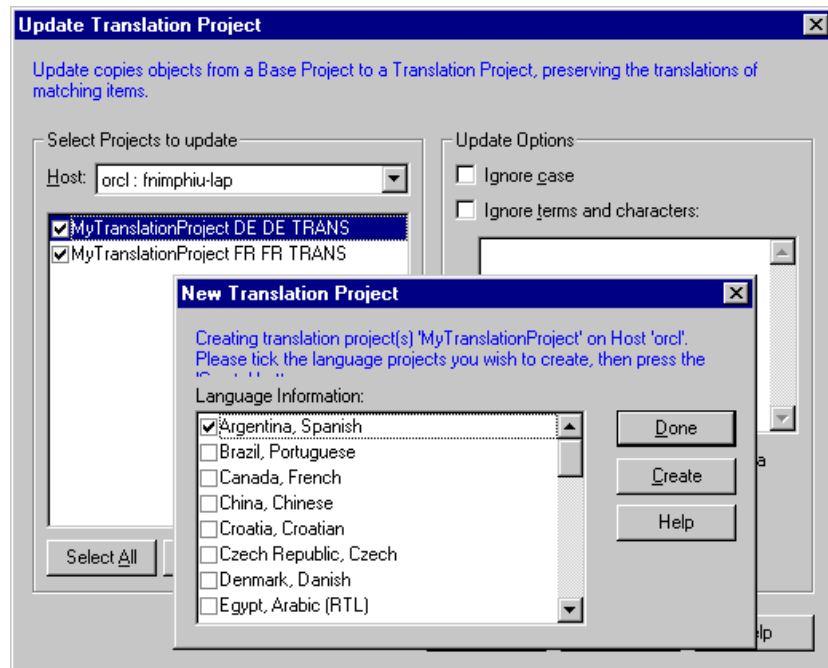


Figure 4: Create and update multiple translation projects with the strings contained in a base project

To capture existing translation strings from an application module, open the translation project created for this language and select the “Tools | File Import” menu option, which brings up the file import dialog.

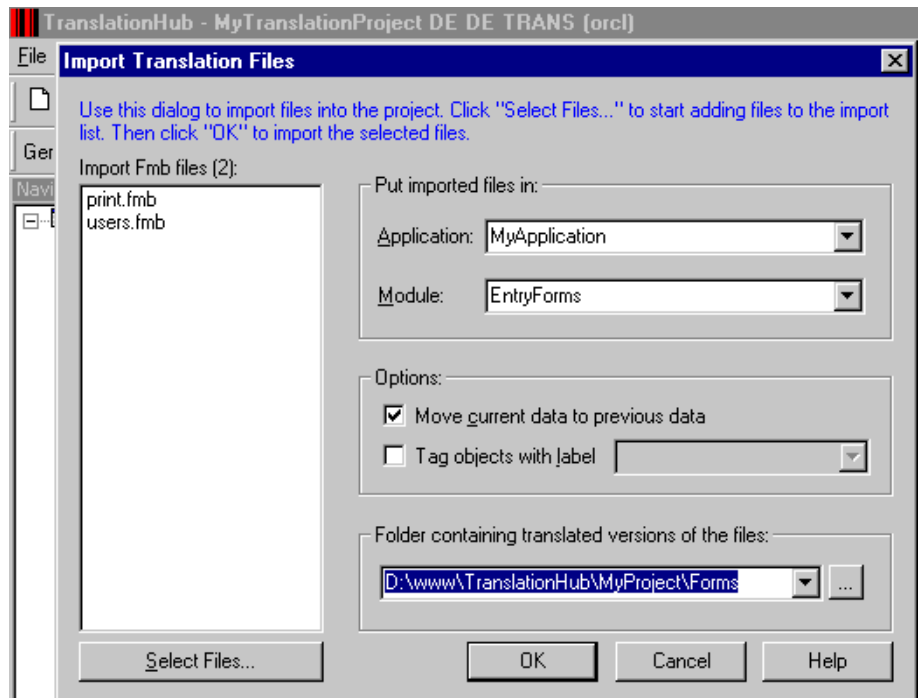


Figure 5: Import dialog of a translation project

Reports rex files contain only one language and thus one file needs to be created for each translation language and one for the base development language. To import translation strings from a Reports module you need to import both, the file containing the base development language and the file containing the translation strings for the current open translation project. Use the “Folder containing translated versions of the files” input field to reference the location of the translated Reports modules on the file system while selecting the base development language file using the “Select Files ...” button.

Script-driven migration

The TranslationHub command-line¹ executable is a powerful API that can perform all the migration steps exposed through the graphical user interface. Although the graphical user interface is handier, using the command line is much faster.

TranslationHub command-line executable

To start a migration performed by the command-line executable, navigate to the Oracle_Home\bin directory of the Oracle9i Forms installation. Type *otbatch* followed by the name and location of the batch script and press enter:

```
otbatch <script_filename>.txt
```

¹ The command-line interface is supported for migration only and does not yet expose the full set of commands

There are at least two batch files used within this migration: the file that is provided as an argument to the otbatch command, which contains the actual Translationhub commands, and the file containing the modules to perform this action against.

TranslationHub list of commands

The following command-line arguments are supported for migration:

- NewProject
- OpenProject
- Update
- Merge
- Delete Project
- Rename Project
- Import

To avoid unintended modifications, be aware that there is no rollback command for restoring data that has been changed or deleted accidentally.

New Project

Creates a new project in TranslationHub

Syntax

NewProject , *Project name, language, country, base language, base country, project type, host alias, overwrite*

Parameter	Description	Value
Project name	The name portion of the project, for example, in the project “Test EN US BASE alias”, ‘Test’ is the name portion.	String.
Language	The two-letter language identifier.	Should be a valid identifier as defined in the Hublang.ini file.
Country	The two-letter country identifier.	Should be a valid identifier as defined in the Hublang.ini file.
Base Language	The two-letter base language identifier.	For a base project this can be blank. For a translation project, it should be a valid identifier as

		defined in the Hublang.ini file.
Base Country	The two-letter base country identifier.	For a base project this can be blank. For a translation project, it should be a valid identifier as defined in the Hublang.ini file.
Project type	Indicates whether this is a base or project.	BASE = base project. TRANS = translation project.
Host alias	The alias name of an Oracle database host.	Should be a valid Oracle alias.
Overwrite	Indicates whether to overwrite the project if it exists.	0 = do not overwrite the project. 1 = overwrite the project.

OpenProject

Opens a project for importing files. Optionally the project can be created if it does not exist.

Syntax:

OpenProject , *Project name, language, country, project type, host alias, create, mode*

Parameter	Description	Value
Project name	The name portion of the project, for example, in the project “Test EN US BASE alias”, ‘Test’ is the name portion.	String.
Language	The two-letter language identifier.	Should be a valid identifier as defined in the Hublang.ini file.
Country	The two-letter country identifier.	Should be a valid identifier as defined in the Hublang.ini file.
Project type	Indicates whether this is a base or translation project.	BASE = base project. TRANS = translation project.
Host alias	The alias name of an Oracle database host.	Should be a valid Oracle database alias.
Create	Indicates whether to create the	0 = do not create the project.

	project if it does not exist.	1 = create the project.
Mode	Open the project in shared or exclusive mode.	0 = exclusive mode. 1 = shared mode.

Update

Updates a named project with the content of an open project.

Syntax

Update , *Project name, language, country, host alias, overwrite, ignore case, update from text, save output*

Parameter	Description	Value
project name	Name of project to be updated.	String.
language	The two-letter language identifier.	Should be a valid identifier as defined in the Hublang.ini file.
country	The two-letter country identifier.	Should be a valid identifier as defined in the Hublang.ini file.
host alias	The alias name of an Oracle database host.	Should be a valid Oracle database alias.
Overwrite	Overwrite existing items (do not make old).	0 = do not overwrite. 1 = overwrite.
ignore case	Ignore case of use text when inheriting translations.	0 = case sensitive. 1 = case insensitive.
update from text	Update translations from text (not just ID).	0 = do not update from text. 1 = update from text.
save output	Specifies whether the output from this operation, normally seen in the output window, is saved to the output file.	0 = do not save output. 1 = save output.

Merge

The currently opened project will be merged with the content of the specified project. The open project becomes the resulting project, while the merger project remains as a standalone project.

Syntax

Merge , *Project name, host alias, keep existing items*

Parameter	Description	Value
Project name	The name portion of the project, for example, in the project “Test EN US BASE alias”, ‘Test’ is the name portion.	String.
Host alias	The alias name of an Oracle database host.	Should be a valid Oracle database alias.
Keep existing items	Indicates whether to keep existing items in the merged project if the two projects have the same objects.	0 = Overwrite existing data. 1 = Keep existing data .

Rename Project

Syntax

RenameProject , *Project name, language, country, project type, host alias, new project name*

Parameter	Description	Value
Project name	The name portion of the project to rename, for example, in the project “Test EN US BASE alias”, ‘Test’ is the name portion.	String.
Language	The two-letter language identifier.	Should be a valid identifier as defined in the Hublang.ini file.
Country	The two-letter country identifier.	Should be a valid identifier as defined in the Hublang.ini

		file.
Project type	Indicates whether this is a base or translation project.	BASE = base project. TRANS = translation project.
Host alias	The alias name of an Oracle database host.	Should be a valid Oracle database alias.
New project name	The name portions of a full project name. The language, country, project type and host alias values are taken from the source project to make up the full new project name.	User defined

Delete Project

Syntax

DeleteProject , *project name, language, country, project type, host alias*

Parameter	Description	Value
Project name	The name portion of the project, for example, in the project “Test EN US BASE alias”, ‘Test’ is the name portion.	String.
Language	The two-letter language identifier.	Should be a valid identifier as defined in the Hublang.ini file.
Country	The two-letter country identifier.	Should be a valid identifier as defined in the Hublang.ini file.
Project type	Indicates whether this is a base or translation project.	BASE = base project. TRANS = translation project.
Host alias	The alias name of an Oracle database host.	Should be a valid Oracle database alias.

Import

Syntax

Import , *Project File*, *Translation Directory*, *File Format*, *Version*, *App*, *Mod*, *Overwrite*,
Always Return Success, *Save Output*

Parameter	Description	Value
Project File	A text file containing a list of files to be imported.	String.
Translation Directory	Folder containing corresponding Translation files (use "." if none).	String.
File Format	The type of the files to be imported – FMB, MMB, REPORT	String.
Version	The version of files being imported.	String.
App	Application into which files are to be imported.	String.
Mod	Module into which files are imported (if blank, extracts to 'Unknown Module').	String.
Overwrite	Boolean (1 or 0). Overwrite existing data in the project. Note that it puts the data into the Current string and does not change the Old string.	0 = do not overwrite existing data. 1 = overwrite existing data.
Always Return Success	Value that indicates whether to continue processing the batch file if an error is found.	0 = do not continue processing. 1 = do continue processing.
Save Output	Specifies whether the output from this operation, normally seen in the output window, is saved to the output file.	0 = do not save output. 1 = save output.

If 'App' parameter is blank, then the import command imports to 'Unknown App'.

If 'Mod' parameter is blank, then the import command imports to 'Unknown Module'.

MIGRATION EXAMPLE

In this example, an application string of two Reports modules and two Forms modules is captured in TranslationHub. The base development language is American English, while the available translation languages are German and French.

The scripts provided here as examples do not contain all possible TranslationHub commands, but ought to provide you with the basis for using additional commands where needed.

For clarity, strings stored in Reports modules are captured independently from the strings stored in Forms modules.

There is no reason why the complete migration cannot be done in a single script file. However, note that if you use a single migration file, the file containing the source modules must be different for importing Forms and Reports modules. It is not possible to use the same import command for both Reports and Forms definition files.

The script file containing the TranslationHub commands can be commented out with a semicolon in front of each line. With the leading semicolon, the following line is not executed although it contains a valid TranslationHub delete command:

```
; DeleteProject , "myProject", "DE", "DE", "BASE", "ORCL"
```

A script file can contain two sections: a [REPORT] section, specifying log files and their location, and a [RUN] section containing the actual TranslationHub batch commands.

Capturing translation strings stored in Oracle Reports

As noted earlier , Oracle9i Reports modules need to be converted into the Reports rex file format before you can import any strings into TranslationHub. Because rex file formatted Reports application modules contain only one set of language strings, there are at least two Reports rex file modules required when you are capturing an existing translation: The first rex file contains the base development language strings, used by TranslationHub as a reference, while the second rex file contains the language strings previously translated with Oracle Translation Builder.

At least two scripts are involved when capturing existing translation strings from a Reports rex module format.

The first script, the migration script, contains the actual TranslationHub commands to be executed by the command-line utility. These commands actually perform the string capturing.

The second script lists the application source file names that the strings are captured from.

Converting Reports using the command line

The command line allows you to convert a series of Reports modules from an rdf format into a rex format using batch processing. To create language specific Reports rex formats, you'll need to set the NLS_LANG parameter to the target language before executing the rwconverter command. To extract existing German language strings out of a Reports rdf module, issue the following command before creating the rex file:

```
Set NLS_LANG=GERMAN_GERMANY.WE8ISO8859P1
```

To obtain existing French translations, issue:

```
Set NLS_LANG=FRENCH_FRANCE.WE8ISO8859P1
```

After setting the language environment, you are ready to run the Oracle9i Reports converter.

For the conversion to be successful, there must exist a translation in the rdf module that matches the language setting specified in the NLS_LANG variable. If no translation strings exist, the created rex file contains the strings of the base development language.

RWCONVERTER command-line arguments

For information on using the Reports converter, refer to the Oracle9i Reports Developer online

Only a subset of the available converter command-line arguments is required to convert an Oracle9i Reports paper layout module from an rdf format to a rex format:

STYPE= RDFFILE or REXFILE

SOURCE= source_name or (source_name1,source_name2,...)

DTYPE= RDFFILE or REPFIL or REXFILE

DEST=destination_name or (destination_name1,destination_name2,...) or pathname

LOGFILE=logfile

OVERWRITE= YES or NO or PROMPT

BATCH = YES or NO

Reports rdf to rex file format conversion example

To convert two Reports modules, Rep1.rdf and Rep2.rdf, from an rdf format to a rex format, use the following command. Set batch=yes to perform batch processing.

```
Rwconverter stype=RDFFILE source=(Rep1.rdf,Rep2.rdf) dtype=REXFILE  
dest=(Rep1.rex, Rep2.rex) logfile=log.txt overwrite=yes batch=yes
```

For this example the log.txt file contains the following information

Converting 'Rep1.rdf' to 'Rep1.rex'...

Converting 'Rep2.rdf' to 'Rep2.rex'...

Converting Reports using the graphical UI

The Oracle9i Reports converter can be started either from the program bar of the Forms and Reports product group or from the command line. The latter makes it easier to set the `nls_lang` environment to the language that should be contained in the resulting Reports rex file. To start the Reports converter for creating a German rex file version, issue the following commands from the bin directory of your Oracle9i Reports Home:

1. Set `NLS_LANG=GERMAN_GERMANY.WE8ISO8859P1`
2. `rwconverter`

To convert a Reports rdf file to a rex file format, select the Reports rdf file in the source field. Define the destination type to be “Reports ASCII file (rex)” and determine a location for the new file to be written to.

If you are preparing for TranslationHub, this file should be written to the application language specific subdirectory created earlier.

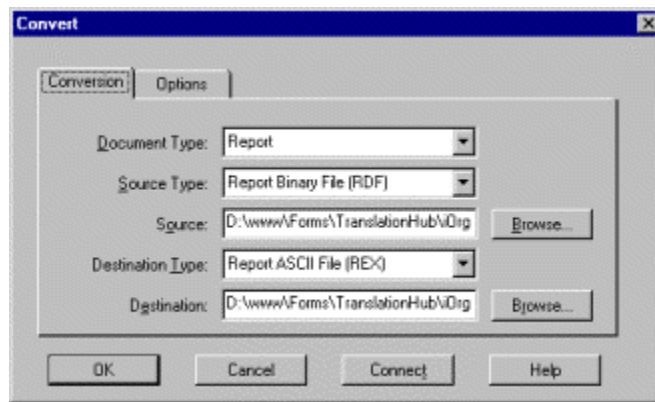


Figure 6: Reports Converter UI, converting rdf formats into rex formats

The Reports converter graphical UI can also be used to create a Reports rdf module out of a rex file definition. Before you can run a translated Reports module, it needs to be converted back into a “.rdf” file and optionally compiled into a Reports “.rep” file.

Sample migration

The following sequence needs to be contained in the migration script: Create a base translation project.

1. Create a translation project for German and French.
2. Open the base translation project.

3. Import the base development language string into the base project.
4. Update the translation projects with the strings from the base project.
5. Open the German translation project.
6. Capture existing German translation strings from a Reports rex file.
7. Open the French translation project.
8. Capture existing French translation strings from a Reports rex file.

Script 1: TranslationHub script for capturing strings stored in Reports rex file formats

```
[Report]
ReportFile=c:\temp\9iTrans.log
OutputWindowFile=c:\temp\9iTransError.out

[Run]

;New Project Syntax
;NewProject, <Project name>,<language>, <country>, <base ;language>,
; <base country>, <project type>, <host alias>, overwrite
;(yes = 1, no=0)
;
;Create new base project 'FormsRepTrans' with NLS_LANG
;AMERICA_AMERICAN
NewProject,"FormsRepTrans","EN","US","BASE","ORCL",1
;
;Create a new translation project for German and French
;
NewProject,"FormsRepTrans","DE","DE","TRANS","ORCL",1
NewProject,"FormsRepTrans","FR","FR","TRANS","ORCL",1
;
;For Canadian French it would have been
;NewProject,"FormsRepTrans","FR","CA","EN","US","TRANS",
;"ORCL",1
;Import the development language strings into the base ;project
;
;Open Project Syntax
;OpenProject, <Project name>, <language>, <country>,<project type>,
;<host alias>, <create>, <mode>
;
OpenProject, "FormsRepTrans","EN","US","BASE","ORCL",0,0
;
;Import Syntax
;Import, <Project File>, <Translation Directory>,
```

```

; <File Format>, <Version>, <App>, <Mod>, <Overwrite>, <Always Return
; Success>, <Save Output>
;
Import,
"modules.txt", ".", "REPORT", "1", "MyReptestApp", "AppReports", 1, 1, 1
;
; Update, translation projects with the content of the base project
; Update, Syntax
;
; Update, <Project name>, <language>, <country>, <host alias>,
; <overwrite>, <ignore case>,
; <update from text>, <save output>
Update, "FormsRepTrans", "DE", "DE", "ORCL", 1, 1, 0, 1
Update, "FormsRepTrans", "FR", "FR", "ORCL", 1, 1, 0, 1
;
; Capture existing translation strings into the translation projects
;
OpenProject, "FormsRepTrans", "DE", "DE", "TRANS", "ORCL", 0, 0
Import, "modules.txt", " D:\MYPROJECT\REPORTS\DE", "REPORT",
"1", "MyReptestApp", "AppReports", 1, 1, 1

OpenProject, "FormsRepTrans", "FR", "FR", "TRANS", "ORCL", 0, 0
Import, "modules.txt", " D:\MYPROJECT\REPORTS\FR", "REPORT",
"1", "MyReptestApp", "AppReports", 1, 1, 1
;
; Done
;

```

The migration script uses a file named "modules.txt", which contains the names and locations of the Reports source files.

Script 2: “modules.txt” script containing the Reports modules

```

; imports file from the directory storing the OTB translations
d:\MyProject\Reports\Rep1.rex
d:\MyProject\Reports\Rep2.rex

```

To run the migration script, make sure that both the migration script itself and the file containing the module names share the same directory. From the command line in the Oracle9iDS_Home\bin directory, type:

Otbatch <location of migration file>\<migration file name>

Script 3: Excerpt of the TranslationHub log file created by this script file

```
Importing Base...
MyReptestApp/AppReports/Rep1.rpx/Report:Parameter/
  d: \MyProject\Reports\Rep1.rpx
MyReptestApp/AppReports/Rep1.rpx/Report:Layout/
  d:\www\TranslationHub\MyProject\Reports\Rep1.rpx
MyReptestApp/AppReports/Rep2.rpx/Report:Parameter/
  d: \MyProject\Reports\Rep2.rpx
MyReptestApp/AppReports/Rep2.rpx/Report:Layout/
  d: \ MyProject\Reports\Rep2.rpx
Import - done

Updating...
MyReptestApp/AppReports/Rep1.rpx/Report:Parameter/
MyReptestApp/AppReports/Rep1.rpx/Report:Layout/
MyReptestApp/AppReports/Rep2.rpx/Report:Parameter/
MyReptestApp/AppReports/Rep2.rpx/Report:Layout/
Update - done
...
```

Capturing translation strings stored in Forms

TranslationHub can directly read from Forms fmb files, which is why there are no additional file conversions required. A Forms fmb file, translated with Oracle Translation Builder, can contain strings of several different languages. The process of importing the strings into TranslationHub is similar to Reports, except that there is only one Forms application module involved.

If a translated Forms application module is created by TranslationHub, it contains only two types of strings: those of the base development language, and those of the language chosen as the TranslationHub project language.

Sample migration script

For the Forms migration in this example, the assumption is that the TranslationHub base translation project and the translation projects already exist. If they haven't been created yet, the script must create them first.

The following sequence needs to be contained in the migration script:

1. Open the base translation project.
2. Import the base development language string into the base project.
3. Update the translation project with the strings from the base project.
4. Open the German translation project.
5. Capture existing German translation strings from a Forms fmb file.

6. Open the French translation project.
7. Capture existing French translation strings from a Forms fmb file.

Script 4: TranslationHub script for capturing strings stored in Forms fmb file formats

```
[Report]
ReportFile=c:\temp\9iTrans2.log
OutputWindowFile=c:\temp\9iTransError2.out

[Run]
;Import the development language strings into the base project
;
;Open Project Syntax
;OpenProject, <Project name>, <language>, <country>, <project type>,
;<host alias>, <create>, <mode>
;
OpenProject, "FormsRepTrans", "EN", "US", "BASE", "ORCL", 0, 0
;
;Import Syntax
;Import, <Project File>, <Translation Directory>, <File Format>,
<Version>, <App>,
;<Mod>, <Overwrite>, <Always Return Success>, <Save Output>
;
Import, "modules2.txt", ".", "FMB", "1", "MyReptestApp", "AppForms", 1, 1, 1
;
;Update, translation projects with the content of the base project
;Update, Syntax
;
;Update, <Project name>, <language>, <country>, <host alias>,
;<overwrite>, <ignore case>,
;<update from text>, <save output>
Update, "FormsRepTrans", "DE", "DE", "ORCL", 1, 1, 0, 1
Update, "FormsRepTrans", "FR", "FR", "ORCL", 1, 1, 0, 1
;
;Capture existing translation strings into the translation projects
;
OpenProject, "FormsRepTrans", "DE", "DE", "TRANS", "ORCL", 0, 0
Import, "modules2.txt", " D: \MYPROJECT\FORMS\DE", "FMB",
"1", "MyReptestApp", "AppForms", 1, 1, 1
OpenProject, "FormsRepTrans", "FR", "FR", "TRANS", "ORCL", 0, 0
Import, "modules2.txt", " D:\MYPROJECT\FORMS\FR", "FMB",
"1", "MyReptestApp", "AppForms", 1, 1, 1
;Done
```

As you can see, the script commands are the same as those used when capturing strings in Oracle Reports modules. One difference, (aside from the fact that the TranslationHub projects are assumed already to have been created), is that the import command uses “FMB” to specify the import file type.

Script 5: “modules2.txt” script containing the Forms modules

```
; imports file from the directory storing the OTB translations
d:\MyProject\Forms\forms1.fmb
d:\MyProject\Forms\forms2.fmb
```

To run the migration script, make sure that both the migration script itself and the file containing the module names share the same directory. From the command line in the Oracle9iDS_Home\bin directory type:

Otbatch <location of migration file>\<migration file name>

Migration results

Opening the TranslationHub IDE after running the migration script shows that three new projects have been created. All projects have a name of “FormsRep Trans”, as specified in the migration script, but each is distinguished by project type and language.

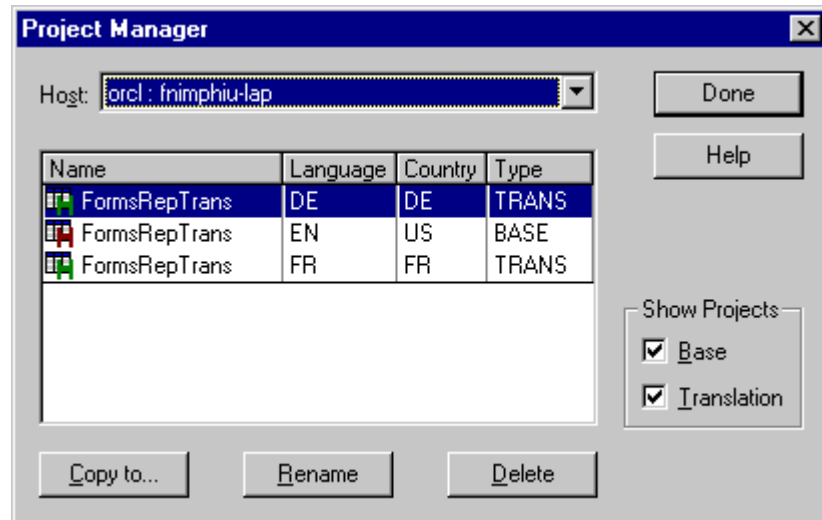


Figure 7: Project Manager showing projects created by the migration script

The translation project for German contains the base language strings in English as well as the German translations.

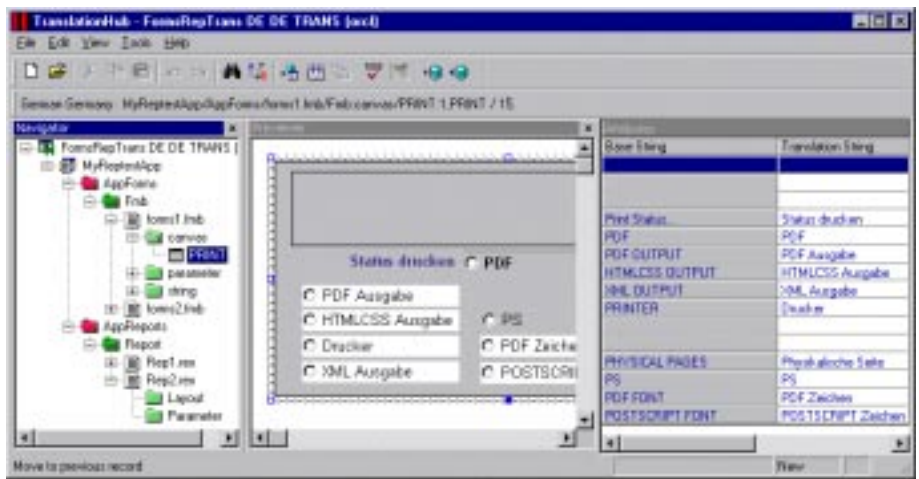


Figure 8: TranslationHub with German translation project showing Forms

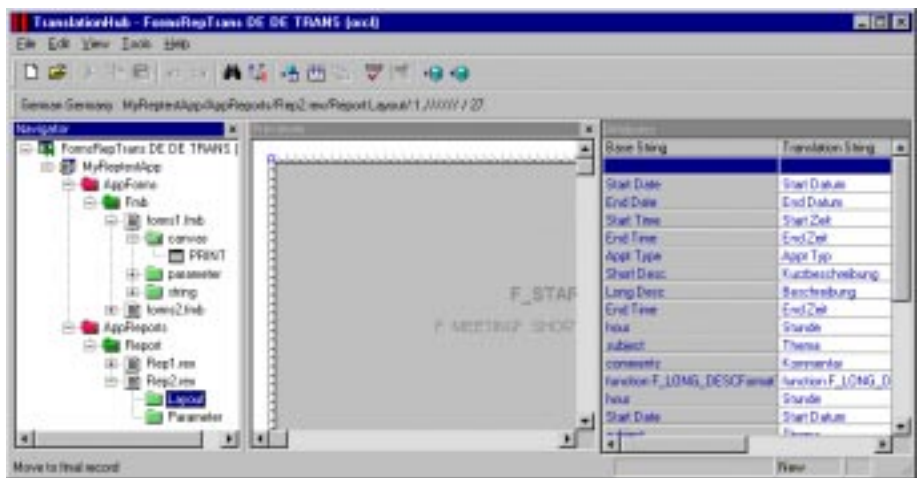


Figure 9: TranslationHub with German translation project showing Reports

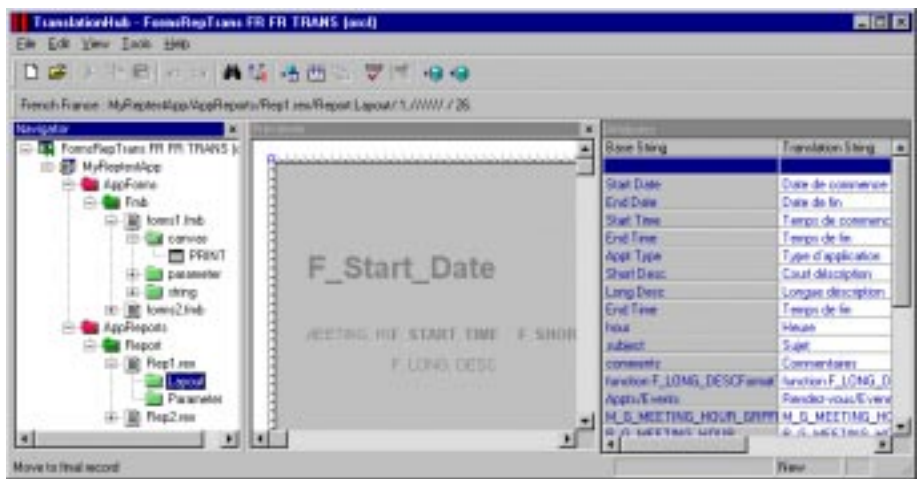


Figure 10: TranslationHub with French translation project showing Reports

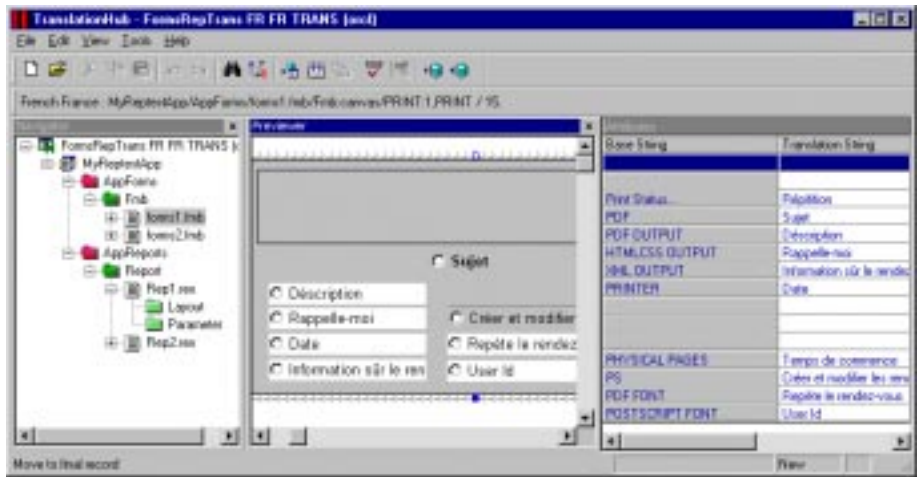


Figure 11: TranslationHub with French translation project showing Forms

Testing the migration success

There is no dedicated test utility to quantify whether or not migrating from Oracle Translation Builder to TranslationHub was successful. But TranslationHub has a utility that can be used to test the quality of a translation.

For every translation project in TranslationHub, the “Test and Fix” option is accessible from the “Tools” menu. The “Test and Fix” option, among others, allows you to search for strings that haven’t been translated.

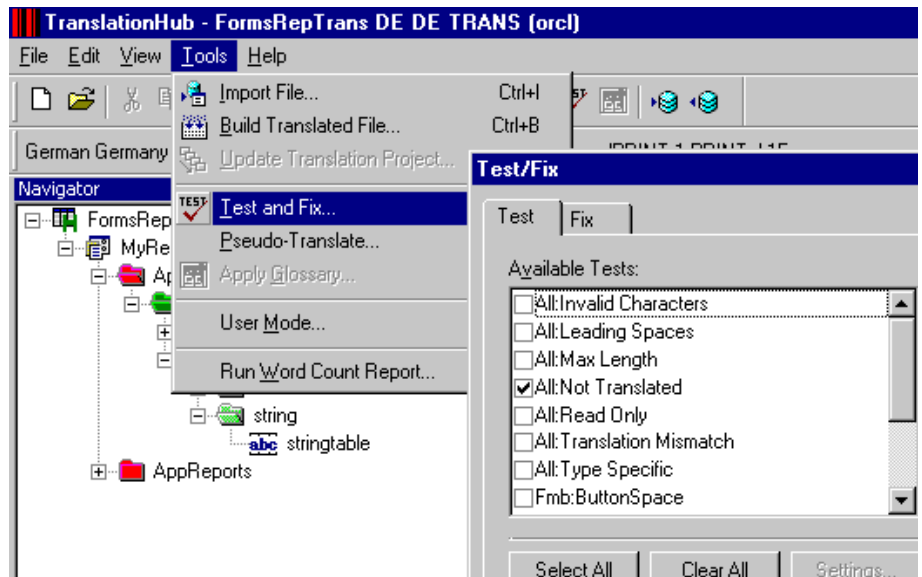


Figure 12: TranslationHub integrated translation quality tests

Because the translation project was created as an update operation of the base project, it contains all the strings available in the base development modules. Assuming that all the strings in an application module have been translated using Oracle Translation Builder once you import the translated strings into

TranslationHub, there should be no string from the base modules without a translation.

Another way to evaluate the success of the migration is to have a look into the TranslationHub log files that are written for each action performed.

CAPTURING STRINGS FROM FORMS 4.5 MODULES

To read strings stored in Forms 4.5 modules into TranslationHub you must first upgrade them to Forms 6i. Because of the architectural change in Forms from Forms 4.5 to Forms 6i, only strings of one language are included in the fmb modules after the upgrade using the Forms compiler ifcmp60.

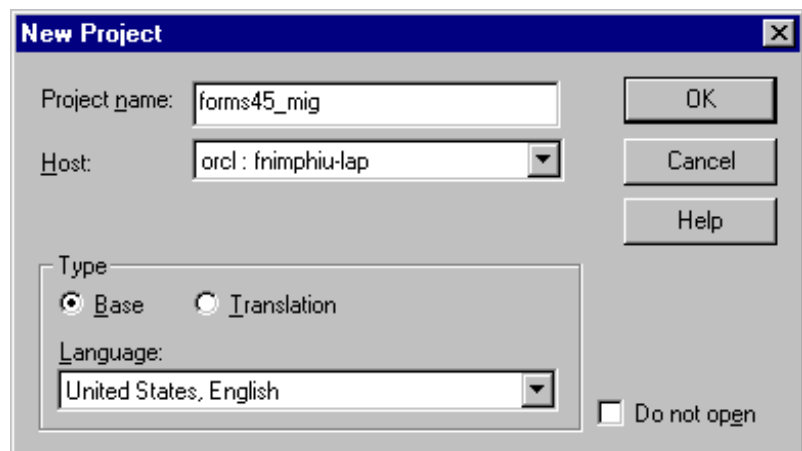
If your Forms 4.5. module contains more than one translation language, then, to retain strings of all languages, you must perform a separate migration for each individual language.

If the Forms 4.5. module was developed in English, and if translations have been done for French, German, and Italian, then to capture those strings, beginning with the German, into TranslationHub, here is what you'd do:

Note : Before upgrading Forms 4.5 modules to Forms 6i, make sure that no other language supplement files than the ones used with the base development language are installed on the Pc. If you already have other language supplement files installed, then please delete these before upgrading your application to Forms 6i .

The problem caused by the existence of language supplement files is a change in the internal ID used with boilerplates, making it impossible for TranslationHub to associate labels with their translation strings (Bug 2288871)

1. Create a base project in TranslationHub for the application. The language of the base project must match the language used whilst developing the application



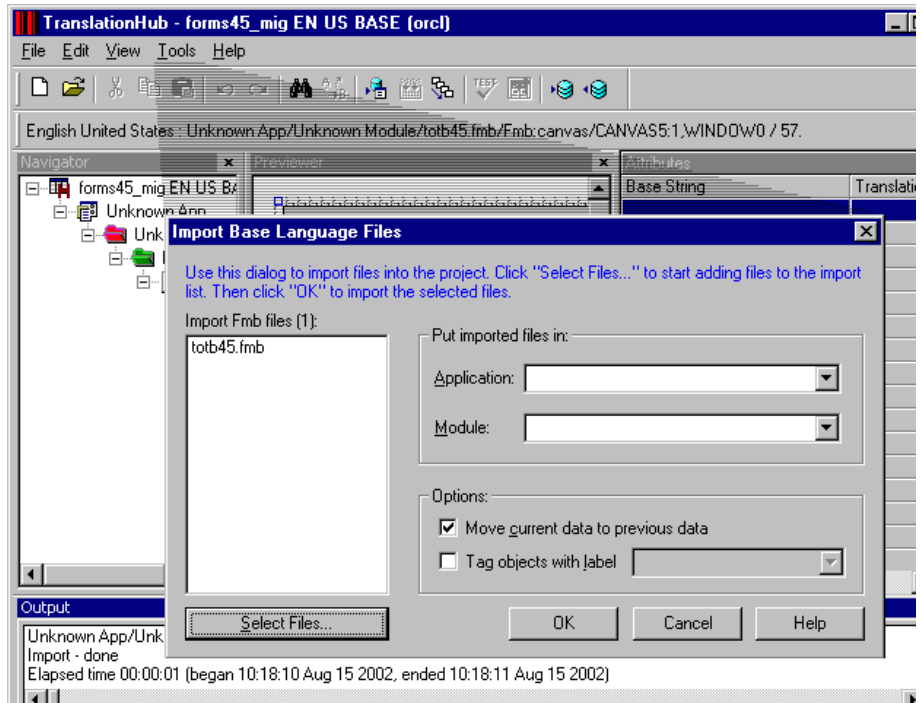
Upgrade Forms 4.5 to Forms6i, using the command line, for example:

```
set NLS_LANG= AMERICAN_AMERICA.WE8ISO8859P1
```

```
ifcmp60 module=totb45.fmb upgrade=yes version=45 batch=yes  
userid=scott/tiger@alias_base build=no
```

This creates an upgraded fmb file containing the language determined by the NLS_LANG environment variable (American in the above case) using the Western European character set.

In TranslationHub use the Tools → Import File menu option to import the base development strings into the TranslationHub Base project.



2. For each translation language, upgrade Forms 4.5 to Forms 6i using the command line:

```
set NLS_LANG= <LANGUAGE>_<COUNTRY>. <CHARSET>
```

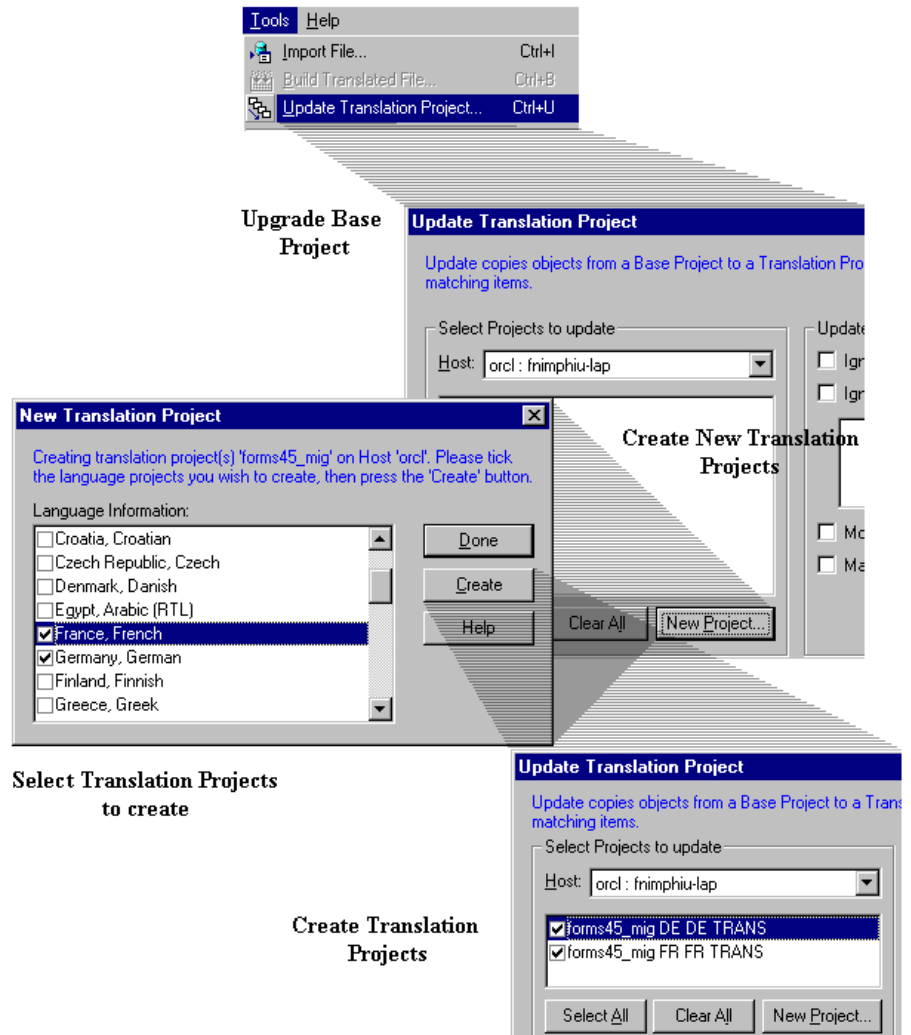
for example

```
set NLS_LANG= FRENCH_FRANCE.WE8ISO8859P1
```

```
ifcmp60 module=totb45.fmb upgrade=yes version=45 batch=yes  
userid=scott/tiger@alias_base build=no
```

This creates an upgraded fmb file containing the language determined by the NLS_LANG environment variable (French in the above case).

3. Create a translation projects in TranslationHub for all translation languages contained in the Forms module (French and German in this example). This can be done individually for each language or in one working step using the Tools → Update Translation Project menu option.

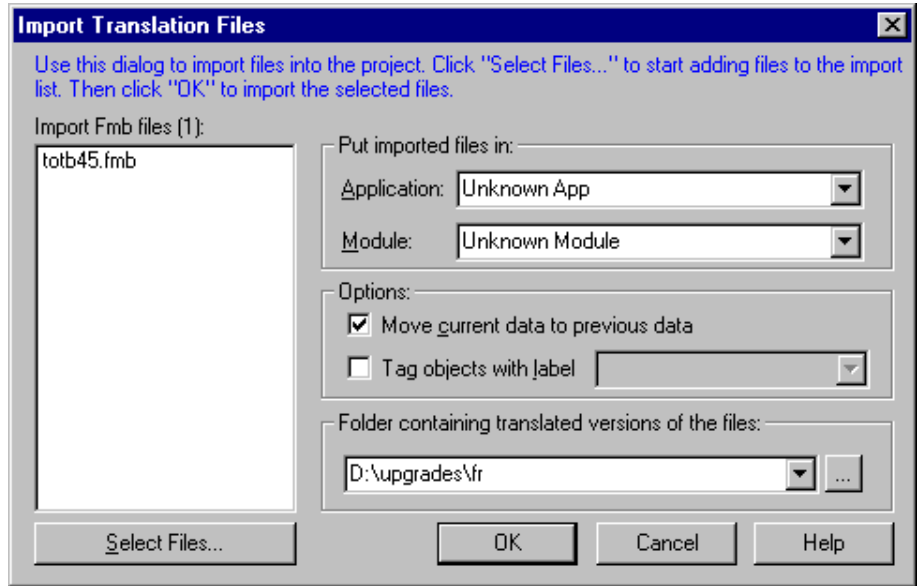


In the following dialog click onto the “New Project” button.

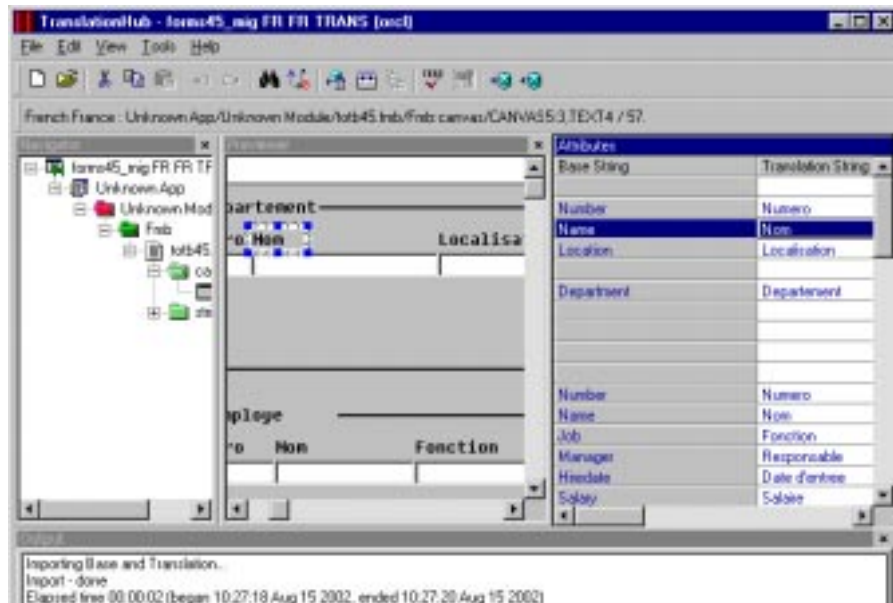
4. Import the upgraded Forms6i fmb file containing the base development language and select the folder containing the translated Forms module created in step 2 in the “Folder containing translated versions of the files” input field of the import dialog.

It is recommended that forms module fmb files containing the translation strings be saved in a subdirectory under the directory containing the upgraded files of the base language.

Note: The “Move current data to previous data” checkbox shown in the screenshot below is not relevant for the migration. This checkbox is useful only when importing the same translated Forms fmb file again.



- Repeat steps 2, (3), 4 and 5 for each language contained in the Forms 4.5. module.



6. Upgrade your Forms6i fmb files to Forms9i. TranslationHub, as a subcomponent of Forms 9.0.2 and 9.0.3, is not certified for exporting translation string into Forms6i modules, which means that currently only read operations are supported².

² The certification for write operations is targeted for Forms 9.0.4, which is expected for end of 2002

FAQ

Q: Can I use TranslationHub with Forms6i modules?

A: No. TranslationHub is not certified with Forms6i. This is planned for one of the later patch sets.

Q : TranslationHub is not certified with Forms6i modules in its base release. Will it become a migration path from Forms6i to Forms9i in a future patch set ?

A: No. TranslationHub takes a Forms module and extracts the translatable strings from it. When building a translated version of a Forms module then all that TranslationHub does is to take the original Forms module and write back the translation strings. TranslationHub does no migration or PL/SQL conversion.

Q: Is Oracle Translation Builder (OTB) desupported?

A: No. TranslationHub is for translation of Forms9i modules and OTB is still the supported translation environment for Forms 6i modules. The reason for planning to expose TranslationHub to Forms6i customers is simply for customer convenience, avoiding the maintenance of two separate translation repositories.

Q: I am using Forms 4.5, how do I migrate to TranslationHub?

A: The only supported Forms release for TranslationHub is Forms9i. If you want to migrate from OTM to TranslationHub then you first have to upgrade your Forms to Oracle9i Forms. From here you can import the translation strings contained in a Forms module as described throughout this paper.

Q: Is there a script available that allows me to migrate the Oracle Translation Builder repository to TranslationHub?

A: No.

Q: When using Oracle Translation Builder then all translation language strings could be stored in one Reports rdf file. With TranslationHub I have to convert one rdf file per language out of a translated rex file. Why is that?

A: TranslationHub uses the Reports rex format as a translation source. The rex format can only store one set of translation strings which is why you cannot get a multiple language rdf file out of one rex file.

Q: When building Reports rex files then the NLS_LANG parameter is used to capture translated language strings. For Forms fmb files TranslationHub directly uses ISO locales from the hublang.ini to load translation strings into the translation repository. Will this mean that translations are captured differently between Forms and Reports?

A: If your Forms application modules have been translated with OTB and OTM before, then the answer is no. There are no differences. NLS_LANG in Forms and

Reports and the ISO locale in TranslationHub are used to identify a translated string version in a module, this does not have an impact on the actual strings read.

Q: How can I test that a Reports rex file contains the target translation strings, exported from an rdf file format?

A: Use the Reports converter “rwconverter” to convert the rex file back into an rdf file format. Open the rdf file in the Reports Developer which now should show the translation strings.

Q: When I run the migration script then an error occurs saying that the command “ ” is not valid. What wrong in my script

A: Most likely you are using comments in your script. Check if each comment line is prefixed with a semicolon and not with a comma or a dot.

Q: The translation strings are the same as the base development strings when opening the TranslationHub translation project

A: A possible reason is that your base source file does not contain a translation for the specified language.

CONCLUSION

TranslationHub is the new, more powerful translation tool for translating Oracle9i Forms modules (fmb) and Oracle9i Reports paper layout modules (rdf).

The migration path from the Oracle Translation Manager or Oracle Translation Builder tools, is to use the translation strings stored in the Forms and Reports application definition files and store these strings into the TranslationHub database tables. To capture the translated application strings, either the TranslationHub graphical user interface can be used or the command line based TranslationHub batch executable.

The Reports definition file needs to be converted into the rex format, which can be performed using the Reports conversion utility. Because the rex format can only contain one language, you may need to change your translation processes.

Capturing translation strings stored in Forms modules of release 6i is not supported in the TranslationHub base release but is planned for a future patch set.



Migrating from Oracle Translation Builder to Oracle9i Forms and Oracle9i Reports TranslationHub
June 2002

Author: Frank Nimphius

Contributing Authors: Duncan Mills, Chris Lowes

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
www.oracle.com

Oracle Corporation provides the software
that powers the internet.

Oracle is a registered trademark of Oracle Corporation. Various
product and service names referenced herein may be trademarks
of Oracle Corporation. All other product and service names
mentioned may be trademarks of their respective owners.

Copyright © 2002 Oracle Corporation
All rights reserved.