Quick Reference for Translating Complex File Formats

Once upon a time the overwhelming majority of computerized translation was done in word processing or text formats. If there was any question about format it was maybe whether the source was in *Microsoft Word* or *WordPerfect* format (with the latter most often being the case!). Today, things clearly have changed. There are a host of formats out there and fewer and fewer translators, especially those who work in more or less technical fields, get by with just using word processing application(s).

While I have dealt with the more common office formats in earlier sections (see *Office Suites* on page 117), in this section I have attempted to categorize the most commonly required more advanced file formats. You will find descriptions of the programs for which these are written, how to distinguish between the translatable vs. untranslatable parts, and how these formats are supported by computer-assisted translation tools.

The categories of file formats are the following:

- Desktop publishing formats (formats for programs such as *QuarkXPress*, *PageMaker*, *InDesign*, *FrameMaker*, etc.)
- Graphic formats (pixel-based: .jpg, .gif, .bmp, .tiff, etc., and vector-based: .eps, .ai, etc.)
- Tagged files (.html, .xml, .sgml, etc.)
- Software development formats (binary formats: .dll, .exe, .ocx, etc. and text-based formats: .rc, .properties, .resx, etc.)
- Help systems (*WinHelp* and *HTMLHelp*)
- Database-based data

Desktop Publishing Formats

For the basics of desktop publishing formats, please read *Desktop Publishing Programs* on page 151. In this chapter, you will find some of the specific steps that need to be taken for the different DTP formats.

There are two different approaches to translating DTP files that depend on whether they come from programs that are intended for design-oriented publications (*Adobe PageMaker*, *QuarkXPress*, and *Adobe InDesign*) or from programs for content-oriented publications (*Adobe FrameMaker* and *Corel Ventura*).

First of all, any of these formats is, of course, directly translatable in its own environment—i.e., you can overwrite the text of a *PageMaker* file within *PageMaker*—but you will have to save these formats to a non-compiled format (i.e., text-based format) to process them in a computer-assisted translation tool.

In the design-oriented programs, text is handled in individual and independent text boxes that can be placed anywhere in the application. These text boxes, the so-called "stories," can be saved in individual text boxes from which the text has to be manually exported into an application-specific text format and re-imported if you want to process it in a translation environment program. While this is theoretically not an issue, it is super time-consuming when you have to do this for tens or even hundreds of stories in one document. This means also that even if CAT programs claim that they can process the native export format of *PageMaker*, *Quark*, or *InDesign*, only a few allow the batch processing of all the text fragments involved.

Another time-consuming task for any of these formats is that due to text-expansion, the stories will have to be resized after translation—so you need to make sure that you take that into consideration when accepting a job or quoting for a job in any of these programs!

This is not where the problems stop, though. Especially *QuarkXPress* (up to version 6.5) and *PageMaker* are still very "last century" when it comes to processing multilingual text. Though Unicode (see page 4) is a widely accepted standard that makes it easy to mix and match different writing systems on web pages and all kinds of other documents, DTP programs are

not up to par on this. Even though *Quark* does now support Unicode with its version 7 (released in the summer of 2006), *PageMaker* most likely will not because the folks at Adobe have a better choice when it comes to processing Unicode: *InDesign*.

InDesign

After a fairly unsuccessful version 1, *InDesign* really gained traction beginning with version 2. Presently you will encounter *InDesign* files that are created in versions 2, CS (3), or CS2 (4). To efficiently translate in *InDesign* you will need a program that exports all the stories (the above-mentioned text boxes) into one large file which can be processed in a computer-assisted translation tool. (Of course, it is possible to translate directly within *InDesign*, but the emphasis was on "efficient.")

Trados offers little plug-ins as part of all its versions of the Workbench product that support InDesign versions 2 or CS (the plug-ins are stored under `C:\Program Files\TRADOS\Txx_xx\FI\IND`—follow the instructions in the help file on how to install the plug-ins). Once you have installed the plug-in and opened the *InDesign* file, you will see a new *Trados* menu with all the necessary commands to export and re-import your file.

```
<STORY NAME="119" LOCATION="P16"><UNICODE-WIN>
<Version:2.000000><FeatureSet:InDesign-Roman><ColorTable:=<Black:COLOR:CMYK:Proc
ess:0.000000,0.000000,0.000000,1.000000>>
<DefineCharStyle:Bold Oblique=<Nextstyle:Bold
Oblique><cColor:Black><cTypeface:67 Bold Condensed
Oblique><cSize:12.000000><cHorizontalScale:1.000000><cLigatures:0><cTracking:0><
cBaselineShift:0.000000><cCase:Normal><cStrokeColor:None><cVerticalScale:1.00000
0><cNoBreak:0><cUnderline:0><cFont:Univers><cPosition:Normal><cStrikethru:0><cCo
lorTint:-1.000000>>
<DefineParaStyle:Normal=<Nextstyle:Normal><cTypeface:Medium><cLigatures:0><cBase
lineShift:0.000000><pHyphenationLadderLimit:0><pAutoLeadPercent:1.204987><pMinCh
arBeforeHyphen:3><pHyphenateCapitals:0><pShortestWordHyphenated:6><pHyphenationZ
one:0.000000><cFont:Helvetica><pDesiredWordSpace:1.100006><pMaxWordSpace:2.50000
0><pMinWordSpace:0.850006><pMaxLetterspace:0.039993><cHang:Baseline><pRuleAboveC
olor:Black><pRuleAboveTint:100.000000><pRuleBelowColor:Black><pRuleBelowTint:100
.000000>>
<ParaStyle:Normal><pBodyAlignment:Justify><CharStyle:Bold Oblique>
Note•<CharStyle:><cTypeface:47 Light Condensed
Oblique><cFont:Univers>:Throughout your workout you will hear a series of
"beeps" from your console. This is perfectly normal. These beeps signify various
changes taking place in the unit's programming. For instance, you will hear one
beep every time you press a key on the keypad, and one beep to engage higher
function keys such as X-Mode, and two beeps when they have selected a program
for you. If you've<cTypeface:><cFont:><cTypeface:47 Light
Condensed><cFont:Univers> <cTypeface:><cFont:><cTypeface:47 Light Condensed
Oblique><cFont:Univers>decided to end your workout and hit the Cool Down button
you will hear three. Stepping away or stopping completely will engage the Pause
mode which will signal with three beeps that you've stopped your workout before
it was finished and will begin a ten minute countdown before exiting the program
you were using. You will also hear beeps as you move from the warm-up into each
program.<cTypeface:><cFont:><cTypeface:47 Light Condensed><cFont:Univers>
<cTypeface:><cFont:><pBodyAlignment:></STORY>
```

Figure 147: View of Trados-exported InDesign text file in a text editor

As you can see in the above illustration, the text file is not just a "normal" text file; instead, it is a "tagged" text file where only the smallest part is actually translatable (essentially everything that in not enclosed by <tag markers>) and all the other data stores information about details such as formatting. While it theoretically would be possible to translate this within *Microsoft Word* or a text editor, it would be foolish to even try—chances are that you would break the code or overlook text.

Programs such as *Trados TagEditor* or *Déjà Vu*, however, recognize these files as *InDesign* files, protect all coding information, and display only translatable text.

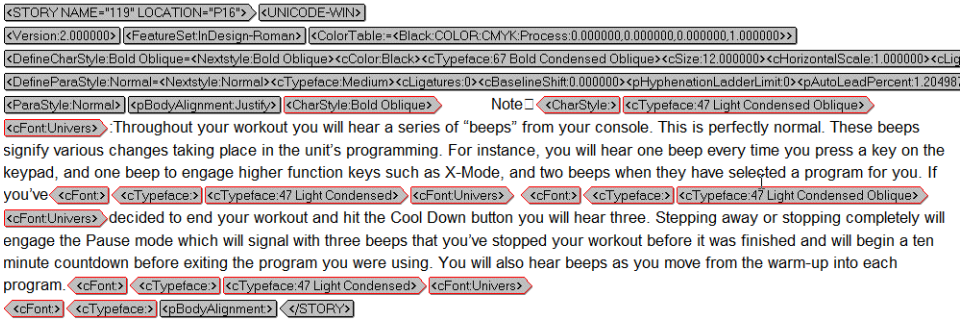


Figure 148: View of Trados-exported InDesign 2 text file in Trados TagEditor

The latest version *Star Transit* (with a separate plug-in) offers the option of translating *InDesign* files, but just for versions 2 and CS. *Heartsome* as well as the latest versions of *Trados* and *SDLX* support *InDesign* CS2 files in their *InDesign*-specific XML format (.inx) which, like the tagged text files, can be reimported once the translation is finished.

ECM-Engineering (see www.ecm-engineering.de) offers an independent filter for all current versions of *InDesign* (including CS2). This application allows you to export into an RTF format that is supported by *Trados*, *Wordfast*, *SDLX*, or *Déjà Vu* and it gives you the choice to do this in a tagged format or in a WYSIWYG (what-you-see-is-what-you-get) format with intact formatting.

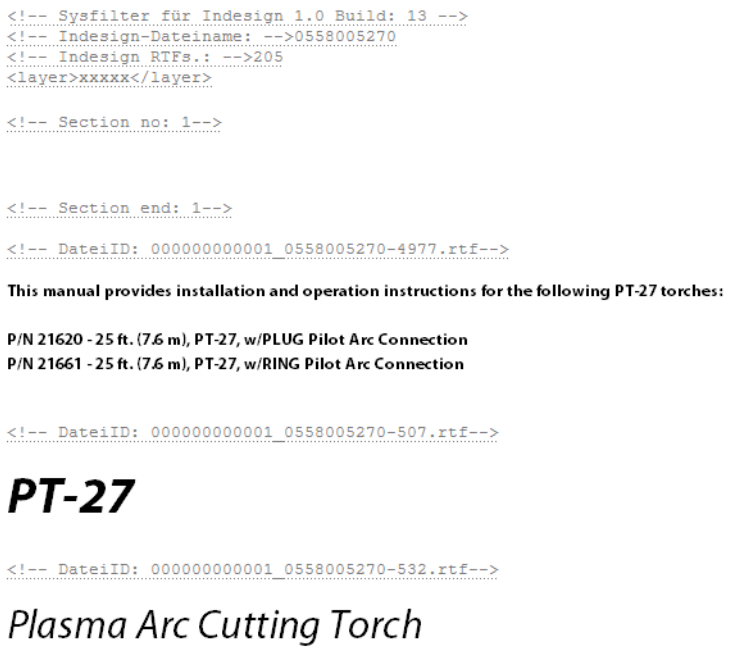


Figure 149: View of ECM-exported InDesign file in WYSIWYG format

PageMaker

To translate *PageMaker* files (an increasingly rare occurrence because Adobe is trying to push *InDesign* over *PageMaker*) with a computer-assisted translation tool, you could either use *Star Transit* with a separate plug-in with support for *PageMaker* 6-7 or a plug-in that comes with the *Trados* product called *Story Collector for PageMaker* and supports versions 6.5 and 7.

To install the *Trados* plug-in, open the help file under `C:\Program Files\TRADOS\Txx_xx\FI\PM` for further instruction. Once the plug-in is installed, open the *PageMaker* file in *PageMaker* and you'll find the command *Trados Story Collector* under **Utilities> Plug-ins**.

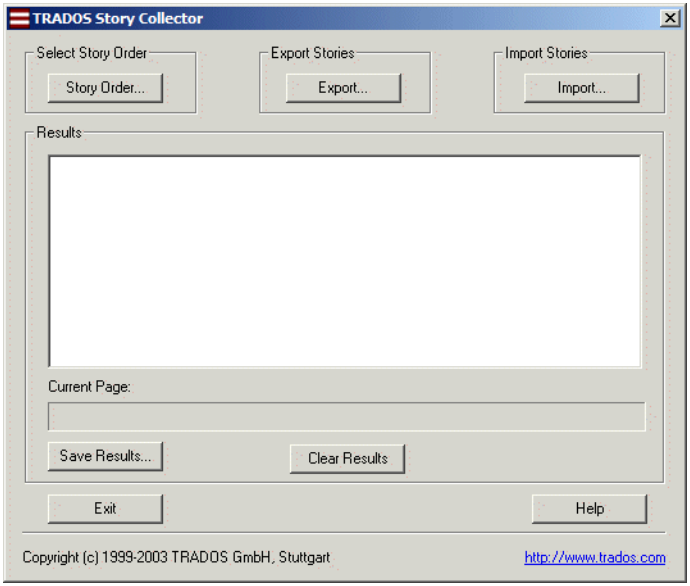


Figure 150: Trados Story Collector for PageMaker

Export all the stories into one large *PageMaker*-specific text file, save the original *PageMaker* file (important!), and translate the exported text file with *TagEditor* or any other application that supports the *PageMaker* format. The import process is virtually the same as the export and should go seamlessly.

For specific illustrations of the differences in exported text files in a text editor vs. TagEditor, see the InDesign-specific illustration on page 222—PageMaker (and QuarkXPress) have a very similar appearance.

All of the above is true for Western languages and to some degree for Eastern European languages. Any of the more complex languages, however, including the bi-directional languages (Hebrew and Arabic) or the Asian double-byte languages, are flat-out not supported in the Western versions of *PageMaker*.

Though you can purchase language-specific versions for these languages, it would make a LOT more *sense* to convert to *InDesign* and take it from there. Because *InDesign* and *PageMaker* are both Adobe products, the upgrade path is relatively easy (both in terms of purchasing a less expensive competitive upgrade version of *InDesign* when you already own *PageMaker* and in terms of converting the files).

QuarkXPress

Despite the fact that *Quark* has never been very popular in the translation community (because of a lack of Unicode support until recently and different and more expensive versions for different languages, etc.), it is (still) the market leader in desktop publishing, so it is not too surprising that there is decent support for different versions of *Quark* among the translation environment tools.

- *Star Transit* offers a separate plug-in that supports the batch processing of the English (and *Passport*) versions 3-6 for both the Windows and Mac platforms.
- *Trados* offers plug-ins for versions 4.1-6 for English (and *Passport*) and version 4.1 for Japanese.
- *SDLX* offers a plug-in for the English (and *Passport*) versions 4-6 for the Mac.

All of these plug-ins were preceded by a program called CopyFlow (see www.napsys.com) which, just like these programs, allows for the batch export and import of text from Quark files. It may still be worthwhile to take a look at Napsys' website—for instance, they offer plug-ins for Asian versions of Quark that no one else does. Also, users of some programs (I know of Déjà Vu, but there may be others as well) are eligible for a discount on CopyFlow products.

Also, the only plug-in for QuarkXPress 7 is presently the Mac version of CopyFlow.

Because of Unicode support in *Quark*'s version 7, it is now possible to write in all languages that are covered by Unicode within *Quark*. However, if you need language-specific capabilities (spell-checking, hyphenation, etc.), you may have to invest in one of the many language-specific versions.

The European language *Passport* edition of *Quark*, which has additional spell-checking and hyphenation capabilities for Western and European languages, is supported by the above mentioned tools. If you only have the (cheaper) English version, you need to make sure to ask your client to save the file as a "Single Language" file; if the *Passport* edition was used, you will not be able to open the file otherwise.

QuarkXPress's last Arabic edition was for version 4.1. Fortunately, however, there are *XTensions*—*QuarkXPress*-specific plug-ins—for the English version of *Quark* that extend its ability to write in Hebrew, Arabic, Farsi, and Jawi. *ArabicXT*, *HebrewXT*, *FarsiXT*, and *JawiXT* are all available at www.arabicsoftware.net for versions 5 and 6 of Quark.

Unfortunately, only ArabicXT is available for both Windows and Mac platforms, while the others are available only for Mac.

It becomes much more hairy with the Asian double-byte languages. While the Japanese version 4.1 is supported by the *Trados* plug-in and several others by *CopyFlow*, it at least means that you have to have several versions of *Quark* for different languages, plug-ins, and platforms.

Adobe FrameMaker and Corel Ventura

The content-oriented formats—*FrameMaker* or *Ventura*—offer a fairly painless way of saving the original compiled format (i.e., a format whose source coding is not humanly readable) into an interchange text-based format that can be easily processed. Because the emphasis for these files is on text and not on graphics, text is represented in one flow, and can be saved in a simple "Save as" process for each file (which is typically synonymous for one chapter). Also, the very concept of these programs is that there will be as much automation in the layout as possible. This is achieved, for instance, through fairly sophisticated widow and orphan rules so that there is only a small amount of additional pagination.

Furthermore, there is no problem with non-Western languages even in Western versions of the system (provided that your operating system supports it). And lastly, the size of the files tends to be relatively small because graphics are usually linked and not inserted.

Preparing FrameMaker Files

FrameMaker files typically come as part of a project that is organized within a .book file. The easiest and safest way to open them is by opening the .book file first and then opening the .fm files from within there.

Should the .fm files be displayed with an icon in the form of a question mark, you need to delete them from the book with the appropriate command from the Edit menu and then re-add them from within the ***Add*** *menu. Once the files are added, you can easily change the order of the files by simply dragging them within the .book interface.*

You will need to save the compiled .fm format within *FrameMaker* by selecting **File> Save as** and selecting the text-based .mif format. To avoid the individual opening and saving of each file, you can use the well-liked *MifSave* (see http://home.comcast.net/~bruce.foster/MifSave.htm) to do this as a batch process for a whole book.

(By the way, it's totally okay to ask your client to do this for you if you do not have *FrameMaker* on your computer.)

Once all your files are preprocessed, they are supported in any of the larger translation environment tools (*Trados*, *SDLX*, *Déjà Vu*, *Transit*), most of whose representatives will tell you that their *FrameMaker* processing is one of their strongest features—which only goes to show that *FrameMaker* is a very translator-friendly format.

There are slight differences in the way that the different tools process the .mif files. In *Trados* you need to convert the MIF files into .rtf files (so-called "STF" files) with a separate program that is part of the *Trados* suite of tools, the so-called *S-Tagger for FrameMaker* (usually located under **Start> Programs> Trados XX> Filters**), before you can translate them in either *Word* or *TagEditor*. The process of converting the files is slightly confusing if you do it for the first time, but the principle is this:

You will need to create two different directories, one of which will contain a set of files that will only serve as reference files so that the *FrameMaker* .mif files can be reassembled once they are translated in .rtf format. The other will contain the set of files that are actually prepared for translation. Keep that in mind in both conversion processes (into .rtf and back into .mif) so that you select the correct directories either way.

Many translators do not like to work in TagEditor, but in many ways it is much more suited for the translation of FrameMaker files than Word. For one thing, TagEditor offers a much better tag protection than Word, and it also allows for an immediate code integrity check once the translation is done (you can select the code check under ***Tools> Verify*** *in TagEditor). This makes the sluggish tag checking within the S-Tagger application obsolete.*

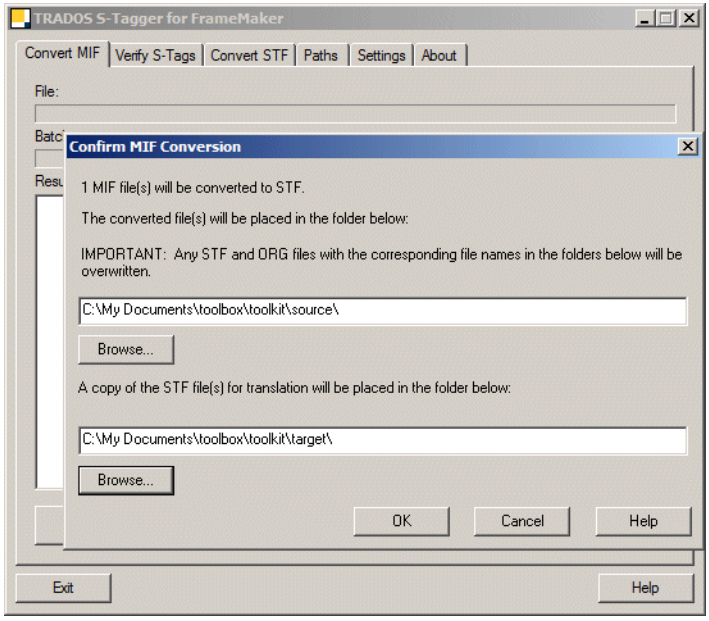


Figure 151: Trados S-Tagger for FrameMaker with tabs to convert files and verify tags

Both *Trados* and *SDLX* create additional files (the *Trados* version is called ancillary.rtf; the SDLX files are SDLX_ix.mif, SDLX_xr.mif, and SDLX_vr.mif), which contain background information such as index markers (*SDLX* only), cross-references, and information from the master page.

The other tools process the .mif files directly and translate all the background information individually for each file.

Preparing Ventura Files

Trados is the only CAT application that supports the *Ventura* format—but don't worry, there are very few translation projects in that format.

The process for translating *Ventura* within *Trados* files is very simple: You will need to export the content of the original .vp files to text files (**File> Export Text> ANSI text**), translate those in *TagEditor*, and reimport the translated text at the place where you want the text to be inserted (**File> Import Text**).

Graphic Formats

Pixel-Based Formats

Most graphic formats (including .jpg, .gif, .bmp, .tiff, and various others) don't contain text. This is true even if it appears to be readable text because the text is nothing more than pixels (little colored dots) on a virtual canvas. While they may form shapes that represent letters, these have nothing to do with the editable letters or words you will deal with in a text editor.

Short of recreating these kinds of graphics from scratch, you will need to get ahold of the "source files." (Yes, I know that clients hate to be asked for that, but typically it helps to mention that otherwise they will have to pay ten times as much.)

Most any of the .jpg-, .gif-, .bmp-, or .tiff-like files were created in a layered file that included one (or several) layers with real, editable text. Since they were most likely created in *Adobe Photoshop*, they will have a .psd extension and can be opened in, well, *Adobe Photoshop*. Nice thing is, Adobe is offering a very low-priced version of its program (see www.adobe.com/products/

photoshopelwin) which is more than adequate for translators to translate the text layers that need to be translated. (By the way, the current editions of *Photoshop* are Unicode-enabled and can thus be used with all languages that are supported by Windows.)

This all may not be good enough, though. Especially if you have a large number of graphics and/or a translation memory database that contains much of the translation that is contained within the graphics, you will not want to perform the translation "manually."

There are a variety of tools that allow for the extraction of text from .psd files into a translation environment tool-specific format:

- Enlaso's and Yves Savourel's *Rainbow* tool does a lot of different things (see www.translate.com/technology/tools), including the seamless preparation of .psd files into XLIFF (for a definition of XLIFF, see page 208) or *Wordfast*- or *Trados*-specific RTF formats. Though this feature is not included in the freely downloadable version, you can request a key to have

this feature added (under **File> Preferences**). If you are not a competitor of Enlaso (i.e., a translation agency), chances are the key will be provided.

```
<?xml version="1.0" ?>
<xliff version="1.0">
 <file original="siteidentifier-placeholder.psd"
  source-language="EN-US" target-language="DE-DE"
  tool="Rainbow" datatype="photoshop">
  <header>
   <skl>
    <external-file uid="424094A74240A588"
     href="siteidentifier-placeholder.psd.skl"/>
   </skl>
   <phase-group>
    <phase phase-name="extract" process-name="extraction"
     tool="Rainbow" date="20050322T150856Z"></phase>
   </phase-group>
  </header>
  <body>
   <trans-unit id="1" maxbytes="104">
    <source xml:lang="EN-US">Optional SW Link 1
Site Wide Link 2
Site Wide Link 3</source>
    <target xml:lang="DE-DE">Optional SW Link 1
Site Wide Link 2
Site Wide Link 3</target>
   </trans-unit>
   <trans-unit id="2" maxbytes="58">
    <source xml:lang="EN-US">Optional Secondary Identifier</source>
    <target xml:lang="DE-DE">Optional Secondary Identifier</target>
   </trans-unit>
   <trans-unit id="3" maxbytes="46">
    <source xml:lang="EN-US">Program/Unit Identifier</source>
    <target xml:lang="DE-DE">Program/Unit Identifier</target>
   </trans-unit>
  </body>
 </file>
```

Figure 152: XLIFF file generated from .psd file with Rainbow

- ECM Engineering (see www.ecm-engineering.de) offers a tool that prepares .psd files in RTF formats which can be processed in tools such as *Trados*, *Déjà Vu*, or *SDLX*.
- Transmissions, LLC (see www.transmissionsllc.com/products) offers a product (as well as a service) that converts .psd files to XML or RTF files.

Vector-Based Formats

The above graphic types are pixel-based graphics. Another kind of graphic that is often used, especially in manuals, is vector-based graphics. You can recognize them by their typical extensions, .eps or .ai. They are very different from pixel-based graphics because they are formed by mathematical formulas rather than by simple dots. So, rather than displaying a wheel by arranging a lot of pixels in a circle, a vector-based graphic would calculate it with some kind of pi-based formula.

In contrast to pixel-based graphics, it is possible to translate most vector-based files directly in applications like *Adobe Illustrator* (see www.adobe.com/products/illustrator) or *Corel Draw* (see www.corel.com/coreldraw). Again, if you would like to either batch process the files or use your translation memory, both the above-mentioned ECM Engineering and Transmissions (see page 231) offer a product for *Illustrator* files that pre-processes the files for the use in computer-assisted translation tools similarly to the way it processes .psd files.

Tagged Formats

Tagged files are files that are text-based and that typically contain a mixture of "normal" translatable text and "tags," elements that allow for the structuring of the content, page layout, text formatting, insertion of images, etc. Examples of tagged files are the exported text-based formats for the translation of content in some desktop publishing programs (see the example on page 222), but more typically tagged formats include HTML, XML, or SGML files (see the definition on page 118).

Tags are typically enclosed in <angled brackets>. Internal tags, such as the <b>bold<\b> tag, are embedded in segments, whereas external tags, such as the <p>paragraph<\p> tag, are located outside sentences.

It is, however, also possible that tags themselves can contain translatable text. One well-known example is the alt tag for image files in HTML:

```
<img alt="translatable text" src="image.jpg">
```

Because tagged text files are "just" text files, they can be translated with a text editor. The reason why this is typically not a good idea is that

- the tags are quite sensitive to corruption, i.e., just deleting or adding a part of a tag may utterly corrupt a file;
- it takes a fairly experienced user to distinguish all instances of translatable text from non-translatable text; and
- though it would be possible to process tagged text files as plain text files in computer-assisted translation programs, it would mess up your translation memories with a lot of unwanted coding information; at the same time, you will not really benefit from your translation memory content because there will be very few matches for heavily coded sentences.

Instead, you should be using computer-assisted translation tools that support tagged text formats, and most of them do (the only major one that does not directly support HTML, XML, and SGML is *MultiTrans*). The concept of supporting these formats is to hide and/or protect any untranslatable information and only to display translatables.

This is relatively easy to do with HTML because it is a defined format that does not allow any deviation, but it is more difficult with XML and SGML files. These files are by definition user-definable and require you to "teach" the program how to interpret any give file. Any of these file types refers to a "Document Type Definition" or .dtd file that determines how each element of the file should be treated.

While the .dtd file for HTML is a global declaration that any of the supporting tools refer to, XML gives a somewhat universal access through a supporting technology that describes how to format or transform the data in an XML document, the so-called Extensible Stylesheet Language (XSL). Many translation environment tools offer a predefined XML filter based on a common set of XSL variables that is often sufficient to process XML files.

As SGML files have no such common denominator, you will need to create a specific "filter" (in *Déjà Vu* terminology) or "settings" (in *Trados* terminology) to process these files.

In *Trados*, tagged files are processed in the *TagEditor* (thus the name). Upon opening any of these file types in *TagEditor*, the following dialog may appear (if it is not displayed automatically, you can open it through **Tools> Tag Settings**):

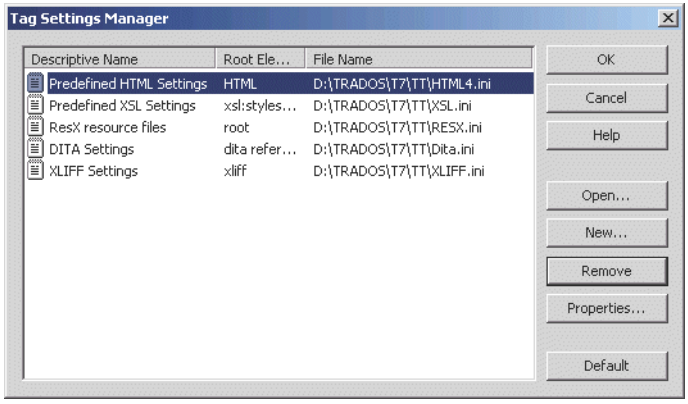


Figure 153: Tag Settings Manager in Trados TagEditor

You can see the predefined HTML and XSL (for XML) settings.

You can change the properties of any of these settings files by selecting **Edit** (something you should only do when you are unhappy with the result the existing settings files produce) or create a new settings file by selecting **Add**.

Other predefined settings in version 7 of TagEditor include those for XLIFF, DITA, and .resx files. One of the great advantages of XML is that it is ideally suited as an exchange format—among many others, it is the source format for the translation exchange formats TMX, TBX, and XLIFF *(see page 208).*

DITA (Darwin Information Typing Architecture) is a new XML-based standard for authoring, producing, and delivering technical information, and .resx files are XML-based resource files for .NET applications.

The creation of a new settings file is largely wizard-based so it is less complicated than it seems. You do, however, need the .dtd file to create the resulting settings .ini file, which in turn allows you to import your tagged text.

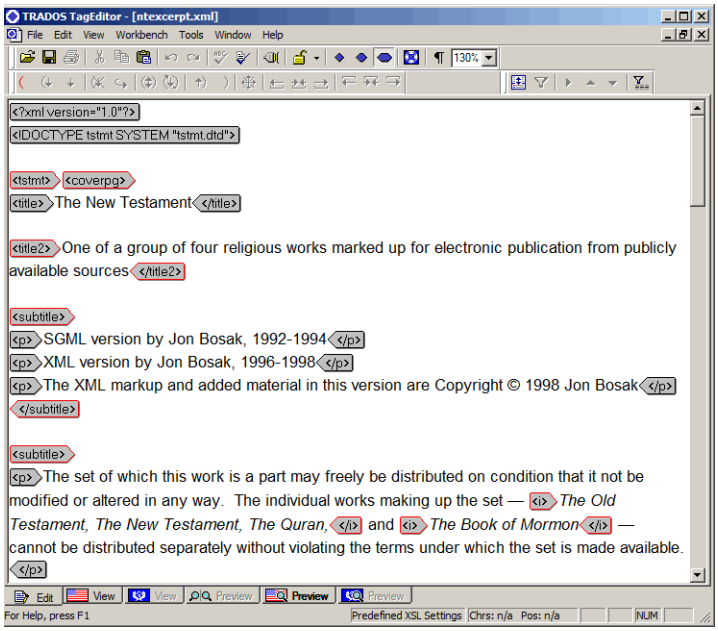


Figure 154: Imported XML file in Trados TagEditor

Déjà Vu also contains a predefined XML "filter" (available under `C:\Program Files\ATRIL\Deja Vu X\Templates\OpenOffice.dvflt`), but just like in Trados it allows you to either edit that existing filter or create filters for other SGML files. You can access this feature by selecting **File> New> SGML/XML**

Filter, and the wizard will lead you through the creation of a very customizable filter file. Unlike in *Trados*, it is possible to forego the import of a DTD file and you can choose to directly import an SGML or XML file to create a filter.

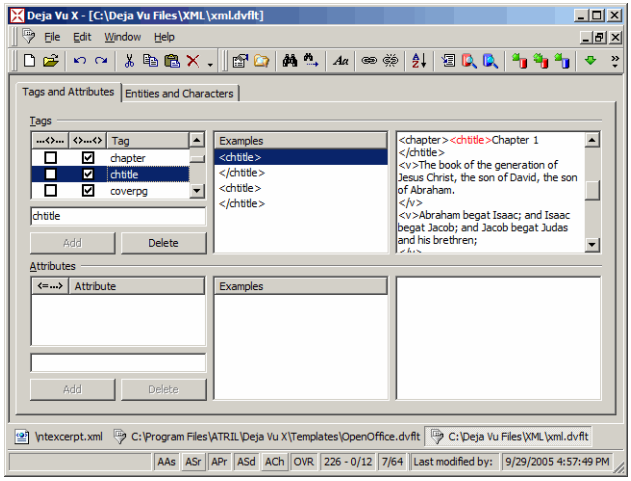


Figure 155: View of an XML filter in Déjà Vu X

As you import the XML or SGML file into *Déjà Vu*, you will need to make sure to select the appropriate SGML/XML filter file during the import process under **Properties**.

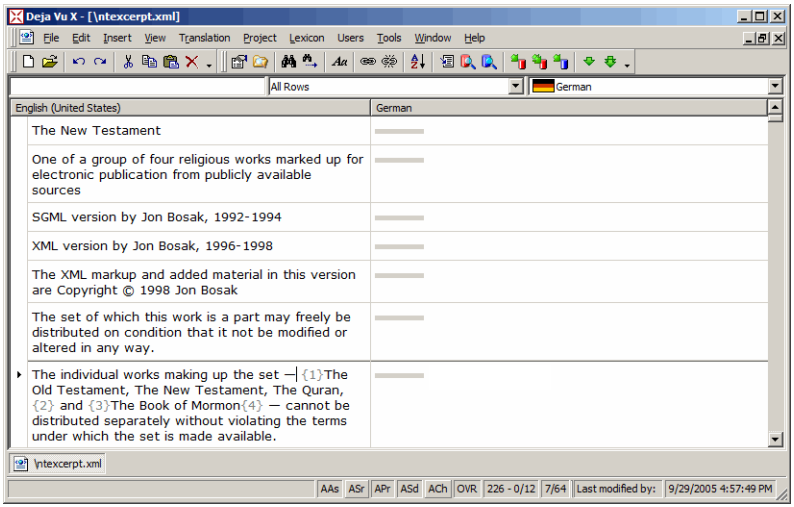


Figure 156: Imported XML file in Déjà Vu X

Most tools, including both *Déjà Vu* and *Trados*, allow the fine-tuning of the filters so that you can exactly determine which parts inside or outside a tag are translatable or to be protected. Typically, it is enough to go through the process of creating a filter or settings file for an XML/SGML project only once because usually all files will adhere to one standard.

Software Development Formats

There is a large variety of software development formats. The most logical distinction between the different categories is the following:

- binary files, i.e., files that cannot be opened and edited with a text editor, and
- flat files, i.e., text-based files that can be opened and edited with a text editor.

The binary files traditionally include formats such as .exe, .dll, or .ocx files. To translate these files, you will either

- need a specific software localization tool that allows the direct translation and necessary strings as well as further language-specific development work and testing (for further information on processing binary file formats, see *Software Localization Tools* on page 206) or
- need to "decompile" these files with a development tool into numerous individual flat files (and later "recompile" the binary file again).

While many software developers view the decompile/recompile process as outdated, others don't, and chances are that you will at some point receive decompiled RC files to translate.

```
//////////////////////////////////////////////////////////////////////////////
//
// Dialog
//

IDD_WEBWIZ_SIGNONDLG DIALOGEX 0, 0, 174, 103
STYLE DS_MODALFRAME | DS_CENTER | WS_POPUP | WS_CAPTION | WS_SYSMENU
EXSTYLE WS_EX_CONTEXTHELP
CAPTION "WebWizard Admin Signon"
FONT 8, "MS Sans Serif", 0, 0, 0x1
BEGIN
    RTEXT           "Enter &Userid:",IDC_STATIC,15,14,54,8,0,WS_EX_RIGHT
    EDITTEXT        IDC_USERID_EDIT,77,13,63,14,ES_UPPERCASE |
                    ES_AUTOHSCROLL
    RTEXT           "&Password:",IDC_STATIC,15,34,54,8,0,WS_EX_RIGHT
    EDITTEXT        IDC_PASSWORD_EDIT,77,33,63,14,ES_UPPERCASE | ES_PASSWORD |
                    ES_AUTOHSCROLL
    RTEXT           "&New Password:",IDC_STATIC,15,54,50,8,0,WS_EX_RIGHT
    EDITTEXT        IDC_NEWPASSWORD_EDIT,77,52,63,14,ES_UPPERCASE |
                    ES_PASSWORD | ES_AUTOHSCROLL
    DEFPUSHBUTTON   "&Sign On",IDOK,29,78,50,14
    PUSHBUTTON      "&Cancel",IDCANCEL,92,78,50,14
END
```

Figure 157: View of a decompiled RC resource file in a simple text editor

In much the same way that tagged files work, it would be possible to translate RC files in a text editor, but it is not advisable to do that, because a) you will most likely overlook text that needs to be translated, b) you may overwrite code where that should not happen, and c) there is just no reason to not use your translation memory for this. In fact, software files are rarely translated

on their own. Typically they are translated as a precursor to accompanying documentation—documentation that will be using references to the translated software over and over again—an ideal scenario for the use of translation environment technology!

Many computer-assisted translation tools support the translation of RC files, including *Déjà Vu*, *SDLX*, *Star Transit*, *across*, and *Trados*.

While the translation of RC files in Trados before version 7 used to be rather tedious, it can now be done directly in the TagEditor which even offers certain RC-specific quality control features, including check for duplicate hotkeys.

"Hotkeys" are the Alt key combinations that allow keyboard access to menus or dialog controls. In RC files they are displayed as ampersands (&) before the hotkey character.

Because the syntax of RC files uses the quotation mark as a functional character, it requires a duplicated quotation mark ("") for every linguistic quotation mark ("Click on ""Next"" to continue"). Some computer-assisted translation tools take that into consideration—others may not. If you are not sure, you can also use 'single' quotes which you do not need to double.

Many newer programming languages do not use a compiled format for their resource files. Often this takes the form of XML-based formats (see for instance the .resx example on page 235) that are more or less supported by all major translation environment tools, while in other instances other formats are used.

Java applications typically use the so-called Java Properties files (.properties).

```
# English-language strings for the Content Installer screens

contentTitle = Content Installer

reset = Reset

hiRes = <B>High Resolution Video and Illustrations</B> - \
Provides the highest quality images and full-motion video \
for the selected Titles, but requires the most disk space \
and the most time to download when used by clients.

loRes = <B>Low Resolution Video</B> - \
Provides full motion video for the selected Titles \
at a lower resolution which takes less disk space \
to store and less time to download when used by clients.

poster = <B>Poster Video</B> - \
Provides still images which illustrate the content \
of the selected Titles. This option takes less disk space \
```

Figure 158: View of a Java Properties file in a simple text editor

The structure of these files is relatively simple: typically everything that follows quotation marks needs to be translated, everything that is preceded by a hash mark (#) is to be excluded, and the translatable strings can contain some HTML-based tags.

Java Properties files are supported by *SDLX*, *OmegaT,* and *Déjà Vu* as well as several localization tools (see *Software Localization Tools* on page 206). *Trados* supports them as well but only through the tedious *T-Windows* application which is nothing short of frustrating to use.

Extensions are always a first indication of what the file type could be if you are not sure what format a certain software file is in but they will often fail you with software files. If you are not sure about the file type open it in a text editor and study the structure of the file. If translatables are enclosed with quotation marks, try to process the file as an RC file (or, in the case of Déjà Vu or SDLX, you can also test it with one of the other software filters). If the translatables are preceded by an equal sign, try to process them with the Properties filter. As all of these files are text-based, this will not damage the files and very often you will find that you "get lucky," even though the file at hand may not be one or the other.

Another emerging text-based software standard are GNU gettext .po and -pot files. These are the translatable language resource files used in the free GNU gettext concept for translating software and documentation. GNU gettext is the de facto standard in many open source projects, and it works with a large variety of programming languages. .po files are typically translated or pretranslated files, whereas .pot files are the translatable templates.

```
#: common/catalog.cpp:506
msgid ""
"Free Software Foundation Copyright does not contain any year. It will not be "
"updated."
msgstr ""
"El copyright de la Free Software Foundation no contine ningún año. No se "
"actualizará."

#: common/catalog.cpp:1743
msgid "loading file"
msgstr "cargando archivo"

#: common/catalog.cpp:2011
msgid "saving file"
msgstr "guardando archivo"

#: common/catalog.cpp:3351
msgid "searching matching message"
msgstr "buscando mensaje coincidente"

#: common/catalog.cpp:3646
msgid "preparing messages for diff"
msgstr "preparando mensajes para diff"

#: common/context.cpp:117
msgid "Corresponding source file not found"
msgstr "No se encontró el mensaje fuente correspondiente"
```

Figure 159: View of a .pot file in a simple text editor

Aside from the internal tools that gettext offers (see www.gnu.org/software/gettext), *Déjà Vu* seems to be the only translation environment tool that handles these files seamlessly.

Help Systems

Help systems—i.e., the documentation resource that is typically part of a software program and can be accessed through the help menu—is a huge topic on its own. I'm not planning to cover this in its entirety, especially because this has already been done so masterfully in Bert Esselink's "*A Practical Guide to Localization*" (John Benjamins, 2000). But there are a few questions that I have been confronted with over and over again, and here are some quick answers for those.

First of all, there is a great variety of help systems (*JavaHelp*, *OracleHelp*, *QuickHelp*), but the two most often-used help systems in the Windows world are *HTMLHelp* and the increasingly outdated *WinHelp*.

WinHelp

The compiled *WinHelp* system typically consists of two files, the .cnt file and the .hlp file. While the .cnt file is a text-based file that contains the table of contents for the help system, the .hlp file is a compiled file that is made up of any number of RTF files.

These RTF files have to follow strict guidelines as to how they are created so that hyperlinks, index markers, sections breaks, etc. function correctly. Most larger translation environment tools (especially those that have been around for a while and seen the heyday of WinHelp) have facilities to accommodate these special features (such as hidden text for hyperlinks or the various kinds of footnotes).

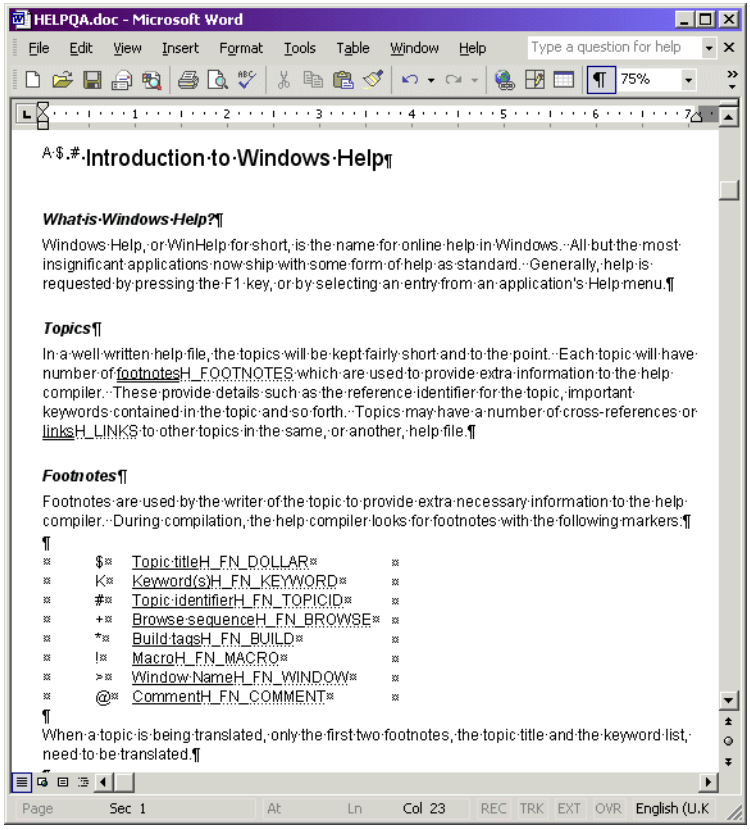
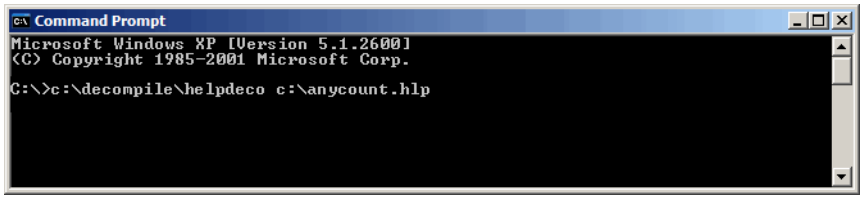


Figure 160: View of an RTF file before its compilation into a WinHelp

In case you receive a .cnt and .hlp file for quoting or even translation purposes, there's an easy way to "decompile" the .hlp file into its RTF components. While there are a number of expensive commercial tools for compiling and decompiling *WinHelps* (for instance, see the well-known *RoboHelp* at www.adobe.com/products/robohelp or RoboHelps rightful

successor Flare under www.madcapsoftware.com) under sourceforge.net/projects/helpdeco you can find the *HelpDeco* application which allows you to break apart your help file and analyze and translate the resulting RTF files (and typically any number of image files).

*The downside is that this is not a particularly user-friendly application. What you need to do to use it is to open a DOS window (**Start> Programs> Accessories> Command Prompt**) and point the HelpDeco application to the help file. So, assuming that you have placed the helpdeco.exe at C:\decompile and your help file anycount.hlp is located right at C:\ you would enter this:*



One file that is also created in the process is an .hpj file, the help project file. Though this file is not to be translated, it is important because it contains the information on how to re-compile the project once the translation is done. The free Microsoft program that can be used to do just that is called *Microsoft Help*

Workshop and can still be found at www.helpmaster.com/hlp-developmentaids-hcw403.htm (Microsoft itself is not distributing it anymore). All you need to do to re-compile—i.e., to recreate the .hlp file—is to open the .hlp file with *Help Workshop* and click **Save and Compile**.

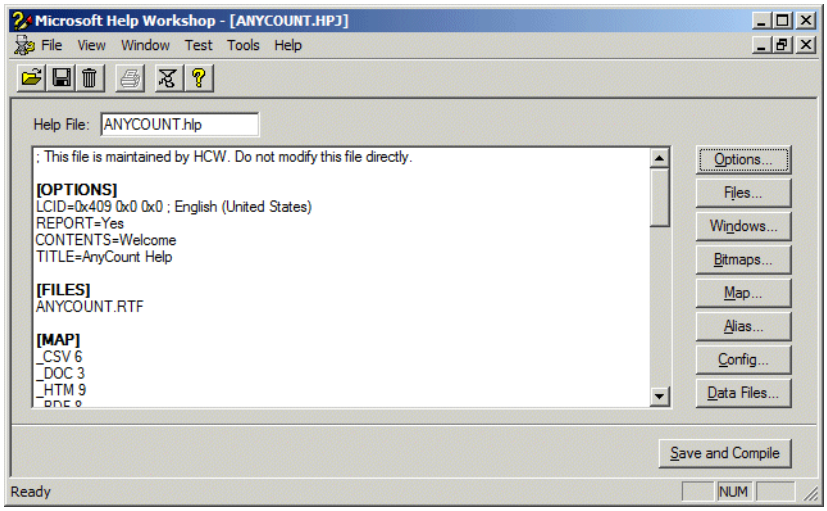


Figure 161: WinHelp compilation in Microsoft Help Workshop

HTMLHelp

The process for *HTMLHelp* is similar but much simpler. Unlike the *WinHelp* system, *HTMLHelp* only consists of one file, the .chm files. True to its name, most of the translatable content of an *HTMLHelp* system is contained in HTML files. To "get to" the HTML files, you will also need to decompile the help file. Fortunately, both the compilation and decompilation are done with the same freely available and easy-to-use tool: *HTML Help Workshop*.

You can download the latest version at msdn.microsoft.com/library/default.asp?url=/library/en-us/htmlhelp/html/hwMicrosoftHTMLHelpDownloads.asp.)

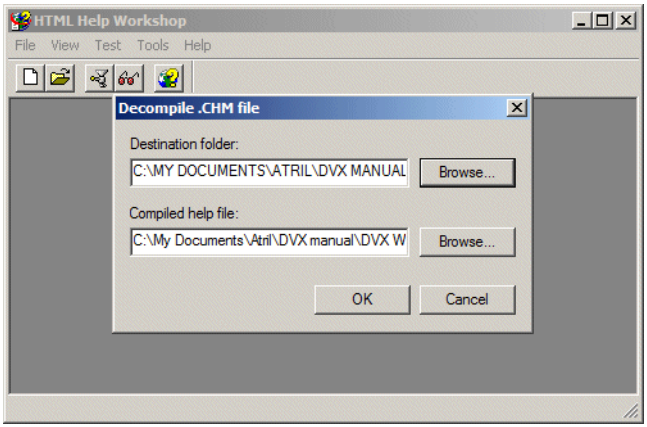


Figure 162: HTMLHelp decompilation in HTML Help Workshop

To decompile an existing help file, just select **File> Decompile**, locate the .chm file, and choose a location to which you would like to export the files. You could receive a great number of different file formats, but the most typical are:

- .hhp: the non-translatable project file (you will need this file to recompile the help),
- .hhc: the translatable (table of) contents file in HTML format,

- .hhk: the translatable index file in HTML format,
- graphic files: these are often translatable and/or have to be replaced with newly created target counter-parts, and
- lots and lots of .html files with lots and lots of translatable content.

Before you start with the translation of your *HTMLHelp* project, here is one thing you should be doing first: Talk to your client about the format in which the authoring of this project took place. Chances are that it was either authored in *FrameMaker* (like this present manual and help system), in some kind of XML form, or even within *Word*. While it is entirely possible and really quite easy to translate the *HTMLHelp* directly, your client may be much better served if you are able to work in the original format. Typically the original authoring environment is set up so that the output can be done in various formats (PDF, printed materials, web based, help systems, etc.), whereas it is much more complicated to do this when you start with a help system.

If your client asks you should translate the help system directly, translate the above-mentioned files, replace the graphics (save them under the same name and the same location), and then recompile the individual files with *HTML Help Workshop*.

Well, usually it doesn't work quite so easily, because chances are some link was corrupted, some graphic is missing, or some file was renamed. So it is advisable to do a quality assurance check which compares the original files and your newly translated files. SDL offers an excellent product for doing just that in *HtmlQA* (see www.sdl.com/products/htmlqa.htm).

The equally helpful sibling product for the WinHelp process is called HelpQA (see www.sdl.com/products/helpqa.htm).

Once you've fixed your errors, you can proceed with the compilation in *HTML Help Workshop*. Just select the .hhp file (make sure that it's placed at the root of your project folder), select **File> Compile**, and your help file will be all ready to go.

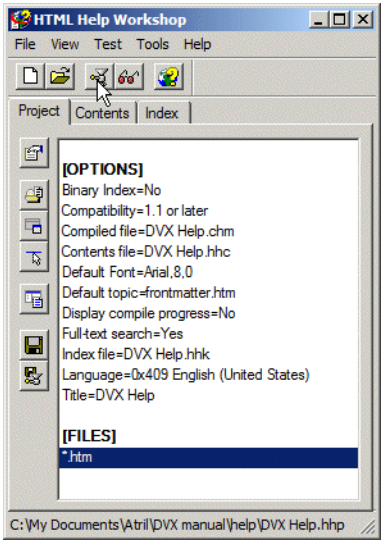


Figure 163: HTMLHelp compilation in HTML Help Workshop

You can also use HTML Help Workshop to convert existing WinHelp projects. When you convert a WinHelp project to an HTML Help project, the New Project Wizard converts the WinHelp project (.hpj) file to an HTML Help project (.hhp) file, the WinHelp topic (.rtf) files to HTML Help topic (.htm, .html) files, the WinHelp contents (.cnt) files to HTML Help contents (.hhc) files, and the WinHelp index to HTML Help index (.hhk) files.

Database-Based Data

It's a strange thing with data in databases. So much of today's translatable content is stored in databases for easy and quick user access (this is especially true for web-based content), but translators are often met with a bit of suspicion when it comes to the translation of that data. And there is probably something to that suspicion. Much like software development files

that developers often have a quasi-emotional attachment to, database administrators rightly so feel quite protective of their database content, translatable or not. As a result, data from databases is often exported to an exchange format such as .csv (comma-separated value file—a text-based file where data is separated by commas, see page 143) or *Excel* spreadsheets. While these formats typically do not provide many problems to be processed by translation environment tools, there are two significant drawbacks:

- depending on the database, the data may not only not have context but may also be concatenated (i.e., one string consists of many pieces that are not necessarily displayed together), thus making it very difficult for the translator to translate appropriately, and
- it represents a great deal of overhead for the database administrator to not only maintain but also to import and export data in various languages on a regular basis.

It is therefore not surprising that a number of tools have tried to come up with solutions to directly translate database content within the database environment. Though there are a great number of different database formats, there are also standards that allow the communication with almost all database formats. ODBC, Open Database Connectivity, is a native interface that allows access to most database management systems and allows for the use of SQL, Structured Query Language, the universally used language to "talk" to databases. By using this interface and this language, a number of computer-assisted translation tools are now able to directly translate database content, even from as complex an environment as *Oracle* or *MS SQL* databases.

Among localization tools, *Multilizer* (see www.multilizer.com) and *Alchemy Catalyst* (see www.alchemysoftware.ie) support the translation of database content; on the side of translation environment tools, the corporate edition of *Déjà Vu X* does.