AppleGlot

User's Guide

For AppleGlot version 1.0b8.1
International Software Support

Original Version by Yan ARROUYE

Table of Contents

About this User's Guide

This User's Guide is still under development as AppleGlot will continue to change until 1.0 is finalized. A more complete User's Guide will be released for the final 1.0 product.

What is Localization?

Localizing means adapting a file to a specific country and/or a specific language.

So localization may consist of:

- Translating program resources
- Translating documentation
- Adapting the file to country specific units of measure, currency sign, date formats, etc.
- Adapting the program to language specific writing rules
- Other file specific localization needs

Fortunately, Macintosh programs are (or should be!) written to minimize the efforts of localization. The Macintosh OS is built to provide both language independent procedures to aid in supporting different writing rules, and to give the application access to country specific symbols, formats, etc. Therefore localizing a Macintosh file will generally consist of just two distinct steps:

- Translating documentation
- Translating file resources

While translating documentation can be a lengthy process, it is a fairly straight forward step, typically using a single word processor. Translating resources however can be a cumbersome time consuming step in the localization process. AppleGlot has been designed to make the job of translating resources easier and more time efficient.

The Three Stages of Localizing File Resources

Localizing file resources is a three stage process:

Stage 1	Resource Text Translation
Stage 2	Adjusting Dialog Layout
Stage 3	Translation Checking

The first stage consists of translating each resource that contains text. For example a typical application file will need resources such as MENUs, DITLs, STRs and STR#s translated. Often many other resources will require translation as well. This stage is time consuming because there are generally many resources to translate and it takes a significant amount of time to ensure that text is translated in the same way, both throughout the file, and throughout different versions of the file.

The second stage is required because translating dialog items may change their length forcing a resizing of the original dialog layout. So one should check every dialog and eventually resize items according to the translated text's length. This could be easily done using ResEdit, or any other resource editor. Resource editors

generally let you both change dialogs items size and change dialog structures by adding, deleting or modifying dialog items. Because it is easy to make such major changes to resources this stage can introduce errors. This is why stage three is crucial in the resource localization process.

When the file has been translated, and dialogs have been adjusted, you must start the third stage process of checking your work to ensure the correctness of your localized file. You must make sure that the translated application's structure (especially dialog structure) has not been damaged during the first two steps. This is very time consuming since you should compare the original and the translated file structures item by item. Problems at this stage could of course send you back to an earlier stage. So the three stages of localizing resources are in fact a cycle that is repeated until stage three tells you the process is complete.

How AppleGlot Localizes Resources

AppleGlot handles all three stages of the translation process described in the previous section.

Currently, AppleGlot can translate all resources that can be defined with ResEdit templates. All the typically translated resource types (MENU's, DITL's, STR's, etc.) already have ResEdit templates, so in general you won't have to worry about creating your own. AppleGlot will contain most all the templates you will need. If you have come across a custom resource that AppleGlot has no previous template definition for, you can easily include your own templates file for AppleGlot's use.

Note that templates internal to AppleGlot may be customized for AppleGlot’s use, and so are not interchangeable with ResEdit templates. If you do use external template files, make sure they are not ones used internally by AppleGlot already.

AppleGlot operates by placing text found in the translatable resources into text files. You can then use your favorite word processor to perform the required translations. This allows the user to concentrate on translation and not get involved in the technical details and the dangers inherent when using resource editors and development environments.

When you are localizing an application, the application will either be "new", or it will be a "revision". AppleGlot takes both cases into consideration. For a new application, besides just pulling out resource text, AppleGlot allows the use of a Language Glossary. A Language Glossary is basically a lookup table of possible

translations you have created previously. The translations in the glossary are used as guesses by AppleGlot when exact matches are found in the original application. For a revised application, AppleGlot can make use of the Language Glossary, but also and more importantly it can make use of previously translated files. Once you have translated text for version 1.0 of your application, for version 2.0 you will only be required to translate new or modified resource text items.

AppleGlot thus can be used for initial (1.0) releases, and subsequent (2.0..) releases. When used for a subsequent release, resources that have not changed between the 1.0 and 2.0 versions, are directly copied over to the translated target file so you don't have to worry about work you had already completed during the translation of version 1.0.

To help you in completing stage 2 of localizing resources, AppleGlot has a built-in dialog editor. This is where you view your dialogs and alerts and resize items where appropriate. No editing of the text is allowed at this point. All text translation is handled by editing text files.

Finally, AppleGlot helps you check your translated file structure by writing both a history of how the translation was done, and any error diagnostics found to a separate text file. This "history" file, currently called "H-'xxx'" where "xxx" is the name of your translated file, is a very important file to check to see is any problems were found while AppleGlot was translating. Errors should never be introduced by AppleGlot itself, but if you use other resource editors on your translated file, it is possible for you to produce errors. AppleGlot can also verify your translated file by checking for missing resources in either the original or the translated file, missing items in resources, and mismatched resource names.

Now let's go into detail on how AppleGlot implements each stage of the resource localization process.

Stage 1: Resource Translation

AppleGlot is a double-clickable application. As with most Macintosh applications, you can open it from the Finder in one of two ways:

- by selecting AppleGlot application icon and double-clicking or choosing Open in the File menu, or

- by selecting one or more AppleGlot document icons and double-clicking or choosing Open in the File menu.

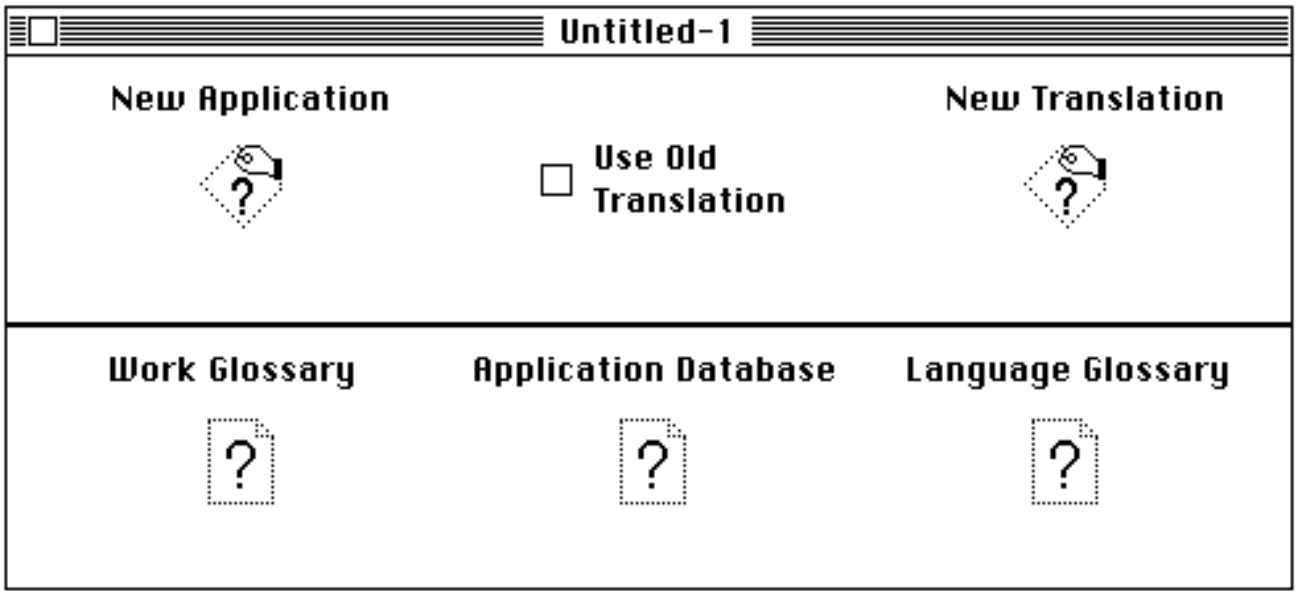
In the first case AppleGlot starts and opens an empty untitled document. In the second case the selected documents are opened.

Using the Translation Set Window

AppleGlot uses many files to translate a file. The set of files you use are accessed through a Translation Set window displaying icons containing question marks. When you click on one of these file icons, a Standard File dialog box is displayed. This dialog box allows you to create or open the file the icon refers to. You can also save Translation Set files to easily save the names of the files you are currently working with.

There are two input modes for the Translation Set window. One mode is for use when you have only version 1.0 of a file, and no previous existing translations. The second mode is for situations where you are translating a 2.0 or later version with a prior version having been already translated.

Let's first look at the window where you are performing a first time translation:



New Application : The file to be translated

New Translation : The translated file that you will use AppleGlot to create.

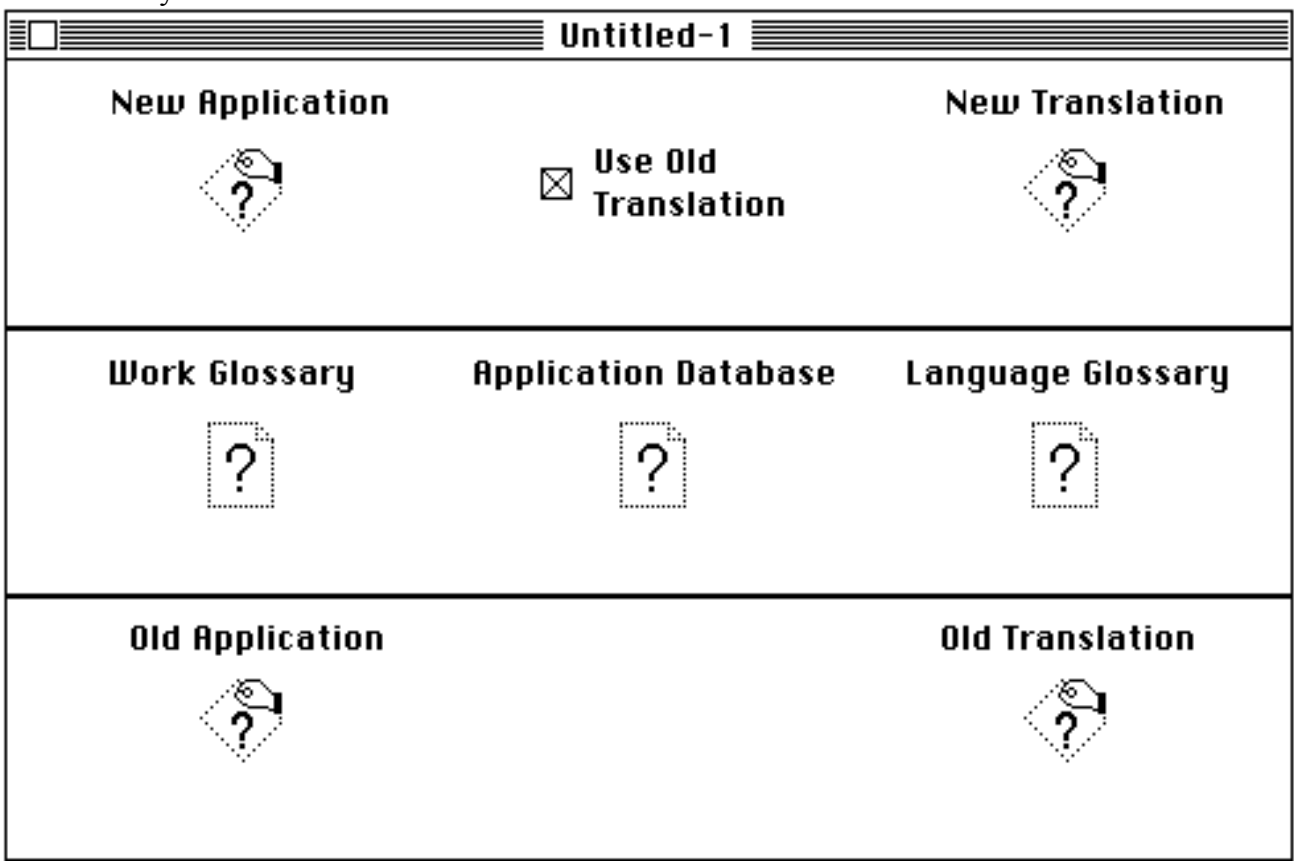
Work Glossary : This file will be used by AppleGlot to list the items it can't translate without your help. You'll have to provide the missing translations before complete translation can be accomplished. You will start off creating a new Work Glossary when you first begin your translation.

Application Database : This glossary is updated by AppleGlot after every translation and eventually will contain all translated items with their translations. You will start off creating a new Application Database when you first begin your translation. You typically will not be editing this glossary.

Language Glossary : The purpose of the Language Glossary is to fill in translations for common terms. This can be especially useful for translations of new applications with no prior localized versions. A Language Glossary that contains translations for "File", "New", "Cut", "Copy" etc. can save you from having to re-translate these terms with each new application. The more terms AppleGlot can find from the Language Glossary means less translation work for you! It is up to you to create this glossary.

It is important to note that only the Language Glossary is not a required input. The New Application, New Translation, Work Glossary, and the Application Database must all be filled in before a translation can begin.

Now let's look at the situation where you wish to perform a translation where a previously translated version of the application exists. You must check the "Use Old Translation" check box to gain access to the old file versions that AppleGlot will make use of. The Translation Set window will look like the following when you check this box:



Old Application : This is the previous original version of the application.

Old Translation : This is the previous translated version of the application.

These files are used to determine which resources haven't been modified since the previous version. AppleGlot copies these unmodified resources from the previously translated version directly into the new translated version. This will save you from re-translating resources that have not changed from version to version.

In this mode, again only the Language Glossary is not a required input. The New Application, New Translation, Work Glossary, Application Database, Old

Application, and Old Translation must all be filled in before a translation can begin.

The File Menu

The File menu lets you create, open and save Translation Sets.

Note that the file menu deals only with Translation Set files, and *not* with the files referenced by the Translation Set window itself. Therefore if you relocate one of the files used by one of your translation set files, AppleGlot will be forced to prompt you for the new location of the file since only the names and paths to the files are stored in the Translation Set file and not the files themselves.

The Translate Item

The Translation menu is the central menu of importance in AppleGlot. The items in this menu are set active and available to you only when you have reached the appropriate stage of your translation.

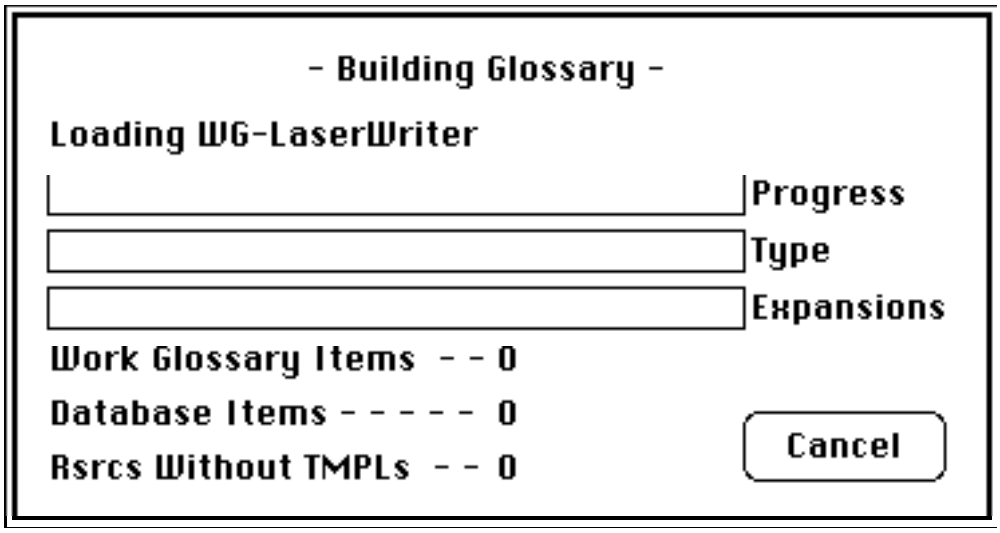
The Translate item is what you will use to translate your file. Only after your Translation Set Window has been properly filled out will this menu item become available for use. Remember that all file entries in the Translation Set Window must be filled out except for the optional Language Glossary.

You will need to select the Translate item at least twice in order to complete a translation. In general, "pass one" of Translate will just build the Work Glossary; only resources not needing translation will be written at this time to the file you are translating. You then edit the Work Glossary to fill in your translations. MultiFinder is quite useful here to switch to your word processor while AppleGlot

is still open. Once you have completed editing your Work Glossary, you must switch back to AppleGlot and select the Translate menu item once again. This "pass two" Translate execution pass will copy over your Work Glossary translations to the translated file and will also copy the information from your Work Glossary to your Application Database.

In the situation where you already have information in your Application Database (such might happen when performing a translation using previously translated files), and all text needing translation is already available in the exact context needed, AppleGlot will not pause to ask you to fill out the Work Glossary since there is nothing to do. In this unusual case, AppleGlot would create the translated file straight away.

During a translation you will see the following progress alert:



This dialog monitors both the translation and verification operations to give the user a visual indication that AppleGlot is really trying to do something and not just sitting there displaying the watch cursor in an infinite loop.

The progress dialog consists of state information display lines, progress indicator bars, and useless information counters that are mostly for your amusement while you wait for a large file to be processed.

The upper status line announces AppleGlots current activity or state in a general sort of way. It tells if AppleGlot is loading files, translating or verifying, etc.

The second status line just below it (just above the progress bars) tries to give more detail on what is going on. During translation this second line contains information about the resource which is currently being worked on.

The top progress bar indicates in a general way the total amount of progress AppleGlot has made in completing the entire translation or verification it is performing. It is only a general indication and not in any way an indication of the time spent or remaining due to the fact that it is a simple counter of the total

number of resources to be translated with a fudge factor to account for some other operations that it knows it will need to do. If there are 99 tiny resources and 1 gigantic one this progress bar will go almost to the end very quickly and then take much longer to go that last tiny increment than it took to go the first 99. There is no way to tell how complex a resource is by simply looking at its size since most fields are variable. A STR# resource could consist of a couple of very large strings or dozens of tiny ones. Only by expanding it can we see what's inside. To the right of this bar is the word 'Progress'.

The middle progress bar gives an indication of how we are doing within a single TYPE of resource. For example if AppleGlot has counted the number of 'CODE' resources then this bar indicates our progress in processing just these 'CODE' resources. AppleGlot may then move on to 'STR ' resources and the indicator will then apply to the 'STR ' resources as they are processed. To the right of this bar is a number which is the total of the number of resources in the type being processed. In addition to this typical usage the middle bar is often used to indicate the progress of some other operation which may take some time, such as when the Work Glossary and Application Database files are being read into memory from the disk. In these cases it is simply an indication that something is happening and will give a fairly accurate representation of the relative amount of time the remainder of the operation will take.

The bottom progress bar is used when AppleGlot is packing the expanded resource into a single block to be written back to the New-Loc file. During the first translation pass this bar sweeps twice as it first processes the expanded resource to isolate text it already has translations for (and can write to the Application Database) from text it doesn't have translations for (and will be put into the Work Glossary). To the right of the bar is a number which indicates the number of expansions of the resource being worked on. Since the number of expansions isn't known until the process is complete we can't use the bar itself during expansion but when the expansions are complete we can use the bar to tell us how the packing is proceeding when we are reassembling the resource. You may notice that during an expansion of a very complex resource which may have many hundreds of pieces that the counter to the right of the bottom bar gets slower and slower. This is due to the memory manager getting overloaded and is the reason that the SpeedZone was created. Some large files with previous translations can put up to 40000 non-relocatable blocks into AppleGlots heap so AppleGlot tries to isolate its expanding and packing from these blocks by doing its work in a separate heap zone. Even this separate zone will slow down when a single resource puts thousands of blocks into it. Fortunately these large resources are not the rule and AppleGlot speeds right along but some STR# resources can get fairly big and bog things down.

Some operations (such as reading and writing the glossary files) may not utilize the middle or bottom progress indicators. Other operations may use them in different ways but generally they are only supposed to give the user a sense that AppleGlot is busy and not stuck in a loop and wasting their time.

At the bottom of the progress dialog are three lines which total the number of items which are being placed into the Work Glossary or Application Database or being simply copied into the New-Loc file. Feel free to become depressed if the count of items which are being placed into the Work Glossary climbs into the hundreds or thousands. It's usually much easier when there is a previous translation to use because AppleGlot will try to preserve as much of your work as possible. Unfortunately you will have to do the work of translating the Work Glossary yourself the first time (or farm it out if you're lucky).

In addition there is a 'Cancel' button. When this button is enabled you may click it to abort the operation currently underway. This cancelling takes place as soon as the process reaches a point where it is safe for AppleGlot to stop the operation.

After a translation pass, you will see the following alert:

The purpose of this alert is to give you "state" information about the progress of your translation.

Translation Status:
This area will tell you one of two possible messages:

(1) "Translation of your file is not complete"

(2) "Translation of your file is complete. You must now 'Verify Translation'.

Message (1) means you need to translate again before your translated file is

completed.

Message (2) means you have completed creating your translated file.

Work Glossary Status:
This area will tell you one of five possible messages:

(1) "Text requiring translation has been found. You must now edit the Work Glossary and 'Translate' again."

(2) "The Work Glossary contains items with no translation."

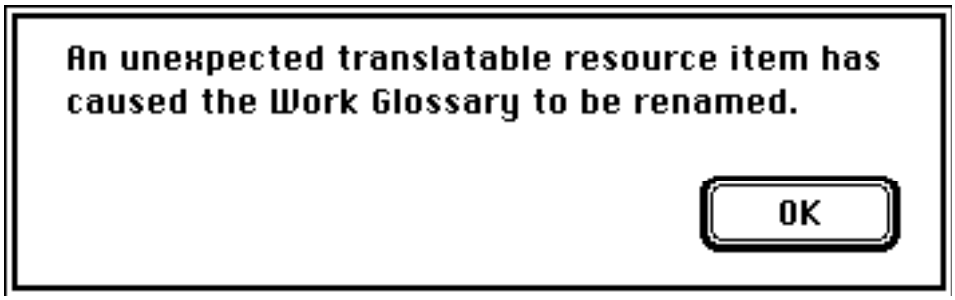
(3) "AppleGlot found no text at all."

(4) "The Work Glossary is empty. No text has been found requiring translation."

(5) "The Work Glossary is fully translated."

Message (1) will be displayed when AppleGlot writes out a new Work Glossary. You will need to fill in missing translations using your word processor. Once you have done this, you must select the 'Translate' menu item again to have AppleGlot move your translations to the Application Database, and to the translated file. Typically, you will only see this message after your 1st translation pass. But you may also see this if your glossaries already exist, and AppleGlot finds a text item in the new original file that is unaccounted for in your current glossaries. Whenever glossaries exist, and AppleGlot finds an unaccounted for item in the new original file, the Work Glossary is re-written with all such unaccounted for items.

Note: When AppleGlot re-writes the Work Glossary, you will be able to find a copy of your original Work Glossary with a numeric suffix in its original folder location. AppleGlot will display the following alert:



This alert will be recorded in the history file. It is also a "timed" alert. Some alerts will "time-out" after a period of time so the translation can continue if you happen not to be at you computer when such an alert appeared.

Message (2) is displayed if your translation is complete, but some items in your Work Glossary have not been translated. This could be just fine since you may may be taking advantage of AppleGlot's "forced translation" feature that will

simply be a write through of the original text item to the translated file. For example, some text cannot be translated when a program looks for precisely those characters.

Message (3) means no text was found (using existing templates) to translate from the new original file.

Message (4) means that all text needing translation has been accounted for in the Application Database and no Work Glossary items were found requiring translation.

Message (5) means that AppleGlot has read in the Work Glossary and all items were found with their corresponding translations.

History File Status:

This area will tell you one of three possible messages:

(1) "The history file contains warnings!"

(2) "The history file contains fatal warnings!"

(3) "The history file is normal."

(4) "The history file was turned off and nothing was written to it."

These messages are fairly self-explanatory. Basically, this area will tell you if the history file is "normal" (no warnings found), or will tell you warnings were issued. The only "fatal" errors are when AppleGlot cannot create files due to lack of disk space. ***Remember that it is your responsibility to judge whether or not any warnings issued by AppleGlot threaten the validity of your translated file!***

How AppleGlot's Glossaries Work

Before a full understanding of how AppleGlot translates a file you must understand how AppleGlot uses the Work Glossary, Application Database, and the Language Glossary in detail.

The Work Glossary, Application Database, and Language Glossary are text files that can be created or edited by any text processor that supports a "Save As Text" option. 'MPS' is now the default creator type for both the glossary text files and the history file, so MPW will be launched upon double-clicking in the Finder instead of AppleGlot. This is a bit non-standard, but probably more practical. Using ResEdit, the user can change this creator type (to their favorite word processor, for example) in STR 1185 for the history file, and STR 1186 for the glossaries if MPW is not your editor of choice. If you still want AppleGlot to be

launched, AppleGlot's creator type is "Yan2".

Glossaries can either be created by AppleGlot or directly by the user. AppleGlot creates and/or updates both the Work Glossary and the Application Database, but not the Language Glossary. This glossary is left to be created/updated by the user.

The glossary format has been designed to be both simple and flexible. A glossary can contain **translation paragraphs** that contain the original text and possibly the text's translation, and optional **context setting paragraphs** that are used to associate the text to particular types of resources. This format is used for all of AppleGlot's glossaries: the Work Glossary, Application Database, and Language Glossary.

Translation Paragraphs

A translation paragraph defines the translation of an original string. The first string is the original string, and the second string is the translation. Both strings are enclosed in angle brackets.

For example

```
<Apple>
<Pomme>
```

is a translation paragraph that defines the translation of the word "Apple" in french.

If the translation string is empty, AppleGlot assumes that the translation is the same as the original string. So if you want to tell AppleGlot that "OK" translates to "OK" you may write either

```
<OK>
<OK>
```

or

```
<OK>
<>
```

If you need to use a '>' sign in a string you have to type it twice ('>>') to avoid confusion with the string endmarker. When AppleGlot writes strings it uses the same convention to represent the '>' sign if found in an original item's text.

For example "this string encloses a SINGLE > bracket" should be written

```
<this string encloses a SINGLE >> bracket>
```

The > bracket is also used in >ddd expressions where ddd is a three digit number

used to represent a character whose ASCII code is less than 32. This is necessary to keep word processors from complaining about non-printable characters. For example, a string that contains a null character (ASCII code zero) will be written with >000 instead of the null character. This is usually the case when translating menus: style is coded as a char and has often a value less than 32.

Examples:

<>012>	is a 1-char string that contains ASCII char 12
<This is a null:>000>	null ended string

WARNING: when using >ddd notation you must provide **3** digits even if the code fits on less digit.

<>12>	is illegal !!!
<>012>	must be used instead

Context Setting Paragraphs

Context setting paragraphs are used to restrict the use of translation paragraphs to some specified resources.

For example a translation may be valid:

- in any resource,
- only in resources of a given type (e.g. MENU)
- only in a given resource (e.g. MENU 3)
- only in a given item type of a given resource type (e.g. as title of a MENU resource),
- only if a given item of a given resource or resource type (e.g. 3rd field of MENU resources or 3rd field of MENU 3).

Context setting paragraphs let you specify one or more translation paragraphs for any of these combinations.

The process of extracting the data fields from a resource is called "resource expansion". A resource's template allows AppleGlot to expand the resource into its individual fields. These fields are then stored internally for individual access.

Note that the text between a set of angle brackets (<xxx>) in your glossaries is one of many fields that compose a resource. AppleGlot will extract and copy to the Work Glossary only fields containing text. Other field types such as rect's, bitmap's, etc., will be copied to the translated file automatically. See the section on "How AppleGlot Handles Resources" for more details.

The format of a context setting paragraph is:

resource type [resource ID] [expansion field type [expansion field sequence number]

Information inside brackets ('[]') is optional.

Resource type : Any valid resource type (a four character sequence such as STR# etc.

Resource ID : The identifying number used to specify a particular resource type.

Expansion field type : This corresponds to the name of the data field. This name is taken from the resource's corresponding template resource. This field should not contain numbers to avoid confusion with the expansion field sequence number.

Expansion field sequence number : Each data field extracted during the resource expansion process is assigned by AppleGlot a sequential number. This number is used by AppleGlot to index into the expanded resource fields. Do not confuse the expansion field sequence number with a ResEdit Item Number! ResEdit items correspond to logical objects such as buttons, controls, etc. A button itself contains more than one data field. It is these individual data fields that AppleGlot numbers for its internal use.

Context setting paragraphs apply to all translation paragraphs that directly follow it sequentially in the file. Once a context appears in a file, it is used to associate all following translation paragraphs until another context setting paragraph is read further on in the file. Translation paragraphs that you wish to use as generic translations regardless of context must appear before any context setting paragraphs in the glossary.

If you want a translation paragraph to be used only when translating MENU resources, you should insert the context setting paragraph:

MENU

To restrict to MENU resource ID 3, insert:

MENU 3

To restrict to menu items only (i.e. not titles) insert:

MENU MENU ITEM

To restrict to the 1st menu item of each menu insert you must look at a previously expanded MENU. Analyzing your expanded MENU from a Work Glossary, you can determine that the 1st menu item is referred to by AppleGlot as expansion field sequence number 9. So your context would look like:

MENU MENU ITEM 9

To restrict to the 1st menu item of MENU resource ID 3 insert:

MENU 3 MENU ITEM 9

It is important to note that "MENU ITEM 9" is **not** the 9th menu item in a menu resource. Yes this is confusing, but you must simply learn that this number is the resource expansion field sequence number. This number is a generic indexing number into **all** the fields that compose the resource.

Glossary Priorities

Once you understand what the context and translation paragraphs are, you must learn how AppleGlot uses the glossaries during the translation process. When you first start your file translation process, you start off by creating a new (empty) Work Glossary. The Work Glossary is so named because it usually will be the **only** file that you will be editing during the translation process. When a translation pass is initiated, AppleGlot will pull each text field from the original file and first look to the Application Database and/or the Language Glossary for help in translating each of those text fields. If no help is available from the Application Database or Language Glossary for a particular text field, an empty (<>) translation is written to the Work Glossary. When the translation pass has completed (unsuccessfully), you must edit the Work Glossary to fill in your translations.

It is important to note that if you do not translate a text item and leave the brackets empty (<>), the next time you execute the Translate menu item, AppleGlot will "force" the original text across to the translated file. This means that for those items that should remain in the original language, no editing work is necessary. This forced translation ability also allows you to "finish" and verify the translated file if for some reason you wanted to run your translated file and your text translations were not yet completed. AppleGlot will allow you to re-edit the Work Glossary, and execute the Translate item at any time.

When a match is looked for from the Application Database or the Language Glossary, the corresponding context associated with the text field becomes important in how AppleGlot handles the translation. Since it is quite possible that the same text string will appear in different resources in an file, these identical text strings would be in different contexts in AppleGlot's glossaries. In order to get the "closest" match for any given text field, AppleGlot will search for a translation repetitively with a less restrictive context by ignoring one or more of the context fields specified in the glossaries.

Context fields are ignored in the following order:

- first the **expansion field sequence number** is ignored,
- then if search fails again the **resource ID** is ignored too,
- then if necessary the **expansion field type** specification is disabled,
- then if still nothing found the **resource type** is ignored (i.e. no context is used).

When AppleGlot is building the Work Glossary, the Language Glossary is being used as a "suggested translations" file. Translation matches that reside in the Language Glossary are **never** directly copied to the translated file. If a match is found, whether the context for the translation paragraph is exact or not, the translation paragraph from the Language Glossary is copied to the Work Glossary. When the translation pass is complete, AppleGlot will prompt you to check your Work Glossary to verify that the translation taken from the Language Glossary is indeed the translation you want.

When the Work Glossary is being created from scratch by AppleGlot, it looks to the Application Database for translation matches. If an inexact context match is found, the translation paragraph from the Application Database is copied to the Work Glossary. AppleGlot will prompt you to check your Work Glossary to verify that the translation taken from the Application Database is indeed the translation you want. Exact context matches present in the Application Database are assumed to be correct translations and so are written directly to the translated file. When you have successfully completed a translation, the Application Database will contain **all** context paragraphs and corresponding translation paragraphs for all the translatable fields in the original file.

In summary, the only time a context paragraph is **not** created in the Work Glossary on a translation pass for a translatable field, is when an exact context match already exists in the Application Database. Exact and inexact context matches for translatable fields in the Language Glossary, and inexact matches from the Application Database are always interpreted as **suggested translations**. Suggested translations are placed in the Work Glossary for your verification.

If a translation paragraph with the same original text strings exists in both the Application Database and the Language Glossary, it is the translation paragraph from the Application Database that is used over the translation paragraph from the Language Glossary. This is true even if the Language Glossary has an exact context match, and the Application Database has the translation paragraph in a completely unrelated context paragraph.

Application Database Case Mismatches

A careful user may notice that the Application Database can appear to be making letter casing mistakes. The letter casing of original and translated strings on a

"forced translation" in the Application Database sometimes don't match!

The confusion lies in the fact that the original string in the newly created Work Glossary *will* maintain case, whereas the original string in the Application Database *may not.* This is not a bug. Your forced translations will in fact be case correct when they are written to the translated file. This is a program "feature" (that should be changed at some time in the future).

It is not the case of the original string in the Application Database that is important to AppleGlot internally (though it sure would be nice for the user!). It is the case of the translation that is important since that is what will be written to the translated file.

The reason the original strings do not retain the case information is for memory saving reasons. Since the original string is not needed by the translated file, all occurrences of a particular string, no matter how the string is cased is mapped to one string.

Language Glossary Example

A sample French to English glossary is listed below :

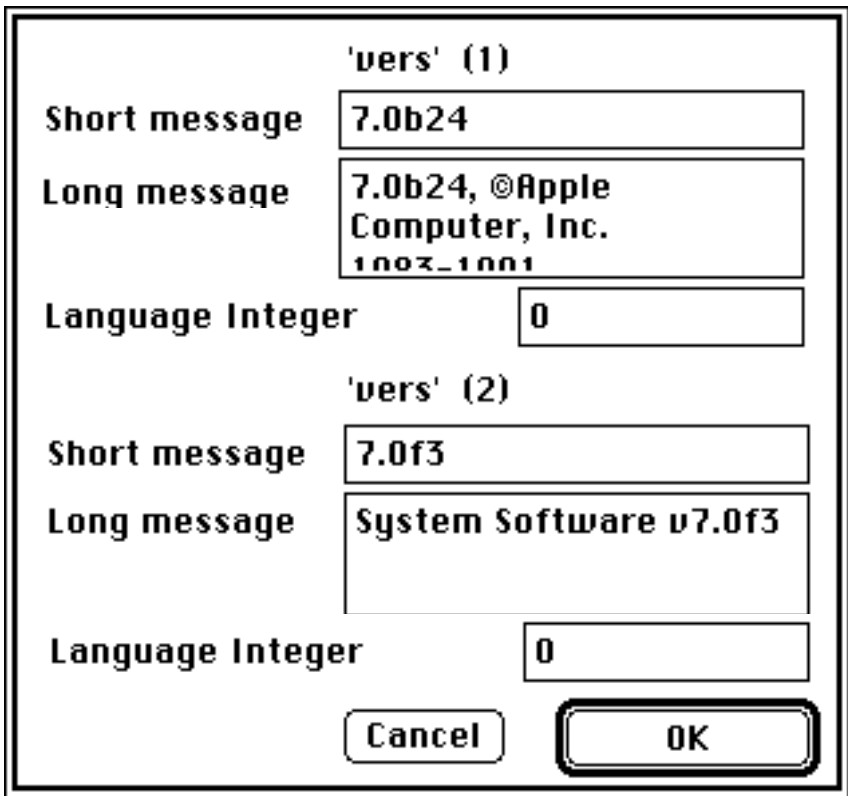
```
        <pomme>           <apple>     ! Always valid
        <bonjour>         <hello>
        <salut !>         <hi !>
        <je m'appele Yan> <my name is Yan>

MENU                                  ! Restrict to MENUs
        <Annuler>         <Undo>

DITL    dialog item text              ! Restrict to dialog
                                ! items
        <Annuler>         <Cancel>    ! (for example buttons)
        <OK>              <>          ! OK translates to OK

DITL 3                          ! DITL 3 only
        <Annuler>         <No>        ! Special translation
        <OK>              <Yes>       ! for DITL #3
```

Cloning Files

If a sysheap attribute bit is set in any resource within any input file in the Translation Set window, this file will now be duplicated. This is necessary since when the sysheap bit is on, the resource manager will load the resource into the system heap and not the application's heap. This is a problem because the system heap is relatively small. This problem caused some of the Finder and System file

translations in the 7.0 Workshop to crash. This does add time, and requires more disk space, but it is a safe solution to the problem. Cloned files are given the names of 'Glot.Vxx.Temp' where 'xx' can be VO1(standing for the old original file), VO2(standing for the new original file), VT1(standing for the old translated file), and VT2(standing for the new translated file). If you should ever see these clone files after you are through using AppleGlot, you may delete them.

Editing the New Translated Application's Version Information



If the "New Application" file you are translating contains a "vers" resource, then once the translation is complete, the version number of the translated file is displayed and underlined below the translated application's file icon in the Translation Set Window. To edit the vers resources, click on the underlined version string. A dialog box lets you change both the short and long version messages as well as the Language Integer field. Other version information is duplicated from the original file to ensure version consistency across original and translated applications.

Resource Movement During Translation

The power of AppleGlot rests with its ability to extract text from your original file and copy the text to a text file (glossaries). The text is then taken from the glossaries and placed in the new translated file. But how the resources with no AppleGlot templates, and the non-text resource items of resources that do have templates are dealt with is transparent to you the user.

In this section, an overall psuedo coded explanation will try to clear up the details

on exactly how AppleGlot processes and creates the resources that make up the final translated file. It is important to note that what is presented is a summary of the methodology used in the 1.0a10 version of AppleGlot, and that the algorithm will be changing until the final 1.0 release.

The psuedo code will contain some abbreviations that are defined as follows:

'V'	represents an entire file or "V"ersion of a file
'R'	represents an individual Resource within a file
'I'	represents the expanded Item from within the resource.
'T'	stands for Translated
'O'	stands for Original
'1'	is the previous version, both original and translation.
'2'	is the current version, both the original and translation.
'WG'	is the Work Glossary.
'AD'	is the Application Database.
'LG'	is the Language Glossary.

So the full abbreviations used mean the following:

VO1 is the old original version.
VO2 is the new original version.
VT1 is the old translated version.
VT2 is the new translated version.
RO1 is an entire resource from within the old original file.
RO2 is an entire resource from within the new original file.
RT1 is an entire resource from within the old translated file.
RT2 is an entire resource from within the new translated file.
IO1 is an individual expanded item from within an old original resource.
IO2 is an individual expanded item from within a new original resource.
IT1 is an individual expanded item from within an old translated resource.
IT2 is an individual expanded item from within a new translated resource.

How AppleGlot Works at a Glance

This simplified explanation assumes previous versions of the files are used.

Note that RO2 is expanded and takes components from RT2 or RT1. Then this modified RO2 will be written to VT2.

```
REPEAT FOR ALL RESOURCES IN THE ORIGINAL FILE
    IF A TMPL EXISTS FOR THAT RESOURCE
        IF THE WG AND AD ARE EMPTY THEN
            CREATE THE WG & AD    ; This will preserve text from RT1 if
                                  ; IO1=IO2.
        ELSE                                ; Else translate the templated resource.
            EXPAND RO1, RT1, RO2
            IF RO1 = RO2 THEN
                COPY RT1 TO VT2
            ELSE
                EXPAND RO2, RT1, RT2   ; Else translate RO2
```

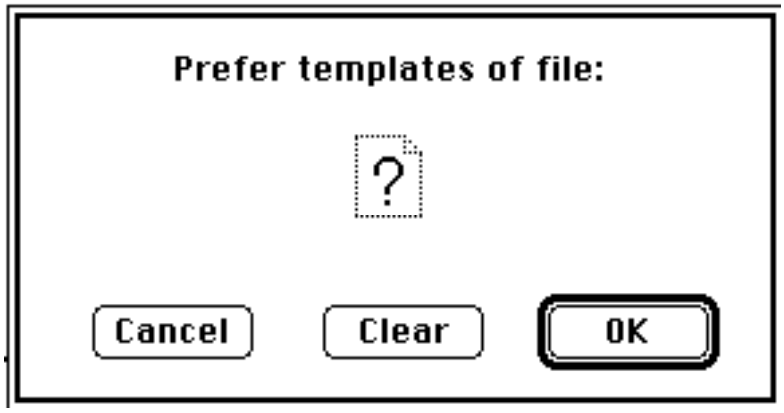
```
                IF RT2 EXISTS                ; if resource already in trans. file
                    IF ANY R01 AND RO2 NON-TEXT ITEMS MATCH THEN
                        MOVE CORRESPONDING IT2 ITEMS INTO RO2
                ELSE
                    IF ANY R01 AND RO2 NON-TEXT ITEMS MATCH THEN
                        MOVE CORRESPONDING IT1 ITEMS INTO RO2
                FOR ALL TEXT ITEMS IN RO2
                    IF TEXT ITEMS MATCH TEXT IN WG OR IN AD THEN
                        SUBSTITUTE TEXT INTO RO2
                    ELSE
                        WRITE ITEM TO WG        ; Note this will overwrite
                                                ; existing WG.
                COMPRESS RO2
                COPY RO2 TO VT2
    ELSE                                    ; No template exists
        IF RO1 = RO2 THEN
            IF RT1 EXISTS THEN
                COPY RT1 TO VT2
        ELSE
            COPY RO2 TO VT2
END REPEAT
```

The Options Menu

Set templates file…

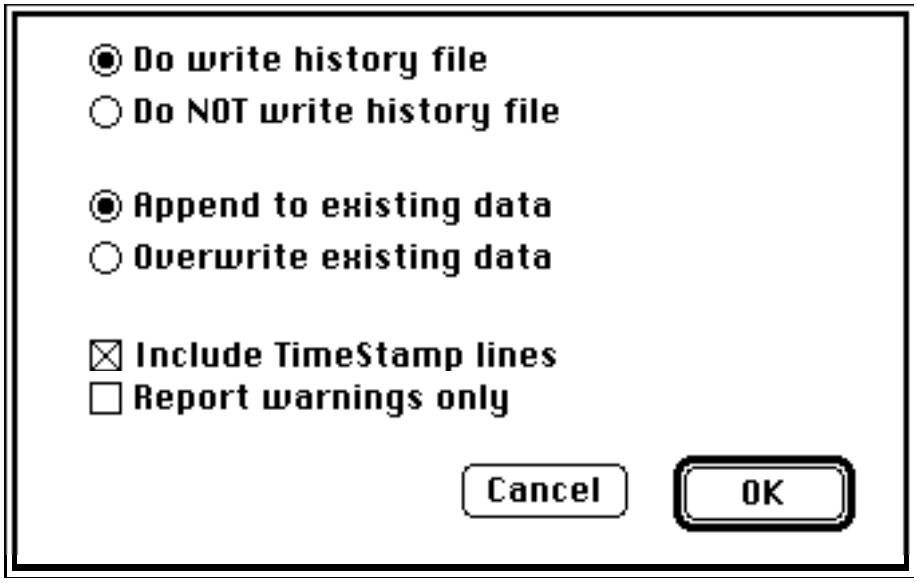
AppleGlot "knows" most of the standard resource types: STR , STR#, MENU, DLOG, DITL, ALRT, etc. AppleGlot understands the contents of these resources by using (possibly modified) ResEdit templates (TMPL resources) stored within AppleGlot's own resource fork. When AppleGlot translates a resource, it uses the item data types within the resources' corresponding template to find out what items in the resources can be translated.

Your file may contain custom resources that have no corresponding templates in AppleGlot. Any resource that has no corresponding template will not be translated by AppleGlot. Therefore you must supply templates for any custom resources in your file that you wish to have translated. This is done by providing to AppleGlot an external template file that contains templates for these custom resource types. The Set templates file... in the Option menu will let you choose the template file by clicking on the file icon as shown in the following dialog:



If a particular template type exists in both inside AppleGlot, and in the user supplied templates file, AppleGlot will always choose to use the user supplied template.

History...



The history file is a intended to be a log of AppleGlot's actions to allow the user to verify what AppleGlot has done. This history file is always named H-"new translation file name" (example: the history file name for a translation of Excel 2.2 with the new translation name of Trans of Excel 2.2 would be H-"Trans of Excel 2.2" (including quotation marks)).

Selecting the "History..." item from the Options menu will bring up a dialog with the following functions:

DO/DO NOT WRITE HISTORY FILE - enables or disables the history file function. Default is Do write history file.

APPEND TO/OVERWRITE EXISTING DATA - if there is an existing history file with the appropriate name and append is selected, all output of the history file will be added to the end of the existing file. If overwrite is selected

and there is an appropriately-named history file already present, its contents will be erased prior to new contents being written. Default is append.

INCLUDE TIMESTAMP LINES - If this item is selected, the time and date of the translation will be written into the history file prior to each translation or verification pass. This option is present mostly for testing purposes. Default is enabled.

REPORT WARNINGS ONLY - If selected, AppleGlot will list only those lines that correspond to a generated warning. Default is disabled.

The history file itself is an MPW-type text file. Its format is as follows:
If the TimeStamp option is enabled, there will be two header lines bearing time information for each translation or verification pass.
The body of the history file is made up of one line of information for each resource in the translation. An example (condensed and shown on three lines because of its length) is given below:

```
        •       'DITL' 130      New-Loc created using New-      US and
AD (or empty string),   Yes TMPL        "Preferences"
        230     240     ---     PURGEABLE---    ---     ---     ---
```

In order, the meaning of the items on the line above are:

•	The bullet indicates an entry has been made to the Work Glossary or Application Database. If an entry has been made in both the Work Glossary and Application Database, there will be two bullets here like this : ••
‘DITL’	This indicates the resource type
130	This is the resource ID number
New-Loc...	Indicates where the translated items originated (old trans or any of the glossaries)
Yes TMPL	Indicates there exists a ResEdit TMPL for this resource type in the templates file
“Preferences”	This is the name of the resource (if any)
230	The size of the original resource
240	The size of the translated resource
---, etc.	Any resource bits (system, purgeable, preload, etc.) which are set

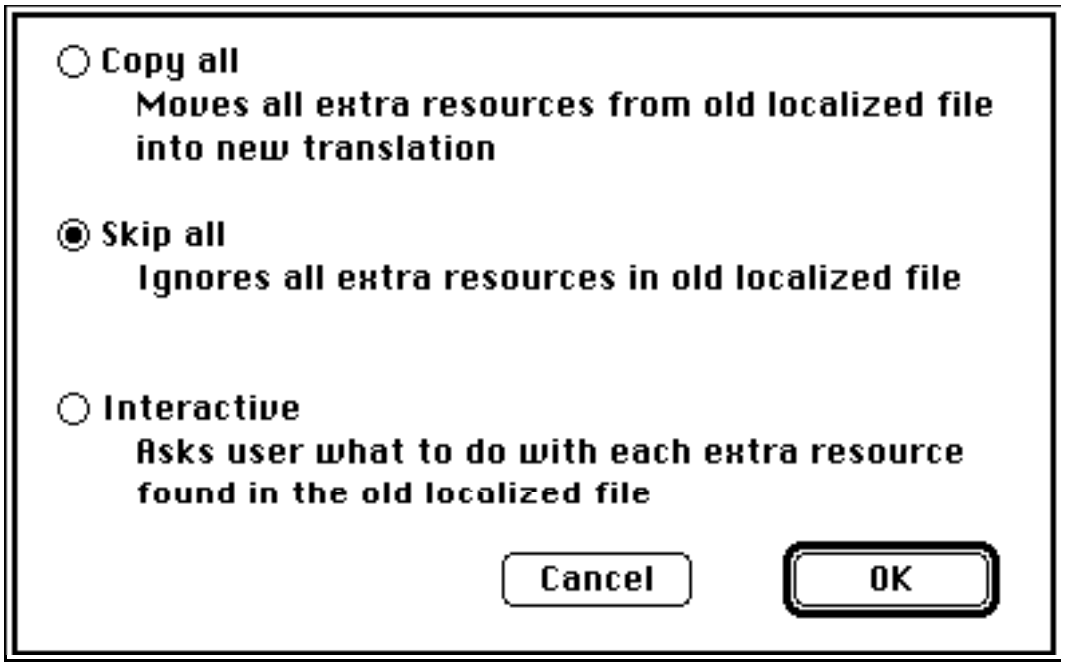
If AppleGlot comes across a questionable condition, a warning is generated in the history file like the following:

```
                        ?? Resource 'EXTEND' bit is set in MacPaint
        'DITL'  128     New-Loc made from New-US and AD (or empty
string),        Yes TMPL        "Brushes"
32      32      ---     PURGEABLE---    ---     ---     EXTEND
```

The warning is associated with DITL 128. A blank line separates the error line from the previous (non-error) line. In this case, the warning that the EXTEND bit is set (normally this bit should not be set, except in certain System 7.0 system files). Note that all warning lines begin with a double question-mark, and that a search for "??" will find all warnings.

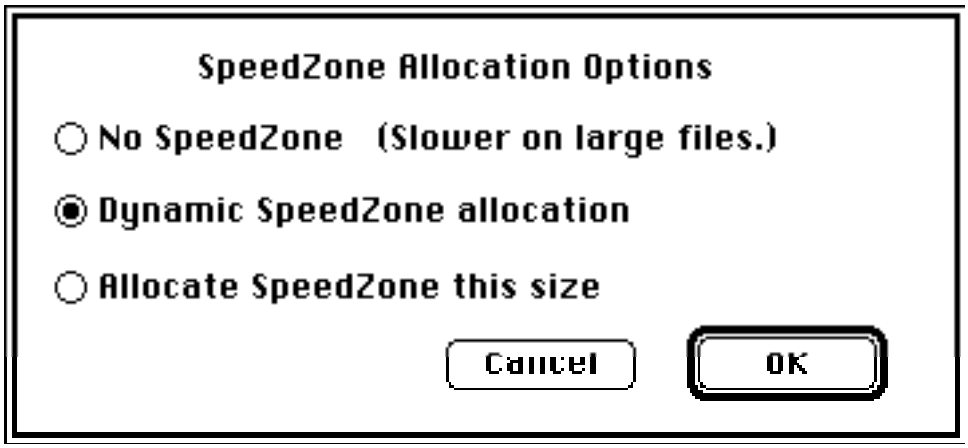
Please refer to Appendix C for a listing of all error messages that can appear in the history file.

Extra Resources...



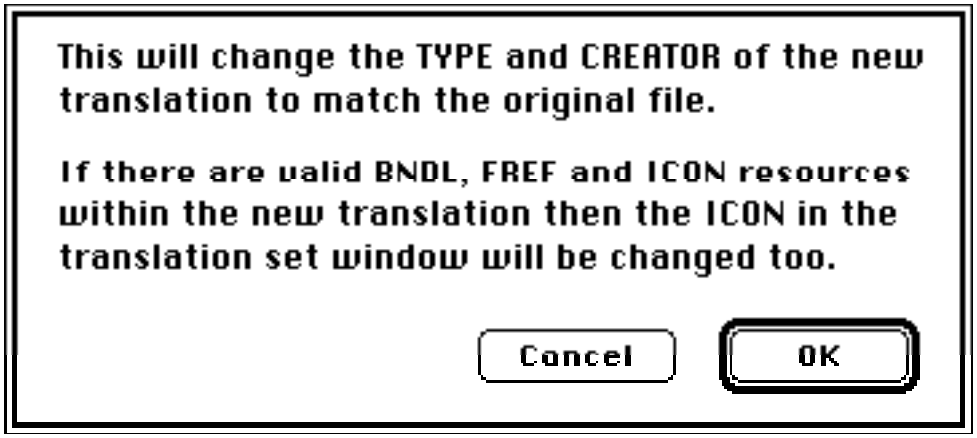
Allows the user to control the copying of extra resources present in the old localized file, and not present in the new original file. The user can specify 'Ignore All', 'Copy All' and 'Interactive' to choose which resources to copy.

SpeedZone...



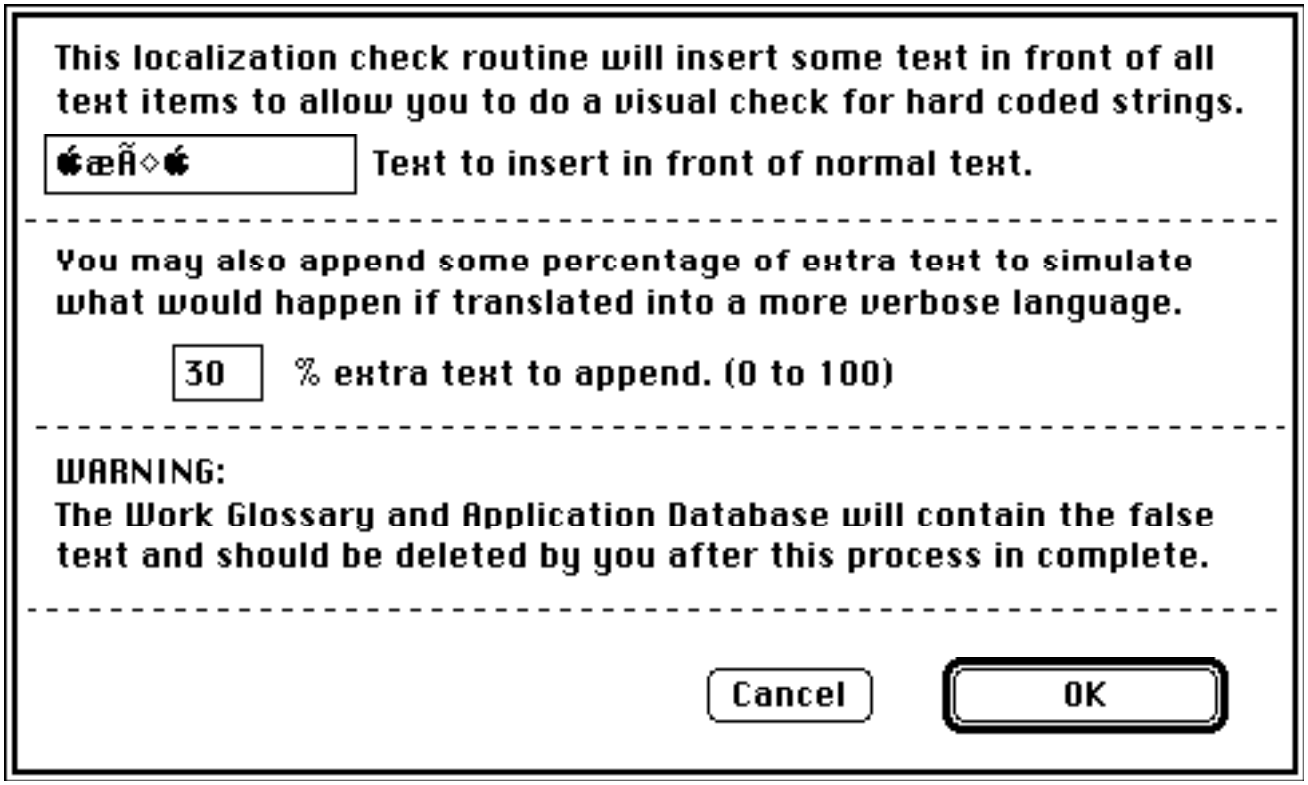
In order to speed up resource expansion a temporary zone is created within AppleGlot's own heap to isolate the memory manager from all the handles and pointers associated with the BTrees. It is not unusual to approach 40000 blocks allocated in AppleGlots heap when doing translations of some of our larger test file combinations. With this many handles and pointers to contend with the memory manager slows down dramatically. Note: The SpeedZone heap is created only if a rather crude calculation determines that the heap will leave a certain number of free bytes in AppleGlots heap. 'kSpeedZoneLeftOver' is the STR resource holding this value so the user can change this if necessary. In addition the calculation is based on the arbitrary assumption that no resource will need more than a fixed number of expansions. 'kSpeedZoneVal1' is the STR resource containing this fixed expansion value so the user has control over this as well. If there isn't enough room to allocate the calculated size block for the SpeedZone then the SpeedZone isn't allocated and translation proceeds with expansions taking place in AppleGlots heap zone at a slower speed which is dependent on the number of blocks being handled by the memory manager. The user can control the SpeedZone in three different ways: 1 - No SpeedZone at all; 2 -Normal dynamic sizing; 3 - Direct assignment of the zone size. Any of these options may become the default if desired.

Force CREATOR & TYPE...



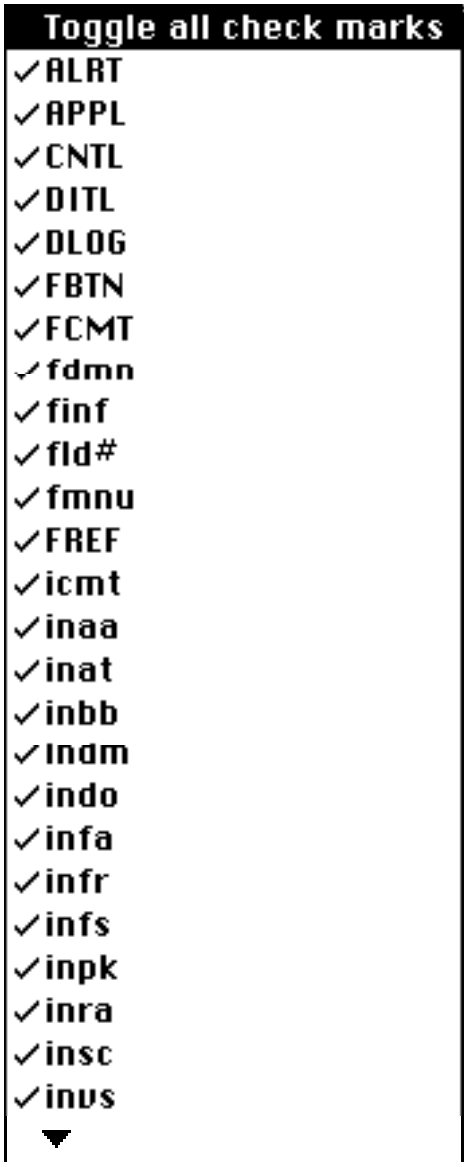
This item allows the user to copy the TYPE and CREATOR from the new original file to the new translated file without verifying. This is not a recommended procedure, and is typically not ever needed, but it is there for the experienced users who think they know what they are doing.

Localization test



This is a "Localizability" testing item that is very useful in catching hard coded strings and string concatenations. It will insert a string (as shown in the above figure) in front of most translatable strings as well as adding some percentage of text to the original text to simulate translation into languages such as German or Polish in which the strings are 30 or 40 percent longer when translated. The maximum extra percentage is 100. Both the prefixed string, and the appended strings are in the 'GetXXXDITL' resource so the user may modify them if different defaults are needed.

TMPLs list



The "TMPLs list" item is a hierarchical menu item that will list AppleGlot's templates contained within its own resource fork. This list does not contain any templates you added by using the "Set templates file..." option. The following templates are supported by AppleGlot 1.0b8: ALRT, APPL,CNTL, DITL, DLOG, FBTN, FCMT, FREF, insc, MACS, MENU, minf, PAPA, PRC3, PSAP, STR#, TEXT, WIND, mst#, mstr, STR, finf, icmt, inbb, inra, infa, infs, inpk, indm, LWSC, pgsz, LWRR, inaa, indo, fld#, invs, infr, inat, fmnu, and fdmn.

A check mark by the template name means that this particular template is "active", so AppleGlot will extract text from this type of resource. If you uncheck a template, AppleGlot will not extract text from that resource type. A 'Toggle all check marks' item reverses the check marks from on to off or from off to on.

Note that templates internal to AppleGlot may have been customized for AppleGlot’s use, and so are not interchangeable with ResEdit templates. If you do use external template files, make sure they are not ones used internally by AppleGlot already.

Stage 2: Adjusting Dialog Layout

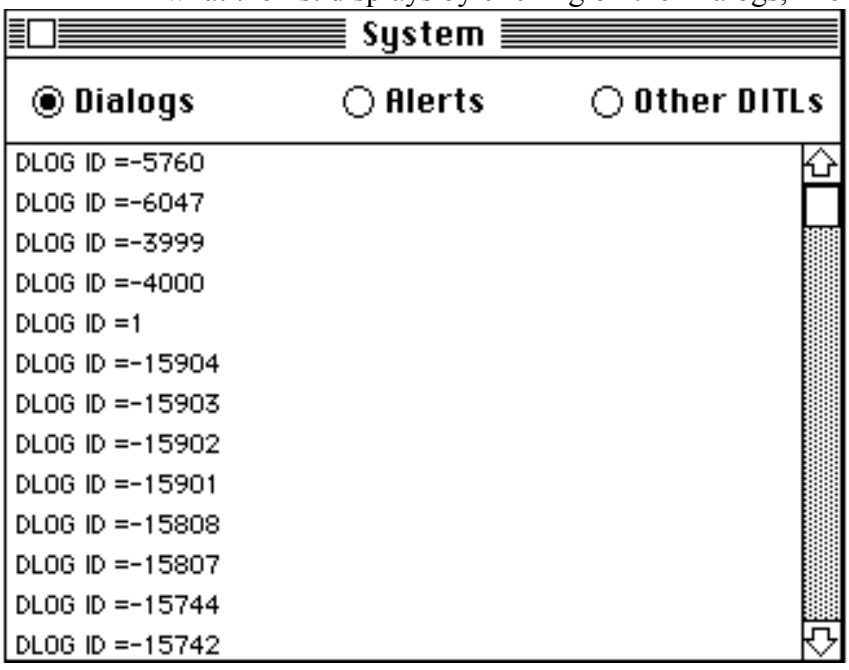
The Edit Menu

The Edit Menu is currently not used by AppleGlot in the translation phase. In the DLOG editor, only Undo Selection and Redo Selection are supported when selecting rectangles. Cut, Copy, Paste, and Clear are not used at any time.

The Adjust dialogs Item

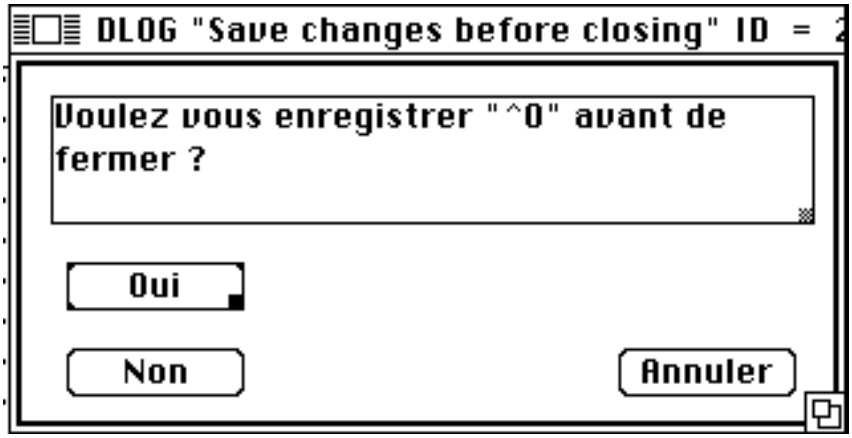
The Adjust dialogs menu item is available when a new translated file is present. You should adjust your dialogs after a successful translation.

Selecting Adjust dialogs opens a resource selector window displaying a list of file dialogs, alerts or unrelated DITLs from the new translated file. You may select what the list displays by clicking on the Dialogs, Alerts and Other DITLs buttons.



To edit a resource, double click on the resource name from the list. An editor window will open and let you adjust the layout.

Editor windows look like the following:

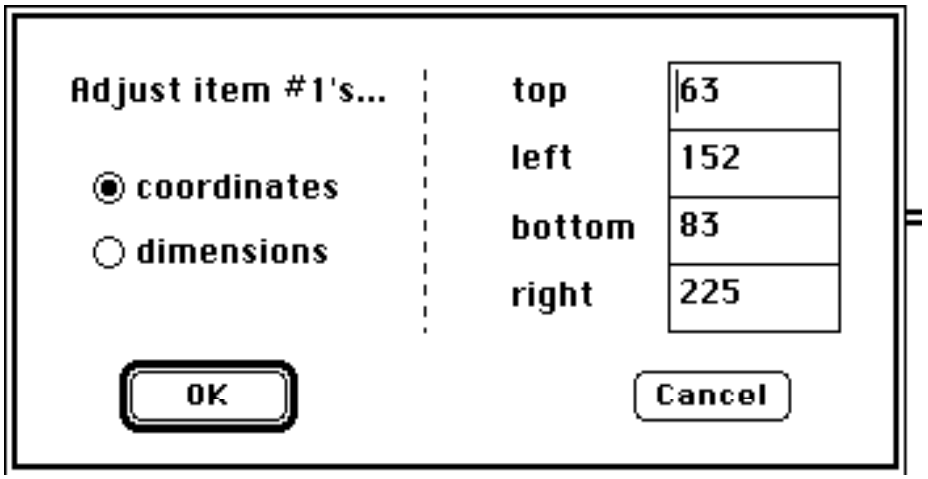


When editing a dialog or an alert, resizing or moving the window will resize or move the dialog/alert on the screen. If you edit an unrelated DITL (Other DITLs button) resizing or moving it has no effect.

You may select an item by clicking on it. The selected item is framed and a black size box appears. You can drag it around or resize it using the size box.

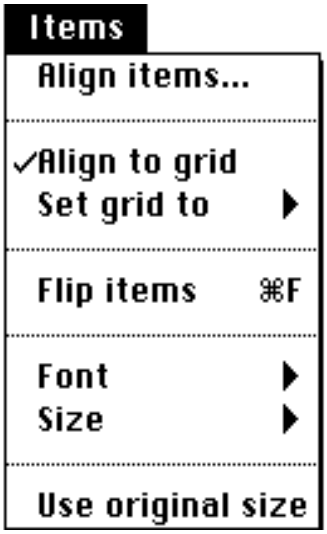
You also may select more than one item by holding the shift key while selecting items. Selected items are framed and have gray size boxes except for the last selected item which has a black size box. The black size box item is known as the "anchor item". There is always an anchor item. Usually the anchor item is the last selected. you can change anchor item just by clicking on the selected item you want to be the anchor. If you deselect the anchor item, another selected item will become the anchor.

Double-clicking on a item opens a dialog box that lets you adjust item coordinates or dimensions more precisely:



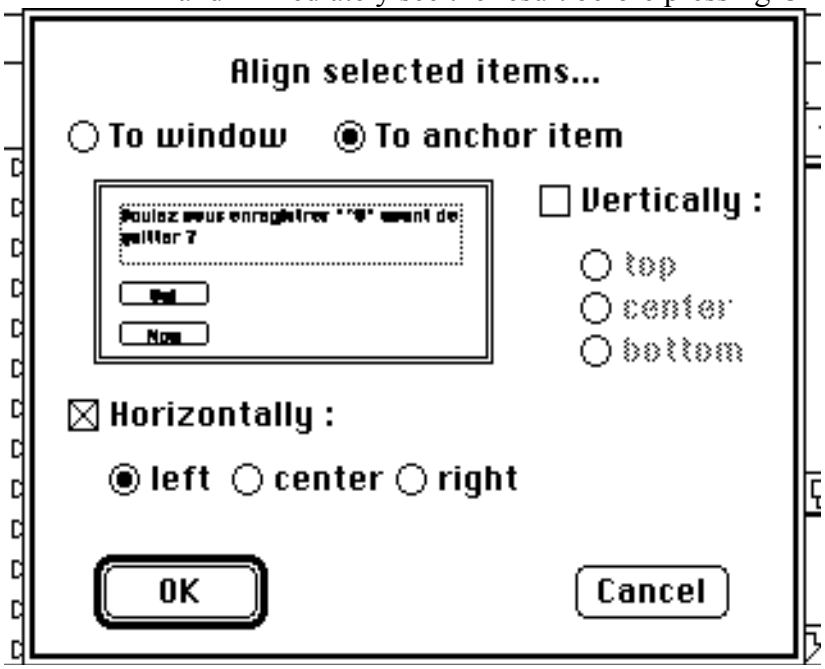
This may be useful to make sure that all buttons have the same dimensions.

When a resource editor window is in front, a new Items menu is added.



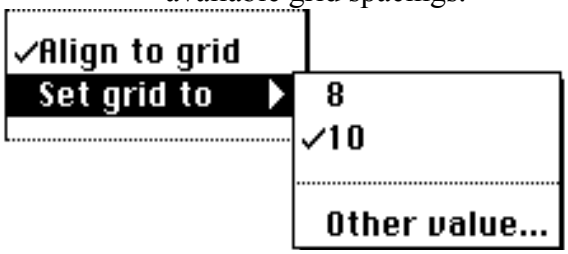
Align items... will display a dialog box that will let you align selected items in different ways.

The To window button lets you align selected items to a window frame. This option is always available. The To anchor item option is available only if more than one item is selected. In this case selected items are aligned to the anchor item (the one with a black size box) which won't move. Remember that you may change the anchor item by clicking on any selected item. The anchor item lets you control alignment in a very precise way. Checking Horizontally or Vertically controls whether horizontal or vertical alignment will be done. The picture below shows left sides alignment only. A reduced view of the edited dialog appears below the Towindow button. This view shows how items will be aligned according to current settings. This is useful since you may try many alignments and immediately see the result before pressing OK.

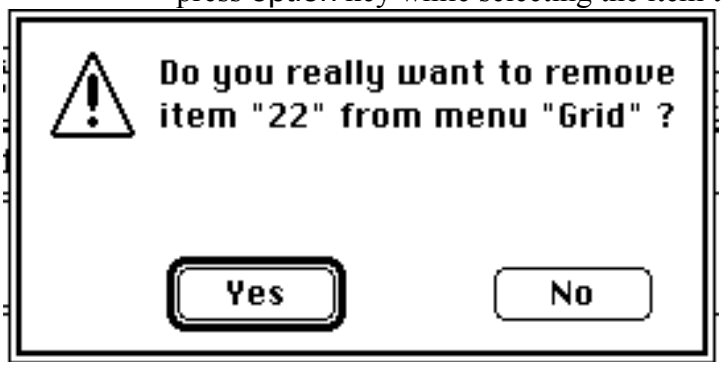


Align to grid helps you align dialog items by restricting them to an invisible grid. This option is set by default. You may want to deselect it to nudge items. The grid spacing is controlled by the Set grid to menu item.

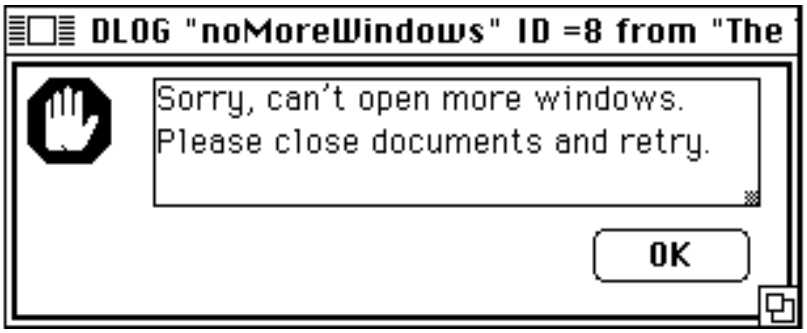
Selecting the Set grid to menu item opens a hierarchical menu displaying the available grid spacings.



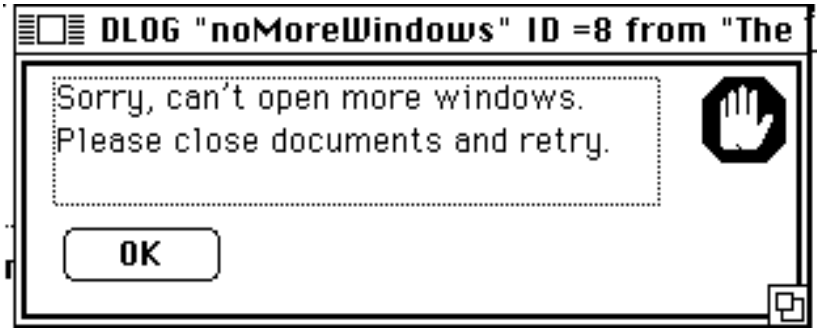
The standard spacing is 8 or 10 pixels. This makes buttons look best. If you want to use another spacing, select the Other value... menu item. This allows you to define your own custom spacing. Custom spacings values are added to the list so they are available for later use. To remove a custom spacing value from the list press option key while selecting the item to delete. An alert will ask confirmation:



The Flip items menu item flips items from the left side to the right side, and items on the right side to the left side. So the following dialog:



becomes:

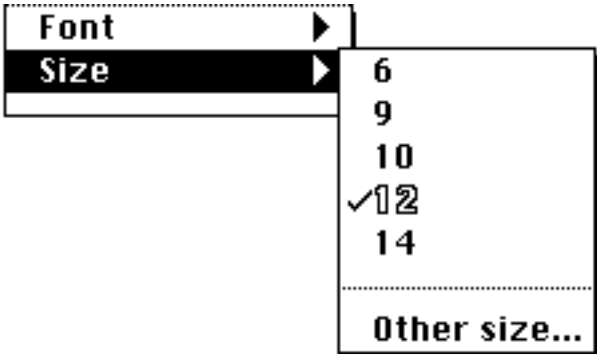


after flipping. This can be very useful when translating a left to right language to a

right to left language (English to Arabic for example).

Selecting the Font menu item lets you choose the font used to display text items in the dialog. This is useful when localizing CDEVs for instance, because they usually display a DITL in control panel window using Geneva 9 instead of standard Chicago 12. It should be noted that this is the *only* use for this function, as it does not actually change the fonts displayed in a dialog or alert box.

Size is available for the same purpose. The Size menu lets you choose many pre-defined sizes. You may even define a custom size if needed by selecting the Other size... command.



Custom sizes are added to the size list so they are directly available for later use. If the size list becomes too long you may want to remove some custom size items. To do this, just press option key while selecting the size you want to remove. A dialog box will ask for confirmation before removing the item.

Again it is important to note that the font and size functions are for visual effect only to enable you to see the dialog as the file would display it. The changes you make are *not* kept by the file you are editing.

The Use original size command is available if the selected (or anchor) item is an icon, a picture or a resource control. Selecting this command reverts item size to its original size (as viewed stand alone). This is useful since it may be difficult to exactly resize pictures, scroll bars, etc.

Stage 3: Translation Checking

The Verify translation Item

Checking the file structure is useful to make sure that the translated file contains the required resources and also checks for damaged resources.

You may select the Verify translation menu item as soon as both new original file and new translated file are set.

You need to select the Verify translation menu item to change the application and creator type of the new translated file to the same type found in the new original file. The translated file is then ready to be run from the Finder.

AppleGlot should never corrupt any resource it creates during the translation process (unless they are corrupted in the original file already!), because it never lets you alter structures critical to the performance of the file. This is one advantage of AppleGlot over using ResEdit as your sole translation tool. ResEdit and other resource editors allow you to easily damage or alter your file by accident. Nevertheless if warnings are generated, the only way to be sure your translation is ok is to closely examine the resource or item the warning is referring to. It is important to note that a proper translation is possible even if a warning is detected by AppleGlot. AppleGlot will always be on the safe side and warn you when possible that something questionable has been detected. But on the other hand, if you cannot resolve the reported error, you must decide to somehow edit directly the questionable resource or simply re-translate the file. Remember that once your Application Database is created, all translations are accounted for. Your next re-translation will place you immediately into the "Adjust Dialogs" and "Verify Translation" stage of the file's localization process.

Appendix A : Customizing Templates for AppleGlot

ResEdit templates are used to decode resources and decide whether a resource is translatable or not. AppleGlot assumes that a resource is translatable when its template contains one or more of the "**translatable item types**" listed below.

CHAR	a single characters
TNAM	four characters
PSTR	a Pascal string (length byte followed by text)
OSTR	odd padded Pascal string
ESTR	even padded Pascal string
WSTR	length word followed by text
LSTR	length long word followed by text

CSTR	a C string (text followed by a null characters)
OCST	odd padded C string
ECST	even padded C string
P0nn	Pascal fixed length string('nn' ranges from $00-$FF)
Cnnn	C fixed length string('nnn' ranges from $000-$FFF)
P240	An AppleGlot unique type defined as a Pascal string that cannot exceed 240 characters.

In some cases you may *not* want an item to be translated despite the fact the item is a "translatable" item.

For example you may not want FREF resource file types to be translated. Because the item field "file type" is defined as a TNAM, AppleGlot would normally translate it. You can prevent AppleGlot from translating such items by:

- changing the field type to an equivalent non-translatable type (TNAM is equivalent to H004, CHAR is equivalent to H001, PSTR is equivalent to OCNT LSTC H001 LSTE, etc.). All translatable types may be replaced by equivalent non translatable types... but this may not be always so easy (try ESTR).
- changing the type name to lower case. This is the simplest method since AppleGlot distinguishes upper case type names from lower case type names and NEVER attempts to translate lower case type name fields. So if you don't want a TNAM field to be translated, just change 'TNAM' to 'tnam' in the template. Unfortunately you can't change type names directly in resource templates because ResEdit checks for unknown types and won't let you set lower case type names. So you'll have to open the template 'as general' in ResEdit and patch the type name in the template ASCII dump.

You may also want to change some field names to make AppleGlot's messages more readable. This is often a good idea since AppleGlot uses template field names to build messages.

For example the "Key Equiv" field in a MENU template will cause messages like

Original Key Equiv: xxxx

to be generated. By replacing "Key Equiv" by "keyboard equivalent" you will get the message

Original keyboard equivalent:xxxx

which is more understandable to someone who is not familiar with resource templates. Because field names are used as part of strings, its a good idea to use only lower case.

AppleGlot unique item types: P240 and AUTO

Since there is a 240 character length restriction on the text fields in DITLs, and mistakes are made exceeding this limit, a special case item type was created so AppleGlot can correctly handle bad DITL's. P240 is similar to OSTR except that it is checked for a maximum character count of 240. Excess characters are reported if the resource is being expanded. In addition the excess characters are truncated if we are packing the resource.

The AUTO field was created to handle both the "new" DLOG, ALRT, and WIND template definitions for system 7, and the "old" style template definitions. The new "optional" auto-position field is handled internally as a combination of AWRD and HLNG. This AUTO field is used to keep "old" style resources "old", and "new" style resources "new".

Appendix B: Known Problems

- We've noticed that AppleGlot crashes on System 7.0b4 when virtual memory is active. Since we aren't sure if it is AppleGlot or virtual memory that is at fault we simply recommend that virtual memory be turned off when using AppleGlot.

- There is no fail-safe on out of memory conditions when the currently active heap zone is the SpeedZone. If it crashes then use the SpeedZone dialog to allocate more memory. Using the SpeecZone dialog you can also turn off the SpeedZone entirely

Appendix C: Warning List

Here is a list of all the '?? WARNING' resource strings that may appear in the history file. Below each one is an explanation of what causes it.

```
resource 'STR ' (kLowDiskSpaceStr, "kLowDiskSpaceStr", nonpurgeable) {
        "       #1 ?? WARNING Disk space appears to be low. Translation could possibly fail without further warning."
};
```

If the history file pre-allocation fails we get this message. In order to be sure that there is at least a reasonable amount of disk space AppleGlot pre-allocates some space for the history file. It may be allocating more space than is actually necessary. The STR resource named

kMinimumDiskSpaceWarningSizeStr contains a number which is the size that AppleGlot uses when allocating the history file.

```
resource 'STR ' (kMaybeNoDiskSpaceStr, "kMaybeNoDiskSpaceStr", nonpurgeable) {
        "       #2 ?? FATAL Probably not enough disk space to create clone file."
};
```

This version of AppleGlot will not load resources into the system heap during translation. To insure that it always loads the resources into AppleGlots own heap even if the resource has its SysHeap bit set AppleGlot simply turns off the SysHeap bit. Since AppleGlot isn't supposed to ever modify the original files it takes the easy way out and if it detects a SysHeap bit set in a file it makes a copy of that files entire resource fork so that it can modify the copy. This means that in these cases there must be enough room on the disk for a copy of the file. Usually if one file has a SysHeap bit sit all three files (New-US, Old-US and Old-Loc) have a SysHeap bit set too and so AppleGlot will need three times as much space if there is an old translation being processed. AppleGlot tries to allocate the disk space prior to copying the resource fork and if the allocation fails it outputs this message. In addition AppleGlot will pre-allocate some space for the history file as well since it knows that it will need to write to it at least a few times during normal operation. The only way around this one is to free up enough space on the disk for AppleGlot to do its work. Files such as the System are well over a megabyte in size and if there is a previous translation being used AppleGlot needs several megabytes to create its clone files, history file and the New-Loc file as well.

```
resource 'STR ' (kBadAutoField, "kBadAutoField", nonpurgeable) {
        "       #3 ?? WARNING Incorrect size for AUTO field. Is ^0, should be 2."
};
```

When packing a resource after expanding and translating it if AppleGlot finds that the size of the data for an AUTO field isn't 2 bytes it outputs this message. This shouldn't be possible and is most likely the result of an internal AppleGlot bug.

```
resource 'STR ' (kDataForkByteMismatch, "kDataForkByteMismatch", nonpurgeable) {
        "       #4 ?? WARNING Byte MISMATCH in data fork. ^0"
};
```

If the data forks of both the New-US and New-Loc files are the same length AppleGlot will check them byte-by-byte and if any single byte is different it will report the problem.

```
resource 'STR ' (kResTooBigTooFit, "kResTooBigTooFit", nonpurgeable) {
        "       #5 ?? WARNING !!! Resource too large to be loaded into memory"
};
```

During translation and verification AppleGlot needs to load each resource into memory to work on it. Prior to each load AppleGlot gets the size of the resource and asks the memory manager if there is enough room for the resource. If the memory manager say no we get this error message. When doing a translation pass and there is a previous translation to examine for reusable items AppleGlot needs to load three copies of each resource into its heap. When it is done working on them it creates a fourth copy and puts it into the New-Loc file. Usually translatable resources aren't too gigantic so they will fit without trouble unless you try to run AppleGlot in a much smaller space then it likes to run in. Non-translatable resources can be a bit of a problem sometimes. AppleGlot still needs to load copies into memory to examine them byte-by-byte to see if the New-US and Old-US resources are identical so it can copy the Old-

Loc resource into the New-Loc file. Sometimes application files contain non-translatable resources greater than 600K in size and if you try to get four of them into the heap at the same time AppleGlot isn't going to like it very much and will tell you that it failed to work correctly. The only way to fix this situation is to increase the amount of memory allocated to AppleGlot and try again.

```
resource 'STR ' (kUnexpectedTranslatableItem, "kUnexpectedTranslatableItem", nonpurgeable)
{
	"	#6 ?? WARNING !!! Unexpected translatable item caused old Work Glossary to be renamed as \"^0\""
};
```

During translation pass #1 the Work Glossary text file is created and each text item which AppleGlot can not find an exact translation for will be written to this file. After translation pass #1 has completed the Work Glossary has a complete list of everything that AppleGlot couldn't figure out. During translation pass #2 every possible translation is assumed to be accounted for in either the Work Glossary or the Application Database and no more errors need to be written into the Work Glossary. If anything goes wrong and AppleGlot encounters something it didn't expect to encounter in translation pass #2 this message reports that it had a problem. It often means that something in the Work Glossary context line or <New-US text> line was accidentally changed resulting in a translation changing its identity and confusing AppleGlot into sending a new unknown string into the Work Glossary. The original Work Glossary is renamed and your efforts at translation are not lost but you should examine the older Work Glossary or Application Database to see if there is something wrong which could be corrected.

```
resource 'STR ' (kVerifyVO2IndexedResourceMissing,
"kVerifyVO2IndexedResourceMissing", nonpurgeable) {
	"	#7 ?? WARNING Missing indexed resource '^0' in New-US"
};
```

During verification AppleGlot gets all resources of a given type in map index order. It gets a count of the number of resources of a given type in the New-US file and then proceeds to get each of them in turn. If the resource somehow fails to load AppleGlot reports it with this string. It would most likely be the result of an internal AppleGlot bug.

```
resource 'STR ' (kRsrcAttributesDifferent, "kRsrcAttributesDifferent", nonpurgeable) {
	"	#8 ?? WARNING Resource attributes are different!    ^0"
};
```

When AppleGlot is verifying the New-US and New-Loc files it examines the attribute flags of both files. If they are not identical it reports the problem. This usually means that someone has used ResEdit to modify the attributes of the New-Loc file.

```
resource 'STR ' (kResourceMapAttributesDontMatch, "kResourceMapAttributesDontMatch",
nonpurgeable) {
	"	#9 ?? WARNING Map attributes are different!    ^0"
};
```

AppleGlot compares the resource map attribute of the New-US and the New-Loc files. If they are not identical this message reports the error. This usually means that someone has used ResEdit to modify the attributes of the New-Loc file.

```
resource 'STR ' (kVerifyVO2IndexedRT2ResourceMissing,
"kVerifyVO2IndexedRT2ResourceMissing", nonpurgeable) {
        "       #10 ?? WARNING Indexed resource ^0 missing from translation"
};
```

During verification AppleGlot gets all resources of a given type in map index order. It expects every resource in the New-US file to be in the New-Loc file as well. If it fails to get a resource from the New-Loc file when it requests it by index AppleGlot will report this message.

```
resource 'STR ' (kVerifyExtraVT2Resource, "kVerifyExtraVT2Resource", nonpurgeable) {
        "       #11 ?? WARNING Extra resource ^0 in New-Loc which isn't in New-US"
};
```

During verification there is a special sweep of the New-Loc file looking for resources which aren't in the New-US file. If an extra resource is found in the New-Loc file which isn't in the New-US file this message reports the news. This means that there may have been resources manually moved into the New-Loc file by ResEdit or there may have been extra resources in the Old-Loc file which were copied into the New-Loc file during translation.

```
resource 'STR ' (kVerifyBytesDifferent, "kVerifyBytesDifferent", nonpurgeable) {
        "       #12 ?? WARNING Bytes not equal ^0"
};
```

When a resource without a TMPL in the New-Loc file is found to be of the same size as the same resource in the New-US file AppleGlot expects the internal bytes to be identical as well. If so much as one byte is different then AppleGlot writes this comment. It is possible that there was one or more small modifications made with ResEdit which AppleGlot doesn't know anything about. If AppleGlot copies a resource from the Old-Loc file into the New-Loc file it is possible that the differences were made by ResEdit during an earlier localization.

```
resource 'STR ' (kVerifySizesDifferent, "kVerifySizesDifferent", nonpurgeable) {
        "       #13 ?? WARNING Sizes not equal ^0"
};
```

If there is no TMPL for a resource then AppleGlot won't change it and expects it not to change. When AppleGlot finds a difference in the sizes of the New-Loc and New-US resources it writes this comment. There are times that these resources will be changed and not be the same size such as when a PICT resource is changed during localization. If a changed PICT (or other type) resource is copied from the Old-Loc to the New-Loc files AppleGlot will still report that the resource size is not the same as the resource in the New-US file. It is up to the user to check that the resource in the New-Loc file is valid. Note that the Signature and 'vers' resource types are special and won't be checked even though they have no TMPL in AppleGlot.

```
resource 'STR ' (kVerifyNamesDifferent, "kVerifyNamesDifferent", nonpurgeable) {
        "       #14 ?? WARNING Resource names are different ^0"
};
```

If the verify pass finds that the name of the New-Loc resource isn't identical to the name of the New-US resource then AppleGlot writes this message into the history file. It may or may not be of any importance that the names match. It depends on whether or not the resource is called from the program by name or by number. If it is called by number then the name is of no concern but if the resource is called by name then the program will fail to find the

resource.

```
resource 'STR ' (kVerifyCancelled, "kVerifyCancelled", nonpurgeable) {
	"\0x0d\0x0d\0x0d	#15 ?? !!!! Verification process CANCELLED !!!!
\0x0d\0x0d\0x0d"
};
```

This one means that the user clicked on the CANCEL button in the progress dialog during the verification pass and the verification was aborted before it was completed.

```
resource 'STR ' (kVerifyExpansionsNotEqual, "kVerifyExpansionsNotEqual", nonpurgeable) {
	"	#16 ?? WARNING Non-translatable field in New-Loc not same as New-US."
};
```

During verification the New-US and New-Loc resources are expanded and inspected on a field by field basis. If any of the fields which are not RECTs and are not text are not identical this message informs us that some item has changed which AppleGlot didn't expect to change. This will happen if you use ResEdit to modify some numeric field or something similar and then run AppleGlot again. Since you changed a field that AppleGlot would not have changed it will report that there is a problem. You will need to use ResEdit to examine the two resources to see what has changed and if it is something you find acceptable or not. It is quite possible that you didn't run ResEdit on the resource but instead AppleGlot copied it from the Old-Loc file for you. In this case the result is still the same, a non-translatable field is different in the New-Loc resource and AppleGlot will write this comment into the history file.

```
resource 'STR ' (kHistoryNullMissing, "kHistoryNullMissing", nonpurgeable) {
	"	#17 ?? WARNING Null missing from end of field in file ^0"
};
```

Fields of type CSTR are expected to have a terminating NUL character. If during expansion of the resource the end of the block is reached without encountering a NUL this is the message which informs you of the problem.

```
resource 'STR ' (kHistoryRsrcLong, "kHistoryRsrcLong", nonpurgeable) {
	"	#18 ?? WARNING Resource too long for TMPL in ^0"
};
```

If during the expansion of a resource the TMPL indicates that the resource has been fully expanded but there remains more data in the resource block beyond AppleGlots byte pointer this message will be written. Either the TMPL is incorrect or the resource truly has some extra bytes after all the valid stuff. I suppose this could happen if AppleGlot gets lost due to an internal bug.

```
resource 'STR ' (kHistStr255TooBig, "kHistStr255TooBig", nonpurgeable) {
	"	#19 ?? WARNING PString > 255 characters in ^0"
};
```

If during expansion of a resource item of type PSTR the data should become larger than 255 bytes this message will be written into the history file. Since the PSTR field takes the first byte of the data as its length and a byte can't exceed FF(hex) or 255(decimal) this string could only occur if there was a bug in AppleGlot itself which caused the block containing the PString characters to expand beyond 255 bytes.

```
resource 'STR ' (kHistTranslatedStr255TooBig, "kHistTranslatedStr255TooBig", nonpurgeable)
{
	"	#20 ?? WARNING Translated PString > 255 characters in ^0. String clipped to
255 characters."
};
```

When a translated PSTR, ESTR or OSTR field in the Work Glossary or Application Database exceeds 255 characters in length you will get this message because a PString can only hold 255 characters at most. The resulting resource will have the string in this field truncated to 255 characters so it will be a valid resource but its contents in the particular field in question will be incomplete.

```
resource 'STR ' (kHistTranslatedStr240TooBig, "kHistTranslatedStr240TooBig", nonpurgeable)
{
	"	#21 ?? WARNING Translated PString > 240 characters in ^0. String clipped to
240 characters."
};
```

If a translation in your Work Glossary (or Application Database) contains a string which is longer than 240 characters and the TMPL field is P240 then you get this message in the history file. AppleGlot will truncate the string to 240 characters to create a correct DITL but the contents of the DITL won't be complete as far as the PString is concerned.

```
resource 'STR ' (kHistExpandedStr240TooBig, "kHistExpandedStr240TooBig", nonpurgeable)
{
	"	#22 ?? WARNING Resource field PString > 240 characters in ^0"
};
```

When expanding a resource field of type P240 from the new/old original files or the new/old translated files, the maximum number of characters in the PString is 240. If the length byte of the PString is greater than F0(hex) or 240(decimal) then this message is written into the history file. P240 is a special field type designed to check DITL text items for correctness. The DITL will work correctly but Inside Mac states that text items have a maximum of 240 characters and when a DITL with more than 240 characters is opened in ResEdit the editor will usually complain that some field following the text item is incorrect. Editing with the ResEdit TMPL editor will work in spite of the invalid field.

```
resource 'STR ' (kMovedVT1ExtraResourceToVT2, "kMovedVT1ExtraResourceToVT2",
nonpurgeable) {
	"	#23 ?? WARNING Found extra resource in ^0 "
};
```

When AppleGlot encounters an extra resource in the Old-Loc file which wasn't in the New-US file and the user chooses to copy it into the New-Loc file AppleGlot will write this line into the history file.

```
resource 'STR ' (kSkippedVT1ExtraResource, "kSkippedVT1ExtraResource", nonpurgeable) {
	"	#24 ?? WARNING Ignored extra resource in ^0"
};
```

When during translation AppleGlot encounters extra resources in the Old-Loc file but the user chooses to not copy them into the New-Loc file AppleGlot will always make a note of the fact that it is skipping the resource so you will later know exactly what it did when you read

the history file.

```
resource 'STR ' (kHistResExtendedWarning, "kHistResExtendedWarning", nonpurgeable) {
        "       #25 ?? WARNING Resource 'EXTEND' bit is set in ^0. May be a
COMPRESSED resource."
};
```

AppleGlot isn't prepared at the present time to deal with ANY of the resource extensions. The only ones defined so far are compression schemes. This message will occur during verify when the resource attributes are checked and the Extended bit is set. During translation this message may appear when the New-US resource is loaded for translation if the Extended bit is set. It can also occur when an extra resource is copied from the Old-Loc file into the New-Loc file. In addition it may happen if the New-Loc resource has had its Extended resource bit set somehow. There have been cases where the Extended bit was set in a resource but ResEdit didn't put a check in the Compressed item box (I suppose that the resource wasn't actually compressed). When the resource was DeRez'd it was found to have attributes of $01 so be sure you use DeRez to check it as well as using ResEdit.

```
resource 'STR ' (kVerifyExtraVO2Resource, "kVerifyExtraVO2Resource", nonpurgeable) {
        "       #26 ?? WARNING Extra resource ^0 in New-US which isn't in New-Loc"
};
```

During verification there is a special sweep of the New-US file looking for resources which aren't in the New-Loc file. If extra resource are found in the New-US file which aren't in the New-Loc file this message reports it. Usually this means that there have been resources manually removed from the New-Loc file by ResEdit.

```
resource 'STR ' (kHistoryRsrcShort, "kHistoryRsrcShort", nonpurgeable) {
        "       #27 ?? WARNING Resource too short for TMPL. Missing ^0"
};
```

When a resource is being expanded and the TMPL tells AppleGlot to expect more data but the resource suddenly ends we post this message into the history file. Generally AppleGlot will continue to parse the TMPL and fill the expansion fields with NUL characters to pad in the missing pieces. The resulting resource will conform to the expected size but its fields may not contain the correct data if NUL isn't valid.

```
resource 'STR ' (kP0nnStrTooLong, "kP0nnStrTooLong", nonpurgeable) {
        "       #28 ?? WARNING String length byte too big for field. ^0"
};
```

When expanding a 'P0nn' field in a resource this message will appear in the history file if the length byte of the PString is greater than nn. If the PString was created using a hex editor in ResEdit this could occur if you don't pay attention to the constraints of the P0nn maximum size.

```
resource 'STR ' (kP0nnTranslatedStrTooLong, "kP0nnTranslatedStrTooLong", nonpurgeable) {
        "       #29 ?? WARNING Work Glossary string too big for field. ^0"
};
```

This string appears if the Work Glossary (or Application Database) contain a translated string which is too long to fit the specified size for a P0nn field. nn is two hex characters which specify the maximum length of the PString. For example, if the field is P031 and the translation

is longer than 31(hex) or 49(decimal) characters you'll get this error message.

```
resource 'STR ' (kCnnnNullNotFound, "kCnnnNullNotFound", nonpurgeable) {
	"	#30 ?? WARNING NUL terminator missing from field. ^0"
};
```

When expanding Cnnn fields AppleGlot expects to find a NUL character someplace within the limits of the field. If no NUL is located this is the message telling us that something is wrong with the CString assigned to the field.

```
resource 'STR ' (kNoNulForLSTZ, "kNoNulForLSTZ", nonpurgeable) {
	"	#31 ?? WARNING LSTZ NUL terminator not found. ^0"
};
```

During expansion of LSTZ fields in a resource AppleGlot expects the field to be terminated by a NUL byte. LSTZ fields are the last group of fields in a resource and if AppleGlot should happen to encounter the end of a resource without finding the termination NUL it will put this message into the history file. See the TMPL for MENU for an example of a LSTZ loop.

```
resource 'STR ' (kChangedResProblemSTR, "kChangedResProblemSTR", nonpurgeable) {
	"	#32 ?? WARNING ChangeResource Error = ^0"
};
```

When AppleGlot is moving a translated resource into the New-Loc file it must put it into the same resource map position so the resource map sequence remains unchanged. In order to do this AppleGlot keeps the handle to the resource which it pulled out of the New-Loc file which it used to preserve any non-translatable items. It then tosses away the contents of the handle and substitutes the newly packed resource for the original New-Loc resource. It next makes a call to ChangedResource to tell the resource manager that it needs to rewrite the resource into the New-Loc file using the old handle. If there is anything wrong in AppleGlots internal logic and the call to ChangedResource fails AppleGlot will write this message into the history file.

```
resource 'STR ' (kCnnnShort, "kCnnnShort", nonpurgeable) {
	"	#33 ?? WARNING String shorter than Cnnn. ^0"
};
```

This one may be more annoying then helpful. If a CString is shorter than the full 'Cnnn' field size this message is placed into the history file. It seems that the rules for 'Cnnn' fields allow for CStrings from zero length up to nnn-1 in length.

```
resource 'STR ' (kCnnnTooLong, "kCnnnTooLong", nonpurgeable) {
	"	#34 ?? WARNING String longer than Cnnn allows. ^0"
};
```

This could happen during packing of a resource if the text assigned into a field of type 'Cnnn' is too large to fit the field. 'Cnnn' is a variable sized CString which isn't allowed to exceed nnn in length where nnn is three hex characters. When this message appears in the history file AppleGlot has truncated the CString by simply placing a NUL in the last position in the field.

```
resource 'STR ' (kExpansionUnknownTMPLFieldTYPE,
```

"kExpansionUnknownTMPLFieldTYPE", nonpurgeable) {
	"	#35 ?? WARNING Unknown TMPL field type during expansion. ^0"
};

During expansion of resources AppleGlot is using the field type from the TMPL resource to tell it how to expand each field in the resource. If there is a TMPL field which is new that AppleGlot doesn't yet know about it will output this string into the history file. If there is a simple misspelling in the TMPL type field the result will be the same.

---

resource 'STR ' (kPackingUnknownTMPLFieldTYPE, "kPackingUnknownTMPLFieldTYPE", nonpurgeable) {
	"	#36 ?? WARNING Unknown TMPL field type during packing. ^0"
};

When AppleGlot is packing a resource it has just expanded and finds an expansion type that it doesn't recognize this is the string that is displayed. It is the result of an internal logic error in AppleGlot itself. If it understood what the type was during expansion it should also understand the same type during packing. Since the expansion and packing routines are entirely separate this is basically an error message used to keep the programmer on his toes.

---

resource 'STR ' (kHistCancelled, "kHistCancelled", nonpurgeable) {
	"\0x0d\0x0d\0x0d	#39 ?? !!!! Translation process ABORTED !!!! \0x0d\0x0d\0x0d"
};

This happens if the user clicks the CANCEL button in the progress dialog or if the history file itself isn't correctly initialized. But since the history file couldn't receive the string if it wasn't open this really only happens if you cancel the translation by clicking the button in the progress dialog.

---

resource 'STR ' (kHistoryNoHistory, "kHistoryNoHistory", nonpurgeable) {
	"	#40 ???????????????? - Translation sequence not monitored! - AppleGlot bug"
};

This is seldom seen string is the result of an empty flag pattern which would mean that AppleGlot didn't do anything to the resource being translated. It should only be seen if there is an internal AppleGlot logic error.

---

resource 'STR ' (kHistoryHasBug, "kHistoryHasBug", nonpurgeable) {
	"	#41 ???????????????? - Undocumented translation sequence! - AppleGlot bug"
};

This little gem is what pops out if AppleGlot sets a flag pattern to document the specific internal paths AppleGlot used to perform a single resources translation and it later finds that there is no matching message to describe the sequence. It is therefore the result of an internal AppleGlot bug.

---

resource 'STR ' (kCloneAbort, "kCloneAbort", nonpurgeable) {
	"	#42 ?? !!!!!!!! --- Clone file creation ABORTED --- !!!!!!!!"
};

This happens if the file associated with one of the four file icons contains a system bit which forces AppleGlot into making a copy of the file. If there isn't enough disk space to hold the copy then we can't proceed. There are several other internal checks during the cloning

process which could possibly abort the operation too, such as not finding string resources needed to name the clone file or if there is a logic error in AppleGlots code.
