AppleGlot 3.2 User's Guide

Contents

What is AppleGlot?

AppleGlot is a localization tool for Mac OS X software. AppleGlot can extract localizable information from Mac OS X software into a text file, where linguistic translators who have no engineering background can easily translate the strings into the target language. AppleGlot then reinserts these strings back into the application. AppleGlot also supports incremental localization; once software has been localized, subsequent use of AppleGlot extracts only the new or changed items in the file that potentially require localization. AppleGlot supports Cocoa and Carbon applications, bundled applications, and shared libraries or frameworks.

AppleGlot IS

· A text extraction tool. Extracted text is placed in Work Glossary files to be translated.

· An incremental update tool. Much of the localization work from previously localized software can be preserved for new builds or releases of software. AppleGlot is capable of preserving translated strings and localized layout information.

AppleGlot IS NOT

· A translator. AppleGlot is a text swapping program. A native speaker of the target language is expected to translate the Work Glossary files that AppleGlot produces. AppleGlot can then substitute the original text in the resource files with the localized text. AppleGlot is able to use glossary files from previous localizations, but this only works on 100% text matches; it will not do partial translations.

· A resource editor. Use Interface Builder, Resorcerer® or PowerPlant Constructor to adjust the non-text components of resources.

AppleGlot is designed to speed up the localization process by providing two services:

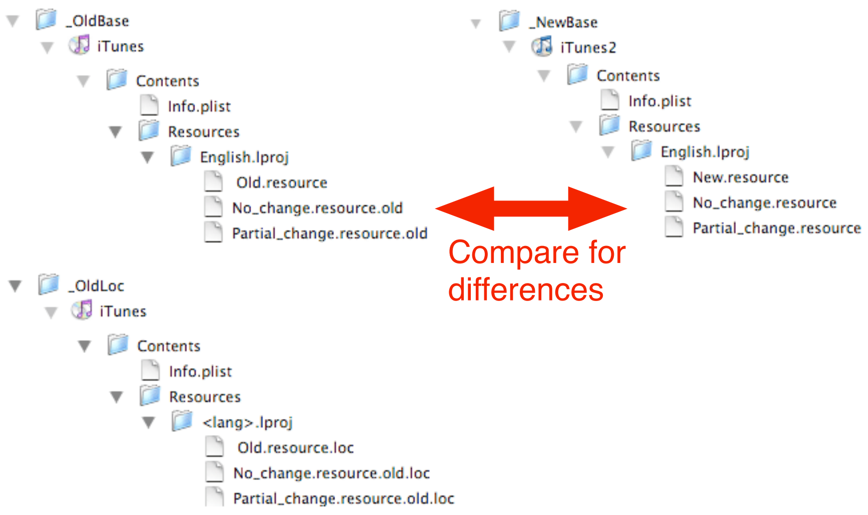
1. Text Extraction and Insertion - AppleGlot extracts text from resource files of Mac OS X software, which usually are located in localized resource directories (.lproj directories like English.lproj or Japanese.lproj) in the Resources folders in software bundles. AppleGlot extracts strings from Classic files, Carbon resource files (.rsrc files), Cocoa nib files and strings files into XML files called Work Glossary files, where the UI strings can be translated. After UI strings are translated in the Work Glossary files, AppleGlot needs to be run again to insert the translated strings back into the resource files of the software bundle.

2. Incremental Localization - AppleGlot does incremental updates from previously localized versions of software, by allowing localizers to obtain only the new or changed portion of strings from the software it is localizing. This minimizes the amount of effort needed to update the localization for a new release by preserving previously localized resources, or portions of resources and avoids the need to repeat earlier work. Localizers only need to work on items that have changed, or are new in a new release.

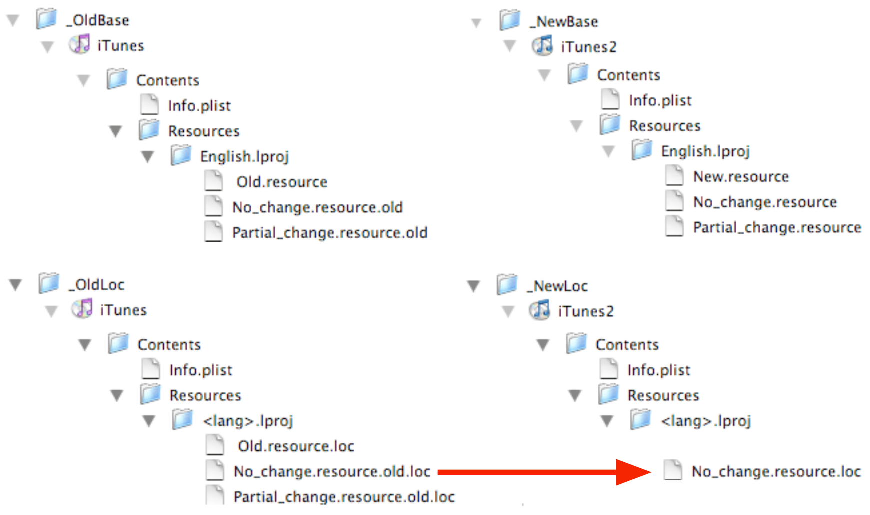
What does AppleGlot do?

In general, AppleGlot will perform the following actions to achieve incremental localization:

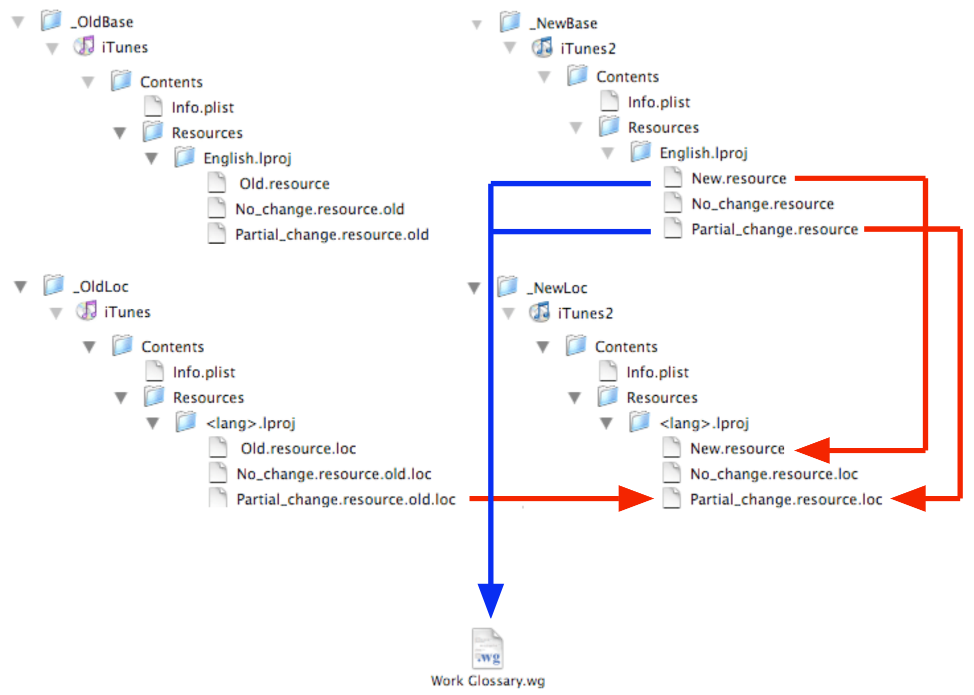
1. AppleGlot compares previous version of the base software (_OldBase) and current version of the base software (_NewBase).



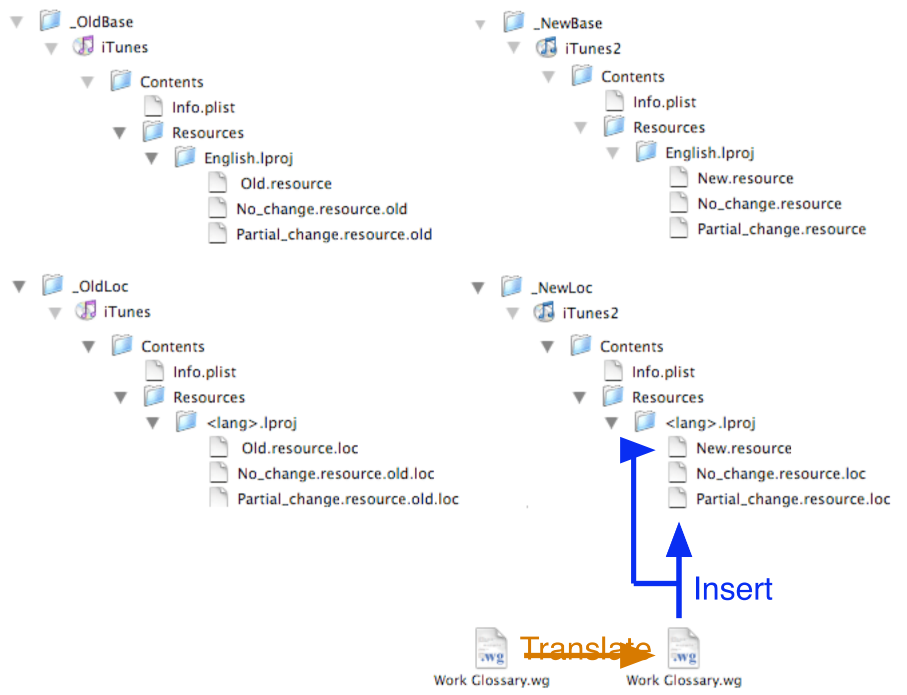
2. From the result, the unchanged portion of the previous localization (_OldLoc) will be leveraged to the current localization (_NewLoc).



3. Localizable strings in the changed or new portions of the software (_NewBase) will be extracted to the Work Glossary text file. The new portion of the base software is copied into the current localization (_NewLoc). The portion of the old localization that could be leveraged to the current localization will be carried over to the current localization.



4. After the Work Glossary is translated, AppleGlot will insert the translated strings into the current localization (_NewLoc).



What's new in AppleGlot 3.2

- Improved leveraging - AppleGlot no longer modifies NewLoc if files in NewBase and OldBase are the same.
- Performance enhancement - A lot of internal processes were reviewed; files are processed more efficiently.
- Command Line Interface - AppleGlot now provides a command line interface along with GUI. For details, see Appendix A.
- Default Name List - Basically default names and titles in nib file such as "OtherViews", "Box", "Window" et cetera shouldn't be exposed to users at all. That is, these are not required to be localized in general. AppleGlot now moves such item directly to Application Dictionary file and labels it as 'origin = "Default Name List".'

Requirement for AppleGlot

• AppleGlot 3.2 requires Mac OS X 10.2 or later. It is highly recommended to run on Mac OS X 10.3 or later.
• AppleGlot 3.2 requires appropriate developer package that is supported in your OS platform. AppleGlot requires a tool called nibtool to extract strings from nib files. nibtool will be installed in /usr/bin by the developer package. AppleGlot always uses nibtool located in /usr/bin.
• nibtool requires objects.nib (or keyedobjects.nib, or objects.xib), info.nib and classes.nib in a nib file package to extract localizable data. (The info.nib and classes.nib are often stripped from shipping products from Apple, as it is unnecessary for run-time operation. This means AppleGlot cannot process nib files from some of products shipped by Apple.)
• AppleGlot environment should only be placed on local HFS+ volumes. Mounted AFP volumes, HFS volumes, or UFS volumes are not supported.

AppleGlot Environment

AppleGlot requires a particular environment to organize the various files it needs or will create. An environment is simply a set of folders with specific names. AppleGlot provides the "Create Empty Environment..." menu option in the "Tools" menu to allow you to create and name an empty environment automatically. You could do this manually using the Finder but you would need to get all the folder names spelled exactly right, therefore it is much simpler to let AppleGlot create its environment for you.

The AppleGlot environment folder contains the following folders:

_ApplicationDictionaries

Application Dictionary files (.ad) files are generated after the initial pass. Application Dictionary files contain all "Base Language - Localized Language" pairs of translated strings that are leveraged as 100% matches from files in _OldLoc, _WorkGlossary or _LanguageGlossaries folders into _NewLoc folder. Contents of Application Dictionary file always reflects the latest strings in the files in _NewLoc folder at any given stage. If none of the strings in _NewLoc folder are translated, Application Dictionary file would be empty. Once the Work Glossary file is translated and the translated strings are inserted from Work Glossary to files in _NewLoc folder, inserted strings will be moved to Application Dictionary file from Work Glossary. This indicates AppleGlot has successfully inserted translated strings into the files in _NewLoc folder. After the initial pass is done, you may edit Application Dictionary files directly to correct translation errors. The corrected translations will be inserted to files in _NewLoc folder after an incremental pass is done.

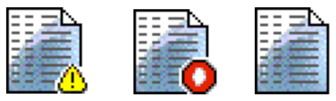
_LanguageGlossaries

Language Glossary files (.lg) should be placed in this folder before the initial pass is performed. If you wish to leverage translations from related software, place the Application Dictionary files from such software here. You are required to change their file extensions from .ad to .lg. No further changes are necessary to use .ad files as Language Glossary files.

Note: *In AppleGlot 3.2, matched strings from the Language Glossary files will not be inserted to _NewLoc after Initial Pass (pass 1). Instead, matched strings will be inserted to the Work Glossary files, where a linguistic translator will be able to review what AppleGlot pulled from the Language Glossary files. After the review, an Incremental Pass (pass 2) will insert matched strings to _NewLoc.*

_Logs

Log files are generated in this folder after each AppleGlot pass is performed. If you find errors or warnings in the logs, you should compare the contents of _NewBase folder and _NewLoc folder to make sure that nothing is corrupted in files in _NewLoc folder.



If you see a log file with the yellow mark on its icon, the log file contains warnings. Red mark on the icon indicates that the log contains errors. Please be sure to check the log if you receive a warning or an error. The rightmost icon shown indicates the AppleGlot didn't encounter any problem during the pass.

For details, please see the section called "How to read AppleGlot logs".

_NewBase
: You are required to place one or more new base software in this folder. AppleGlot will generate localized version of the software you place in this folder in the _NewLoc folder. Contents of this folder are used as basis for current AppleGlot environment.

_NewLoc
: Partially localized version of new base software will be generated here after the Initial Pass. After all AppleGlot cycles are done, you will need to modify resources in this folder to adjust the layout of dialog, etc. to finalize your localized software.

_OldBase
: If available, place previous versions of the base software here. Contents of this folder will be compared against the contents of _NewBase folder to detect changed or new portions in the new software. If the software you place in _NewBase folder has never been localized before, this folder may be left empty.

_OldLoc
: If available, place the localized version of the software you placed in _OldBase folder here. Previous localization work will be extracted from the contents of this folder, and used to create partially localized software in _NewLoc folder. You are required to place a localized version of the file you placed in _OldBase here. AppleGlot assumes that the structure of files in _OldBase and _OldLoc are the same. Otherwise, AppleGlot cannot leverage correct information into _NewLoc. If the software you place in _NewBase folder has never been localized before, this folder may be left empty.

_Temporary
: AppleGlot 3.2 uses this folder as a temporary folder to process Carbon resource files. You should not place anything here. Please keep this folder clean all the time.

_Translators
: Place a set of Monte translators here. AppleGlot contains a resource-parsing engine called Monte to analyze resource manager resources used in Classic files or Carbon resource files. Monte requires you to have one Monte translator for each resource type to parse the resource, and to extract localizable strings, or layout information. You should obtain a copy of the latest Monte translator set provided from the ADC site. If you are

required to create your own Monte translator for your custom resource type, please refer to the MonteSDK.pdf, which can be found on the ADC site.

_WorkGlossary

Work Glossary files (.wg) files are generated after the initial pass. Work Glossary files contain localizable strings from changed or newly added resources. linguistic translator is required to translate these files. Once translation is complete, perform the Incremental Pass of AppleGlot. Translated strings in Work Glossary files will be inserted into the files in _NewLoc folder. Application Dictionary files in _ApplicationDictionaries folder will be updated accordingly (Application Dictionary files always reflect the strings in _NewLoc folder).

AppleGlot Glossary files

Types of glossary files

There are three types of glossary files. Application Dictionary files (.ad) files contain all "Base Language - Localized Language" pairs of translated strings that are leveraged as 100% matches from files in _OldLoc, _WorkGlossary or _LanguageGlossaries folders into the _NewLoc folder. The contents of Application Dictionary file reflect the contents of files in _NewLoc folder at any given stage. Work Glossary files (.wg) contain localizable strings from changed or newly added resources. A linguistic translator is required to translate the files generated in this folder. Language Glossary files is used differently by AppleGlot. Language Glossary files (.lg) is the Application Dictionary files from previously localized software. When AppleGlot performs the Initial Pass, it parses all Language Glossary files in _LanguageGlossaries folder, then will try to find any new or changed strings that exactly match the strings from the Language Glossary file and will use the translation previously made in the previous localization.

Glossary file	**Description**
Application Dictionary	Contains all "Base Language - Localized Language" pairs of translated strings in _NewLoc. Content reflects the contents of files in _NewLoc at any given stage.
Work Glossary	Contains strings from changed or new resources. Linguistic translator is required to translate the strings.
Language Glossary	Application Dictionary files from previously localized software. Translation in Language Glossary file is leveraged to Work Glossary file, if 100% match is found.

Format of glossary files

Glossary files follow the XML standard. All glossary files should be opened and saved in UTF-8 encoding. Pay extra attention to the preference setting of your text editor. If you open glossary files in legacy encoding other than UTF-8, you will find high ASCII characters in the files are displayed as garbled characters. This is because character mapping of the high ASCII range is different among other encodings. If you find garbled characters, you should make sure that you are opening the file in UTF-8. Also ensure that the line break code in the glossary files are “LF” (0x0a) using appropriate text editor or hex editor.

It is not recommended to change the character encoding of the glossary file to one of the other encodings, then change it back to UTF-8. There are various round-trip conversion problems between Unicode and other encodings because Unicode is a much larger character set. If you are required to do so, please pay extra attention.

Note: *Although it is not recommended, AppleGlot is also capable of generating glossary files in UTF-16 as more text editors support UTF-16 better than UTF-8.*

Because glossary files are XML, the following characters are illegal in the glossary files. If you need to use following characters in the translation, please be sure to use following entity references in the glossary files:

```
&     &amp;
<     &lt;
>     &gt;
"     &quot;
'     &apos;
```

Note: *AppleGlot does not support CDATA section in the glossary files.*

This is an example of an entry in AppleGlot3 glossary files:

```
<File>
<Filepath>iTunes2.app/Contents/Plug-ins/Nomad Jukebox
Plugin.bundle/Contents/Resources/English.lproj/Localized.rsrc</Filepat
h>

<TextItem>
<Description></Description>
<Position>'CNTL' -20459:title</Position>
<TranslationSet>

      <base loc="en"                            >Kind</base>
      <tran loc="ja" origin="OldLoc exact match">種類</tran>

</TranslationSet>
</TextItem>

...
</File>
```

Tags	**Description**
<Filepath>	The path of the file where the strings are extracted from.
<Description>	If the text item is extracted from a nib file, description of the item will be inserted here.
<Position>	If the text item is extracted from a resource file, description of the item will be inserted here.
<base>	This is where base string (usually in English) is extracted.

<tran>	This is where translated string will be inserted either by AppleGlot or linguistic translator. In the origin attribute, AppleGlot places explanation of where the string was inherited from

When AppleGlot inserts a pre-translated string to Work Glossary file, AppleGlot uses the "origin=" attribute in the <trans> tags to indicate where the string is leveraged from:

`<tran origin="LG matched text">`

Strings are found in LG files, exactly matched. (position tag is not considered in this case).

`<tran origin="AD matched text">`

Strings are found in AD files, exactly matched, but position tag isn't matched. (this attribute will be set in Work Glossary file)

How AppleGlot handles different character encodings in localizing software

AppleGlot extracts strings from Macintosh resource files using a resource parser engine called Monte. Apple provides generic Monte translators that will be used by Monte to extract strings from Macintosh Resource files.

Most resource manager resources used in Macintosh resource files use default character encoding for given languages. For example, in Carbon resource files, all of European languages that Apple ships with current Mac OS X are saved in MacRoman. Japanese resources are saved in MacJapanese, Korean resource are saved in MacKorean, etc.

Monte will decompose packed binary data structures (such as resource manager resources) to extract text data as described in Monte translators, converts the extracted data to Unicode, then AppleGlot write out Unicode text to the glossary files for translation. When the strings are translated in the glossary files, AppleGlot will pass the translated Unicode strings to Monte, then Monte will insert the translated Unicode text to its original data structure after re-encoding the strings to the native encoding of target language.

For more details, please refer to "MonteSDK.pdf", which can be found on the ADC site.

Special escape sequence in AppleGlot glossary files

You should pay extra attention when using any 1-byte character that is in the range outside of 0x21 to 0x7E (non-low ASCII characters). For example, if you use "©" (small copyright mark) as a paramtext identifier like "©1", there will be a problem when localizing into Asian languages. "©" is 0xA9 in MacRoman, but it is 0xFD in MacJapanese. Since expected code to detect paramtext is 0xA9, "©" would not function as expected if it was saved in MacJapanese. It really should be localize into "ｩ". For Simplified Chinese, the story is even worse, because nothing is mapped to 0xA9 in MacChineseSimp. In order to force insert hex data 0xA9 in the appropriate location, user can specify the following escape sequence denoted as "?#xXXXX;", where XXXX is the four-digit hexadecimal number as the representation of the byte.

The part "XXXX" must be four digits. For values smaller than 0x1000, use preceding zeros like "?#x0096;" (for 0x96). Preceding 0's in the digits of escape sequences are ignored when they are converted to byte. For example, ?#x0096; results in the byte 0x96, not 0x0096. Since the number of the digits in a sequence is supposed to be four, you can't use formats like "?#x96;" or "?#x096;" for the byte 0x96. In addition, ?#x0000; results in 0x0.

If you want to enter multiple bytes, you can use multiple escape sequences. For example, if you put "日本" with MacJapanese encoding, the sequence is "?#x93fa;?#x967b;". (another sequence "?#x0093;?#x00fa;?#x0096;?#x007b;" inserts the same bytes.)

This escape sequence plays the role when:

(1) Macintosh resource files in _OldLoc contain any non-low ASCII character.

They will be extracted into AD file represented with the escape sequence. An actual example of AD file entry is:

```
      <base loc="en"                            >©1 can't be launched be-
cause it is not an application.</base>
      <tran loc="zh_CN" origin="OldLoc exact match">不能 ?#x00a9;1, 因为它不是
应用程序。</tran>
```

(2) You want to insert special characters.

For example, preparing the following entry in a Spanish WG file, you can put the text "La Lega Española" ("ñ" = 0x96 in MacRoman) into the corresponding Spanish rsrc file:

```
      <base loc="en"          >The Spanish League</base>
      <tran loc="es" origin="">La Lega Espa?#x0096;ola</tran>
```

This may be useful if you can't enter "ñ" from the keyboard directly (i.e. if typing option-n n doesn't work).

(3) Line breaks in Macintosh resource files

If strings in Macintosh resources uses the Classic Mac line break (0x0d), strings from nib or .strings file contains Unix line break (0x0a), AppleGlot will escape them as shown below. Unicode line separator u2028 (used by BBEdit), and paragraph separator u2029 are escaped the same way:

```
      <base loc="en"                            >Hello.?#x000a;Point to an
item to learn more about it.?#x000a;(Not all items have extra
information.)?#x000a;?#x000a;</base>
```

```
      <base loc="en"                              >Calibrator 4.0?#x000d;A
ColorSync Display Calibrator Assistant</base>
```

These escapes are done by AppleGlot so that these codes can be safely preserved in the glossary files. Some text editors convert these line breaks into one kind automatically. For example, Classic Macintosh line breaks (0x0d) in a text file would be converted to the Unix counterpart (0x0a). If the code is converted, the string that contained 0x0d will be treated as a different string, so that the translation in the glossary will be ignored and will never be inserted to the resource file.

Supported files

AppleGlot currently supports the following file types:

Macintosh resource files (Classic files / Carbon resource files)

Macintosh resource files, often referred as Carbon resource files. AppleGlot contains a resource-parsing engine called Monte to analyze resource manager resources used in Classic files or Carbon resource files (AppleGlot 2.4 used another custom data descriptor called templates). The data may be stored in either a data fork or a resource fork. In order to extract string data from .rsrc files, appropriate Monte translators are needed for all localizable resource type.

.strings files

AppleGlot supports .strings files in the following format:

```
/* Question in confirmation panel for quitting. */
"Confirm Quit" = "Are you sure you want to quit?";

/* Message when user tries to close unsaved document */
"Close or Save" = "Save changes before closing?";

/* Word for Cancel */
"Cancel";
```

AppleGlot string parser is capable of handling either "key" = "value"; form or just "key"; form. It will also extract the comments in the .strings files to the glossary file so that linguistic translator can refer to the comment from engineers when translating the string. This can allow linguistic translator to pick a translation for the correct context. If the strings file is not in the format above, it is a bug in the software, as CoreFoundation sometimes refuses to read such files in run time (missing semicolon is a common problem). AppleGlot will record such problem as warning in the log.

.nib files (Cocoa/Carbon UI files)

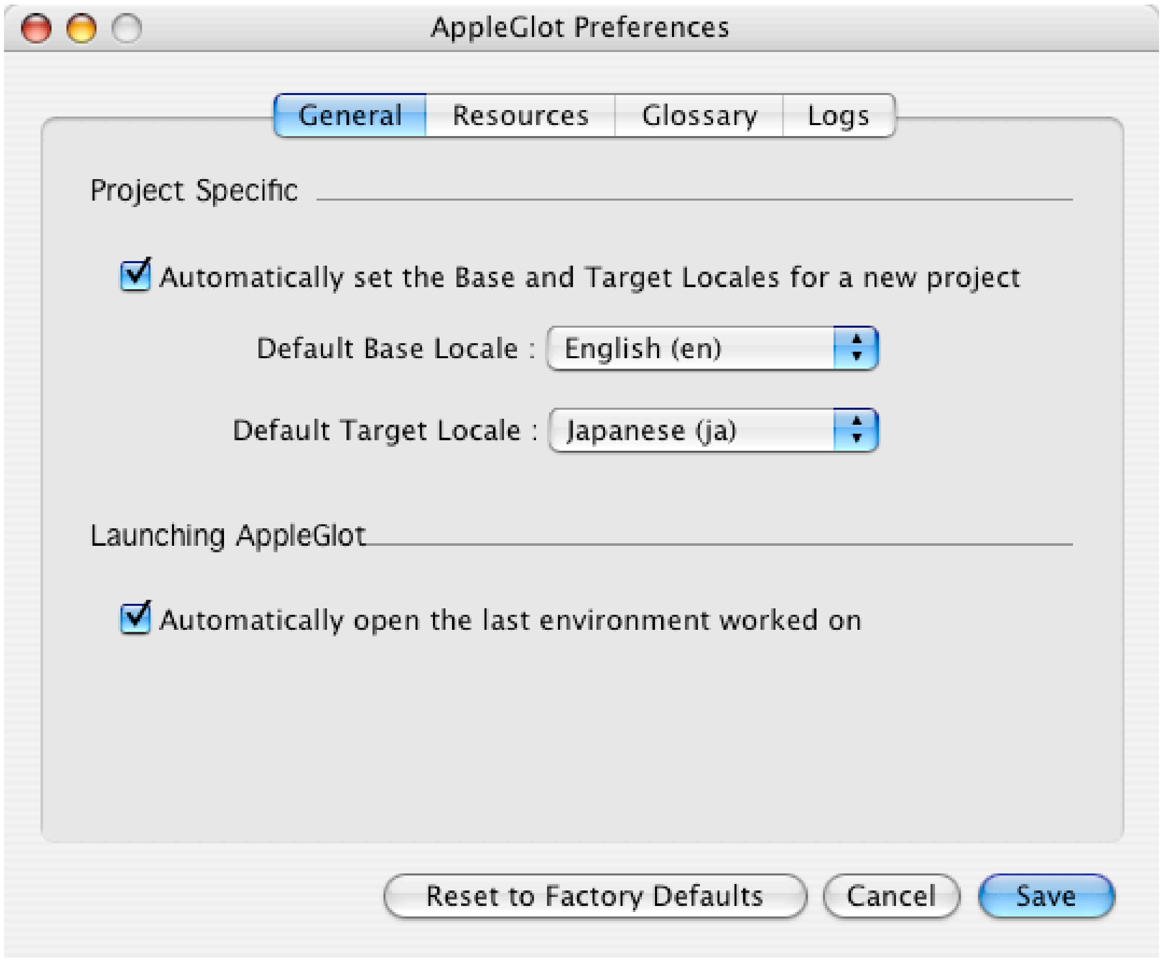
MyDocument.nib

AppleGlot uses a utility called nibtool to extract strings and leverage layout information from Cocoa and Carbon nib files. The tool needs to be installed to /usr/bin. AppleGlot creates temporary files (.nib.strings) for each .nib file to process the strings in .nib files. .nib.strings files will not be removed until you select the "Final Pass (remove temporary working files)" from the "Actions" menu. nibtool requires objects.nib (or keyedobjects.nib), info.nib and classes.nib in a nib to extract data. (The info.nib and classes.nib are often stripped from shipping products from Apple, as it is unnecessary for run-time operation. This means AppleGlot cannot process nib files from some products shipped by Apple.)

AppleGlot Preferences

From the AppleGlot menu, select "Preferences...". The following dialog allows you to change some of the default settings:

General Tab:


AppleGlot Preferences

Project Specific

✓ Automatically set the Base and Target Locales for a new project

Default Base Locale : English (en)

Default Target Locale : Japanese (ja)

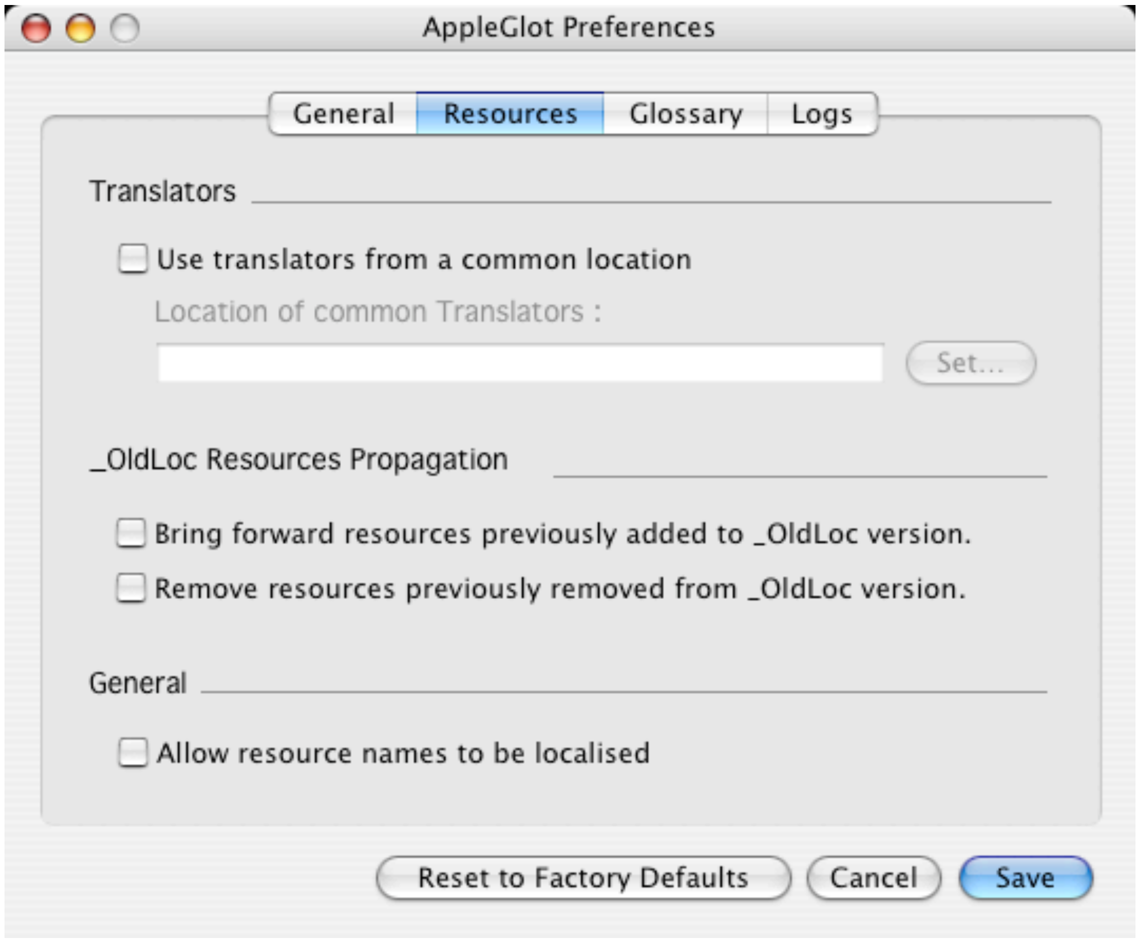
Launching AppleGlot

✓ Automatically open the last environment worked on

"Automatically set the Base and Target Locales for a new project"
This check box allows you to select default base and target locales for any new projects. If you change "Default Target Locale" once, you will not need to change your base and target locales from the "Tools" menu.

"Automatically open the last environment worked on"
This check box allows you to open the last AppleGlot environment you worked on every time your launch AppleGlot. This is selected by default.

Resources Tab:


AppleGlot Preferences

General | Resources | Glossary | Logs

Translators

☐ Use translators from a common location

Location of common Translators :

Set…

_OldLoc Resources Propagation

☐ Bring forward resources previously added to _OldLoc version.

☐ Remove resources previously removed from _OldLoc version.

General

☐ Allow resource names to be localised

Reset to Factory Defaults | Cancel | Save

"Use translators from a common location"

If you are working with a large number of AppleGlot environments, you should consider preparing a common location where AppleGlot can look for Monte translators. By specifying a common location using this check box, you would not have to keep the same set of Monte translators in multiple environments. If you have specific Monte translators that need to be used for certain environment, you can place the special Monte translators to the _Translators folder of that environment.

"Bring forward resources previously added to _OldLoc version."

By checking this check box, any extra resources in Carbon resource files you intentionally added to _OldLoc software will be carried over to _NewLoc.

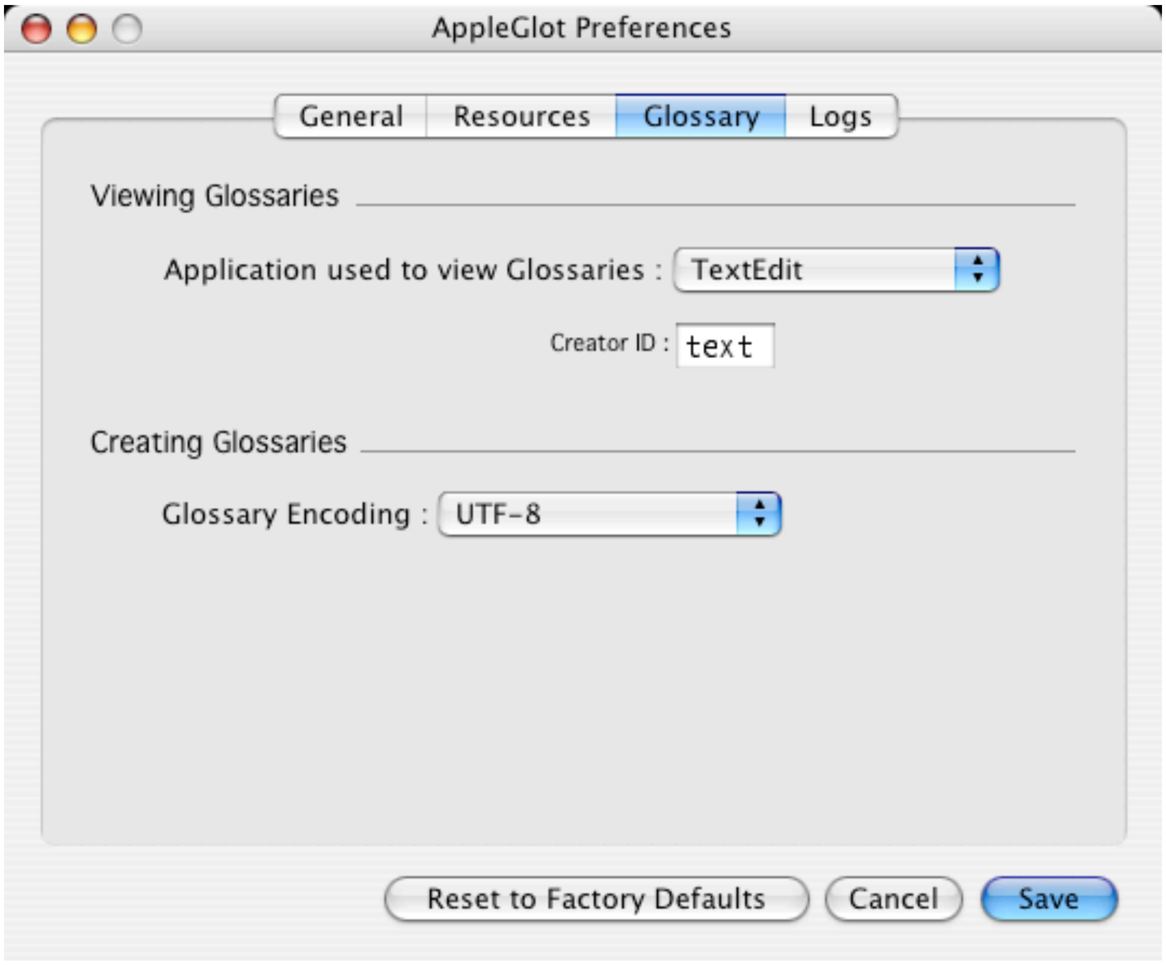
"Remove resources previously removed from _OldLoc version."

By checking this check box, any resources you intentionally removed from Carbon resource files in _OldLoc software will be removed from _NewLoc also.

Note: *Even though you have added / removed extra entries in .strings files, AppleGlot does not handle it the same way it does with Carbon resource files.*

"Allow resource names to be localized":
If your software requires you to localize name of the resource to be localized, check on this check box. AppleGlot will be extracting name of resources to be localized in the Glossary files.

Glossary Tab:



Application used to view Glossaries : TextEdit

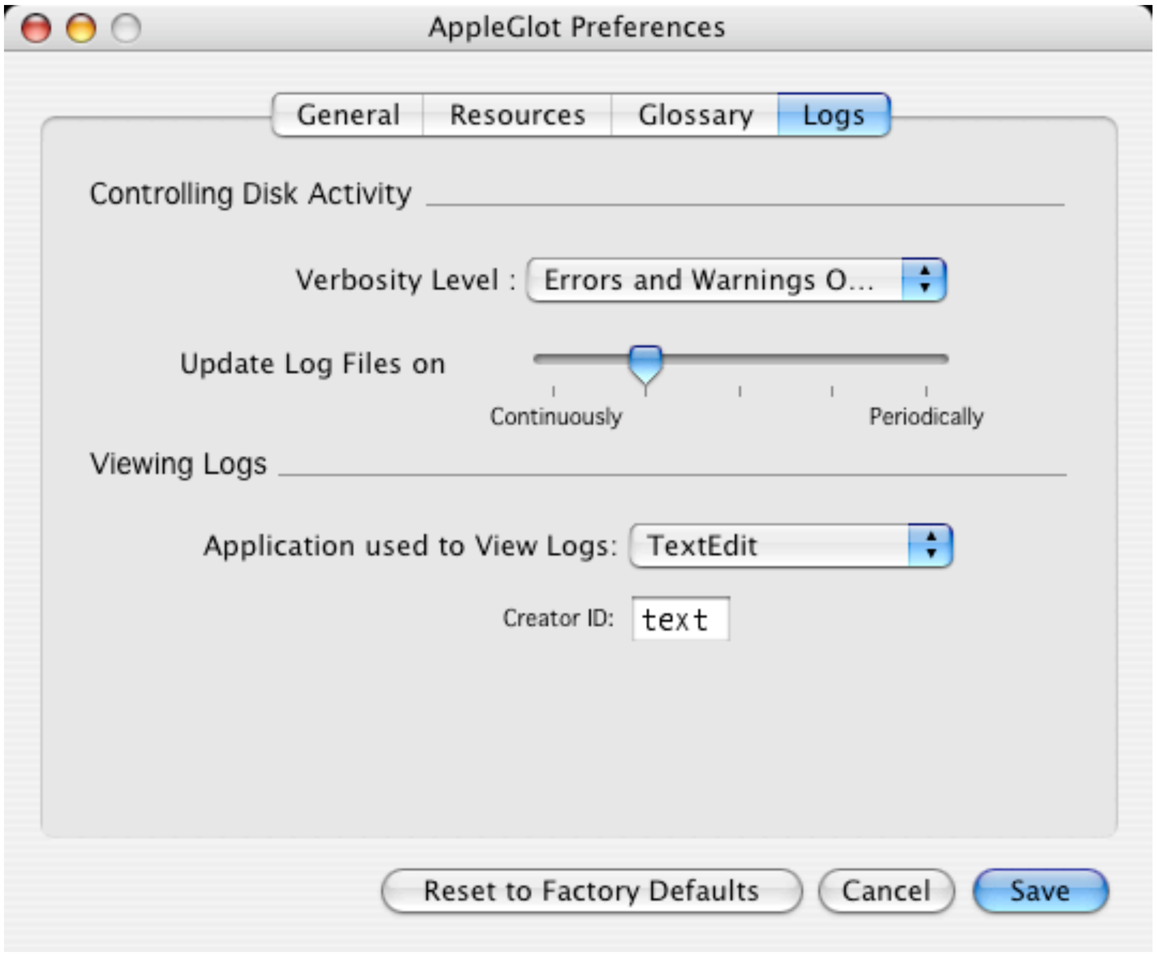
Creator ID : text

Glossary Encoding : UTF-8

"Application used to view Glossaries"
For editing and viewing Glossary files, you are able to select your favorite text editor. You can also set the creator ID of the glossary files.

"Glossary Encoding"
You are able to select encoding of the glossary files. By default, it is set to UTF-8, and should not be changed as glossary files will not be valid XML file any longer (XML must be saved in UTF-8). You can alternatively set to UTF-16 because more text editors handle UTF-16 much more fluently.

Logs:



AppleGlot Preferences

Controlling Disk Activity

Verbosity Level : Errors and Warnings O...

Update Log Files on: Continuously — Periodically

Viewing Logs

Application used to View Logs: TextEdit

Creator ID: text

"Verbosity Level"
You can set Verbosity Level for AppleGlot log, if you record more verbose logs, the performance of AppleGlot will decrease drastically. It is recommended to set the verbosity level to "Errors and Warnings only".

"Update Log Files on Disk"
This is same as "Verbosity Level". It is not recommended to make the log files to frequently.

"Application used to view logs"
You can set your prefered XML viewer.

Basic AppleGlot phases

This is a quick overview of the basic phases of AppleGlot:

1. Creating an AppleGlot Environment. This is done by using the "Create Empty Environment..." menu item in the "Tools" menu. In the "AppleGlot: Choose Folder to be an Environment" dialog box, select a folder to be used as an AppleGlot environment. Several folders are created in the selected folder.

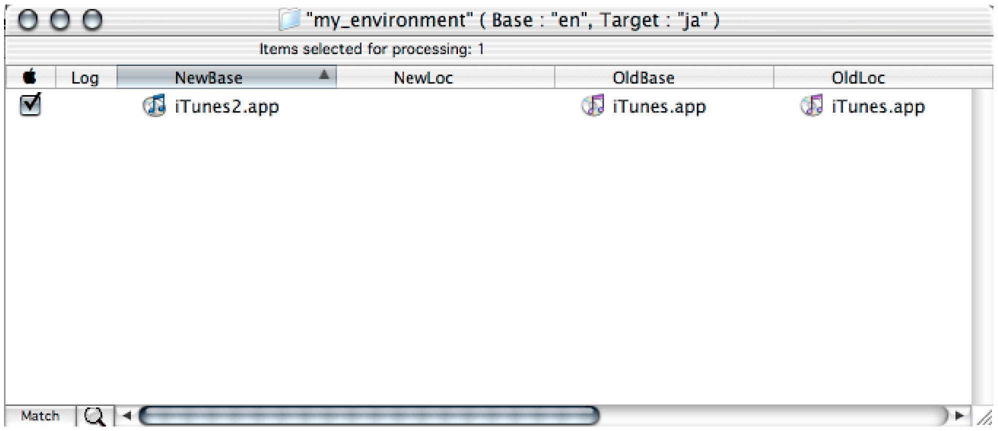
2. Place your source files into the Environment folders. This takes only a minute or so using Finder. Just put copies of new software into the _NewBase. Place old base software if any, and previously localized software into _OldBase and _OldLoc folders into the Environment.

Note: *When you are placing software bundles into these folders, it is advised to keep the bundle structures identical among these for maximum leverage.*

Place Monte Translators into "_Translators" folder in your AppleGlot environment. AppleGlot uses Monte translators to analyze contents of Macintosh resource files (.rsrc). If you prefer to use a set of Monte translators from a common location, you may do that by selecting the location from the Resource tab in AppleGlot preferences.

3. Open an AppleGlot Environment. Select "Open..." from "File" menu to select an AppleGlot environment folder. Contents of _NewBase, _OldBase, and _OldLoc are analyzed and items in these folders will be associated together. AppleGlot then displays the associated items in the Environment in the rows.

You will then need to click the check boxes to put the check marks next to those files you wish AppleGlot to work on.


"my_environment" (Base : "en", Target : "ja")
Items selected for processing: 1
Match

	Log	NewBase	NewLoc	OldBase	OldLoc
✓		iTunes2.app		iTunes.app	iTunes.app

4. Initial Pass (create NewLoc, using _OldLoc if available) - Text extraction and item preservation occurs in the initial pass of AppleGlot. Use the "Initial Pass (create NewLoc, using _OldLoc if available)" menu item in the "Actions" menu. This process usually takes a few minutes. The resulting localized files will appear in the _NewLoc folder. If previous files were placed in _OldBase and _OldLoc, new files in _NewLoc may contain mixed text -- partially from the _NewBase and partially from the _OldLoc.

5. Manual text translation using your favorite text editor - AppleGlot produces the Work Glossary files in the _WorkGlossary folder in the environment after the Initial Pass. Application Dictionary files are also created in the _ApplicationDictionaries folder. Contents of the Application Dictionary file always reflects the latest strings in the files in _NewLoc at any given stage. Linguistic translators might want to reference this file to maintain consistency of the glossaries in the software. You may consider using translation memory tools to reduce the time for manual translation. Please open and save your Work Glossary files in UTF-8 encoding so that AppleGlot can read them. You might want to use ADViewer for editing the glossary files.

6. Incremental Pass (update NewLoc from Glossaries) - Use the "Incremental Pass (update NewLoc from Glossaries)" menu item in the "Actions" menu to move your translations in the Work Glossary files into the files in _NewLoc to replace the mixed text. This process often takes less time than Initial Pass because only those resources listed in the Work Glossary files are processed and updated. You can edit Applications Dictionary files or Work Glossary files to correct translation problem, then run the Incremental Pass to insert corrected translations to localized files in _NewLoc.

7. Final Pass (remove temporary working files) - This pass is almost the same as the Incremental Pass, but it also removes temporary files like .nib.strings files and ~.nib files created during the previous passes. Use the "Final Pass (remove temporary working files)" menu item in the "Actions" menu. If you perform another incremental pass (either Incremental Pass or Final Pass), temporary files will be created again, which would slow down the process. It is recommended to use this pass when you are sure that all strings in the application are translated correctly.

Usually, once the above phases have been completed, AppleGlot portion of the localization process is finished. You must then use a resource editor or other specialized tools to adjust the visual appearance of the translated text and to do any other specialized localization work. This includes localizing files that are not supported by AppleGlot.

The advantage of using AppleGlot is that most of your work is only needed to be done once. Later versions of the same software will only require you to use AppleGlot to update the files, and you will only required to deal with the items that have changed.

***Note:** AppleGlot will compare files in _NewBase and _OldBase. If they are the same, AppleGlot will just copy localized file in _OldLoc to _NewLoc. If the file was one of the supported file type, AppleGlot will try to perform incremental localization. However, if the file was not supported by AppleGlot (i.e, .rtf, .plist, etc), AppleGlot will copy the file in _NewBase to _NewLoc. In this case, localizers will need to localize the file manually from scratch.*

Basic AppleGlot phases in detail

Below is a much-expanded version of the "Basic AppleGlot phases". The purpose of this section is to show each step in more details for user who hasn't seen AppleGlot before. This will help any user understand how AppleGlot works, and what to do in each step to successfully accomplish your localization project.

STEP 1: CREATING THE REQUIRED ENVIRONMENT

AppleGlot requires a specific set of folders consisting of an environment folder containing a set of specifically named folders. You may manually create this environment using the Finder or from Terminal from scratch, however it is much easier to create AppleGlot environment automatically by selecting "Create Empty Environment..." in the "Tools" menu. You will be asked to identify the destination for the environment folder using a standard file dialog.

Note: *Generally, a new environment should be created for each project. To avoid confusion, you SHOULD NOT try to reuse old Environments! The logic relies on certain pre-existing files or being newly created by AppleGlot so if you try to save time by reusing an environment, and if it already contains some NewLoc files or some glossary files, AppleGlot may become confused and you will end up losing time.*

STEP 2: POPULATING THE ENVIRONMENT

The Environment folder will contain the following folders:

- _ApplicationDictionaries
- _LanguageGlossaries
- _Logs
- _NewBase
- _NewLoc
- _OldBase
- _OldLoc
- _Projects
- _Rules
- _Temporary
- _WorkGlossary

Once your environment has been created, the Finder is ideally suited to populating the folders. Use the Finder to move copies of your software into the proper folders.

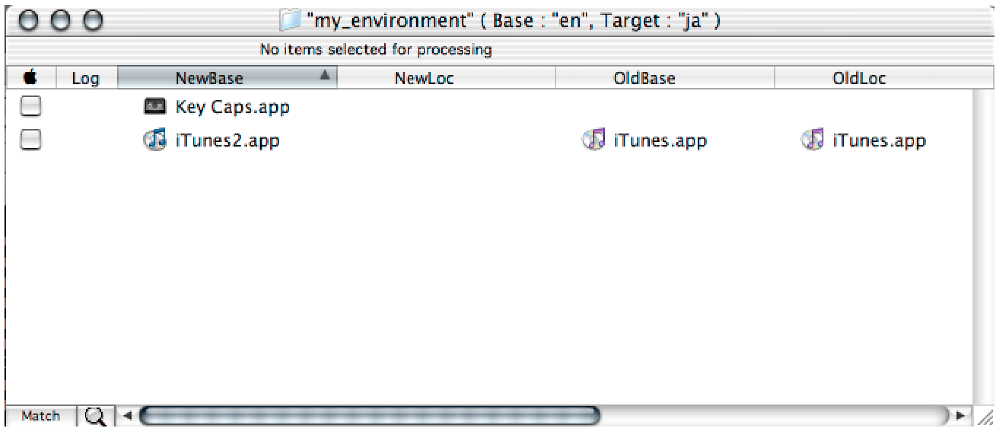
If you are localizing a new piece of software, you only need to move a copy of that software into the _NewBase folder. If you are localizing a new version of a previously localized software, you should move copies of the previous base software into the _OldBase folder and a copy of the accompanying localized version into the _OldLoc folder.

Multiple individual pieces of software may reside within a single environment folder. In most cases AppleGlot will automatically associate the correct files using name of file / bundle, CFBundleIdentifier in info.plist, type and creator information, or ‘ftag’ resources.

Note: *Don't put an Alias to the original file / bundle into the folders. AppleGlot expects that the actual file is in the folder and will not function properly if it finds an alias in the folder.*

STEP 3: OPENING THE ENVIRONMENT

Return to AppleGlot. Select "Open..." in the "File" menu and select the environment folder you created and filled with files. The AppleGlot Environment window will now appear. The environment will be opened with the base and target locale specified in the General tab in the Preferences dialog. You can temporarily change the locales via the "Choose Target and Base Locales" dialog that could be reached from the "Tools" menu.


"my_environment" (Base : "en", Target : "ja")

No items selected for processing

	Log	NewBase	NewLoc	OldBase	OldLoc
☐		Key Caps.app			
☐		iTunes2.app		iTunes.app	iTunes.app

Note: *The actual files / bundles displayed were those AppleGlot found in the specified environment. If you then switch back to the Finder and move or trash the files in your environment, AppleGlot won't detect that and will continue to display the same list. You can select "Refresh" from the "File" menu to reflect the actual contents of the opened environment.*

How auto-associations are made among _NewBase, _OldBase and _OldLoc

In general the process goes like this. First, AppleGlot locates all the files / bundle in the _NewBase, _OldBase and _OldLoc folders and fills in the NewBase column. Next, it auto-associates as many of the files/bundles as it can to fill in the OldBase and OldLoc columns in the AppleGlot window.

AppleGlot will fill in the OldBase and OldLoc columns through auto-association based on the file name or the name of the bundle. If file names were localized in OldLoc, AppleGlot tries to look for the ftag resource to associate them together.

In case of bundle, if your bundle contains Info.plist file in the Contents folder, AppleGlot will look for **CFBundleIdentifier** in the Info.plist file to match the bundles. If AppleGlot doesn't find CFBundleIdentifier in Info.plist, it will try to match included files using full paths from the root of _NewBase, _OldBase and _OldLoc folders. It is recommended that the structure of bundles in _NewBase, _OldBase and _OldLoc to be identical and to include info.plist files in your environment.

Files / bundles that were not auto-associated are not yet displayed in the AppleGlot window. In order to include any of these files that have not been auto-associated, you can ctrl-click on the OldBase and OldLoc columns to manually select a file / bundles that needs to be associated.

How to correct auto-associations
If AppleGlot incorrectly associated file / bundle in OldBase or OldLoc column, you can ctrl-click on the OldBase and OldLoc columns, and select "Break Match" from the context menu. This will break the association and AppleGlot will not use files in OldBase or OldLoc to process files. You can ctrl-click again on the OldBase and OldLoc columns to manually select a file / bundles that need to be associated.

STEP 4: Initial Pass (create NewLoc, using _OldLoc if available)

As you click in the check boxes in the AppleGlot window, check marks will appear or disappear in the left hand column. If you want to check all check boxes, you can select "Tick all" from the Edit menu. You may also select the range of the software by dragging with the mouse, then select "Tick all" from the Edit menu to tick only the items in the selected range.

Once appropriate check boxes are ticked, select "Initial Pass (creating NewLoc, using _OldLoc if available). It will create partially localized versions of _NewBase in the _NewLoc folder if _OldBase and _OldLoc are available. Symlinks in the bundle will be preserved as well.

New or changed strings will be extracted to the Work Glossary file in _WorkGlossary folder. Strings that are not changed from previous localization will be extracted to Application Dictionary file in _ApplicationDictionaries folder.

If you place .lg files (.ad files from previously localized software renamed to .lg) in _LanguageGlossaries folder, AppleGlot will try to insert the translations from the .lg files in WorkGlossary files to reduce the number of strings to be translated.

STEP 5: Manual text translation using your favorite text editor

Use your preferred text editor to translate the text in the Work Glossary files (found in the _WorkGlossary folder). The Work Glossary files are in XML format, which needs to be opened and saved in UTF-8 encoding. You may use translation memory tools to automate the translation. For more information on the glossary files, please see "AppleGlot3 Glossary files". You might want to use ADViewer for editing the glossary files.

STEP 6: Incremental Pass (update NewLoc from Glossaries)

To insert the translated strings in Work Glossary files to the files in _NewLoc, return to AppleGlot, and select "Incremental Pass (update NewLoc from Glossaries)" from the "Actions" menu.

Both Work Glossary file and Application Dictionary file will be updated. All translated strings in Work Glossary file will be moved to the Application Dictionary file to reflect the localization state of files in _NewLoc.

STEP 7: OPTIONAL - Making translation changes in the Application Dictionary

AppleGlot is designed as a large-scale localization tool. All text requiring translation is extracted to the Work Glossary, and once that is translated, all you need to do is to run the incremental pass to complete the NewLoc file.

However, once you complete the NewLoc file, you may notice spelling errors, or for some reason or another want to change some of your translations. Some people decide not to use AppleGlot to make small changes and find it easier to make the changes using resource editors like Interface Builder or Resorcerer®.

Note: *Once you have used resource editors, you cannot go back to AppleGlot without the possibility of AppleGlot overwriting text changes you made in the resource editors!*

Others choose to keep on using AppleGlot to make all changes that are needed to the text. This has the advantage of keeping the Application Database up to date with the released software. A complete and updated Application Dictionary is useful for "proofing" manuals to ensure both the software and manuals use the same terms.

How to read AppleGlot logs

AppleGlot records any problem to log files generated in the _Logs folder of the AppleGlot environment. AppleGlot generates two files for each item it processes. The one with the ".counts" extension has word count information in XML format. The other file keeps records of nibtool output, or any Monte output that might be interesting to see after the processing.

Here are some examples of AppleGlot log output:

When AppleGlot encounters an invalid format XML file (Work Glossary or Application Dictionary), the following error message will be recorded in the log:

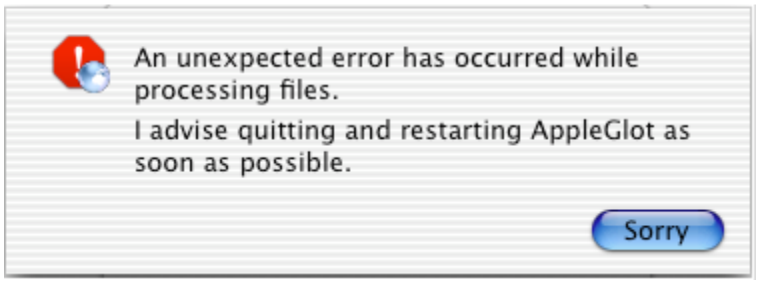
```
20020927 15:23:21 ErrK Illegal characters were encountered while try-
ing to read the Work Glossary associated with the file
"Clocks_Tier1_proj".
```

***Note:** to find the illegal character open the Work Glossary with an XML viewer such as Internet Explorer.*

When glossary files (Language Glossary, Work Glossary or Application Dictionary) are locked or have no appropriate access privilege (e.g., owner:root, mode:0600, etc.);, the following error message will be recorded in the log

```
20020927 16:06:23 ErrK Could not create Application Dictionary for
"Clocks_Tier1_proj"
```

When an AppleGlot environment is broken (for example, _Temporary directory does not exist), no error message will be recorded in the log. Instead, the following dialog box appears then AppleGlot will quit itself:


An unexpected error has occurred while processing files.
I advise quitting and restarting AppleGlot as soon as possible.

FAQ / Limitation / Issues

AppleGlot allows you to open the same environment over and over again

This is considered as a bug in this release. User should not open the same environment multiple times. This will give the impression that multiple languages can be processed from a same environment but that is not true. If you are localizing your software in other languages, create another environment.

Some AppleGlot UI is not functional

Some portion of AppleGlot UI, like context menu or inspector window are not implemented completely therefore not 100% functional. The features that are not described in this document are not supported.

FLTR support

AppleGlot will never support 68K resource display or filtering code resources

Extra / removed entries in .strings file in _OldLoc

"Bring forward resources previously added to _OldLoc version." and "Remove resources previously removed from _OldLoc version." do not apply for the entries you have manually added / removed in the .strings file in _OldLoc.

lproj naming

When AppleGlot generates <lang>.lproj folder in the _NewLoc folder, the languages for following languages will be generated in full language name:

French	(French.lproj)
German	(German.lproj)
Japanese	(Japanese.lproj)
Spanish	(Spanish.lproj)
Dutch	(Dutch.lproj)
Italian	(Italian.lproj)

All other languages will be generated using ISO 639 language code + ISO 3166 country code, such as “da.lproj” (Danish), or “zh_CN.lproj” (Mainland Chinese or Simplified Chinese). If you want to use ISO codes only for lproj names, you may rename the lproj’s after completing AppleGlot processes.

Where AppleGlot environment should be placed

AppleGlot environment should only be placed on an HFS+ volume. Also, you can not place or name an AppleGlot environment where a non-ASCII pass name is used in the path. For example, AppleGlot can not process files in the environment whose name is in Japanese.

AppleGlot Tutorial

This section provides 2 tutorials to localize Mac OS X software using AppleGlot. In the first tutorial, you will learn how to localize a software from scratch. In the latter tutorial, you will learn how to localize a software using previously localized contents.

Tutorial 1: Non-incremental localization

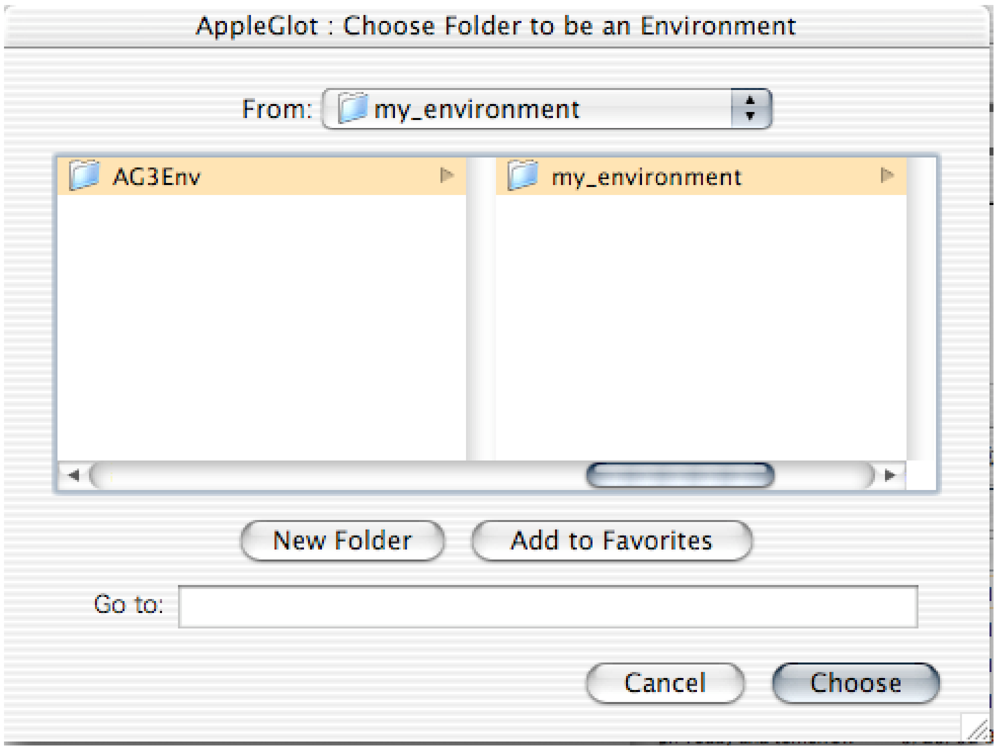
1. Double click the AppleGlot disk image. Drag the “AppleGlot” application from the mounted image to your local HFS+ volume.

2. Create an AppleGlot environment:

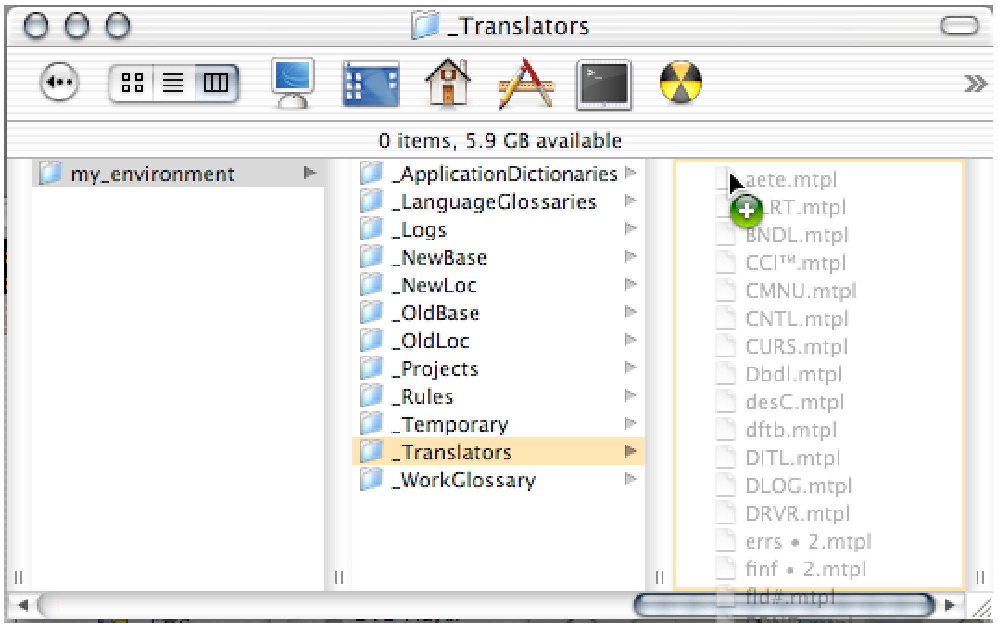
   a. Double click on the AppleGlot application.

   b. From the Tools menu, select "Create Empty Environment...". Click on the "New Folder" button to create a folder that will become your AppleGlot environment.

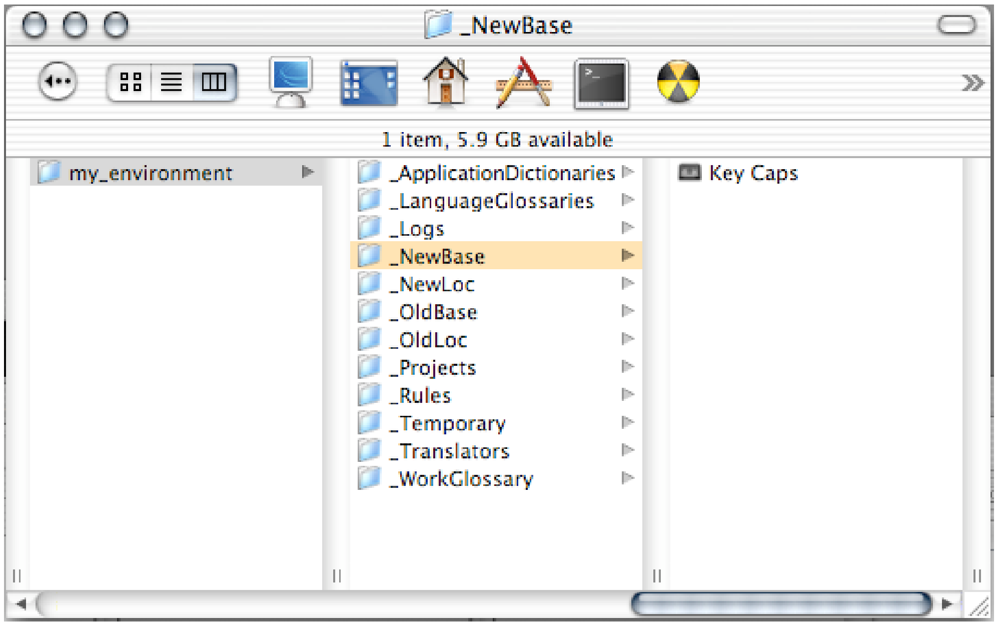
   c. Select this folder and click on the "Choose" button to make the folder into AppleGlot environment.



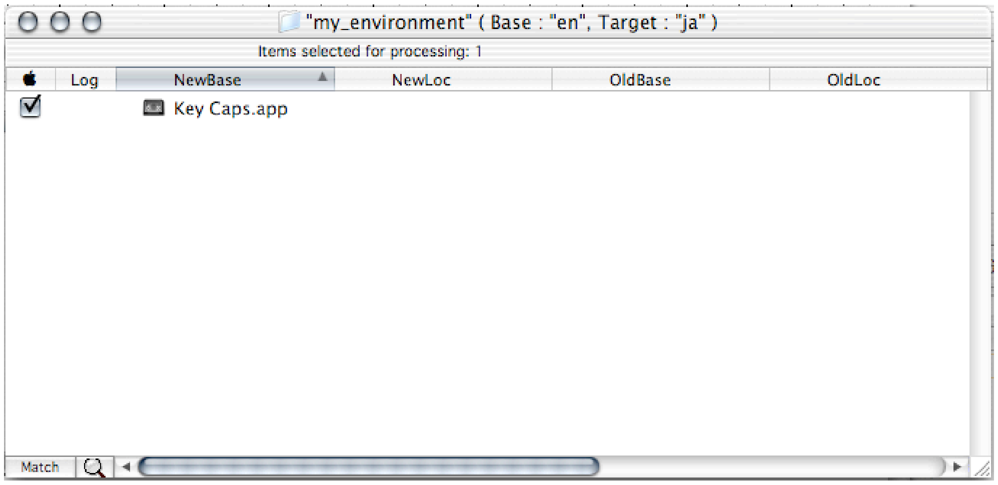
d. Drag the Monte translators to the "_Translators" folder that has just been created in the environment. AppleGlot is now set up to begin processing.



3. Place a copy of the application to be localized in the "_NewBase" folder in the AppleGlot environment.



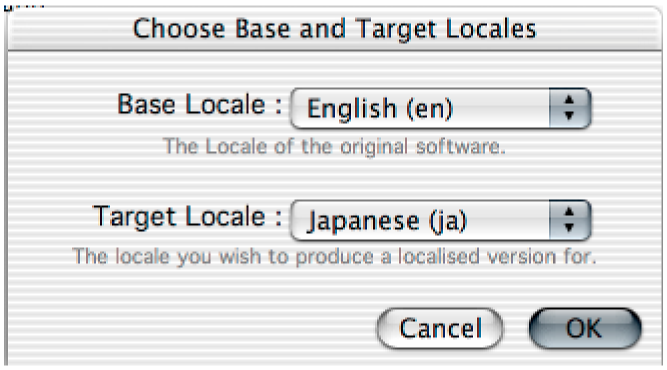
4. Check the check box next to the application(s) to be glotted.


"my_environment" (Base : "en", Target : "ja")

Items selected for processing: 1

	Log	NewBase	NewLoc	OldBase	OldLoc
✓		Key Caps.app			

5. From "Tools" menu, select "Set Target and Base Locals…". Select the base language and the target language.


Choose Base and Target Locales

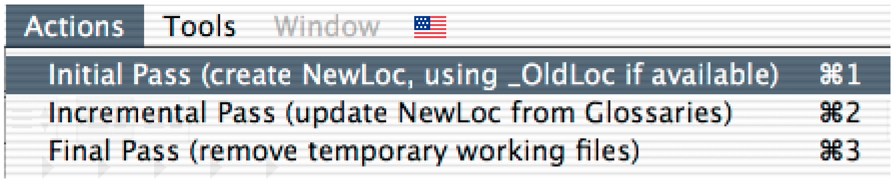
Base Locale : English (en)

The Locale of the original software.

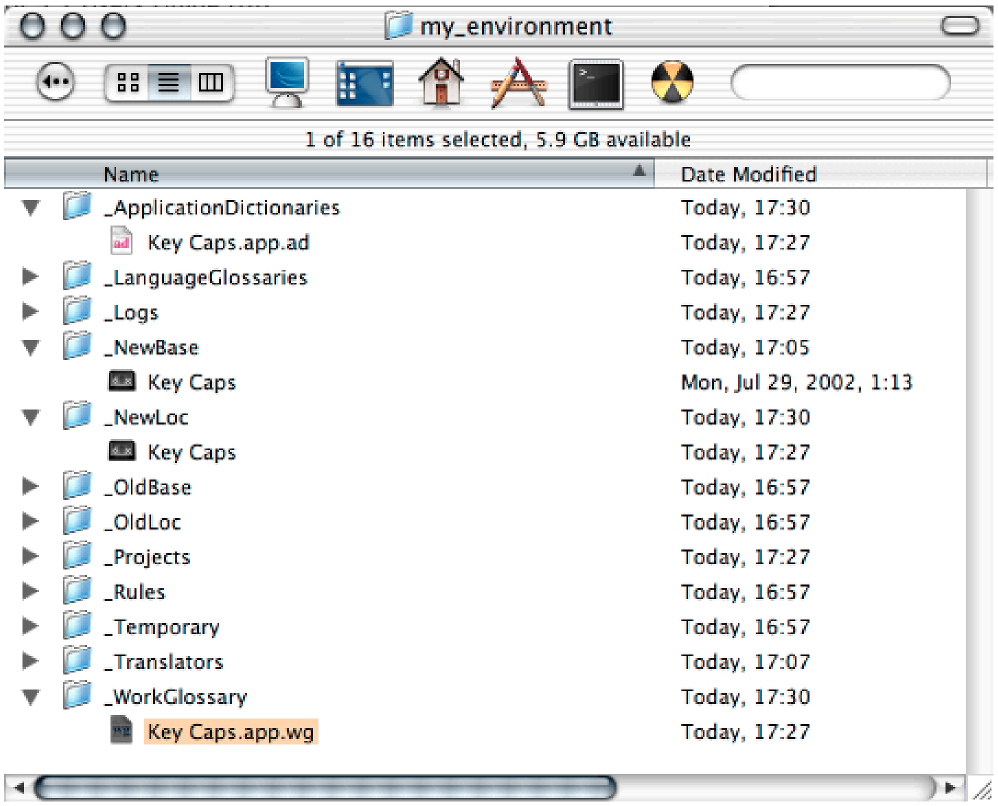
Target Locale : Japanese (ja)

The locale you wish to produce a localised version for.

6. From "Actions" menu, select "Initial Pass (Create NewLoc, using _OldLoc if available). This will extract localizable strings from the application you placed in the _NewBase folder and the files in _NewBase will be copied to _NewLoc folder. Glossary files are produced in the _ApplicationDictionaries folder and the _WorkGlossary folder.


Actions
Tools
Window
Initial Pass (create NewLoc, using _OldLoc if available) ⌘1
Incremental Pass (update NewLoc from Glossaries) ⌘2
Final Pass (remove temporary working files) ⌘3

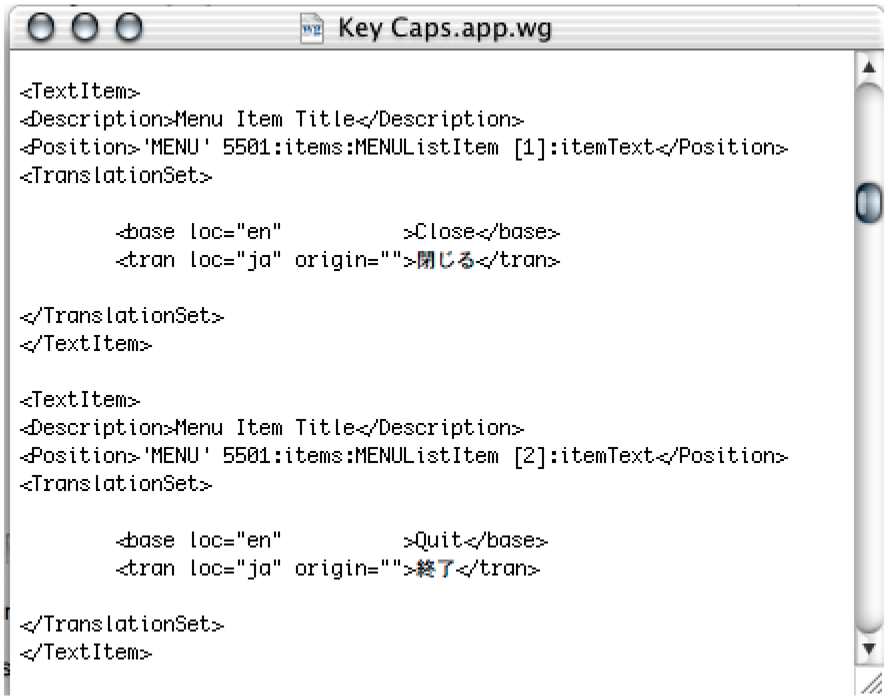
7. Open generated .wg file in _WorkGlossary folder in the environment. You can use your favorite text editor. You need to open the file as UTF-8 encoding. The .wg file should be named after the application name with a .wg extension appended to the end. This file is where the actual translation will take place. Translate this file and save it in UTF-8:



Name	Date Modified
_ApplicationDictionaries	Today, 17:30
Key Caps.app.ad	Today, 17:27
_LanguageGlossaries	Today, 16:57
_Logs	Today, 17:27
_NewBase	Today, 17:05
Key Caps	Mon, Jul 29, 2002, 1:13
_NewLoc	Today, 17:30
Key Caps	Today, 17:27
_OldBase	Today, 16:57
_OldLoc	Today, 16:57
_Projects	Today, 17:27
_Rules	Today, 16:57
_Temporary	Today, 16:57
_Translators	Today, 17:07
_WorkGlossary	Today, 17:30
Key Caps.app.wg	Today, 17:27

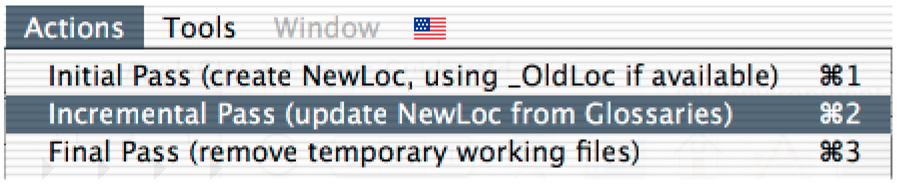
a. Choose a string to translate.

b. Insert the cursor between the <tran loc=> tag and the </tran> tag and type the translated string:



```
<TextItem>
<Description>Menu Item Title</Description>
<Position>'MENU' 5501:items:MENUListItem [1]:itemText</Position>
<TranslationSet>

        <base loc="en"          >Close</base>
        <tran loc="ja" origin="">閉じる</tran>

</TranslationSet>
</TextItem>

<TextItem>
<Description>Menu Item Title</Description>
<Position>'MENU' 5501:items:MENUListItem [2]:itemText</Position>
<TranslationSet>

        <base loc="en"          >Quit</base>
        <tran loc="ja" origin="">終了</tran>

</TranslationSet>
</TextItem>
```

8. After translation is done, return to AppleGlot. From the "Actions" menu, select "Incremental Pass (update NewLoc from Glossary). This will insert the translated strings in the Work Glossary file into the files in _NewLoc. Translated text items will be moved to the Application Dictionary after this pass.

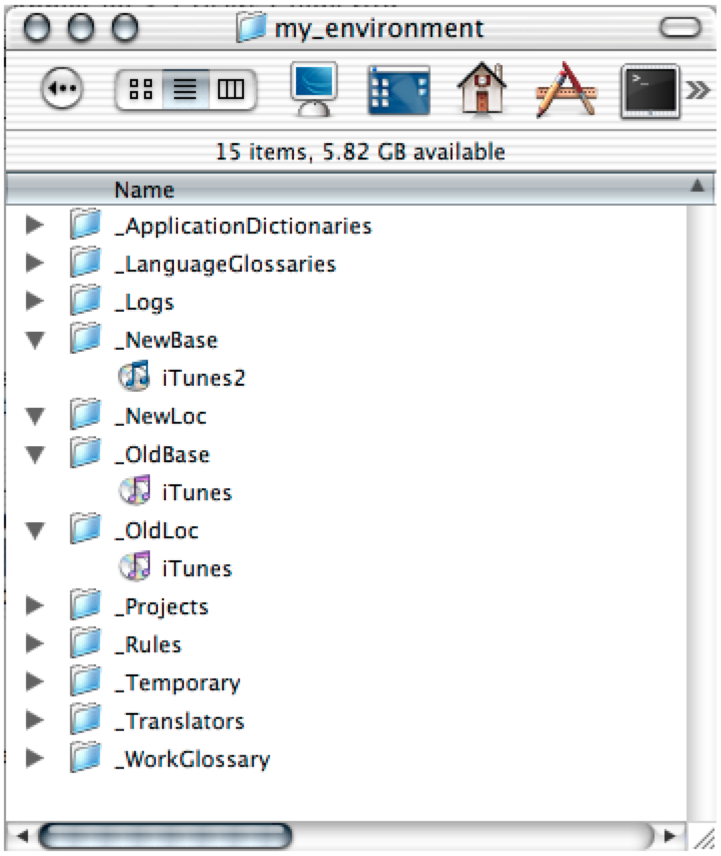

Actions
Tools
Window
Initial Pass (create NewLoc, using _OldLoc if available) ⌘1
Incremental Pass (update NewLoc from Glossaries) ⌘2
Final Pass (remove temporary working files) ⌘3

9. The translated application(s) can be found in the "_NewLoc" folder. You should adjust visual resources (such as nib files or DITL, WIND, PPob etc.) using appropriate resource editors like Interface Builder, Resorcerer® or PowerPlant Constructor.

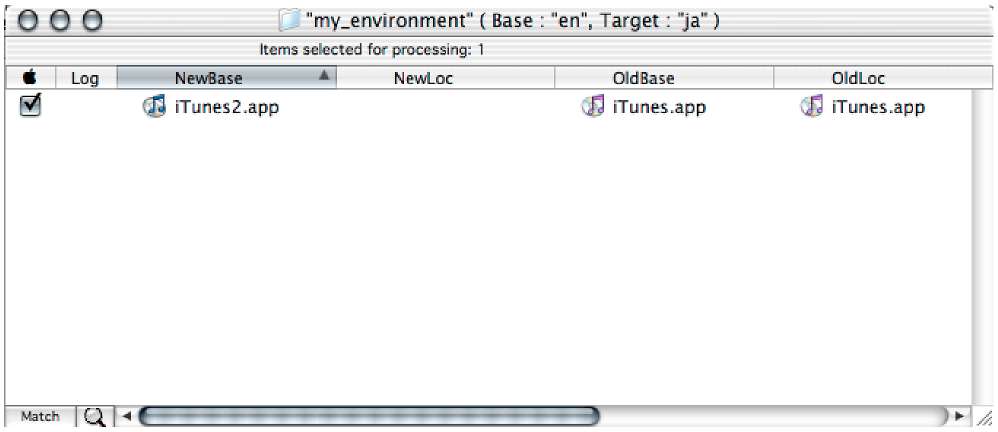
Tutorial 2: Incremental Localization

1. Follow step 1 and step 2 in Tutorial 1.

2. Place a copy of the application to be localized in the "_NewBase" folder in the AppleGlot environment. Also place an older version of the application you placed in "_NewBase" into the "_OldBase" folder. You are also required to place previously localized application to "_Old-Loc".



3. Check the check box next to the application(s) to be glotted.

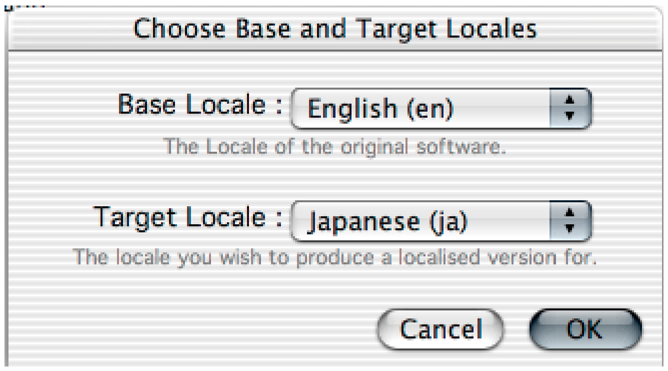


"my_environment" (Base : "en", Target : "ja")

Items selected for processing: 1

	Log	NewBase	NewLoc	OldBase	OldLoc
✓		iTunes2.app		iTunes.app	iTunes.app

4. From the "Tools" menu, select "Set Target and Base Locals…". Select the base language and the target language.



Choose Base and Target Locales

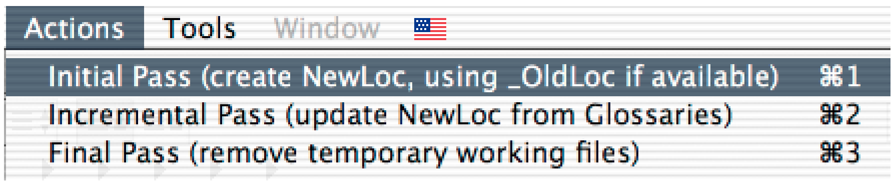
Base Locale : English (en)

The Locale of the original software.

Target Locale : Japanese (ja)

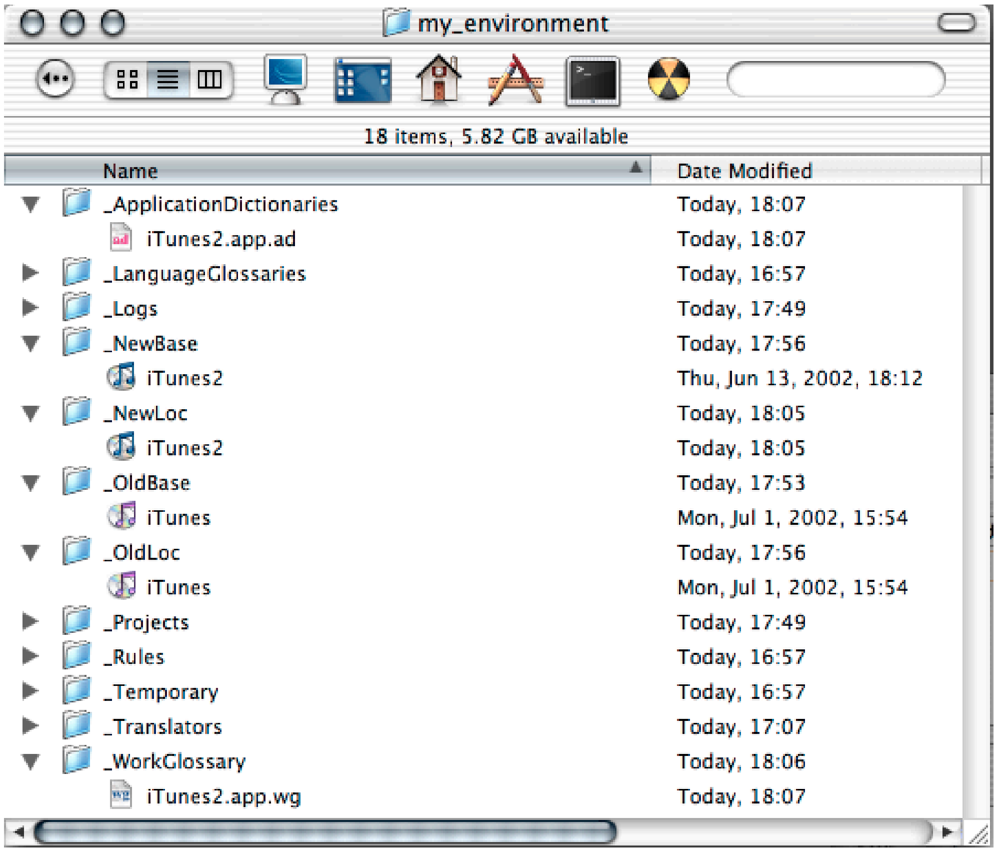
The locale you wish to produce a localised version for.

5. From "Actions" menu, select "Initial Pass (Create NewLoc, using _OldLoc if available). This will cross-reference the files in _NewBase, _OldBase and _OldLoc in order to create partially localized files in _NewLoc.



Actions
Tools
Window
Initial Pass (create NewLoc, using _OldLoc if available) ⌘1
Incremental Pass (update NewLoc from Glossaries) ⌘2
Final Pass (remove temporary working files) ⌘3

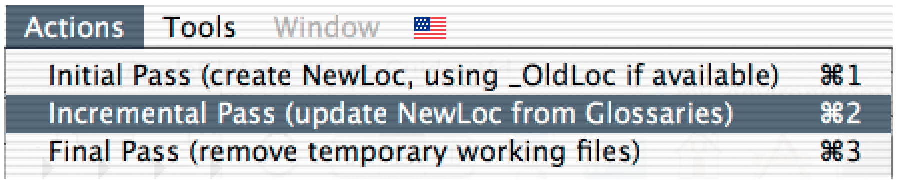
6. Only strings that are new or changed in _NewBase are required to be translated. Such strings will be extracted to the Work Glossary file in the _WorkGlossary folder in the environment. The strings that are already localized and represent a 100% match in the _OldLoc are saved in the Application Dictionary in the _ApplicationDictionaries folder. Any engineering changes (such as font, layout, properties of dialog, for example) made in previous localization will be leveraged to _NewLoc as much as possible.



Name	Date Modified
_ApplicationDictionaries	Today, 18:07
iTunes2.app.ad	Today, 18:07
_LanguageGlossaries	Today, 16:57
_Logs	Today, 17:49
_NewBase	Today, 17:56
iTunes2	Thu, Jun 13, 2002, 18:12
_NewLoc	Today, 18:05
iTunes2	Today, 18:05
_OldBase	Today, 17:53
iTunes	Mon, Jul 1, 2002, 15:54
_OldLoc	Today, 17:56
iTunes	Mon, Jul 1, 2002, 15:54
_Projects	Today, 17:49
_Rules	Today, 16:57
_Temporary	Today, 16:57
_Translators	Today, 17:07
_WorkGlossary	Today, 18:06
iTunes2.app.wg	Today, 18:07

7. Open the generated .wg file using your preferred text editor. The .wg file should be named after the application name with a .wg extension appended to the end. This file is where the actual translation will take place. Open the file in UTF-8. Translate, then save the file in UTF-8.

8. From "Actions" menu, select "Incremental Pass (update NewLoc from Glossary). This will insert the translated strings in the Work Glossary file to the files in _NewLoc. Translated text items will be moved to Application Dictionary after this pass (as they are considered to be "blessed").


Actions
Tools
Window
Initial Pass (create NewLoc, using _OldLoc if available) ⌘1
Incremental Pass (update NewLoc from Glossaries) ⌘2
Final Pass (remove temporary working files) ⌘3

9. The translated application(s) can be found in the "_NewLoc" folder. You should adjust visual resources (such as nib files or DITL, WIND, PPob etc.) using appropriate resource editors like Interface Builder, Resorcerer® or PowerPlant Constructor.

Glossary

AppleGlot environment
The set of folders with the appropriate name that AppleGlot works on.

Application Dictionary
An XML file which contains all of the OldLoc based translated string pairs in the files in _NewLoc folder. Application Dictionary files are saved in the _ApplicationDictionaries folder in AppleGlot environment. This file has “.ad” extension.

Bundle
The structure scheme that Apple recommends for developers to organize Mac OS software. This allows software to have multiple localized resources, for example.

creator ID
The ID that indicates which application created the file.

Glossary files
In this document, glossary files refers to Application Dictionary (.ad) files, Language Dictionary (.lg) files, and Work Glossary (.wg) files.

info.plist
The file that should be placed in a bundle to define various attributes of the bundle.

Language Dictionary
Application Dictionary files from the previous AppleGlot projects that could be placed in the _LanguageGlossary folder of an AppleGlot environment. These files need to have extension .lg instead of .ad. AppleGlot tries to find the exact match of a new string in the Language Dictionary file.

Monte
A resource-parsing engine included in AppleGlot. Monte is used to extract localizable data from Carbon resource files or Classic files. Please refer to MonteSDK.pdf which can be found on the ADC site.

Monte Translators
Custom data descriptors for Monte. In order for Monte to extract / insert data from certain resource types the corresponding Monte translator is required to be placed in the AppleGlot environment. Please refer to MonteSDK.pdf, which can be found on the ADC site.

nibtool
A utility program to extract / insert localizable data from Cocoa nib files. AppleGlot uses nibtool to handle Cocoa nib files. nibtool is included in developer packages and is installed to /usr/bin.

Symlinks

Also known as Symbolic links. Symlinks are files that act as pointers to other files.

Unicode

An universal character set defined and maintained by the Unicode Consortium whose goal is to include characters for all of the world's written languages. Mac OS X has Unicode as its internal encoding to develop a true multilingual, internationalized platform.

UTF-16

An encoding method for Unicode character set. The first 65536 Unicode characters are represented as two bytes, the other ones as four bytes (using surrogate). This is the encoding method that is often referred to as "Unicode". Byte Ordering Mark (BOM) is placed in the beginning of UTF-16 data to indicate the byte order of a text. The value of BOM in U+FEFF. So, if the first two bytes of the data was 0xFFFE (not defined in Unicode) that indicates the data has to be read in reverse endian. UTF-16 encoding is generally used at higher levels of the Mac OS X system.

UTF-8

An encoding method for Unicode character set. The 7 bit ASCII characters (0x00~0x7f) are encoded using 1 byte. All other characters are encoded using multiple bytes up to 6 bytes. Much of CJK characters are encoded using 3 bytes. This is currently a very popular encoding method because it has full compatibility with ASCII and XML files are encoded in UTF-8 by default. AppleGlot's glossary files are encoded in UTF-8 by default.

Work Glossary

An XML file which contains all required new strings. Human translators will work on this file to fill in the translation of the base software in the <trans> tags. Once the translations are completed, AppleGlot will insert translated strings to files in _NewLoc. Work Glossary files are saved in the _WorkGlossary folder in AppleGlot environment. This file has ".wg" extension.

Appendix A: Command line AppleGlot

With AppleGlot 3.2, a command line interface is provided. That allows the use of AppleGlot in an automated environment. The command is installed at:

```
AppleGlot.app/Contents/Resources/appleglot
```

This command provides the same functionality as the GUI version.

[1] Available commands

```
list              print list of components in NewBase
getlangs          get the current base language and target language
setlangs          set the current base language and target language
populate          create NewLoc, using OldLoc if available
update            update NewLoc from Glossaries
finalize          remove temporary working files
create            create empty environment
```

Use ‘appleglot -h command_name’ in Terminal for the detail of each command.

[2] How to tell the location of AppleGlot environment to ./appleglot

By default, ./appleglot recognizes the current working directory as the top directory of the environment. ./appleglot has -d option to know the location of environment also. So, you can choose one of the following ways to run ./appleglot:

```
a) % cd /path/to/an/env; appleglot a_command ...
b) % appleglot -d /path/to/an/env a_command ...
```

[3] Example of an ./appleglot session

Consequently, a typical usage of ./appleglot would be something like:

```
% cd /path/to/an/ag3env
% appleglot list
iPodDriver_Tier1_proj
iTunes_Tier1_proj
% appleglot setlangs en fr
% appleglot populate iTunes_Tier1_proj
% appleglot update iTunes_Tier1_proj
% appleglot finalize iTunes_Tier1_proj
```