Ada Format

Overview

The Ada Format utility is a simple program that uses the Ada Information Server (AIS) version 1.31 or later to format Ada source files according to a predefined style.

In this preliminary version the style is merely to

1. Change all keywords to lower case
2. Capitalize the e in numeric literals
3. Replace the representation of all identifier usage with that of their definition.

For example, if a variable was defined as My_Variable but then used as my_variable, Ada compilers would not complain. However this utility would modify the usage to be the same as the definition.

The utility takes up to four parameters, supplied in any order:

1. The name of a file containing configuration information
2. Source specification
3. Destination folder
4. Optional keyword *Overwrite*

Configuring Ada_Format

The configuration file supplied to Ais_Format is a standard windows ini file. If only the filename and extension is provided the file is assumed to be either within the normal search path of windows initialization files or in the same folder as the client executable.

If a configuration file is not specified then the configuration file Ada_Format.ini is assumed.

The structure of the file consists of a single section called Ais_Client followed by a number of keywords assigned a value.

For Example:

```
[Ais_Client]
Server=D:\Iltis\tools\Ais_95.exe
Pipename=Ais_Pipe
Pipelength=2000
Timeout=60
Compiler=Aonix
Path=W:\Iltis\Mx1\Src;T:\Src;B:\Libraries\Iltis\Src;B:\Libraries\Bindings\Src;S:\Aonix;
Errors=1
Units=Matching
```

Keyword definitions

- Server - Full file specification of the AIS server to use.
- Pipename - Name of the Windows named pipe to be used in Client/Server communication.
- Pipelength - Maximum transfer size using the named pipe. If not provided a default length of 1000 is used.
- Timeout - Number of minutes the AIS server should remain active after being started or following a service request. Maximum 2'184 minutes.
  If not specified then the server will remain active until explicitly closed by

the client. A value of zero specifies that the server should not remain active after use, however this value is not recommended.

- Compiler - Specifies the compiler being used.
  This indicates which compiler extensions should be recognized.
  - Aonix
  - Gnat
- Path - Details the search path the server should use to locate files.
  Folders are specified in the order they are to be searched, separated by semicolons (;). A folder specification terminated by * indicates all files in all nested subfolders. For example D:\Ada*
  Note: The last folder in the path should be the compiler specific folder (For Aonix, these files are available from the White Elephant web site)
- Units - Defines how Ada packages are stored in files
  - Matching - Packages are stored one per file with the file name corresponding to the file name
  - Single - Packages are stored one per file however the file name may differ from the package name
  - Multiple - Standard Ada. There are no restrictions on how and where packages are stored.

  If not provided Matching Units is assumed.

Specifying the source files

Ada Format supports either a single file or the name of a folder suffixed with an asterisk.
Eg D:\Ada*

In the latter case all files in the folder with the extension ada, ads or adb are processed.
Note: Subfolders are ignored.

Note also that all files that are to be processed as well as all the files they reference must be included in the list of folders specified in the *Path* item in the configuration file (see above).

Specifying the destination

In this version of Ada Format the original source files are not overwritten. Instead a destination folder must be specified into which Ada Format will create its output files, each with the same name as the source file being processed.

If an output file already exists within the destination folder then the keyword *Overwrite* must be specified to allow the existing files to be overwritten.

Ada Format Notes:

1. Only Units=Matching is currently implemented.
   Ie. For a package to be located it is assumed to be contained in a file of the same name.
   Specifications in files with extensions .ads and implementations in files with extensions .adb
   For files with the extension .ada if the filename ends with an underscore it is assumed to contain a specification otherwise an implementation.
   The dot notation for Child packages is replaced with a minus.
   For example the specification of Win32.Winbase is assumed to be contained in the file Win32-Winbase.ads
2. Subunits are not supported.
3. Files supplied by Aonix have been modified to conform to Units=Matching and these together with the package Standard.ads are provided as part of the AIS project. These files should be copied into a folder and the folder included in the source search path.
4. Standard.ads defines specifications that are implicitly defined by the compiler and are therefore Aonix specific.
5. Setting the sever timeout value too small will result in the server perhaps restarting unnecessarily and thereby having to redo a lot of preparation work that could otherwise have been avoided. On the other hand setting the value too high will cause the server to stay around long after it has stopped being used. As the server can potentially consume a lot of memory this might be undesirable. Setting the timeout value is therefore a very personal choice and depends very much on the way a particular person chooses to work.