**Hello good sirs,**

Please note that in this ISAPI, with performance being much more of an issue than flexibility, certain values have been hard-coded which would not normally be a good idea to hard-code in a more modular application. In general, modularity and flexibility was often sacrificed for the sake of performance. Just be glad there's a config file!

**I. Project Organization**

- Header files (.h) for classes are located in the "Classes" directory

- Header files (.h) which are NOT for classes are located in "Header Files" directory

- Files containing implementation of headers are located in the "Impl" directory

- .cpp files are located in "Source Files" directory

The code is object-oriented. Class declarations are separate from class implementations. Class declarations follow the convention "ClassName.h" while class implementations follow the convention "ClassName.impl.h".

There is only one "source" file (.cpp file) – that is ISAPI.cpp. In it is defined the 2 DLL entry points for an ISAPI .dll. Notice those same 2 function names in the HTNAffiliate.def file.

Notice also that the pIsapi variable in ISAPI.cpp is APPLICATION GLOBAL – there is only one instance of this class per APPLICATION. Important to know, and keep in mind threading issues. This object must be 100% thread-safe.

**II. Configuration**

**CONSTANTS.H**

- Please be familiar with ALL of the settings in this file and what they do. This is important.

- Please be aware that these settings are COMPILED into the DLL.

- Please double-check this file before each and every build.

**CONFIG FILE**

- The config file is a text file on disk, in a location readable by the ISAPI

- The location of the config file is specified in the CONSTANTS.H file

- There should be NO UNNECESSARY WHITESPACE in the config file

- The config file has the following settings:

- PassthruUrl: the URL of the passthrough page (relative to the root, with leading '/')

- IgnoreWebDirs: requests for this dir will be ignored (relative to the root, with leading '/')

for example, suppose the value of this setting in the config file is "/XXX/YYY/" )

any requests for 'http://www.hoteltravel.com/XXX/YYY/\*' will be ignored (including all files and subdirectories)

- DebugFilePath: path to the DEBUG file. This is the destination of the output of debug diagnostic messages when the \_\_DEBUG\_\_ flag in constants.h is defined.

- WebsiteFilePath: path to the websites file. This is a file (automatically generated) that contains a list of all affiliate websites. The ISAPI compares these against the referer to determine if a given request is an affiliate request.

- ConfigFileRefreshMinutes: after reading the config file, the ISAPI will cache the config settings and values. Every N minutes it will refresh its cache as long as the web server is running and accepting requests. This setting defines the value of N. If omitted, this setting will have a default value as defined in the CONSTANTS.H.

- WebsiteFileRefreshMinutes: after reading the websites file, the ISAPI will cache the list of websites. Every N minutes it will refresh its cache as long as the web server is running and accepting requests. This setting defines the value of N. If omitted, this setting will have a default value as defined in the CONSTANTS.H.

**III. Installation (IIS 5)**

[1] Checklist:

* Check the constants.h to be sure that the settings are apropriate for the release environment.
* Especially check the "LOCATION\_" setting. Is it correct for the target environment?
* Check the build settings of this project; are they correct? is the target platform (64-bit vs. 32-bit) correct?
* Is the build config in Release mode or Debug mode?

[2] Check the location of the config file. If one does not exist, create one with the appropriate settings in it. See example config file below.

EXAMPLE: (copy/paste)

PassthruUrl=/agents/passthrough.asp

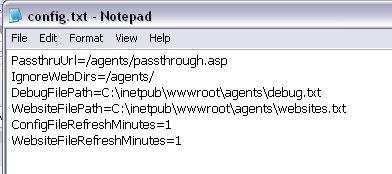
IgnoreWebDirs=/agents/

DebugFilePath=C:\inetpub\wwwroot\agents\debug.txt

WebsiteFilePath=C:\inetpub\wwwroot\agents\websites.txt

ConfigFileRefreshMinutes=10

WebsiteFileRefreshMinutes=10



[3] Compile the DLL, and move the compiled DLL to where you want it to live. Make sure that the permissions on the DLL are set such that IIS will be able to read it.

[4] Open inetmgr. Go to the properties of the website in which you want to install the ISAPI.

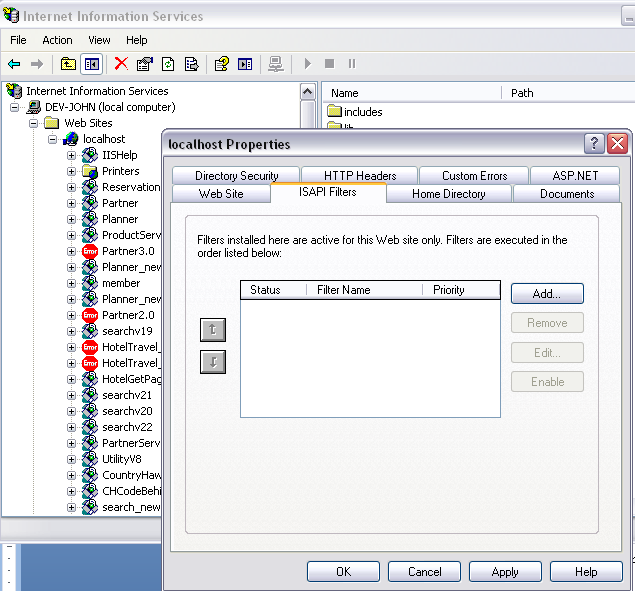
- Go to the ISAPI Filters tab

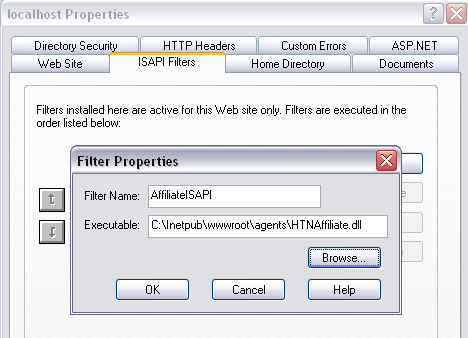
- Click the Add button

- For "Filter Name", give the ISAPI any name you want

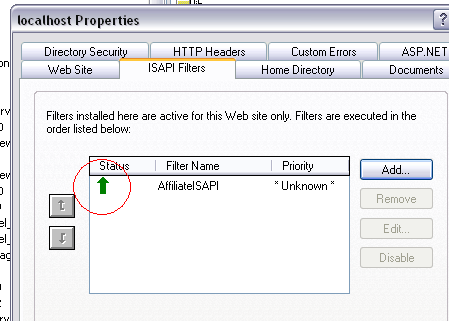
- For "Executable", fish for the path to the ISAPI dll from step 3

- Click "OK", "Apply", "OK", etc.

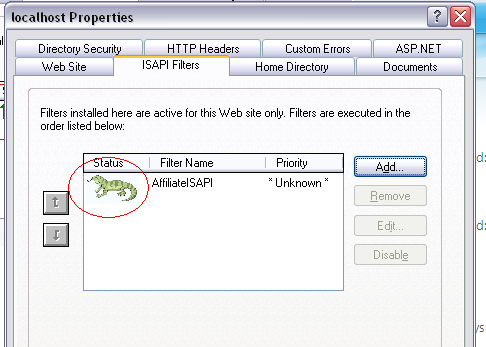




[5] Restart IIS. After IIS restarts, wait for your website to crash. If no crash, then probably nothing has gone horribly wrong. Check in IIS Manager under "ISAPI Filters" see that that the ISAPI you added in the previous step, has a green arrow pointing upwards to the left of its name. If so, then everything should be good.



**GOOD!!!**



**NOT GOOD!!!**

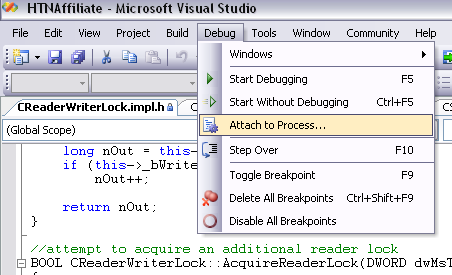
**IV. Debugging (IIS 5) with Visual Studio**

It's possible to debug the ISAPI using breakpoints in Visual Studio, if you compile in Debug mode and attach a debugger.

[1] Follow the steps above for "Installation", making sure that the ISAPI is compiled in DEBUG build mode. (If built in Release mode, the debugger will not attach)

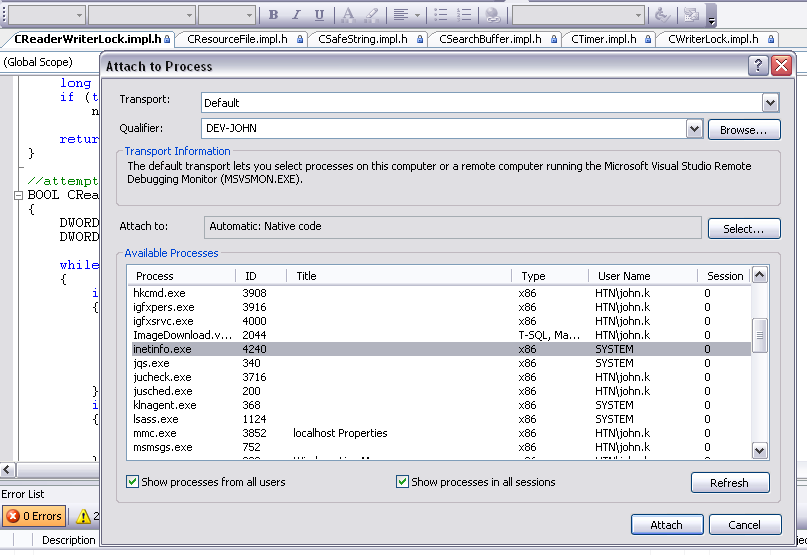
[2] Make sure IIS is running, and the green arrow is pointing UP (see under "Installation" above).

[3] Go to Visual Studio. Under the "Debug" menu, choose "Attach to Process".



[4] Find and select "inetinfo.exe" from the list. If not present, it may be that IIS is not running. **Note: make sure those 2 checkboxes at the bottom are both CHECKED** ("Show processes from all users" and "Show processes from all sessions") Otherwise you may not see the inetinfo.exe process.

[5] Click the "Attach" button.



[6] Now, the debugger has been attached. Set a breakpoint somewhere in your code. Keep in mind that HttpFilterProc is the entry point of the application; it will (should) run for every request to your website if your ISAPI is installed correctly.

[7] Make an HTTP request to your website (using a browser for example). Did the breakpoint hit?