Writing plug-in filters for Termite

Termite supports plug-in filters that allow you to filter, insert and modify any incoming or outgoing data. Filters allow you to give different representations of the data —for example in a graph, or to add functions that Termite does not support out-of-the-box.

To write a filter, you must create a DLL with a set of exported functions. The main purpose of this document is to describe those functions. For particular filters, tighter interaction with Termite is needed, and for those Termite defines two messages that your filter can send.

The extension of the DLL must be ".FLT" for Termite to recognize it as a filter. You should copy the DLL into to same directory as where the Termite program is itself.

If a filter does not use a particular function, it may omit it completely. For example, if your filter does not do anything special in function `flt_Unload()`, you may leave it out of the filter. However, a filter must have at least one of the following functions: `flt_Receive()`, `flt_Transmit()`, `flt_Process()` or `flt_HotKey()`. A filter that lacks all four of those, will not be loaded by Termite.

Functions

`BOOL flt_Load(HWND hwnd, LPCSTR ProfileName, int Build)`

This is the first function that Termite calls, after having loaded the filter DLL in memory.

`hwnd`	The handle to the main window of Termite.
`ProfileName`	The full path to the INI file that Termite uses. The filter can use this name to store its configuration.
`Build`	The build number of Termite, for distinguishing versions of the Termite application.
Return	This function must return TRUE if it can load successfully. If the function returns FALSE, Termite will unload it.
Notes	To create a (toolbar) window in Termites interface, the plug-in filter must send the message `UM_PLUGINWINDOW` to the Termite main window from its `flt_Load()` function. For example: `SendMessage(hwnd, UM_PLUGINWINDOW, nnn, 0L);` The `wParam` parameter ("nnn" in the above example) is the height of the window (in pixels). The return value of the `SendMessage()` call is the window handle. Typically, a filter will subclass this window in order to receive notifications for any controls that it creates in it.

`void flt_Unload(void)`

After calling this notification function, Termite will physically unload the filter DLL. For filters that allocate dynamic memory or other resources, this is a good moment to free these resources.

LPCSTR flt_Receive(LPCSTR Text, LPINT Size)

flt_Receive() is called after Termite has received new data. It is also called after not receiving data for some time-out (which is hard-coded to 0.5 second). If a plug-in filter buffers data internally, this time-out allows the filter to parse the remaining data.

Text	The contents of the received data.
Size	On input, this parameter points to the size of the data block that parameter "Text" holds. Note that the data in "Text" need to be zero-terminated. On output, this parameter must hold the new size of the data block. This parameter may be zero on input, so that the plug-in can pass any data that it had buffered internally to Termite.
Return	If flt_Receive() does not change the string, it can return NULL or return the original string. Otherwise, the function should return a pointer to a modified buffer (and store the size of that buffer in parameter "Size").
Notes	If the result of flt_Receive() is a shorter string than the input string, the function may change the string in place (but this is not encouraged). If the string changes (and especially if the result is bigger than the input string), the function should allocate memory for the modified string. The function should also keep track of the allocated memory, because it must free the memory itself: either on a next call, or on flt_Unload(). It is suggested that the filter creates an auto-growing output buffer. If the filter wishes to remove all data, it must set parameter "Size" to zero on output. It should return a pointer to the input buffer (parameter "Text"); specifically, it should not return NULL. The input and output strings are not necessarily zero-terminated; the filter must adjust the length to the number of bytes it returns (parameter "Size").

LPCSTR flt_Process(LPCSTR Text, LPINT Size)

This function is an alias for flt_Receive(). The first version of Termite that offered plug-in filter support only provided filtering of received data —not of transmitted data. In the current version of Termite, the alias of flt_Process() is kept for backward compatibility with old filters. However, new filters should use flt_Receive() instead.

LPCSTR flt_Transmit(LPCSTR Text, LPINT Size)

flt_Transmit() is called before Termite transmits data that the user has typed in.

Text	The contents of the data to be transmitted.
Size	On input, this parameter points to the size of the data block that parameter "Text" holds. Note that the data in "Text" need to be zero-terminated. On output, this parameter must hold the new size of the data block.
Return	If flt_Transmit() does not change the string, it can return NULL or return the original string. Otherwise, the function should return a pointer to a modified buffer (and store the size of that buffer in parameter "Size").
Notes	If the result of flt_Transmit() is a shorter string than the input string, the function may change the string in place (but this is not encouraged). If the string changes (and especially if the result is bigger than the input string), the function should allocate memory for the modified string. The function should also keep track of the allocated memory, because it must free the memory itself: either on a next call, or on flt_Unload(). It is suggested that the filter creates an auto-growing output buffer. If the filter wishes to remove all data, it must set parameter "Size" to zero on output. It should return a pointer to the input buffer (parameter "Text"); specifically, it should not return NULL.

The input and output strings are not necessarily zero-terminated; the filter must adjust the length to the number of bytes it returns (parameter "Size").

LPCSTR flt_HotKey(int vKey, LPCSTR Text, LPINT Size)

flt_HotKey() is called when the user types a function key in Termite.

vKey	The virtual key code of the function key (e.g. VK_F1).
Text	On entry, this is typically an empty string, but if another filter has also handled the same function key, there may be text stored in this parameter.
Size	On input, this parameter points to the size of the data block that parameter "Text" holds. It is typically zero. Note that the data in "Text" need to be zero-terminated. On output, this parameter must hold the new size of the data block.
Return	If flt_HotKey() does not handle the function key, it can return NULL or return the original string. Otherwise, the function should return a pointer to a buffer with the replacement text for the function key (and store the size of that buffer in parameter "Size").
Notes	It is up to the filter to decide how to respond to double definitions of the same function key. If a filter detects that another filter has already handled the key (by inspecting parameter "Text"), it may either give a warning, or append its own definition to the text already present.

BOOL flt_Config(void)

flt_Config() is called when the user chooses to configure the filter.

Return	This function must return TRUE if the configuration was successfully changed, and FALSE if the user cancelled the operation or if there was an error.

Messages

UM_PLUGINWINDOW

Defined as (WM_APP + 1)
To create a window in Termites interface, the plug-in filter must send the message UM_PLUGINWINDOW to the Termite main window from its flt_Load() function. See function flt_Load() for details. Note that sending this message when the filter is not in its "flt_Load()" state, does nothing.

UM_COMMHANDLE

Defined as (WM_APP + 2)
This message returns the handle of the serial port that Termite has opened. If no serial port is open, the result of this message is the value INVALID_HANDLE_VALUE.