DLL Support

The /dll, $dll and $dllcall features allow you to make calls to DLL's designed to work with AdiIRC. The main reason you would want to do this is that processing information in a DLL can be far faster than doing so in a script, so for intensive data processing a DLL would be more efficient.

If you want to create a DLL for use with AdiIRC you need to create a specific routine like this in your DLL file.

For technical reasons, the DLL must be compiled for 64 bit on 64 bit versions of AdiIRC.

int __stdcall procname(HWND mWnd, HWND aWnd, char *data, char *parms, BOOL show, BOOL nopause)

"procname" is the name of the routine you will call from AdiIRC, you can create as many as you need and they can be named anything.

You may need to create a .def file with the "procname" names exported when compiling your DLL.

Parameters

mWnd - The handle to the main AdiIRC window.
aWnd - The handle of the window in which the command is being issued, this might not be the currently active window if the command is being called by a remote script.
data - The information that you wish to send to the DLL. On return, the DLL can fill this variable with the command it wants AdiIRC to perform if any.
parms - Can be filled by the DLL on return with parameters that it wants AdiIRC to use when performing the command that it returns in the data variable.
show - Is FALSE if the . prefix was specified to make the command quiet, or TRUE otherwise.
nopause - Is TRUE if AdiIRC is in a critical routine and the DLL must not do anything that pauses processing in AdiIRC, eg. the DLL should not pop up a dialog.

Returns

The routine can return an integer to indicate what it wants AdiIRC to do:

0 - Means that AdiIRC should /halt processing.
1 - Means that AdiIRC should continue processing.
2 - Means that it has filled the data variable with a command which it wants AdiIRC to perform, and has filled parms with the parameters to use, if any, when performing the command.
3 - Means that the DLL has filled the data variable with the result that $dll as an identifier should return.

Keeping a DLL Loaded after a call

By default a DLL is unloaded immediately after you make the /dll, $dll or $dllcall call. You can keep a DLL loaded by including a LoadDll() routine in your DLL, which AdiIRC calls the first time you load the DLL:

void __stdcall LoadDll(LOADINFO*);

```
typedef struct {
DWORD  mVersion;
HWND   mHwnd;
BOOL   mKeep;
BOOL   mUnicode;
DWORD  mBeta;
DWORD  dBytes;
} LOADINFO;
```

Struct parameters

mVersion - Contains the AdiIRC version number in the low and high words.
mHwnd - Contains the window handle to the main AdiIRC window.
mKeep - Set to TRUE by default, indicating that AdiIRC will keep the DLL loaded after the call. You can set mKeep to FALSE to

make AdiiRC unload the DLL after the call.
mUnicode - Indicates that text is in unicode as opposed to ansi.
mBeta - Contains the AdiIRC beta version number, for public beta releases.
dBytes - Maximum byte size for the mData parameter.

Unloading a DLL

You can unload a loaded DLL by using the -u switch:

```
/dll -u <filename>
```

AdiIRC will automatically unload a DLL if it is not used for ten minutes, or when AdiIRC exits.

You can also define an UnloadDll() routine in your DLL which AdiIRC will call when unloading a DLL to allow it to clean up.

int __stdcall UnloadDll(int mTimeout);

mTimeout can be:

0 - UnloadDll() is being called due to a DLL being unloaded with /dll -u
1 - UnloadDll() is being called due to a DLL not being used for ten minutes. The UnloadDll() routine can return 0 to keep the DLL loaded, or 1 to allow it to be unloaded.
2 - UnloadDll() is being called due to a DLL being unloaded when AdiIRC exits.