Apache Module mod_so

Description:	Loading of executable code and modules into the server at start-up or restart time
Status:	Extension
Module Identifier:	so_module
Source File:	mod_so.c
Compatibility:	This is a Base module (always included) on Windows

Summary

On selected operating systems this module can be used to load modules into Apache at runtime via the Dynamic Shared Object[1] (DSO) mechanism, rather than requiring a recompilation.

On Unix, the loaded code typically comes from shared object files (usually with `.so` extension), on Windows this may either the `.so` or `.dll` extension.

Warning

Apache 1.3 modules cannot be directly used with Apache 2.0 - the module must be modified to dynamically load or compile into Apache 2.0.

Topics

Directives

Creating Loadable Modules for Windows

Note

The module name format changed for Windows with Apache 1.3.15 and 2.0 - the modules are now named as mod_foo.so

While mod_so still loads modules with ApacheModuleFoo.dll names, the new naming convention is preferred; if you are converting your loadable module for 2.0, please fix the name to this 2.0 convention.

The Apache module API is unchanged between the Unix and Windows versions. Many modules will run on Windows with no or little change from Unix, although others rely on aspects of the Unix architecture which are not present in Windows, and will not work.

When a module does work, it can be added to the server in one of two ways. As with Unix, it can be compiled into the server. Because Apache for Windows does not have the `Configure` program of Apache for Unix, the module's source file must be added to the ApacheCore project file, and its symbols must be added to the `os\win32\modules.c` file.

The second way is to compile the module as a DLL, a shared library that can be loaded into the server at runtime, using the `LoadModule` directive. These module DLLs can be distributed and run on any Apache for Windows installation, without recompilation of the server.

To create a module DLL, a small change is necessary to the module's source file: The module record must be exported from the DLL (which will be created later; see below). To do this, add the `AP_MODULE_DECLARE_DATA` (defined in the Apache header files) to your module's module record definition. For example, if your module has:

```
module foo_module;
```

Replace the above with:

```
module AP_MODULE_DECLARE_DATA foo_module;
```

Note that this will only be activated on Windows, so the module can continue to be used, unchanged, with Unix if needed. Also, if you are familiar with `.DEF` files, you can export the module record with that method instead.

Now, create a DLL containing your module. You will need to link this against the libhttpd.lib export library that is created when the libhttpd.dll shared library is compiled. You may also have to change the compiler settings to ensure that the Apache header files are correctly located. You can find this library in your server root's modules directory. It is best to grab an existing module .dsp file from the tree to assure the build environment is configured correctly, or alternately compare the compiler and link options to your .dsp.

This should create a DLL version of your module. Now simply place it in the `modules` directory of your server root, and use the `LoadModule` directive to load it.

LoadFile Directive

Description:	Link in the named object file or library
Syntax:	`LoadFile` *`filename`* `[`*`filename`*`] ...`
Context:	server config
Status:	Extension
Module:	mod_so

The LoadFile directive links in the named object files or libraries when the server is started or restarted; this is used to load additional code which may be required for some module to work. *Filename* is either an absolute path or relative to ServerRoot[2].

For example:

```
LoadFile libexec/libxmlparse.so
```

LoadModule Directive

Description:	Links in the object file or library, and adds to the list of active modules
Syntax:	`LoadModule` *`module filename`*
Context:	server config
Status:	Extension
Module:	mod_so

The LoadModule directive links in the object file or library *filename* and adds the module structure named *module* to the list of active modules. *Module* is the name of the external variable of type `module` in the file, and is listed as the Module Identifier[3] in the module documentation. Example:

```
LoadModule status_module modules/mod_status.so
```

loads the named module from the modules subdirectory of the ServerRoot.

URI References

[1] http://httpd.apache.org/docs-2.1/dso.html
[2] http://httpd.apache.org/docs-2.1/mod/core.html#serverroot
[3] http://httpd.apache.org/docs-2.1/mod/module-dict.html#ModuleIdentifier