IIS and ISAPI REWRITE

By Timothy Penner, Technical Services Team Member, 4D Inc.

Technical Note 09-27

Table of Contents

Abstract

This Technical Note discusses how to use Microsoft IIS as a frontend to 4D's Web Server. This is achieved through the use of the 3rd party ISAPI filter from Helicon Tech, ISAPI_Rewrite. Various sample configurations are given and their options are discussed. This information should give the 4D Developer the knowledge necessary to provide an IIS frontend to their 4D Web Server if their clients request it.

Introduction

Through the use of the 3rd party ISAPI filter from Helicon Tech, ISAPI_Rewrite; it is very simple to use Microsoft IIS as a frontend to 4D's Web Server. This Tech Note explores various sample configurations for the ISAPI_Rewrite filter and their options are discussed as well. This information should give the 4D Developer the knowledge necessary to provide an IIS frontend to their 4D Web Server if their clients request it.

What is it?

IIS stands for "Internet Information Services" which is Microsoft's Web Server that is built in to most modern Microsoft Operating Systems. IIS uses an API known as ISAPI. ISAPI stands for "Internet Server Application Programming Interface" and is an API of Microsoft's IIS.

ISAPI_Rewrite is an ISAPI filter created by Helicon Tech that can be used in order to catch specific requests on an IIS server, send those requests to another server (such as 4D's Web Server), and then proxy the response back to the requestor. Helicon Tech describes the ISAPI_Rewrite filter as:

> ISAPI_Rewrite is a powerful URL manipulation engine based on regular expressions. It acts mostly like Apache's mod_Rewrite, but is designed specifically for Microsoft's Internet Information Server (IIS). ISAPI_Rewrite is an ISAPI filter written in pure C/C++ so it is extremely fast. ISAPI_Rewrite gives you the freedom to go beyond the standard URL schemes and develop your own scheme.
>
> Source: http://www.isapirewrite.com/ (product overview)

The ISAPI_Rewrite filter can be obtained from Helicon Tech from the following address: http://www.helicontech.com/download-ISAPI_Rewrite3.htm

Why use it?

Microsoft IIS is an industry standard Web Server that many organizations use (mainly because it is built into the Microsoft server Operating Systems). A lot of enterprise IT departments are comfortable with IIS and prefer to deploy on what

they are comfortable with. This can sometimes make it difficult to sell the idea of using 4D's built in web server for large corporate projects where the IT folks only want to deploy on what they know.

Luckily, with the help of a 3rd party ISAPI filter, you can use IIS as a frontend to 4D's Web Server thus allowing the IT folks to continue using the product they know and love, yet also serving up the dynamic (or static) content from 4D' Web Server.

With the ISAPI_Rewrite filter you can:

- Perform Search Engine Optimizations to dynamic content
- Block hotlinking of your data files by other sites
- Develop a custom authorization scheme and manage access to the static files using custom script and database
- Proxy content of one site into a directory on another site
- Make your intranet server
- Create dynamic host-header based sites using a single physical site
- Return browser-dependant content even for static pages
- And many other problems could be solved with the power of the regular expression engine built into the ISAPI_Rewrite filter.

This Tech Note focuses primarily on using IIS as a Frontend (reverse proxy) to 4D's Web Server.

How does it work?

The ISAPI_Rewrite filter is an extension to IIS not 4D; therefore it must be installed on the same machine as the IIS Web Server (Download available here: http://www.helicontech.com/download-ISAPI_Rewrite3.htm). The filter does not need to be installed on the same machine as the 4D Web Server ***unless*** the IIS Web Server and the 4D Web Server are both on the same machine. Obviously, if the 4D Web Server and IIS Web Server are not on the same machine, they must be able to properly communicate with one another via the ports specified in the configuration.

Note: There is no additional configuration needed on the 4D side; simply install the ISAPI Filter on the IIS machine and configure it to point to 4D's Web Server.

Typically the IIS Web Server will be running on the standard port (80) and 4D's Web Server will be running on a non-standard port (such as 8080). The ISAPI_Rewrite filter is then configured to catch certain requests and proxy those matching requests over to the 4D Web Server.

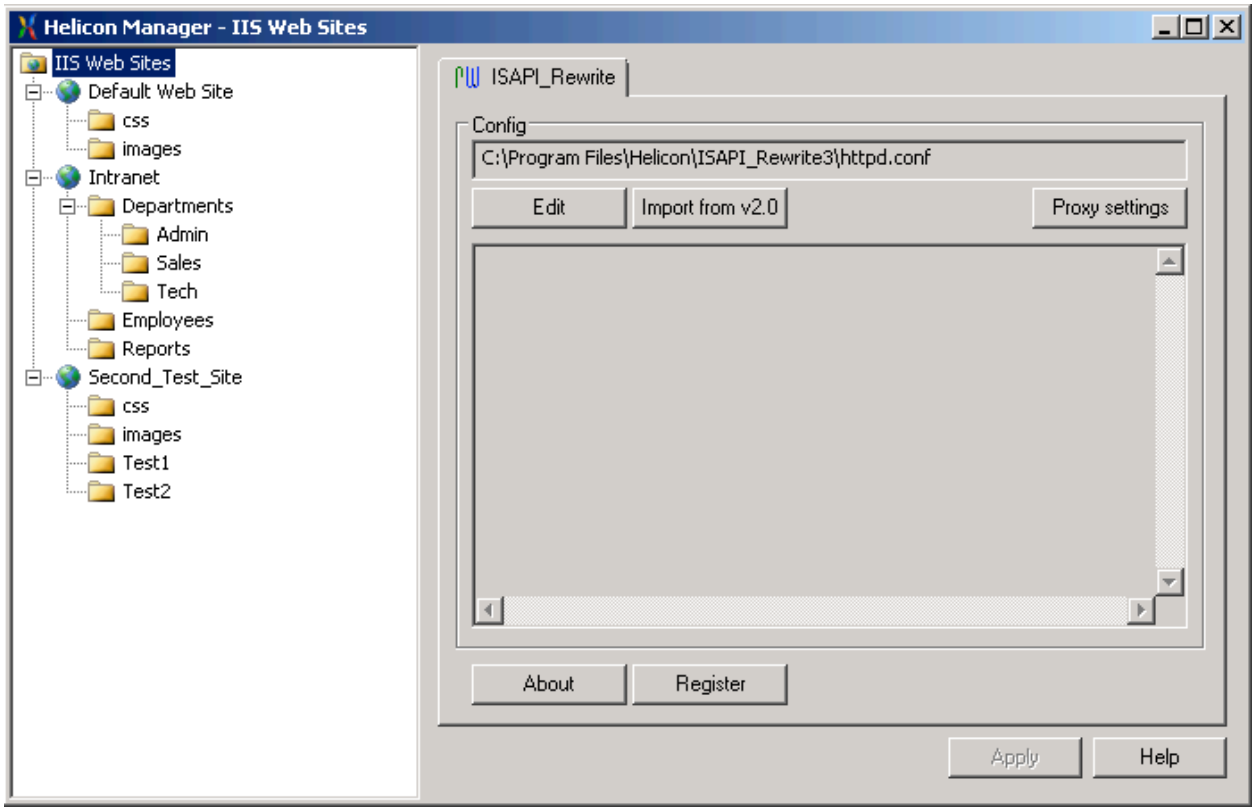
The ISAPI_Rewrite filter uses a regular expression engine to bring capabilities similar to mody_proxy of Apache over into the IIS world. The configuration is broken down into a two different categories:

- Global configuration that applies to all sites; this is stored in the applications installation directory at (by default) C:\Program Files\Helicon\ISAPI_Rewrite3\httpd.conf
- Individual configurations for each individual site and the directories found within. Those configuration files are stored as .htaccess files and reside in the directory they apply to.

The configuration is exposed through the use of the Helicon Manager (also referred to as the ISAPI_Rewrite manager).

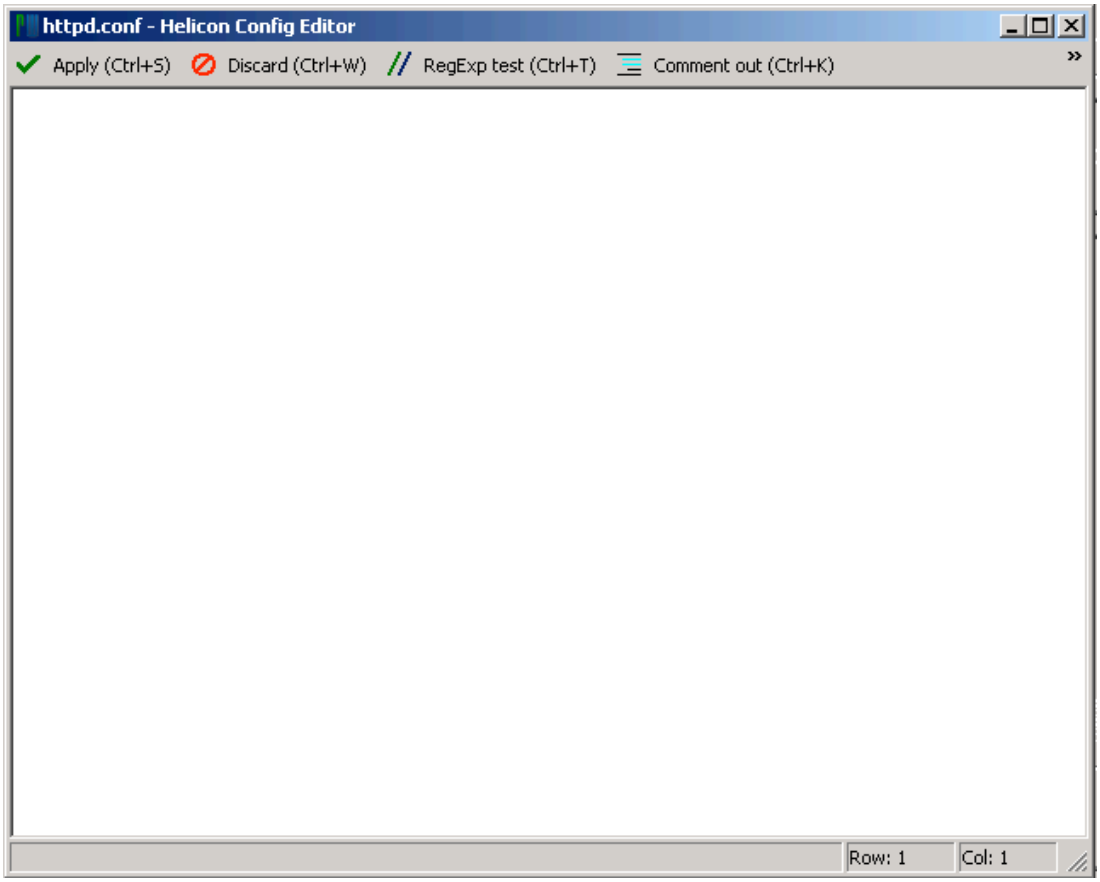
Using the Helicon ISAPI REWRITE Manager application

The ISAPI_REWRITE filter comes with a manager application that can be used for implementing the configurations. The following image shows the Helicon Manager when it is first launched:



This Tech Note does the entire configuration in the global configuration file. To modify the configuration file, first you must select the level at which you want to modify (i.e. select the "IIS Web Sites" icon if you want to modify the global configuration) and then click on the 

button. This brings up the editor window:

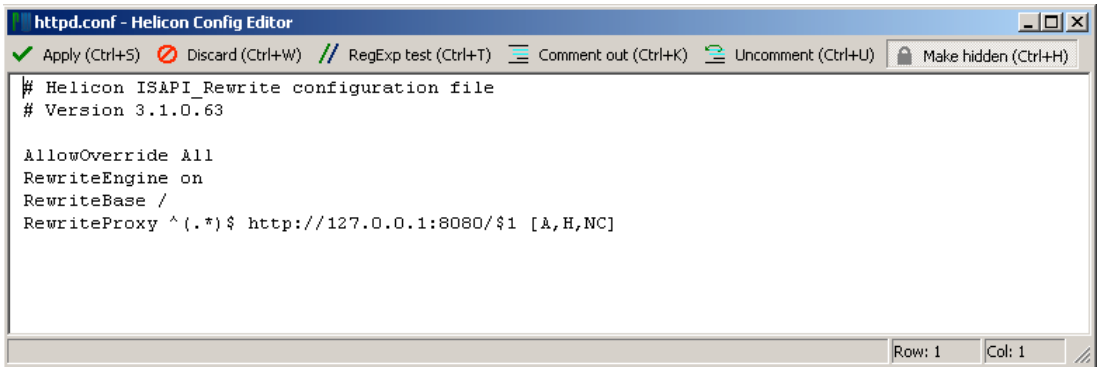


The Helicon Config Editor allows you to enter or modify the specific configuration used. Let's start by entering the following sample configuration:

```
# Helicon ISAPI_Rewrite configuration file
# Version 3.1.0.63

AllowOverride All
RewriteEngine on
RewriteBase /
RewriteProxy ^(.*)$ http://127.0.0.1:8080/$1 [A,H,NC]
```

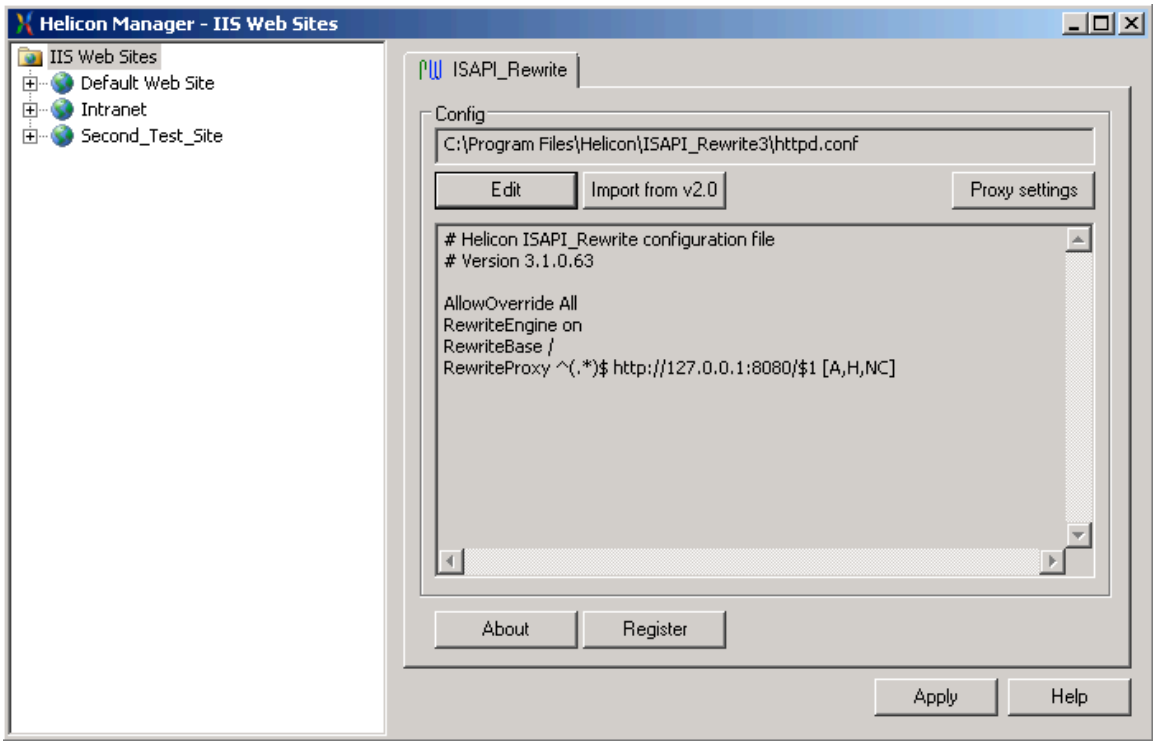
Once typed it should look like the following:

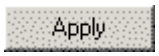


After modifying the contents of this window, simply press the 

button from the toolbar.

After clicking Apply, the changes to a configuration file are saved and you are returned to the Helicon Manager;



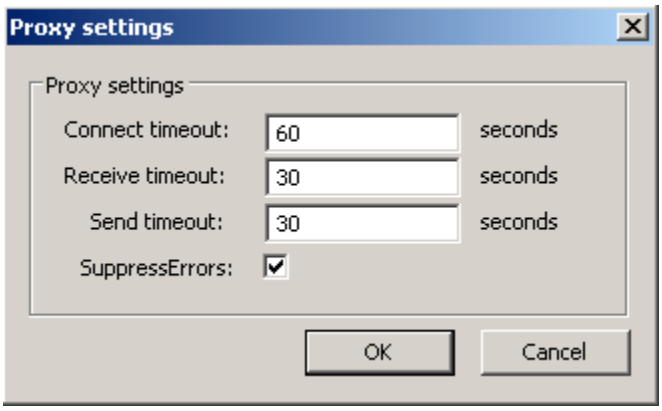
However, the changes do not become active until you click on the 

button from within the Helicon Manager.

Changing the proxy settings

The Helicon Manager also allows the IIS administrator to alter the proxy timeout settings which are available through the 

button. The configuration options include: Connect timeout (default 60 seconds), Receive timeout (default 30 seconds), Send timeout (default 30 seconds), and an option to Suppress Errors (default enabled).



Note: Since stepping through code in the debugger may take longer than the default timeout specified; it is suggested that you increase these timeouts while debugging your application to avoid issues related to timeouts.

Directives used in this Tech Note

This section discusses the directives that are used in this Tech Note. For a complete list of available directives and their uses please check HeliconTech's documentation page for the ISAPI_Rewrite filter available here: http://www.helicontech.com/isapi_rewrite/doc/

LogLevel

```
Directive: LogLevel
Description: Sets the level of logging for general errors
Syntax: LogLevel Level
Default: LogLevel warn
Context: server config
```

The LogLevel directive sets the verbosity of general logging output, not related to rewriting process itself.

Here is a list of available values for Level parameter of the LogLevel directive:

- emerg
- alert
- crit
- error
- warn
- notice
- info
- debug

Currently ISAPI_Rewrite logs only debug and crit error messages.

Set LogLevel debug to troubleshoot configuration file loading problems.

For full documentation on the **LogLevel** directive please visit:
http://www.helicontech.com/isapi_rewrite/doc/LogLevel.htm

RewriteLogLevel

```
Directive: RewriteLogLevel
Description: Sets the level of logging
Syntax: RewriteLogLevel Level
Default: RewriteLogLevel 0
Context: server config
```

The RewriteLogLevel directive sets the verbosity of logging output. The default value of 0 means no logging while maximum level of 9 means all actions will be logged.

Using higher values of logging may slow down ISAPI_Rewrite operation. It is recommended that you disable logging by setting log level to 0 after the debugging of your ruleset has been completed.

For full documentation on the **RewriteLogLevel** directive please visit:
http://www.helicontech.com/isapi_rewrite/doc/RewriteLogLevel.htm

RewriteLog

```
Directive: RewriteLog
Description: Sets name of ISAPI_Rewrite log file
Syntax: RewriteLog file-path
Default: RewriteLog installdir\rewrite.log
Context: server config
```

The RewriteLog directive sets the name and path of the logfile where ISAPI_Rewrite logs all of its actions. For example:

RewriteLog "C:\local\path\rewrite.log"

For full documentation on the **RewriteLog** directive please visit:
http://www.helicontech.com/isapi_rewrite/doc/RewriteLog.htm

ErrorLog

```
Directive: ErrorLog
Description: Location of general errors file
Syntax: ErrorLog file-path
Default: ErrorLog installdir\rewrite.log
Context: server config
```

The ErrorLog directive sets the name and path of the logfile where ISAPI_Rewrite logs general errors and messages such as httpd.conf file load, .htaccess file load, etc. For example:

ErrorLog "C:\local\path\error.log"

For full documentation on the **ErrorLog** directive please visit:
http://www.helicontech.com/isapi_rewrite/doc/ErrorLog.htm

AllowOverride

```
Directive: AllowOverride
Description: Explicitly sets the base URL for per-directory rewrites.
Syntax: AllowOverride All|None|directive-type [directive-type] ...
Default: AllowOverride All
Context: server config, virtual host, directory
```

The AllowOverride directive declares which directives in distributed *.htaccess* files can override directives from the parent levels (*httpd.conf*). In context of ISAPI_Rewrite, this directive in fact enables or disables *.htaccess* files for a specific virtual host (web site) or directory. Only three values are currently supported: **All**, **None** and **FileInfo.**

All and **FileInfo** enables .htaccess file and all ISAPI_Rewrite directives in it. **None** disables all .htaccess files and directives. This directive is inheritable. This means if you specify *AllowOverride none* for some directory or virtual host, .htaccess files are also disabled for all subdirectories.

For full documentation on the **AllowOverride** directive please visit:
http://www.helicontech.com/isapi_rewrite/doc/AllowOverride.htm

RewriteEngine

```
Directive: RewriteEngine
Description: Enables or disables runtime rewriting engine
Syntax: RewriteEngine on|off
Default: RewriteEngine off
Context: server config, virtual host, directory, .htaccess
```

The RewriteEngine directive enables or disables rewriting runtime. Use RewriteEngine off instead of commenting out rewrite rules if you need to disable ISAPI_Rewrite module or specific .htaccess file.

WARNING! *Due to the number of support queries, HeliconTech had to enable rewriting engine by default whilst in Apache rewriting engine by default is off. Please keep in mind this small incompatibility and always specify explicit status of rewriting engine in each configuration file, whether you need it on or off.*

For full documentation on the **RewriteEngine** directive please visit:
http://www.helicontech.com/isapi_rewrite/doc/RewriteEngine.htm

RewriteBase

```
Directive: RewriteBase
Description: Explicitly sets the base URL for per-directory rewrites.
Syntax: RewriteBase URL-path
Default: RewriteBase requested-directory-path
```

Context: directory, .htaccess

When the *RewriteRule* directive is used in per-directory configuration files (*.htaccess*), it automatically strips the local directory prefix from the path and applies rules only to the remainder. **RewriteBase** directive allows you to explicitly specify a base for the rules, i.e. the part that is to be stripped.

Unlike Apache mod_rewrite, ISAPI_Rewrite has access not only to the physical path, but also to the virtual path and can automatically choose the correct base. So this directive is preserved only for compatibility reasons.

URL-path can be a root relative path or empty. An empty *URL-path* value means that rule base is equal to the web site root.

For full documentation on the **RewriteBase** directive please visit:
http://www.helicontech.com/isapi_rewrite/doc/RewriteBase.htm

RewriteProxy

Directive: RewriteProxy
Description: Proxy request to a distant server
Syntax: RewriteProxy Pattern Substitution [flags]
Context: server config, virtual host, directory, .htaccess

The RewriteProxy directive causes the resulting URL to be internally treated as a target on another server and immediately (i.e. rules processing stops here) passed to the remote server. Response of the remote server is then passed back to the client. Proxy requires you to specify a fully qualified URL, starting from protocol, host name, etc. ISAPI_Rewrite uses ISAPI extension to handle proxy requests.

Syntax and operation is the same as for RewriteRule directive, but RewriteProxy directive supports some additional flags:

H (preserve **H**ost**)**

The H flag means that the proxy module is to use the Host header sent in the original request when connecting to a remote server. Without this flag, the proxy module composes the Host header from the host name and a port of the remote server.

A **(A**dd authentication headers**)**

The A flag allows passing of authentication information from proxy to an internal server when client authentication against a proxy server is used.

Proxy module will append headers

```
X-ISRW-Proxy-AUTH-TYPE,
X-ISRW-Proxy-AUTH-USER,
```

```
X-ISRW-Proxy-LOGON-USER,
X-ISRW-Proxy-REMOTE-USER
```

Corresponding to server variables

```
AUTH_TYPE,
AUTH_USER,
LOGON_USER,
REMOTE_USER
```

To a request sent to a proxied server.

CR (use **Cr**edentials**)**

The CR flag means the proxy module tries to login on a remote server with the credentials specified in the URL or basic authentication headers. With this flag you can use http://user:password@host.com/path/ syntax as a URL within substitution string.

For full documentation on the **RewriteProxy** directive please visit: http://www.helicontech.com/isapi_rewrite/doc/RewriteProxy.htm

For full documentation on the **RewriteRule** directive please visit: http://www.helicontech.com/isapi_rewrite/doc/RewriteRule.htm

Examples

Although the Helicon ISAPI_REWRITE filter allows the developer complete customization by giving them the ability to write their own configuration files; this section covers the various configurations that 4D developers may be most interested in and may use most frequently.

It is suggested that you use the logging mechanisms built into Helicon ISAPI_Rewrite filter, at the very least for debugging while testing this approach in your own environment. Here is the code snippet of the configuration that activates the log.

```
# Helicon ISAPI_Rewrite configuration file
# Version 3.1.0.63

LogLevel debug
RewriteLogLevel 9

RewriteLog C:\Program Files\Helicon\ISAPI_Rewrite3\rewrite.log
ErrorLog C:\Program Files\Helicon\ISAPI_Rewrite3\error.log
```

The above snippet activates the debug log and the rewrite log and specifies a location for both to be saved at. For brevity, this information is only included in the first example but can be used in all of them.

Proxy only a specific directory to 4D

This section covers how to use the Helicon ISAPI_REWRITE filter for IIS to make a specific directory on IIS proxy its content from 4D.

```
# Helicon ISAPI_Rewrite configuration file
# Version 3.1.0.63

LogLevel debug
RewriteLogLevel 9

RewriteLog C:\Program Files\Helicon\ISAPI_Rewrite3\rewrite.log
ErrorLog C:\Program Files\Helicon\ISAPI_Rewrite3\error.log

AllowOverride All
RewriteEngine on
RewriteBase /
RewriteProxy ^4D/(.*)$ http://127.0.0.1:8080/$1 [A,H,NC]
```

The above configuration file catches all incoming IIS requests that begin with **4D/** and proxies the request to 127.0.0.1:8080; which is the address and port that our 4D Web Server is running on. In this way, 4D gets the web request as if it was being directly requested from 4D itself. In this configuration, you can think of http://mysite.com/4D/ as the root of the 4D Web Server. For example, http://mysite.com/4D/reports/sales.shtml would actually be grabbing http://127.0.0.1:8080/reports/sales.shtml

Note: In this example the /4D/ in the URL is the entry point used by IIS and is completely stripped out of the HTTP request that is sent to 4D.

Proxy only SOAP requests to 4D

This section covers how to use the Helicon ISAPI_REWRITE filter for IIS to proxy SOAP requests to 4D's Web Server.

```
# Helicon ISAPI_Rewrite configuration file
# Version 3.1.0.63

AllowOverride All
RewriteEngine on
RewriteBase /
RewriteProxy ^4DSOAP(.*)$ http://127.0.0.1:8080/4DSOAP/$1 [A,H,NC]
```

The above configuration file catches all incoming IIS requests that begin with 4DSOAP and proxies the request to http://127.0.0.1:8080/4DSOAP/; which is the address and port that our 4D Web Server is running on. In this way, 4D gets the soap request as if it was being directly requested from 4D itself. In this configuration, the only requests that are redirected to 4D's Web Server are the soap requests beginning with 4DSOAP.

Proxy SOAP requests and a specific directory to 4D

This section covers how to use the Helicon ISAPI_REWRITE filter for IIS to proxy SOAP requests to 4D and make a specific directory on IIS proxy its content from 4D.

```
# Helicon ISAPI_Rewrite configuration file
# Version 3.1.0.63

AllowOverride All
RewriteEngine on
RewriteBase /
RewriteProxy ^4D/(.*)$ http://127.0.0.1:8080/$1 [A,H,NC]
RewriteProxy ^4DSOAP(.*)$ http://127.0.0.1:8080/4DSOAP/$1 [A,H,NC]
```

The above configuration file catches all incoming IIS requests that begin with either 4D/ or 4DSOAP and proxies the request to the 4D Web Server running on 127.0.0.1:8080 –

- If the request begins with 4D/ it is proxied to the root of the 4D web server and the "4D/" is stripped from the request received from 4D.
- If the request begins with 4DSOAP it is proxied to the 4D web server as if the request was made directly to the 4D web server.

In this way, SOAP requests beginning with 4DSOAP as well as requests beginning with 4D/ are both proxied to 4D's Web Server.

Proxy all requests to 4D

This section covers how to use the Helicon ISAPI_REWRITE filter for IIS to proxy all of its content from 4D.

```
# Helicon ISAPI_Rewrite configuration file
# Version 3.1.0.63

AllowOverride All
RewriteEngine on
RewriteBase /
RewriteProxy ^(.*)$ http://127.0.0.1:8080/$1 [A,H,NC]
```

The above configuration file catches incoming IIS requests and proxies the request to 127.0.0.1:8080; which is the address and port that our 4D Web Server is running on. In this way, 4D gets all requests from the IIS server. The root of the IIS web server can be thought of as the entry point to 4D's Web Server. For example http://mysite.com/reports/sales.shtml would actually be grabbing http://127.0.0.1:8080/reports/sales.shtml.

Considerations about the WSDL file

Although the content of the 4DWSDL file can be proxied, the rewrite engine does not seem to properly strip out the port number. This can cause problems when

automatically creating web service proxy methods based on a WSDL file. For this reason, it is recommended that you save the 4DWSDL file and manually edit the location attribute of the "soap:address" element for each web service your database offers. This can be accomplished by loading the 4DWSDL file in a browser and then choosing File -> Save As from the Browser's menu. This will save the file in XML format, you can then place this file on your website and give this new address to the people who will be connecting to your web services; alternatively you could also send the modified file to them via email.

Conclusion

This Technical Note described the general concepts of proxying content from 4D into IIS using the Helicon ISAPI_Rewrite filter. The full version of the filter was used in this Tech Note and various configurations were discussed. This information should give the 4D Developer the knowledge necessary to provide an IIS frontend to their 4D Web Server if their clients request it.

More information on Helicon products including licensing information for the ISAPI_Rewrite 3.x filter can be found here:
http://www.helicontech.com/ISAPI_Rewrite/

ISAPI_Rewrite 3.x Download:
http://www.helicontech.com/download-ISAPI_Rewrite3.htm

Support Forums for ISAPI_Rewrite 3.x:
http://www.helicontech.com/forum/forum_topics-FID-10.htm

Documentation for ISAPI_Rewrite 3.x:
http://www.helicontech.com/isapi_rewrite/doc/