URL Rewrite Rules

URL Rewriting is a feature of most mainstream web server products, including IIS, Apache, and NGINX. The act of "rewriting" the URL allows a client to submit one URL but the web server converts it into another before processing it, passing it along to another server, or passing it to a script engine (like OECGI4.exe).

While there is no *technical* requirement to invoke URL Rewriting, it is highly recommended in order to provide URLs that are technology neutral and semantically cleaner. What does this mean? First, consider that all CGI script engines require that the URL include the name of the engine in order to be properly executed. For instance:

```
http://www.mysite.com/cgi-bin/oecgi4.exe/archives/articles
```

The presence of "cgi-bin/oecgi4.exe" leaves a technology specific signature in the URL. There could be a number of reasons why this is undesirable. For example, there may come a time to replace OECGI4.exe with something else, such as OECGI5.exe or another script engine altogether. Perhaps you would prefer not to reveal what kind of script engine you are using in order to discourage certain modes of hacking. In this case, this would be a preferable URL:

```
http://www.mysite.com/archives/articles
```

By removing any references to the technology, the URL is cleaner and easier to publish. It also allows the URL semantic to represent a resource rather than broadcast that some script is running on the server. Finally, if the technology ever needed to be swapped out on the server, no clients would ever need to be rewritten. The URL would remain the same as long as URL Rewriting continued to be used to maintain a consistent URL.

We wrote an article on our blog site called Hiding OECGIx.exe Using URL Rewriting that explains how to create Rewrite Rules for IIS and Apache. Most URL Rewrite modules use regular expressions, so once you have grasped the technique for one web server you should be able to apply it onto another. Advanced rewrite rules that do more than simply hide OECGI and are well suited for use with the SRP HTTP Framework can be found below.

Rewrite Rules for SRP HTTP Framework on Various Web Servers

The following is a list of rewrite rules specifically for the SRP HTTP Framework under various web servers.

In all examples below *api* is a user defined path and can be substitued for whatever path you choose for your API. Also, OECGI4.exe could be replaced with OECGI3.exe or whatever version of OECGI that is in place.

Abyss

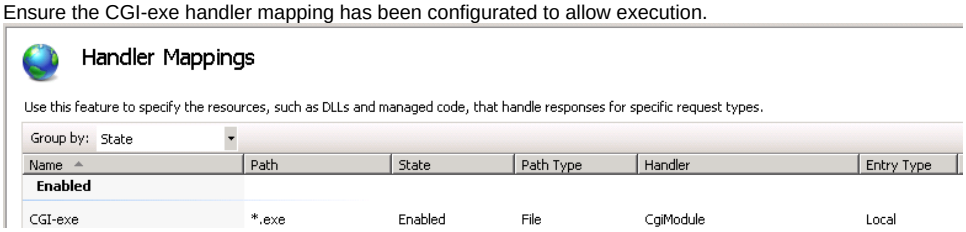
Setting	Value
Type	Global
Base virtual path	
Virtual path	^/api$
If this rule matches	Perform an internal redirection
Destination	/api/cgi-bin/oecgi4.exe
Next action	Stop matching

Setting	Value
Type	Global
Base virtual path	
Virtual path	^/api(\/)(.*)$
If this rule matches	Perform an internal redirection
Destination	/api/cgi-bin/oecgi4.exe/$2
Next action	Stop matching

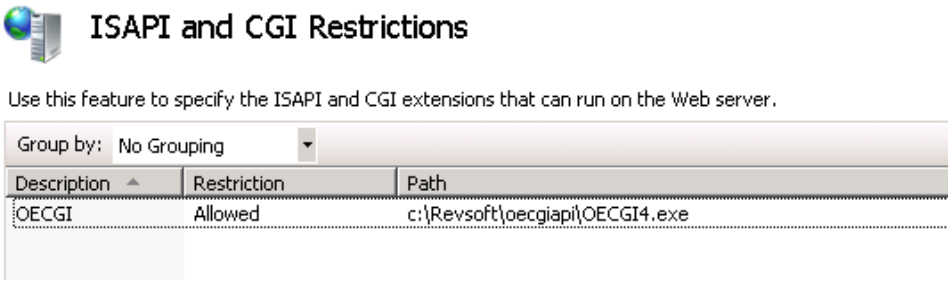
IIS

The instructions for IIS cover two possible configuration scenarios, setting up a site dedicated for your API endpoint or adding an API endpoint to an exsiting site as a virtual directory. Please follow the section which applies to your configuration. Regardless of your configuration you must make security changes in IIS to enable the execution of EXE CGI scripts:

1. Ensure the CGI-exe handler mapping has been configurated to allow execution.



2. Add an ISAPI and CGI Restriction to enable OECGI4.exe to execute from the location where it resides. The actual path depends upon your site configuration and will be referenced in the instructions below.



Dedicated API site Domain or Subdomain

When setting up a website dedicated to host the API endpoint (i.e. api.example.com) follow these steps:

1. Place OECGI4.exe in the website root directory so it is accessible using the path api.example.com/oecgi4.exe
2. Update your *ISAPI and CGI Restrictions* settings to ensure OECGI4.exe is allowed to execute from the website root directory.
3. Create the following URL Rewrite rules in the root directory:

Setting	Value
Match URL	^api$
Action type	Rewrite
Action URL	oecgi4.exe
Stop processing after match	Checked

Setting	Value
Match URL	^api([_0-9a-z-/]+)$
Action type	Rewrite
Action URL	oecgi4.exe/{R:1}
Stop processing after match	Checked

Or you can copy and paste these rules directly into the IIS web.config file:

```
<rule name="Root API" stopProcessing="true">
 <match url="^api$" />
 <conditions logicalGrouping="MatchAll" trackAllCaptures="false" />
 <action type="Rewrite" url="oecgi4.exe" />
</rule>
<rule name="API" stopProcessing="true">
 <match url="^api([_0-9a-z-/]+)$" />
 <conditions logicalGrouping="MatchAll" trackAllCaptures="false" />
 <action type="Rewrite" url="oecgi4.exe/{R:1}" />
</rule>
```

Your API will be accessible by two URLs. The primary URL for your API will be www.example.com/api but you you can access www.example.com/oecgi4.exe/ to bypass the URL rewrite rules which may be useful to do as a troubleshooting step.

Adding an API to an Existing Site

When adding an API endpoint to an existing IIS site using a virtual directory (i.e. www.example.com/api) follow these steps:

1. On the web server copy OECGI4.exe into a separate directory to be used as a CGI directory such as *C:\revsoft\oecgiapi*

2. Update your *ISAPI and CGI Restrictions* settings to ensure OECGI4.exe is allowed to execute from the directory *C:\revsoft\oecgiapi*.
3. In the website root directory create a virtual directory named *oecgiapi* pointing to a local directory *c:\revsoft\oecgiapi*. OECGI should be accessible using the url www.example.com/oecgiapi/oecgi4.exe

   IMPORTANT: The name of the local directory is irrelevant to the rewrite rules but it is very important the virtual directory name is not be the same name as the API endpoint name. If your virtual directory name and API endpoint name are the same IIS will skip the rewrite rules. This guide assumes the API URL will be /api and that this does not exist as a virtual or phyiscal directory because the rewrite rules will rewrite the non-existant /api path to /oecgiapi which does exist.

4. Create the following URL Rewrite rules in the root site directory:

Setting	Value
Match URL	^api$
Action type	Rewrite
Action URL	oecgiapi/oecgi4.exe
Stop processing after match	Checked

Setting	Value
Match URL	^api([_0-9a-z-/]+)$
Action type	Rewrite
Action URL	oecgiapi/oecgi4.exe/{R:1}
Stop processing after match	Checked

Or you can copy and paste these rules directly into the IIS web.config file:

```
<rule name="Root API" stopProcessing="true">
 <match url="^api$" />
 <conditions logicalGrouping="MatchAll" trackAllCaptures="false" />
 <action type="Rewrite" url="oecgiapi/oecgi4.exe" />
</rule>
<rule name="API" stopProcessing="true">
 <match url="^api([_0-9a-z-/]+)$" />
 <conditions logicalGrouping="MatchAll" trackAllCaptures="false" />
 <action type="Rewrite" url="oecgiapi/oecgi4.exe/{R:1}" />
</rule>
```

Your API will be accessible by two URLs. The primary URL for your API will be www.example.com/api but you you can access www.example.com/oecgiapi/oecgi4.exe/ to bypass the URL rewrite rules which may be useful to do as a troubleshooting step.

Apache

```
<IfModule mod_rewrite.c>

  RewriteEngine on

  RewriteBase /

  RewriteRule ^api$ /cgi-bin/OECGI4.exe

  RewriteRule ^api/(.*)$ /cgi-bin/OECGI4.exe/$1

</IfModule>
```