Apache, Mod_Proxy, & 4D

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

Technical Note 09-38

Table of Contents

Table of Contents 2
Abstract 3
Introduction 3
What is it? 3
Why use it? 4
How does it work? 5
  Modifying a configuration file 5
Directives used in this Tech Note 6
  ProxyRequests 6
  ProxyPass 6
  ProxyReverse 7
Examples 8
  Proxy only a specific directory to 4D 9
  Proxy only SOAP requests to 4D 9
  Proxy SOAP requests and a specific directory to 4D 9
  Proxy all requests to 4D 10
Considerations about the WSDL file 10
Conclusion 11

Abstract

This Technical Note discusses how to use Apache HTTP Server as a front end to 4D's Web Server. This is achieved through the use of the mod_proxy module. Various sample configurations are given and their options are discussed. This information should give the 4D Developer the knowledge necessary to provide an Apache front end to their 4D Web Server if their clients request it.

Introduction

Through the use of the mod_proxy module; it is very simple to use Apache as a frontend to 4D's Web Server. This Tech Note explores various sample configurations for the mod_proxy module and their options are discussed as well. This information should give the 4D Developer the knowledge necessary to provide an Apache frontend to their 4D Web Server if their clients request it.

What is it?

Apache is an Open Source HTTP WebServer project of The Apache Software Foundation. Apache has proven itself as a widely-used HTTP WebServer. The license can be found here:

http://www.apache.org/licenses/

Mod_proxy is an Apache Module created by The Apache Software Foundation that can be used in order to catch specific requests on an Apache HTTP Server, send those requests to another server (such as 4D's Web Server), and then proxy the response back to the requestor. The Apache Foundation describes the Mod_proxy module as follows:

> This module implements a proxy/gateway for Apache. It implements proxying capability for AJP13 (Apache JServe Protocol version 1.3), FTP, CONNECT (for SSL), HTTP/0.9, HTTP/1.0, and HTTP/1.1. The module can be configured to connect to other proxy modules for these and other protocols.
>
> Apache's proxy features are divided into several modules in addition to mod_proxy: mod_proxy_http, mod_proxy_ftp, mod_proxy_ajp, mod_proxy_balancer, and mod_proxy_connect. Thus, if you want to use one or more of the particular proxy functions, load mod_proxy and the appropriate module(s) into the server (either statically at compile-time or dynamically via the LoadModule directive).
>
> In addition, extended features are provided by other modules. Caching is provided by mod_cache and related modules. The ability to contact remote servers using the SSL/TLS protocol is provided by the SSLProxy* directives

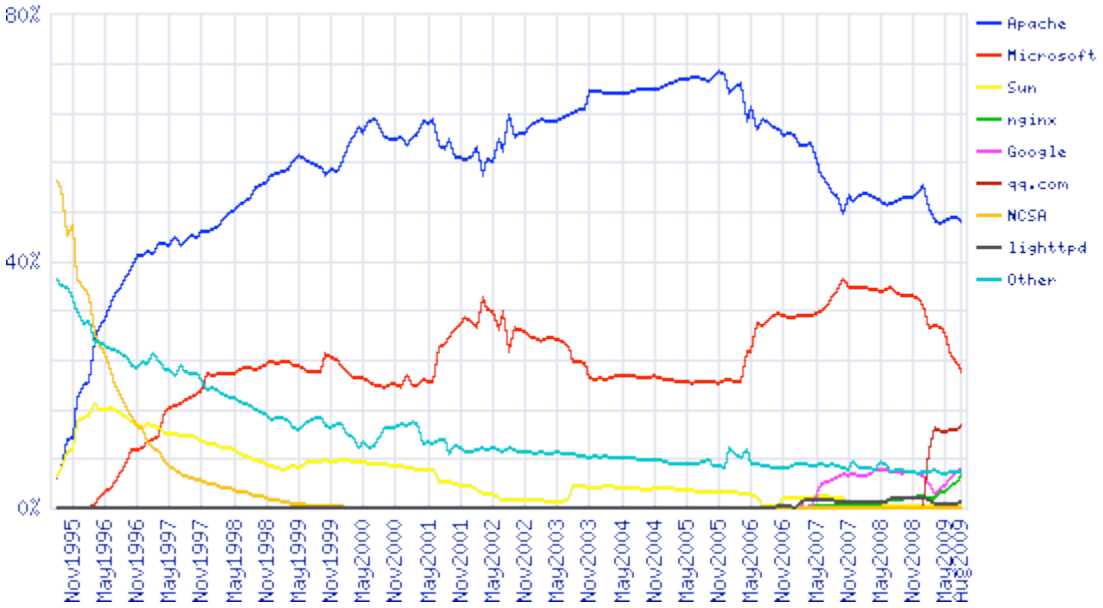
of mod_ssl. These additional modules will need to be loaded and configured to take advantage of these features.

Source: http://httpd.apache.org/docs/2.2/mod/mod_proxy.html

The Mod_proxy module is installed at the time of installing Apache.

Why use it?

Apache is widely used among many organizations. According to the Apache Software Foundation it has been the most popular web server on the Internet since April 1996 (as depicted in the following graph that was current at the time this document was published).



Developer	July 2009	Percent	August 2009	Percent	Change
Apache	113,019,868	47.17%	104,611,555	46.30%	-0.87
Microsoft	55,918,254	23.34%	49,579,507	21.94%	-1.39
qq.com	30,447,369	12.71%	30,278,988	13.40%	0.69
Google	14,226,904	5.94%	14,213,976	6.29%	0.35
nginx	10,174,573	4.25%	11,502,109	5.09%	0.84
lighttpd	1,326,240	0.55%	2,025,521	0.90%	0.34

Source: http://news.netcraft.com/archives/web_server_survey.html

A lot of enterprise IT departments are comfortable with Apache 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 mod_proxy, you can use Apache 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.

Although mod_proxy can be configured in a number of ways, this Tech Note focuses primarily on using Apache as a Front end (reverse proxy) to 4D's Web Server.

How does it work?

Mod_proxy is a module for Apache, not 4D; therefore it must be installed and loaded on the same machine as the Apache Web Server. The module does not need to be installed or loaded on the same machine as the 4D Web Server ***unless*** the Apache Web Server and the 4D Web Server are both on the same machine. Obviously, if the 4D Web Server and Apache 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 load the mod_proxy module into the Apache configuration and then configure it to point to 4D's Web Server.

Typically the Apache Web Server runs on the standard port (80) and 4D's Web Server runs on a non-standard port (such as 8080). The Mod_proxy module is then configured to catch certain requests and proxy those matching requests over to the 4D Web Server.

The configuration is broken down into a two different categories:

- Global configuration that applies to all sites; httpd.conf – depending on the OS used and installation method, this file may or may not be in the following locations, but typically can be found here:
  Mac OS X = /private/etc/apache2/httpd.conf
  Windows = C:\Program Files\Apache\conf\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 files are plain text and can be edited through the use of any standard text editor.

Modifying a configuration file

The Mod_proxy module uses standard Apache configuration files. This means you can place your configuration into the global httpd.conf file or into a individual .htaccess file.

The individual .htaccess files are normally located in the directory that they apply to, where typically the global configuration files are found at the following locations:

Mac OS X = /private/etc/apache2/httpd.conf
Window = C:\Program Files\Apache\conf\httpd.conf

These files can be modified with any standard plain text editor such as notepad++ on Windows and TextWrangler on Mac OS X – but any plain text editor will work.

This Tech Note does the entire configuration in the global configuration file.

Note: After making changes to the configuration files you must restart the Apache service before the changes become affective.

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 the documentation page for the Mod_proxy module available here:
http://httpd.apache.org/docs/2.0/mod/mod_proxy.html

ProxyRequests

Description:	Enables forward (standard) proxy requests
Syntax:	ProxyRequests On|Off
Default:	ProxyRequests Off
Context:	server config, virtual host
Status:	Extension
Module:	mod_proxy

This directive allows or prevents Apache from functioning as a forward proxy server. (Setting *ProxyRequests* to *Off* does not disable the use of the *ProxyPass* directive).

In a typical reverse proxy configuration, this option should be set to *Off*.

In order to get the functionality of proxying HTTP or FTP sites, you need *mod_proxy_http* or *mod_proxy_ftp* (or both) present in the server.

Warning: Do not enable ProxyRequests until you have secured your server. Open proxy servers are dangerous both to your network and to the Internet at large.

ProxyPass

Description:	Maps remote servers into the local server URL-space
Syntax:	ProxyPass [path] !|url

Context: server config, virtual host, directory
Status: Extension
Module: mod_proxy

This directive allows remote servers to be mapped into the space of the local server; the local server does not act as a proxy in the conventional sense, but appears to be a mirror of the remote server. "[path]" is the name of a local virtual path. "url" is a partial URL for the remote server and cannot include a query string.

Suppose the local server has address http://example.com/; then

```
ProxyPass /mirror/foo/ http://backend.example.com/
```

causes a local request for http://example.com/mirror/foo/bar to be internally converted into a proxy request to http://backend.example.com/bar.

When used inside a *<Location>* section, the first argument is omitted and the local directory is obtained from the *<Location>.* The example above could be re-written as:

```
<Location /mirror/foo/>
    ProxyPass http://backend.example.com/
</Location>
```

The ! directive is useful in situations where you do not want to reverse-proxy a subdirectory. For example

```
ProxyPass /mirror/foo/i !
ProxyPass /mirror/foo http://backend.example.com
```

proxies all requests to /mirror/foo to backend.example.com except requests made to /mirror/foo/i.

Note: Order is important. You need to put the exclusions before the general ProxyPass directive.

The *ProxyRequests* directive should usually be set off when using *ProxyPass*.

If you require a more flexible reverse-proxy configuration, see the *RewriteRule* directive with the [P] flag.

ProxyReverse

Description: Adjusts the URL in HTTP response headers sent from a reverse proxied server

Syntax: ProxyPassReverse [path] url
Context: server config, virtual host, directory
Status: Extension
Module: mod_proxy

This directive lets Apache adjust the URL in the Location, Content-Location and URI headers on HTTP redirect responses. This is essential when Apache is used as a reverse proxy to avoid by-passing the reverse proxy because of HTTP redirects on the backend servers which stay behind the reverse proxy.

Only the HTTP response headers specifically mentioned above will be rewritten. Apache will not rewrite other response headers, nor will it rewrite URL references inside HTML pages. This means that if the proxied content contains absolute URL references, they will by-pass the proxy. A third-party module that looks inside the HTML and rewrite URL references is Nick Kew's mod_proxy_html.

"[path]" is the name of a local virtual path. "url" is a partial URL for the remote server - the same way they are used for the ProxyPass directive.

For example, suppose the local server has address http://example.com/. Then

```
ProxyPass /mirror/foo/ http://backend.example.com/
ProxyPassReverse /mirror/foo/ http://backend.example.com/
```

not only causes a local request for the http://example.com/mirror/foo/bar to be internally converted into a proxy request to http://backend.example.com/bar (the functionality *ProxyPass* provides here). It also takes care of redirects the server backend.example.com sends: When http://backend.example.com/bar is redirected to http://backend.example.com/quux, Apache adjusts this to http://example.com/mirror/foo/quux before forwarding the HTTP redirect response to the client. Note that the hostname used for constructing the URL is chosen in respect to the setting of the *UseCanonicalName* directive.

When used inside a <Location> section, the first argument is omitted and the local directory is obtained from the <Location>. The first example above could be re-written as:

```
<Location /mirror/foo/>
    ProxyPass http://backend.example.com/
    ProxyPassReverse http://backend.example.com/
</Location>
```

Note: The ProxyPassReverse directive can also be used in conjunction with the proxy pass-through feature (RewriteRule ... [P]) from mod_rewrite because its doesn't depend on a corresponding ProxyPass directive.

Examples

Although the mod_proxy module 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.

Proxy only a specific directory to 4D

This section covers how to use the Mod_proxy module for Apache to make a specific directory on Apache proxy its content from 4D.

```
# start
<IfModule mod_proxy.c>
    ProxyRequests Off
    <Location /4D/>
           ProxyPass http://127.0.0.1:8080/
           ProxyPassReverse http://127.0.0.1:8080/
    </Location>
</IfModule>
# end
```

The above configuration file catches all incoming Apache 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 Apache 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 Mod_proxy module for Apache to proxy SOAP requests to 4D's Web Server.

```
# start
<IfModule mod_proxy.c>
    ProxyRequests Off
    <Location /4DSOAP>
           ProxyPass http://127.0.0.1:8080/4DSOAP
           ProxyPassReverse http://127.0.0.1:8080/4DSOAP
    </Location>
</IfModule>
# end
```

The above configuration file catches all incoming Apache 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 Mod_proxy module for Apache to proxy SOAP requests to 4D and make a specific directory on Apache proxy its content from 4D.

```
# start
<IfModule mod_proxy.c>
   ProxyRequests Off
   <Location /4D/>
          ProxyPass http://127.0.0.1:8080/
          ProxyPassReverse http://127.0.0.1:8080/
   </Location>
   <Location /4DSOAP>
          ProxyPass http://127.0.0.1:8080/4DSOAP
          ProxyPassReverse http://127.0.0.1:8080/4DSOAP
   </Location>
</IfModule>
# end
```

The above configuration file catches all incoming Apache 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 Mod_proxy module for Apache to proxy all of its content from 4D.

```
# start
<IfModule mod_proxy.c>
   ProxyRequests Off
   <Location />
          ProxyPass http://127.0.0.1:8080/
          ProxyPassReverse http://127.0.0.1:8080/
   </Location>
</IfModule>
# end
```

The above configuration file catches incoming Apache 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 Apache server. The root of the Apache 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 Apache using the mod_proxy module. Various configurations were discussed in this Technical Note. This information should give the 4D Developer the knowledge necessary to provide an Apache front end to their 4D Web Server if their clients request it.

More information on the Apache HTTP Server can be found here:
http://httpd.apache.org/

Information on Apache Licensing can be found here:
http://www.apache.org/licenses/

More information on the Mod_Proxy module:
http://httpd.apache.org/docs/2.0/mod/mod_proxy.html

Community Support and Mailing Lists can be found here:
http://httpd.apache.org/lists.html