Pacfiles R iRules?

The Bhattman, 2011-07-01

by Bhattman @ gmail dot com

Pac Files are used throughout various companies to direct traffic out proxy systems but is not well understood by most and how it relates to F5 ADCs. Hopefully this article can clear it up for you and give you a better how idea of how to use it with the ADC.

What are pac file? A pac file is a text file containing JavaScript code. It is commonly called proxy.pac, but it can be named differently. It's mostly used for 2 primary reasons:

- Companies have multiple proxies where traffic is directed through, either for redundancy or access to networks that are normally closed off.
- Split domain, where a single domain (i.e., domain.com) is hosted both internally and externally.

A very basic pac file contains a single function. It's function is to direct users through a proxy or proxies based on a URL or website request. A more advanced pacfile script can direct users out through a proxy or proxies based on user's IP address or subnet.

The pac file settings are usually the following link *http://pac.webserver.com/proxy.pac*. Pac files can also be hosted on different ports where the links for eample port 7070 (i.e., *http://pac.webserver.com:7070/proxy.pac)*

The pac file setting is configured in the browsers option or preference settings. Most cases they are labeled clearly.

The pac file can be hosted on the user's PC or it can be hosted on a web server (i.e. Apache or IIS). Most companies host in a web server, for centralized control. It's important to remember that, so far no browser supports more then one pacfile at the user browser level. However, there are certain 3rd party extensions that can be used to get around this limitation. However, it's usually designed for a specific browser rather then a universal version.

So now you understand what a pac file is, what does a standard syntax look like?

Here is very basic example:

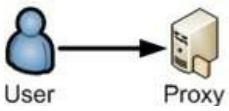
```
1: function FindProxyForURL(url, host) {
2:
3:
4:     if (shExpMatch(url, "http://10.*")||
5:     shExpMatch(url, "https://10.*")||
6:     shExpMatch(url, "ftp://10.*")||
```

```
    shExpMatch(url, "http://localhost*")||
    shExpMatch(url, "https://localhost*")||
    shExpMatch(url, "http://127.0.0.1*")||
    shExpMatch(url, "https://127.0.0.1*")||
    shExpMatch(url, "http://172.*")||
    shExpMatch(url, "https://172.*")||
    shExpMatch(url, "ftp://172.*"))
    return "DIRECT";


    if (dnsDomainIs(host, "insidesite.com")||
    dnsDomainIs(host, "servername.google.com"))
    return "DIRECT";

  return "PROXY proxy.internaldomain.com:8080";
  }
}
```

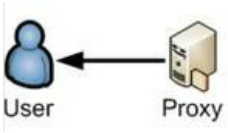
Simple? It's actually simple once you know what each lines are doing. Let me explain: Lines 4 to 14 state that if a user entered a HTTP request that's a private address or a localhost or loopback, should go direct and not use a proxy. Lines 17 to 19 state that if a user entered a HTTP request where the hostname matches either those 2 entries - should go direct and not use a proxy. Line 21 is the catch all – if the user's HTTP request does not match any of the IF conditional statements, the script forces a request to go to proxy.internaldomain.com on port 8080. It's important to note that the pacfile scripts are processed at the client and NOT on the server that is delivering the pacfile.

Now that you the basic idea of pac file syntax, the next step is to know how it works with respect to user, proxy and a web server holding a webpage.

Let's say the Pac file is hosted on a proxy server also running a web server. Let's assume that the user already has the proxy pac file settings already configured in their browser.



Step 1: When the user starts the internet browser, browser retrieves the pac file .

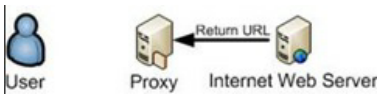


Step 2: The pacfile is downloaded to the browser's cache and process any URL requests (For example the user's browser could have home page going to www.cnn.com) before it let's the user enter information in the URL

Step 3: The user then enters in the URL http://www.google.com

Step 4: The browser runs through the pacfile script like the one above, attempting to match to a IF conditional statement. In our example it won't match any IF statement statement – thus by default is sent to "PROXY proxy.internaldomain.com:8080"

Step 5: Proxy makes a request to the internet web server and requests for www.google.com page on behalf of the user.



Step 6: The page is returned to the proxy.

Step 7: As a final step, the proxy sends the request web page to user.

Okay this is great. What does this have to do with ADCs? As it turns out there is one disadvantage for pacfiles – hosting a pacfile. It requires a web server. While the thought of hosting a web page might not be earth shattering it does become a problem when you run into the following scenario.

- Your IT organization might mandate a reduction in the amount of web servers in the environment.
- Your IT organization could contain a mixture of Squid, Bluecoat, and/or Ironport proxy systems. Each proxy has their own distinctive method to deliver a pac file, some might provide you flexibility to control your pac file deliver and some might not.
- Your IT organization wants to deliver different pac files based on users location or type of browser or IP addresses.

What if you can host the pac file in an iRule? A centralized and consistent approach to deliver a pac file(s) to a user without requiring a web server OR relying on proxy vendors.

The iRule

IMPORTANT NOTE: From this point the iRule is written to conform with v10

So how do we get an iRule to deliver a pacfile?

At this point we know the following:

- Pac file is a JavaScript code that the browsers understands
- Pac file is delivered via HTTP REQUEST and RESPONSE

```
function FindProxyForURL(url, host) {

 if (isPlainHostName(host))

 return "DIRECT";

 

 if (shExpMatch(url, "http://10.*")||

 shExpMatch(url, "https://10.*")||

 shExpMatch(url, "ftp://10.*")||

 shExpMatch(url, "http://localhost*")||
```

```
shExpMatch(url, "https://localhost*")||
```

```
shExpMatch(url, "http://127.0.0.1*")||
```

```
shExpMatch(url, "https://127.0.0.1*")||
```

```
shExpMatch(url, "http://172.*")||
```

```
shExpMatch(url, "https://172.*")||
```

```
shExpMatch(url, "ftp://172.*"))
```

```
return "DIRECT";
```

```

```

```
if (dnsDomainIs(host, "inside.com")||
```

```
dnsDomainIs(host, "inside.net")||
```

```
dnsDomainIs(host, "outside.net"))
```

```
return "DIRECT";
```

```

```

```
    return "PROXY proxy.internaldomain.com:8080";
```

```
 }
```

```
}
```

What's the first step? Since the script needs to be human readable with indents and spaces we need to set the entire Pac file script into a global variable for example "pacfile"

```
1:  set static::pacfile {
```

```
2:  function FindProxyForURL(url, host) {
3:
4:    if (isPlainHostName(host))
5:    return "DIRECT";
6:
7:    if (shExpMatch(url, "http://10.*")||
8:    shExpMatch(url, "https://10.*")||
9:    shExpMatch(url, "ftp://10.*")||
10:   shExpMatch(url, "http://localhost*")||
11:   shExpMatch(url, "https://localhost*")||
12:   shExpMatch(url, "http://127.0.0.1*")||
13:   shExpMatch(url, "https://127.0.0.1*")||
14:   shExpMatch(url, "http://172.*")||
15:   shExpMatch(url, "https://172.*")||
16:   shExpMatch(url, "ftp://172.*"))
17:   return "DIRECT";
18:
19:
20:   if (dnsDomainIs(host, "inside.com")||
21:   dnsDomainIs(host, "inside.net")||
```

```
22:   dnsDomainIs(host, "outside.net"))
23:   return "DIRECT";
24:
25:  return "PROXY proxy.internaldomain.com:8080";
26:  }
27: }
28:
```

The event to use is when RULE_INIT since we are setting the script into a global variable. Your code looks like the following

```
1: when RULE_INIT {
2:  set static::pacfile {
3:  function FindProxyForURL(url, host) {
4:
5:   if (isPlainHostName(host))
6:   return "DIRECT";
7:
8:   if (shExpMatch(url, "http://10.*")||
9:   shExpMatch(url, "https://10.*")||
10:   shExpMatch(url, "ftp://10.*")||
11:   shExpMatch(url, "http://localhost*")||
```

```
12:   shExpMatch(url, "https://localhost*")||
13:   shExpMatch(url, "http://127.0.0.1*")||
14:   shExpMatch(url, "https://127.0.0.1*")||
15:   shExpMatch(url, "http://172.*")||
16:   shExpMatch(url, "https://172.*")||
17:   shExpMatch(url, "ftp://172.*"))
18:   return "DIRECT";
19:
20:
21:   if (dnsDomainIs(host, "inside.com")||
22:   dnsDomainIs(host, "inside.net")||
23:   dnsDomainIs(host, "outside.net"))
24:   return "DIRECT";
25:
26:  return "PROXY proxy.internaldomain.com:8080";
27:  }
28:  }
29: }
```

The next step is to use HTTP_REQUEST event as the trigger for the request and use HTTP::response command to send the the contents of the pacfile back to the requestor. We also need to include a MIME type so the browser knows that proxy pac file is being send. This is a requirement. We also need to make sure that requestor's browser does not cache the file for any length of time. This is because we want to make updates and get immediate changes to the requestor.

Based on this the code should look something like the following

```
1: when HTTP_REQUEST {
2:  switch -glob [HTTP::uri] {
3:    "/proxy.pac" {
4:   HTTP::respond 200 content $static::pacfile "Content-Type" "application/x-ns-proxy-autoconfig" "pragma" "n
5:   return
6:                }
7:  }
8: }
```

Putting it all together the entire code should look like the following

```
when RULE_INIT {
  set static::pacfile {
  function FindProxyForURL(url, host) {

    if (isPlainHostName(host))
    return "DIRECT";

    if (shExpMatch(url, "http://10.*")||
    shExpMatch(url, "https://10.*")||
    shExpMatch(url, "ftp://10.*")||
    shExpMatch(url, "http://localhost*")||
    shExpMatch(url, "https://localhost*")||
    shExpMatch(url, "http://127.0.0.1*")||
    shExpMatch(url, "https://127.0.0.1*")||
    shExpMatch(url, "http://172.*")||
    shExpMatch(url, "https://172.*")||
    shExpMatch(url, "ftp://172.*"))
    return "DIRECT";

```

```
20:     if (dnsDomainIs(host, "inside.com")||
21:     dnsDomainIs(host, "inside.net")||
22:     dnsDomainIs(host, "outside.net"))
23:     return "DIRECT";
24:
25:   return "PROXY proxy.internaldomain.com:8080";
26:   }
27:   }
28: }
29:
30: when HTTP_REQUEST {
31:   switch -glob [HTTP::uri] {
32:     "/proxy.pac" {
33:       HTTP::respond 200 content $static::pacfile "Content-Type" "application/x-ns-proxy-autoconfig" "pragma
34:       return
35:     }
36:   }
37: }
```

As you can see you now have the ability to deliver a pac file.

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc.
Corporate Headquarters
info@f5.com

F5 Networks
Asia-Pacific
apacinfo@f5.com

F5 Networks Ltd.
Europe/Middle-East/Africa
emeainfo@f5.com

F5 Networks
Japan K.K.
f5j-info@f5.com