Reverse Proxy Installation for Windows

KB16016

See the ShoreTel Planning and Installation guide appendix on "Installing and Configuring Reverse Proxy Servers for ShoreTel…" for details on using a Linux Apache Server as a reverse proxy.

You can get Apache for Windows from this link…

http://httpd.apache.org/docs/2.0/platform/windows.html#down

Note:* *Setting up security in Apache is ultimately your responsibility. This information is being provided simply as an example!

However, before we dive into the setup of configuration files, you should double check that you have the correct SSL Certificate for Apache.

Apache Certificate Info:

Although Apache can operate on Windows, the SSL Certificate is not in the same format as a Certificate for Windows.

Windows Formatted Certificate:

A Certificate for Windows is usually in a PKCS#12/PFX format. These Certificates usually end with .pfx or .p12. This format is a binary format that imbeds the certificate, any intermediate certificates, and the private key into one encrypted file. They can also be in a PKSCS#7 or P7B format. These Certificates usually end with .p7b or .p7c.

Apache Formatted Certificate:

A Certificate for Apache is usually in a PEM format. These Certificates usually end with .pem, .crt, .cer. and .key. This format is a Base64 encoded ASCII file. See an example of this format below.

Reverse Proxy Installation for Windows

KB16016

Converting a Certificate from PFX to PEM

Most SSL Certificate providers give the Certificate in both formats. However, if they do not provide non-windows Certificates, you will need to convert the Certificate to the format used by Apache. Here are some basic instructions to convert the Certificate.

If these instructions do not work for you, please refer to the OpenSSL guide to complete this task. In the example below, you will need to use **OpenSSL from the command line.**

1. Open the command line on Windows and navigate to the OpenSSL installation on your server. Typically this would be located here: "C:\Apache Software Foundation\Apache2.2\bin"

2. Next, enter this command: "openssl pkcs12 -in filename.pfx -nocerts -out key.pem"

   This exports the private key file from the pfx file

3. Next, enter this command "openssl pkcs12 -in filename.pfx -clcerts -nokeys -out cert.pem"

   This exports the certificate file from the .pfx file

4. Next, enter this command "openssl rsa -in key.pem -out server.key"

   This removes the passphrase from the private key so Apache does not prompt you for your password when it is starting.

5. Change the .pem to .crt or you can use OpenSSL to convert the between. Actually, as mentioned above, these formats are that same. In the example below, I use ".crt"

   This removes the passphrase from the private key so Apache does not prompt you for your password when it is starting.

   Again, hopefully your provider gave you the correct SSL Certificate.

Reverse Proxy Installation for Windows

KB16016

Update APACHE for Windows

Everything in this first section is the same as the current Appendix in the ShoreTel Installation Guide "**Installing and Configuring Reverse Proxy Servers for ShoreTel…"**, however, the section "#   SSL Engine Switch:" is copied from a Windows Apache Configuration File.

For the Certificate Files, these files are used:

1. server.crt – This is the actual Certificate
2. NetworkSolutions_CA.crt – This is the intermediate given by Network Solutions. **Note:** This is optional and depends on the Certificate Provider.
3. server.key – this is the private key

I have highlighted and placed in **bold** the lines which point the Certificate Files.

Okay, here are the options for the httpd-vhosts.conf file:

```
Listen 5500

# Use name-based virtual hosting.

NameVirtualHost *:5500

<VirtualHost *:5500>

    RewriteEngine on
    RewriteLog "logs/devnosproxy.localhost-rewrite.log"
    RewriteLogLevel 3
    RewriteRule ^/theme/(.+)$ /director2/theme/$1 [P]
    RewriteRule ^/yui_2.7.0/(.+)$ /director2/yui_2.7.0/$1 [P]
    RewriteRule ^/js/(.+)$ /director2/js/$1 [P]

    ProxyPass /authenticate/        http://10.0.0.1/
    ProxyPassReverse /authenticate/     http:// 10.0.0.1/

    ProxyPass /cas/ http:// 10.0.0.1:5447/
    ProxyPassReverse /cas/ http:// 10.0.0.1:5447/

    ProxyPass /director2/ http:// 10.0.0.1:5449/
    ProxyPassReverse /director2/ http:// 10.0.0.1:5449/
    ErrorLog "logs/devnosproxy.localhost-error.log"
    CustomLog "logs/devnosproxy.localhost-access.log" combined


#   SSL Engine Switch:
#   Enable/Disable SSL for this virtual host.
SSLEngine on

#   SSL Cipher Suite:
#   List the ciphers that the client is permitted to negotiate.
#   See the mod_ssl documentation for a complete list.
SSLCipherSuite
ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL

#   Server Certificate:
#   Point SSLCertificateFile at a PEM encoded certificate.  If
#   the certificate is encrypted, then you will be prompted for a
#   pass phrase.  Note that a kill -HUP will prompt again.  Keep
#   in mind that if you have both an RSA and a DSA certificate you
#   can configure both in parallel (to also allow the use of DSA
#   ciphers, etc.)
SSLCertificateFile "C:/Apache Software
Foundation/Apache2.2/conf/server.crt"
#SSLCertificateFile "C:/Apache Software Foundation/Apache2.2/conf/server-
dsa.crt"

#   Server Private Key:
#   If the key is not combined with the certificate, use this
#   directive to point at the key file.  Keep in mind that if
#   you've both a RSA and a DSA private key you can configure
#   both in parallel (to also allow the use of DSA ciphers, etc.)
SSLCertificateKeyFile "C:/Apache Software
Foundation/Apache2.2/conf/server.key"
#SSLCertificateKeyFile "C:/Apache Software Foundation/Apache2.2/conf/server-
dsa.key"

#   Server Certificate Chain:
#   Point SSLCertificateChainFile at a file containing the
#   concatenation of PEM encoded CA certificates which form the
#   certificate chain for the server certificate. Alternatively
#   the referenced file can be the same as SSLCertificateFile
#   when the CA certificates are directly appended to the server
#   certificate for convinience.
#SSLCertificateChainFile "C:/Apache Software Foundation/Apache2.2/conf/server-ca.crt"

SSLCertificateChainFile "C:/Apache Software Foundation/Apache2.2/conf/NetworkSolutions_CA.crt"

#   Certificate Authority (CA):
#   Set the CA certificate verification path where to find CA
#   certificates for client authentication or alternatively one
#   huge file containing all of them (file must be PEM encoded)
#   Note: Inside SSLCACertificatePath you need hash symlinks
#         to point to the certificate files. Use the provided
#         Makefile to update the hash symlinks after changes.
#SSLCACertificatePath "C:/Apache Software Foundation/Apache2.2/conf/ssl.crt"
#SSLCACertificateFile "C:/Apache Software Foundation/Apache2.2/conf/ssl.crt/ca-bundle.crt"

#   Certificate Revocation Lists (CRL):
#   Set the CA revocation path where to find CA CRLs for client
#   authentication or alternatively one huge file containing all
#   of them (file must be PEM encoded)
#   Note: Inside SSLCARevocationPath you need hash symlinks
#         to point to the certificate files. Use the provided
#         Makefile to update the hash symlinks after changes.
#SSLCARevocationPath "C:/Apache Software Foundation/Apache2.2/conf/ssl.crl"
#SSLCARevocationFile "C:/Apache Software Foundation/Apache2.2/conf/ssl.crl/ca-bundle.crl"

#   Client Authentication (Type):
#   Client certificate verification type and depth.  Types are
#   none, optional, require and optional_no_ca.  Depth is a
#   number which specifies how deeply to verify the certificate
#   issuer chain before deciding the certificate is not valid.
#SSLVerifyClient require
#SSLVerifyDepth  10

#   Access Control:
#   With SSLRequire you can do per-directory access control based
#   on arbitrary complex boolean expressions containing server
#   variable checks and other lookup directives.  The syntax is a
#   mixture between C and Perl.  See the mod_ssl documentation
#   for more details.
#<Location />
#SSLRequire (    %{SSL_CIPHER} !~ m/^(EXP|NULL)/ \
#            and %{SSL_CLIENT_S_DN_O} eq "Snake Oil, Ltd." \
#            and %{SSL_CLIENT_S_DN_OU} in {"Staff", "CA", "Dev"} \
#            and %{TIME_WDAY} >= 1 and %{TIME_WDAY} <= 5 \
#            and %{TIME_HOUR} >= 8 and %{TIME_HOUR} <= 20       ) \
#           or %{REMOTE_ADDR} =~ m/^192\.76\.162\.[0-9]+$/
#</Location>

#   SSL Engine Options:
#   Set various options for the SSL engine.
#   o FakeBasicAuth:
#     Translate the client X.509 into a Basic Authorisation.  This means that
#     the standard Auth/DBMAuth methods can be used for access control.  The
#     user name is the `one line' version of the client's X.509 certificate.
#     Note that no password is obtained from the user. Every entry in the user
#     file needs this password: `xxj31ZMTZzkVA'.
#   o ExportCertData:
#     This exports two additional environment variables: SSL_CLIENT_CERT and
#     SSL_SERVER_CERT. These contain the PEM-encoded certificates of the
#     server (always existing) and the client (only existing when client
#     authentication is used). This can be used to import the certificates
#     into CGI scripts.
#   o StdEnvVars:
#     This exports the standard SSL/TLS related `SSL_*' environment variables.
#     Per default this exportation is switched off for performance reasons,
#     because the extraction step is an expensive operation and is usually
#     useless for serving static content. So one usually enables the
#     exportation for CGI and SSI requests only.
#   o StrictRequire:
#     This denies access when "SSLRequireSSL" or "SSLRequire" applied even
#     under a "Satisfy any" situation, i.e. when it applies access is denied
#     and no other module can change it.
#   o OptRenegotiate:
#     This enables optimized SSL connection renegotiation handling when SSL
#     directives are used in per-directory context.
#SSLOptions +FakeBasicAuth +ExportCertData +StrictRequire
<FilesMatch "\.(cgi|shtml|phtml|php)$">
    SSLOptions +StdEnvVars
```

```
</FilesMatch>
<Directory "C:/Apache Software Foundation/Apache2.2/cgi-bin">
   SSLOptions +StdEnvVars
</Directory>

#   SSL Protocol Adjustments:
#   The safe and default but still SSL/TLS standard compliant shutdown
#   approach is that mod_ssl sends the close notify alert but doesn't wait for
#   the close notify alert from client. When you need a different shutdown
#   approach you can use one of the following variables:
#   o ssl-unclean-shutdown:
#     This forces an unclean shutdown when the connection is closed, i.e. no
#     SSL close notify alert is send or allowed to received.  This violates
#     the SSL/TLS standard but is needed for some brain-dead browsers. Use
#     this when you receive I/O errors because of the standard approach where
#     mod_ssl sends the close notify alert.
#   o ssl-accurate-shutdown:
#     This forces an accurate shutdown when the connection is closed, i.e. a
#     SSL close notify alert is send and mod_ssl waits for the close notify
#     alert of the client. This is 100% SSL/TLS standard compliant, but in
#     practice often causes hanging connections with brain-dead browsers. Use
#     this only for browsers where you know that their SSL implementation
#     works correctly.
#   Notice: Most problems of broken clients are also related to the HTTP
#   keep-alive facility, so you usually additionally want to disable
#   keep-alive for those clients, too. Use variable "nokeepalive" for this.
#   Similarly, one has to force some clients to use HTTP/1.0 to workaround
#   their broken HTTP/1.1 implementation. Use variables "downgrade-1.0" and
#   "force-response-1.0" for this.
BrowserMatch ".*MSIE.*" \
         nokeepalive ssl-unclean-shutdown \
         downgrade-1.0 force-response-1.0

#   Per-Server Logging:
#   The home of a custom SSL log file. Use this when you want a
#   compact non-error SSL logfile on a virtual host basis.
CustomLog "C:/Apache Software Foundation/Apache2.2/logs/ssl_request.log" \
          "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"

</VirtualHost>
```