

RedirectSSL

Redirect Request to SSL

Let's say you want <http://www.example.com/secure/> to always be sent over SSL (I presume here that both the normal and the SSL vhost have the same content). You could do this by linking to the correct page from within your HTML pages... but there will always be some user who will sneak by it that way.

- [Redirect Request to SSL](#)
 - [Using virtual hosts \(using redirect\)](#)
 - [Using .htaccess files and redirect](#)
 - [Using mod_rewrite](#)

Using virtual hosts (using redirect)

When using SSL, you will frequently have at least two virtual hosts: one on port 80 to serve ordinary requests, and one on port 443 to serve SSL. If you wish to redirect users from the non-secure site to the SSL site, you can use an ordinary [Redirect](#) directive inside the non-secure [VirtualHost](#):

Note: The [NameVirtualHost](#) directive only applies to the 2.2.x releases of httpd.

```
NameVirtualHost *:80
<VirtualHost *:80>
    ServerName mysite.example.com
    DocumentRoot /usr/local/apache2/htdocs
    Redirect /secure https://mysite.example.com/secure
</VirtualHost>

<VirtualHost _default_:443>
    ServerName mysite.example.com
    DocumentRoot /usr/local/apache2/htdocs
    SSLEngine On
# etc...
</VirtualHost>
```

When redirecting everything you don't even need a DocumentRoot:

```
NameVirtualHost *:80
<VirtualHost *:80>
    ServerName www.example.com
    Redirect / https://secure.example.com/
</VirtualHost>

<VirtualHost _default_:443>
    ServerName secure.example.com
    DocumentRoot /usr/local/apache2/htdocs
    SSLEngine On
# etc...
</VirtualHost>
```

Note: Once the configuration is working as intended, a permanent redirection can be considered. This avoids caching issues by most browsers while testing. The directive would then become:

```
Redirect permanent / https://secure.example.com/
```

Using .htaccess files and redirect

Redirect can also be used inside .htaccess files or to address particular URLs, as in:

Example:

```
Redirect /login https://mysite.example.com/login
```

Using mod_rewrite

While the `<VirtualHost>` solution is recommended because it is simpler and safer, you can also use `mod_rewrite` to get the same effect as described here: [RewriteHTTPToHTTPS](#)