

# Error Handling

When you're running a Web site, things go wrong. And when they do, it's important that they are handled gracefully, so that the user experience is not too greatly diminished. In this chapter, you'll learn how to handle error conditions, return useful messages to the user, and capture information that will help you fix the problem so that it does not happen again.

## 9.1 Handling a Missing Host Field

### Problem

You have multiple virtual hosts in your configuration, and at least one of them is name-based. For name-based virtual hosts to work properly, the client must send a valid `Host` field in the request header. This recipe describes how you can deal with situations in which the field is *not* included.

### Solution

Add the following lines to your *httpd.conf* file:

```
Alias /NoHost.cgi /usr/local/apache/cgi-bin/NoHost.cgi
RewriteEngine On
RewriteCond "%{HTTP_HOST}" "^$"
RewriteRule "(.*)" "/NoHost.cgi$1" [PT]
```

The file *NoHost.cgi* can contain something like the following:

```
#!/usr/bin/perl -Tw

my $msg = "To properly direct your request, this server requires that\n"
. "your Web client include the HTTP 'Host' request header field.\n"
. "The request which caused this response did not include such\n"
. "a field, so we cannot determine the correct document for you.\n";
print "Status: 400 Bad Request\r\n\r\n"
. "Content-type: text/plain\r\n\r\n"
. "Content-length: " . length($msg) . "\r\n\r\n"
. "\r\n\r\n"
```

```
. $msg;  
exit(0);
```

## Discussion

Once the directives in the solution are in place, all requests made of the server that do not include a `Host` field in the request header will be redirected to the specified CGI script, which can take appropriate action.

The solution uses a CGI script so that the response text can be tailored according to the attributes of the request and the server's environment. For instance, the script might respond with a list of links to valid sites on the server, determined by the script at runtime by examining the server's own configuration files. If all you need is a "please try again, this time with a `Host` field" sort of message, a static HTML file would suffice. Replace the *RewriteRule* directive in the solution with that below, and create the *no host.html* accordingly.

```
RewriteRule ".*" "/nohost.html" [PT]
```

A more advanced version of the script approach could possibly scan the *httpd.conf* file for *ServerName* directives, construct a list of possibilities from them, and present links in a 300 `Multiple Choices` response. Of course, there's an excellent chance they wouldn't work, because the client would *still* not be including the `Host` field.

## See Also

- [http://httpd.apache.org/docs/mod/mod\\_rewrite.html](http://httpd.apache.org/docs/mod/mod_rewrite.html)

## 9.2 Changing the Response Status for CGI Scripts

### Problem

There may be times when you want to change the status for a response—for example, you want 404 `Not Found` errors to be sent back to the client as 403 `Forbidden` instead.

### Solution

Point your *ErrorDocument* to a CGI script instead of a static file. The CGI specification permits scripts to specify the response status code.

In addition to the other header fields the script emits, like the `Content-type` field, include one named `Status` with the value and text of the status you want to return:

```
#!/bin/perl -w  
print "Content-type: text/html;charset=iso-8859-1\r\n";  
print "Status: 403 Access denied\r\n";  
:
```

## Discussion

If Apache encounters an error processing a document, such as not being able to locate a file, by default it will return a canned error response to the client. You can customize this error response with the *ErrorDocument* directive, and Apache will generally maintain the error status when it sends your custom error text to the client.

However, if you want to change the status to something else, such as hiding the fact that a file doesn't exist by returning a Forbidden status, you need to tell Apache about the change.

This requires that the *ErrorDocument* be a dynamic page, such as a CGI script. The CGI specification provides a very simple means of specifying the status code for a response: the *Status* CGI header field. The Solution shows how it can be used.

## See Also

- Chapter 8
- <http://httpd.apache.org/docs/mod/core.html#errordocument>
- [http://www.rfc-editor.org/cgi-bin/rfcdoctype.pl?loc=RFC&lets-go=387&type=ftp&file\\_format=txt](http://www.rfc-editor.org/cgi-bin/rfcdoctype.pl?loc=RFC&lets-go=387&type=ftp&file_format=txt)

ampersands..

## 9.3 Customized Error Messages

### Problem

You want to display a customized error message, rather than the default Apache error page.

### Solution

Use the *ErrorDocument* directive in *httpd.conf*:

```
ErrorDocument 405 /errors/notallowed.html
```

### Discussion

The *ErrorDocument* directive allows you to create your own error pages to be displayed when particular error conditions occur. In the previous example, in the event of a 405 status code (*Method Not Allowed*), the specified URL is displayed for the user, rather than the default Apache error page.

The page can be customized to look like the rest of your Web site. When an error document looks significantly different from the rest of the site, this can leave the user feeling disoriented, or she may feel as if she has left the site that she was on.

## See Also

- <http://httpd.apache.org/docs/mod/core.html#error-document>

## 9.4 Providing Error Documents in Multiple Languages

### Problem

On a multilingual (content-negotiated) Web site, you want your error documents to be content-negotiated as well.

### Solution

The Apache 2.0 default configuration file contains a configuration section, initially commented out, that allows you to provide error documents in multiple languages customized to the look of your Web site, with very little additional work.


Uncomment those lines. You can identify the lines by looking for the following comment in your default configuration file:

```
# The internationalized error documents require mod_alias, mod_include
# and mod_negotiation. To activate them, uncomment the following 30 lines.
```

In Apache 1.3 this is harder, but there's a solution in the works, as of this writing, that will make it similar to the 2.0 implementation. Check the Apache Cookbook Web site for more information.

### Discussion

The custom error documents provided with Apache 2.0 combine a variety of techniques to provide internationalized error messages. As of this writing, these error messages are available in German, English, Spanish, French, Dutch, Swedish, Italian, and Portuguese. Based on the language preference set in the client browser, the error message is delivered in the preferred language of the end user.

Using content negotiation, the correct variant of the document (i.e., the right language) is selected for the user, based on her browser preference settings. For more information about content negotiation, see the content negotiation documentation at <http://httpd.apache.org/docs-2.0/content-negotiation.html> (for Apache 2.0) or <http://httpd.apache.org/docs/content-negotiation.html> (for Apache 1.3). 

In addition to delivering the error message in the correct language, this functionality also lets you customize the look of these error pages so that they resemble the rest of your Web site. To facilitate this, the files *top.html* and *bottom.html*, located in the *include* subdirectory of the *error* directory, should be modified to look like the standard header and footer content that appears on your Web site. The body of the error message documents is placed between the header and the footer to create a page that is less

jarring to users when they transition from your main site to the error pages that are generated.

You also will note that the error documents themselves contain SSI directives, which are used to further customize the error documents for the user. For example, in the case of the 404 (file not found) error document, the page will provide a link back to the page that the user came from, if the environment variable `HTTP_REFERER` is defined, and if that variable is not found, the page will merely notify the user that the URL was not found. Other SSI directives may be put in these documents, if you wish, to further customize them.

## See Also

- <http://httpd.apache.org/docs/content-negotiation.html>
- <http://httpd.apache.org/docs-2.0/content-negotiation.html>
- <http://Apache-Cookbook.Com/>
- Recipe 8.9

## 9.5 Redirecting Invalid URLs to Some Other Page

### Problem

You want all “not found” pages to go to some other page instead, such as the front page of the site, so that there is no loss of continuity on bad URLs.

### Solution

Use the *ErrorDocument* directive to catch 404 (Not Found) errors:

```
ErrorDocument 404 /index.html
DirectoryIndex index.html /path/to/notfound.html
```

### Discussion

The recipe given here will cause all 404 errors—every time someone requests an invalid URL—to return the URL */index.html*, providing the user with the front page of your Web site, so that even invalid URLs still get valid content. Presumably, users accessing an invalid URL on your Web site will get a page that helps them find the information that they were looking for.

By contrast, this behavior may confuse the user who believes she knows exactly where the URL should take her. Make sure that the page that you provide as the global error document does in fact help people find things on your site, and does not merely confuse or disorient them. You may, as shown in the example, return them to the front page of the site. From there they should be able to find what they were looking for.

When users get good content from bad URLs, they will never fix their bookmarks and will continue to use a bogus URL long after it has become invalid. You will continue to get 404 errors in your log file for these URLs, and the user will never be aware that they are using an invalid URL. If, by contrast, you actually return an error document, they will immediately be aware that the URL they are using is invalid and will update their bookmarks to the new URL when they find it.

Note that, even though a valid document is being returned, a status code of 404 is still returned to the client. This means that if you are using some variety of tool to validate the links on your Web site, you will still get good results, if the tool is checking the status code, rather than looking for error messages in the content.

## See Also

- <http://httpd.apache.org/docs/mod/core.html#errordocument>
- [http://httpd.apache.org/docs/mod/mod\\_dir.html](http://httpd.apache.org/docs/mod/mod_dir.html)

## 9.6 Making Internet Explorer Display Your Error Page

### Problem

You have an *ErrorDocument* directive correctly configured, but IE is displaying its own error page, rather than yours.

### Solution

Make the error document bigger—at least 512 bytes.

### Discussion

Yes, this seems a little bizarre, and it is. In this case, Internet Explorer thinks it knows better than the Web site administrator. If the error document is smaller than 512 bytes, it will display its internal error message page, rather than your custom error page, whenever it receives a 400 or 500 series status code. This size is actually configurable in the browser, so this number may in fact vary from one client to another. “Friendly error messages” also can be turned off entirely in the browser preferences.

This can be extremely frustrating the first time you see it happen, because you just know you have it configured correctly and it seems to work in your other browsers. Furthermore, when some helpful person tells you that your error document just needs to be a little larger, it’s natural to think that he is playing a little prank on you, because this seems a little too far-fetched.

But it’s true. Make the page bigger. It needs to be at least 512 bytes, or IE will ignore it and gleefully display its own “friendly” error message instead.

Exactly what you fill this space with is unimportant. You can, for example, just bulk it up with comments. For example, repeating the following comment six times would be sufficient to push you over that minimum file size:

```
<!-- message-obscuring clients are an abomination
      and an insult to the user's intelligence -->
```

## See Also

- <http://httpd.apache.org/docs/mod/core.html#errordocument>

## 9.7 Notification on Error Conditions

### Problem

You want to receive email notification when there's an error condition on your server.

### Solution

Point the *ErrorDocument* directive to a CGI program that sends mail, rather than to a static document:

```
ErrorDocument 404 /cgi-bin/404.cgi
```

*404.cgi* looks like the following:

```
#!/usr/bin/perl
use Mail::Sendmail;
use strict;

my $message = qq~
Document not found: $ENV{REQUEST_URI}
Link was from: $ENV{HTTP_REFERER}
~;

my %mail = (
    To => 'admin@server.com',
    From => 'website@server.com',
    Subject => 'Broken link',
    Message => $message,
);
sendmail(%mail);

print "Content-type: text/plain\n\n";
print "Document not found. Admin has been notified\n";
```

### Discussion

This recipe is provided as an example, rather than as a recommendation. On a Web site of any significant size or traffic level, actually putting this into practice generates a substantial quantity of email, even on a site that is very well maintained. This is because

people mistype URLs, and other sites, over which you have no control, will contain incorrect links to your site. It may be educational, however, to put something like this in place, at least briefly, to gain an appreciation for the scale of your own Web site.

The *ErrorDocument* directive will cause all 404 (Document Not Found) requests to be handled by the specified URL, and so your CGI program gets run and is passed environment variables that will be used in the script itself to figure out what link is bad and where the request came from.

The script used the *Mail::Sendmail* Perl module to deliver the email message, and this module should work fine on any operating system. The module is not a standard part of Perl, so you may have to install it from CPAN (<http://www.cpan.org/>). A similar effect can, of course, also be achieved in PHP or any other programming language.

The last two lines of the program display a very terse page for the user, telling him that there was an error condition. You may wish, instead, to have the script redirect the user to some more informative and attractive page elsewhere on your Web site. This could be accomplished by replacing those last two lines with something like the following:

```
print "Location: http://server.name/errorpage.html\n\n";
```

This would send a redirect header to the client, which would display the specified URL to the user.

## See Also

- <http://httpd.apache.org/docs/mod/core.html#errordocument>