Learn how to use the Find and Replace config.txt directives to locate and replace text in EZproxy as web pages are retrieved.

**Find** and **Replace** are position-dependent config.txt directives that always appear as part of a database definition.

These directives always appear paired together, with **Find** appearing directly before its corresponding **Replace**.

These directives identify text that EZproxy should locate and replace as web pages are retrieved. Typical use is to alter JavaScript that may otherwise be incompatible with EZproxy.

**Find** accepts an arbitrary text string that should be located in the web page being retrieved. **Replace** accepts the text that should be used in place of the Find text.

In addition to simple text, **Replace** accepts several special sequences to provide more generic rewriting. These include:

<table>
<thead>
<tr>
<th>SEQUENCE</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>^A</td>
<td>See example below</td>
</tr>
<tr>
<td>^D</td>
<td>Substitute the domain of the EZproxy server without a leading period, such as <code>ezproxy.yourlib.org</code> or <code>68.15.177.100</code></td>
</tr>
<tr>
<td>^H</td>
<td>Substitute the name of the EZproxy server</td>
</tr>
<tr>
<td>^L</td>
<td>Substitute the domain of the EZproxy server with a leading period unless the EZproxy server is using a numeric name, such as <code>ezproxy.yourlib.org</code> or <code>68.15.177.100</code></td>
</tr>
<tr>
<td>^P hostname^</td>
<td>Look up the EZproxy hostname for the http version of <code>hostname</code> and substitute it</td>
</tr>
<tr>
<td>^S hostname^</td>
<td>Look up the EZproxy hostname for the https version of <code>hostname</code> and substitute it</td>
</tr>
</tbody>
</table>

**^A Example**

`^A` is the most unusual **Replace** option. Its only valid placement is right after `http://` or `https://` and it specifies that the hostname that follows should not be rewritten, but rather should be left as-is. It is designed for use with form fields that contain values that are URLs that should not be rewritten.

For example, Lexis-Nexis uses a form field called `_PRIORREFERER` that must be the actual URL of the prior page, not the rewritten form. These statements demonstrate how the **Find** statement provides enough of a search string to
locate this particular form variable, and the **Replace** statement repeats the exact same text adding ^A after http:// to indicate that this is the point at which the URL starts that must not be rewritten:

```
Find NAME="_PRIORREFERER" VALUE="http://
Replace NAME="_PRIORREFERER" VALUE="http://^A"
```

**Advanced Find/Replace using States**

**Find** and **Replace** rules can be associated with states. Through the use of states, the application of rules can be dependent on other rules having already occurred. The need to apply states for Find/Replace rules is infrequent, and in most instances, this capability is not needed.

The syntax for state qualification is:

```
Find -State= state1[+state2+state3...]
FindText
Replace -State= state1[+state2+state3...]
ReplaceText
Replace -AddState= state1[+state2+state3...]
ReplaceText
Replace -RemoveState= state1[+state2+state3...]
ReplaceText
```

There are two special states: start and all. When EZproxy starts processing a page, the default state is start. For a **Find** rule, the special state all will match to any state.

For **Find**, the state(s) specified after -State= must match to the current states for the Find rule to match. Once a Find matches, the **Replace** rule has the option to alter the current state(s) with -State, which specifies exactly which state(s) should be active, with -AddState to add state(s) to be active, with -RemoveState to remove state(s) from being active, or any combination of -State, -AddState, and -RemoveState.

The main application for states is to automate submitting a username/password based form to automate access to a resource. By using states, it is often possible to detect when a first attempt is made to fill in the form, and then trigger the automatic submission of the login form, while also allowing failed attempts to be detected and to have those attempts avoid submission of the login form, avoiding the possibility of an endless loop of browser attempts to access a resource with the wrong username or password.