EZproxy reads the config.txt file when determining whether or not to proxy resources. The first step in understanding how this process works is understanding the construction of URLs. The information in URLs forms the basis for EZproxy's understanding of what resources can be proxied as a starting point for users to begin searches, and what URLs linked on a starting point site can be proxied once a user has begun his or her search.

**Overview**

The relationship between URL components determines what resources need their own database stanzas in the config.txt file and what resources can be added to the larger umbrella stanza under which they fall. Generally this can be determined by first examining whether a content provider offers access to multiple resources and the relationship between the content provider's URL and the URLs of the individual resources within that provider's database. Many content providers operate databases containing numerous journals or other resources, and as such, a carefully constructed, single database stanza will provide users with access to both the homepage of the database and all the journals it contains. This also means that you may not need to update your config.txt file and database stanzas every time you subscribe to a new resource.

The first step in determining how that single database stanza should be created is to examine the URLs of your content provider's homepage and the resources you subscribe to and then to determine the relationship between them.

**URL terminology**

The following definitions are used to describe the different parts of a URL. The simplified definitions given are adequate to understand these terms' use within EZproxy documentation, but they are over-generalized from the terms' exact meanings. Understanding the components of a URL will help you to determine the relationship between the databases you subscribe to and the individual resources, and create better database stanzas.

<table>
<thead>
<tr>
<th>TERM</th>
<th>DEFINITION</th>
<th>EXAMPLES</th>
</tr>
</thead>
</table>
| scheme | The protocol used for retrieval of the URL | • http - indicates an unsecured connection  
• https - indicates a secure connection |

Note: Although many other schemes exist, for the purposes of this document, only these two
<table>
<thead>
<tr>
<th>TERM</th>
<th>DEFINITION</th>
<th>EXAMPLES</th>
</tr>
</thead>
<tbody>
<tr>
<td>hostname</td>
<td>The name or address of the webserver to be accessed. Hostname is not case sensitive.</td>
<td>schemes will be used.</td>
</tr>
<tr>
<td>port</td>
<td>A number used to identify a specific webserver at the provided hostname. When omitted, a scheme-specific default value is used.</td>
<td>Hostname is not case sensitive, the two hostnames above are equivalent.</td>
</tr>
<tr>
<td>origin</td>
<td>The unique combination of a scheme, hostname, and port, combined as <code>scheme://hostname:port</code>.</td>
<td>Note: Because hostnames are not case sensitive, the two hostnames above are equivalent.</td>
</tr>
<tr>
<td>path</td>
<td>The portion of the URL from a slash (/) following the origin up to the query or fragment. When omitted, the default path <code>/</code> is used.</td>
<td><code>80 - the default value for http</code>  <code>443 - the default value for https</code></td>
</tr>
<tr>
<td>query</td>
<td>The portion of the URL from the first question mark (<code>?</code>), following the path, and up to the fragment. If the first question mark in a URL appears after a hash (<code>#</code>), that section is not the query, but rather part of the fragment.</td>
<td><code>http://www.somedb.com:80</code>  <code>https://www.somedb.com:443</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>http://www.somedb.com/subject#period</code>  <code>https://www.anotherdb.com/topic#?modern</code></td>
</tr>
</tbody>
</table>
**Examples**

**How EZproxy reads URL components**

The following discussion provides an introduction to similarities and differences between URLs, based on the terminology in the previous tab. These characteristics impact on the way in which EZproxy determines whether to proxy a resource or not when reading the config.txt file is covered as well. For a more detailed discussion of the different directives used within config.txt and how they impact proxying, please see Config.txt Directives: An Introduction to Database Stanzas.

In general, EZproxy ignores the path, query, and fragment when reading the config.txt file and determining whether to proxy a resource. These additional URL components are only needed when creating the Starting Point URLs. For more information about Starting Point URLs, please see Starting point URLs and config.txt.

### SAMPLE URLs AND THEIR COMPONENTS

<table>
<thead>
<tr>
<th>TERM</th>
<th>DEFINITION</th>
<th>EXAMPLES</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL 1: <a href="http://www.somedb.com">http://www.somedb.com</a></td>
<td>scheme</td>
<td>http</td>
</tr>
<tr>
<td></td>
<td>hostname</td>
<td><a href="http://www.somedb.com">www.somedb.com</a></td>
</tr>
<tr>
<td></td>
<td>port</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>origin</td>
<td><a href="http://www.somedb.com:80">http://www.somedb.com:80</a></td>
</tr>
<tr>
<td></td>
<td>path</td>
<td>/</td>
</tr>
<tr>
<td></td>
<td>query</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fragment</td>
<td></td>
</tr>
</tbody>
</table>

### RELATIONSHIPS BETWEEN URLs

**URL 1 and 2**


are functionally equivalent even though URL 1 uses the default port and URL 2 uses the default path. (Because no port is listed, and the scheme for URL 1 is http, the port defaults to 80, and thus the origin for URL 1, http://www.somedb.com:80 looks just like URL 2). Creating a database stanza using URL 1 would also provide your users with access to URL 2, and vice versa, with URL 2 providing access to URL 1.

**URL 2: http://www.somedb.com:80**

**URLs 1, 2 and 3**
**SAMPLE URLS AND THEIR COMPONENTS**

<table>
<thead>
<tr>
<th>Scheme</th>
<th>http</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hostname</td>
<td><a href="http://www.somedb.com">www.somedb.com</a></td>
</tr>
<tr>
<td>Port</td>
<td>80</td>
</tr>
<tr>
<td>Origin</td>
<td><a href="http://www.somedb.com:80">http://www.somedb.com:80</a></td>
</tr>
<tr>
<td>Path</td>
<td>/</td>
</tr>
<tr>
<td>Query</td>
<td></td>
</tr>
<tr>
<td>Fragment</td>
<td></td>
</tr>
</tbody>
</table>

**RELATIONSHIPS BETWEEN URLS**

URLS 1, 2, and 3

- [http://www.somedb.com](http://www.somedb.com)
- [http://www.somedb.com:80](http://www.somedb.com:80)
- [http://www.somedb.com/search](http://www.somedb.com/search)

All use the same origin, even though 1 and 3 depend on the default port, 2 has an explicit port, and 3 has a path. Creating a database stanza using URL 1, 2, or 3 would provide your users with access to any of these URLs (1, 2, or 3).

**URL 3: http://www.somedb.com/search?q=ancient**

<table>
<thead>
<tr>
<th>Scheme</th>
<th>http</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hostname</td>
<td><a href="http://www.somedb.com">www.somedb.com</a></td>
</tr>
<tr>
<td>Port</td>
<td>80</td>
</tr>
<tr>
<td>Origin</td>
<td><a href="https://www.somedb.com:80">https://www.somedb.com:80</a></td>
</tr>
<tr>
<td>Path</td>
<td>/search</td>
</tr>
<tr>
<td>Query</td>
<td>?q=ancient</td>
</tr>
<tr>
<td>Fragment</td>
<td></td>
</tr>
</tbody>
</table>

**URLS 3 and 4**

- [https://www.somedb.com/search?q=ancient](https://www.somedb.com/search?q=ancient)

Are not functionally equivalent as they use different schemes. These URLs would need to be listed separately in a database stanza in order for users to access them.

**URL 4: https://www.somedb.com/search?q=ancient**

<table>
<thead>
<tr>
<th>Scheme</th>
<th>https</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hostname</td>
<td><a href="http://www.somedb.com">www.somedb.com</a></td>
</tr>
<tr>
<td>Port</td>
<td>443</td>
</tr>
<tr>
<td>SAMPLE URLs AND THEIR COMPONENTS</td>
<td>RELATIONSHIPS BETWEEN URLS</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>origin <a href="http://www.somedb.com:443">http://www.somedb.com:443</a></td>
<td></td>
</tr>
<tr>
<td>path /search</td>
<td></td>
</tr>
<tr>
<td>query ?q=ancient</td>
<td></td>
</tr>
<tr>
<td>fragment</td>
<td></td>
</tr>
</tbody>
</table>

**URL 5: http://www.somedb.com:8080/history?era=darkages**

- **scheme** http
- **hostname** www.somedb.com
- **port** 8080
- **origin** http://search.somedb.com:8080
- **path** /history
- **query** ?era=darkages
- **fragment**

**URL 6: http://search.somedb.com:8080/history?era=darkages**

- **scheme** http
- **hostname** search.somedb.com
- **port** 8080
- **origin** http://search.somedb.com:8080
- **path** /history

**URLs 5 and 6**

http://www.somedb.com:8080/history?era=darkages and http://search.somedb.com:8080/history?era=darkages are not functionally equivalent as they use different hostnames. Providing access to both of these URLs would require multiple directive lines within a single stanza.
<table>
<thead>
<tr>
<th>SAMPLE URLs AND THEIR COMPONENTS</th>
<th>RELATIONSHIPS BETWEEN URLs</th>
</tr>
</thead>
<tbody>
<tr>
<td>query</td>
<td>?era=darkages</td>
</tr>
<tr>
<td>fragment</td>
<td></td>
</tr>
</tbody>
</table>

**URL 7:** http://search.somedb.com:8080/history/?modern

<table>
<thead>
<tr>
<th>scheme</th>
<th>http</th>
</tr>
</thead>
<tbody>
<tr>
<td>hostname</td>
<td>search.somedb.com</td>
</tr>
<tr>
<td>port</td>
<td>8080</td>
</tr>
<tr>
<td>origin</td>
<td><a href="http://search.somedb.com:8080">http://search.somedb.com:8080</a></td>
</tr>
<tr>
<td>path</td>
<td>/history</td>
</tr>
<tr>
<td>query</td>
<td></td>
</tr>
<tr>
<td>fragment</td>
<td>#?modern</td>
</tr>
</tbody>
</table>

URL 7 does not have a query since the first question mark (?) appears after the first hash (#).