Operating system requirements

EZproxy v6.6.2 is supported under two different operating systems:

- Linux
- Windows

The supported versions of these operating systems along with their minimum hardware requirements can be found at Requirements.

Administrative updates

Changed default config.txt to Proxy By Hostname

The default config.txt has been changed to state that proxy by port is deprecated and to include Option ProxyByHostname by default.

This change does not touch existing config.txt files, so any existing sites using proxy by port will be unaffected by this change.

Enhancements

Encrypted usernames in SSO

The EZproxy SSO Single Sign On (SSO) option previously allowed release of either the EZproxy username or the static value anonymous. It is now possible to specify on a per-SSO configuration what value to release for the username using the new SSOUUsername config.txt directive. The SSOUUsername directive must be used in conjunction with an SSO directive and must appear after the corresponding SSO directive. Usage is:

```
SSOUUsername site expression
```

where site is the SSO site code and expression is an expression to generate the value to be released for the username.

There is also a new Expression function PPID to generate a Private Personal Identifier (PPID) based on a provided relying party code. By using a different relying party code for SSO configuration, unique values can be
EZproxy now uses OpenSSL 1.0.2s

EZproxy v6.6.2 was built with the most current Long Term Support release of OpenSSL (1.0.2s), which was released on May 28, 2019.

BlockCountryChange exceptions

A new user.txt directive AllowCountryChange is introduced which can override the behavior of Option BlockCountryChange for specifically targeted users.

The user.txt example below shows one was to associate a username with this behavior, but any condition/action can be used to apply this behavior to an account similar to how Admin is applied to an account.

user.txt

::Common
IfUser admin2; AllowCountryChange
/Common
admin::admin
admin2::admin

Modify handling of Strict-Transport-Security header

The HTTP Strict-Transport-Security (STS) header has an includeSubdomains option. When this is present, it tells a browser not only to enforce the use of https for the hostname of the current requests, but also for any requests that includes any subdomains of the current hostname. This behavior is incompatible for proxy by hostname.

As an example, consider the site https://www-somedb-com.ezproxy.yourlib.org making this request while the host http://www.otherdb.com.ezproxy.yourlib.org also needs to be handled. If the STS header specifies includeSubdomains, this tells the browser that anything that ends in .ezproxy.yourlib.org must also be forced to https which blocks the other host that requires http.

To prevent this behavior, EZproxy now blocks includeSubdomains in this header.

Socket logging debugging now generates Fiddler compliant log file

EZproxy can now capture the browser to EZproxy and EZproxy to content provider sessions into a Fiddler Session Archive. Admin users and users assigned to the new Admin.Fiddler group can control and use this feature at the new /fiddler URL. A “Fiddler capture” option also appears in the main administration page.

For more information, see Use socket level logging to generate a Fiddler compliant log file.
Increased maximum size of URL

Based on the suggestion of RFC7230 that an HTTP server should support request lines up to 8000 characters, EZproxy now allows URLs to be 8000 characters long. URLs of this length should always work during the proxying process. If a target URL of this length is presented in a starting point URL, though, the URL may end up being truncated if it has to be transformed through the login process in ways that encode it to a longer length.

HTTP headers extended

The HTTPHeader directive has been extended to allow specifying additional response headers that should be sent from EZproxy when it is responding to request when it is acting as its own server such as during login process or presentation of admin pages. The syntax is:

```
HTTPHeader -server header expression
```

header: HTTP header to be sent.

expression: provides the header as hh:header to the user specified expression; if the returned value is non-empty, send the header with that value; if the returned value is empty, suppress the header.

A sample use might be:

```
HTTPHeader -server X-Frame-Options "DENY"
```

to include the X-Frame-Options header with a value of DENY. This combination would prevent the login page from being presented inside a frame.

These directives are global in scope since they affect non-database oriented operation of EZproxy.

These directives cannot be used to override or replace any of the headers already sent by EZproxy. Care should be taken when choosing to add additional headers that are sent back from EZproxy.

Configuration option to change default from encryption of attributes to no encryption

Some SAML services do not accepted encrypted attributes. To allow EZproxy to interact with more local SAML services, we have expanded the ShibbolethMetadata configuration options.

The ShibbolethMetadata config.txt now has three new options when processing a Response document from an IdP:

- `SignResponse=[true|false]` determines whether or not the Response document must be signed. The default value is true consistent with past EZproxy requirements.
- `SignAssertion=[true|false]` determines whether or not the Assertion within the Response must be signed. The default value is false consistent with past EZproxy requirements, but see upgrade warning below.
- `EncryptAssertion=[true|false]` determine whether or not the Assertion within the Response document must be encrypted. The default is true consistent with past EZproxy requirements.
To maximize security, we recommend leaving the default settings in effect, but these values are now configurable to meet the unique site needs.

**SAML Upgrade Notes**

Earlier versions of EZproxy would also accept Response documents in which the Response was not signed, the Assertion was signed, and the Assertion was not encrypted. When updating to this release, any site relying on this behavior will need to add the following to their ShibbolethMetadata directive. After configuring this option, libraries may need to toggle options in the response config to find a setting that matches your local needs.

```
-SignResponse=false -SignAssertion=true -EncryptAssertion=false \
```

Additionally, libraries who have configured their shibuser.txt file to make authorization decisions will need to put quotes around the URN parameter in the If Any(auth:"URL"; "employee").

```
If Any(auth:"URL"; "employee")
```

Libraries should confirm that their usernames are mapping correctly from the local identity provider. In some cases, usernames were not mapped correctly. When viewing the server status screen libraries should see that usernames are displaying correctly relative to your institution (firstInitialLastName, etc), rather than a random hashed string ID. If your library's usernames are showing as a hashed string ID, contact support for more information.

**CAS SP support expanded to include UTF-16**

EZproxy only accepted UTF-8 interactions with CAS IDP servers. EZproxy can now handle CAS IDP interactions correctly with UTF-16.

**CAS: support additional attribute release by EZproxy CAS server**

EZproxy has the ability to act as a CAS Identity Provider to a CAS Service Provider (CAS SP). This functionality has been greatly enhanced. It is enabled in config.txt by adding an entry such as:

```
CASServiceURL [-anonymous] [-Scope=scope] WildURL
```

where *WildURL* is a case-insensitive simple wildcard match (supports * and ?) for a CAS SP to authorize. Although not recommended, the value * can be used to allow any CAS Service Provider to gain access.

**Byte Serving Support**

HTTP supports a feature called byte serving in which a client can request a specific range of bytes from a server. EZproxy previously disabled this functionality since the rewriting process rearranges the contents of a response, thereby invalidating any pre-computed offsets to ranges within the response.

EZproxy has special functionality built-in to detect when PDF files are requested so that it can enable byte serving for these files. Other content providers will benefit from having similar mechanisms available to mark specific content for
byte serving.

EZproxy needs to decide whether or not to pass through the headers used for byte serving before it knows whether or not the response can be rewritten which poses a problem. Although there is a specific sequence that can be used with PDF content, these do not generalize to arbitrary content.

To enable byte serving, a new ByteServe config.txt directive has been created. This directive takes a URL specified as a case-insensitive regular expression. Sample use for deslibris.ca is:

- ByteServe https://www.deslibris.ca/docs/.*

The first directive matches their ebook content delivery URLs. The second is a special entry that flags that specific URL for byte serving. It needs to be enabled as their JavaScript performs a test to see if it can retrieve that file using byte serving. If it cannot, then it will not use byte serving for the other content.

A URL that has been declared for byte serving will not be rewritten, even if the client and server do not actually use byte serving. It should be used with great care.

This directive may be repeated multiple times and is position independent within config.txt. For convenience, the best placement is within the stanza for the content provider (e.g., after the Title directive), even though there is nothing position dependent about it appearing within a stanza.

**Added ability to add HttpOnly to set cookies**

When a web server sends a Set-Cookie directive, it can include an optional flag HttpOnly to tell the browser that it should not reveal the value of the cookie to JavaScript. When this flag is enabled, the browser will still send the cookie as part of its requests. Using this option increases the security around the EZproxy cookie that identifies the current session.

Although the flag was called HttpOnly, the same behavior occurs whether the requests uses http or https.

This option is off (the "No" version) by default, retaining EZproxy’s previous behavior. It can be controlled by the config.txt directives:

- Option EZproxyCookieHttpOnly
- Option NoEZproxyCookieHttpOnly

There may be content providers using JavaScript to snoop for the presence of the EZproxy cookie to determine that EZproxy is active. Enabling this option will block this behavior which may cause access problems.

**Bug fixes**
Allow SAML attributes that contain spaces to be accessed from shibuser.txt

SAML Attributes can be accessed from shibuser.txt using the auth: namespace with expressions such as auth:uid. SAML Attributes can include spaces such as Member Of but spaces could not previously be included following auth:. The auth: namespace has been changed to allow the use of quotation marks around an attribute name, allowing these values to be accessed with syntax like auth:"Member Of".

BlockCountryChange

Option BlockCountryChange in 6.4 and possibly as far back as 6.1 had not been working. Proper behavior has been restored.

IncludeFile with non-existent or invalid filename caused EZproxy to crash

Prior to EZproxy 5.5, if config.txt contained an IncludeFile directive with an invalid name, this information was recorded in messages.txt before EZproxy stopped. As of EZproxy 5.5, the filename stopped being recorded in messages.txt but would display on the console.

Previous fixes were intended to restore the pre-5.5 behavior and worked during testing on Windows. However, on Linux, this change now causes EZproxy to crash at startup without revealing the invalid filename.

Correcting the included filename corrects the issue. If it is unclear what filename is causing the problem, starting up EZproxy using a copy of 6.2.2 will reveal the name of the file to the console. Once the error is fixed, 6.3.5 will start without any other issue related to this problem.

RemoteIPHeader and RemoteIPTrustedProxy reports error when in fact it is working properly

When the RemoteIPHeader or RemoteIPTrustedProxy directives appeared in config.txt, they were handled properly, but EZproxy displayed a warning that they were unrecognized. These erroneous errors have been removed.

EZproxy failed to rewrite a URL if it appeared as the very first thing in a webserver response

Previous versions of EZproxy with Javascript processing enabled failed to rewrite a URL if it was the first thing in a webserver response. This bug has been corrected.

Connection refused when remote host has 6 or more IPv6 addresses

The EZproxy document for “Option IPv6” states that unless this option is present, EZproxy will only use IPv4 in outgoing requests. The code as implemented in EZproxy did not match this behavior. As a result, sites that had IPv6 addresses associated with them were often tested first via IPv6 until those requests timed out, then favoring IPv4.

The internal behavior of EZproxy has been changed to match the documented behavior https://help.oclc.org/
Eliminated use by IntrusionAPI of separate intrusion.pem file

In EZproxy 6.5, the IntrusionAPI generated intrusion.pem from its internal IntrusionAPI configuration as the certificate authority used to verify the SSL certificate of the intrusion web service. If EZproxy was started as root using RunAs to change the user after startup, this interface created intrusion.pem as root and mode 0600 which prevented this file from being readable when needed.

EZproxy now reads the certificate authority from memory instead of depending on this external file, avoiding this conflict. Sites may safely delete any existing intrusion.pem file as it is no longer used.

Important links

Product website

More product information can be found at: http://www.oclc.org/ezproxy.en.html

Support website(s)

Support information for this product and related products can be found at:

- EZproxy support site
- EZproxy release notes and known issues
- Contact OCLC Support
- OCLC Community Center