### SLIDE 0 – WELCOME

Welcome. Thanks to Theda and Tasha at OhioLINK for inviting me to speak about EZproxy this morning. And, thanks to everyone for joining the session.

Today, I'm going to talk a bit about contextualizing EZproxy within a set of interconnected systems, with the goal of improved problem reporting and troubleshooting.

### SLIDE 1 – INTRO/BACKGROUND

Just a little background on me. I'm currently a Library Systems Analyst at OhioNET. From day-to-day, I cover a wide range of technology-focused tasks... but roughly 25% of my time is spent maintaining EZproxy.

For the last four years, I've been managing 60+ instances of EZproxy for a variety of institutional types: private and public academic libraries, medical libraries, high school libraries, and law libraries.

While each of these institutional types bringing its own set of troubleshooting challenges, the goal of this presentation not to take a deep dive into any specifics, but to provide you with an understanding of how EZproxy generally fits into the overall technical infrastructure of library systems.

### SLIDE 2 – SCHEMATIC AERIAL

Here's a schematic diagram to visually illustrate the placement of EZproxy within the larger context of interrelated system.

As you can see, EZproxy – which is a middleware application that provides IP-based access to electronic resources – is a single component in a much more complex and richer ecosystem.

As each of these components can potentially be a point of failure, we can quickly see that what may *appear* to be an EZproxy issue, may in fact, not be a problem specific to EZproxy at all.

The challenge of troubleshooting EZproxy issues is that we need to consider the range of potential failures that are possible at each stage in the remote access workflow, as an issue that at first manifests itself in the EZproxy workflow, might need to be resolved within another component within the workflow.

To do this, we'll take a quick walk through each of these steps, briefly pausing on each component to emphasize common issues.

### SLIDE 3 - USER ENVIRONMENT

The EZproxy workflow begins within what I'll call the user environment.

There are two fundamental *technical* components of the user environment to consider.

- The first is the: Network Context
- The second is the: Browser being used to making the request

### SLIDE 4 - NETWORK SEGMENT

Institutions using EZproxy rely on IP recognition to provide access to subscription content.

Each computer that is connected to a network – whether that's a local network or the greater internet – is given an IP address to identify itself on the network.

EZproxy provides access to the subscription resources by ensuring that the computer – Or more specifically, the IP address of the computer – that actually makes the request for subscription content is seen to be coming from an IP address that has been registered for access with the vendor.

As EZproxy is solely an IP-based authorization tool, for troubleshooting purposes, it's critical to have an understanding of the user's network context.

In the pre-COVID world, classifications of on-campus and off-campus were commonly used to talk about network context. However, even then, these two categories were not sufficiently robust to adequately map the terrain.

For example,

- <u>On-Campus</u>: (Physically)
  - Although most of us are not physically on-campus at the moment, from the perspective of EZproxy, on-campus merely means the range of IP addresses associated with the campus.
  - Considerations: Wifi network, extended IP range for the new computer lab
- <u>Off-Campus</u>: (Physically)
  - 5G/LTE – Mobile phones
  - VPN – Does the VPN have it's on IP address? Or, does it assign an on-campus IP to those connections?

In a pre-COVID world, it was totally possible to be physically on-campus, while virtually off-campus. So, you need to ask yourself, is EZproxy even involved?

EZproxy provides a built-in tool for identifying the IP address

### SLIDE 5 - BROWSER: RENDERING ENGINES

Aside from the network, another major component of the user environment is the web browser that's making the request. One of the web browser's main roles is to interpret the underlying code of the subscription platform into a visual display, or interface, but browsers are not created equal.

One major difference between browsers is the rendering engine. The rendering engine is a component of the browser application that's responsible for interpreting the underlying HTML, images, JavaScript, and CSS into an organized layout that the browser displays as the user interface.

Each rendering engine processes the underlying code of a vendor platform, in mainly consistent, but slightly idiosyncratic ways.

There are three main rendering engines:

- Blink
- Gecko
- Webkit

If a feature of a vendor platform is working in Firefox, but not Google Chrome, the rendering engine might be the culprit.

Special Note: Brave (Chromium-based)

- Shields protects your privacy as you browse by making you harder to track from site to site.
- Disable shields for library tools: can interfere with form submit and display of objects on the vendor platform UI

### SLIDE 6 - BROWSER: VERSION & PLUGINS

Less often, but potentially important, is knowing which version of a browser that’s being used. Bugs exist in all software, and it’s possible that a specific version of a browser has a bug.

Another common non-EZproxy reason, that often *appears* to be an EZproxy issue, is interference from browser plugins. I’ve seen issues more than a few times with overzealous privacy-focused plugins.

For example, the Electronic Frontier Foundations Privacy Badger is notorious for iframe rendering issues.

### SLIDE 7 - STARTING POINT URLS

Moving on to the next section in the workflow.

When the user makes a request for a subscription resource, they're generally clicking on a link to kick off the process. The link is what I, and the EZproxy documentation, will call a *starting point URL*.

Starting point URLs are links that, at minimum, define the requested resource, as well as specify an EZproxy prefix. The EZproxy prefix routes the request to a specific instance of EZproxy.

For simplicity, I’m going to lump starting point URLs into two major categories:

- Library controlled
- Unknown

### SLIDE 8 – LIBRARY CONTROLLED

Understandably, librarians focus on the elements that they can control, or on the properties that are considered within the library’s purview. As providing exemplar starting point URLs is not only essential to a proper EZproxy workflow, it’s foundational to work.

- A-Z Database Lists

- Administrative Dashboards (EBSCOhost, OCLC FirstSearch)
- Library Website (possibly)
- Learning Management Systems (perhaps)

proxy prefix and example

### SLIDE 9 - UNKNOWNS

On the flip side, there are plenty of locations that are outside of library control…. and you're definitely going to run into some head scratchers.

I've seen my fair share of oddball URLs, when I've asked a student for the URL that they are using to access a resource.

### SLIDE 10 – CONFIGURATION OF EZPROXY APPLICATION

As mentioned earlier, the EZproxy prefix in the starting point URL is what routes a request for subscription content through EZproxy. At this point, the users request gets handed off to EZproxy to complete the transaction.

In this section, I'll be a bit more about:

- EZproxy's Role
- Database stanzas (config.txt)
- Common issues
- Troubleshooting

### SLIDE 11 – WHAT DOES EZPROXY DO?

The question becomes, what does this linchpin application that sitting in the middle of my communication with subscription vendors actually do?

### SLIDE 12 – EZPROXY DOES A FEW THINGS

Recap

- Confirms users are authorized via some authentication process, which will be discussed in more detail during the Authentication section of this presentation.
- Makes the request for content on behalf of the user
  - The key is that the request comes from EZproxy's IP address (a static, consistently known IP address), not the user's actual IP address... this piece is crucial, and more detail will be provided in the Vendor section of this presentation.
- EZproxy retrieves the subscription content and returns it to the user's browser. Hence; EZproxy proxies (and I'm using my air quotes here) the request
- Log activity
  - For statistical, troubleshooting, license compliance, and security purposes

### SLIDE 13 – EXAMPLE OCLC STANZA

Every subscription resource – that authorized users are permitted to access remotely – needs to be configured in EZproxy.

The configuration is built upon the concept of database stanzas, each containing a set of directives, which tell EZproxy how to handle a user's request for subscription content. Database stanzas are maintained and created in a special file called config.txt.

Care must be taken to keep this file tidy and free of errors, as properly constructed database stanzas are essential to EZproxy fulfilling its role successfully.

Here's an example stanza for Oxford Scholarship Online. We'll step through each of the lines, called directives, in the stanza.

### SLIDE 14 – TITLE

The first directive of a database stanza is the Title. The main role of this directive is to indicate the start of a new database stanza.

Before the popularity boom of Springshare products, it was more common for libraries to use EZproxy's built-in database menu. As a secondary role, the title directive becomes the link text for the database in EZproxy's built-in A-Z list.

### Slide #15 - URL

The URL directive can be used only once per database. Since most folks no longer use the built-in EZproxy menu, the URL directive is essentially, a glorified HJ directive. Which we'll talk about next.

### SLIDE 16 - HJ

The HJ directive (Or, Host JavaScript) authorizes EZproxy to use a host as a starting point URL. Without a doubt, it's the most frequently used directive. It's the HJ lines that allow you to use the proxy prefix in front of a link.

If either an HJ or URL directive is not included in config.txt, EZproxy returns the all too common 'needhost' error.

599 Starting point URL referenced a host that EZproxy is not configured to proxy.

ex. {host}

### SLIDE 17 - DJ

The final directive that I'll mention is the DJ (Or, Domain JavaScript) directive. This directive tells EZproxy to proxy URLs under a certain domain. However, the DJ directive does not authorize a starting point. It tells EZproxy that – once a user has clicked on a valid starting point link – go ahead and proxy any other links from this specific domain.

### SLIDE 18 - Common Errors & Issues

### SLIDE 19 - needhost.html

Here is the needhost error messages that I mentioned earlier. Attempting to proxy a starting point URL that has not been configured in config.txt will produce this error.

### SLIDE 20 - Not defining HTTPS protocol on Starting Point URLs

EZproxy hosts are not protocol agnostic. If you want to allow both HTTP and HTTPS to be used in a starting point URL, you must specifically define both in the database stanza.

### SLIDE 21 - Passing the wrong account ID in Starting Point URLs

Another all too common occurrence is passing the wrong parameter values in a starting point URL.

With vendors that rely on account IDs being passed in the URL (e.g. Gale), you can run into issues where: (1) All of the IP addresses are registered in the admin dashboard, and (2) All of the entitled products are enabled on the institutions account, but access is still denied.

As the wrong account ID is being used.

### SLIDE 22 – Hand-off to OhioLINK