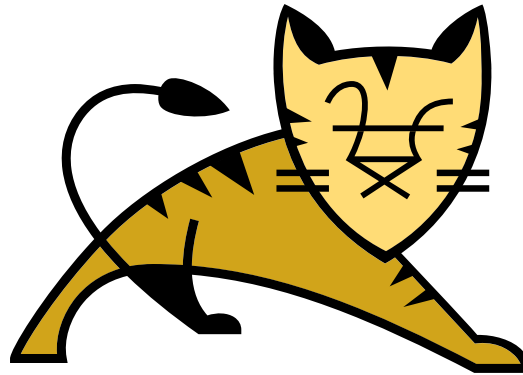


Migrating from AJP to HTTP

It's About Time



Christopher Schultz
Total Child Health, Inc.
ASF Member, Tomcat PMC, Security Team

QR for Slides:



Apache JServ Protocol: Overview

- Originally developed in 1997
- Often claimed to be “*the way*” to proxy Apache httpd to Tomcat
- Current version is 1.3 (ajp13), released ca. 2001
- Extension proposal exists; some elements have been implemented
- v1.4 proposal exists; essentially a dead proposal (and mostly a copy of the 1.3-extension proposal)

Apache JServ Protocol: Overview

- Packet-oriented binary protocol
- Encodes data with multiple tags
 - Proxy info (method/protocol/URI, remote IP/port, server IP/port, “Secure” flag)
 - HTTP Headers
 - Request attributes
 - Request/response body

Apache JServ Protocol: Overview

- Has HTTP header name optimization à la h2
 - Accept = 0xA001, Content-Length=0x008, etc.
- Has similar attribute name optimization
- Lots of proxy information is sent as request attributes
 - TLS info (protocol, cipher suite, key size, client cert, etc.)

Apache JServ Protocol: Overview

- Connections are expected to be long-lived

mod_jk

- A module for Apache https (and Microsoft IIS!)
- Provides proxying using AJP (!)
- Long history
- Very reliable
- Good runtime status and (re)configurability options
 - Change worker state, etc.

AJP Strengths

- When using default mod_jk configuration
 - All proxy info sent (method, protocol, client IP:port, etc.)
 - All TLS info sent (protocol, cipher suite, cert, etc.)
- HttpServletRequest is populated with the above
 - Just like a direct-request to Tomcat (without a proxy)

AJP Shortcomings

- AJP is a poor choice for non-HTTP/1.1
 - WebSocket
 - HTTP/2
- No security
 - No authentication (“secret” isn’t a secret)
 - No encryption

Aside: AJP (lack of) Security

- Connections are unencrypted
 - Cannot stress this enough

Aside: AJP (lack of) Security

- “Secret” isn’t a secret
 - It’s a cleartext string sent across an unencrypted connection
 - An attacker who can see the AJP traffic can read the secret
 - Therefore the secret provides (virtually) no protection

Aside: AJP (lack of) Security

- AJP connections are inherently trusted
 - No authentication
 - Client
 - Server
 - All status information is accepted without question
 - Like client IP:port
 - Like “secure” flag

Aside: AJP (lack of) Security

- AJP connections are inherently trusted
 - All *other* information is accepted without question
 - Like request attributes
 - Some request attributes have (*very*) special meaning (!)
 - CVE-2020-1938

Practical AJP

- Protect your endpoints
 - Only bind to `localhost`
 - `stunnel`
 - provides authenticated, encrypted connections
- Websocket, h2
 - Just have to live without it

Why Suffer?

- No advantages of AJP over other options
 - Except out-of-the-box configuration
 - ...and industry inertia

Aside: Do you need a proxy?

- Is a reverse proxy really necessary?
 - Bad reasons for a proxy
 - Performance! (“Tomcat is slow”)
 - Serving static content (see above)
 - Security! (“Tomcat is less-secure than \$proxy)

Goals

1. Eliminate the need for mod_jk to exist

- mod_jk is not bundled with Apache httpd
 - (e.g. win32/64 builds from Apache Lounge have a separate download; some Linux repos have a package available)
- mod_proxy_ajp exists, comes bundled

2. Eliminate the need for AJP to exist

- Use HTTP instead

Methodology

- Switch from mod_jk to mod_proxy_ajp
 - Requires extensive changes to httpd configuration
 - Requires minor changes to Tomcat configuration
 - Possibly no changes
 - Requires some improvements to mod_proxy and mod_proxy_balancer

Methodology

- Switch from `mod_proxy_ajp` to `mod_proxy_http`
 - Requires minor changes to `httpd` configuration
 - Requires more extensive changes to Tomcat configuration

Using mod_proxy

- Lots of changes to directives
 - JkMount → ProxyPass /ProxyPassReverse
 - Worker properties → Directives
 - Load balancer workers → use mod_proxy's balancer system
 - Might want to consider using ProxyErrorOverride

Using mod_proxy

mod_jk	mod_proxy
JkMount /app/*.jsp	ProxyPass /app/ ajp://host/app/ ProxyPassReverse /app/ ajp://host/app/
worker.host/worker.port/worker.type	(in ProxyPass URL)
worker.max_packet_size	ProxyIOBufferSize

Worker parameters are listed in the documentation for ProxyPass:
https://httpd.apache.org/docs/2.4/mod/mod_proxy.html#proxypass

Using mod_proxy

mod_jk	mod_proxy
<code>worker.sticky_session=true</code>	ProxyPass parameters: <code>stickysession=JSESSIONID jsessionid</code> <code>scolonpathdelim=On</code>
<code>worker.type=lb</code>	<code><Proxy "balancer://lb-name"></code> BalancerMember "ajp://host:port" <code></Proxy></code> ProxyPass /app/ balancer://lb-name/app/
<code>worker.route</code>	BalancerMember parameter: <code>route</code>

Worker parameters are listed in the documentation for ProxyPass:
https://httpd.apache.org/docs/2.4/mod/mod_proxy.html#proxypass

Using mod_proxy

- ProxyPass is both more and less flexible than JkMount
 - Always uses prefix-matches
 - ... unless you are using ProxyPassMatch
 - Sometimes require many more ProxyPass directives than JkMounts
- Same url-remapping caveats as with mod_jk
 - Bottom line: don't do it

Using mod_proxy

- jk-status worker is replaced by Balancer Manager
- Similar to jk-status console

Switching to mod_proxy_ajp

- Allows you to remove a component: mod_jk
- Retains most everything else
 - Protocol
 - Some configuration
- Evolution, not revolution

Switching to mod_proxy_ajp

- Apache httpd
 - LoadModule (mod_proxy, mod_proxy_ajp, mod_proxy_balancer, mod_lbmethod_byrequests)
 - Replace workers.properties + JkMounts with <Proxy>, ProxyPass, ProxyPassReverse, and various attributes
 - Using ajp:// as your proxy protocol

Switching to mod_proxy_ajp

- Tomcat
 - No changes (!)
 - Tomcat is still using AJP, httpd is simply using mod_proxy_ajp module instead of mod_jk

Switching to mod_proxy_ajp

- Other considerations
 - Monitoring
 - Scripting
 - Worker state changes (e.g. ACT → DIS → STO)
 - Detecting degraded workers
 - To encourage faster draining of Tomcat nodes
 - https://bz.apache.org/bugzilla/show_bug.cgi?id=64338

Switching to mod_proxy_http

- New security options
 - TLS
 - Granular acceptance of proxy-info (e.g. X-Forwarded-*, etc.)
- Better protocol support
 - HTTP/1.x (plaintext easier to debug)
 - WebSocket (okay, mod_proxy_wstunnel)
 - h2 (not yet: support added in httpd 2.5+)

Switching to mod_proxy_http

- Apache httpd
 - Change ajp:// to http(s)://

Switching to mod_proxy_http

- Tomcat
 - Enable HTTP <Connector>
 - Pick-up proxy info
 - RemoteIPValve
 - Provides request info such as isSecure, getProtocol, getRemoteHost, etc. (affects access logs)
 - SSLValve
 - Provides TLS handshake info such as TLS session id, chosen cipher suite, client certificate, etc.

Switching to mod_proxy_http

- Other considerations
 - Monitoring (Tomcat <Connector>s)
 - Mutually-authenticated TLS (httpd presents client-cert to Tomcat)
 - httpd: SSLProxyMachineCertificateFile
 - Tomcat: <SSLHostConfig caCertificateFile / truststoreFile

Using mod_proxy

- Offers some interesting new capabilities
 - Heterogeneous protocols in balancers (!)

```
<Proxy "balancer://lb-name">
```

```
  BalancerMember ajp://host1:port
```

```
  BalancerMember ajp://host2:port
```

```
  BalancerMember http://host3:port
```

```
  BalancerMember http://host4:port
```

```
</Proxy>
```


Using mod_proxy

- Offers some interesting new capabilities

- Fallback to static content \O/

```
<Proxy "balancer://lb-name">
```

```
    BalancerMember http://host1:port
```

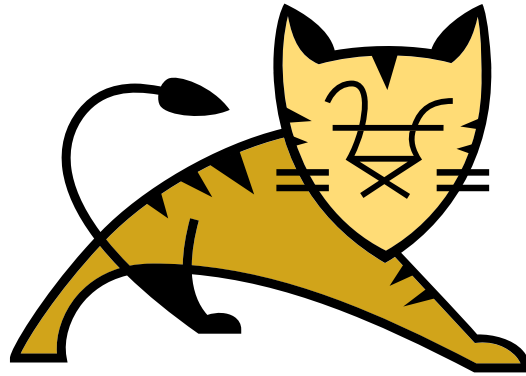
```
    BalancerMember http://host2:port
```

```
    BalancerMember http://127.0.0.1/down-pages/ status=+H
```

```
</Proxy>
```

Migrating from AJP to HTTP

- Migrate from mod_jk → mod_proxy_ajp
- Migrate from mod_proxy_ajp → mod_proxy_http
- Reduce complexity of deployment
 - Remove mod_jk (a “third-party” module)
- Improve security of proxy connections
 - Authenticated TLS



Questions



[https://people.apache.org/~schultz/presentations/ApacheCon NA 2020/Migrating from AJP to HTTP.pdf](https://people.apache.org/~schultz/presentations/ApacheCon%20NA%202020/Migrating%20from%20AJP%20to%20HTTP.pdf)
Sample code available in the same directory.