The Proxy Server

1 Purpose

The proxy server acts as an intermediate server that relays requests between a client and a server. The proxy server keeps track of all the client-server interactions, which allows you to monitor exactly what is going on, without having to access the main server.

You can find the proxy server in the FileNet WSM installation folder, in the folder

```
/opt/helpers/proxy.jar
```

You can use the proxy server to monitor all client-server interaction, regardless of the communication protocol. For example, you can monitor the following protocols:

- HTTP for Web pages
- HTTPS for secure Web pages
- SMTP for email messages
- LDAP for user management

You can also use the proxy server as a simple port forwarding proxy if you need to test a WSM instance using a different port number than your standard port.

2 Usage Examples

Cookies

The following log entry example shows all cookies and their values send by the client on the 6th connection opened since the proxy server started:

C-6-#000635 -> [Cookie: cq3session=7e39bc51-ac72-3f48-88a9-ed80dbac0693; Show=ShowMode; JSESSIONID=68d78874-cabf-9444-84a4-538d43f5064d]

Headers

The following log entry example shows that the server is able to make a keep-alive connection and the content length header was properly set:

```
S-7-#000017 -> [Connection: Keep-Alive ]
...
S-7-#000107 -> [Content-Length: 124 ]
```

Checking if Keep-Alive Works

Keep-alive means that a client re-uses the connection to the server to transports multiple files (the page code, pictures, style sheets and so on). Without keep-alive, the client has to establish a new connection for each request.

To check if keep-alive works, start the proxy server and request a page. If keep-alive is working, the connection counter should never go above 5 to 10 connections. If keep-alive is not working, the connection counter increases rapidly.

Finding Lost Requests

If you lose requests in a complex server setting, for example with a firewall and a dispatcher, you can use the proxy server to find out where the request was lost. In case of a firewall, start one proxy server before the firewall and one after it, and see if all the requests come through.

Hanging Requests

If you experience hanging requests from time to time, start a proxy server and wait, or write the access log into a file with a timestamp. When a request hangs, you can see how many connections where open and which request is causing trouble.

3 Starting the Proxy Server

You can find the proxy server in the FileNet WSM installation folder, in the folder

```
/opt/helpers/proxy.jar
```

To start the proxy server, type the following text on the command line:

```
java -jar proxy.jar <host> <remoteport> <localport>
[options]
```

The following options are available:

-q (Quiet Mode)

Does not write the requests to the console window. Use this if you do not want to slow down the connection, or if you log the output to a file (see -logfile option).

-b (Binary Mode)

If you are looking for specific byte combinations in the traffic, enable binary mode. The output will then contain the hexadecimal and character output.

-t (Time stamp log entries)

Adds a time stamp to each log output. The time stamp is in seconds, so it may not be suitable for checking single requests. Use it to locate events that occurred at a specific time if you use the proxy server over a longer time period.

-logfile <filename> (Write to log file)

Writes the client-server conversation to a log file. This parameter works also in quiet mode.

-i <numIndentions> (add indention)

Each active connection is indented for better readability. Default is 16 levels. This feature is new in prox.jar version 1.16.

4 Reading the Log

The log entries produced by the proxy.jar have the following format:

```
[timestamp (optional)] [Client/Server]-
[ConnectionNumber]-[BytePosition] ->[Character Stream]
```

For example, a request for a Web page may look as follows:

```
C-0-#000000 -> [GET
/author/prox.html?CFC_cK=1102938422341 HTTP/1.1 ]
```

- `C` signifies that this entry comes from the client (it is a request for a Web page)
- `0` is the connection number (the connection counter starts at 0)
- `#00000` the offset in the byte stream. This is the first entry, so the offset is 0.
- `[GET ?]` is the content of the request, in the example one of the HTTP headers (`url`).

When a connection closes, the following information is logged:

```
C-6-Finished: 758 bytes (1.0 kb/s)
S-6-Finished: 665 bytes (1.0 kb/s)
```

This shows the number of bytes that passed between client and server on the 6th connection and at the average speed.

Example

A template produces the following code when requested:

```
<html>
  <head>
    <title>Welcome</title>
  </head>
  <body>
    Welcome to Playground<br>
    <img src="/logo.gif">
  </body>
</html>
```

WSM is running on localhost:4303, so start the proxy server as following:

```
java -jar proxy.jar localhost 4303 4444 -logfile test.log
```

You can access the server (localhost:4303) without the proxy server, but if you access it via localhost:4444, the proxy server logs the communication.

Open a browser and access a page created with the above template. After that, look at the log file.

Note: Until proxy.jar version 1.14, the log entries of one connection are not synchronized, meaning the log entries of one client/server connection are not necessary in the correct sequential order. The newer versions of the proxy server do not have this issue.

When starting, the following start-up info is written to the log:

```
starting proxy for localhost:4303 on port 4444
using logfile: C:\CQUnify355default\opt\helpers\test.log
```

Start of the first connection (0) requesting the main HTML page, the following header fields are listed:

```
C-0-#000000 -> [GET
/author/prox.html?CFC_cK=1102936796533 HTTP/1.1 ]
C-0-#000053 -> [Accept: image/gif, image/x-xbitmap,
image/jpeg, image/pjpeg, application/vnd.ms-powerpoint,
application/vnd.ms-excel, application/msword, appl]
C-0-#000194 -> [ication/x-shockwave-flash, */* ]
C-0-#000227 -> [Accept-Language: de-ch ]
C-0-#000251 -> [Accept-Encoding: gzip, deflate ]
C-0-#000283 -> [User-Agent: Mozilla/4.0 (compatible; MSIE
6.0; Windows NT 5.0) ]
C-0-#000347 -> [Host: localhost:4444 ]
```

The Client requests a keep-alive connection, so the server can send multiple files over the same connection:

```
C-0-#000369 -> [Connection: Keep-Alive ]
```

The proxy server is a good tool to verify if cookies are properly set or not. Here, we see the cq3session cookie generated by WSM, the show mode switch cookie generated by the CFC, and a cookie named JSESSIONID. This one is automatically created by JSP if not explicitly switched off using <%@ page session="false" %>:

```
C-0-#000393 -> [Cookie: Show=ShowMode;
cq3session=3bce15cf-1575-1b4e-8ea6-0d1a0c64738e;
JSESSIONID=4161a56b-f193-d748-88a5-e09c5ff7ef2a ]
C-0-#000514 -> [ ]
S-0-#000000 -> [HTTP/1.0 200 OK ]
```

The server will close connection 0 after the request. Keep-alive is not possible, because the request has a question mark. This means

that the server can not return a cached version, and therefore cannot determine the content length at this point, which is required for a keep-alive connection.

```
S-0-#000017 -> [Connection: Close ]
S-0-#000036 -> [Server: Communique Servlet Engine/3.5.5 ]
S-0-#000077 -> [Content-Type: text/html;charset=iso-8859-
1 ]
S-0-#000121 -> [Date: Tue, 14 Dec 2004 09:46:44 GMT ]
S-0-#000158 -> [Set-Cookie: JSESSIONID=4161a56b-f193-d8-
88a5-e09c5ff7ef2a;Path=/author ]
S-0-#000232 -> [ ]
```

Here, the server starts sending the HTML code on connection 0:

```
S-0-#000234 -> [<html> ]
S-0-#000242 -> [.<head> ]
S-0-#000251 -> [..<title>Welcome</title> ]
S-0-#000277 -> [.</head> ]
S-0-#000287 -> [.<body> ]
S-0-#000296 -> [..Welcome to Playground<br> ]
S-0-#000325 -> [..<img src="/author/logo.gif"> ]
S-0-#000357 -> [.</body> ]
S-0-#000367 -> [</html>]
```

Connection 0 closes immediately after the HTML file has been served:

```
C-0-Finished: 516 bytes (0.0 kb/s)
S-0-Finished: 374 bytes (0.0 kb/s)
```

Now, the output starts for connection 1, which downloads the image contained in the HTML code:

```
C-1-#000000 -> [GET /author/logo.gif HTTP/1.1 ]
C-1-#000031 -> [Accept: */* ]
C-1-#000044 -> [Referer:
http://localhost:4444/author/prox.html?CFC_cK=11029367965
33 ]
C-1-#000114 -> [Accept-Language: de-ch ]
C-1-#000138 -> [Accept-Encoding: gzip, deflate ]
C-1-#000170 -> [User-Agent: Mozilla/4.0 (compatible; MSIE
6.0; Windows NT 5.0) ]
C-1-#000234 -> [Host: localhost:4444 ]
```

Again, the client requests a keep-alive connection:

```
C-1-#000256 -> [Connection: Keep-Alive ]
C-1-#000280 -> [Cookie: Show=ShowMode;
cq3session=3bce15cf-1575-1b4e-8ea6-0d1a0c64738e;
JSESSIONID=4161a56b-f193-d748-88a5-e09c5ff7ef2a ]
C-1-#000401 -> [ ]
S-1-#000000 -> [HTTP/1.0 200 OK ]
```

For connection 1, the server agrees on keep-alive, because the image is static, and therefore the content length is known.

```
S-1-#000017 -> [Connection: Keep-Alive ]
S-1-#000041 -> [Server: Communique Servlet Engine/3.5.5 ]
S-1-#000082 -> [Content-Type: image/gif ]
```

The server returns the content length of the image on connection 1:

```
S-1-#000107 -> [Content-Length: 124 ]
S-1-#000128 -> [Date: Tue, 14 Dec 2004 09:46:44 GMT ]
S-1-#000165 -> [ ]
```

Now that the content length is established, the server sends the image data on connection 1:

```
S-1-#000167 ->
[GIF87a......... .........   ... .....,.......
...I....0.A..8......YDA.W...1..`i.`..6...Z...$@.F..)`..f.
.A.....iu.........$..;]
```

After the keep-alive timeout has been reached, connection 1 closes as well:

```
S-1-Finished: 291 bytes (0.0 kb/s)
C-1-Finished: 403 bytes (0.0 kb/s)
```

The above example is comparatively simple, because the two connections occur one after an other (first the server returns the HTML code, then the browser requests the image and opens a new connection).

In practice, a page may generate many parallel requests for images, style sheets, JavaScript files and so on. This means that the logs have overlapping entries of parallel open connections. In that case, we recommended to use option -i to improve readability.