Runtime Infrastructure - Feature #5776

reduce memory requirements for the FWD client

10/26/2021 10:22 AM - Greg Shah

Status:	Test	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Tomasz Domin	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**version:**	
vendor_id:	GCD		

Description

Related issues:

Related to User Interface - Feature #5742: improve web client startup time	**Test**
Related to User Interface - Bug #2766: implement a text metrics server process	**New**
Related to Base Language - Feature #4065: server-side processing of client pl...	**WIP**

History

#1 - 10/26/2021 10:23 AM - Greg Shah

The GUI client uses a huge amount of memory. We have had to set it around 1GB for some installations, which is a ridiculous amount.

Constantin notes the following:

> This is somehow unrelated, is about reducing the memory footprint of the FWD client: with every memptr and I think for COPY-LOB, too, we transfer the entire LOB to the client. In some cases, this is in ~100MB range, for a large customer application. So we bring the entire memptr into the client JVM as byte[], and transfer it to the FWD server, when needed. For BLOB and other cases of LOB-COPY usage, is the same, we transfer the entire LOB to the FWD client. We should do this in chunks (10MB?), to reduce the footprint of in-memory byte[] instances at the FWD client.
>
> Beside this, the other large consumer of memory are the text and font metrics, which are transferred to the client JVM completely. There was a task at some point to use a metrics server running on Windows, for GUI clients.
>
> If we solve these two, I think we can bring the memory footprint for the FWD client to 128MB or less. The UI tree shouldn't be that much of a consumer, so even 64MB should be enough.

#2 - 10/26/2021 10:34 AM - Greg Shah

> with every memptr and I think for COPY-LOB, too, we transfer the entire LOB to the client

The memptr data only persists on the client. This is required to properly work with native APIs, so it is by design. A memptr is literally a hardware address of virtual memory for the client process. However, I don't understand exactly what you are saying here about transferring the entire LOB to the client. I guess this is how we would do it if some server-side data was being copied in bulk to a memptr. In such a case, we have no option but to push the data to the client. But normal memptr operations aren't bulk-copying of data. It is more like "write a 64-bit integer into this part of the memptr buffer" which will result in small amounts of data being pushed to the client, but not megabytes.

As are as COPY-LOB, copying a LOB to the cliend would only be needed for a copy to a memptr or a copy to a file. Is that what you are talking about?

> For BLOB and other cases of LOB-COPY usage, is the same, we transfer the entire LOB to the FWD client.

BLOB is server-side data only. In the 4GL it is managed by the database. It never needs to go to the client unless it is being copied into a memptr or a user-controlled file. Are you saying we implement BLOB as a client-side stream? That is not good and needs to be resolved ASAP.

> Beside this, the other large consumer of memory are the text and font metrics, which are transferred to the client JVM completely. There was a task at some point to use a metrics server running on Windows, for GUI clients.

I think you are referring to #2766 (text metrics server) which was needed for #2826 (dynamic font sizing).

I was **not** wanting the text metrics server to be a requirement for all installations. At best, it is a fallback plan for any dynamic font sizing issues which we cannot calculate properly within FWD. The huge downside to having that external server is that it has to be maintained and managed over time. It complicates the runtime installations. It is an unwanted mess to be avoided when possible.

I would rather have the captured text metrics on the server and send only those strings that are necessary down to the client. So instead of pushing all metrics there, we handle them lazily. Part of that will be to send the metrics down with any string data that is sent (as a single call down to the client). Perhaps a good way to do that is to store these metrics inside the widget/frame configs (for labels and string literals). That means it could be part of the screen definition at conversion time. The other part of this will be the client up-calling when needed to check for strings that are client-side generated (user edits).

#3 - 10/26/2021 10:45 AM - Greg Shah

The BLOB and CLOB support was added in #3500. That task has no useful details about the actual implementation, just some early discussion of the requirements and some ideas. Can someone please explain the current implementation?

#4 - 10/26/2021 10:54 AM - Constantin Asofiei

Greg Shah wrote:

> The memptr data only persists on the client. This is required to properly work with native APIs, so it is by design. A memptr is literally a hardware address of virtual memory for the client process. However, I don't understand exactly what you are saying here about transferring the entire LOB to the client. I guess this is how we would do it if some server-side data was being copied in bulk to a memptr. In such a case, we have no option but to push the data to the client.

I mean when we need to do something with all the memptr data, there are cases when we just read the entire memptr in a byte[] and transfer it to the server - this requires to read the entire memptr into the client JVM. Another example, memptr.writeByteRange will transfer the entire byte[] in a single call, and it forces the client JVM to hold it, before being written to the native memory. The same paradigm applies to other cases, like COPY-LOB: TargetLobFile will do a Stream.write(byte[]) for the entire source, no matter if the source is 100s of MB.

> But normal memptr operations aren't bulk-copying of data. It is more like "write a 64-bit integer into this part of the memptr buffer" which will result in small amounts of data being pushed to the client, but not megabytes.

Yes, but the 'edge cases' will force the client JVM to hold the entire byte[], which can be huge - even if there is a single operation in the entire application which requires this memptr memory usage.

> BLOB is server-side data only. In the 4GL it is managed by the database. It never needs to go to the client unless it is being copied into a memptr or a user-controlled file. Are you saying we implement BLOB as a client-side stream? That is not good and needs to be resolved ASAP.

No, I didn't mean that we keep BLOBs on the client. I meant when you need to do something on the client-side with the BLOB (like COPY-LOB, to write it to a file or memptr), it gets transferred to the client via a single API call with a byte[] parameter.

In other words: there are cases in FWD where we use a single byte[] API call to transfer data between the server and the client (and viceversa) - and this forces the client to hold the entire byte[] in the JVM heap. But the source of the byte[] data can be arbitrarily large, and this forces the FWD client to have enough JVM heap to accommodate the largest data which the application needs to process.

In my view, the goal of this task is to identify all APIs which work with byte[] in FWD to transfer data over the network, and see if the code using this API can be refactored to transfer the data in chunks, to not force the client to hold this full data in the JVM heap, just to end up in a memptr or in a file.

> I would rather have the captured text metrics on the server and send only those strings that are necessary down to the client. So instead of pushing all metrics there, we handle them lazily. Part of that will be to send the metrics down with any string data that is sent (as a single call down to the client). Perhaps a good way to do that is to store these metrics inside the widget/frame configs (for labels and string literals). That means it could be part of the screen definition at conversion time. The other part of this will be the client up-calling when needed to check for strings that are client-side generated (user edits).

Yes, I think this is the best approach. We can also use an expiring cache to ensure we don't OOME the client.

#5 - 10/26/2021 11:25 AM - Greg Shah

> In my view, the goal of this task is to identify all APIs which work with byte[] in FWD to transfer data over the network, and see if the code using this API can be refactored to transfer the data in chunks, to not force the client to hold this full data in the JVM heap, just to end up in a memptr or in a file.

OK, good. I think we are on the same page. We should write some common utility code to handle the "chunking" transfer of large byte[] data to and from the client.

#6 - 10/26/2021 11:33 AM - Constantin Asofiei

Another idea: for COPY-LOB when both the source and the target are on the client (like file to memptr), we should do the full operation on the client... to avoid bringing the data in memory on the server, just to end up back on the client.

#7 - 10/26/2021 11:56 AM - Greg Shah

Yes, memptr/file to memptr/file (m to m, m to f, f to m, f to f) should be an all client operation in a single downcall.

#10 - 12/09/2021 12:21 PM - Greg Shah

- Assignee set to Constantin Asofiei

#11 - 05/03/2022 07:02 AM - Greg Shah

- Related to Feature #5742: improve web client startup time added

#12 - 05/03/2022 07:02 AM - Greg Shah

- Assignee changed from Constantin Asofiei to Tomasz Domin

#13 - 05/03/2022 07:03 AM - Greg Shah

> The BLOB and CLOB support was added in #3500. That task has no useful details about the actual implementation, just some early discussion of the requirements and some ideas. Can someone please explain the current implementation?

Eric: We still need your response on this.

#14 - 05/19/2022 12:58 PM - Tomasz Domin

- Status changed from New to WIP

Currently I work mainly with hotel app, I need to setup some other application (e.g. one of customer's apps),
Hotel app probably has different memory requirements and does not allocate much memory - based on basic java console memory usage observation - 80MB Nonheap + 150MB heap.

#15 - 05/20/2022 10:17 AM - Tomasz Domin

I've setup customer application and started conversion.
I guess I need to leave it over weekend based on estimated conversion time :)

#16 - 05/20/2022 10:28 AM - Eugenie Lyzenko

Tomasz Domin wrote:

> I've setup customer application and started conversion.
> I guess I need to leave it over weekend based on estimated conversion time :)

Tell me your CPU spec and I will estimate how much time you have for coffee :)

#17 - 05/20/2022 10:34 AM - Roger Borrello

Remember you can work with pre-build binaries, instead of converting yourself. When I run a conversion of this customer app it can be 24 hours.
Look over the wiki for "Using-pre-built-Binaries" and reach out to me with questions.

#18 - 05/20/2022 11:11 AM - Tomasz Domin

Roger Borrello wrote:

> Remember you can work with pre-build binaries, instead of converting yourself. When I run a conversion of this customer app it can be 24 hours.
> Look over the wiki for "Using-pre-built-Binaries" and reach out to me with questions.

Thank you.
I'll try both ways. As conversion stared lets see how it goes, and in parallel will setup project from pre-built-Binaries.

#19 - 05/23/2022 11:12 AM - Tomasz Domin

I've converted customer app over weekend.
Deployed and started swing client - app works.

Two remarks:

- small change to build.xml to use a proper heap for aspectJ:

```
   <iajc destDir="${build.home}/classes.aop"
            source="1.7"
            target="1.7"
```

```
            verbose="false"
+            fork="true"
+            maxmem="${compileMemory}"
            showweaveinfo="false">
```

- needed to add db.java.udfs=true to make udf-s install during ant db.create

Roger -

- will customer app work in embedded mode ?
- I have issues running virtual desktop mode - after logging in I get HTTP 404 on getting /index.html?token=xxxx

Do you have any hints on above ?
I've already checked wiki pages (very good source of info).

#20 - 05/23/2022 11:32 AM - Roger Borrello

Tomasz Domin wrote:

> Two remarks:
>
> - small change to build.xml to use a proper heap for aspectJ:
>   [...]
> - needed to add db.java.udfs=true to make udf-s install during ant db.create

If that first change is for any configuration, feel free to check it into bzr. The UDF one... does that allow you to no longer install PL/Java, because I have not made attempts to run in that configuration. I would really like to switch the customer over to that configuration, so when we get yours running 100%, I'd like to get familiar with all changes necessary.

> Roger -
>
> - will app work in embedded mode ?

I have not tried embedded mode. So another place of new learning.

- I have issues running virtual desktop mode - after logging in I get HTTP 404 on getting /index.html?token=xxxx

Do you have "localhost" in the webClient "host"? Are you running on a single server, or going across a network?

#21 - 05/23/2022 11:40 AM - Tomasz Domin

Roger Borrello wrote:

- I have issues running virtual desktop mode - after logging in I get HTTP 404 on getting /index.html?token=xxxx

Do you have "localhost" in the webClient "host"? Are you running on a single server, or going across a network?

Thank you. Changing IP address "10.x.x.x" to "localhost" helped.

#22 - 05/23/2022 12:07 PM - Greg Shah

Embedded mode is not yet functional. See #4205.

#23 - 05/23/2022 03:24 PM - Tomasz Domin

Initial analysis proofs that almost 93% of heap is taken by Map<String, int[]> textMetrics that stores textMetrics retrieved from the server.

```
Class Name                                                                | Shallow Heap | Retained Heap | Percenta
ge
--------------------------------------------------------------------------------------------------------------------
---
class com.goldencode.p2j.security.ContextLocal @ 0xc0145560               |           32 |   452,167,208 |     93.8
9%
|- java.util.HashMap @ 0xc01e7db8                                         |           48 |   452,165,888 |     93.8
9%
|  |- java.util.HashMap$Node[64] @ 0xd951b360                           |          272 |   452,165,840 |     93.8
9%
|  '- Total: 1 entry                                                     |              |               |
|- java.lang.Object[18] @ 0xc00a7a38                                      |           88 |         1,240 |      0.0
0%
|- java.util.TreeMap @ 0xc020df88                                         |           48 |            48 |      0.0
0%
'- Total: 3 entries                                                       |              |               |
com.goldencode.p2j.ui.client.gui.driver.web.GuiWebDriver @ 0xc003e550|          200 |     6,639,624 |      1.3
8%
Total: 2 entries                                                          |              |               |
--------------------------------------------------------------------------------------------------------------------
---
```

```
      private final Map<String, int[]> textMetrics = new HashMap<String, int[]>();
// on start
            Map<String, int[]> legacyTextMetrics = server.readLegacyTextMetrics();
            textMetrics.putAll(legacyTextMetrics);
```

As legacyTextMetrics stores all text with all possible fonts most of its data will not be used.
The goal is to replace legacyTextMetrics with a cache that fills up with data on demand using subsequent @int[] getLegacyTextMetrics(String text, String key)@

#24 - 05/23/2022 03:28 PM - Roger Borrello

Tomasz Domin wrote:

> Initial analysis proofs that almost 93% of heap is taken by Map<String, int[]> textMetrics that stores textMetrics retrieved from the server.

Would this vary by how big text-metrics.xml is? This customer's version is **237,046,142**.

#25 - 05/23/2022 04:02 PM - Tomasz Domin

Roger Borrello wrote:

> Tomasz Domin wrote:
>
> > Initial analysis proofs that almost 93% of heap is taken by Map<String, int[]> textMetrics that stores textMetrics retrieved from the server.
>
> Would this vary by how big text-metrics.xml is? This customer's version is **237,046,142**.

Yes, this is exact in-memory representation of text-metrics.xml.

#26 - 05/24/2022 04:00 AM - Tomasz Domin

I've removed retrieval of full text metrics on FontManager initialization and made textMetrics a LRUCache with a fixed size of 1000 elements.
Memory usage went down as expected - total heap size is about 60MB now.

Spent some time on debugging and statistics analysis on getTextMetrics method:

```
                      all calls/cache miss(not found in cache)/server miss (not found on server side)
after app startup:    383      /17                            /9
after screen switch:  411      /35                            /27
after product search: 17648    /475                           /237
```

It seems cache hit ratio is high (>95%) so it may be enough to make use of the approach.
Still - there could be applications that would need retrieving full text metrics at client startup or tuning cache size.

I think text metrics handling should be application specific - thus there should be configuration options to change behavior to choose from:

- get all text metrics on client startup (boolean)
- specify text metrics local cache size (int)
- enable further text metrics retrieval from server - kind of optimization to use only local text size calculation (boolean)

#27 - 05/24/2022 06:43 AM - Greg Shah

Agreed. I like the plan.

> enable further text metrics retrieval from server - kind of optimization to use only local text size calculation (boolean)

Please see #2766. Is this the same idea?

#28 - 05/24/2022 06:43 AM - Greg Shah

- Related to Bug #2766: implement a text metrics server process added

#29 - 05/24/2022 10:39 AM - Tomasz Domin

Greg Shah wrote:

> Agreed. I like the plan.
>
> > enable further text metrics retrieval from server - kind of optimization to use only local text size calculation (boolean)
>
> Please see #2766. Is this the same idea?

I rather meant - disable querying server for missing metrics, as client always queries server in case text-metrics is missing even if complete text-metrics file has been download on start. And only if server query returns nothing it calculates metrics locally. Anyway I've not implemented that issue, as I've found it may be not needed.

In meantime I found another way to minimize text-metrics server to client transfers to speed up client startup and reduce memory usage and not to reduce client performance too much.

I've implemented text-metrics cache on both sides - client and server.

- Server text-metrics cache is a set of items that are the most frequently requested by clients - for now I am using LRU cache, it may be switched to MFU Cache in optimum case (I didnt check code for existing implementation)
- Client text-metrics cache is a set of local items and it is initialized with server side Server text-metrics cache on startup, and later its filled during runtime with additional text-metrics entries as needed by application - retrieved from server individually or calculated locally if not available on a server.

So when the first (after server restart) client connects it has to retrieve all text metrics one-by-one from server, as server cache thus client cache is empty, so it may perform slow.
But the second client should get from a server subset of text metrics based on actual client usage.

Statistics looks promising:

```
                all calls/cache miss(not found in cache)/server miss (not found on server side)
First client:   390      /17                             /9
Second client:  400      /0                              /0
```

Further optimizations:

- switch Server text-metrics cache to MFU cache so the most frequently needed entries are cached with higher prority
- serialize Server text-metrics cache during runtime to disk/database, so evenfirst client gets a proper cache data on startup

#30 - 05/24/2022 10:51 AM - Tomasz Domin

I have two questions - maybe someone has quick answers

- what is preferred way for passing configuration from server to client ? Do I need to expand ServerExports interface ? Currently I am passing cache size to client a bit under-the-hood and I dont like it ;)
- what is threading model for server/client in regards to access shared data e.g. FontTable on server and FontManager on Client - I presumed both client and server are multithreaded, so I synchronize on LRUCache access, but it may be not true for client.

#31 - 05/24/2022 11:26 AM - Tomasz Domin

Some testing in meantime:
Original application first startup time: 25s memory usage: 456MB
Original application second startup time: 17s memory usage: 456MB

Text metrics cached application first startup time: 16,8s memory usage: 62MB
Text metrics cached application second startup time: 7s-8s memory usage: 62MB

The tests were performed on local machine, I guess due to transfer size the difference will be even bigger for distributed setup.
The question that left is if Text metrics cached application will perform well if distributed.

#32 - 05/25/2022 07:32 AM - Tomasz Domin

- % Done changed from 0 to 40

As it was more relevant the text metrics optimizations are moved to #5742 and prepared for review #5742#note-12
Nevertheless I will continue to work on other possible memory reduction in this issue.

#33 - 05/25/2022 08:04 AM - Greg Shah

> - what is preferred way for passing configuration from server to client ? Do I need to expand ServerExports interface ? Currently I am passing cache size to client a bit under-the-hood and I dont like it ;)

To "one time" push some state from the client to the server at startup, we have void MainEntry.setClientParams(ClientParameters) which is called from ClientCore.start(). State can be added to the ClientParameters class.

We really need something that handles the reverse path. There is quite a bit of state that we manage on a class by class basis, which should just be queried at the start of a session. We could change void MainEntry.setClientParams(ClientParameters) into ServerParameters MainEntry.configure(ClientParameters) and use this same round trip to pass pack server-side configuration. I would think that the most common stuff configured on the server side (e.g. propath, numeric/date formats ...) would be good to include in the ServerParameters. It would be a nice improvement.

> - what is threading model for server/client in regards to access shared data e.g. FontTable on server and FontManager on Client - I presumed both client and server are multithreaded, so I synchronize on LRUCache access, but it may be not true for client.

Both are multithreaded, however the majority of processing occurs on a single active thread on either one side of the "conversation" or the other side. The control flow moves across the network and things on the side without control flow are largely suspended. There are exceptional cases where we have asynchronous processing such as the KeyReader in the TypeAhead class on the client. The protocol handling threads (each side has its own reader/writer) are asynchronous as well. It is fairly complicated. There can be quite a bit of recursion. The server can call the client, which can call the server, which can call the client... all before any of the calls return, unwinding the stack.

Making the data structures safe for multithreaded access is the right thing to do. We will have to see the implementation of the locking to know if there are any kinds of deadlocks that can occur.

#34 - 05/25/2022 09:39 AM - Tomasz Domin

During debugging I've found funny thing.
I've noticed that client application allocates a lot of heap memory just after its started and then memory usage is greatly reduced (to about 32-33 MB).
I've played a bit with -Xmx option and catching OOMs, so I found an issue in bmp icon file decoding - imgop-bmp library, which is used for bmp file decoding, allocates memory for palette. Unfortunately it also does so in case bitmap is 24-bits, so it allocates new int[16 777 217] which makes unnecessary allocation for 64MB.
Memory is being released image processing, but this makes setting up heap size larger than needed by 64MB.

Obviously its a bug, I will check if newer version has this issue corrected.
The other option would be converting icon file to 256 colors at most.

#35 - 05/25/2022 09:53 AM - Constantin Asofiei

Please check what constraints are for ChUI clients; for example, is this imgop-bmp library loaded for ChUI? I hope not.

The point is: we have Appserver agents which (currently) require a standalone FWD client, and if the memory for this can be lowered bellow 32MB, that would be great.

OTOH, there is another limitation: transferring large amounts of data over the network - I don't recall exactly how many cases there are, but at least with current memptr support (as this is allocated on the client), loading a file into memptr is done completely on the client, but after that any work with this memptr (like saving it into a longchar) will load it completely in the client JVM memory, to be able to transfer it to the server-side in one go; so if you have a 100MB file, then you need enough heap to store this on the client-side.

#36 - 05/25/2022 10:15 AM - Greg Shah

The memptr allocates memory outside of the Java heap. It uses the CRT (C language runtime) memory allocation functions. This memory does not need to be planned for and cannot really be planned for since it is dynamic (runtime based, driven by converted 4GL code) and application specific.

#37 - 05/25/2022 10:15 AM - Greg Shah

Sergey/Eugenie: Do you have any thoughts on the icon color depth? What compatibility requirements do we have?

#38 - 05/25/2022 10:37 AM - Constantin Asofiei

Greg Shah wrote:

> The memptr allocates memory outside of the Java heap. It uses the CRT (C language runtime) memory allocation functions. This memory does not need to be planned for and cannot really be planned for since it is dynamic (runtime based, driven by converted 4GL code) and application specific.

I'm not talking about the memptr stores the data outside of Java heap. I'm talking about if we need to transfer the entire memptr data from the client to the server (or vice-versa) - to send it over the network in its entirety, it must be loaded into the JVM heap; and if the data is 100MB, the JVM heap must allow it.

I can't find where, but we had this conversion before; when transferring memptr over the network, the client heap must be set to allow enough heap to hold the entire data - regardless if after that ends up allocated outside of the Java heap.

With memptr on server-side, this issue is somehow reduced. But we need to identify if there are any other places where a large blob of data is ends

up loaded in the client JVM entirely.

#39 - 05/25/2022 11:00 AM - Greg Shah

For this problem we need to implement transfer in chunks. I think we've discussed that before as well. Perhaps it is time to implement that so we can avoid heap issues.

#40 - 05/25/2022 11:33 AM - Tomasz Domin

Greg Shah wrote:

> Sergey/Eugenie: Do you have any thoughts on the icon color depth? What compatibility requirements do we have?

I've tested the latest version (3.8.2) of imageio-bmp and issue still exists, so upgrade does not help.
To be precise: the 64MB of unneeded memory is allocated when processing 24-bit BMP images with mask - like 24bit BMP icons.

#41 - 05/25/2022 11:46 AM - Sergey Ivanovskiy

Tomasz Domin wrote:

> Greg Shah wrote:
>
> > Sergey/Eugenie: Do you have any thoughts on the icon color depth? What compatibility requirements do we have?
>
> I've tested the latest version (3.8.2) of imageio-bmp and issue still exists, so upgrade does not help.
> To be precise: the 64MB of unneeded memory is allocated when processing 24-bit BMP images with mask - like 24bit BMP icons.

Please explain more thoroughly what is the issue that you grasped. What is the code?

#42 - 05/25/2022 11:53 AM - Eugenie Lyzenko

Greg Shah wrote:

> Sergey/Eugenie: Do you have any thoughts on the icon color depth? What compatibility requirements do we have?

As far as I know we use twelve-monkeys BMP plugin. The icon support can be found here:
https://github.com/haraldk/TwelveMonkeys/wiki/BMP-Plugin

#43 - 05/25/2022 01:31 PM - Tomasz Domin

Sergey Ivanovskiy wrote:

> Tomasz Domin wrote:
>
> > I've tested the latest version (3.8.2) of imageio-bmp and issue still exists, so upgrade does not help.
> > To be precise: the 64MB of unneeded memory is allocated when processing 24-bit BMP images with mask - like 24bit BMP icons.
>
> Please explain more thoroughly what is the issue that you grasped. What is the code?

There an issue in imageio-bmp library when it processes 24bit BMP images with mask e.g. BMP icons for customer application.
Such image consists of two bitmaps - actual image bitmap and image mask bitmap.
Library by mistake allocates memory for 64MB image mask color map. The allocated memory is not even used, as mask bitmap is only 1-bit deep and color map is not needed.

BitmapMask constructor calls BitmapIndexed constructor with inherited pHeader having set number of colors to 2^24.

```
    public BitmapMask(final DirectoryEntry pParent, final DIBHeader pHeader) {
        super(pParent, pHeader);
        bitMask = new BitmapIndexed(pParent, pHeader);
    }
```

getColorCount() returns 2^24 for 24bpp image.

```
    public BitmapIndexed(final DirectoryEntry pEntry, final DIBHeader pHeader) {
        super(pEntry, pHeader);
        bits = new int[getWidth() * getHeight()];

        // NOTE: We're adding space for one extra color, for transparency
        colors = new int[getColorCount() + 1];
    }
```

That causes short allocation and release of 64 MB of heap memory, which result in OOM in case -XmX is less then about 160MB

#44 - 05/25/2022 03:42 PM - Sergey Ivanovskiy

Cool! My be the library used this color array for another cases or there is a bug here.

#45 - 05/31/2022 10:55 AM - Tomasz Domin

- File fwd-imageio-bmp-3.1.2.zip added

- File image-io-bmp-3.1.2-fwd.patch added

- % Done changed from 40 to 50

Fixed the issue with image-io-bmp-3.1.2 unneeded memory allocation.
Attached:
image-io-bmp-3.1.2-fwd.patch - patch to sources
fwd-imageio-bmp-3.1.2.zip - version to put into maven repository.

I will update p2j 3821c build.gradle once image-io-bmp patched version is uploaded into GCD maven repository.

#46 - 05/31/2022 10:57 AM - Tomasz Domin

- % Done changed from 50 to 40

- File image-io-bmp-3.1.2-fwd.diff added

Tomasz Domin wrote:

> Fixed the issue with image-io-bmp-3.1.2 unneeded memory allocation.
> Attached:
> image-io-bmp-3.1.2-fwd.diff - patch to sources
> fwd-imageio-bmp-3.1.2.zip - version to put into maven repository.
>
> I will update p2j 3821c build.gradle once image-io-bmp patched version is uploaded into GCD maven repository.

#47 - 05/31/2022 10:57 AM - Tomasz Domin

- File deleted (image-io-bmp-3.1.2-fwd.patch)

#48 - 05/31/2022 11:18 AM - Tomasz Domin

Greg Shah wrote:

> For this problem we need to implement transfer in chunks. I think we've discussed that before as well. Perhaps it is time to implement that so we can avoid heap issues.

Should CLOB transfer in chunks become a part of this ticket ?
If so - I need a use case - I mean how to trigger CLOB transfer in customer application.

After startup client app is holding about 32 MB of heap memory, before GC its maxed to 64MB as I've specified -Xmx64m
I need to test it would work with client application in longer term.

Checking further:

- 4,8 MB consumed by downloaded fonts
- 6,8 MB consumed by virtual screen
- 2,7 MB consumed by method routing maps
- and following 50% by other small objects.

When I push -Xmx32M into client configuration the client does not start (OOM).

#49 - 06/01/2022 01:06 PM - Greg Shah

> I will update p2j 3821c build.gradle once image-io-bmp patched version is uploaded into GCD maven repository.

I've applied (unzipped) everything in the zip to the artifacts directory (except the manifest).

> Should CLOB transfer in chunks become a part of this ticket ?

Yes, I think it needs to be solved here so that we don't cause OOME.

> If so - I need a use case - I mean how to trigger CLOB transfer in customer application.

I think it is not just CLOB, but also BLOB and MEMPTR.

I think the easiest way is to use a temp-table and COPY-LOB from the CLOB/BLOB to a file. The MEMPTR can be done directly without a temp-table.

Something like this:

```
def temp-table something
   field big-txt as clob
   field big-bin as blob.

create something.

copy-lob from file clob_output.txt to something.big-txt.
copy-lob from file blob_output.bin to something.big-bin.
copy-lob from something.big-txt to clob_output.txt.
copy-lob from something.big-bin to blob_output.bin.

def var ptr as memptr.

copy-lob from something.big-txt to ptr.
copy-lob from ptr to something.big-txt.

set-size(ptr) = 0.

copy-lob from something.big-bin to ptr.
copy-lob from ptr to something.big-bin.

set-size(ptr) = 0.
```

#50 - 06/02/2022 12:32 PM - Tomasz Domin

Greg Shah wrote:

> I will update p2j 3821c build.gradle once image-io-bmp patched version is uploaded into GCD maven repository.

I've applied (unzipped) everything in the zip to the artifacts directory (except the manifest).

Updated build.gradle to reference patched fwd-image-io-bmp instead of image-io-bmp
Committed 3821c/13944.

#51 - 06/02/2022 12:34 PM - Tomasz Domin

> Should CLOB transfer in chunks become a part of this ticket ?

Yes, I think it needs to be solved here so that we don't cause OOME.

I've tested following code with chui, need to test it with gui.
For CHUI all transfers seem to executed on server side in server process.

```
def temp-table something
   field big-txt as clob
   field big-bin as blob.
```

```
         create something.
         copy-lob from file "/tmp/clob_output.txt" to something.big-txt.
         copy-lob from file "/tmp/blob_output.bin" to something.big-bin.
         copy-lob from something.big-txt to FILE "/tmp/clob_output-2.txt".
         copy-lob from something.big-bin to FILE "/tmp/blob_output-2.bin".
         def var ptr as memptr.
         //copy-lob from something.big-txt to ptr.
         //copy-lob from ptr to something.big-txt.
         //set-size(ptr) = 0.
         copy-lob from something.big-bin to ptr.
         copy-lob from ptr to something.big-bin.
         set-size(ptr) = 0.
         message "Test lob done".
```

Restoring LOBs is commented as there were some issues with codepage - I guess I need to configure sth here.
Will test it with hotel_gui.

#52 - 06/02/2022 12:34 PM - Tomasz Domin

- % Done changed from 40 to 50

#53 - 06/02/2022 12:57 PM - Greg Shah

> For CHUI all transfers seem to executed on server side in server process.

That should not be the case unless COPY-LOB is implemented completely on the server side. I hope not, that would be badly broken.

#54 - 06/03/2022 03:13 PM - Tomasz Domin

Of course this was my mistake before I've entered debugging session. Transfers are also on client side.
One thing that worries me that swing client or term client uses 60 MB memory on startup - then its reduced to 23 MB

Running debugging sessions on server and client, checking flows.
No code yet.

#55 - 06/06/2022 02:24 AM - Tomasz Domin

Some last week findings:
There is a dedicated class on client side responsible for reading/writing files StreamDaemon.java
StreamDaemon.java reads data in chunks, so it does not use a lot of memory for reading operation - this is to be left as it is.
StreamDaemon.java writes data in one piece, so memory goes up during writing operations
I work on memptr analysis. I thing found that memory was not fully released on set-size(ptr) = 0
I also need to check conversion scenarios (for LONGCHAR and CLOB).

Regarding memptr - it is possible to mark memptr as server side resource, if this works (to be checked) in server-side-resources section, so it could be used for client memory usage minimization.
There is a similar service on client side as for stream which is called MemoryDaemon
There is a potential issue with memptr transfer to client - network buffers are not shrinked back after the memptr transfer from client to server.

#56 - 06/06/2022 09:09 AM - Tomasz Domin

Tomasz Domin wrote:

> There is a potential issue with memptr transfer to client - network buffers are not shrinked back after the memptr transfer from client to server.

This is a generic problem with reusable ByteArrayOutputStream - this object only grows over time.
I'll try to update Protocol.java but there are several other places in code when ByteArrayOutputStream in reusable manner that probably would need to be reviewed.

#57 - 06/06/2022 10:46 AM - Tomasz Domin

Tomasz Domin wrote:

> Regarding memptr - it is possible to mark memptr as server side resource, if this works (to be checked) in server-side-resources section, so it could be used for client memory usage minimization.

I guess this is related with #4065 where server-side-resources were implemented.
Following code quits (makes application restart) on client when testing with server-side-resources=memptr

```
def temp-table something
   field big-txt as clob
   field big-bin as blob.

   create something.
   def var ptr as memptr.
   copy-lob from file "/tmp/blob_output.bin" to something.big-bin.
   copy-lob from something.big-bin to ptr.
```

Anyway - working server-side-resources=memptr should solve memory usage on client for large memory allocations for memptr.

As reading from client stream reads memory in chunks so it does not consume much memory, the only thing to be fixed is writing BLOB/CLOB to file.

#58 - 06/06/2022 10:46 AM - Tomasz Domin

- Related to Feature #4065: server-side processing of client platform dependencies added

#59 - 06/06/2022 10:53 AM - Greg Shah

> Following code quits (makes application restart) on client when testing with server-side-resources=memptr

Is this the same the the blob/clob issues or a separate problem?

#60 - 06/06/2022 12:57 PM - Tomasz Domin

Greg Shah wrote:

> Following code quits (makes application restart) on client when testing with server-side-resources=memptr
>
> Is this the same the the blob/clob issues or a separate problem?

No, it seems a problem with security configuration of the app for that feature. Will check it tomorrow. I just wanted to see memory usage in action for server-side-resources=memptr

I've patched Protocol.java and memory usage is lower and does not grow over time.

#61 - 06/07/2022 03:28 AM - Tomasz Domin

I've added security configuration and server-side-resources=memptr feature started to work correctly with following memory usage tests results.
It seems that the feature can be used for reducing memory usage on client side.
The only thing to fix is writing to files on client side to make it write files in chunks.

Results:

```
         copy-lob from something.big-bin to ptr. -> no memory usage increase on client
         PUT-BYTE(ptr,1) = 65.
         copy-lob from ptr to something.big-bin. -> no memory usage increase on client
         set-size(ptr) = 0.
         copy-lob from file "/tmp/blob_output.bin" to ptr. -> no memory usage increase on client
         copy-lob from ptr to file "/tmp/blob_output.bin". -> memory increase due to writing to file in single
 chunk
         set-size(ptr) = 0.
```

Server configuration:

```
  <node class="container" name="">
    <node class="container" name="server">
      <node class="container" name="runtime">
        <node class="container" name="default">
          <node class="string" name="server-side-resources">
            <node-attribute name="value" value="memptr"/>
          </node>
...
```

Security configuration:

```
  <node class="container" name="">
      <node class="container" name="acl">
```

```
        <node class="container" name="net">
          <node class="container" name="1950">
            <node class="strings" name="subjects">
              <node-attribute name="values" value="all_others"/>
            </node>
            <node class="netRights" name="rights">
              <node-attribute name="permissions" value="'0111'B"/>
            </node>
            <node class="resource" name="resource-instance">
              <node-attribute name="reference" value="com.goldencode.p2j.util.LowLevelBuffer"/>
              <node-attribute name="reftype" value="TRUE"/>
            </node>
          </node>
...
```

#62 - 06/07/2022 04:49 AM - Tomasz Domin

Regarding writing LOB files input/output.
Reading LOB files is implemented using InputStreamWrapper which reads data in chunks (1MB by default).
Writing LOB files is implemented by calling Stream interface directly. Unfortunately OutputStreamWrapper is not writing data in chunks so switching file class writing to remote file wont immediately solve the case.

My idea is to perform two steps:
- implement chunked writing in OutputStreamWrapper - this would also save potential memory issues with other places it is in use
- switch TargetLobFile to OutputStreamWrapper

I'll also check for other direct references to Stream for potential memory usage problems.

#63 - 06/07/2022 06:41 AM - Greg Shah

Good plan.

#64 - 06/07/2022 04:17 PM - Tomasz Domin

- File testblob.p added

- % Done changed from 50 to 70

Implemented #5776#note-62

Tested with chui_hotel swing client with heap limit set to -Xmx64m, configuration as in #5776#note-61 and input files/blobs/clobs 100MB big (no way it could fit in client memory).

Executed procedure testblob.p which tests different cases.
During execution:

- Memory usage on client has not exceeded 60Mb, no OOM occurs.
- Execution of procedure allocates up to 2GB of heap RAM on Server, which goes down to 500MB on GC pass after procedure completes, and comes back to 200MB when client is closed

I need to check non-heap memory allocation/release and I will also review code and submit a patch tomorrow.

#65 - 06/08/2022 06:32 AM - Tomasz Domin

- File 5776-1.diff added

- Status changed from WIP to Review

- % Done changed from 70 to 80

I've run more tests and it seems hotel_chui swing/console client also executes the test correctly when having set "-Xmx32m". But its probably the minimum as I cannot connect jconsole, as it needs some memory to operate. Note that for other apps memory allocated to client depends on use case.

Generation of test input files:

```
dd bs=100 count=1048576 </dev/urandom >/tmp/blob_input.bin
yes "this is test file" | head -c 100MB > /tmp/clob_input.txt
```

After running the test its possible to check result:

```
diff /tmp/blob_input.bin /tmp/blob_output-1.bin
diff /tmp/blob_input.bin /tmp/blob_output-2.bin
diff /tmp/clob_input.txt /tmp/clob_output-1.txt
```

Attached a patch 5776-1.diff that allows processing BLOB/CLOB objects with low memory requirements on client.

Please review and let me know if I should be looking for further memory savings on client side.

#66 - 06/08/2022 10:55 AM - Tomasz Domin

- % Done changed from 80 to 90

I checked non-heap memory usage and it is correctly allocated and de-allocated.
Above mentioned 500 GB heap seems to be allocated to internal in-memory H2 temp database. It is released after client disconnects.

I've fixed hotel_chui to build with the latest p2j 3821c and committed to samples/hotel (revno: 72)

#67 - 06/08/2022 01:45 PM - Greg Shah

Hynek: Please review.

I'm OK with closing the task at this level. We can always look further in the future if needed, but 64MB for a GUI client is pretty good.

#68 - 06/08/2022 01:46 PM - Roger Borrello

Are there changes to our projects' directories that we should plan on making, in order to benefit from this great enhancement?

#69 - 06/08/2022 01:52 PM - Greg Shah

The directory changes are in relation to #4065. They aren't necessary for this task, per se.

#70 - 06/09/2022 01:21 AM - Tomasz Domin

Greg Shah wrote:

> The directory changes are in relation to #4065. They aren't necessary for this task, per se.

Roger
Directory changes from #4065 and #5776#note-61 are needed if your application is making use of memptr to store large objects - so they are moved from client side to server side.
Otherwise no changes required. Optionally you may reduce heap memory size in startup scripts or web client spawning section of the directory.

I've tested a bit client application and heap memory usage was about 27-30MB, but I havent tested all use cases and probably more is needed when application runs longer.

#71 - 06/09/2022 09:12 AM - Hynek Cihlar

Code review 3821c revision 13944. The changes are good. But my worry is about maintaining the TwelveMonkeys changes. Did you consider forking the project on Github and integrating the changes back in the base project?

#72 - 06/09/2022 09:35 AM - Hynek Cihlar

Code review 3821c revision 13944.

Protocol.java and OutputStreamWrapper.java require copyright year to be bumped up to 2022.

In OutputStreamWrapper:

off > b.length check is redundant in if ((off + len) > b.length || off < 0 || off > b.length || len < 0).

I think the reallocation in write method:

```
         if (remaining > 0 && remaining < WRITE_CHUNK_SIZE)
         {
            buffer = new byte[remaining];
         }
```

is unnecessary if you change out.write(buffer, 0, buffer.length); to write chunk size or remaining size bytes.

Please add javadoc to TargetLobFile.openStream method and move it in the private section of of the file.

#73 - 06/09/2022 10:47 AM - Tomasz Domin

- File 5776-2.diff added

Thank you Hynek.

> Code review 3821c revision 13944. The changes are good. But my worry is about maintaining the TwelveMonkeys changes. Did you consider forking the project on Github and integrating the changes back in the base project?

I was rather considering submitting a minor issue report with a diff.
Note that we are using quite old version of the library. We need sooner or later to migrate to version supporting Java 9+ (or to the latest - depending on bugs discovered).
My little concern is the library growing in size over time with functionality not needed for p2j project.

> Code review 3821c revision 13944.
>
> In OutputStreamWrapper:
>
> off > b.length check is redundant in if ((off + len) > b.length || off < 0 || off > b.length || len < 0).

Correct.
I've also added if (remaining == 0) so buffer memory is not allocated when it does not need to.

> I think the reallocation in write method:
> [...]
>
> is unnecessary if you change out.write(buffer, 0, buffer.length); to write chunk size or remaining size bytes.

out.write() always sends out full array no matter of length and offset values, so this is needed to minimize communication effort in case out is remote. Otherwise partially empty buffer would be transmitted. I presume re-allaction is cheaper than over-transmission.

> Please add javadoc to TargetLobFile.openStream method and move it in the private section of of the file.

Done

Attached corrected diff file 5776-2.diff

#74 - 06/09/2022 11:38 AM - Hynek Cihlar

Tomasz Domin wrote:

> out.write() always sends out full array no matter of length and offset values,
> so this is needed to minimize communication effort in case out is remote.
> Otherwise partially empty buffer would be transmitted. I presume re-allaction is cheaper than over-transmission.

I don't think you can assume any fragmentation or flushing of the underlying output stream at this level. And so it's best to keep this concern to the caller.

#75 - 06/09/2022 01:39 PM - Tomasz Domin

- File 5776-3.diff added

Hynek Cihlar wrote:

> Tomasz Domin wrote:
>
> > out.write() always sends out full array no matter of length and offset values,
> > so this is needed to minimize communication effort in case out is remote.
> > Otherwise partially empty buffer would be transmitted. I presume re-allaction is cheaper than over-transmission.
>
> I don't think you can assume any fragmentation or flushing of the underlying output stream at this level. And so it's best to keep this concern to the caller.

I've removed buffer reallocation as you've asked , out.write() always sends out full buffer to underlying stream.

I've tested the change with patched hotel_chui.

Attached please find corrected 5776-3.diff

#76 - 06/09/2022 01:45 PM - Hynek Cihlar

Tomasz Domin wrote:

> Attached please find corrected 5776-3.diff

The changes look good.

#77 - 06/10/2022 08:11 AM - Alexandru Lungu

I have an issue when using ./gradlew all to build FWD:

```
> Could not resolve all files for configuration ':fwdAllCompile'.
   > Could not resolve com.twelvemonkeys.imageio:fwd-imageio-bmp:3.1.2.
     Required by:
         project :
      > Could not resolve com.twelvemonkeys.imageio:fwd-imageio-bmp:3.1.2.
         > Could not get resource 'https://repository.jboss.org/nexus/content/groups/public/com/twelvemonkeys/imageio/fwd-imageio-bmp/3.1.2/fwd-imageio-bmp-3.1.2.pom'.
            > Could not HEAD 'https://repository.jboss.org/nexus/content/groups/public/com/twelvemonkeys/imageio/fwd-imageio-bmp/3.1.2/fwd-imageio-bmp-3.1.2.pom'. Received status code 503 from server: Service Unavailable
```

This is due to the new fwd-imageio-bmp, which is no longer resolved by the jboss repository, but through the goldencode repository. The issue here is that the jboss repository is responding with 503 instead of 404 (jboss issue? nexus issue? my environment issue?). Anyways, Gradle should query the next repository when 404 is encountered. Unfortunately 503 is blocking for the dependency lookup and Gradle fails. I suggest the following workaround: move the jboss at the end of the repositories list. Please let me know if you can reproduce this issue.

#78 - 06/10/2022 08:31 AM - Tomasz Domin

Alexandru Lungu wrote:

> I have an issue when using ./gradlew all to build FWD:
>
> [...]
>
> This is due to the new fwd-imageio-bmp, which is no longer resolved by the jboss repository, but through the goldencode repository. The issue here is that the jboss repository is responding with 503 instead of 404 (jboss issue? nexus issue? my environment issue?). Anyways, Gradle should query the next repository when 404 is encountered. Unfortunately 503 is blocking for the dependency lookup and Gradle fails. I suggest the following workaround: move the jboss at the end of the repositories list. Please let me know if you can reproduce this issue.

I cannot replicate, for me it works just fine. Attached logs and gradle version information.
IMHO Based on the allowed 3rd party libraries list I guess fwd repository should be first and the only one in the list.

Logs:

```
14:24:12.393 [DEBUG]  Performing HTTP HEAD: https://repo.maven.apache.org/maven2/com/twelvemonkeys/imageio/fwd
-imageio-bmp/3.1.2/fwd-imageio-bmp-3.1.2.pom
14:24:12.417 [INFO]  Resource missing. [HTTP HEAD: https://repo.maven.apache.org/maven2/com/twelvemonkeys/imag
eio/fwd-imageio-bmp/3.1.2/fwd-imageio-bmp-3.1.2.pom]
14:24:12.418 [DEBUG]  Performing HTTP HEAD: https://repo.maven.apache.org/maven2/com/twelvemonkeys/imageio/fwd
-imageio-bmp/3.1.2/fwd-imageio-bmp-3.1.2.jar
14:24:12.442 [INFO]  Resource missing. [HTTP HEAD: https://repo.maven.apache.org/maven2/com/twelvemonkeys/imag
eio/fwd-imageio-bmp/3.1.2/fwd-imageio-bmp-3.1.2.jar]
14:24:12.443 [DEBUG]  Performing HTTP HEAD: https://repo1.maven.org/maven2/com/twelvemonkeys/imageio/fwd-image
io-bmp/3.1.2/fwd-imageio-bmp-3.1.2.pom
14:24:12.470 [INFO]  Resource missing. [HTTP HEAD: https://repo1.maven.org/maven2/com/twelvemonkeys/imageio/fw
d-imageio-bmp/3.1.2/fwd-imageio-bmp-3.1.2.pom]
14:24:12.470 [DEBUG]  Performing HTTP HEAD: https://repo1.maven.org/maven2/com/twelvemonkeys/imageio/fwd-image
io-bmp/3.1.2/fwd-imageio-bmp-3.1.2.jar
14:24:12.494 [INFO]  Resource missing. [HTTP HEAD: https://repo1.maven.org/maven2/com/twelvemonkeys/imageio/fw
d-imageio-bmp/3.1.2/fwd-imageio-bmp-3.1.2.jar]
14:24:12.497 [DEBUG]  Performing HTTP HEAD: https://repository.jboss.org/nexus/content/groups/public/com/twelv
emonkeys/imageio/fwd-imageio-bmp/3.1.2/fwd-imageio-bmp-3.1.2.pom
14:24:12.641 [INFO]  Resource missing. [HTTP HEAD: https://repository.jboss.org/nexus/content/groups/public/co
m/twelvemonkeys/imageio/fwd-imageio-bmp/3.1.2/fwd-imageio-bmp-3.1.2.pom]
14:24:12.641 [DEBUG]  Performing HTTP HEAD: https://repository.jboss.org/nexus/content/groups/public/com/twelv
emonkeys/imageio/fwd-imageio-bmp/3.1.2/fwd-imageio-bmp-3.1.2.jar
14:24:12.883 [INFO]  Resource missing. [HTTP HEAD: https://repository.jboss.org/nexus/content/groups/public/co
m/twelvemonkeys/imageio/fwd-imageio-bmp/3.1.2/fwd-imageio-bmp-3.1.2.jar]
14:24:12.884 [DEBUG]  Performing HTTP HEAD: https://proj.goldencode.com/artifacts/com/twelvemonkeys/imageio/fw
d-imageio-bmp/3.1.2/fwd-imageio-bmp-3.1.2.pom
14:24:13.023 [DEBUG]  Performing HTTP GET: https://proj.goldencode.com/artifacts/com/twelvemonkeys/imageio/fwd
-imageio-bmp/3.1.2/fwd-imageio-bmp-3.1.2.pom.sha1
14:24:30.348 [DEBUG]  Performing HTTP HEAD: https://proj.goldencode.com/artifacts/com/twelvemonkeys/imageio/fw
d-imageio-bmp/3.1.2/fwd-imageio-bmp-3.1.2.jar
14:24:30.493 [DEBUG]  Performing HTTP GET: https://proj.goldencode.com/artifacts/com/twelvemonkeys/imageio/fwd
-imageio-bmp/3.1.2/fwd-imageio-bmp-3.1.2.jar.sha1
```

```
./gradlew -version
------------------------------------------------------------
Gradle 5.6.4
------------------------------------------------------------

Build time:   2019-11-01 20:42:00 UTC
Revision:     dd870424f9bd8e195d614dc14bb140f43c22da98

Kotlin:       1.3.41
Groovy:       2.5.4
Ant:          Apache Ant(TM) version 1.9.14 compiled on March 12 2019
JVM:          1.8.0_312 (Private Build 25.312-b07)
OS:           Linux 5.13.0-48-generic amd64
```

#79 - 06/10/2022 09:51 AM - Alexandru Lungu

I am running the same version of Gradle. It is odd the fact that when I access https://repository.jboss.org/nexus/content/groups/public/com/twelvemonkeys/imageio/fwd-imageio-bmp/3.1.2/fwd-imageio-bmp-3.1.2.pom I get

```
Service Unavailable
The server is temporarily unable to service your request. Please try again later.

Reference #6.ddb9c451.1654868577.12f61ef4
```

However, when accessing some other URLS:

- https://repository.jboss.org/nexus/content/groups/public/com/twelvemonkeys/imageio/ - I get 200 OK and a list of several resources.
- https://repository.jboss.org/nexus/content/groups/public/com/twelvemonkeys/imageio/fwd-imageio-bmp - I get 404 Not Found
- https://repository.jboss.org/nexus/content/groups/public/com/twelvemonkeys/imageio/fwd-imageio-bmp/3.1.2 - I get 503 Service Unavailable
- https://repository.jboss.org/nexus/content/groups/public/com/twelvemonkeys/imageio/fwd-imageio-bmp/3.1.2/fwd-imageio-bmp-3.1.2.pom - I get 503 Service Unavailable

#80 - 06/10/2022 09:56 AM - Tomasz Domin

Alexandru Lungu wrote:

> I am running the same version of Gradle. It is odd the fact that when I access https://repository.jboss.org/nexus/content/groups/public/com/twelvemonkeys/imageio/fwd-imageio-bmp/3.1.2/fwd-imageio-bmp-3.1.2.pom I get
>
> [...]

Its strange, do you have any kind of proxy ?
For me:

```
curl -v https://repository.jboss.org/nexus/content/groups/public/com/twelvemonkeys/imageio/fwd-imageio-bmp/3.1
.2/fwd-imageio-bmp-3.1.2.pom
....
> GET /nexus/content/groups/public/com/twelvemonkeys/imageio/fwd-imageio-bmp/3.1.2/fwd-imageio-bmp-3.1.2.pom H
TTP/1.1
> Host: repository.jboss.org
> User-Agent: curl/7.68.0
> Accept: */*
>
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
* old SSL session ID is stale, removing
* Mark bundle as not supporting multiuse
< HTTP/1.1 404 Not Found
....
```

#81 - 06/10/2022 10:28 AM - Alexandru Lungu

I fired two curls (interval of some seconds) in my terminal and I got:

```
...
> GET /nexus/content/groups/public/com/twelvemonkeys/imageio/fwd-imageio-bmp/3.1.2/fwd-imageio-bmp-3.1.2.pom H
TTP/1.1
> Host: repository.jboss.org
> User-Agent: curl/7.68.0
> Accept: */*
>
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
* old SSL session ID is stale, removing
* Mark bundle as not supporting multiuse
< HTTP/1.1 503 Service Unavailable
...
```

and

```
> GET /nexus/content/groups/public/com/twelvemonkeys/imageio/fwd-imageio-bmp/3.1.2/fwd-imageio-bmp-3.1.2.pom H
TTP/1.1
> Host: repository.jboss.org
> User-Agent: curl/7.68.0
> Accept: */*
>
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
* old SSL session ID is stale, removing
* Mark bundle as not supporting multiuse
< HTTP/1.1 404 Not Found
...
```

The first was resolved to 80.239.150.57:443 and the second to 104.126.36.216:443. I explicitly "curled" each one and it seems that only the second is returning 404. I guess a load-balancer just saved me from a broken service and moved me to a working one. Now things are back to normal; sorry for the trouble.

#83 - 07/17/2022 11:03 AM - Greg Shah

If I understand correctly, we are good to go with the proposed change. Is there any reason to **not** commit this?

#84 - 08/30/2022 07:51 PM - Greg Shah

Tomasz: Go ahead and merge the 5776-3.diff changes.

#85 - 08/30/2022 07:51 PM - Greg Shah

Will this change the results of our testing in #6560?

#86 - 10/14/2022 05:06 PM - Greg Shah

> Tomasz: Go ahead and merge the 5776-3.diff changes.

Tomasz: Please merge the changes.

#87 - 10/17/2022 03:08 AM - Tomasz Domin

- Status changed from Review to Test

- File 5776-4.diff added

Needed to update the patch
Committed revision 14299.

#88 - 10/17/2022 03:08 AM - Tomasz Domin

- % Done changed from 90 to 100

#89 - 10/17/2022 09:15 AM - Roger Borrello

Are there any tuning considerations we should make to our client configurations already documented?

#90 - 10/17/2022 09:50 AM - Tomasz Domin

Roger Borrello wrote:

> Are there any tuning considerations we should make to our client configurations already documented?

The feature is not configurable. Please report if there are any issues or slowdowns related with LOB data transfer.

Files

fwd-imageio-bmp-3.1.2.zip	4.04 MB	05/31/2022	Tomasz Domin
image-io-bmp-3.1.2-fwd.diff	1.35 KB	05/31/2022	Tomasz Domin

testblob.p	1.3 KB	06/07/2022	Tomasz Domin
5776-1.diff	8.65 KB	06/08/2022	Tomasz Domin
5776-2.diff	9.5 KB	06/09/2022	Tomasz Domin
5776-3.diff	9.37 KB	06/09/2022	Tomasz Domin
5776-4.diff	9.01 KB	10/17/2022	Tomasz Domin