

Monitoring a z/OS Liberty server using JMX and REST clients



Monitoring a z/OS Liberty server

Table of Contents

Using Java Management Extensions with Liberty	2
Liberty Server Considerations.....	3
Enable required Liberty features	3
Define and provide access to the require SAF resources	4
Export the server's Certificate Authority chain.....	5
Identify the JMX service endpoint	6
Configuring the Java Console (JConsole) GUI	7
Create a JConsole command file.....	8
Completing the configuration of the client's environment.....	8
Configuring TLS support	8
Create a local trust store	10
Using the JConsole	11
Exploring the information available in the JConsole tabs	12
Explore the Liberty server's available MBeans.....	15
Using the IBM API Explorer to access MBeans using REST	22
Accessing MBean ConnectionManagerMBean.....	26
Accessing MBean ServerXMLConfigurationMBean	30
Using Postman to access MBeans using REST	32
Accessing MBean JvmStats.....	33
Accessing MBean ConnectionManagerMBean.....	36
Using cURL to access MBeans using REST	39
Building cURL commands from IBM API Explorer.....	39
Building cURL commands from Postman.....	40

Monitoring a z/OS Liberty server

Using Java Management Extensions with Liberty

Monitoring and/or obtaining information about the current settings and general health of a Liberty server running on z/OS can easily be done by enabling Java Management Extensions (JMX) in the server. Once JMX is enabled, JMX clients can invoke the server's Java Virtual Machine (JVM) management beans (MBeans) of the Liberty server to obtain important Java related information in real time. The MBeans available in a Liberty will vary based which features are or not installed as well as the presence of certain configuration elements.

For details on Liberty's support for remote JMX clients, see *Connecting to Liberty using JMX* at URL https://www.ibm.com/docs/en/was-liberty/zos?topic=SS7K4U_liberty/com.ibm.websphere.wlp.doc/ae/twlp_admin_jmx.htm. And for a list of Liberty provided MBeans, see URL <https://www.ibm.com/docs/sk/was-liberty/base?topic=liberty-list-provided-mbeans>.

In this document I describe the steps I followed to enable JMX and Mbean access in my Liberty servers. With the server properly configured, the server's MBeans can be invoked from JMX clients like JConsole* and REST clients like Postman, cURL, and the IBM API Explorer. The document also provide examples of using these clients to monitor a Liberty server running on z/OS.

*Please note that most Java Developers Kit (JDK) or Java Runtime Environment (JRE) instances include a Java application known as the Java Console(JConsole). JConsole is a Java GUI client which can be used to manage and monitor local and remote JVMs. For more details on JConsole, see URL <https://docs.oracle.com/javase/8/docs/technotes/guides/troubleshoot/tooldescr009.html>

Monitoring a z/OS Liberty server

Liberty Server Considerations

This section describes the details of the configuration changes made and security resources defined to enable both JMX and REST access to a Liberty server endpoint.

Enable required Liberty features

Liberty features *monitor-1.0* and *restConnector-2.0* are required to enable JMX clients like JConsole and REST clients like Postman and cURL to access a Liberty Server. I ensured these features were configured in a `<featureManager>` section of my server XML and then I ensure that they had been enabled in the target server.

Note: Liberty servers on z/OS also require the use of TLS for REST client access. It is my opinion that that enabling TLS connection and other required security features is easier done using SAF security.

```
<!-- Enable features -->
<featureManager>
  <feature>appSecurity-2.0</feature>*
  <feature>zosSecurity-1.0</feature>*
  <feature>transportSecurity-1.0</feature>*
  <feature>apiDiscovery-1.0</feature>*
  <feature>monitor-1.0</feature>
  <feature>restConnector-2.0</feature>
</featureManager>
```

*I always automatically add these features when enabling SAF and TLS security.

The *monitor-1.0* adds standard monitoring MBeans as well as MXBeans, the latter provide monitoring for specific runtime components. For more information about the *monitor-1.0* feature including the MXBeans added by this feature, see URL

<https://www.ibm.com/docs/en/was-liberty/zos?topic=environment-monitoring-monitor-10>.

The *restConnector-2.0* feature adds remote REST client access to a set of administrative APIs over HTTP. For more information about the *restConnector-2.0* feature, see URL

<https://www.ibm.com/docs/en/was-liberty/zos?topic=features-admin-rest-connector-20>.

Monitoring a z/OS Liberty server

Define and provide access to the require SAF resources

Access to Liberty MBeans and REST administrative APIs is controlled by access to SAF EJBRole resources, see URL <https://www.ibm.com/docs/en/was-liberty/zos?topic=liberty-mapping-management-roles-zos> for more information.

Below are examples of the commands I used to define and permit access to the *Administrator* and *Reader* EJBRoles.

```
RDEFINE EJBROLE BBGZDFLT.com.ibm.ws.management.security.resource.Administrator OWNER(SYS1) UACC(NONE)
RDEFINE EJBROLE BBGZDFLT.com.ibm.ws.management.security.resource.Reader OWNER(SYS1) UACC(NONE)
PERMIT BBGZDFLT.com.ibm.ws.management.security.resource.Administrator CLASS(EJBROLE) ID(ADMNUSRS) ACCESS(READ)
PERMIT BBGZDFLT.com.ibm.ws.management.security.resource.Reader CLASS(EJBROLE) ID(OTHUSRS) ACCESS(READ)

SETR RACLIST(EJBROLE) REFRESH
```

Using the above RACF resources, members of group *ADMNUSRS* will have full access to the JMX resources and MBeans (REST GET, POST, PUT and DELETE). Members of group *OTHUSRS* have limited access to the MBeans (REST GET only). For example, members of the OTHUSRS group will not be able to initiate a garbage collection or access MBeans or change attributes.

Note: the value for *BBGZDFLT* used in the commands was derived from the value of the *profilePrefix* attribute used in the *<safCredentials/>* configuration element.

```
<safCredentials unauthenticatedUser="WSGUEST"
  suppressAuthFailureMessages="false" profilePrefix="BBGZDFLT" />
```

Monitoring a z/OS Liberty server

Export the server's Certificate Authority chain

Connecting to a z/OS Liberty server to access its MBeans and REST APIs requires the use of TLS. When TLS is required, the client must be prepared to handle the TLS handshake information sent from the server. The Liberty server will send a message to client after a connection request with public details based its server certificate. Completing the TLS handshake on the client requires access to the CA chain used to sign the personal certificate that is being used by the server endpoint during the handshake. This section describes what is required on the z/OS side

- Locate in the server's XML configuraton the `<sslDefault/>` configuraton element and use it information to locate the `<keyStore>` element that identifies the trust store key ring used for inbound TLS connections. This will be the `keyStore` identified by the `trustStoreRef` attribute in the SSL repertoire identify by the `sslRef` attribute in the

```
<sslDefault sslRef="DefaultSSLSettings" />
<ssl id="DefaultSSLSettings"
  keyStoreRef="CellDefaultKeyStore"
  trustStoreRef="CellDefaultKeyStore" />
<keyStore id="CellDefaultKeyStore"
  location="safkeyring:///Liberty.KeyRing"
  password="password" type="JCERACFKS"
  fileBased="false" readOnly="true" />
```

`<sslDefault/>` configuration element.

Next, identify the Certificate Authority (CA) chain used to sign the server's personal certificate by displaying the contents of the key ring using a RACDCERT LISTRING command:

`racdcert id(libserv) listring(Liberty.KeyRing)`

Digital ring information for user LIBSERV:

Ring:

>Liberty.KeyRing<

Certificate Label Name	Cert Owner	USAGE	DEFAULT
Liberty Client Cert	ID (LIBSERV)	PERSONAL	YES
Liberty CA	CERTAUTH	CERTAUTH	NO

- Export or extract the Certificate Authority (CA) chain used to sign the server's personal certificate by using a RACDCERT EXPORT command to export the public portion of the CA certificate chain to a MVS dataset.

`RACDCERT CERTAUTH EXPORT(LABEL('Liberty CA')) DSN('USER1.LIBCA.PEM')`

This command should be repeated if there are intermediate CA certificates in the CA chain in the keyring.

Monitoring a z/OS Liberty server

Identify the JMX service endpoint

Obtain and record the service endpoint to be used with JConsole by starting in the server's configuration directory, *\$WLP_USER_DIR/servers/myServer/* and then go into the *logs/state* subdirectory, e.g., *\$WLP_USER_DIR/servers/myServer/logs/state/*. Browse the file *com.ibm.ws.jmx.rest.address* and record the contents of this file for use for the JConsole server's endpoint. This endpoint will be required to configure the JMX client with the connection URL for the Liberty server.

service:jmx:rest://wg31.washington.ibm.com:9443/IBMJMXConnectorREST

Monitoring a z/OS Liberty server

Configuring the Java Console (JConsole) GUI

Locate an instance of the JConsole executable on the client. The JConsole GUI is started by invoking a *jconsole.exe* executable. If Java has been installed on the client, then this executable is already installed. The *jconsole.exe* executable can be located the *bin* directory of any Java Developers Kit (JDK) or Java Runtime Environment (JRE) instance that has been installed on a client.

- I used the command *dir jconsole* /s* in a DOS command prompt in the client's root directory to search for the directories containing a *jconsole.exe* file. I had several copies on my client, and, in my case, I chose to use the version shipped with IBM Explorer, e.g., *C:\Program Files\IBM\zOS_Explorer\jdk\bin*.

```
C:\>dir jconsole* /s
Volume in drive C is Windows
Volume Serial Number is DA1F-5D24

Directory of C:\Program Files\IBM\Developer_for_zOS\jdk\bin

08/24/2023  04:48 PM                23,912 jconsole.exe
              1 File(s)                23,912 bytes

Directory of C:\Program Files\IBM\zOS_Explorer\jdk\bin

05/19/2023  07:57 AM                23,912 jconsole.exe
              1 File(s)                23,912 bytes

Directory of
C:\Users\948478897\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.10.v20240120-1143\jre\bin

01/20/2024  12:28 PM                23,944 jconsole.exe
              1 File(s)                23,944 bytes

Directory of C:\Users\948478897\.vscode\extensions\redhat.java-1.30.0-win32-x64\jre\17.0.10-win32-x86_64\bin

04/30/2024  02:48 PM                23,944 jconsole.exe
              1 File(s)                23,944 bytes
```


Monitoring a z/OS Liberty server

Create a JConsole command file

- Create a command file in a directory that was listed in the PATH environment variable. The command file should have the contents below (all on one line).

```
C:\Progra~1\IBM\zOS_Explorer\jdk\bin\jconsole  
-J-Djava.class.path=c:\z\jconsole\jconsole.jar;c:\z\jconsole\tools.jar;  
c:\z\jconsole\restConnector.jar  
-J-Djavax.net.ssl.trustStore=C:\z\jconsole\jconsole.jks  
-J-Djavax.net.ssl.trustStorePassword=changeit  
-J-Djavax.net.ssl.trustStoreType=jceks
```

- The *java.class.path* directive added the required Java Archive files to the Java CLASSPATH. Remember that connecting or accessing a z/OS Liberty server from Jconsole requires the use of TLS. The *javax.net.ssl...* directives identify the local attributes required to complete a TLS handshake with the z/OS Liberty server, e.g., a local file to be used for a trust store, the trust store's password, and the trust store's type.

Completing the configuration of the client's environment

- Download the required JAR files from OMVS directories below and placed them on a local drive on the client, e.g., /z/jconsole.

```
/usr/lpp/java/J8.0_64/lib/jconsole.jar  
/usr/lpp/java/J8.0_64/lib/tools.jar  
/usr/lpp/IBM/zosconnect/v3r0/wlp/clients/restConnector.jar
```

The location of the last JAR file assume that z/OS Connect is installed. If it is not, an instance of the *restConnector.jar* can be found in your OMVS or USS directory by enter this command in the root directory in an OMVS shell. ***find . -name restConnector.jar***

```
JOHNSON:/:> find . -name restConnector.jar  
./usr/lpp/liberty_zos/21.0.0.9/clients/restConnector.jar  
./usr/lpp/liberty_zos/22.0.0.3/clients/restConnector.jar  
./usr/lpp/cicsts/cicsts54/wlp/clients/restConnector.jar  
./usr/lpp/cicsts/cicsts61/wlp/clients/restConnector.jar  
./usr/lpp/mqm/V9R2M5/web/clients/restConnector.jar  
./usr/lpp/mqm/V9R3M0/web/clients/restConnector.jar  
./usr/lpp/IBM/zosconnect/v3r0/wlp/clients/restConnector.jar
```

Configuring TLS support

- To complete the client's TLS setup, create a local trust store which will contain the public Certificate Authority (CA) certificates used to sign the certificate being sent by the server during the handshake.

Monitoring a z/OS Liberty server

Download the exported CA chain dataset(s) to the client, for example case using FTP.

```
C:\z>ftp wg31
Connected to wg31.washington.ibm.com.
220-FTP 20:17:47 on 2024-05-09.
220 Connection will close if idle for more than 200 minutes.
501 command OPTS aborted -- no options supported for UTF8
User (wg31.washington.ibm.com:(none)): user1
331 Send password please.
Password:

230 USER1 is logged on. Working directory is "USER1.".
ftp> mget certauth.pem
200 Representation type is Ascii NonPrint
mget CERTAUTH.PEM? y
200 Port request OK.
125 Sending data set USER1.CERTAUTH.PEM
250 Transfer completed successfully.
ftp: 1236 bytes received in 0.00Seconds 1236000.00Kbytes/sec.
```

Monitoring a z/OS Liberty server

Create a local trust store

Create the local JSE trust store that will be used with JConsole using the Java keytool command.

```
keytool -import -v -trustcacerts -alias "Liberty CA" -file CERTAUTH.PEM -keystore myTest.jks
```

```
Enter keystore password: changeit

Re-enter new password: changeit

Owner: CN=CA for Liberty, OU=LIBERTY
Issuer: CN=CA for Liberty, OU=LIBERTY
Serial number: 0
Valid from: Tue Mar 05 23:00:00 EST 2024 until: Tue Dec 31 22:59:59 EST 2024
Certificate fingerprints:
    SHA1: 23:DB:1B:2B:61:E9:14:9D:40:2A:65:39:1E:E0:B4:27:EB:3A:15:99
    SHA256:
CE:BC:52:C3:95:C7:EE:15:CA:15:A7:6E:82:4D:38:34:80:E1:03:A9:50:3D:E2:20:E2:41:18:58:12:C9:
E7:41
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3

Extensions:

#1: ObjectId: 2.16.840.1.113730.1.13 Criticality=false
0000: 16 30 47 65 6E 65 72 61    74 65 64 20 62 79 20 74    .0Generated by t
0010: 68 65 20 53 65 63 75 72    69 74 79 20 53 65 72 76    he Security Serv
0020: 65 72 20 66 6F 72 20 7A    2F 4F 53 20 28 52 41 43    er for z/OS (RAC
0030: 46 29                                F)

#2: ObjectId: 2.5.29.19 Criticality=true
BasicConstraints:[
    CA:true
    PathLen:2147483647
]

#3: ObjectId: 2.5.29.15 Criticality=true
KeyUsage [
    Key_CertSign
    Crl_Sign
]

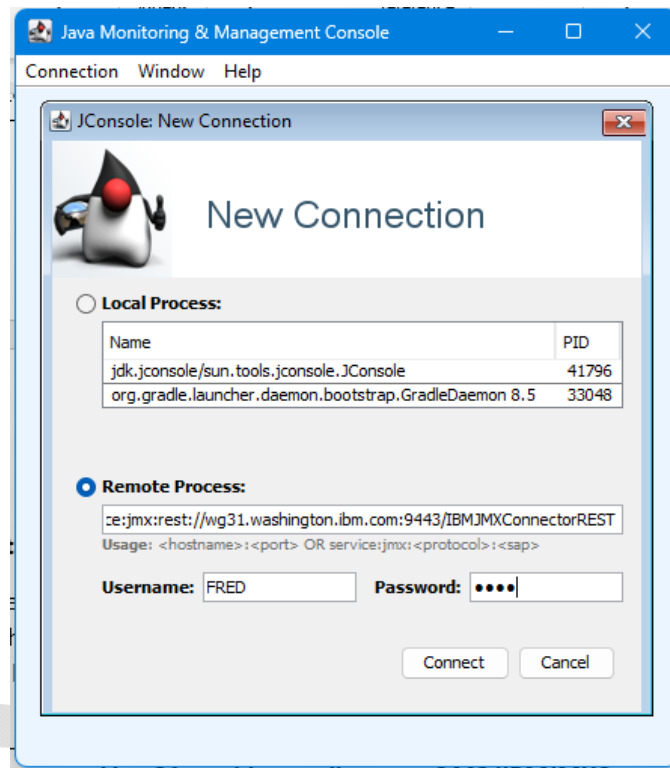
#4: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 98 34 67 0E 58 B8 D4 C6    15 16 39 53 E3 05 E6 6F    .4g.X.....9S...o
0010: 60 1F 97 77                                `..w
]
]

Trust this certificate? [no]: yes
Certificate was added to keystore
[Storing myTest.jks]
```

Monitoring a z/OS Liberty server

Using the JConsole

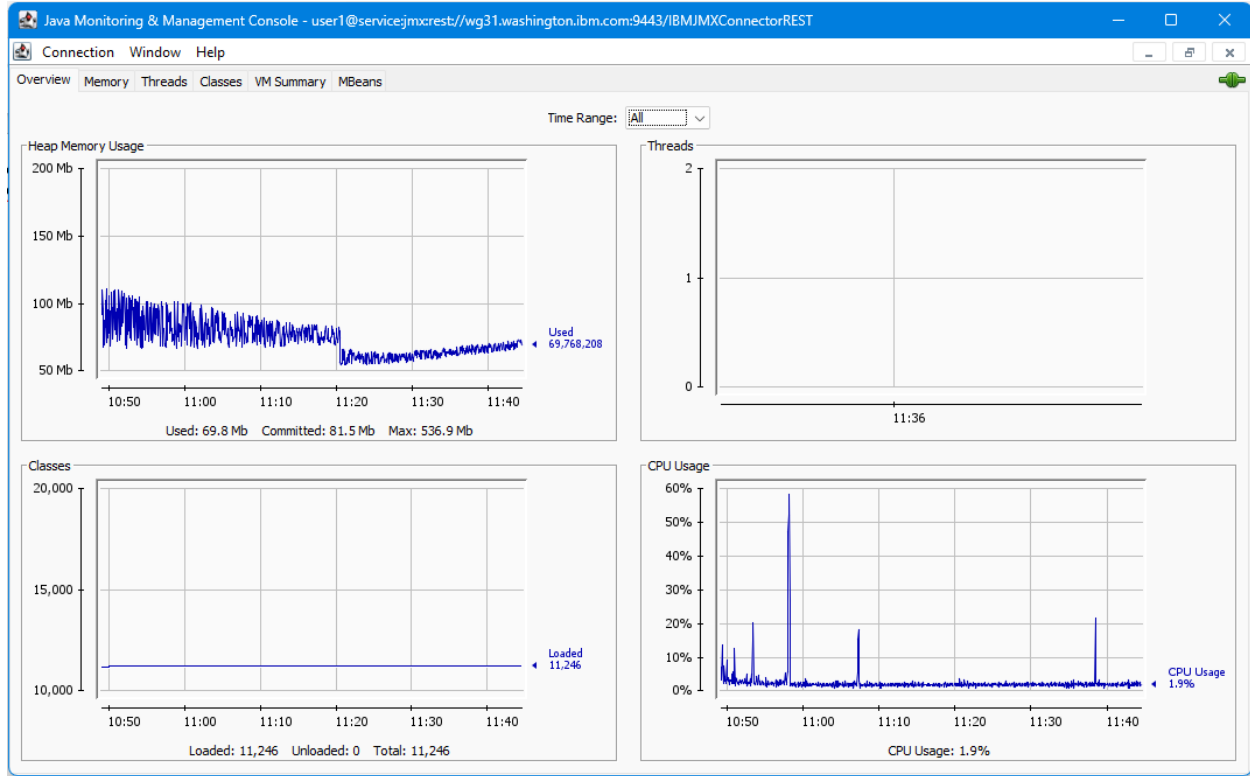
Invoked the command file created earlier to start the JConsole GUI and entered the service endpoint value obtained above for the *Remote Process* endpoint on the *JConsole: New Connection* window and enter an identity and password of a user who has READ access to the one of the EJBRole resources (Administrator or Reader).



Monitoring a z/OS Liberty server

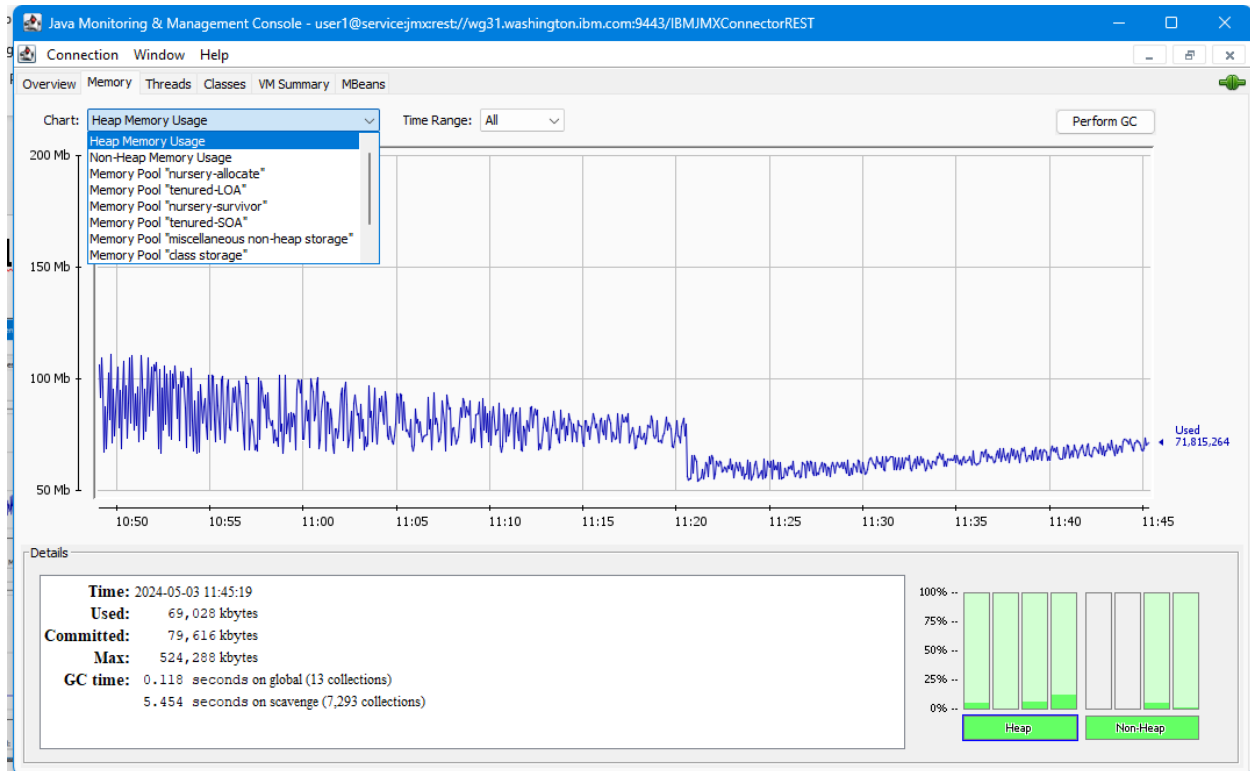
Exploring the information available in the JConsole tabs

- The initial screen contains graphs for *Heap Memory Usage*, *Threads*, *Classes*, and *CPU Usage* which will be continuously updated.



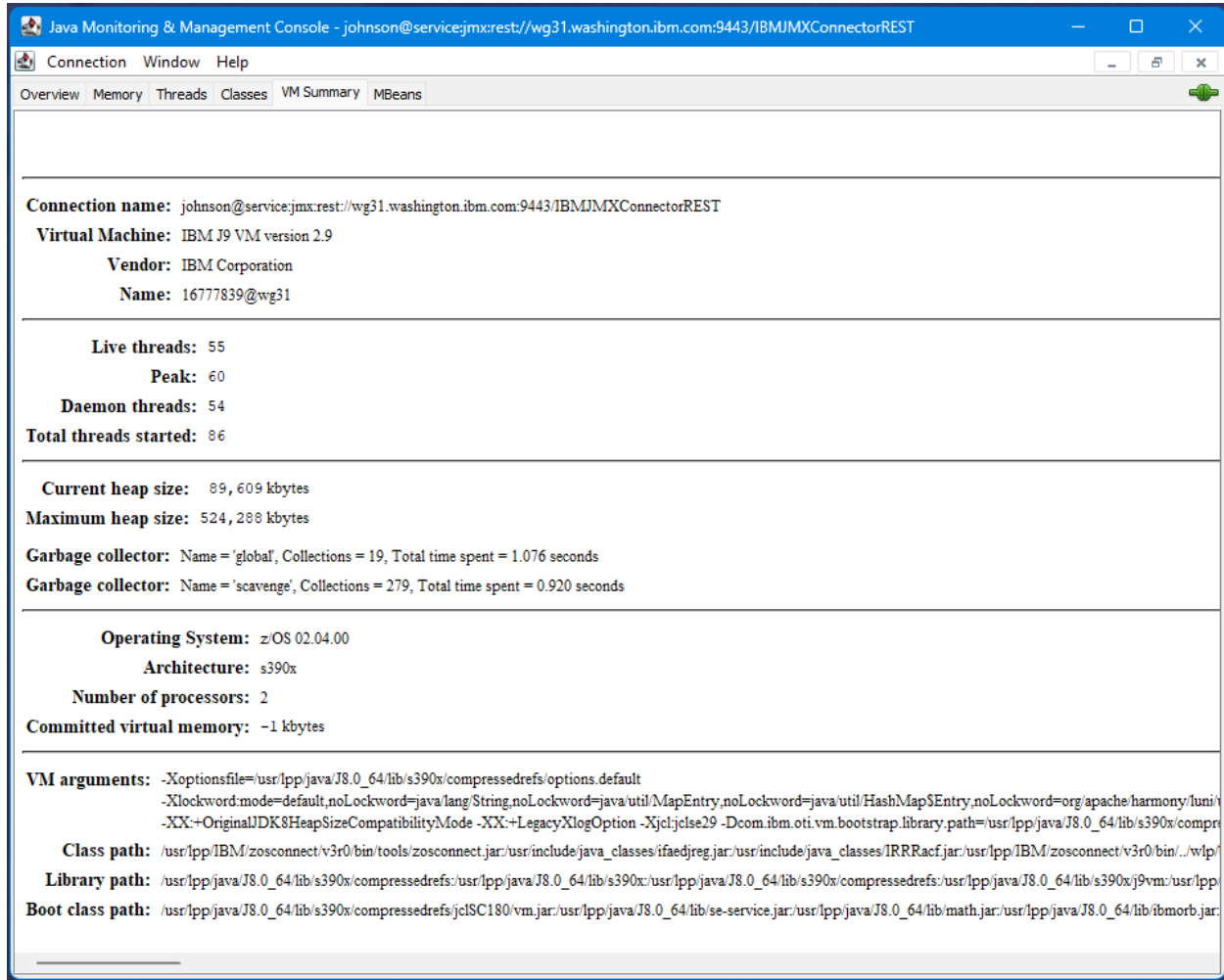
Monitoring a z/OS Liberty server

- Go to the *Memory* tab and use the pull-down arrow to select *Heap Memory Usage*. Note the details will be displayed at the bottom of the pane. The details at the bottom will change when switching between the various memory options. Go back to *Heap Memory Usage* and press the **Perform GC** button and note how the heap memory usage changes in both the chart and details pane.



Monitoring a z/OS Liberty server

- Go to the *VM Summary* tab and review the information available on this window. Information regarding number of thread history, the current and maximum heaps size, details regarding garbage collections and details about the operating system and number of processors will be displayed.



The screenshot displays the Java Monitoring & Management Console window. The title bar indicates the connection to 'johnson@servicejmxrest://wg31.washington.ibm.com:9443/IBMJMXConnectorREST'. The 'VM Summary' tab is selected, showing various system metrics. The console is divided into several sections: Connection details, Thread statistics, Heap and Garbage Collector information, Operating System details, and VM arguments.

Connection name: johnson@servicejmxrest://wg31.washington.ibm.com:9443/IBMJMXConnectorREST
Virtual Machine: IBM J9 VM version 2.9
Vendor: IBM Corporation
Name: 16777839@wg31

Live threads: 55
Peak: 60
Daemon threads: 54
Total threads started: 86

Current heap size: 89,609 kbytes
Maximum heap size: 524,288 kbytes

Garbage collector: Name = 'global', Collections = 19, Total time spent = 1.076 seconds
Garbage collector: Name = 'scavenge', Collections = 279, Total time spent = 0.920 seconds

Operating System: z/OS 02.04.00
Architecture: s390x
Number of processors: 2
Committed virtual memory: -1 kbytes

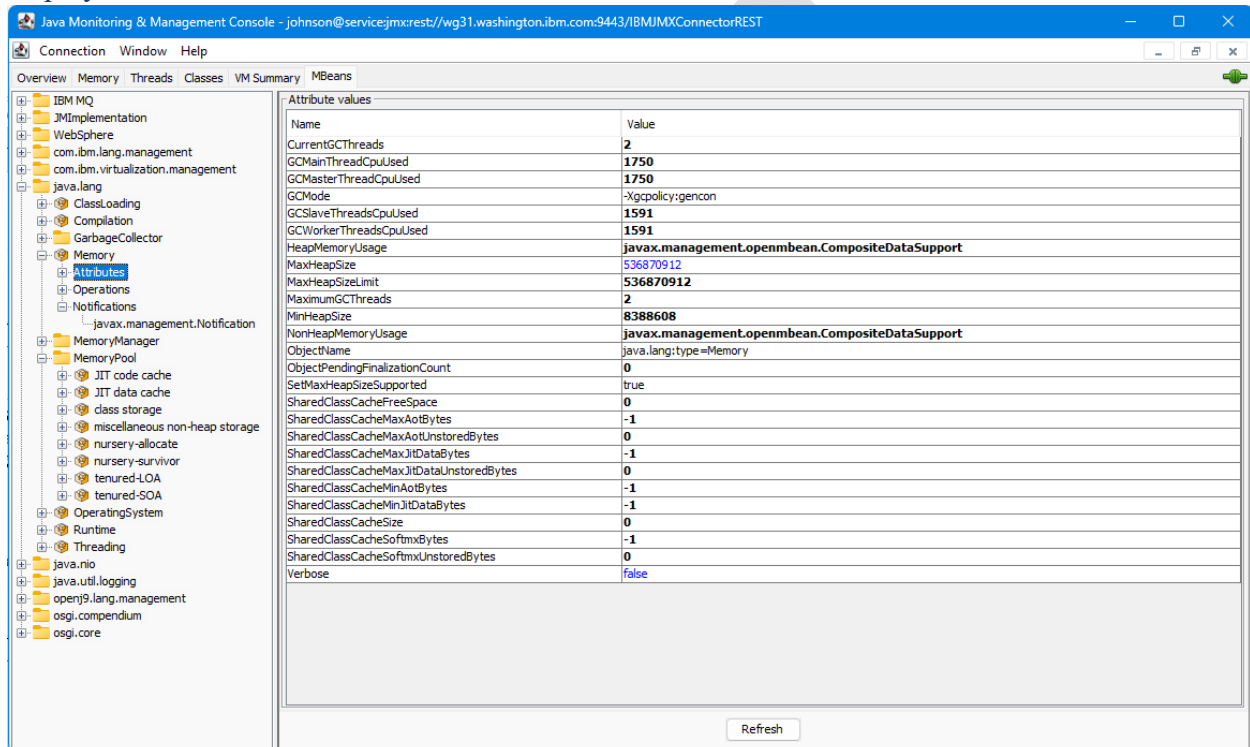
VM arguments: -Xoptionsfile=/usr/lpp/java/J8.0_64/lib/s390x/compressedrefs/options.default
-Xlockword.mode=default,noLockword=java/lang/String,noLockword=java/util/MapEntry,noLockword=java/util/HashMap\$Entry,noLockword=org/apache/harmony/luni/v
-XX:+OriginalJDK8HeapSizeCompatibilityMode -XX:+LegacyXlogOption -Xjcl:jclse29 -Dcom.ibm.oti.vm.bootstrap.library.path=/usr/lpp/java/J8.0_64/lib/s390x/compre
Class path: /usr/lpp/IBM/zosconnect/v3r0/bin/tools/zosconnect.jar:/usr/include/java_classes/ifaedjreg.jar:/usr/include/java_classes/IRRRacf.jar:/usr/lpp/IBM/zosconnect/v3r0/bin/.../wlp/...
Library path: /usr/lpp/java/J8.0_64/lib/s390x/compressedrefs:/usr/lpp/java/J8.0_64/lib/s390x:/usr/lpp/java/J8.0_64/lib/s390x/compressedrefs:/usr/lpp/java/J8.0_64/lib/s390x/j9vm:/usr/lpp/...
Boot class path: /usr/lpp/java/J8.0_64/lib/s390x/compressedrefs/jclSC180/vm.jar:/usr/lpp/java/J8.0_64/lib/se-service.jar:/usr/lpp/java/J8.0_64/lib/math.jar:/usr/lpp/java/J8.0_64/lib/ibmorb.jar:

Monitoring a z/OS Liberty server

Explore the Liberty server's available MBeans

The *MBeans* tab will display the JMX message beans available in this server. The list of MBeans is built when the JConsole client connects to the Liberty server. The list of MBeans will vary based on the configuration of the server. Most of the MBeans are provided as part of the base Java platform or as part of the enhancements added by IBM Java. The MBeans provided by Liberty are in the WebSphere folder.

- In general, selecting an MBeans' *Attributes* property displays the current values associated with that MBean. For example, selecting the attributes for the *java.lang:type=Memory* Mbean displays these attribute values:



The screenshot shows the Java Monitoring & Management Console window. The left pane displays a tree view of MBeans, with the *Attributes* property selected under the *Memory* MBean. The right pane displays a table of attribute values for the *java.lang:type=Memory* MBean.

Name	Value
CurrentGCThreads	2
GCMainThreadCpuUsed	1750
GCMasterThreadCpuUsed	1750
GCMode	-Xgcpolicy:gencon
GCSlaveThreadsCpuUsed	1591
GCWorkerThreadsCpuUsed	1591
HeapMemoryUsage	javax.management.openmbean.CompositeDataSupport
MaxHeapSize	536870912
MaxHeapSizeLimit	536870912
MaximumGCThreads	2
MinHeapSize	8388608
NonHeapMemoryUsage	javax.management.openmbean.CompositeDataSupport
ObjectName	java.lang:type=Memory
ObjectPendingFinalizationCount	0
SetMaxHeapSizeSupported	true
SharedClassCacheFreeSpace	0
SharedClassCacheMaxAotBytes	-1
SharedClassCacheMaxAotUnstoredBytes	0
SharedClassCacheMaxJitDataBytes	-1
SharedClassCacheMaxJitDataUnstoredBytes	0
SharedClassCacheMinAotBytes	-1
SharedClassCacheMinJitDataBytes	-1
SharedClassCacheSize	0
SharedClassCacheSoftmxBytes	-1
SharedClassCacheSoftmxUnstoredBytes	0
Verbose	false

Refresh

Monitoring a z/OS Liberty server

• Note: the units of the values displayed on screens like the one above for *MaxHeapSize* are in bits. These numbers can be converted using online tools like the one available at URL <https://bit-calculator.com/bit-calculator>. For example, the value for 536870912 bits can be converted as shown below:

0000
1100
0101

Bit Calculator

Bit Calculator

Bit Shift Calculator

Binary Calculator

Number Base Converter

Articles

Bit Calculator

Make binary conversion between different units

Quantity

536870912

Unit

bit - b

☐ Network (Kilo = 1000) ☒ Storage (Kilo = 1024)

Storage, K = 1024

bit - b	536870912
Byte - B	67108864
Kilobit - Kb	524288
Kilobyte - KB	65536
Megabit - Mb	512
Megabyte - MB	64
Gigabit - Gb	0.5
Gigabyte - GB	0.0625
Terabit - Tb	0.00048828125
Terabyte - TB	0.00006103515625
Petabit - Pb	4.76837158203125e-7
Petabyte - PB	5.960464477539063e-8
Exabit - Eb	4.656612873077393e-10
Exabyte - EB	5.820766091346741e-11
Zettabit - Zb	4.547473508864641e-13
Zettabyte - ZB	5.684341886080802e-14
Yottabit - Yb	4.440892098500626e-16
Yottabyte - YB	5.551115123125783e-17

Monitoring a z/OS Liberty server

- Expanding an MBean's *Operations* displays a list of functions that MBean can perform. For example, selecting the *operations* for MBean *com.ibm.virtualization.management.GuestOS* displays these operations. Selecting operations *retrieveMemoryUsage* and clicking on *retrieveMemoryUsage* beside *CompositeData* on the *Operation invocation* page retrieves the results below:

The screenshot shows the Java Monitoring & Management Console interface. On the left, the MBean tree is expanded to show the *com.ibm.virtualization.management.GuestOS* MBean, with the *retrieveMemoryUsage* operation selected. The main panel displays the *Operation invocation* page for *retrieveMemoryUsage*. Below this, the *MBeanOperationInfo* table is shown:

Name	Value
Operation:	
Name	retrieveMemoryUsage
Description	retrieveMemoryUsage
Impact	UNKNOWN
ReturnType	javax.management.openmbean.CompositeData

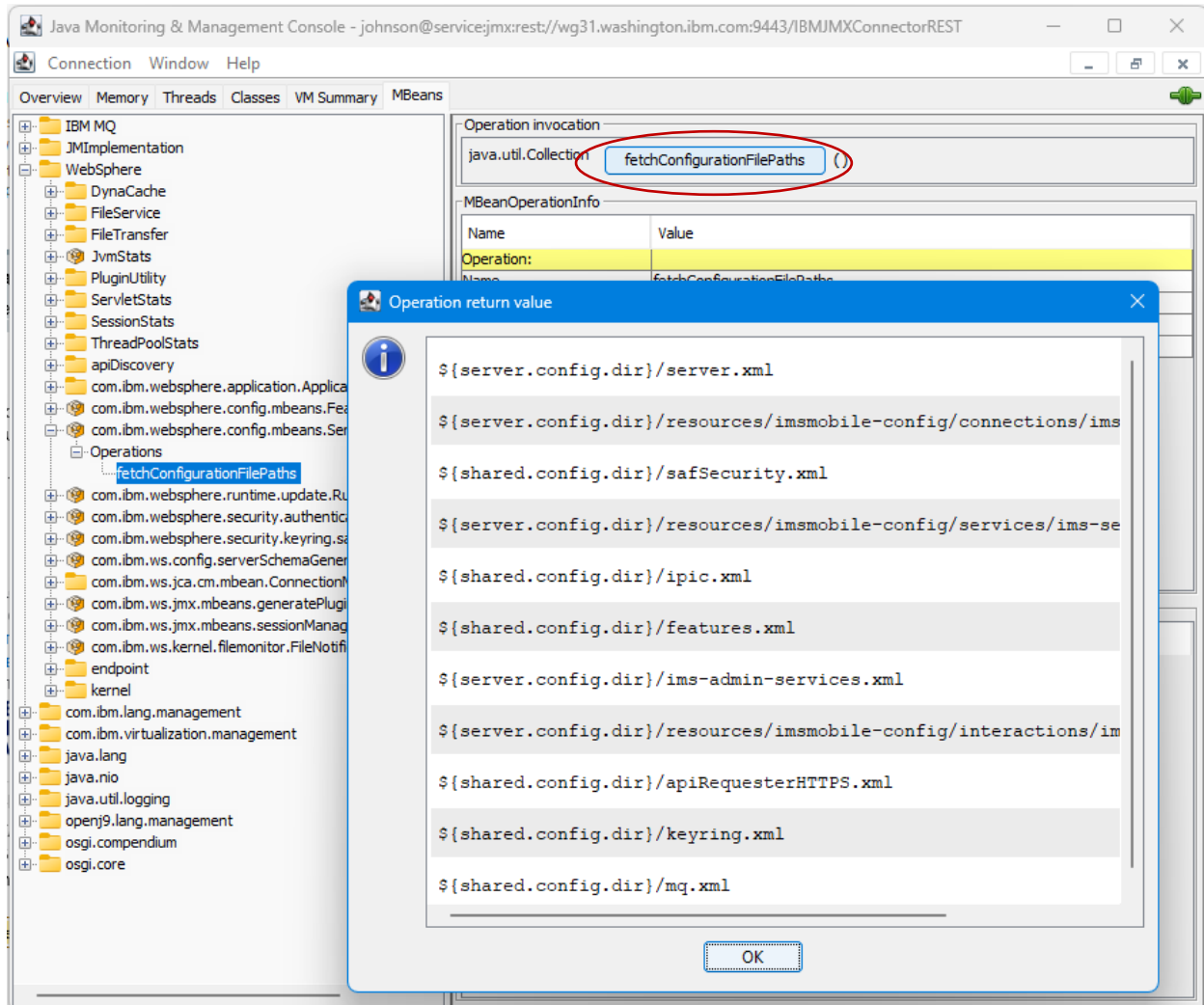
An *Operation return value* dialog box is open, displaying the results of the *retrieveMemoryUsage* operation. The dialog shows a table with the following data:

Name	Value
maxMemLimit	8589934592
memUsed	5863325696
timestamp	1715972210905632

The dialog also includes navigation controls for *Tabular Data Navigation* and *Composite Data Navigation*, and an *OK* button.

Monitoring a z/OS Liberty server

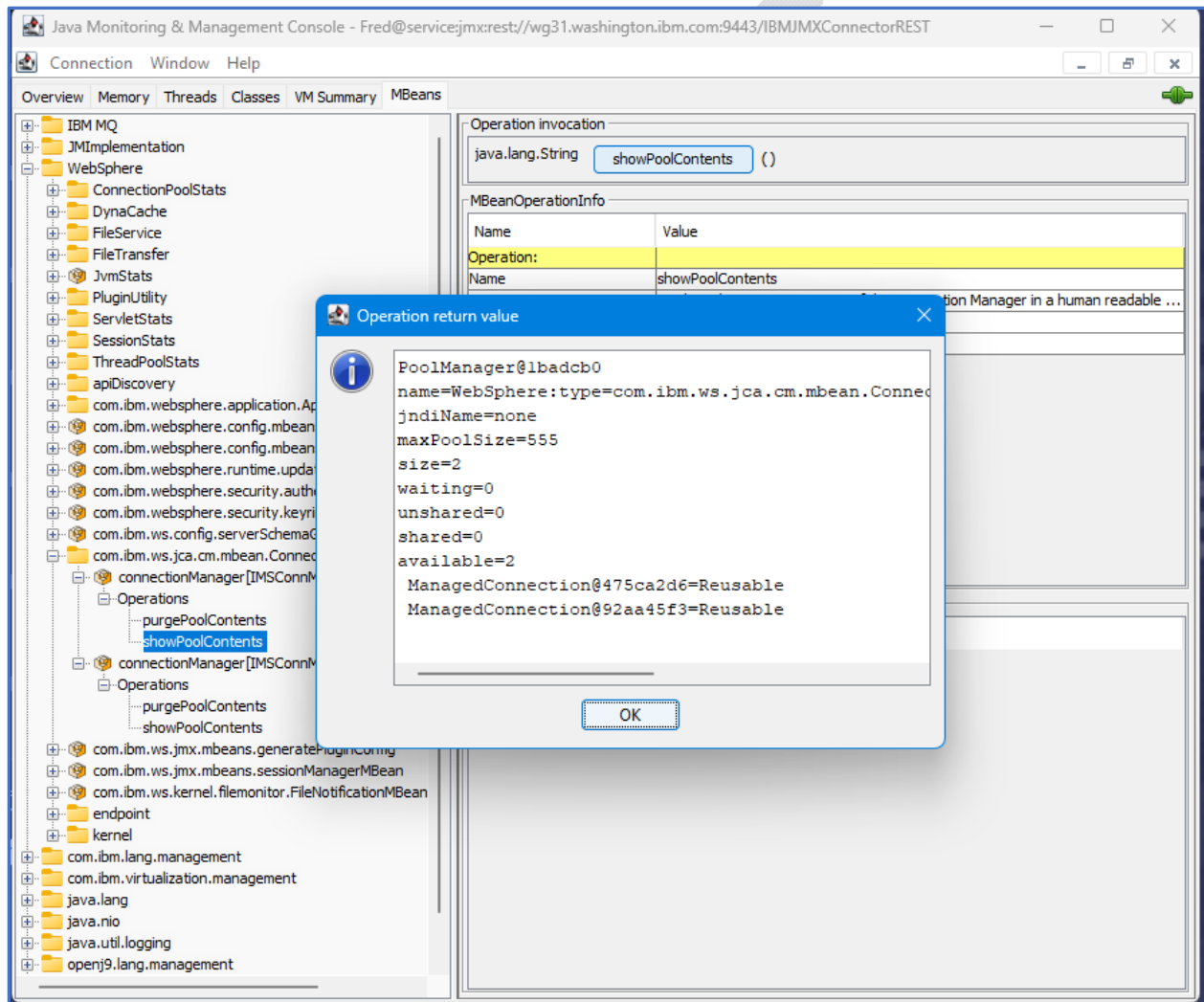
- Go to the *MBeans* tab and expand the *WebSphere* folder and then expand the *com.ibm.websphere.config.MBeans.ServerXMLConfigurationMBean* MBean. Expand *Operations* and then select operation *fetchConfigurationFilePaths*. On the *Operation invocation* pane, click operation area containing the string *fetchConfigurationFilePaths* beside *java.util.Collection*. This will invoke this MBean and display the results in a pop-up window as shown below:



This displays the files used to configure the server.

Monitoring a z/OS Liberty server

- Expand the *WebSphere* folder and then expand the *com.ibm.ws.jca.cm.MBean.ConnectionManagerMBean* MBean. Note that this MBean will only appear if there is one or more connection managers configured in the server XML. Expand a connection manager configuration, e.g., *connectionManager[IMSConnMgr1]* and then then expand the *Operations* under that connection manager. There will be two operations available for a specific connection manger, *purgePoolContents* and *showPoolContents*. Select *showPoolContents*. On the *Operation invocation* pane, click the area containing the string *showPoolContents* beside *java.util.String* This will invoke this MBean and display the results in a pop-up window as shown below:



This is showing you real time status of the connection pool.

Monitoring a z/OS Liberty server

- On the *MBeans* tab, expand the *java.lang* folder and then expand the *Memory* MBean. Select *Attributes* and this will display the current values of memory related properties, i.e. *GCMODE*, *MaxHeapSize*, *MaxHeapSize Limit*, and *MinHeapSize*.

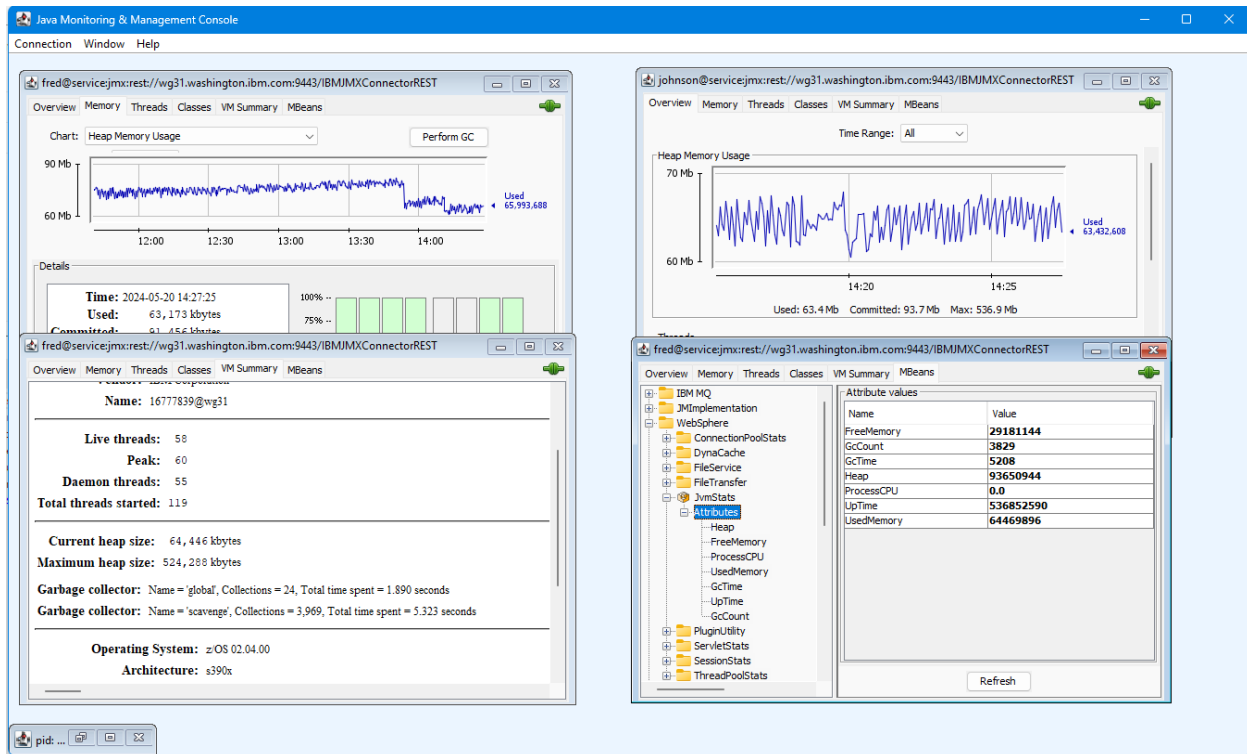
The screenshot shows the Java Monitoring & Management Console interface. The left sidebar displays a tree view of MBeans, with the *java.lang* folder expanded, showing *Memory* and *Attributes* selected. The main panel displays the *Attribute values* table for the *Memory* MBean.

Name	Value
CurrentGCThreads	2
GCMainThreadCpuUsed	1730
GCMasterThreadCpuUsed	1730
GCMODE	-Xgcpolicy:gencon
GCSlaveThreadCpuUsed	1584
GCWorkerThreadCpuUsed	1584
HeapMemoryUsage	javax.management.openmbean.CompositeDataSupport
MaxHeapSize	536870912
MaxHeapSizeLimit	536870912
MaximumGCThreads	2
MinHeapSize	8388608
NonHeapMemoryUsage	javax.management.openmbean.CompositeDataSupport
ObjectName	java.lang:type=Memory
ObjectPendingFinalizationCount	0
SetMaxHeapSizeSupported	true
SharedClassCacheFreeSpace	0
SharedClassCacheMaxAotBytes	-1
SharedClassCacheMaxAotUnstoredBytes	0
SharedClassCacheMaxJitDataBytes	-1
SharedClassCacheMaxJitDataUnstoredBytes	0
SharedClassCacheMinAotBytes	-1
SharedClassCacheMinJitDataBytes	-1
SharedClassCacheSize	0
SharedClassCacheSoftmxBytes	-1
SharedClassCacheSoftmxUnstoredBytes	0
Verbose	false

Refresh

Monitoring a z/OS Liberty server

- Note that multiple connections can be established to the same Liberty server on z/OS from one instance of the JConsole GUI. This gives one the ability to monitor and manage multiple areas at the same time.



Monitoring a z/OS Liberty server

Using the IBM API Explorer to access MBeans using REST

Now access the Liberty MBeans from a REST client. Once such client is the *IBM API Explorer* which was added to the Liberty server by adding feature *apiDiscovery-1.0*. This enabled REST access to my server from a web browser.

- Begin by accessing the server's *IBM API Explorer* web page by using URL, e.g., <https://wg31.washington.ibm.com:9443/ibm/api/explorer/>. Use the *List Operation* option to list the operations in **IBMJMXConnectorREST:JMX REST Connector MBeans and File Transfer APIs**.

The screenshot shows the IBM API Explorer web interface in a browser. The URL bar displays <https://wg31.washington.ibm.com:9443/ibm/api/explorer/#/>. The page title is "Liberty REST APIs" with the subtitle "Discover REST APIs available within Liberty". Below this, there is a section for "API Discovery : APIs available from the API Discovery feature" with links for "Show/Hide", "List Operations", and "Expand Operations". The main content area is titled "IBMJMXConnectorREST : JMX REST Connector MBeans and File Transfer APIs" and lists various REST endpoints. Each entry includes a method (GET, POST, DELETE), the endpoint path, and a brief description. For example, the first entry is a GET request to `/IBMJMXConnectorREST/defaultDomain` to "Get the default domain". Other entries include endpoints for domains, file collections, file status, and MBean information.

Method	Endpoint	Description
GET	<code>/IBMJMXConnectorREST/defaultDomain</code>	Get the default domain
GET	<code>/IBMJMXConnectorREST/domains</code>	Get the list of domains
POST	<code>/IBMJMXConnectorREST/file/collection</code>	Delete multiple files within a single REST invocation.
GET	<code>/IBMJMXConnectorREST/file/status</code>	Retrieves multiple status, with the option of filtering the results
GET	<code>/IBMJMXConnectorREST/file/status/{taskId}</code>	Retrieve status information about a specific task.
GET	<code>/IBMJMXConnectorREST/file/status/{taskId}/hosts</code>	Retrieve status information about the hosts of a specific task.
GET	<code>/IBMJMXConnectorREST/file/status/{taskId}/hosts/{hostName}</code>	Retrieve a list of step details that were taken within a specific host in a specific task.
GET	<code>/IBMJMXConnectorREST/file/status/{taskId}/properties</code>	Retrieves a list of available properties from a specific task.
GET	<code>/IBMJMXConnectorREST/file/status/{taskId}/properties/{property}</code>	Fetch the value of a property in a specific task.
DELETE	<code>/IBMJMXConnectorREST/file/{filePath}</code>	Delete a remote file.
GET	<code>/IBMJMXConnectorREST/file/{filePath}</code>	Download the contents of the specified remote file
POST	<code>/IBMJMXConnectorREST/file/{filePath}</code>	Upload a file to a remote location.
GET	<code>/IBMJMXConnectorREST/instanceOf</code>	MBean specific is instanceOf specified class
GET	<code>/IBMJMXConnectorREST/mbeanCount</code>	Get the count of MBeans
GET	<code>/IBMJMXConnectorREST/mbeans</code>	Retrieves list of MBeans

Monitoring a z/OS Liberty server

When using *IBM API Explorer* focus on these selections.

GET	/IBMjMXConnectorREST/mbeans	Retrieves list of MBeans
POST	/IBMjMXConnectorREST/mbeans	Retrieves list of MBeans by filtered Query Expression
POST	/IBMjMXConnectorREST/mbeans/factory	Create MBean
DELETE	/IBMjMXConnectorREST/mbeans/{objectName}	Deletes the MBean
GET	/IBMjMXConnectorREST/mbeans/{objectName}	Returns all endpoints
GET	/IBMjMXConnectorREST/mbeans/{objectName}/attributes	Retrieves MBean values of several attributes
POST	/IBMjMXConnectorREST/mbeans/{objectName}/attributes	Sets MBean value attributes
GET	/IBMjMXConnectorREST/mbeans/{objectName}/attributes/{attribute}	Gets the value of a specific attribute of a named MBean.
PUT	/IBMjMXConnectorREST/mbeans/{objectName}/attributes/{attribute}	Sets the value of a specific attribute of a named MBean.
POST	/IBMjMXConnectorREST/mbeans/{objectName}/operations/{operation}	Invokes an operation on an MBean.

As you see there are REST APIs provide for obtaining a list of MBeans (GET). REST APIs for getting(GET) and setting(POST) an MBean's attributes. And a REST API for invoking (POST) and MBean's operation. The other REST APIs are interesting but beyond scope of the document.

Monitoring a z/OS Liberty server

- I needed to display a list of the available MBeans. Select the *GET* method beside */IBMJMXConnectorREST/mbeans* to expose the interface as shown below. Note that you could have display a specific MBean by entering the MBean's object name or class name. But now we want to display them all so leave both fields empty.

GET

/IBMJMXConnectorREST/mbeans

Retrieves list of MBeans

Implementation Notes

Retrieve a list of MBeans that match the specified ObjectName and ClassName

Response Class (Status 200)

successful operation

Model

Example Value

```
[
  {
    "objectName": "string",
    "className": "string",
    "URL": "string"
  }
]
```

Response Content Type

application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
objectName	<input type="text"/>	Name of the MBean object	query	string
className	<input type="text"/>	Name of the class	query	string
com.ibm.websphere.jmx.connector.rest.routing.hostName	<input type="text"/>	Host name for the server level routing	header	string
com.ibm.websphere.jmx.connector.rest.routing.serverName	<input type="text"/>	Server name for the server level routing	header	string
com.ibm.websphere.jmx.connector.rest.routing.serverUserDir	<input type="text"/>	Server user directory for the server level routing	header	string

Try it out!

[Hide Response](#)

Monitoring a z/OS Liberty server

- Pressing the **Try it out!** button will invoke this MBean and display the results in the *Response Body* section.

[Try it out!](#) [Hide Response](#)

Curl

```
curl -X GET --header 'Accept: application/json' 'https://wg31.washington.ibm.com:9443/IBMJMXConnectorREST/mbeans'
```

Request URL

```
https://wg31.washington.ibm.com:9443/IBMJMXConnectorREST/mbeans
```

Response Body

```
{
  "objectName": "WebSphere:type=com.ibm.ws.jca.cm.mbean.ConnectionManagerMBean,name=connectionManager[IMSConnMgr2]",
  "className": "com.ibm.ws.jca.cm.mbean.ConnectionManagerMBean",
  "URL": "/IBMJMXConnectorREST/mbeans/WebSphere%3Aname%3DconnectionManager%5BIMSConnMgr2%5D%2Ctype%3Dcom.ibm.ws.jca.cm.mbean.ConnectionManagerMBean"
},
{
  "objectName": "WebSphere:feature=apiDiscovery,name=APIDiscovery",
  "className": "com.ibm.ws.nest.api.discovery.internal.mbean.APIDiscovery",
  "URL": "/IBMJMXConnectorREST/mbeans/WebSphere%3Afeature%3DApiDiscovery%2Cname%3DAPIDiscovery"
},
{
  "objectName": "osgi.core:type=framework,version=1.7,framework=org.eclipse.osgi,uuid=63a83d8a-56db-4a3b-832a-80a2bce38eb5",
  "className": "org.apache.aries.jmx.framework.Framework",
  "URL": "/IBMJMXConnectorREST/mbeans/osgi.core%3Aframework%3Dorg.eclipse.osgi%2Ctype%3Dframework%2Cuuid%3D63a83d8a-56db-4a3b-832a-80a2bce38eb5"
},
{
  "objectName": "WebSphere:name=com.ibm.ws.config.serverSchemaGenerator",
  "className": "com.ibm.ws.config.schemagen.internal.ServerSchemaGeneratorImpl",
  "URL": "/IBMJMXConnectorREST/mbeans/WebSphere%3Aname%3Dcom.ibm.ws.config.serverSchemaGenerator"
},
{
  "objectName": "WebSphere:type=ThreadPoolState,name=DefaultExecutor"
}
```


Monitoring a z/OS Liberty server

- Next obtain details about this MBean, e.g., operations, attributes, etc. locate the GET method for Mbean `/IBMJMXConnectorREST/mbeans/{objectName}` to expose this Mbean's interface. Enter the *objectName* for the IMSConnMgr2 connection pool MBean, `WebSphere:type=com.ibm.ws.jca.cm.mbean.ConnectionManagerMBean,name=connectionManager[IMSConnMgr2]`.

Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
objectName	WebSphere:type=com.ibm.ws.jca.cm.mbean	Name of the MBean object	path	string
com.ibm.websphere.jmx.connector.rest.routing.hostName		Host name for the server level routing	header	string
com.ibm.websphere.jmx.connector.rest.routing.serverName		Server name for the server level routing	header	string
com.ibm.websphere.jmx.connector.rest.routing.serverUserDir		Server user directory for the server level routing	header	string

[Try it out!](#) [Hide Response](#)

- Pressing the **Try it Out** button provided the results below in the *Response Body*. Scrolling down, locate the `showPoolContents` operation, as well as the URL (URIPath) for use when accessing this operation from a REST client.

Response Body

```
VALUES: []
}
}
},
"URL": "/IBMJMXConnectorREST/mbeans/WebSphere%3DconnectionManager%5BIMSConnMgr2%5D%2Ctype%3Dcom.ibm.ws.jca.cm.mbean.ConnectionManagerMBean/operations/purgePoolContents"
},
{
  "name": "showPoolContents",
  "description": "Displays the current contents of the Connection Manager in a human readable format.",
  "descriptor": {
    "names": [],
    "values": []
  },
  "impact": "0",
  "returnType": "java.lang.String",
  "signature": [],
  "URL": "/IBMJMXConnectorREST/mbeans/WebSphere%3DconnectionManager%5BIMSConnMgr2%5D%2Ctype%3Dcom.ibm.ws.jca.cm.mbean.ConnectionManagerMBean/operations/showPoolContents"
}
]
}
```

Monitoring a z/OS Liberty server

To invoke an operation of an MBean requires the use of a POST to URI path for MBean `/IBMJMXConnectorREST/mbeans/{objectName}/operations/{operations}`

GET	/IBMJMXConnectorREST/mbeans	Retrieves list of MBeans
POST	/IBMJMXConnectorREST/mbeans	Retrieves list of MBeans by filtered Query Expression
POST	/IBMJMXConnectorREST/mbeans/factory	Create MBean
DELETE	/IBMJMXConnectorREST/mbeans/{objectName}	Deletes the MBean
GET	/IBMJMXConnectorREST/mbeans/{objectName}	Returns all endpoints
GET	/IBMJMXConnectorREST/mbeans/{objectName}/attributes	Retrieves MBean values of several attributes
POST	/IBMJMXConnectorREST/mbeans/{objectName}/attributes	Sets MBean value attributes
GET	/IBMJMXConnectorREST/mbeans/{objectName}/attributes/{attribute}	Gets the value of a specific attribute of a named MBean.
PUT	/IBMJMXConnectorREST/mbeans/{objectName}/attributes/{attribute}	Sets the value of a specific attribute of a named MBean.
POST	/IBMJMXConnectorREST/mbeans/{objectName}/operations/{operation}	Invokes an operation on an MBean.

- Locate the POST method for this Mbean and exposed this Mbean's interface. Enter the MBean's object name, *WebSphere:type=com.ibm.ws.jca.cm.mbean.ConnectionManagerMBean,name=connectionManager[IMSConnMgr2]*, along with the name of the *operations*, *showPoolContents*, in the areas beside *objectName* and *operations* respectively. Since this was a POST request, a simple JSON message consisting of a beginning and ending braces, {}, is required in the *JSON Request* area.

Parameter	Value	Description	Parameter Type	Data Type
JSON Request	<div><div>{ }</div><div>Parameter content type: application/json</div></div>	JSON Request	body	Model Example Value
objectName	n,name=connectionManager[IMSConnMgr2]	Name of the MBean object	path	string
operation	showPoolContents	Name of the specific operation	path	string
com.ibm.websphere.jmx.connector.rest.routing.hostName		Host name for the server level routing	header	string
com.ibm.websphere.jmx.connector.rest.routing.serverName		Server name for the server level routing	header	string
com.ibm.websphere.jmx.connector.rest.routing.serverUserDir		Server user directory for the server level routing	header	string

[Try it out](#) [Hide Response](#)

Monitoring a z/OS Liberty server

- Pressing the **Try it Out** button invoked this operation provided the results showing the connection pool status in the *Response Body* as shown below.

[Try it out!](#) [Hide Response](#)

Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{}' 'https://wg31.washington.ibm.com:9443/IBMJMXConnectorREST/mbeans/WebSphere%3Atype%3Acom.ibm.ws.jca.cm.mbean.ConnectionManagerMBean%2Cname%3DconnectionManager%5BIMSConnMgr%2C%5D/operation'
```

Request URL

```
https://wg31.washington.ibm.com:9443/IBMJMXConnectorREST/mbeans/WebSphere%3Atype%3Acom.ibm.ws.jca.cm.mbean.ConnectionManagerMBean%2Cname%3DconnectionManager%5BIMSConnMgr%2C%5D/operation'
```

Response Body

```
are:type=com.ibm.ws.jca.cm.mbean.ConnectionManagerMBean,name=connectionManager-IMSConnMgr21\njndiName=none\nmaxPoolSize=155\nsize=0\nwaiting=0\nunshared=0\nshared=0\navailable=0\n"
```

Response Code

```
200
```

This is same information displayed by JConsole.



Monitoring a z/OS Liberty server

Accessing MBean ServerXMLConfigurationMBean

Next, display the server XML configuration files. Use the same process as before, that is starting with the list of MBeans, identifying the Mbean's *objectName* of *WebSphere:name=com.ibm.websphere.config.mbeans.ServerXMLConfigurationMBean* and operation name of *fetchConfigurationFilePath*.

Note that invoking this MBean required the use of the POST method and the addition of a simple JSON message with a beginning and ending braces, {}.

Parameter	Value	Description	Parameter Type	Data Type
JSON Request	<div><div>{}</div><div>Parameter content type: application/json</div></div>	JSON Request	body	Model Example Value
objectName	com.ibm.websphere.config.mbeans.ServerXMLConfigurationMBean	Name of the MBean object	path	string
operation	fetchConfigurationFilePaths	Name of the specific operation	path	string
com.ibm.websphere.jmx.connector.rest.routing.hostName		Host name for the server level routing	header	string
com.ibm.websphere.jmx.connector.rest.routing.serverName		Server name for the server level routing	header	string
com.ibm.websphere.jmx.connector.rest.routing.serverUserDir		Server user directory for the server level routing	header	string

[Try it out!](#) [Hide Response](#)

With a response message of:

Response Body
<pre>{ "value": ["\${server.config.dir}/server.xml", "\${server.config.dir}/resources/ismobile-config/connections/ims-connections.xml", "\${shared.config.dir}/safSecurity.xml", "\${server.config.dir}/resources/ismobile-config/services/ims-services.xml", "\${shared.config.dir}/ipic.xml", "\${shared.config.dir}/features.xml", "\${server.config.dir}/ims-admin-services.xml", "\${server.config.dir}/resources/ismobile-config/interactions/ims-interactions.xml", "\${shared.config.dir}/apiRequesterHTTPS.xml", "\${shared.config.dir}/keyring.xml", "\${shared.config.dir}/mq.xml", "\${shared.config.dir}/web.xml"], "type": { "className": "java.util.HashSet", "items": ["java.lang.String", "java.lang.String"] } }</pre>
Response Code
200

Monitoring a z/OS Liberty server

- This same process, starting with the list of MBeans, identifying an Mbean's *objectName* and *attributes* can be used to the current value of key attributes, e.g. *heap*, *GcCount*, etc in the case of a *JvmStats* Mbean.

The screenshot shows a web browser window displaying the REST API documentation for the endpoint `/IBM/JMXConnectorREST/mbeans/{objectName}/attributes`. The page includes implementation notes, a response class (Status 200), and a table of parameters.

Implementation Notes
Retrieves the values of several attributes of a named MBean. 'value' is a POJO parameter. To get more information about declaring a POJO, do a GET request `/IBM/JMXConnectorREST/pojo`.

Response Class (Status 200)
Generated MBeans
Model Example Value

```
{
  "name": "string",
  "value": [
    {
      "value": "${server.output.dir}/server.xml",
      "type": "java.lang.string"
    }
  ]
}
```

Parameters

Parameter	Value	Description	Parameter Type	Data Type
objectName	<input type="text" value="WebSphere:type=JvmStats"/>	Name of the MBean object	path	string
attribute	<input type="text" value="heap"/>	Name of the attribute	query	Array[string]
<code>com.ibm.webSphere.jmx.connector.rest.routing.hostname</code>	<input type="text"/>	Host name for the server level routing	header	string
<code>com.ibm.webSphere.jmx.connector.rest.routing.servername</code>	<input type="text"/>	Server name for the server level routing	header	string
<code>com.ibm.webSphere.jmx.connector.rest.routing.serveruserdir</code>	<input type="text"/>	Server user directory for the server level routing	header	string

Try it out! [Hide Response](#)

Curl

```
curl -X GET --header 'Accept: application/json' 'https://wg31.washington.ibm.com:9443/IBM/JMXConnectorREST/mbeans/WebSphere:type=JvmStats/attributes?attribute=heap'
```

Request URL

```
https://wg31.washington.ibm.com:9443/IBM/JMXConnectorREST/mbeans/WebSphere:type=JvmStats/attributes?attribute=heap
```

Response Body

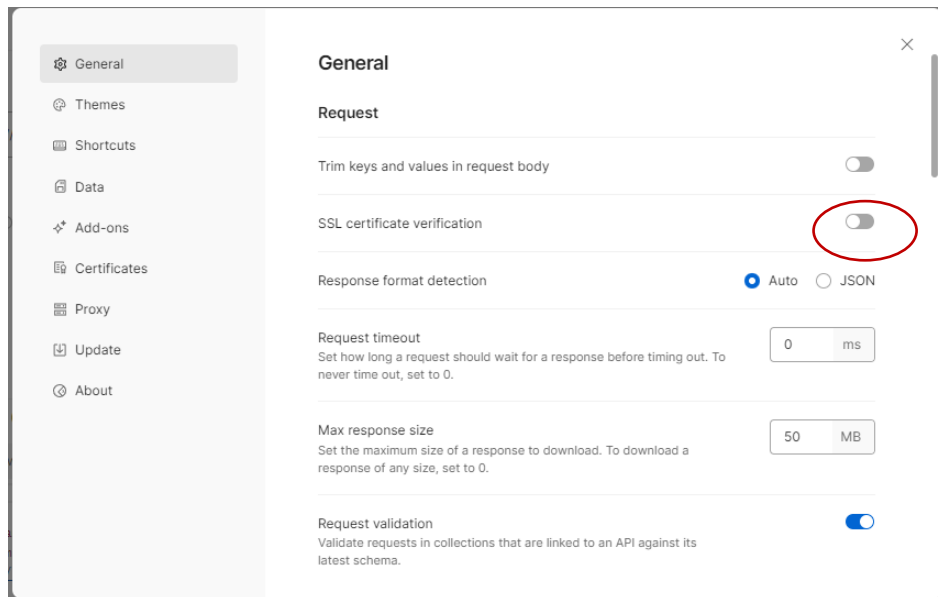
```
{
  "name": "heap",
  "value": [
    {
      "value": "88684672",
      "type": "java.lang.Long"
    }
  ]
}
```


Monitoring a z/OS Liberty server

Using Postman to access MBeans using REST

Another useful tool for accessing the Liberty MBeans from a REST client is Postman.

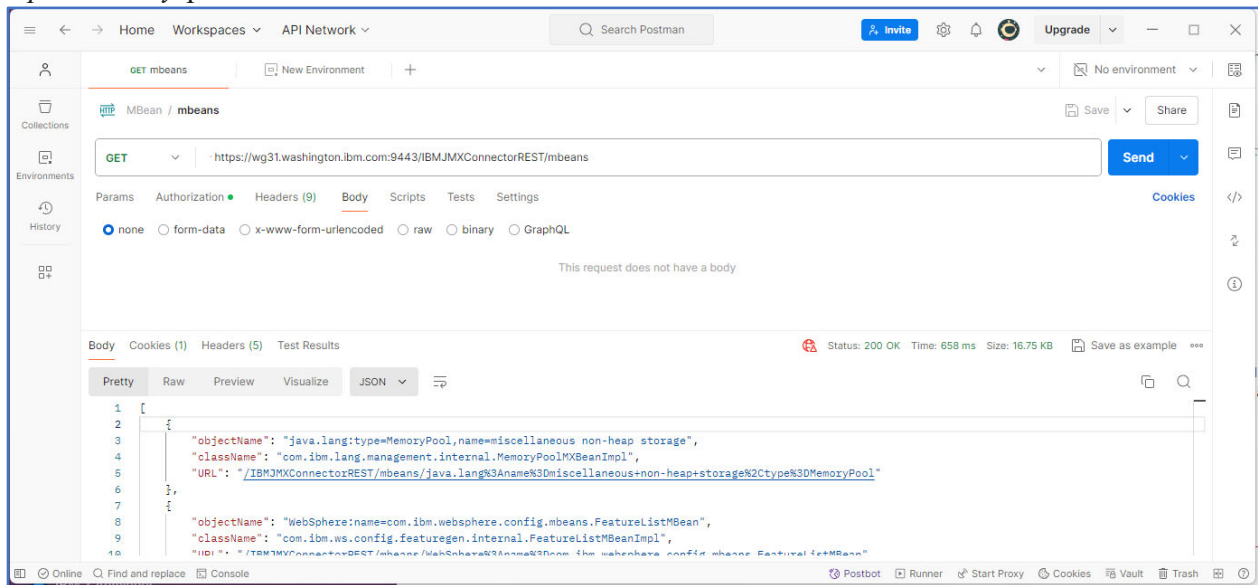
Some housekeeping is required before using Postman to access the Liberty MBeans. First, access the Postman *Settings* options using the sprocket on the toolbar and turned off *SSL certificate verification*. The same local key store used for JConsole could have been used but running with server verification is sufficient at this time.



- The next step is to provide some basic credentials. Select the *Authorization* tab and entered a valid *Username* and *Password*. This identity must have access to the RACF EJBRoles protecting the JMX and REST resources in the server.

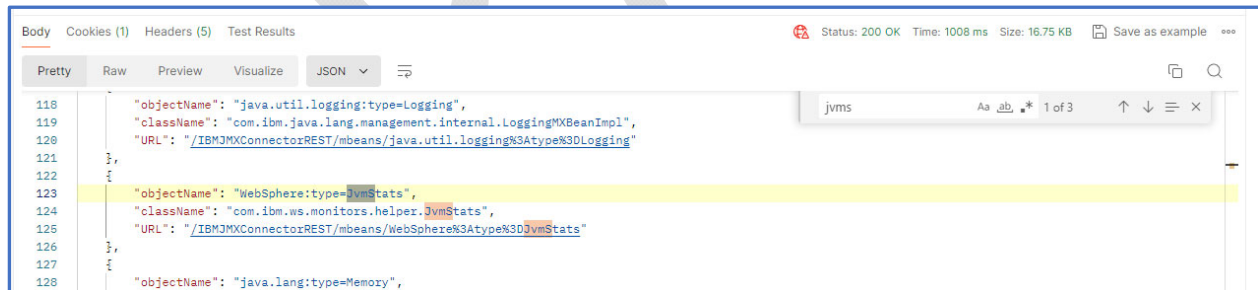
Monitoring a z/OS Liberty server

- Next select the *Body* tab and use the pull-down to select the **GET** method and enter this URL <https://wg31.washington.ibm.com:9443/IBMJMXConnectorREST/mbeans>. Pressing the **Send** button invoked this MBean and returned a list of the MBeans available at the server in the *Response Body* pane.



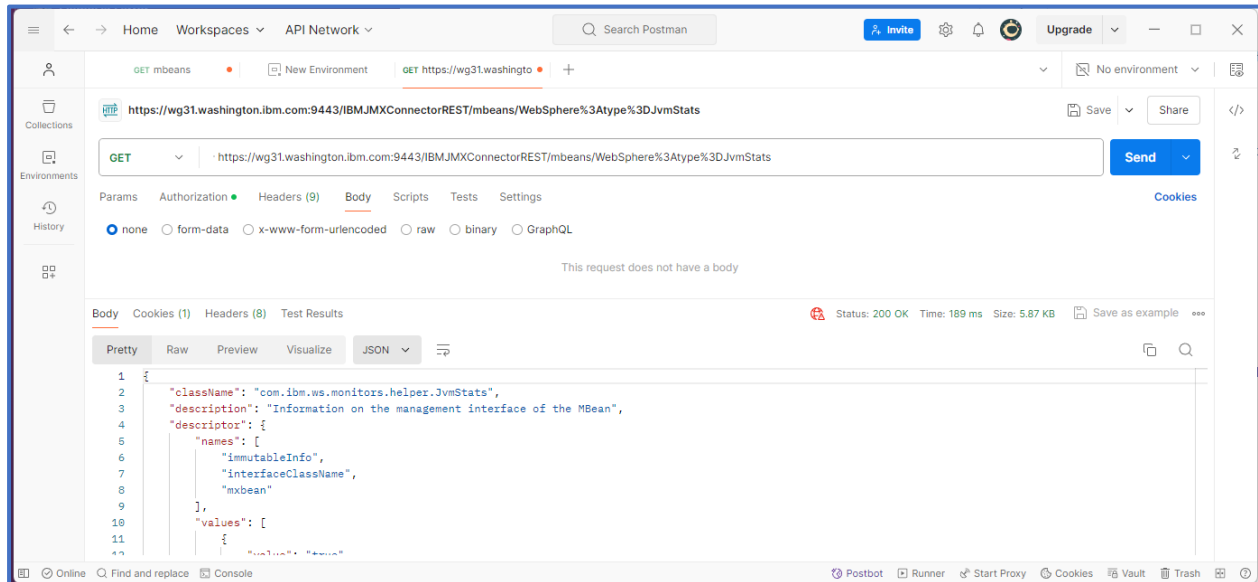
Accessing MBean JvmStats

- Use the search tool to locate the MBean in which we are interested, *JvmStats*.

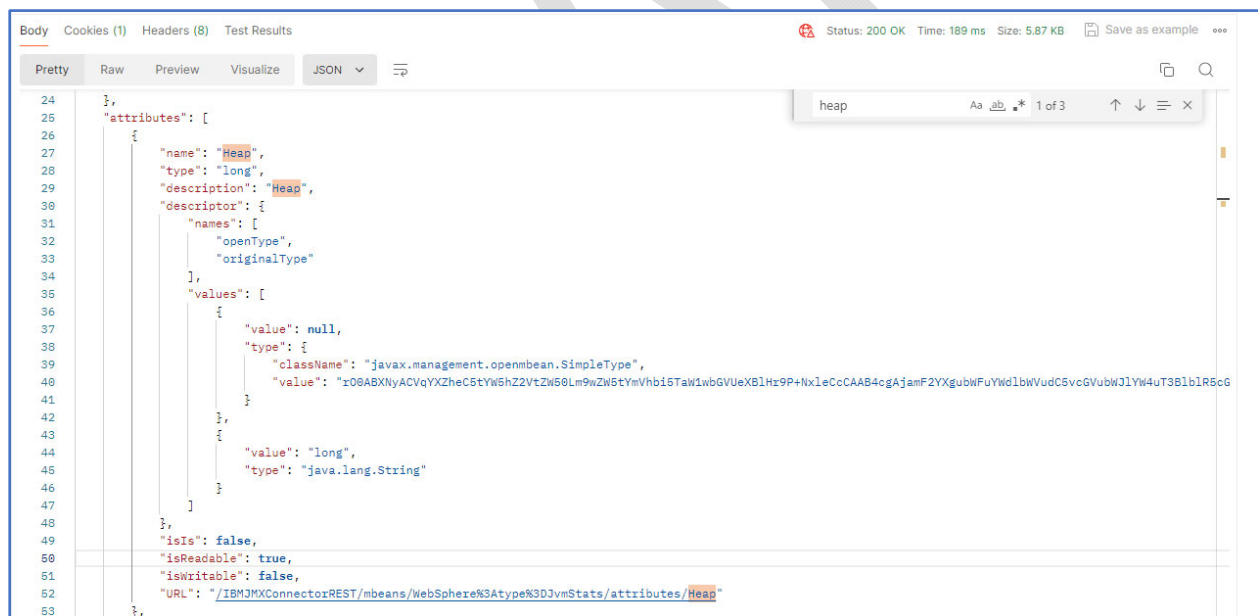


Monitoring a z/OS Liberty server

- Simply click on the URL in the *Response*, e.g., `"IBMJMXConnectorREST/mbeans/WebSphere%3Atype%3DJvmStats"` and this will open a new Postman window with the full URL already filled in. You may have to add the same credentials for this request, but then just click the **Send** button to invoke this new MBean.

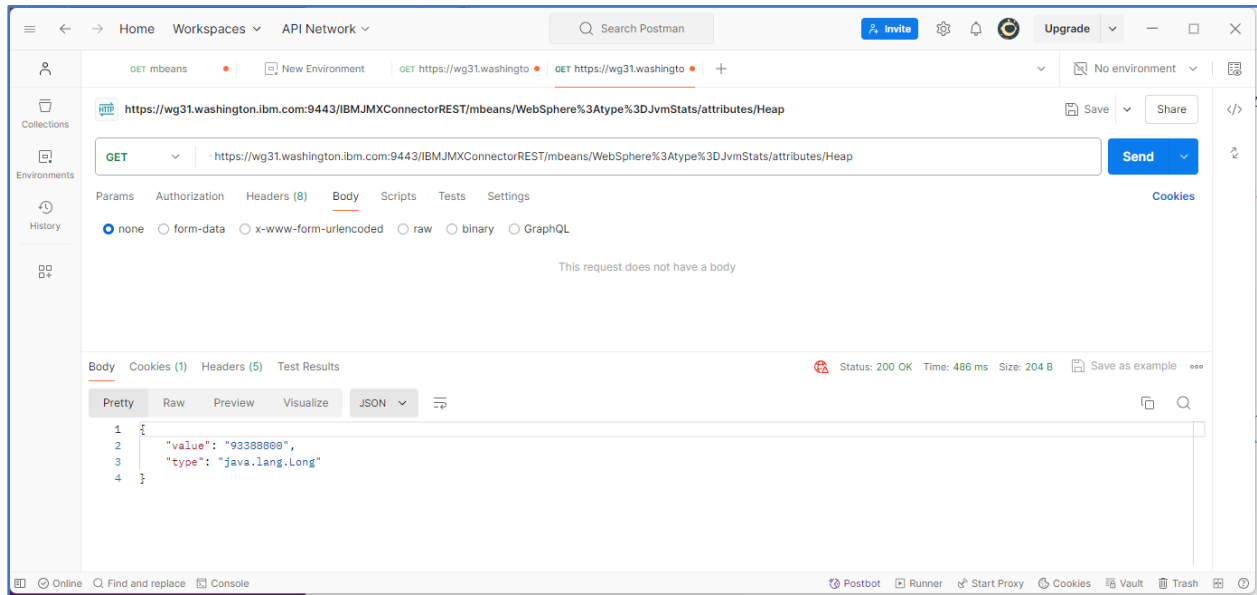


- Use the search tool again to search for a particular attribute, e.g. *heap*.



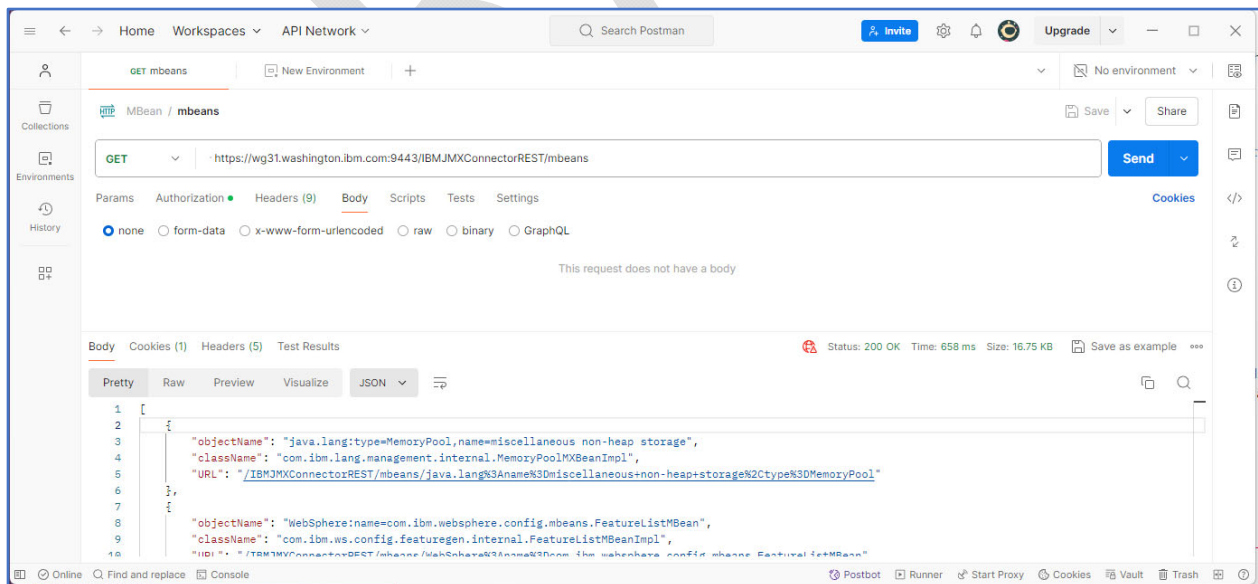
Monitoring a z/OS Liberty server

- Click on the URL, e.g., ["/IBMJMXConnectorREST/mbeans/WebSphere%3Atype%3DJvmStats/attributes/Heap"](https://wg31.washington.ibm.com:9443/IBMJMXConnectorREST/mbeans/WebSphere%3Atype%3DJvmStats/attributes/Heap) and press the **Send** button on the new window to invoke this MBean.



Next let's explore display connection pool information.

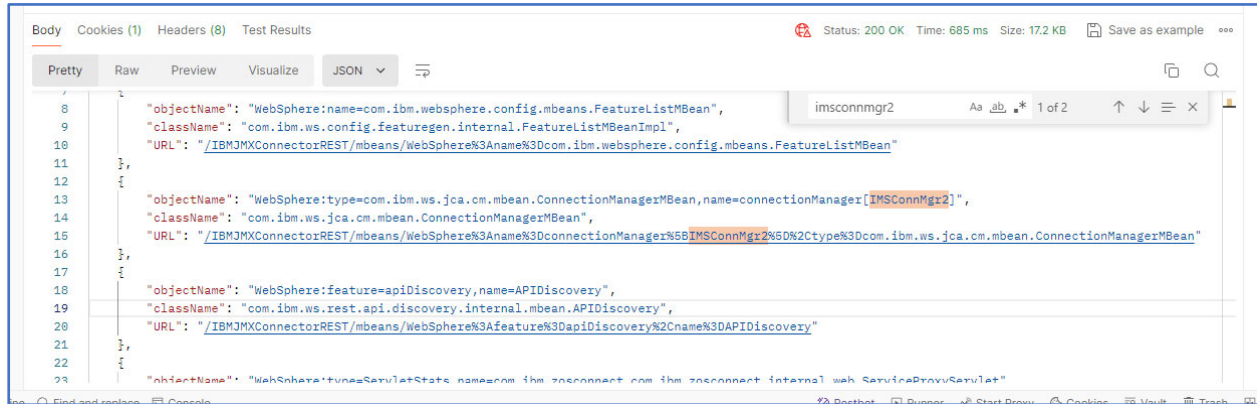
- Go back to the Postman tab where display the list of MBeans by sending a **GET** to URL <https://wg31.washington.ibm.com:9443/IBMJMXConnectorREST/mbeans>.



Monitoring a z/OS Liberty server

Accessing MBean ConnectionManagerMBean

- Use the search tool to locate the connection pool in which we are interested, *IMSConnMgr2*.

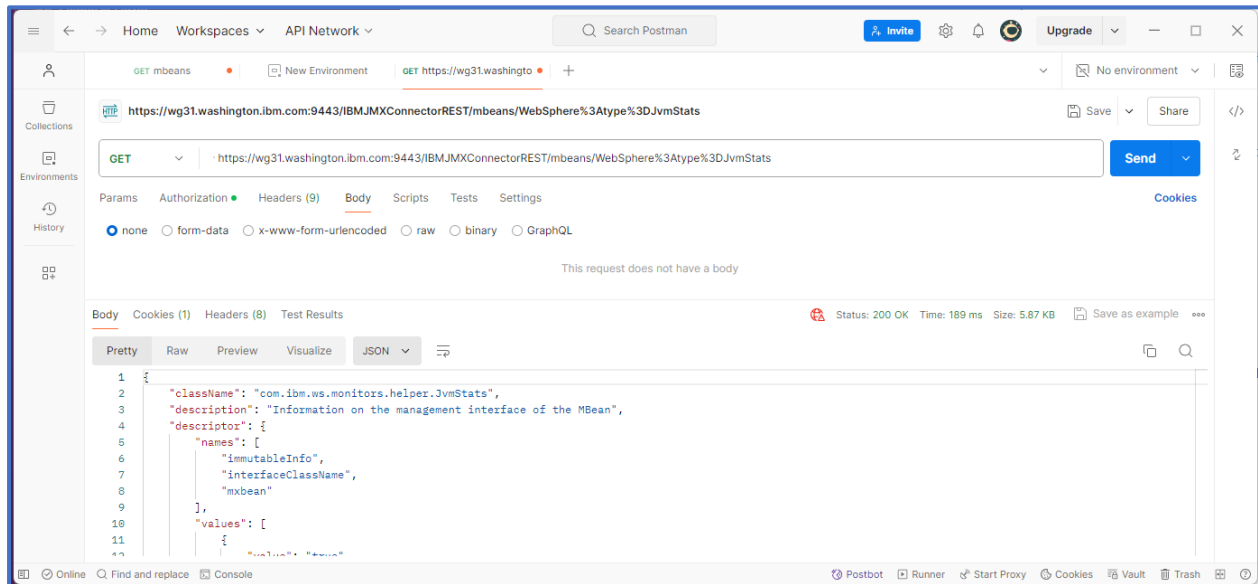


The screenshot shows a REST client interface with a JSON response. The response is a list of MBean objects. The second object, representing the ConnectionManagerMBean, is highlighted. A search bar at the top right contains the text 'imsconnmgr2', and the search results show the object name 'connectionManager[IMSConnMgr2]'.

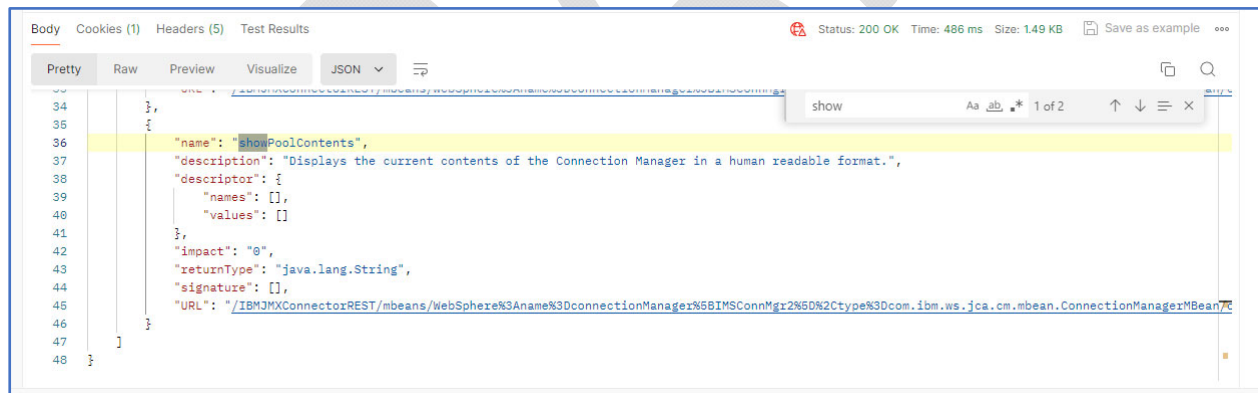
```
8  {
9    "objectName": "WebSphere:name=com.ibm.websphere.config.mbeans.FeatureListMBean",
10   "className": "com.ibm.ws.config.featuregen.internal.FeatureListMBeanImpl",
11   "URL": "/IBMJMXConnectorREST/mbeans/WebSphere%3Aname%3Dcom.ibm.websphere.config.mbeans.FeatureListMBean"
12 },
13 {
14   "objectName": "WebSphere:type=com.ibm.ws.jca.cm.mbean.ConnectionManagerMBean,name=connectionManager[IMSConnMgr2]",
15   "className": "com.ibm.ws.jca.cm.mbean.ConnectionManagerMBean",
16   "URL": "/IBMJMXConnectorREST/mbeans/WebSphere%3Aname%3DconnectionManager%5BIMSConnMgr2%5D%2Ctype%3Dcom.ibm.ws.jca.cm.mbean.ConnectionManagerMBean"
17 },
18 {
19   "objectName": "WebSphere:feature=apiDiscovery,name=APIDiscovery",
20   "className": "com.ibm.ws.rest.api.discovery.internal.mbean.APIDiscovery",
21   "URL": "/IBMJMXConnectorREST/mbeans/WebSphere%3Afeature%3DapiDiscovery%2Cname%3DAPIDiscovery"
22 },
23 {
24   "objectName": "WebSphere:tunesServletStats,name=com.ibm.ws.connector.com.ibm.ws.connector.internal.web.ServletProvServlet"
25 }
```

Monitoring a z/OS Liberty server

- Simply click on the URL in the *Response*, e.g., [/IBMJMXConnectorREST/mbeans/WebSphere%3Aname%3DconnectionManager%5BIMSCnnMgr2%5D%2Ctype%3Dcom.ibm.ws.jca.cm.mbean.ConnectionManagerMBean](https://wg31.washington.ibm.com:9443/IBMJMXConnectorREST/mbeans/WebSphere%3Aname%3DconnectionManager%5BIMSCnnMgr2%5D%2Ctype%3Dcom.ibm.ws.jca.cm.mbean.ConnectionManagerMBean) and this will open a new Postman window with the full URL already filled in. A new set of credentials may be needed for this request but then just click the **Send** button to invoke this new MBean.

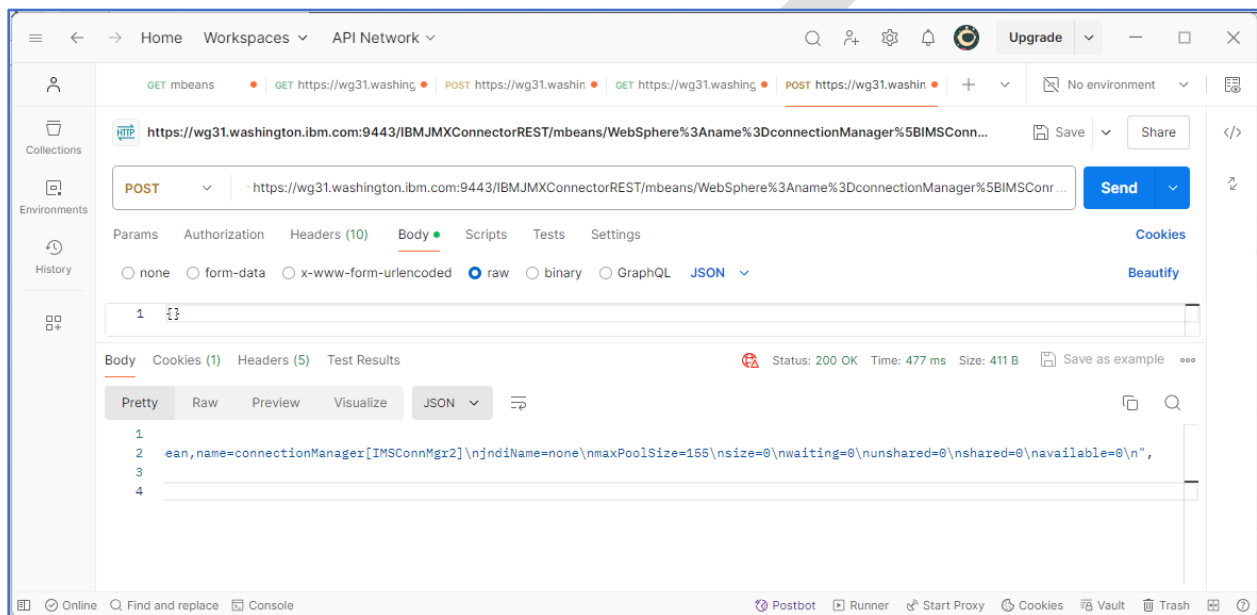


- Use the search tool again to search for a particular *operation*, e.g. *showPoolContents*.



Monitoring a z/OS Liberty server

- Click on the URL, e.g.,
[/IBMJMXConnectorREST/mbeans/WebSphere%3Aname%3DconnectionManager%5BIMSConnMgr2%5D%2Ctype%3Dcom.ibm.ws.jca.cm.mbean.ConnectionManagerMBean/operations/showPoolContents](https://wg31.washington.ibm.com:9443/IBMJMXConnectorREST/mbeans/WebSphere%3Aname%3DconnectionManager%5BIMSConnMgr2%5D%2Ctype%3Dcom.ibm.ws.jca.cm.mbean.ConnectionManagerMBean/operations/showPoolContents) and press the **Send** button on the new window to invoke this MBean. Invoking the MBean should failed with an HTTP 405 error, *method not allowed*. To address this, changed the method to **POST**. Since we are now using a **POST**, a request message must be added. Simply checked the radio *Body* in the Postman request message section and used the pull down to select *raw*. Entered a simple JSON message consisting of a beginning and ending braces, `{}`. Now pressing the **Send** button successfully accesses the MBean to display the connection pool details.



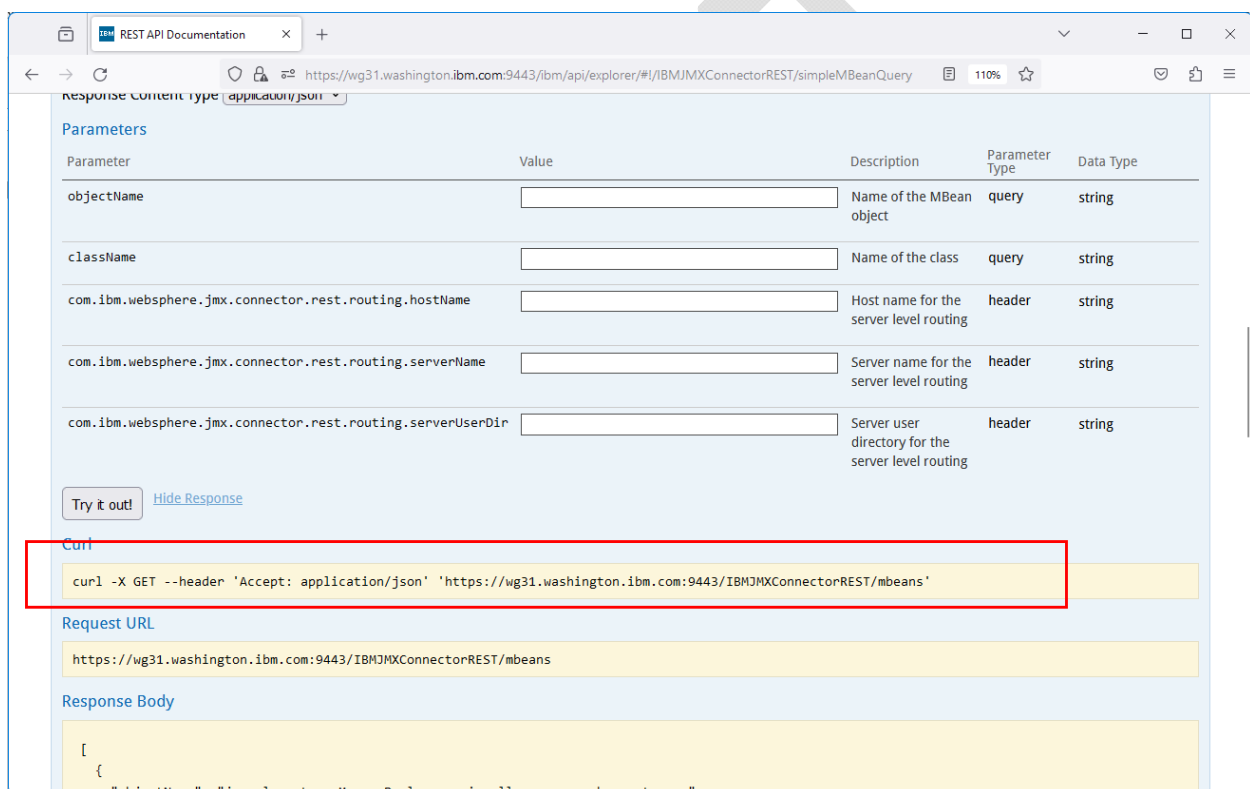
Monitoring a z/OS Liberty server

Using cURL to access MBeans using REST

Accessing the MBeans from a REST client from a cURL (client URL) is another option. The best way to create the cURL command for accessing MBeans by using either the *IBM API Explorer* or *Postman* to invoke the MBean first. The advantage of using these tools first is once you are able to access an MBean with these tools, they will provide you with the syntax of cURL command to accomplish the same results.

Building cURL commands from IBM API Explorer

Note that when the *IBM API Explorer* provides the results, the “almost” equivalent cURL is also displayed (see below).



I said almost because there are some cURL parameters that must be added and the use of single quotes have to be replaced by double quotes. Using the above example, the curl command would need to be changed

From:

```
curl -X GET --header 'Accept: application/json'
'https://wg31.washington.ibm.com:9443/IBMJMXConnectorREST/mbeans'
```

To:

```
curl -X GET --cacert CERTAUTH.PEM --user USER1:USER1 --header "Accept:
application/json"
https://wg31.washington.ibm.com:9443/IBMJMXConnectorREST/mbeans
```

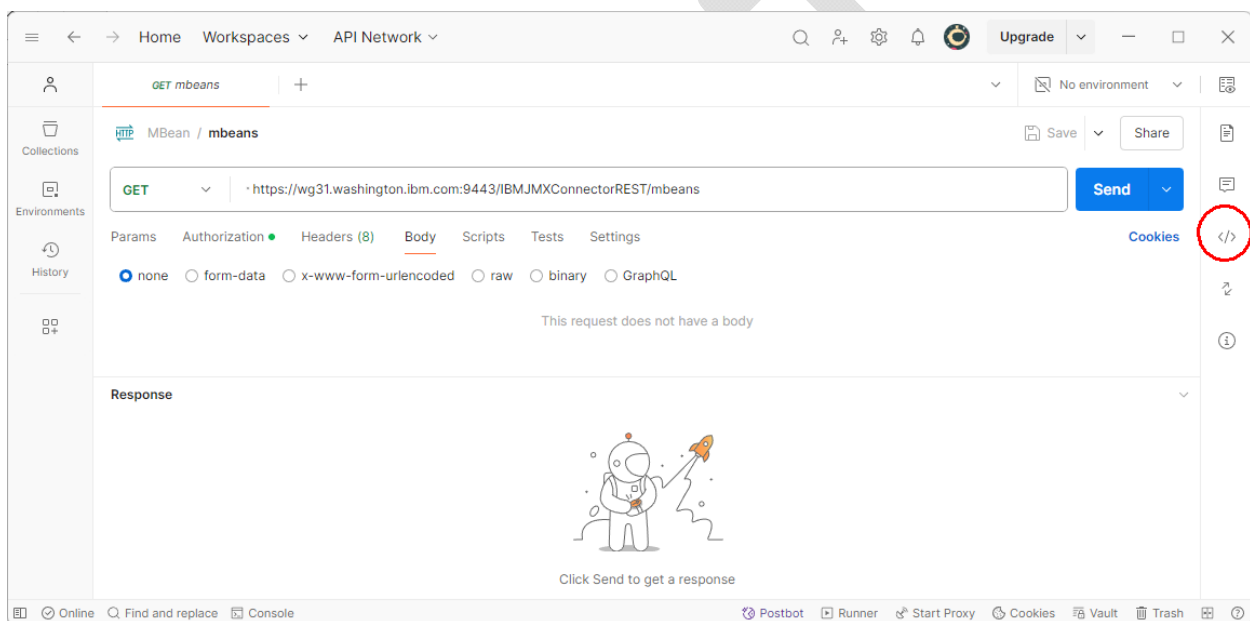

Monitoring a z/OS Liberty server

Where `--cacert` identifies a local trust store and `--user` provides basic authentication credentials. cURL has the option of providing `--insecure` as an alternative to providing a local truststore. With `--insecure` provided, the SSL handshake completes without verifying the server's certificate information.

Note in the above command, the certificate authority certificate file, CERTAUTH.PEM, is the same file used to configure TLS for the JConsole GUI client.

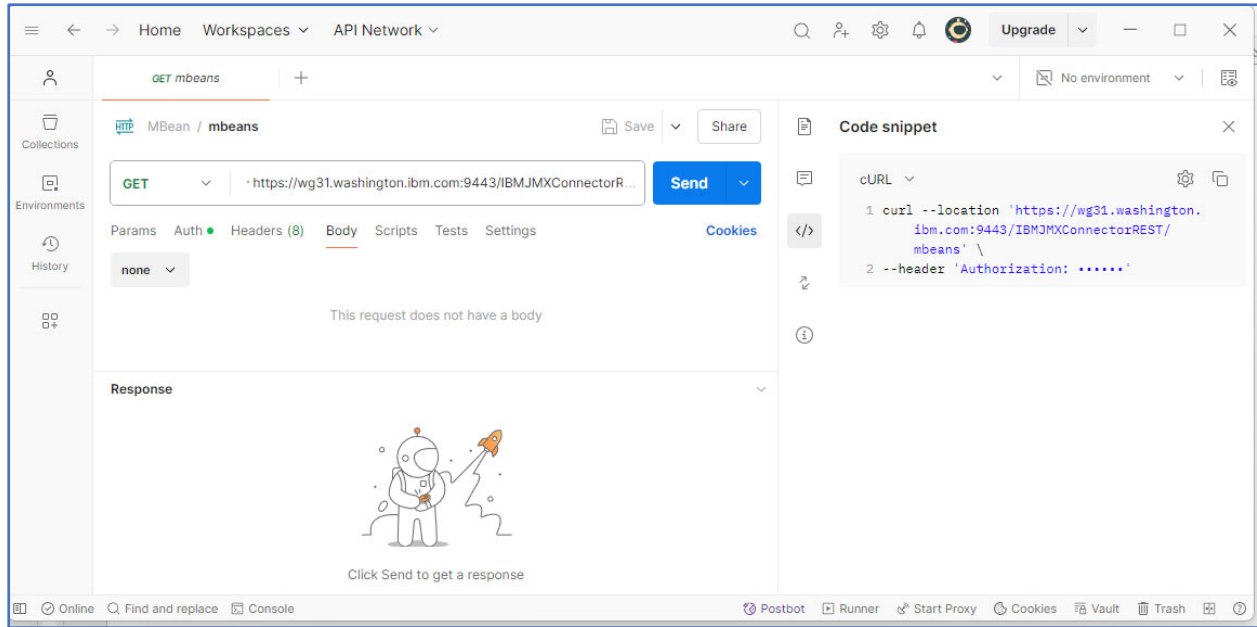
Building cURL commands from Postman

With Postman you can monitor the equivalent cURL command by pressing the `</>` icon on the side tab.



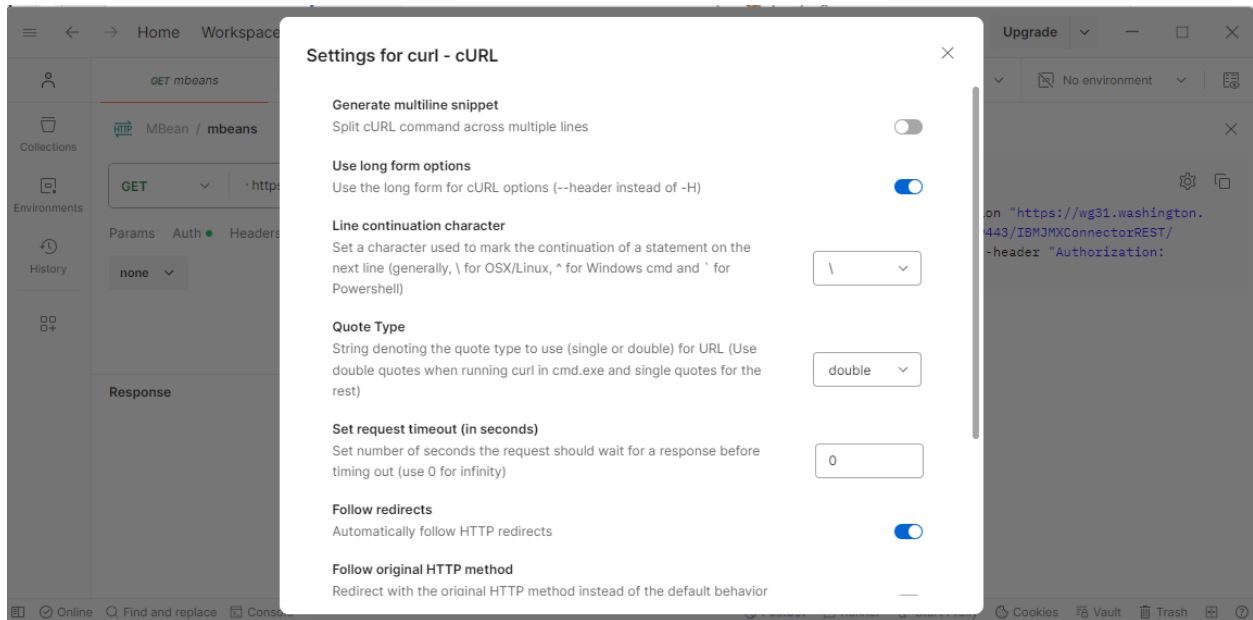
Select this option will display the equivalent cURL command. I selected the sprocket in this pane in order to customize the cURL command to way needed for a DOS command prompt.

Monitoring a z/OS Liberty server

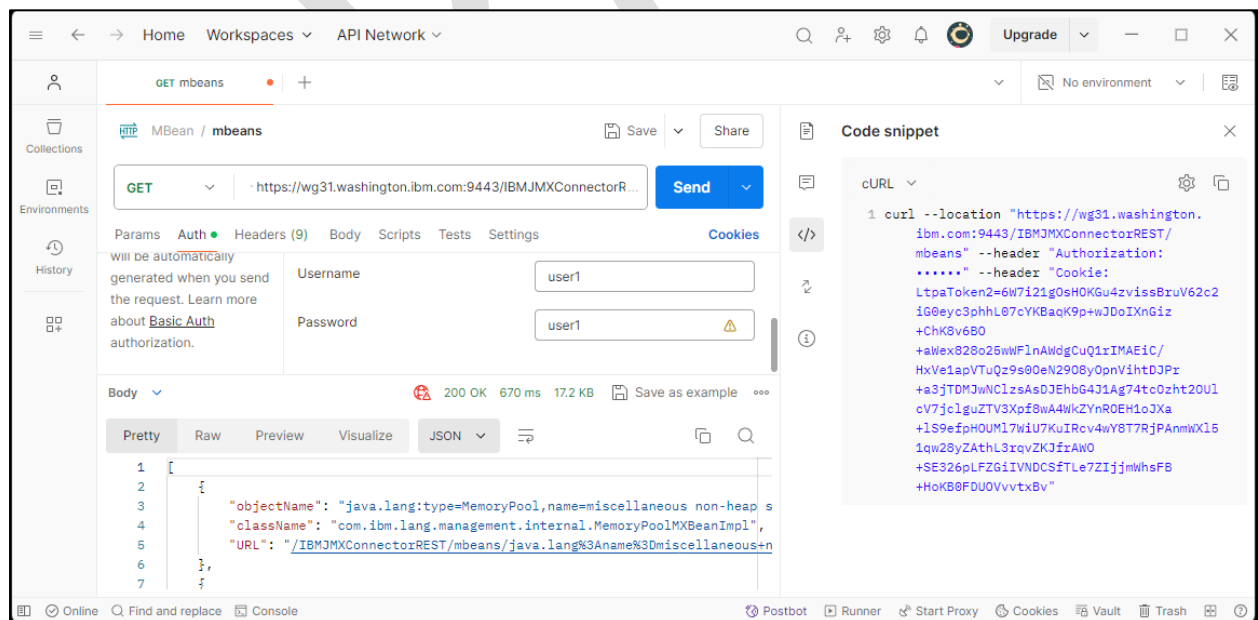


Monitoring a z/OS Liberty server

I changed the cURL settings so that *Generate multiline snippet* was turned off and the *Quote Type* would use double quotes rather than single quotes.



Now the cURL command in the Code snippet pane can be copied and used externally from Postman with a couple of exceptions. The *-header* for *Authorization* and the *-header* for *Cookie* should be removed before or after the cURL command is copied.



Monitoring a z/OS Liberty server

Below are some examples of using these techniques to build various cURL command and their results when invoked.

```
C:\z\jconsole>curl --cacert CERTAUTH.PEM --user USER1:USER1
```

```
https://wg31.washington.ibm.com:9443/IBMJMXConnectorREST/mbeans/WebSphere%3Aname%3DDefault+Executor%2Ctype%3DThreadPoolStats/attributes
```

```
[{"name":"PoolSize","value":{"value":"20","type":"java.lang.Integer"}}, {"name":"ActiveThreads","value":{"value":"1","type":"java.lang.Integer"}}, {"name":"PoolName","value":{"value":"Default Executor","type":"java.lang.String"}}]
```

```
C:\z\jconsole>curl --cacert CERTAUTH.PEM --user USER1:USER1
```

```
https://wg31.washington.ibm.com:9443/IBMJMXConnectorREST/mbeans/WebSphere%3Aname%3DDefault+Executor%2Ctype%3DThreadPoolStats/attributes/PoolName
```

```
{"value":"Default Executor","type":"java.lang.String"}
```

```
C:\z\jconsole>curl --cacert CERTAUTH.PEM --user USER1:USER1
```

```
https://wg31.washington.ibm.com:9443/IBMJMXConnectorREST/mbeans/WebSphere%3Aname%3DDefault+Executor%2Ctype%3DThreadPoolStats/attributes/PoolSize
```

```
{"value":"20","type":"java.lang.Integer"}
```

```
C:\z\jconsole>curl --cacert CERTAUTH.PEM --user USER1:USER1
```

```
https://wg31.washington.ibm.com:9443/IBMJMXConnectorREST/mbeans/WebSphere%3Aname%3DDefault+Executor%2Ctype%3DThreadPoolStats/attributes/ActiveThreads
```

```
{"value":"1","type":"java.lang.Integer"}
```

```
C:\z\jconsole>curl -X POST --cacert CERTAUTH.PEM --user USER1:USER1 -d "{}" --header "Content-Type: application/json" --header "Accept: application/json"
```

```
https://wg31.washington.ibm.com:9443/IBMJMXConnectorREST/mbeans/WebSphere%3Aname%3DconnectionManager%5BIMSConnMgr2%5D%2Ctype%3Dcom.ibm.ws.jca.cm.mbean.ConnectionManagerMBean/operations/showPoolContents
```

```
{"value":"PoolManager@de62db71\nname=WebSphere:type=com.ibm.ws.jca.cm.mbean.ConnectionManagerMBean,name=connectionManager[IMSConnMgr2]\njndiName=none\nmaxPoolSize=155\nsize=0\nwaiting=0\nunshared=0\ninshared=0\navailable=0\n","type":"java.lang.String"}
```

```
C:\z\jconsole>curl --location
```

```
"https://wg31.washington.ibm.com:9443/IBMJMXConnectorREST/mbeans/WebSphere%3Atype%3DJvmStats/attributes" --insecure --user USER1:user1
```

```
[{"name":"Heap","value":{"value":"139067392","type":"java.lang.Long"}}, {"name":"FreeMemory","value":{"value":"37360080","type":"java.lang.Long"}}, {"name":"ProcessCPU","value":{"value":"1.0","type":"java.lang.Double"}}, {"name":"UsedMemory","value":{"value":"101707312","type":"java.lang.Long"}}, {"name":"GcTime","value":{"value":"977","type":"java.lang.Long"}}, {"name":"UpTime","value":{"value":"93425998","type":"java.lang.Long"}}, {"name":"GcCount","value":{"value":"347","type":"java.lang.Long"}}]
```