Memory Management

Servoy is built using Java technology and as such the memory management for Servoy is basically Java Memory management. This chapter describes the basics of Java Memory management, which will be sufficient for all but the most extensive and complex deployment scenario's.

In This Chapter

Java Memory Management Introduction

The basics of Java Memory management is straight forward:

- A Java process uses several memory spaces (called the Perm Space or Heap Space for example), each of which use a dedicated block of memory
- For each space the Java process determines the initial and maximum amount of memory it is allowed to use, of not explicitly specified through configuration. The defaults values differ per JVM implementation and can depend on the hardware on which the JVM is started.
- The initial and maximum's for each space can also be explicitly specified through configuration
- The initial memory sizes for all spaces are allocated at startup of the Java process
- The Java process will allocate more memory only when required, up the specified maximum per space
- Allocated memory that is not used anymore will periodically be released in a process called Garbage collection. In Java this is a fully automated process
- If the Java process requires more memory than the maximum in any of the spaces, the Java process will throw a relevant exception, for example an OutOfMemory or Stack Overflow exception. These exceptions should be prevented as they both degrade performance, but more importantly can cause unexpected behavior.

Java Memory Spaces

The two most important spaces of memory that a Java process uses are the Java Heap Space and the Java Perm Space.

Java Heap Space

The Heap Space is the main space used by Java while operating. As such this space will use the most memory. The settings related to the Java Heap Space are the following:

- Xmx - Maximum Heap Space value
- Xms - Initial Heap Space Value

When the Maximum Heap Space size is too low for that is actually needed by the Java process, the following exception will be thrown: *java.lang.OutOfMemoryError: Java heap space*

Java Perm Space (java 7)

The Perm Space is the space used by Java to store loaded classes and other static objects. The setting related to the Java Perm Space is only the following.

- XX:MaxPermSize - Maximum Perm Space value

When the Maximum Perm Space size is too low for that is actually needed by the Java process, the following exception will be thrown: *java.lang.OutOfMemoryError: PermGen space*

⚠ The above mentioned settings are settings that can be applied when launching a Java process command-line or form a .bat/.sh script. Certain application might provide other ways to specify the same settings, for example the Service component that is part of the Servoy distribution. For more information on how to configure these settings withing Servoy, read the next paragraph.

Java metaspace (java 8 and higher, replacement of the java 7 perm space above)

Java 8 changed the way it stored classes and is using now native memory for that instead of a special java heap

these settings can be adjusted by using the following vm arguments:

-XX:MetaspaceSize=200M -XX:MaxMetaspaceSize=200M -XX:MinMetaspaceFreeRatio=0 -XX:MaxMetaspaceFreeRatio=100

this sets it to use 200MB for that if you get a *java.lang.OutOfMemoryError: metaspace* because a lot of plugins are used then you need to update that value to something higher like 300/400.

Server Memory Management

Memory management for a Servoy Application Server revolves around managing the values for the Java Heap Space and Java Perm Space.

The default values are:

- Java Heap Space: 1280Mb
- Java Perm Space: 128Mb (java7)
- Java meta space: 200Mb (java8+)

Depending on how the Servoy Application Server is launched, the location where these values can be managed differs:

- Started using the Service component > `{serverInstall}\application_server\service\wrapper.conf`
- Manually started stand-alone > `{serverInstall}\application_server\servoy_server.bat/.sh`
- Manually started in Servoy Cluster > `{serverInstall}\application_server\terracotta/start_servoy_clustered.bat/.sh`

Determining the Required Java Heap Space Value

The components that contribute to the Java Heap Space requirements for a Servoy Application Server consists of the following parts:

- The Servoy Application Server itself
- The memory usage related to active/idle database connections
- The memory usage related to running Servoy Clients

The memory usage of a standard Servoy Application Server is limited to a couple of Mb.

For database connections assume 2Mb per connection, so sum the Maximum Active connections settings on each configured and enabled Database Server and multiply that by 2Mb to determine the maximum memory consumption by all database connections

The memory usage for the running Servoy Clients is more difficult to obtain, as it depends on the number of running Clients, the type of the running Servoy Clients, the way the Servoy Solutions that are running are built and the way the users use these Solutions. These variables together make it difficult to provide hard numbers on how to configure the memory settings beforehand. Getting it right means monitoring and tuning.

As a starting point the following rule of thumb can be used to determine the initial settings:

- Smart Client using the Servoy HTTP Tunnel: assume 900kb for each Smart Client
- Smart Client using another connection mode than the HTTP tunnel: assume 600kb for each Smart Client
- Web Client, Headless Client (incl. the RESTfull plugin connection poolsize) & Batch processors: the required memory for each client depends on the size and design of the solution, but should be in the order of magnitude of a couple of Mb per Client. For more guidelines, see below.

Smart Client Memory Usage vs. Other Servoy Client

The Smart Client runs on the client machine, not on the Servoy Application Server like the other Servoy Clients, like the Web Client, Headless Client or Batch Processor. The Smart Client is only registered with the Application Server and thus consumes far less memory on the Application Server compared to the other Servoy Clients. See Smart Client Memory management for more information on configuring the Memory settings for Smart Clients specifically.

Web Client, Headless Client & Batch Processors Memory Usage

The memory usage of Web Clients, Headless Clients and Batch Processors depends highly on the design of the solution. While Servoy optimizes many things to keep the memory footprint as low as possible, it is logical that a simple solution showing just one form at the time has a different memory footprint than a solution that has a very complex UI, showing 20 Forms and/or different sets of data at the same time.

The memory footprint is not linear with the number of Forms a Solution contains, as Forms are instantiated on a need-to basis.

As there is no way to determine up front how much memory a Web/Headless/Batch Processor Client will use, the only way to dimension the memory correctly is by monitoring and tuning it.

For the initial dimensioning, use the following rules of thumb:

- Small solution (1 - 50 forms), simple user interface: 2Mb
- Medium size solution (51 - 500 forms), average complex user interface: 10Mb
- Large solution (> 500 forms), complex user interface: 20Mb

Configuring the Java Heap Space Value

The setting that requires tuning based on the determined required memory for the Servoy Application Server is the Maximum Heap Space setting (Xmx). By default the Maximum Heap Space is set to 1028MB within Servoy. This setting should be changed when:

- The expected load is higher (see previous paragraph for determining the required memory)
- The actual used memory is >= 70% of the specified maximum. The actually used memory can be found on the main page of the Admin page, under System Information: Heap memory: allocated=549184K, used=371473K, max=699072K
- When there is plenty of free real memory available on the OS level. Java processes in general perform better when not having memory constraints. For example, when 2Gb free real memory is left, add the 2Gb to the maximum heap size. The Java process will only take what it needs.

Depending on how the Servoy Application Server is started, the Memory setting need to be applied in a different location.

Auto Start: wrapper.conf

When using the Service component to automatically launch the Servoy Application Server when the machine on which it is installed is booted, the memory settings for the Application Server can be configured inside `{serverInstall}\application_server\service\wrapper.conf` by altering the `wrapper.java.maxmemory` setting:

```
wrapper.java.additional.5=-XX:MetaspaceSize=200M
wrapper.java.additional.6=-XX:MaxMetaspaceSize=200M
wrapper.java.additional.7=-XX:MinMetaspaceFreeRatio=0
wrapper.java.additional.8=-XX:MaxMetaspaceFreeRatio=100
wrapper.java.additional.9=-XX:MaxPermSize=128m
wrapper.java.additional.10=-XX:+UseG1GC
# make sure that the hardcoded System.gc() or Runtime.gc() are ignored (rmi does this)
wrapper.java.additional.11=-XX:+DisableExplicitGC
# Initial Java Heap Size (in MB), when G1GC is used just set the init memory to the max memory
wrapper.java.initmemory=1280
# Maximum Java Heap Size (in MB)
wrapper.java.maxmemory=1280
```

For more information on configuring the Service component, see Running the Server As a Service

Manual Start: servoy_server.bat/.sh or ../terracotta/start_servoy_clustered.bat/.sh (when running Servoy Cluster)

When starting the Servoy Application Server manually, either through `{serverInstall}\application_server\servoy_server.bat/.sh` *or* `{serverInstall}\application_server\terracotta/start_servoy_clustered.bat/.sh`, the memory settings for the Application Server can be configured inside *the .bat/.sh*, by altering the -Xmx value:

```
java ... -Djava.awt.headless=true -Xmx1280m -Xms64m -XX:MetaspaceSize=200M -XX:MaxMetaspaceSize=200M -XX:
MinMetaspaceFreeRatio=0 -XX:MaxMetaspaceFreeRatio=100  .....
```

⚠ **Specifying -Xmx and -Xms values**

When specifying the -Xmx and -Xms values, the following needs to be taken into account:

- The value can only be an integer value
- The value needs to be places directly behind the parameter, no spaces or equal signs
- The value needs to be post fixed with the unit: 'm' or 'M' for megabytes, 'g' or 'G' for gigabytes
- When changing the -Xms options, make sure that the -Xmx option has a higher value
- Both values need to be lower than the available memory on the machine

In a default Servoy Application Server installation, the Maximum Heap Space value (-Xmx) is set to 1280m. For servers that have limited memory, this value could be too high, resulting in the following error when trying to start the server:

```
Error occurred during initialization of VM
Could not reserve enough space for object heap
Error: Could not create the Java Virtual Machine.
Error: A fatal exception has occurred. Program will exit.
```

Either lower the -Xmx value or make sure that more memory is available for the Java process on the server, by closing other applications or adding more physical memory to the server.

32bit Java Memory Assignment Limitation

⚠ An 32 bit JVM will allow a maximum memory assignment of 2Gb in total. In order to assign more than 2Gb of memory, a 64 bit JVM on a 64 bit OS is required.

Determining the Required Java Metaspace or Perm Space Value

By default the Java Heap Space size is set to 128Mb/200Mb, which is a sufficient value for most deployments. A scenario in which this value needs increasing is when the Servoy Application Server is loaded up with a lot of additional plugins, beans, LAF's and drivers.

Configuring the Java Perm/Metaspace Space Value

For all three scenario's that the Servoy Application Server can be started (as a service, manual stand-alone or manual clustered), the value for the Java Perm Space is controlled by the 'XX:MaxPermSize' setting in the above mentioned configuration location for java 7 or for java 8:

-XX:MetaspaceSize=200M -XX:MaxMetaspaceSize=200M -XX:MinMetaspaceFreeRatio=0 -XX:MaxMetaspaceFreeRatio=100

Monitoring the Memory Usage

The actual memory usage of a Servoy Application Server can be observed on the Servoy Admin page. The 'Servoy Server Status' overview on the 'Servoy Server Home' page displays the allocated, used and maximum values for the Heap memory and all other spaces combined (non-Heap memory).

Or through Monitoring Servoy with JMX

Running big servers with big heaps for many Web/NGClients

For large heaps (more then 4G) that needs to handle quite a lot of Web or NGClients the vm needs some tuning to be able to handle this nicely.

Use the latest Java 8 and use the G1 Garbagabe collector for this and make sure that the initial heap and the maximum heap ist just set to the same value:

`-server -Xms4G -Xmx4G -XX:+UseG1GC -XX:MetaspaceSize=200M -XX:MaxMetaspaceSize=200M -XX:MinMetaspaceFreeRatio=0 -XX:MaxMetaspaceFreeRatio=100 -XX:+DisableExplicitGC`

Where the 4G for the initial (Xms) and the max (Xmx) should be set to the maximum heap you want or can assign to the java vm.

Smart Client Memory Management

Similar to the Servoy Application Server, the Servoy Smart Client is also a Java application and as such the same memory management also applies to the Smart Client.

Through the Servoy Admin page, the following settings are exposed that related to the Smart Client's memory configuration:

Setting	What is does	Comment
servoy. initialClientHeap	Sets the Initial Heap Space size	
servoy. maxClientHeap	Sets the Maximum Heap Space size	
servoy. vmClientArgs	optional arguments that can be send to the Smart Client's JVM	Set to '-XX:SoftRefLRUPolicyMSPerMB=3600000' by default: this setting improves performance

Within the Servoy Smart Client, the actual memory usage can be monitored through the **Help** > **About** menu item.