Integrated Capture Points Guide

Java Configuration

5/8/2025

Java Configuration

> **Important**
>
> The 8.5.1 family of Interaction Server releases support JRE 1.8. For a full list of supported Java versions, see the eServices page in the *Genesys Supported Operating Environment Reference Guide*.

Use the latest Java JRE versions for JMS Capture Points and Groovy transformation scripts, as well as for File Capture Points if Groovy transformation scripts are used (for example, for iWD compatibility mode). Versions before JRE 1.5 are not supported.

Here is a general description of the configuration requirements for Java:

- Configure the `jvm-path` option in Interaction Server. In the `java-config` section, the `jvm-path` option must specify the path to the `jvm.dll` file (for Windows) or `libjvm.so` file (for UNIX platforms). Interaction Server requires this to start JVM by means of JNI. This option is required for JMS Capture Points and Groovy transformation scripts.
- Configure the `jvm-options` section in Interaction Server. This section lists JVM option pairs, for example `["-Xmx256m",""]` or `["-Djava.class.path",".;C:\myjars\my-jar.jar;C:\myotherjars\my-other-jar.jar"]`. If JMS Capture Points or Groovy transformations are present, the option `-Djava.class.path` must contain a path to the Genesys-provided JAR files, as well as the Message Queue provider-specific JAR files, which are required in order for JMS and Groovy scripts to run.

These options are explained in more detail below.

For more information about these and other Capture Point-related Interaction Server options, refer to the eServices Reference Manual.

Configuring Interaction Server to Load the Java Virtual Machine (JVM)

To enable JMS capture point functionality or Groovy transformation functionality, Interaction Server must be configured to load the Java Virtual Machine. The latest JRE 1.5 or 1.6 is required (JDK is not required). Take care to specify the correct virtual machine with regard to the architecture; that is, for 64-bit Interaction Server, 64-bit JVM must be used and for 32-bit Interaction Server, 32-bit JVM must be used.

Interaction Server java-config Section

The section should contain only one option: `jvm-path`. This option specifies the full path to the

`jvm.dll` (on the Windows platform) or to `libjvm.so` (on UNIX platforms). If this option is not present, Interaction Server does not attempt to load JVM. The following is an example of this option for the Windows platform:

```
jvm-path=C:\Program Files\Java\jdk1.6.0_13\jre\bin\server\jvm.dll
```

The following is an example of this option for Solaris 10:

```
jvm-path=/usr/local/java/jdk1.6.0_22/jre/lib/sparcv9/server/libjvm.so
```

Note that JVM comes in two flavors: client and server. The server VM is preferred since it is optimized for long-running processes and mostly runs compiled code, while the client VM starts up faster but runs slower using an interpreted mode of execution.

Interaction Server jvm-options Section

This section specifies options that are used to run the JVM. Interaction Server composes the startup string for the JVM containing all of the options specified in this section.

-Xss1m

This option, with empty value, is required for all platforms. It specifies that the Java stack size should be 1 megabyte.

-Xoss1m

This option, with empty value, is required for all platforms. It specifies that the Native code stack size should be 1 megabyte.

It is important to note that Interaction Server creates many working threads to perform its tasks. If the stack size is set to be large, the multiplicity of threads will consume an unnecessarily large amount of memory. Many UNIX systems have unreasonably large default setting for stack size; the recommended stack size for Interaction Server is 1 megabyte.

Corrections to new section **-Djava.class.path**

> This option specifies the list of the files required to access JMS or for Groovy transformation functionality. On Windows, the semicolon (;) is used as a list separator, and the colon (:) on non-Windows platforms.
>
> The Interaction Server Installation package provides several JAR files that implement Java wrappers and they should be present in the list along with the path to the JAR files from the JMS provider as necessary.
>
> Below is a sample of a minimal class path for non-Windows that contains all the standard JAR files provided with Interaction Server:
>
> -Djava.class.path=transformation\xml_transformer_capture_point.jar:transformation\groovy-all-1.7.3.jar:transformation\xercesImpl.jar:transformation\xsltc.jar:jms\jms_wrapper.jar

> **Important**
> Numbers in the name of the file **<groovy-all-1.7.3.jar>** represent the version of the Groovy language library. With future releases the installation package may contain a newer version of it with a different number.

For OpenMQ, the provider-specific jar files are:

- **jms.jar**
- **imq.jar**
- **fscontext.jar**

For TIBCO, the provider-specific jar files are:

- jms.jar
- tibjms.jar

For ActiveMQ, the provider-specific jar file is **activemq-all-5.N.N.jar**, where *<5.N.N>* represents the specific version number from your installation of the ActiveMQ.

Additionally, a special file with the list of message queues should be configured, packed into the JAR file and the JAR added to the list. See vendor documentation on ActiveMQ about JNDI support for more details.

Below is a sample of the **-Djava.class.path** option value when Interaction Server is run on Windows and OpenMQ JMS is installed in the default destination:

-Djava.class.path=transformation\xml_transformer_capture_point.jar; transformation\groovy-all-1.7.3.jar;transformation\xercesImpl.jar; transformation\xsltc.jar;jms\jms_wrapper.jar;C:\Program Files\Sun\MessageQueue\mq\lib\fscontext.jar;C:\Program Files\Sun\MessageQueue\mq\lib\jms.jar;C:\Program Files\Sun\MessageQueue\mq\lib\imq.jar

-Djava.library.path

This option specifies the path to native libraries that might be required by JVM or specific JMS providers. On the Windows platform it is usually not necessary to specify this option. On UNIX platforms this option must specify the path to the JRE libraries and in certain cases the path to `libjvm.so` itself. For example, the IMB AIX platform requires `libjvm.so` to be in the library path since standard native libraries depend on it and will not load if it is not in the library path.

Take extreme care to specify the library path to the same JRE directory from which `libjvm.so` is loaded (the `jvm-path` option). If these do not match, it is often hard to find the reason why the solution is not working.

The following is an example of the option for IBM AIX platform (assuming 64-bit Interaction Server):
`-Djava.library.path=/lib:/usr/java6_64/jre/lib/ppc64:/usr/java6_64/jre/lib/ppc64/j9vm`

For AIX, in most cases you must modify the `./run.sh` file that was prepared for you during the

installation process, as follows:

1. Locate the string `./interaction_server -host <your_host> -port 8001 -app "InteractionServer"`
2. Add to the begining of it an expression that sets the `LIBPATH` environment. The resulting string will be:
   ```
   env LIBPATH=/lib:/usr/java6_64/jre/lib/ppc64:/usr/java6_64/jre/lib/ppc64/j9vm:$LIBPATH
   ./interaction_server -host <your_host> -port 8001 -app "InteractionServer"
   ```

Special Handling of xercesImpl.jar in JRE 1.5

If you are using JRE 1.5, there might be a conflict between the libraries included in the `groovy-all-1.7.3.jar` and `xercesImpl.jar`, resulting in the inability of JVM to either work with Xerces classes or to correctly execute the optional XML schema validation. To avoid this situation when working with 1.5 JRE, do the following must during configuration of the parameters:

- Move the `xercesImpl.jar` file from the `.\transformation` directory to the `.\transformation\endorsed` directory.
- In the `jvm-options` section of the Interaction Server object, create an option called `-Djava.endorsed.dirs` and give it the value `.\transformation\endorsed`.
- Ensure that the option `-Djava.class.path` correctly refers to the `xercesImpl.jar` located in the `.\transformation\endorsed` directory.

Operating System Environment

Interaction Server itself does not make use of any environment variables and should not require Java to be in the path or `JAVA_HOME` environment variable to be set. But if these are set, they **must** refer to the same JRE that is configured in the Interaction Server configuration options.

Different operating systems have different default settings for maximum number of threads a process can create. Interaction Server can and will create a few dozen threads. It is important that limits set for the operating system allow creating a few hundred threads. The default value of `1024` should be sufficient for almost all purposes. Consult with your system administrator to check the operating system limits and ensure that these are adequate for Interaction Server.

For example, the following might be required for AIX to change the limit (assuming that Interaction Server runs under the `itxsrvuser` account):

```
chuser threads=2048 itxsrvuser
```

Another important operating system parameter is the stack size for the thread. As previously mentioned, Interaction Server creates many threads and requires reasonable stack size for the threads. Some systems might have a default in the vicinity of 256 MB or more, which will definitely lead to problems when a process tries to create a few dozen threads. The stack size should be set to 2 MB for Interaction Server. The following command changes the thread stack size for most UNIX operating systems:

```
ulimit -s 2048
```

Again, consult your system administrator to check and ensure the correct operating system limits are in place before running Interaction Server.