Using the SAP Java Connector in Productions

Version 2025.1
2025-06-03

InterSystems Corporation   1 Memorial Drive   Cambridge MA 02142   www.intersystems.com

Using the SAP Java Connector in Productions
PDF generated on 2025-06-03
InterSystems IRIS® Version 2025.1

InterSystems Corporation disclaims responsibility for errors which may appear in this document, and it reserves the right, in its sole discretion and without notice, to make substitutions and modifications in the products and practices described in this document.

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Response Center (WRC)

Tel:	+1-617-621-0700
Tel:	+44 (0) 844 854 2917
Email:	support@InterSystems.com

Table of Contents

1 Overview of SAP Java Connector 1

2 Setup Tasks for the SAP Java Connector 3

- 2.1 Setting Up the Java Gateway 3
- 2.2 Installing the SAP JCo Jar File 4
- 2.3 Generating Proxy Classes for SAP JCo 4
- 2.4 Testing the SAP Connection 4

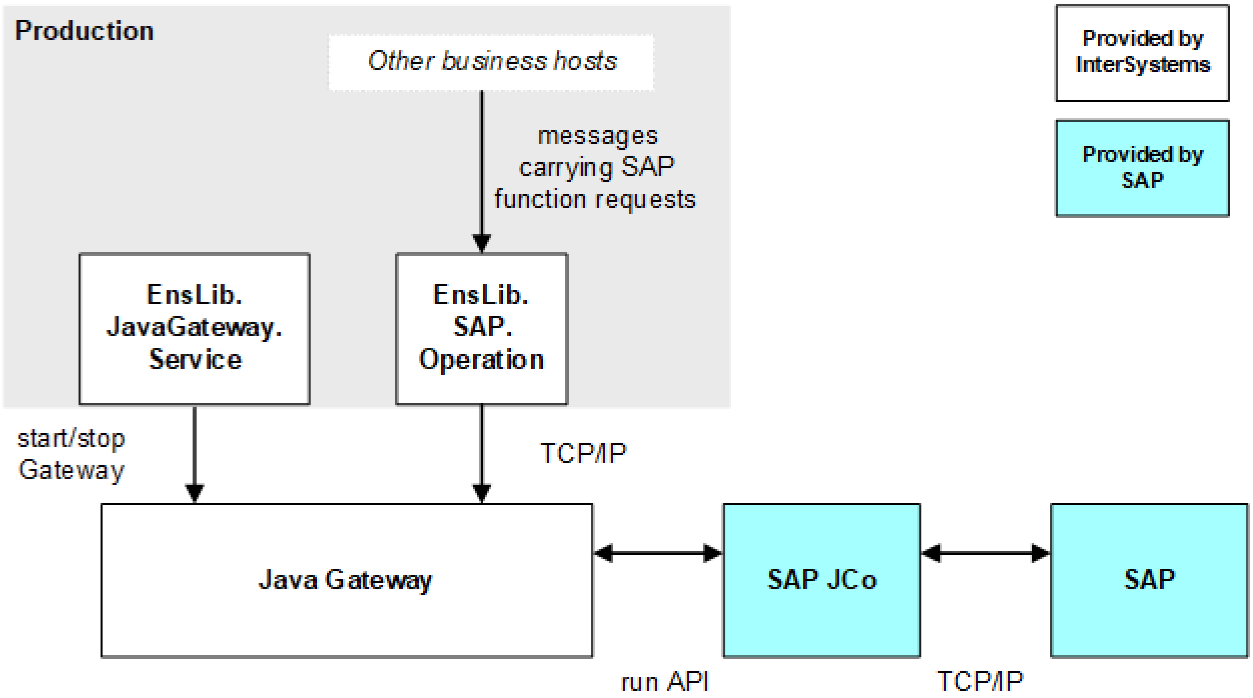
3 Using the SAP Java Connector 7

- 3.1 Basics 7
- 3.2 Settings for EnsLib.JavaGateway.Service 7
- 3.3 Settings for EnsLib.SAP.Operation 9

1

Overview of SAP Java Connector

SAP Java Connector (SAP JCo) is a Java-based component that supports communication with an SAP Server in both directions. InterSystems provides components that you can add to a production to enable the production to communicate with SAP JCo, and thus with an SAP Server. The following picture shows the architecture:



The architecture includes the Java Gateway, which must be running.

To communicate with SAP JCo, the production must include the following items:

- EnsLib.SAP.Operation, which communicates via TCP/IP with the Java Gateway.
- EnsLib.JavaGateway.Service, which starts and stops the Java Gateway.

  This business host performs an additional function: its settings indicate the location of the Java Gateway. When correctly configured, the EnsLib.SAP.Operation business host retrieves those settings and uses them. Thus it is not necessary to set any environment variables.

  Unlike most business hosts in a production, EnsLib.JavaGateway.Service does not handle any production messages.

It is not necessary to manually create message classes to carry the SAP requests within the production. InterSystems provides a CSP page that you can use to generate the message classes.

2

Setup Tasks for the SAP Java Connector

Before you can use the SAP components in a production, you must perform the setup activities discussed in this topic.

To access SAP, it is necessary to provide a username and password. This means that you must also create production credentials that contain an SAP username and password. See Defining Production Credentials.

2.1 Setting Up the Java Gateway

The Java Gateway server runs within a JVM, which can be on the same machine as InterSystems IRIS or on a different machine. Complete the following setup steps on the machine on which the Java Gateway will run:

1. Install the Java Runtime Environment (for example, JRE 1.8.0_67).

2. Make a note of the location of the installation directory for JRE. This is the directory that *contains* the subdirectories bin and lib.

   This is the value that you would use for *JAVA_HOME* environment variable. For example: `c:\Program Files\Java\jre8`

   You use this information later when you configure your production.

3. Also make a note of the Java version. If you are uncertain about the Java version, open a DOS window, go to the bin subdirectory of your Java installation, and enter the following command:

   ```
   java.exe -version
   ```

   You should receive output like the following, depending on your platform:

   ```
   java version "1.8.0_67"
   Java(TM) SE Runtime Environment (build 1.8.0_67-b24)
   Java HotSpot(TM) 64-Bit Server VM (build 23.19-b22, mixed mode)
   ```

It is not necessary to set any environment variables. To access the JVM, InterSystems IRIS uses information contained in the production.

2.2 Installing the SAP JCo Jar File

Obtain, from SAP, the SAP Java Connector 3.x, as appropriate for your operating system. Generally, this is provided as a compressed file. Uncompress it and place the contents in a convenient location. The directory should contain the following items:

- examples subdirectory
- javadoc subdirectory
- Readme.txt file
- sapjco3.dll file
- sapjco3.jar file
- sapjcomanifest.mf file

2.3 Generating Proxy Classes for SAP JCo

To communicate with SAP JCo, your interoperability-enabled namespace must contain proxy classes that represent SAP JCo. To generate these classes, do the following:

1. Start the Java Gateway.

   The easiest way to do this is as follows:

   a. Create a simple production that contains only one business host: EnsLib.JavaGateway.Service. For details about configuring settings for this business host, see EnsLib.JavaGateway.Service Settings.

   b. Start the production, which starts the Java Gateway.

2. In the Terminal, change to your interoperability-enabled namespace and use the **ImportSAP()** method of EnsLib.SAP.BootStrap, as follows:

   ```
   do ##class(EnsLib.SAP.BootStrap).ImportSAP(pFullPathToSAPJarFile,pPort,pAddress)
   ```

   Where:

   - *pFullPathToSAPJarFile* is the full path to the SAP Jar file.
   - *pPort* is the port used by the Java Gateway.
   - *pAddress* is the IP address used by the Java Gateway.

2.4 Testing the SAP Connection

To test the SAP connection, do the following in the Terminal (or in code):

1. Create an instance of EnsLib.SAP.Utils.
2. Set the following properties of that instance. These are string properties unless otherwise noted.

- SAPClient—SAP Client e.g 000.
- SAPUser—Username that has access to the SAP server.
- SAPPassword—Password for the user.
- SAPLanguage
- SAPHost— Host name or IP address of the SAP server.
- SAPSystemNumber—SAP SystemNumber e.g 00.
- JavaGatewayAddress—IP address or name of the machine where the JVM to be used by the Java Gateway server is located.
- JavaGatewayPort—Port used by the Java Gateway.
- SAPTransactionAutoCommit—Specifies whether to execute the BAPI "BAPI_TRANSACTION_COMMIT" after a successful BAPI/RFC-call. This property is %Boolean.

3. Call the PingSAP() method of your instance. This method connects to SAP and performs a dynamic invocation of the STFC_CONNECTION function. It returns a %Status.

3

Using the SAP Java Connector

This topic describes how to add the required components to your production so that it can send requests to SAP. Also see Setup Tasks.

3.1 Basics

Add the following business hosts to your production.

- The business service EnsLib.JavaGateway.Service. Configure this business host as described later in this topic.
- The business operation EnsLib.SAP.Operation.

  Configure this business host as described in the later in this topic.
- One or more business hosts that send SAP request messages to EnsLib.SAP.Operation, as needed.

  Use the message classes that you generated. Your business hosts should create instances of these classes, set properties as applicable, and send the messages to the instance of EnsLib.SAP.Operation.

3.2 Settings for EnsLib.JavaGateway.Service

Configure the settings for EnsLib.JavaGateway.Service so that it can find the Java Gateway. These settings are:

ExternalServerName

The value of this setting should be an external language server name as described in Managing External Server Connections. This setting supersedes the settings in Server and Port and the associated gateway configuration settings such as ClassPath, the heartbeat settings, the UsePassphrase setting, and the JVM setting. Note it is now not possible to start a gateway that uses a passphrase without using an external language server name.

Address

IP address or name of the machine where the JVM to be used by the Java Gateway Server is located.

Port

Port number to which the Java Gateway connects. The default is 55555.

HeartbeatInterval

Number of seconds between each communication with the Java Gateway to check whether it is active. When enabled, the minimum value is 5 seconds and the maximum value is 3600 seconds (1 hour). The default is 10 seconds. A value of 0 disables this feature.

HeartbeatFailureTimeout

Number of seconds without responding to the heartbeat, to consider that the Java Gateway is in failure state. If this value is smaller than the HeartbeatInterval property, the gateway is in failure state every time the Java Gateway communication check fails. The maximum value is 86400 seconds (1 day). The default is 30 seconds.

HeartbeatFailureAction

Action to take if the Java Gateway goes into a failure state. Setting it to Restart (default) causes the Java Gateway to restart. Setting it to Alert generates an alert entry in the Event Log. This is independent of the **Alert on Error** setting.

HeartbeatFailureRetry

Time to wait before retrying the HeartbeatFailureAction if the Java Gateway server goes into failure state, and stays in failure state. The default is 300 seconds (5 minutes). A value of 0 disables this feature, meaning that once there is a failure that cannot be immediately recovered, there are no attempts at automatic recovery.

JavaHome

Location of the JVM (This is the value that you would use for *JAVA_HOME* environment variable). It is used to find the target JVM and assemble the command to start the Gateway.

If there is a default JVM on the machine that is usable without the need to specify its location, you can leave this setting blank.

ClassPath

Class path containing the files to be passed as an argument when starting the JVM. You must include any jar file that define classes you are importing via the Java Gateway. There is no need to include InterSystems' .jar files used by the Java Gateway. If you are specifying file paths containing spaces or multiple files, you should quote the classpath and supply the appropriate separators for your platform.

The following is an example semicolon-separated list of file paths for a Microsoft Windows platform:

```
C:\Library\mygateway.jar;"C:\Jar files\utilities.jar"
```

Note that additional paths for the classpath can be specified in business operations derived from EnsLib.JavaGateway.AbstractOperation. See the property AdditionalPaths in that class.

JVMArgs

Optional arguments to be passed to the Java Virtual Machine (JVM) to include when assembling the command to start the Java Gateway. For example, you can specify system properties: *Dsystemvar=value* or set the maximum heap size: *Xmx256m* and so on, as needed.

JDKVersion

Version of JDK used to select the intended version of the InterSystems .jar files. It is used to assemble the command to start the Java Gateway. For example: `Java 1.7`

Logfile

Fully qualified name of a file to log all communication between the InterSystems IRIS server and the Java Gateway. Usually this setting should be left blank, except when troubleshooting. These messages include acknowledgment of opening and closing connections to the server, as well as any difficulties encountered in mapping Java classes to InterSystems IRIS proxy classes.

JavaDebug

Allow a Java debugger (such as Eclipse or JSwat) to attach. If True, enables Java debugging. The default is False.

JavaDebugPort

Specify the port on which to listen. The default is 8000.

JavaDebugSuspend

If Yes, suspend the JVM on start to wait for the debugger to attach. The default is No.

3.3 Settings for EnsLib.SAP.Operation

EnsLib.SAP.Operation sends requests to SAP JCo, via the Java Gateway. For this business host, specify the following settings:

SAPClient

SAP Client e.g 000.

SAPCredentials

This is the name of the set of production credentials to use when accessing the SAP server. See Defining Production Credentials.

SAPLanguage

SAPHost

Host name or IP address of the SAP server.

SAPSystemNumber

SAP SystemNumber e.g 00.

SAPTransactionAutoCommit

Specifies whether to execute the BAPI "BAPI_TRANSACTION_COMMIT" after a successful BAPI/RFC-call.

SAPResponseHandler

Configuration item in this production that should receive the SAP response.

JavaGatewayConfigItemName

Name of the (required) configuration item that hosts the Java Gateway.

For settings not listed here, see Settings in All Productions.