Java Connectivity

Created by Stefan Hainer, last modified by Stefan Gass on Dec 13, 2017

What is the SAP JCo and how can I get it?

SAP JCo (SAP Java Connector) is a middleware component and a development library that enables a Java application to communicate with a SAP system via SAP's RFC protocol. The SAP JCo supports both directions for communication: inbound remote function calls (Java calls ABAP) and outbound Remote Function Calls (ABAP calls Java).

SAP JCo is available for 2 distinguished use cases, the standalone version and the version integrated in the SAP J2EE Engine. These two JCo variants have a compatible API but are implemented differently, so you must not mix them within one installation.

The integrated JCo is shipped as part of the SAP Application Server Java and is part of the SAP J2EE Engine Core installation packages. The standalone JCo for usage outside a SAP J2EE Engine is available for download on the SAP Service Marketplace at http://service.sap.com/connectors --> SAP Java Connector.

back

What are the JCo RFC Provider and the SAP JRA?

JCo RFC Provider

The JCo RFC Provider (technical internal name: rfcengine) is a service within the J2EE engine. It offers a dispatching functionality for incoming RFC requests to EJBs (Enterprise Java Beans). Internally it uses the SAP JCo for handling the RFC outbound communication (ABAP to Java). The JCo RFC Provider registers itself at a SAP gateway and listens for incoming RFC requests that will then be dispatched to the matching Java Bean. Therefore the name of the Java Bean must be identical to the Remote Function Module Name being called and it must be registered at the JNDI naming service. Furthermore the Java Bean must implement the mandatory method processFunction(JCO.Function func).

SAP Java Resource Adapter (SAP JRA) 1.0

The SAP Java Resource Adapter is a J2EE connector for SAP systems. The SAP JRA implements the JCA (J2EE Connector Architecture) specification 1.0. It internally uses the SAP JCo and wraps the JCo APIs in order to offer the standard APIs specified by the JCA 1.0 (J2EE Connector Architecture). As the JCA 1.0 only specifies a Common Client Interface, the JRA 1.0 can only be used for inbound Remote Function Calls (Java to ABAP).

SAP Java Resource Adapter (SAP JRA) 1.5

The SAP Java Resource Adapter 1.5 is the successor of the SAP JRA 1.0 and implements the JCA (J2EE Connector Architecture) specification 1.5. The JCA specification 1.5 introduces an Inbound Communication Model with a Message Inflow Contract and an EJB Invocation Model. So based on this, SAP JRA 1.5 also adds the capability of doing outbound Remote Function Calls (ABAP to Java) to SAP JRA 1.0.

back

How can I activate / deactivate the JCo trace?

Trace activation with JCo API calls

If you are developing your own JCo application you may use the following JCo API calls to switch on the JCo tracing:
-> JCO.setTraceLevel(int)
-> JCO.setTracePath(String)
As an alternative you may also use the following setProperty calls:
-> JCO.setProperty("jco.trace_level", int)
-> JCO.setProperty("jco.trace_path", String)

We recommend to always offer this tracing ability in your own application by providing an appropriate configuration GUI in order to be able to switch JCo tracing on and off during runtime.

JCo trace activation via Java system properties

At start-up JCo evaluates the Java system properties for a specified tracing configuration. So you are also able to switch on JCo tracing even if the application itself does not offer this JCo trace configuration possibility. Therefore please specify the following Java system properties in the command line:

-> java -Djco.trace_level=<int> -Djco.trace_path=<path> .....

If you would like to change these Java system properties in the SAP J2EE Engine environment please use the ConfigTool for doing this modification.

Some J2EE applications offering JCo and RFC trace activation during runtime:

-> The JCo RFC Provider service GUI (since NW04s)
-> The JRA in the Connector Container service (please see the Managed Connection Properties page)
-> With a specially deployed JRfcTests.ear servlet application (attached to SAP note 628962)

Additional RFC tracing capabilities

Since JCo is internally using the RFC and CPIC libraries, the related traces of these components are also sometimes required for an error analysis.

RFC trace activation on a per connection basis:

Please specify the following logon property
-> for a JCO.Client: "jco.client.trace" must be set to "1"
-> for a JCO.Server: "jco.server.trace" must be set to "1"
or call the appropriate JCo API
-> for a JCO.Client: JCO.Client.setTrace(true)
-> for a JCO.Server: JCO.Server.setTrace(true)
This setting must be done before the connection is opened. It will not have any effect on an already open connection.

RFC trace activation from the communication partner side

Please set the Trace check box option in the relevant symbolic destination using the transaction SM59.

RFC trace activation for all connections with a JCo standalone application

Please set the following environment variables at the operating system level before you start your application:

```
RFC_TRACE=1
RFC_TRACE_DIR=<path>
```

A sample for Windows would look like this:

```
set RFC_TRACE=1
set RFC_TRACE_DIR=C:\tmp\traces
```

RFC trace activation for all connections within the J2EE Engine

Please call the following JCo API
-> JCO.setMiddlewareProperty("jrfc.trace", "1")
or set the following Java system property by using the ConfigTool
-> -Djrfc.trace=1
These settings will be effective only for newly opened connections or - in case of the Java system property - only after restart of the J2EE Engine.

back

What JCo trace level do I need?

JCo offers trace levels from 0 to 10. The amount of traced data increases with the chosen trace level. Each trace level also contains all the trace data from the lower trace levels. For details about the traced data please see the table below. If you are using a high JCo trace level with lots of JCo calls and transferred content, please make sure that enough disk space is available.

Level	Trace content
0	no trace
1	JCo version and runtime environment info + important public API calls
2	+ additional public API calls (e.g. getClient and releaseClient)
3	+ internal middleware calls (JNI / JRfc layer)
4	+ more internal middleware details (e.g. enter/leave API info)
5	+ record memory allocation info + important caller stack trace info (e.g. for removePool, setTraceLevel)
6	+ RFC meta data (name, type, offset, length, import/export-options) + ASCII content data (first 1000 chars of structs / first 5 rows of tables)
7	+ additional Hex values for content data
8	+ full content data dump (no char or row limit)
9	+ Java to/from C marshalling field data and codepage converter calls
10	+ memory leak analysis info (Record ObjectIDs & detailed freeRecord)

back

What are the different trace files for?

These trace files may be created depending on your trace settings:

JCo 2.x standalone:

JCO<date>_<timestamp>.trc : This is the JCo trace produced directly from JCo.

rfc<ProcessID>_<ThreadID>.trc : This is the RFC protocol trace produced by the used RFC library (librfc32.dll / librfccm.so / librfccm.sl) and the trace from the CPIC library which is used by the RFC library.

CPIC<ProcessID> : This is the low level CPIC protocol trace produced by the CPIC library that is statically linked to the RFC library.

dev_rfc.trc : This is the RFC error log that is always produced if an RFC error occurs. It cannot be switched off.

JCo integrated in J2EE engine:

defaultTrace.<#>.trc : This is the J2EE Engine's default application trace. Although JCo is a class library and not an application many applications are logging catched JCo exceptions here.

JCO<date>_<timestamp>.trc : This is the JCo trace produced directly from JCo.

jrfc<ProcessID>_<ThreadID>.trc : This is the RFC protocol trace produced by the underlying Java RFC layer.

dev_server<ServerNodeID>.out : This is the developer trace of the jlaunch executable. Besides other things it also contains the low level CPIC protocol trace produced by the CPIC library that is statically linked to this executable.

dev_jrfc.trc : This is the RFC error log that is always produced if an RFC error occurs. It cannot be switched off.

back

There are always trace files created. How to get rid of this?

The JCO tracing can be activated in several ways. Therefore you have to check different locations if you would like to find out where the trace has been activated or which application switched it on.

Please check the following:

1. Has the trace been activated using system properties? Set the -Djrfc.trace=0 JVM system property in the ConfigTool to reset tracing.
2. Has the trace been activated using the appropriate checkbox in the JCo Provider Service? Check with NWA / VisualAdmin the JCO RFC Provider Service, have a look at the "Special Settings" tab of every single RFC listener
3. Has the trace been activated setting the "jco.client.trace=1" option as logon parameter for a single connection or a connection pool? Reset the option.
4. Has an application activated the trace through setting the property JCO.Client.setTrace(true) or calling JCO.Client.setTrace(true) within the java program coding? Check the application parameters/options.
   Hint: If you have no idea which application may have activated the trace, look into the traced data and check if you are able to identify the application using the traced content, e.g. search for the connection information within the trace.
5. Has the tracing been switched on through trace propagation from the communication partner system? Check the communication partner settings e.g. using transaction SM59 and check for 'export trace' flag.
   Hint: If you don't know the destination, have a look into transaction SE16, table RFCDES and search for all destinations with the enabled RFC tracing (specify RFCOPTIONS string "**T=Y**").
   Caution: If the tracing has been switched on through trace propagation from the communication partner side, the respective JCo RFC server program/application needs to be restarted after changing the trace settings in the SM59 configuration.

back

My JCo standalone program works fine, but creates connection related error entries in syslog and trace? Why?

If your JCO standalone program does not properly close the used connections, error traces and syslog entries may be written at the communication partner side.
Syslog entries like

```
R49 Communication error, CPIC return code 020, SAP return code 223
```

or trace entries in the gateway trace

```
connection to partner broken / Connection reset by peer
```

may be related to a program error within your JCO program. Make sure that you have disconnected (calling: JCO.Client's .disconnect() function) for each JCO.Client where you have called JCO.Client.connect() before. If your program terminates without disconnecting, it looks like an erroneous program termination to the communication partner. Correct your program.

back

Some JCo connections are never closed. Why?

JCo connections are normal RFC connections and therefore have an unlimited lifetime. There is no timeout mechanism that closes an RFC connection automatically. It is always the application that is responsible for disconnecting an open JCO.Client or alternatively releasing it to a JCo pool when it is no longer needed.

back

What are the different JCo pooling parameters for?

Maximum Pool Size

Maximum number of connections which will be kept open by the pool for possible reuse. These connections will be automatically closed if they cannot be reused for more than the "Connection Timeout" period.

Maximum Connections

Maximum number of connections which can be allocated from the pool. This allows you to create more connections as specified by the "Maximum Pool Size" parameter, for example for temporary peak usage times. If the value for "Maximum connections" is less than the value of the parameter "Maximum Pool Size", the parameter will automatically be reset to the value of "Maximum Pool Size". All allocated connections exceeding the "Maximum Pool Size" will be closed immediately, if they are released from the application to the pool again.

Connection Timeout

Defines the time period that a connection pool keeps its connections open for reuse. In order for this timeout to become effective the connection must have been released to the pool. It does not affect

connections that have been requested from the pool and are still under control of the application. A pooled connection is being regarded as timed out and will be automatically closed when it has not been handed out by the pool for the specified "Connection Timeout" time interval. The default value is 10 minutes (600,000 ms). Connections will be handed out by the pool following the LIFO principle (Last-In - First-Out).

Maximum Waiting Time

Defines the maximum time to wait to obtain a requested connection. If the connection pool is exhausted (that means that the "Maximum Connections" limit is reached) and another thread is requesting an additional connection, this is the time that is being waited for some connection to be released by another thread so that that one can be handed out to the waiting thread. If the maximum waiting time is reached, and no connection became available in the mean time, then a JCO.Exception with the key JCO_ERROR_RESOURCE is thrown. The default value for the "Maximum Waiting Time" is 30 seconds (30.000 ms).

back

Where can I find usefull JCo notes?

549268 SAP JCo release and support strategy

628962 How to switch on the SAP JCo trace

723562 SAP JCo standalone, Configuration and Requirements

896317 Trace JRA (Java Resource Adapter)

1066427 Trace JCo RFC Provider Service

460089 Minimum authorization profile for external RFC programs

568271 Lifetime of an RFC connection

back

Where can I find additional information in the SDN?

Codegallery: JCO for Dummys

Codegallery: Implementing JCO based applications

Configure SLD for JCo and Creation of JCo Destinations

Troubleshooting SAP Java Connector

The Java Connector(Jco),Enterprise Connector

SAP Application Hierarchy using JAVA

back

No labels

1 Comment

Guest

Why does "Trobleshooting ...." leads to "JCo for Dummies?" This is not very helpful.

Regards
Patrick