JT400

JT/400 Component

The **`jt400`** component allows you to exchanges messages with an AS/400 system using data queues.

Maven users will need to add the following dependency to their `pom.xml` for this component:

```
<dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-jt400</artifactId>
    <version>x.x.x</version>
    <!-- use the same version as your Camel core version -->
</dependency>
```

URI format

```
jt400://user:password@system/QSYS.LIB/LIBRARY.LIB/QUEUE.DTAQ[?options]
```

To call remote program (**Camel 2.7**)

```
jt400://user:password@system/QSYS.LIB/LIBRARY.LIB/program.PGM[?options]
```

You can append query options to the URI in the following format, `?option=value&option=value&...`

URI options

For the data queue message exchange:

Name	Default value	Description
`ccsid`	default system CCSID	Specifies the CCSID to use for the connection with the AS/400 system.
`format`	`text`	Specifies the data format for sending messages valid options are: `text` (represented by `String`) and `binary` (represented by `byte[]`)
`consumer.delay`	`500`	Delay in milliseconds between each poll.
`consumer.initialDelay`	`1000`	Milliseconds before polling starts.
`consumer.userFixedDelay`	`false`	`true` to use fixed delay between polls, otherwise fixed rate is used. See ScheduledExecutorService in JDK for details.
`guiAvailable`	`false`	**Camel 2.8:** Specifies whether AS/400 prompting is enabled in the environment running Camel.
`keyed`	`false`	**Camel 2.10:** Whether to use keyed or non-keyed data queues.
`searchKey`	`null`	**Camel 2.10:** Search key for keyed data queues.
`searchType`	`EQ`	**Camel 2.10:** Search type which can be a value of `EQ`, `NE`, `LT`, `LE`, `GT`, or `GE`.
`connectionPool`	`AS400ConnectionPool` instance	**Camel 2.10:** Reference to an com.ibm.as400.access.AS400ConnectionPool instance in the Registry. This is used for obtaining connections to the AS/400 system. The look up notation ('#' character) should be used.
`secured`	`false`	**Camel 2.16:** Whether to use SSL connections to the AS/400

For the remote program call (**Camel 2.7**)

Name	Default value	Description
`outputFieldsIdx`		Specifies which fields (program parameters) are output parameters.

`fieldsLength`		Specifies the fields (program parameters) length as in the AS/400 program definition.
`format`	`text`	**Camel 2.10:** Specifies the data format for sending messages valid options are: `text` (represented by `String`) and `binary` (represented by `byte[]`)
`guiAvailable`	`false`	**Camel 2.8:** Specifies whether AS/400 prompting is enabled in the environment running Camel.
`connectionPool`	`AS400ConnectionPool` instance	**Camel 2.10:** Reference to an com.ibm.as400.access.AS400ConnectionPool instance in the Registry. This is used for obtaining connections to the AS/400 system. The look up notation ('#' character) should be used.

Usage

When configured as a consumer endpoint, the endpoint will poll a data queue on a remote system. For every entry on the data queue, a new `Exchange` is sent with the entry's data in the *In* message's body, formatted either as a `String` or a `byte[]`, depending on the format. For a provider endpoint, the *In* message body contents will be put on the data queue as either raw bytes or text.

Connection pool

Available as of Camel 2.10

Connection pooling is in use from Camel 2.10 onwards. You can explicit configure a connection pool on the Jt400Component, or as an uri option on the endpoint.

Remote program call (Camel 2.7)

This endpoint expects the input to be either a String array or byte[] array (depending on format) and handles all the CCSID handling through the native jt400 library mechanisms. A parameter can be *omitted* by passing null as the value in its position (the remote program has to support it). After the program execution the endpoint returns either a String array or byte[] array with the values as they were returned by the program (the input only parameters will contain the same data as the beginning of the invocation)
This endpoint does not implement a provider endpoint!

Example

In the snippet below, the data for an exchange sent to the `direct:george` endpoint will be put in the data queue `PENNYLANE` in library `BEATLES` on a system named `LIVERPOOL`.
Another user connects to the same data queue to receive the information from the data queue and forward it to the `mock:ringo` endpoint.

```
public class Jt400RouteBuilder extends RouteBuilder {
    @Override
    public void configure() throws Exception {
       from("direct:george").to("jt400://GEORGE:EGROEG@LIVERPOOL/QSYS.LIB/BEATLES.LIB/PENNYLANE.DTAQ");
       from("jt400://RINGO:OGNIR@LIVERPOOL/QSYS.LIB/BEATLES.LIB/PENNYLANE.DTAQ").to("mock:ringo");
    }
}
```

Remote program call example (Camel 2.7)

In the snippet below, the data Exchange sent to the direct:work endpoint will contain three string that will be used as the arguments for the program “compute” in the library “assets”. This program will write the output values in the 2nd and 3rd parameters. All the parameters will be sent to the direct:play endpoint.

```
public class Jt400RouteBuilder extends RouteBuilder {
    @Override
    public void configure() throws Exception {
       from("direct:work").to("jt400://GRUPO:ATWORK@server/QSYS.LIB/assets.LIB/compute.PGM?fieldsLength=10,
10,512&ouputFieldsIdx=2,3").to(“direct:play”);
    }
}
```

Writing to keyed data queues

```
from("jms:queue:input")
.to("jt400://username:password@system/lib.lib/MSGINDQ.DTAQ?keyed=true");
```

Reading from keyed data queues

```
from("jt400://username:password@system/lib.lib/MSGOUTDQ.DTAQ?keyed=true&searchKey=MYKEY&searchType=GE")
.to("jms:queue:output");
```

See Also

- Configuring Camel
- Component
- Endpoint
- Getting Started