

Exec

Exec component

Available in Camel 2.3

The exec component can be used to execute system commands.

Dependencies

Maven users need to add the following dependency to their pom.xml

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-exec</artifactId>
  <version>${camel-version}</version>
</dependency>
```

where `${camel-version}` must be replaced by the actual version of Camel (2.3.0 or higher).

URI format

```
exec://executable[?options]
```

where `executable` is the name, or file path, of the system command that will be executed. If executable name is used (e.g. `exec:java`), the executable must in the system path.

URI options

Name	Default value	Description
args	null	The arguments of the executable. The arguments may be one or many whitespace-separated tokens, that can be quoted with " - e.g. <code>args="arg 1" arg2</code> will use two arguments <code>arg 1</code> and <code>arg2</code> . To include the quotes use "" - e.g. <code>args=""arg 1""</code> <code>arg2</code> will use the arguments "arg 1" and <code>arg2</code> .
workingDir	null	The directory in which the command should be executed. If null, the working directory of the current process will be used.
timeout	Long.MAX_VALUE	The timeout, in milliseconds, after which the executable should be terminated. If execution has not completed within the timeout, the component will send a termination request.
outFile	null	The name of a file, created by the executable, that should be considered as its output. If no <code>outFile</code> is set, the standard output (stdout) of the executable will be used instead.
binding	a DefaultExecBinding instance	A reference to a <code>org.apache.commons.exec.ExecBinding</code> in the Registry .
commandExecutor	a DefaultCommandExecutor instance	A reference to a <code>org.apache.commons.exec.ExecCommandExecutor</code> in the Registry that customizes the command execution. The default command executor utilizes the commons-exec library , which adds a shutdown hook for every executed command.
useStderrOnEmptyStdout	false	A boolean indicating that when <code>stdout</code> is empty, this component will populate the Camel Message Body with <code>stderr</code> . This behavior is disabled (false) by default.

Message headers

The supported headers are defined in `org.apache.camel.component.exec.ExecBinding`.

Name	Type	Message	Description
ExecBinding.EXEC_COMMAND_EXECUTABLE	String	in	The name of the system command that will be executed. Overrides <code>executable</code> in the URI.
ExecBinding.EXEC_COMMAND_ARGS	java.util.List<String>	in	Command-line arguments to pass to the executed process. The arguments are used literally - no quoting is applied. Overrides any existing <code>args</code> in the URI.
ExecBinding.EXEC_COMMAND_ARGS	String	in	Camel 2.5: The arguments of the executable as a Single string where each argument is whitespace separated (see <code>args</code> in URI option). The arguments are used literally, no quoting is applied. Overrides any existing <code>args</code> in the URI.
ExecBinding.EXEC_COMMAND_OUT_FILE	String	in	The name of a file, created by the executable, that should be considered as its output. Overrides any existing <code>outFile</code> in the URI.

ExecBinding. EXEC_COMMAND_TIMEOUT	long	in	The timeout, in milliseconds, after which the executable should be terminated. Overrides any existing <code>timeout</code> in the URI.
ExecBinding. EXEC_COMMAND_WORKING_DIR	String	in	The directory in which the command should be executed. Overrides any existing <code>workingDir</code> in the URI.
ExecBinding. EXEC_EXIT_VALUE	int	out	The value of this header is the <i>exit value</i> of the executable. Non-zero exit values typically indicate abnormal termination. Note that the exit value is OS-dependent.
ExecBinding.EXEC_STDERR	java.io. InputStream	out	The value of this header points to the standard error stream (<code>stderr</code>) of the executable. If no <code>stderr</code> is written, the value is <code>null</code> .
ExecBinding. EXEC_USE_STDERR_ON_EMPTY_STDOUT	boolean	in	Indicates that when <code>stdout</code> is empty, this component will populate the Camel Message Body with <code>stderr</code> . This behavior is disabled (<code>false</code>) by default.

Message body

If the Exec component receives an `in` message body that is convertible to `java.io.InputStream`, it is used to feed input to the executable via its `stdin`. After execution, [the message body](#) is the result of the execution,- that is, an `org.apache.camel.components.exec.ExecResult` instance containing the `stdout`, `stderr`, `exit value`, and `out file`. This component supports the following `ExecResult` [type converters](#) for convenience:

From	To
ExecResult	java.io.InputStream
ExecResult	String
ExecResult	byte []
ExecResult	org.w3c.dom.Document

If an `out file` is specified (in the endpoint via `outFile` or the message headers via `ExecBinding.EXEC_COMMAND_OUT_FILE`), converters will return the content of the `out file`. If no `out file` is used, then this component will convert the `stdout` of the process to the target type. For more details, please refer to the [usage examples](#) below.

Usage examples

Executing word count (Linux)

The example below executes `wc` (word count, Linux) to count the words in file `/usr/share/dict/words`. The word count (output) is written to the standard output stream of `wc`.

```
from("direct:exec")
.to("exec:wc?args=- -words /usr/share/dict/words")
.process(new Processor() {
    public void process(Exchange exchange) throws Exception {
        // By default, the body is ExecResult instance
        assertIsInstanceOf(ExecResult.class, exchange.getIn().getBody());
        // Use the Camel Exec String type converter to convert the ExecResult to String
        // In this case, the stdout is considered as output
        String wordCountOutput = exchange.getIn().getBody(String.class);
        // do something with the word count
    }
});
```

Executing java

The example below executes `java` with 2 arguments: `-server` and `-version`, provided that `java` is in the system path.

```
from("direct:exec")
.to("exec:java?args=-server -version")
```

The example below executes `java` in `c:\temp` with 3 arguments: `-server`, `-version` and the system property `user.name`.

```
from("direct:exec")
.to("exec:c:/program files/jdk/bin/java?args=-server -version -Duser.name=Camel&workingDir=c:/temp")
```

Executing Ant scripts

The following example executes [Apache Ant](#) (Windows only) with the build file `CamelExecBuildFile.xml`, provided that `ant.bat` is in the system path, and that `CamelExecBuildFile.xml` is in the current directory.

```
from("direct:exec")
.to("exec:ant.bat?args=-f CamelExecBuildFile.xml")
```

In the next example, the `ant.bat` command redirects its output to `CamelExecOutFile.txt` with `-l`. The file `CamelExecOutFile.txt` is used as the out file with `outFile=CamelExecOutFile.txt`. The example assumes that `ant.bat` is in the system path, and that `CamelExecBuildFile.xml` is in the current directory.

```
from("direct:exec")
.to("exec:ant.bat?args=-f CamelExecBuildFile.xml -l CamelExecOutFile.txt&outFile=CamelExecOutFile.txt")
.process(new Processor() {
    public void process(Exchange exchange) throws Exception {
        InputStream outFile = exchange.getIn().getBody(InputStream.class);
        assertInstanceOf(InputStream.class, outFile);
        // do something with the out file here
    }
});
```

Executing echo (Windows)

Commands such as `echo` and `dir` can be executed only with the command interpreter of the operating system. This example shows how to execute such a command - `echo` - in Windows.

```
from("direct:exec").to("exec:cmd?args=/C echo echoString")
```

See Also

- [Configuring Camel](#)
- [Component](#)
- [Endpoint](#)
- [Getting Started](#)