Esper

Esper

The Esper component supports the Esper Library for Event Stream Processing. The **camel-esper** library is provided by the Camel Extra project which hosts all *GPL related components for Camel.

URI format

```
esper:name[?options]
```

When consuming from an Esper endpoint you must specify a **pattern** or **eql** statement to query the event stream.

Pattern example:

```
from("esper://cheese?pattern=every event=MyEvent(bar=5)")
  .to("activemq:Foo");
```

EQL example:

```
from("esper://esper-dom?eql=insert into DomStream select * from org.w3c.dom.Document")
  .to("log://esper-dom?level=INFO");
from("esper://esper-dom?eql=select childNodes from DomStream")
  .to("mock:results");
```

Options

Name	Default Value	Description
`configured`	false	Available as of camel-extra 2.11.3: If flag is set to 'true' the default Esper configuration file (esper.cfg.xml) will be used. To configure Esper via a configuration file, please refer to the Esper documentation
`pattern`		The Esper Pattern expression as a String to filter events
`eql`		The Esper EQL expression as a String to filter events

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

EsperMessage

From **Camel 2.12** onwards the esper consumer stores new and old events in the `org.apacheextras.camel.component.esper.EsperMessage` message as the input Message on the Exchange. You can get access to the esper event beans from java code with:

```
  EventBean newEvent = exchange.getIn(EsperMessage.class).getNewEvent();
  EventBean oldEvent = exchange.getIn(EsperMessage.class).getOldEvent();
```

By default if you get the body of `org.apacheextras.camel.component.esper.EsperMessage` it returns the new `EventBean` as in previous versions.

Demo

There is a demo which shows how to work with ActiveMQ, Camel and Esper in the Camel Extra project

See Also

- Configuring Camel
- Component
- Endpoint
- Getting Started

- Esper Camel Demo