APNS

Apns Component

Available as of Camel 2.8

The **apns** component is used for sending notifications to iOS devices. The apns components use javapns library.
The component supports sending notifications to Apple Push Notification Servers (APNS) and consuming feedback from the servers.

The consumer is configured with 3600 seconds for polling by default because it is a best practice to consume feedback stream from Apple Push Notification Servers only from time to time. For example: every 1 hour to avoid flooding the servers.

The feedback stream gives informations about inactive devices. You only need to get this informations every some hours if your mobile application is not a heavily used one.

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

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

URI format

To send notifications:

```
apns:notify[?options]
```

To consume feedback:

```
apns:consumer[?options]
```

Options

Producer

Property	Default	Description
`tokens`		Empty by default. Configure this property in case you want to statically declare tokens related to devices you want to notify. Tokens are separated by comma.

Consumer

Property	Default	Description
`delay`	`3600`	Delay in seconds between each poll.
`initialDelay`	`10`	Seconds before polling starts.
`timeUnit`	`SECONDS`	Time Unit for polling.
`userFixedDelay`	`true`	If `true`, use fixed delay between pools, otherwise fixed rate is used. See ScheduledExecutorService in JDK for details.

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

Component

The `ApnsComponent` must be configured with a `com.notnoop.apns.ApnsService`. The service can be created and configured using the `org.apache.camel.component.apns.factory.ApnsServiceFactory`. See further below for an example. And as well in the test source code.

Exchange data format

When Camel will fetch feedback data corresponding to inactive devices, it will retrieve a List of InactiveDevice objects. Each InactiveDevice object of the retrieved list will be setted as the In body, and then processed by the consumer endpoint.

Message Headers

Camel Apns uses these headers.

Property	Default	Description
`CamelApnsTokens`		Empty by default.
`CamelApnsMessageType`	`STRING, PAYLOAD, APNS_NOTIFICATION`	In case you choose PAYLOAD for the message type, then the message will be considered as a APNS payload and sent as is. In case you choose STRING, message will be converted as a APNS payload. From **Camel 2.16** onwards APNS_NOTIFICATION is used for sending message body as com.notnoop.apns.ApnsNotification types.

ApnsServiceFactory builder callback

`ApnsServiceFactory` comes with the empty callback method that could be used to configure (or even replace) the default `ApnsServiceBuilder` instance. The signature of the method could look as follows:

```
protected ApnsServiceBuilder configureServiceBuilder(ApnsServiceBuilder serviceBuilder);
```

And could be used like as follows:

```
ApnsServiceFactory proxiedApnsServiceFactory = new ApnsServiceFactory(){

  @Override
  protected ApnsServiceBuilder configureServiceBuilder(ApnsServiceBuilder serviceBuilder) {
    return serviceBuilder.withSocksProxy("my.proxy.com", 6666);
  }

};
```

Samples

Camel Xml route

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:camel="http://camel.apache.org/schema/spring"
       xsi:schemaLocation="
        http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-
2.5.xsd
        http://camel.apache.org/schema/spring http://camel.apache.org/schema/spring/camel-spring.xsd">

        <!-- Replace by desired values -->
        <bean id="apnsServiceFactory" class="org.apache.camel.component.apns.factory.ApnsServiceFactory">

                <!-- Optional configuration of feedback host and port -->
                <!-- <property name="feedbackHost" value="localhost" /> -->
                <!-- <property name="feedbackPort" value="7843" /> -->

                <!-- Optional configuration of gateway host and port -->
                <!-- <property name="gatewayHost" value="localhost" /> -->
                <!-- <property name="gatewayPort" value="7654" /> -->

                <!-- Declaration of certificate used -->
                <!-- from Camel 2.11 onwards you can use prefix: classpath:, file: to refer to load the
certificate from classpath or file. Default it classpath -->
                <property name="certificatePath" value="certificate.p12" />
                <property name="certificatePassword" value="MyCertPassword" />

                <!-- Optional connection strategy - By Default: No need to configure -->
                <!-- Possible options: NON_BLOCKING, QUEUE, POOL or Nothing -->
                <!-- <property name="connectionStrategy" value="POOL" /> -->
                <!-- Optional pool size -->
                <!-- <property name="poolSize" value="15" /> -->

                <!-- Optional connection strategy - By Default: No need to configure -->
                <!-- Possible options: EVERY_HALF_HOUR, EVERY_NOTIFICATION or Nothing (Corresponds to NEVER
javapns option) -->
                <!-- <property name="reconnectionPolicy" value="EVERY_HALF_HOUR" /> -->
        </bean>

        <bean id="apnsService" factory-bean="apnsServiceFactory" factory-method="getApnsService" />

        <!-- Replace this declaration by wanted configuration -->
        <bean id="apns" class="org.apache.camel.component.apns.ApnsComponent">
                <property name="apnsService" ref="apnsService" />
        </bean>

        <camelContext id="camel-apns-test" xmlns="http://camel.apache.org/schema/spring">
                <route id="apns-test">
                        <from uri="apns:consumer?initialDelay=10&amp;delay=3600&amp;timeUnit=SECONDS" />
                        <to uri="log:org.apache.camel.component.apns?showAll=true&amp;multiline=true" />
                        <to uri="mock:result" />
                </route>
        </camelContext>

</beans>
```

Camel Java route

Create camel context and declare apns component programmatically

```
    protected CamelContext createCamelContext() throws Exception {
        CamelContext camelContext = super.createCamelContext();

        ApnsServiceFactory apnsServiceFactory = new ApnsServiceFactory();
        apnsServiceFactory.setCertificatePath("classpath:/certificate.p12");
        apnsServiceFactory.setCertificatePassword("MyCertPassword");

        ApnsService apnsService = apnsServiceFactory.getApnsService(camelContext);

        ApnsComponent apnsComponent = new ApnsComponent(apnsService);
        camelContext.addComponent("apns", apnsComponent);

        return camelContext;
    }
```

ApnsProducer - iOS target device dynamically configured via header: `"CamelApnsTokens"`

```
    protected RouteBuilder createRouteBuilder() throws Exception {
        return new RouteBuilder() {
            public void configure() throws Exception {
                from("direct:test")
                    .setHeader(ApnsConstants.HEADER_TOKENS, constant(IOS_DEVICE_TOKEN))
                    .to("apns:notify");
                }
        }
    }
```

ApnsProducer - iOS target device statically configured via uri

```
    protected RouteBuilder createRouteBuilder() throws Exception {
        return new RouteBuilder() {
            public void configure() throws Exception {
                from("direct:test").
                to("apns:notify?tokens=" + IOS_DEVICE_TOKEN);
            }
        };
    }
```

ApnsConsumer

```
from("apns:consumer?initialDelay=10&delay=3600&timeUnit=SECONDS")
    .to("log:com.apache.camel.component.apns?showAll=true&multiline=true")
    .to("mock:result");
```

See Also

- Component
- Endpoint
- Blog about using APNS (in french)