Zookeeper

ZooKeeper

Available as of Camel 2.9

The ZooKeeper component allows interaction with a ZooKeeper cluster and exposes the following features to Camel:

1. Creation of nodes in any of the ZooKeeper create modes.
2. Get and Set the data contents of arbitrary cluster nodes. (data being set must be convertable to byte[])
3. Create and retrieve the list the child nodes attached to a particular node.
4. A Distributed RoutePolicy that leverages a Leader election coordinated by ZooKeeper to determine if exchanges should get processed.

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

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

URI format

```
zookeeper://zookeeper-server[:port][/path][?options]
```

The path from the uri specifies the node in the ZooKeeper server (aka znode) that will be the target of the endpoint.

Options

Name	Default Value	Description
`path`		The node in the ZooKeeper server (aka znode)
`listChildren`	`false`	Whether the children of the node should be listed
`repeat`	`false`	Should changes to the znode be 'watched' and repeatedly processed.
`backoff`	`5000`	The time interval to backoff for after an error before retrying.
`timeout`	`5000`	The time interval to wait on connection before timing out.
`create`	`false`	Should the endpoint create the node if it does not currently exist.
`createMode`	`EPHEMERAL`	The create mode that should be used for the newly created node (see below).
`sendEmptyMessageOnDelete`	`true`	**Camel 2.10:** Upon the delete of a znode, should an empty message be send to the consumer

Use cases

Reading from a znode.

The following snippet will read the data from the znode '/somepath/somenode/' provided that it already exists. The data retrieved will be placed into an exchange and passed onto the rest of the route.

```
from("zookeeper://localhost:39913/somepath/somenode").to("mock:result");
```

if the node does not yet exist then a flag can be supplied to have the endpoint await its creation

```
from("zookeeper://localhost:39913/somepath/somenode?awaitCreation=true").to("mock:result");
```

Reading from a znode - (additional Camel 2.10 onwards)

When data is read due to a WatchedEvent received from the ZooKeeper ensemble, the CamelZookeeperEventType header holds ZooKeeper's EventType value from that WatchedEvent. If the data is read initially (not triggered by a WatchedEvent) the CamelZookeeperEventType header will not be set.

Writing to a znode.

The following snippet will write the payload of the exchange into the znode at '/somepath/somenode/' provided that it already exists

```
from("direct:write-to-znode").to("zookeeper://localhost:39913/somepath/somenode");
```

For flexibility, the endpoint allows the target znode to be specified dynamically as a message header. If a header keyed by the string 'CamelZooKeeperNode' is present then the value of the header will be used as the path to the znode on the server. For instance using the same route definition above, the following code snippet will write the data not to '/somepath/somenode' but to the path from the header '/somepath/someothernode'. **Notice** the testPayload must be convertable to byte[] as the data stored in ZooKeeper is byte based.

```
Object testPayload = ...
template.sendBodyAndHeader("direct:write-to-znode", testPayload, "CamelZooKeeperNode", "/somepath
/someothernode");
```

To also create the node if it does not exist the 'create' option should be used.

```
from("direct:create-and-write-to-znode").to("zookeeper://localhost:39913/somepath/somenode?create=true");
```

Starting **version 2.11** it is also possible to **delete** a node using the header 'CamelZookeeperOperation' by setting it to 'DELETE'.

```
from("direct:delete-znode").setHeader(ZooKeeperMessage.ZOOKEEPER_OPERATION, constant("DELETE")).to
("zookeeper://localhost:39913/somepath/somenode");
```

or equivalently

```
<route>
  <from uri="direct:delete-znode" />
  <setHeader headerName="CamelZookeeperOperation">
     <constant>DELETE</constant>
  </setHeader>
  <to uri="zookeeper://localhost:39913/somepath/somenode" />
</route>
```

ZooKeeper nodes can have different types; they can be 'Ephemeral' or 'Persistent' and 'Sequenced' or 'Unsequenced'. For further information of each type you can check here. By default endpoints will create unsequenced, ephemeral nodes, but the type can be easily manipulated via a uri config parameter or via a special message header. The values expected for the create mode are simply the names from the CreateMode enumeration

- PERSISTENT
- PERSISTENT_SEQUENTIAL
- EPHEMERAL
- EPHEMERAL_SEQUENTIAL

For example to create a persistent znode via the URI config

```
from("direct:create-and-write-to-persistent-znode").to("zookeeper://localhost:39913/somepath/somenode?
create=true&createMode=PERSISTENT");
```

or using the header 'CamelZookeeperCreateMode'. **Notice** the testPayload must be convertable to byte[] as the data stored in ZooKeeper is byte based.

```
Object testPayload = ...
template.sendBodyAndHeader("direct:create-and-write-to-persistent-znode", testPayload,
"CamelZooKeeperCreateMode", "PERSISTENT");
```

ZooKeeper enabled Route policy.

ZooKeeper allows for very simple and effective leader election out of the box; This component exploits this election capability in a RoutePolicy to control when and how routes are enabled. This policy would typically be used in fail-over scenarios, to control identical instances of a route across a cluster of Camel based servers. A very common scenario is a simple 'Master-Slave' setup where there are multiple instances of a route distributed across a cluster but only one of them, that of the master, should be running at a time. If the master fails, a new master should be elected from the available slaves and the route in this new master should be started.

The policy uses a common znode path across all instances of the RoutePolicy that will be involved in the election. Each policy writes its id into this node and zookeeper will order the writes in the order it received them. The policy then reads the listing of the node to see what postion of its id; this postion is used to determine if the route should be started or not. The policy is configured at startup with the number of route instances that should be started across the cluster and if its position in the list is less than this value then its route will be started. For a Master-slave scenario, the route is configured with 1 route instance and only the first entry in the listing will start its route. All policies watch for updates to the listing and if the listing changes they recalculate if their route should be started. For more info on Zookeeper's Leader election capability see this page.

The following example uses the node '/someapplication/somepolicy' for the election and is set up to start only the top '1' entries in the node listing i.e. elect a master

```
ZooKeeperRoutePolicy policy = new ZooKeeperRoutePolicy("zookeeper:localhost:39913/someapp/somepolicy", 1);
from("direct:policy-controlled").routePolicy(policy).to("mock:controlled");
```

See Also

- Configuring Camel
- Component
- Endpoint
- Getting Started