配置Endpoint

Endpoints用于连接服务。Endpoint是一个特别的通道，在这个通道上一个服务可以接收到另一个服务发送的消息。例如，一个采购组件通过HTTP可以接收到订单请求。一旦订单被组件处理完毕，一个JMS消息会通过一个topic被发送出去通知审计系统，这时，一个response就会通过HTTP被发送回来。

这里描述如何配置一个Endpoint. 想了解更多的关于Endpoint的属性、元素配置，请看[Endpoint Configuration Reference](http://www.mulesoft.org/documentation/display/MULE3USER/Endpoint+Configuration+Reference)。

**基础配置**

In its most basic form, an endpoint consists of a [transport](http://www.mulesoft.org/documentation/display/MULE3USER/Connecting+Using+Transports) and a transport-specific channel/destination/resource used to identify the channel and location where two services can exchange information. For example:

在最基础的配置方式中，一个Endpoint包含一个transport 和一个特定的transport通道/目标/源。。。。

**URI-style Endpoints**

<inbound-endpoint address="udp://localhost:65432"/>

<jetty:inbound-endpoint address="http://localhost:60211/mycomponent1" exchange-pattern="request-response" />

<outbound-endpoint address="smtp://user:secret@smtp.host"/>

<inbound-endpoint address="jms://test.queue"/>

习惯上，就像上面的例子一样，endpoints被指定为一个URI。这种形式还是被支持的，但是否真的就说明它更实用，这就取决于你的应用了。

然而，在Mule 2.0，推荐的做法是通过transport-specific命令空间去指定endpoints，就像下面的例子。

**Transport-specific Endpoints**

<file:inbound-endpoint path="./.mule/in"

comparator="org.mule.transport.file.comparator.OlderFirstComparator" reverseOrder="true"/>

<ssl:endpoint name="clientEndpoint" host="localhost" port="60198"/>

<jetty:endpoint name="serverEndpoint" host="localhost" port="60203" path="services/Foo" />

<imaps:endpoint name="global1s" host="localhost" password="secret" port="123" user="bob"/>

<rmi:endpoint name="BadType" host="localhost" port="1099" object="MatchingUMO" method="reverseString"/>

<quartz:endpoint name="qEP6" repeatCount="10" repeatInterval="1000" jobName="job"/>

<jms:inbound-endpoint queue="test.queue"/>

**动态Endpoints**

从Mule 3开始，outbound endpoint可以是动态的。这意味 着endpoint的URI可以是一个表达式。表达式的值在消息发送之前已经确定。这就允许消息的目标由消息的内容或者其它的属性值来决定。Dynamic endpoint也可以使用上面所示的格式。

**Dynamic Endpoints**

<outbound-endpoint address="smtp://user:secret@#[header:OUTBOUND:host]"/>

<http:outbound-endpoint host="localhost" port="#[header:INBOUND:portNumber]" path="orderService"/>

<jms:outbound-endpoint host="localhost" queue="#[registry:defaultJmsQueue]"/>

The only part of an endpoint URI that cannot be dynamic is the scheme. Don't do this:

在endpoint URI中，唯一不能动态配置的就是scheme，千万不要像下面这样：

**Illegal Dynamic Endpoint**

<outbound-endpoint address="#[header:INBOUND:endpointType]//localhost:8080/service"/>

**连接器**

很多情况下，与endpoint相关的连接器，可以简单地假定是基于transport和隐式创建的。然而，如果同一个transport中存在多个connector，或者对connector中使用了非默认的设置， 那么你必须使用connector-ref属性从endpoint中引用connector。

**Specifying a connector**

<inbound-endpoint address="tcp://localhost:65432" connector-ref="tcpConnector1"/>

<tcp:inbound-endpoint host="localhost" port="65433" connector-ref="tcpConnector2"/>

**属性**

Endpoint中的属性可以用来自定义行为。在endpoint上设置的任何属性会覆盖掉关联的transport的connector的默认属性。例如，一个SMTP外向endpoint可以设置formAddress的值为workflow1以覆盖connector的默认值sysadmin。如果使用transport-specific endpoints，那么在XML schema中任何标准的endpoint属性都会成为有效字段。也可以指定一个非标准属性。例如：

**Setting properties**

<!-- Standard properties -->

<quartz:endpoint name="qEP6" repeatCount="10" repeatInterval="1000" jobName="job"/>

<!-- Non-standard properties -->

<quartz:endpoint name="qEP7" jobName="job2">

<property key="actionOnTimeout" value="self-destruct"/>

<property key="precision" value="2.5"/>

</quartz:endpoint>

**交换模式**

默认地，endpoints都是单向的；这就是说，它们只接收（或发送）消息，但不返回（或接收）responses。想要设置endpoint监听respoonse，你只要设置exchange-pattern="request-response"。HTTP/S,SSL,TCP还有servlet endpoints都不用设置这个值，因为request-response就是它们的默认值。

事务

在endpoint接收到或发送event时，可以开始或提交事务。Endpoint必需是同步的，事务支持在很大程度上取决于所使用的特定的transport。想了解更信息请看[Transaction Management](http://www.mulesoft.org/documentation/display/MULE3USER/Transaction+Management)。

**Transaction example**

<jms:inbound-endpoint queue="in">

<jms:transaction action="BEGIN\_OR\_JOIN"/>

</jms:inbound-endpoint>

**编码**

Endpoind使用endcoding来转换消息内容。在inbound endpoint中，它被用来把接收到的字节转换成字符。在outbound endpoint中，它用来把将要发送的字符转换成字节。如果endpoint中没有设置encoding，则会使用Mule配置中的默认endcoding，就是默认为UTF-8。

**Encoding example**

<inbound-endpoint address="tcp://localhost:65432" encoding="iso-8859-1"/>.

**Mime类型**

Mime类型与endpoint的消息关联。当设置在inbound endpoint上，它表示期待接收的消息的消息类型。接收不同的MIME类型的消息将导致一个异常。当设置在outbound endpoint上，其结果是在所有待发邮件中设置MIME类型

**MimeType example**

<inbound-endpoint address="tcp://localhost:65432" mimeType="text/xml"/>

**再投递策略**

Inbound endpoint上可以设置再投递策略。跟可以在JMS brokers上设置最大再投递数相似，而且解决了一个相似的问题：如果有异常导致读取消息被回滚一遍又一遍，该如何避免无限循环呢？这里有一个例子：

**MimeType example**

<flow name ="syncFlow" processing-strategy="synchronous">

<file:inbound-endpoint path="/tmp/file2ftp/ftp-home/dirk">

<idempotent-redelivery-policy maxRedeliveryCount="3">

<dead-letter-queue>

<vm:outbound-endpoint path="error-queue" />

</dead-letter-queue>

</idempotent-redelivery-policy>

</file:inbound-endpoint>

...

如果在flow后面的东西抛出异常，文件不会被消费掉，而是会被重新处理。idempotent-redelivery-policy确保它不会被重新处超过3次；在那之后，它会被发送到vm：error-queue, 在那里它会被当作错误事件处理掉。

**消息处理器**

消息处理器是什么？它是任何带有Mule消息和处理（转换，过滤，分割等）Mule消息的东西的一个简单接口。实现这个简单的接口的最大优势之一是，消息处理器可以链接在一起，以任何顺序，可以有任意数量，他们可以很容易地交换。这些事情在Mule 3之前不可能发生的。

在endpoint中，以下消息处理器是允许的：

* 转换器
* 过滤器
* 安全过滤器
* 聚合器
* 分离器
* 自定义消息处理器

你可以把任意多个这些消息处理器作为子元素放在一个endpoint(inbound或outbound)中，它们会按照排列的顺序作用在所有经过endpoint的消息上。

.在一个同步的outbound endpoint中，涉及到一个response消息，任何数量的消息处理器也可以被放进一个response包装器中，它们将按照排列的次序应用在response消息上。

请注意，这些元素可以声明为局部（即嵌入到endpoint中）或全局（参考通过一个ref=“foo”的属性）。

**转换器**

转换器可以在endpoint上配置，用于封闭转换逻辑，以使得可以根据需求而被重复使用。

用子消息处理器元素在endpoint上配置转换器。配置在inbound endpoint上的转换器用来转换接收到的消息，而在outbound endpoint上它被用来转换将要发送出去的消息。

Response转换器可以嵌套在<resopnse>元素里面。配置在inbound endpoint上的转换器作用在消息发送回transport前，而在outbound endpoint上的转换器则作用在

所有的消息处理器都配置在endpoint上，它们配置的顺序就是它们执行的顺序。

<inbound-endpoint address="file://./test-data/in">

<xml-to-object-transformer/>

<expression-filter expression=""/>

<transformer ref="ExceptionBeanToErrorMessage"/>

<response>

<custom-transformer class=""/>

</response>

</inbound-endpoint>

在上面的例子中，你可以看到两个请求转换器配置好了，一个会在表达式过滤器之前运行，而另一个在过滤器之后运行。配置在<response>元素里面的自定义转换器会作用在response消息上。

**全局endpoint**

在上面的endpoints例子中，尽管全局转换器可以使用{{<transformer ref=””/>元素从endpoints中引用，但还支持一种快捷方式。

 transformer-refs 和 responseTransformer-refs属性可以快捷简单地引用全局endpoints.

<inbound-endpoint address="file://./test-data/in" transformer-refs="globalTransformer1 globalTransformer2" responseTransformer-refs="globalTransformer2"/>

所有以这种方式引用的转换器会被添加到消息处理器列表的最后，也会在最后才运行。如果你想让它在其它东西之前运行，像过滤器或者需要以指定的顺序使用全局和局部endpoints, 那么你就需要使用<transformer>元素了。

过滤器

Endpoint中可以包含一个过滤器，用于选择性地忽略掉某些消息。过滤器可以是transport-spicific的，比如JMS选择器或者文件过滤器，又或者可以是一个通用的过滤器，如JXPath.。并不是所有的transports都支持过滤，而且在一个endpoint中使用某些transports设置过滤器，会出现unsupportOperationException。更多信息请看 [Using Filters](http://www.mulesoft.org/documentation/display/MULE3USER/Using+Filters)。

**Filter example**

<jms:endpoint queue="in.queue">

<jms:selector expression="JMSPriority > 5"/>

</jms:endpoint>

<vm:endpoint name="fruitBowlEndpoint" path="fruitBowlPublishQ">

<message-property-filter pattern="foo=bar"/>

</vm:endpoint>

**其它消息处理器**

尽管过滤器和转换器是endpoint中最常用的消息处理器，但你也可以简便地配置其它的消息处理器，更多关于有效的消息处理器，请看 [Message Sources and Message Processors](http://www.mulesoft.org/documentation/display/MULE3USER/Message+Sources+and+Message+Processors) page。

**Endpoint的使用**

Endpoint可以在以下地方使用：

* Inbound路由器
* Outbound路由器
* 服务
* 异常策略

Inbound路由器

请看 [Inbound Routers]。

**Inbound router**

<service name="Receiver">

<inbound>

<vm:inbound-endpoint path="inbound.channel"/>

<wire-tap-router>

<vm:outbound-endpoint path="tapped.channel"/>

</wire-tap-router>

</inbound>

<component class="com.acme.SomeService"/>

</service>

**Outbound路由器**

请看 [Outbound Routers].

**Outbound routers**

<service name="MessageChunker">

<inbound>

<jms:inbound-endpoint queue="big.messages"/>

</inbound>

<outbound>

<message-chunking-router messageSize="10">

<jms:outbound-endpoint queue="small.chunks"/>

</message-chunking-router>

</outbound>

</service>

<service name="LenderGatewayService">

<inbound>

<inbound-endpoint ref="LenderGateway" />

</inbound>

<outbound>

<chaining-router>

<outbound-endpoint ref="LenderService" />

<outbound-endpoint ref="BankingGateway" transformer-refs="SetLendersAsRecipients ObjectToJMSMessage" />

</chaining-router>

</outbound>

</service>

**服务**

作为一种快捷方式，在某些情况下没有路由器的endpoint可以直接配置在服务中。

**Implicit router**

<service name="Echo">

<inbound>

<!-- Inbound router is implicit -->

<stdio:inbound-endpoint system="IN"/>

</inbound>

<echo-component/>

<outbound>

<!-- Outbound router is explicit -->

<pass-through-router>

<stdio:outbound-endpoint system="OUT"/>

</pass-through-router>

</outbound>

</service>

**异常策略**

一个单错误的endpoint可以配置在一个异常策略上，请看[Error Handling](http://www.mulesoft.org/documentation/display/MULE3USER/Error+Handling)。

**Exception strategy**

<service name="dataService">

<inbound>

...cut...

</inbound>

<component class="com.acme.DataProcessor"/>

<outbound>

...cut...

</outbound>

<default-service-exception-strategy>

<jms:outbound-endpoint queue="error.queue"/>

</default-service-exception-strategy>

</service>

**全局endpoints**

要拥有一个组织得很好的配置文件，全局endpoint是一个推荐的最佳实践，但不是必需的。

**Global endpoint example**

<file:endpoint name="fileReader" reverseOrder="true" comparator="org.mule.transport.file.comparator.OlderFirstComparator"/>

...cut...

<model>

<service name="Priority1">

<file:inbound-endpoint ref="fileReader" path="/var/prio1"/>

...cut...

</service>

<service name="Priority2">

<file:inbound-endpoint ref="fileReader" path="/var/prio2"/>

...cut...

</service>

</model>

在上面的例子中，”fileReader” endpoint作为inbound endpoint的一个模板使用。reverseOrder andcomparator属性只需要声明一次，但是inbound endpoint中的path属性都 是不同的。