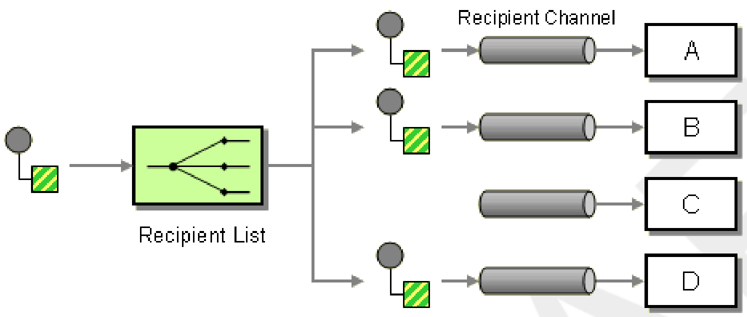
Recipient List

Overview

A *recipient list* is a type of router that sends each incoming message to multiple different destinations. In addition, a recipient list typically requires that the list of recipients be calculated at run time.

Figure 17. Recipient List Pattern



Recipient list with fixed destinations

The simplest kind of recipient list is where the list of destinations is fixed and known in advance and the exchange pattern is *InOnly*. In this case, you can hardwire the list of destinations into the `to()` Java DSL command.

Note

The examples given here, for the recipient list with fixed destinations, work *only* for the *InOnly* exchange pattern (similar to a pipeline on page 27). If you want to create a recipient list for exchange patterns with *Out* messages, use the multicast pattern instead.

Java DSL example

The following example shows how to route an *InOnly* exchange from a source endpoint, `queue:a`, to a fixed list of destinations:

```
from("seda:a").to("seda:b", "seda:c", "seda:d");
```

XML configuration example

The following example shows how to configure the same route in XML:

```
<camelContext id="buildStaticRecipientList" xmlns="http://act
ivemq.apache.org/camel/schema/spring">
  <route>
    <from uri="seda:a"/>
    <to uri="seda:b"/>
    <to uri="seda:c"/>
    <to uri="seda:d"/>
  </route>
</camelContext>
```

Recipient list calculated at run time

In most cases, when you use the recipient list pattern, you want the list of recipients to be calculated at runtime. For this, you can use the `recipientList()` processor, which takes a list of destinations as its sole argument. Because Mediation Router applies a type converter to the list argument, it should be possible to use most standard Java list types here (for example, a collection, a list or an array). For more details about type converters, see Built-In Type Converters in the *Programmer's Guide*.

Java DSL example

The following example shows how to extract the list of destinations from a message header called `recipientListHeader`, where the header value is a comma-separated list of endpoint URIs:

```
from("direct:a").recipientList(header("recipientListHead
er").tokenize(","));
```

In some cases, if the header value is a list type, you might be able to use it directly as the argument to `recipientListHeader()`. For example:

```
from("seda:a").recipientList(header("foo"));
```

However, this example is entirely dependent on the manner in which the underlying component parses this particular header. If the component parses the header as a simple string, this example would *not* work. You have to know how the underlying component parses its header data—see *Components* in the *Component Reference*.

XML configuration example

The following example shows how to configure the preceding route in XML, where it is assumed that the underlying component parses the `foo` header as a list type:

```
<camelContext id="buildDynamicRecipientList" xmlns="http://act
ivemq.apache.org/camel/schema/spring">
  <route>
    <from uri="seda:a"/>
    <recipientList>
      <header name="foo"/>
    </recipientList>
  </route>
</camelContext>
```