



# Red Hat Integration 2022.q3

## Kamelets Reference

Kamelets Reference



# Red Hat Integration 2022.q3 Kamelets Reference

---

Kamelets Reference

## Legal Notice

Copyright © 2022 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux<sup>®</sup> is the registered trademark of Linus Torvalds in the United States and other countries.

Java<sup>®</sup> is a registered trademark of Oracle and/or its affiliates.

XFS<sup>®</sup> is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL<sup>®</sup> is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js<sup>®</sup> is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack<sup>®</sup> Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

## Abstract

Kamelets are reusable route components that hide the complexity of creating data pipelines that connect to external systems.

## Table of Contents

<b>PREFACE</b> .....	<b>18</b>
MAKING OPEN SOURCE MORE INCLUSIVE	18
<b>CHAPTER 1. AWS DYNAMODB SINK</b> .....	<b>19</b>
1.1. CONFIGURATION OPTIONS	19
1.2. DEPENDENCIES	20
1.3. USAGE	20
1.3.1. Knative Sink	21
1.3.1.1. Prerequisite	21
1.3.1.2. Procedure for using the cluster CLI	21
1.3.1.3. Procedure for using the Kamel CLI	21
1.3.2. Kafka Sink	21
1.3.2.1. Prerequisites	22
1.3.2.2. Procedure for using the cluster CLI	22
1.3.2.3. Procedure for using the Kamel CLI	22
1.4. KAMELET SOURCE FILE	22
<b>CHAPTER 2. AVRO DESERIALIZE ACTION</b> .....	<b>23</b>
2.1. CONFIGURATION OPTIONS	23
2.2. DEPENDENCIES	23
2.3. USAGE	23
2.3.1. Knative Action	24
2.3.1.1. Prerequisite	24
2.3.1.2. Procedure for using the cluster CLI	24
2.3.1.3. Procedure for using the Kamel CLI	25
2.3.2. Kafka Action	25
2.3.2.1. Prerequisites	26
2.3.2.2. Procedure for using the cluster CLI	26
2.3.2.3. Procedure for using the Kamel CLI	26
2.4. KAMELET SOURCE FILE	26
<b>CHAPTER 3. AVRO SERIALIZE ACTION</b> .....	<b>27</b>
3.1. CONFIGURATION OPTIONS	27
3.2. DEPENDENCIES	27
3.3. USAGE	27
3.3.1. Knative Action	28
3.3.1.1. Prerequisite	28
3.3.1.2. Procedure for using the cluster CLI	28
3.3.1.3. Procedure for using the Kamel CLI	28
3.3.2. Kafka Action	29
3.3.2.1. Prerequisites	29
3.3.2.2. Procedure for using the cluster CLI	29
3.3.2.3. Procedure for using the Kamel CLI	30
3.4. KAMELET SOURCE FILE	30
<b>CHAPTER 4. AWS KINESIS SINK</b> .....	<b>31</b>
4.1. CONFIGURATION OPTIONS	31
4.2. DEPENDENCIES	31
4.3. USAGE	31
4.3.1. Knative Sink	32
4.3.1.1. Prerequisite	32
4.3.1.2. Procedure for using the cluster CLI	32

4.3.1.3. Procedure for using the Kamel CLI	32
4.3.2. Kafka Sink	32
4.3.2.1. Prerequisites	33
4.3.2.2. Procedure for using the cluster CLI	33
4.3.2.3. Procedure for using the Kamel CLI	33
4.4. KAMELET SOURCE FILE	33
<b>CHAPTER 5. AWS KINESIS SOURCE</b> .....	<b>34</b>
5.1. CONFIGURATION OPTIONS	34
5.2. DEPENDENCIES	34
5.3. USAGE	34
5.3.1. Knative Source	34
5.3.1.1. Prerequisite	35
5.3.1.2. Procedure for using the cluster CLI	35
5.3.1.3. Procedure for using the Kamel CLI	35
5.3.2. Kafka Source	35
5.3.2.1. Prerequisites	36
5.3.2.2. Procedure for using the cluster CLI	36
5.3.2.3. Procedure for using the Kamel CLI	36
5.4. KAMELET SOURCE FILE	36
<b>CHAPTER 6. AWS LAMBDA SINK</b> .....	<b>37</b>
6.1. CONFIGURATION OPTIONS	37
6.2. DEPENDENCIES	37
6.3. USAGE	37
6.3.1. Knative Sink	37
6.3.1.1. Prerequisite	38
6.3.1.2. Procedure for using the cluster CLI	38
6.3.1.3. Procedure for using the Kamel CLI	38
6.3.2. Kafka Sink	38
6.3.2.1. Prerequisites	39
6.3.2.2. Procedure for using the cluster CLI	39
6.3.2.3. Procedure for using the Kamel CLI	39
6.4. KAMELET SOURCE FILE	39
<b>CHAPTER 7. AWS REDSHIFT SINK</b> .....	<b>40</b>
7.1. CONFIGURATION OPTIONS	40
7.2. DEPENDENCIES	40
7.3. USAGE	41
7.3.1. Knative Sink	41
7.3.1.1. Prerequisite	41
7.3.1.2. Procedure for using the cluster CLI	41
7.3.1.3. Procedure for using the Kamel CLI	42
7.3.2. Kafka Sink	42
7.3.2.1. Prerequisites	42
7.3.2.2. Procedure for using the cluster CLI	42
7.3.2.3. Procedure for using the Kamel CLI	43
7.4. KAMELET SOURCE FILE	43
<b>CHAPTER 8. AWS SNS SINK</b> .....	<b>44</b>
8.1. CONFIGURATION OPTIONS	44
8.2. DEPENDENCIES	44
8.3. USAGE	44
8.3.1. Knative Sink	44

8.3.1.1. Prerequisite	45
8.3.1.2. Procedure for using the cluster CLI	45
8.3.1.3. Procedure for using the Kamel CLI	45
8.3.2. Kafka Sink	45
8.3.2.1. Prerequisites	46
8.3.2.2. Procedure for using the cluster CLI	46
8.3.2.3. Procedure for using the Kamel CLI	46
8.4. KAMELET SOURCE FILE	46
<b>CHAPTER 9. AWS SQS SINK</b> .....	<b>47</b>
9.1. CONFIGURATION OPTIONS	47
9.2. DEPENDENCIES	47
9.3. USAGE	47
9.3.1. Knative Sink	47
9.3.1.1. Prerequisite	48
9.3.1.2. Procedure for using the cluster CLI	48
9.3.1.3. Procedure for using the Kamel CLI	48
9.3.2. Kafka Sink	48
9.3.2.1. Prerequisites	49
9.3.2.2. Procedure for using the cluster CLI	49
9.3.2.3. Procedure for using the Kamel CLI	49
9.4. KAMELET SOURCE FILE	49
<b>CHAPTER 10. AWS SQS SOURCE</b> .....	<b>50</b>
10.1. CONFIGURATION OPTIONS	50
10.2. DEPENDENCIES	50
10.3. USAGE	50
10.3.1. Knative Source	51
10.3.1.1. Prerequisite	51
10.3.1.2. Procedure for using the cluster CLI	51
10.3.1.3. Procedure for using the Kamel CLI	51
10.3.2. Kafka Source	51
10.3.2.1. Prerequisites	52
10.3.2.2. Procedure for using the cluster CLI	52
10.3.2.3. Procedure for using the Kamel CLI	52
10.4. KAMELET SOURCE FILE	52
<b>CHAPTER 11. AWS 2 SIMPLE QUEUE SERVICE FIFO SINK</b> .....	<b>54</b>
11.1. CONFIGURATION OPTIONS	54
11.2. DEPENDENCIES	54
11.3. USAGE	54
11.3.1. Knative Sink	55
11.3.1.1. Prerequisite	55
11.3.1.2. Procedure for using the cluster CLI	55
11.3.1.3. Procedure for using the Kamel CLI	55
11.3.2. Kafka Sink	55
11.3.2.1. Prerequisites	56
11.3.2.2. Procedure for using the cluster CLI	56
11.3.2.3. Procedure for using the Kamel CLI	56
11.4. KAMELET SOURCE FILE	56
<b>CHAPTER 12. AWS S3 SINK</b> .....	<b>58</b>
12.1. CONFIGURATION OPTIONS	58
12.2. DEPENDENCIES	58

12.3. USAGE	58
12.3.1. Knative Sink	58
12.3.1.1. Prerequisite	59
12.3.1.2. Procedure for using the cluster CLI	59
12.3.1.3. Procedure for using the Kamel CLI	59
12.3.2. Kafka Sink	59
12.3.2.1. Prerequisites	60
12.3.2.2. Procedure for using the cluster CLI	60
12.3.2.3. Procedure for using the Kamel CLI	60
12.4. KAMELET SOURCE FILE	60
<b>CHAPTER 13. AWS S3 SOURCE</b> .....	<b>61</b>
13.1. CONFIGURATION OPTIONS	61
13.2. DEPENDENCIES	61
13.3. USAGE	61
13.3.1. Knative Source	61
13.3.1.1. Prerequisite	62
13.3.1.2. Procedure for using the cluster CLI	62
13.3.1.3. Procedure for using the Kamel CLI	62
13.3.2. Kafka Source	62
13.3.2.1. Prerequisites	63
13.3.2.2. Procedure for using the cluster CLI	63
13.3.2.3. Procedure for using the Kamel CLI	63
13.4. KAMELET SOURCE FILE	63
<b>CHAPTER 14. AWS S3 STREAMING UPLOAD SINK</b> .....	<b>64</b>
14.1. CONFIGURATION OPTIONS	64
14.2. DEPENDENCIES	65
14.3. USAGE	65
14.3.1. Knative Sink	65
14.3.1.1. Prerequisite	66
14.3.1.2. Procedure for using the cluster CLI	66
14.3.1.3. Procedure for using the Kamel CLI	66
14.3.2. Kafka Sink	66
14.3.2.1. Prerequisites	67
14.3.2.2. Procedure for using the cluster CLI	67
14.3.2.3. Procedure for using the Kamel CLI	67
14.4. KAMELET SOURCE FILE	67
<b>CHAPTER 15. CASSANDRA SINK</b> .....	<b>68</b>
15.1. CONFIGURATION OPTIONS	68
15.2. DEPENDENCIES	69
15.3. USAGE	69
15.3.1. Knative Sink	69
15.3.1.1. Prerequisite	69
15.3.1.2. Procedure for using the cluster CLI	69
15.3.1.3. Procedure for using the Kamel CLI	70
15.3.2. Kafka Sink	70
15.3.2.1. Prerequisites	70
15.3.2.2. Procedure for using the cluster CLI	70
15.3.2.3. Procedure for using the Kamel CLI	71
15.4. KAMELET SOURCE FILE	71
<b>CHAPTER 16. CASSANDRA SOURCE</b> .....	<b>72</b>



16.1. CONFIGURATION OPTIONS	72
16.2. DEPENDENCIES	73
16.3. USAGE	73
16.3.1. Knative Source	73
16.3.1.1. Prerequisite	74
16.3.1.2. Procedure for using the cluster CLI	74
16.3.1.3. Procedure for using the Kamel CLI	74
16.3.2. Kafka Source	74
16.3.2.1. Prerequisites	75
16.3.2.2. Procedure for using the cluster CLI	75
16.3.2.3. Procedure for using the Kamel CLI	75
16.4. KAMELET SOURCE FILE	75
<b>CHAPTER 17. EXTRACT FIELD ACTION</b> .....	<b>76</b>
17.1. CONFIGURATION OPTIONS	76
17.2. DEPENDENCIES	76
17.3. USAGE	76
17.3.1. Knative Action	76
17.3.1.1. Prerequisite	77
17.3.1.2. Procedure for using the cluster CLI	77
17.3.1.3. Procedure for using the Kamel CLI	77
17.3.2. Kafka Action	77
17.3.2.1. Prerequisites	78
17.3.2.2. Procedure for using the cluster CLI	78
17.3.2.3. Procedure for using the Kamel CLI	78
17.4. KAMELET SOURCE FILE	78
<b>CHAPTER 18. FTP SINK</b> .....	<b>79</b>
18.1. CONFIGURATION OPTIONS	79
18.2. DEPENDENCIES	79
18.3. USAGE	80
18.3.1. Knative Sink	80
18.3.1.1. Prerequisite	80
18.3.1.2. Procedure for using the cluster CLI	80
18.3.1.3. Procedure for using the Kamel CLI	80
18.3.2. Kafka Sink	81
18.3.2.1. Prerequisites	81
18.3.2.2. Procedure for using the cluster CLI	81
18.3.2.3. Procedure for using the Kamel CLI	81
18.4. KAMELET SOURCE FILE	82
<b>CHAPTER 19. FTP SOURCE</b> .....	<b>83</b>
19.1. CONFIGURATION OPTIONS	83
19.2. DEPENDENCIES	83
19.3. USAGE	84
19.3.1. Knative Source	84
19.3.1.1. Prerequisite	84
19.3.1.2. Procedure for using the cluster CLI	84
19.3.1.3. Procedure for using the Kamel CLI	84
19.3.2. Kafka Source	85
19.3.2.1. Prerequisites	85
19.3.2.2. Procedure for using the cluster CLI	85
19.3.2.3. Procedure for using the Kamel CLI	85
19.4. KAMELET SOURCE FILE	86

<b>CHAPTER 20. HAS HEADER FILTER ACTION</b> .....	<b>87</b>
20.1. CONFIGURATION OPTIONS	87
20.2. DEPENDENCIES	87
20.3. USAGE	87
20.3.1. Knative Action	87
20.3.1.1. Prerequisite	88
20.3.1.2. Procedure for using the cluster CLI	88
20.3.1.3. Procedure for using the Kamel CLI	88
20.3.2. Kafka Action	88
20.3.2.1. Prerequisites	89
20.3.2.2. Procedure for using the cluster CLI	89
20.3.2.3. Procedure for using the Kamel CLI	89
20.4. KAMELET SOURCE FILE	90
<b>CHAPTER 21. HOIST FIELD ACTION</b> .....	<b>91</b>
21.1. CONFIGURATION OPTIONS	91
21.2. DEPENDENCIES	91
21.3. USAGE	91
21.3.1. Knative Action	91
21.3.1.1. Prerequisite	92
21.3.1.2. Procedure for using the cluster CLI	92
21.3.1.3. Procedure for using the Kamel CLI	92
21.3.2. Kafka Action	92
21.3.2.1. Prerequisites	93
21.3.2.2. Procedure for using the cluster CLI	93
21.3.2.3. Procedure for using the Kamel CLI	93
21.4. KAMELET SOURCE FILE	93
<b>CHAPTER 22. HTTP SINK</b> .....	<b>94</b>
22.1. CONFIGURATION OPTIONS	94
22.2. DEPENDENCIES	94
22.3. USAGE	94
22.3.1. Knative Sink	94
22.3.1.1. Prerequisite	95
22.3.1.2. Procedure for using the cluster CLI	95
22.3.1.3. Procedure for using the Kamel CLI	95
22.3.2. Kafka Sink	95
22.3.2.1. Prerequisites	96
22.3.2.2. Procedure for using the cluster CLI	96
22.3.2.3. Procedure for using the Kamel CLI	96
22.4. KAMELET SOURCE FILE	96
<b>CHAPTER 23. INSERT FIELD ACTION</b> .....	<b>97</b>
23.1. CONFIGURATION OPTIONS	97
23.2. DEPENDENCIES	97
23.3. USAGE	97
23.3.1. Knative Action	97
23.3.1.1. Prerequisite	98
23.3.1.2. Procedure for using the cluster CLI	98
23.3.1.3. Procedure for using the Kamel CLI	98
23.3.2. Kafka Action	98
23.3.2.1. Prerequisites	99
23.3.2.2. Procedure for using the cluster CLI	99
23.3.2.3. Procedure for using the Kamel CLI	99

23.4. KAMELET SOURCE FILE	100
<b>CHAPTER 24. INSERT HEADER ACTION</b> .....	<b>101</b>
24.1. CONFIGURATION OPTIONS	101
24.2. DEPENDENCIES	101
24.3. USAGE	101
24.3.1. Knative Action	101
24.3.1.1. Prerequisite	102
24.3.1.2. Procedure for using the cluster CLI	102
24.3.1.3. Procedure for using the Kamel CLI	102
24.3.2. Kafka Action	102
24.3.2.1. Prerequisites	103
24.3.2.2. Procedure for using the cluster CLI	103
24.3.2.3. Procedure for using the Kamel CLI	103
24.4. KAMELET SOURCE FILE	103
<b>CHAPTER 25. IS TOMBSTONE FILTER ACTION</b> .....	<b>104</b>
25.1. CONFIGURATION OPTIONS	104
25.2. DEPENDENCIES	104
25.3. USAGE	104
25.3.1. Knative Action	104
25.3.1.1. Prerequisite	104
25.3.1.2. Procedure for using the cluster CLI	105
25.3.1.3. Procedure for using the Kamel CLI	105
25.3.2. Kafka Action	105
25.3.2.1. Prerequisites	105
25.3.2.2. Procedure for using the cluster CLI	106
25.3.2.3. Procedure for using the Kamel CLI	106
25.4. KAMELET SOURCE FILE	106
<b>CHAPTER 26. JIRA SOURCE</b> .....	<b>107</b>
26.1. CONFIGURATION OPTIONS	107
26.2. DEPENDENCIES	107
26.3. USAGE	107
26.3.1. Knative Source	107
26.3.1.1. Prerequisite	108
26.3.1.2. Procedure for using the cluster CLI	108
26.3.1.3. Procedure for using the Kamel CLI	108
26.3.2. Kafka Source	108
26.3.2.1. Prerequisites	109
26.3.2.2. Procedure for using the cluster CLI	109
26.3.2.3. Procedure for using the Kamel CLI	109
26.4. KAMELET SOURCE FILE	109
<b>CHAPTER 27. JMS - AMQP 1.0 KAMELET SINK</b> .....	<b>110</b>
27.1. CONFIGURATION OPTIONS	110
27.2. DEPENDENCIES	110
27.3. USAGE	110
27.3.1. Knative Sink	110
27.3.1.1. Prerequisite	111
27.3.1.2. Procedure for using the cluster CLI	111
27.3.1.3. Procedure for using the Kamel CLI	111
27.3.2. Kafka Sink	111
27.3.2.1. Prerequisites	112

---

27.3.2.2. Procedure for using the cluster CLI	112
27.3.2.3. Procedure for using the Kamel CLI	112
27.4. KAMELET SOURCE FILE	112
<b>CHAPTER 28. JMS - AMQP 1.0 KAMELET SOURCE</b> .....	<b>113</b>
28.1. CONFIGURATION OPTIONS	113
28.2. DEPENDENCIES	113
28.3. USAGE	113
28.3.1. Knative Source	113
28.3.1.1. Prerequisite	114
28.3.1.2. Procedure for using the cluster CLI	114
28.3.1.3. Procedure for using the Kamel CLI	114
28.3.2. Kafka Source	114
28.3.2.1. Prerequisites	115
28.3.2.2. Procedure for using the cluster CLI	115
28.3.2.3. Procedure for using the Kamel CLI	115
28.4. KAMELET SOURCE FILE	115
<b>CHAPTER 29. JMS - IBM MQ KAMELET SINK</b> .....	<b>116</b>
29.1. CONFIGURATION OPTIONS	116
29.2. DEPENDENCIES	116
29.3. USAGE	117
29.3.1. Knative Sink	117
29.3.1.1. Prerequisite	117
29.3.1.2. Procedure for using the cluster CLI	117
29.3.1.3. Procedure for using the Kamel CLI	118
29.3.2. Kafka Sink	118
29.3.2.1. Prerequisites	118
29.3.2.2. Procedure for using the cluster CLI	118
29.3.2.3. Procedure for using the Kamel CLI	119
29.4. KAMELET SOURCE FILE	119
<b>CHAPTER 30. JMS - IBM MQ KAMELET SOURCE</b> .....	<b>120</b>
30.1. CONFIGURATION OPTIONS	120
30.2. DEPENDENCIES	120
30.3. USAGE	121
30.3.1. Knative Source	121
30.3.1.1. Prerequisite	121
30.3.1.2. Procedure for using the cluster CLI	121
30.3.1.3. Procedure for using the Kamel CLI	122
30.3.2. Kafka Source	122
30.3.2.1. Prerequisites	122
30.3.2.2. Procedure for using the cluster CLI	122
30.3.2.3. Procedure for using the Kamel CLI	123
30.4. KAMELET SOURCE FILE	123
<b>CHAPTER 31. JSON DESERIALIZE ACTION</b> .....	<b>124</b>
31.1. CONFIGURATION OPTIONS	124
31.2. DEPENDENCIES	124
31.3. USAGE	124
31.3.1. Knative Action	124
31.3.1.1. Prerequisite	124
31.3.1.2. Procedure for using the cluster CLI	125
31.3.1.3. Procedure for using the Kamel CLI	125

31.3.2. Kafka Action	125
31.3.2.1. Prerequisites	125
31.3.2.2. Procedure for using the cluster CLI	126
31.3.2.3. Procedure for using the Kamel CLI	126
31.4. KAMELET SOURCE FILE	126
<b>CHAPTER 32. JSON SERIALIZE ACTION</b> .....	<b>127</b>
32.1. CONFIGURATION OPTIONS	127
32.2. DEPENDENCIES	127
32.3. USAGE	127
32.3.1. Knative Action	127
32.3.1.1. Prerequisite	127
32.3.1.2. Procedure for using the cluster CLI	128
32.3.1.3. Procedure for using the Kamel CLI	128
32.3.2. Kafka Action	128
32.3.2.1. Prerequisites	128
32.3.2.2. Procedure for using the cluster CLI	129
32.3.2.3. Procedure for using the Kamel CLI	129
32.4. KAMELET SOURCE FILE	129
<b>CHAPTER 33. KAFKA SINK</b> .....	<b>130</b>
33.1. CONFIGURATION OPTIONS	130
33.2. DEPENDENCIES	131
33.3. USAGE	131
33.3.1. Knative Sink	131
33.3.1.1. Prerequisite	131
33.3.1.2. Procedure for using the cluster CLI	131
33.3.1.3. Procedure for using the Kamel CLI	132
33.3.2. Kafka Sink	132
33.3.2.1. Prerequisites	132
33.3.2.2. Procedure for using the cluster CLI	132
33.3.2.3. Procedure for using the Kamel CLI	132
33.4. KAMELET SOURCE FILE	133
<b>CHAPTER 34. KAFKA SOURCE</b> .....	<b>134</b>
34.1. CONFIGURATION OPTIONS	134
34.2. DEPENDENCIES	135
34.3. USAGE	135
34.3.1. Knative Source	135
34.3.1.1. Prerequisite	136
34.3.1.2. Procedure for using the cluster CLI	136
34.3.1.3. Procedure for using the Kamel CLI	136
34.3.2. Kafka Source	136
34.3.2.1. Prerequisites	137
34.3.2.2. Procedure for using the cluster CLI	137
34.3.2.3. Procedure for using the Kamel CLI	137
34.4. KAMELET SOURCE FILE	137
<b>CHAPTER 35. KAFKA TOPIC NAME MATCHES FILTER ACTION</b> .....	<b>138</b>
35.1. CONFIGURATION OPTIONS	138
35.2. DEPENDENCIES	138
35.3. USAGE	138
35.3.1. Kafka Action	138
35.3.1.1. Prerequisites	139

---

35.3.1.2. Procedure for using the cluster CLI	139
35.3.1.3. Procedure for using the Kamel CLI	139
35.4. KAMELET SOURCE FILE	139
<b>CHAPTER 36. LOG SINK</b> .....	<b>140</b>
36.1. CONFIGURATION OPTIONS	140
36.2. DEPENDENCIES	140
36.3. USAGE	140
36.3.1. Knative Sink	140
36.3.1.1. Prerequisite	141
36.3.1.2. Procedure for using the cluster CLI	141
36.3.1.3. Procedure for using the Kamel CLI	141
36.3.2. Kafka Sink	141
36.3.2.1. Prerequisites	141
36.3.2.2. Procedure for using the cluster CLI	142
36.3.2.3. Procedure for using the Kamel CLI	142
36.4. KAMELET SOURCE FILE	142
<b>CHAPTER 37. MARIADB SINK</b> .....	<b>143</b>
37.1. CONFIGURATION OPTIONS	143
37.2. DEPENDENCIES	143
37.3. USAGE	144
37.3.1. Knative Sink	144
37.3.1.1. Prerequisite	144
37.3.1.2. Procedure for using the cluster CLI	144
37.3.1.3. Procedure for using the Kamel CLI	145
37.3.2. Kafka Sink	145
37.3.2.1. Prerequisites	145
37.3.2.2. Procedure for using the cluster CLI	145
37.3.2.3. Procedure for using the Kamel CLI	146
37.4. KAMELET SOURCE FILE	146
<b>CHAPTER 38. MASK FIELDS ACTION</b> .....	<b>147</b>
38.1. CONFIGURATION OPTIONS	147
38.2. DEPENDENCIES	147
38.3. USAGE	147
38.3.1. Knative Action	147
38.3.1.1. Prerequisite	148
38.3.1.2. Procedure for using the cluster CLI	148
38.3.1.3. Procedure for using the Kamel CLI	148
38.3.2. Kafka Action	148
38.3.2.1. Prerequisites	149
38.3.2.2. Procedure for using the cluster CLI	149
38.3.2.3. Procedure for using the Kamel CLI	149
38.4. KAMELET SOURCE FILE	149
<b>CHAPTER 39. MESSAGE TIMESTAMP ROUTER ACTION</b> .....	<b>150</b>
39.1. CONFIGURATION OPTIONS	150
39.2. DEPENDENCIES	151
39.3. USAGE	151
39.3.1. Knative Action	151
39.3.1.1. Prerequisite	151
39.3.1.2. Procedure for using the cluster CLI	152
39.3.1.3. Procedure for using the Kamel CLI	152

39.3.2. Kafka Action	152
39.3.2.1. Prerequisites	152
39.3.2.2. Procedure for using the cluster CLI	153
39.3.2.3. Procedure for using the Kamel CLI	153
39.4. KAMELET SOURCE FILE	153
<b>CHAPTER 40. MONGODB SINK</b> .....	<b>154</b>
40.1. CONFIGURATION OPTIONS	154
40.2. DEPENDENCIES	155
40.3. USAGE	155
40.3.1. Knative Sink	155
40.3.1.1. Prerequisite	156
40.3.1.2. Procedure for using the cluster CLI	156
40.3.1.3. Procedure for using the Kamel CLI	156
40.3.2. Kafka Sink	156
40.3.2.1. Prerequisites	157
40.3.2.2. Procedure for using the cluster CLI	157
40.3.2.3. Procedure for using the Kamel CLI	157
40.4. KAMELET SOURCE FILE	157
<b>CHAPTER 41. MONGODB SOURCE</b> .....	<b>158</b>
41.1. CONFIGURATION OPTIONS	158
41.2. DEPENDENCIES	159
41.3. USAGE	159
41.3.1. Knative Source	159
41.3.1.1. Prerequisite	160
41.3.1.2. Procedure for using the cluster CLI	160
41.3.1.3. Procedure for using the Kamel CLI	160
41.3.2. Kafka Source	160
41.3.2.1. Prerequisites	161
41.3.2.2. Procedure for using the cluster CLI	161
41.3.2.3. Procedure for using the Kamel CLI	161
41.4. KAMELET SOURCE FILE	161
<b>CHAPTER 42. MYSQL SINK</b> .....	<b>162</b>
42.1. CONFIGURATION OPTIONS	162
42.2. DEPENDENCIES	162
42.3. USAGE	163
42.3.1. Knative Sink	163
42.3.1.1. Prerequisite	163
42.3.1.2. Procedure for using the cluster CLI	163
42.3.1.3. Procedure for using the Kamel CLI	164
42.3.2. Kafka Sink	164
42.3.2.1. Prerequisites	164
42.3.2.2. Procedure for using the cluster CLI	164
42.3.2.3. Procedure for using the Kamel CLI	165
42.4. KAMELET SOURCE FILE	165
<b>CHAPTER 43. POSTGRESQL SINK</b> .....	<b>166</b>
43.1. CONFIGURATION OPTIONS	166
43.2. DEPENDENCIES	167
43.3. USAGE	167
43.3.1. Knative Sink	167
43.3.1.1. Prerequisite	167

43.3.1.2. Procedure for using the cluster CLI	167
43.3.1.3. Procedure for using the Kamel CLI	168
43.3.2. Kafka Sink	168
43.3.2.1. Prerequisites	168
43.3.2.2. Procedure for using the cluster CLI	168
43.3.2.3. Procedure for using the Kamel CLI	169
43.4. KAMELET SOURCE FILE	169
<b>CHAPTER 44. PREDICATE FILTER ACTION</b> .....	<b>170</b>
44.1. CONFIGURATION OPTIONS	170
44.2. DEPENDENCIES	170
44.3. USAGE	170
44.3.1. Knative Action	170
44.3.1.1. Prerequisite	171
44.3.1.2. Procedure for using the cluster CLI	171
44.3.1.3. Procedure for using the Kamel CLI	171
44.3.2. Kafka Action	171
44.3.2.1. Prerequisites	172
44.3.2.2. Procedure for using the cluster CLI	172
44.3.2.3. Procedure for using the Kamel CLI	172
44.4. KAMELET SOURCE FILE	172
<b>CHAPTER 45. PROTOBUF DESERIALIZE ACTION</b> .....	<b>173</b>
45.1. CONFIGURATION OPTIONS	173
45.2. DEPENDENCIES	173
45.3. USAGE	173
45.3.1. Knative Action	173
45.3.1.1. Prerequisite	174
45.3.1.2. Procedure for using the cluster CLI	174
45.3.1.3. Procedure for using the Kamel CLI	174
45.3.2. Kafka Action	175
45.3.2.1. Prerequisites	175
45.3.2.2. Procedure for using the cluster CLI	175
45.3.2.3. Procedure for using the Kamel CLI	176
45.4. KAMELET SOURCE FILE	176
<b>CHAPTER 46. PROTOBUF SERIALIZE ACTION</b> .....	<b>177</b>
46.1. CONFIGURATION OPTIONS	177
46.2. DEPENDENCIES	177
46.3. USAGE	177
46.3.1. Knative Action	177
46.3.1.1. Prerequisite	178
46.3.1.2. Procedure for using the cluster CLI	178
46.3.1.3. Procedure for using the Kamel CLI	178
46.3.2. Kafka Action	178
46.3.2.1. Prerequisites	179
46.3.2.2. Procedure for using the cluster CLI	179
46.3.2.3. Procedure for using the Kamel CLI	179
46.4. KAMELET SOURCE FILE	180
<b>CHAPTER 47. REGEX ROUTER ACTION</b> .....	<b>181</b>
47.1. CONFIGURATION OPTIONS	181
47.2. DEPENDENCIES	181
47.3. USAGE	181



47.3.1. Knative Action	181
47.3.1.1. Prerequisite	182
47.3.1.2. Procedure for using the cluster CLI	182
47.3.1.3. Procedure for using the Kamel CLI	182
47.3.2. Kafka Action	182
47.3.2.1. Prerequisites	183
47.3.2.2. Procedure for using the cluster CLI	183
47.3.2.3. Procedure for using the Kamel CLI	183
47.4. KAMELET SOURCE FILE	183
<b>CHAPTER 48. REPLACE FIELD ACTION</b> .....	<b>184</b>
48.1. CONFIGURATION OPTIONS	184
48.2. DEPENDENCIES	184
48.3. USAGE	185
48.3.1. Knative Action	185
48.3.1.1. Prerequisite	185
48.3.1.2. Procedure for using the cluster CLI	185
48.3.1.3. Procedure for using the Kamel CLI	185
48.3.2. Kafka Action	186
48.3.2.1. Prerequisites	186
48.3.2.2. Procedure for using the cluster CLI	186
48.3.2.3. Procedure for using the Kamel CLI	186
48.4. KAMELET SOURCE FILE	187
<b>CHAPTER 49. SALESFORCE SOURCE</b> .....	<b>188</b>
49.1. CONFIGURATION OPTIONS	188
49.2. DEPENDENCIES	188
49.3. USAGE	189
49.3.1. Knative Source	189
49.3.1.1. Prerequisite	189
49.3.1.2. Procedure for using the cluster CLI	189
49.3.1.3. Procedure for using the Kamel CLI	189
49.3.2. Kafka Source	190
49.3.2.1. Prerequisites	190
49.3.2.2. Procedure for using the cluster CLI	190
49.3.2.3. Procedure for using the Kamel CLI	190
49.4. KAMELET SOURCE FILE	191
<b>CHAPTER 50. SALESFORCE CREATE SINK</b> .....	<b>192</b>
50.1. CONFIGURATION OPTIONS	192
50.2. DEPENDENCIES	192
50.3. USAGE	192
50.3.1. Knative Sink	193
50.3.1.1. Prerequisite	193
50.3.1.2. Procedure for using the cluster CLI	193
50.3.1.3. Procedure for using the Kamel CLI	193
50.3.2. Kafka Sink	194
50.3.2.1. Prerequisites	194
50.3.2.2. Procedure for using the cluster CLI	194
50.3.2.3. Procedure for using the Kamel CLI	194
50.4. KAMELET SOURCE FILE	194
<b>CHAPTER 51. SALESFORCE DELETE SINK</b> .....	<b>196</b>
51.1. CONFIGURATION OPTIONS	196

51.2. DEPENDENCIES	196
51.3. USAGE	196
51.3.1. Knative Sink	197
51.3.1.1. Prerequisite	197
51.3.1.2. Procedure for using the cluster CLI	197
51.3.1.3. Procedure for using the Kamel CLI	197
51.3.2. Kafka Sink	198
51.3.2.1. Prerequisites	198
51.3.2.2. Procedure for using the cluster CLI	198
51.3.2.3. Procedure for using the Kamel CLI	198
51.4. KAMELET SOURCE FILE	198
<b>CHAPTER 52. SALESFORCE UPDATE SINK</b> .....	<b>200</b>
52.1. CONFIGURATION OPTIONS	200
52.2. DEPENDENCIES	200
52.3. USAGE	201
52.3.1. Knative Sink	201
52.3.1.1. Prerequisite	201
52.3.1.2. Procedure for using the cluster CLI	201
52.3.1.3. Procedure for using the Kamel CLI	201
52.3.2. Kafka Sink	202
52.3.2.1. Prerequisites	202
52.3.2.2. Procedure for using the cluster CLI	202
52.3.2.3. Procedure for using the Kamel CLI	202
52.4. KAMELET SOURCE FILE	203
<b>CHAPTER 53. SFTP SINK</b> .....	<b>204</b>
53.1. CONFIGURATION OPTIONS	204
53.2. DEPENDENCIES	204
53.3. USAGE	205
53.3.1. Knative Sink	205
53.3.1.1. Prerequisite	205
53.3.1.2. Procedure for using the cluster CLI	205
53.3.1.3. Procedure for using the Kamel CLI	205
53.3.2. Kafka Sink	206
53.3.2.1. Prerequisites	206
53.3.2.2. Procedure for using the cluster CLI	206
53.3.2.3. Procedure for using the Kamel CLI	206
53.4. KAMELET SOURCE FILE	207
<b>CHAPTER 54. SFTP SOURCE</b> .....	<b>208</b>
54.1. CONFIGURATION OPTIONS	208
54.2. DEPENDENCIES	208
54.3. USAGE	209
54.3.1. Knative Source	209
54.3.1.1. Prerequisite	209
54.3.1.2. Procedure for using the cluster CLI	209
54.3.1.3. Procedure for using the Kamel CLI	209
54.3.2. Kafka Source	210
54.3.2.1. Prerequisites	210
54.3.2.2. Procedure for using the cluster CLI	210
54.3.2.3. Procedure for using the Kamel CLI	210
54.4. KAMELET SOURCE FILE	211

<b>CHAPTER 55. SLACK SOURCE</b> .....	<b>212</b>
55.1. CONFIGURATION OPTIONS	212
55.2. DEPENDENCIES	212
55.3. USAGE	212
55.3.1. Knative Source	212
55.3.1.1. Prerequisite	213
55.3.1.2. Procedure for using the cluster CLI	213
55.3.1.3. Procedure for using the Kamel CLI	213
55.3.2. Kafka Source	213
55.3.2.1. Prerequisites	214
55.3.2.2. Procedure for using the cluster CLI	214
55.3.2.3. Procedure for using the Kamel CLI	214
55.4. KAMELET SOURCE FILE	214
<b>CHAPTER 56. MICROSOFT SQL SERVER SINK</b> .....	<b>215</b>
56.1. CONFIGURATION OPTIONS	215
56.2. DEPENDENCIES	215
56.3. USAGE	216
56.3.1. Knative Sink	216
56.3.1.1. Prerequisite	216
56.3.1.2. Procedure for using the cluster CLI	216
56.3.1.3. Procedure for using the Kamel CLI	217
56.3.2. Kafka Sink	217
56.3.2.1. Prerequisites	217
56.3.2.2. Procedure for using the cluster CLI	217
56.3.2.3. Procedure for using the Kamel CLI	218
56.4. KAMELET SOURCE FILE	218
<b>CHAPTER 57. TELEGRAM SOURCE</b> .....	<b>219</b>
57.1. CONFIGURATION OPTIONS	219
57.2. DEPENDENCIES	219
57.3. USAGE	219
57.3.1. Knative Source	219
57.3.1.1. Prerequisite	220
57.3.1.2. Procedure for using the cluster CLI	220
57.3.1.3. Procedure for using the Kamel CLI	220
57.3.2. Kafka Source	220
57.3.2.1. Prerequisites	221
57.3.2.2. Procedure for using the cluster CLI	221
57.3.2.3. Procedure for using the Kamel CLI	221
57.4. KAMELET SOURCE FILE	221
<b>CHAPTER 58. THROTTLE ACTION</b> .....	<b>222</b>
58.1. CONFIGURATION OPTIONS	222
58.2. DEPENDENCIES	222
58.3. USAGE	222
58.3.1. Knative Action	222
58.3.1.1. Prerequisite	223
58.3.1.2. Procedure for using the cluster CLI	223
58.3.1.3. Procedure for using the Kamel CLI	223
58.3.2. Kafka Action	223
58.3.2.1. Prerequisites	224
58.3.2.2. Procedure for using the cluster CLI	224
58.3.2.3. Procedure for using the Kamel CLI	224

---

58.4. KAMELET SOURCE FILE	224
<b>CHAPTER 59. TIMER SOURCE</b> .....	<b>225</b>
59.1. CONFIGURATION OPTIONS	225
59.2. DEPENDENCIES	225
59.3. USAGE	225
59.3.1. Knative Source	225
59.3.1.1. Prerequisite	226
59.3.1.2. Procedure for using the cluster CLI	226
59.3.1.3. Procedure for using the Kamel CLI	226
59.3.2. Kafka Source	226
59.3.2.1. Prerequisites	227
59.3.2.2. Procedure for using the cluster CLI	227
59.3.2.3. Procedure for using the Kamel CLI	227
59.4. KAMELET SOURCE FILE	227
<b>CHAPTER 60. TIMESTAMP ROUTER ACTION</b> .....	<b>228</b>
60.1. CONFIGURATION OPTIONS	228
60.2. DEPENDENCIES	228
60.3. USAGE	228
60.3.1. Knative Action	228
60.3.1.1. Prerequisite	229
60.3.1.2. Procedure for using the cluster CLI	229
60.3.1.3. Procedure for using the Kamel CLI	229
60.3.2. Kafka Action	229
60.3.2.1. Prerequisites	230
60.3.2.2. Procedure for using the cluster CLI	230
60.3.2.3. Procedure for using the Kamel CLI	230
60.4. KAMELET SOURCE FILE	230
<b>CHAPTER 61. VALUE TO KEY ACTION</b> .....	<b>231</b>
61.1. CONFIGURATION OPTIONS	231
61.2. DEPENDENCIES	231
61.3. USAGE	231
61.3.1. Knative Action	231
61.3.1.1. Prerequisite	232
61.3.1.2. Procedure for using the cluster CLI	232
61.3.1.3. Procedure for using the Kamel CLI	232
61.3.2. Kafka Action	232
61.3.2.1. Prerequisites	233
61.3.2.2. Procedure for using the cluster CLI	233
61.3.2.3. Procedure for using the Kamel CLI	233
61.4. KAMELET SOURCE FILE	233



## PREFACE

### MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see [our CTO Chris Wright's message](#).

## CHAPTER 1. AWS DYNAMODB SINK

Send data to AWS DynamoDB service. The sent data will insert/update/delete an item on the given AWS DynamoDB table.

Access Key/Secret Key are the basic method for authenticating to the AWS DynamoDB service. These parameters are optional, because the Kamelet also provides the following option 'useDefaultCredentialsProvider'.

When using a default Credentials Provider the AWS DynamoDB client will load the credentials through this provider and won't use the static credential. This is the reason for not having access key and secret key as mandatory parameters for this Kamelet.

This Kamelet expects a JSON field as body. The mapping between the JSON fields and table attribute values is done by key, so if you have the input as follows:

```
{"username":"oscerd", "city":"Rome"}
```

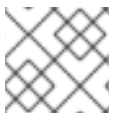
The Kamelet will insert/update an item in the given AWS DynamoDB table and set the attributes 'username' and 'city' respectively. Please note that the JSON object must include the primary key values that define the item.

### 1.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **aws-ddb-sink** Kamelet:

Property	Name	Description	Type	Default	Example
<b>region *</b>	AWS Region	The AWS region to connect to	string		<b>"eu-west-1"</b>
<b>table *</b>	Table	Name of the DynamoDB table to look at	string		
accessKey	Access Key	The access key obtained from AWS	string		
operation	Operation	The operation to perform (one of PutItem, UpdateItem, DeleteItem)	string	<b>"PutItem"</b>	<b>"PutItem"</b>
overrideEndpoint	Endpoint Overwrite	Set the need for overriding the endpoint URI. This option needs to be used in combination with uriEndpointOverride setting.	boolean	<b>false</b>	

Property	Name	Description	Type	Default	Example
secretKey	Secret Key	The secret key obtained from AWS	string		
uriEndpointOverride	Override Endpoint URI	Set the overriding endpoint URI. This option needs to be used in combination with <code>overrideEndpoint</code> option.	string		
useDefaultCredentialsProvider	Default Credentials Provider	Set whether the DynamoDB client should expect to load credentials through a default credentials provider or to expect static credentials to be passed in.	boolean	<b>false</b>	
writeCapacity	Write Capacity	The provisioned throughput to reserved for writing resources to your table	integer	<b>1</b>	

**NOTE**

Fields marked with an asterisk (\*) are mandatory.

## 1.2. DEPENDENCIES

At runtime, the **aws-ddb-sink** Kamelet relies upon the presence of the following dependencies:

- mvn:org.apache.camel.kamelets:camel-kamelets-utils:1.8.0
- camel:core
- camel:jackson
- camel:aws2-ddb
- camel:kamelet

## 1.3. USAGE

This section describes how you can use the **aws-ddb-sink**.



### 1.3.1. Knative Sink

You can use the **aws-ddb-sink** Kamelet as a Knative sink by binding it to a Knative object.

#### aws-ddb-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-ddb-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-ddb-sink
  properties:
    region: "eu-west-1"
    table: "The Table"
```

#### 1.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

#### 1.3.1.2. Procedure for using the cluster CLI

1. Save the **aws-ddb-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f aws-ddb-sink-binding.yaml
```

#### 1.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel aws-ddb-sink -p "sink.region=eu-west-1" -p "sink.table=The Table"
```

This command creates the KameletBinding in the current namespace on the cluster.

### 1.3.2. Kafka Sink

You can use the **aws-ddb-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

#### aws-ddb-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-ddb-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-ddb-sink
  properties:
    region: "eu-west-1"
    table: "The Table"
```

### 1.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 1.3.2.2. Procedure for using the cluster CLI

1. Save the **aws-ddb-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f aws-ddb-sink-binding.yaml
```

### 1.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic aws-ddb-sink -p "sink.region=eu-west-1" -p "sink.table=The Table"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 1.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//aws-ddb-sink.kamelet.yaml>

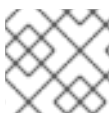
## CHAPTER 2. AVRO DESERIALIZE ACTION

Deserialize payload to Avro

### 2.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **avro-deserialize-action** Kamelet:

Property	Name	Description	Type	Default	Example
<b>schema *</b>	Schema	The Avro schema to use during serialization (as single-line, using JSON format)	string		<pre>{\"type\":   \\\"record\\\",   \\\"namespace\\\":   \\\"com.example\\\",   \\\"name\\\":   \\\"FullName\\\",   \\\"fields\\\":   [{\\\"name\\\":     \\\"first\\\", \\\"type\\\":     \\\"string\\\"},     {\\\"name\\\":     \\\"last\\\", \\\"type\\\":     \\\"string\\\"}]}</pre>
validate	Validate	Indicates if the content must be validated against the schema	boolean	<b>true</b>	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 2.2. DEPENDENCIES

At runtime, the **avro-deserialize-action** Kamelet relies upon the presence of the following dependencies:

- github:openshift-integration.kamelet-catalog:camel-kamelets-utils:kamelet-catalog-1.6-SNAPSHOT
- camel:kamelet
- camel:core
- camel:jackson-avro

### 2.3. USAGE

This section describes how you can use the **avro-deserialize-action**.

### 2.3.1. Knative Action

You can use the **avro-deserialize-action** Kamelet as an intermediate step in a Knative binding.

#### avro-deserialize-action-binding.yaml

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: avro-deserialize-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: '{"first":"Ada","last":"Lovelace"}'
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: json-deserialize-action
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: avro-serialize-action
    properties:
      schema: '{"type": "record", "namespace": "com.example", "name": "FullName", "fields": [{"name": "first", "type": "string"}, {"name": "last", "type": "string"}]}'
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: avro-deserialize-action
    properties:
      schema: '{"type": "record", "namespace": "com.example", "name": "FullName", "fields": [{"name": "first", "type": "string"}, {"name": "last", "type": "string"}]}'
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: json-serialize-action
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel

```

#### 2.3.1.1. Prerequisite

Make sure you have **Red Hat Integration - Camel K** installed into the OpenShift cluster you're connected to.

#### 2.3.1.2. Procedure for using the cluster CLI

1. Save the **avro-deserialize-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f avro-deserialize-action-binding.yaml
```

### 2.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind --name avro-deserialize-action-binding timer-source?
message={'first':"Ada","last":"Lovelace"} --step json-deserialize-action --step avro-serialize-action -p
step-1.schema={'type': "record", "namespace": "com.example", "name": "FullName", "fields":
[{"name": "first", "type": "string"}, {"name": "last", "type": "string"}]} --step avro-deserialize-action -p
step-2.schema={'type': "record", "namespace": "com.example", "name": "FullName", "fields":
[{"name": "first", "type": "string"}, {"name": "last", "type": "string"}]} --step json-serialize-action
channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

### 2.3.2. Kafka Action

You can use the **avro-deserialize-action** Kamelet as an intermediate step in a Kafka binding.

#### avro-deserialize-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: avro-deserialize-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
  properties:
    message: {'first':"Ada","last":"Lovelace"}
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: json-deserialize-action
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: avro-serialize-action
  properties:
    schema: "{\"type\": \"record\", \"namespace\": \"com.example\", \"name\": \"FullName\", \"fields\":
[\"name\": \"first\", \"type\": \"string\"], {\"name\": \"last\", \"type\": \"string\"}]}"
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: avro-deserialize-action
```

```

properties:
  schema: "{\"type\": \"record\", \"namespace\": \"com.example\", \"name\": \"FullName\", \"fields\":
[\"name\": \"first\", \"type\": \"string\"},{\"name\": \"last\", \"type\": \"string\"]}"
- ref:
  kind: Kamelet
  apiVersion: camel.apache.org/v1alpha1
  name: json-serialize-action
sink:
  ref:
  kind: KafkaTopic
  apiVersion: kafka.strimzi.io/v1beta1
  name: my-topic

```

### 2.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 2.3.2.2. Procedure for using the cluster CLI

1. Save the **avro-deserialize-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f avro-deserialize-action-binding.yaml
```

### 2.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```

kamel bind --name avro-deserialize-action-binding timer-source?
message='{\"first\": \"Ada\", \"last\": \"Lovelace\"}' --step json-deserialize-action --step avro-serialize-action -p
step-1.schema='{\"type\": \"record\", \"namespace\": \"com.example\", \"name\": \"FullName\", \"fields\":
[\"name\": \"first\", \"type\": \"string\"},{\"name\": \"last\", \"type\": \"string\"]}' --step avro-deserialize-action -p
step-2.schema='{\"type\": \"record\", \"namespace\": \"com.example\", \"name\": \"FullName\", \"fields\":
[\"name\": \"first\", \"type\": \"string\"},{\"name\": \"last\", \"type\": \"string\"]}' --step json-serialize-action
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic

```

This command creates the KameletBinding in the current namespace on the cluster.

## 2.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//avro-deserialize-action.kamelet.yaml>

## CHAPTER 3. AVRO SERIALIZE ACTION

Serialize payload to Avro

### 3.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **avro-serialize-action** Kamelet:

Property	Name	Description	Type	Default	Example
<b>schema *</b>	Schema	The Avro schema to use during serialization (as single-line, using JSON format)	string		<pre> {"type":   \record",   \namespace":   \com.example",   \name":   \FullName",   \fields":   [{\name":   \first", \type":   \string"},   {\name":   \last", \type":   \string}]} </pre>
validate	Validate	Indicates if the content must be validated against the schema	boolean	<b>true</b>	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 3.2. DEPENDENCIES

At runtime, the **avro-serialize-action** Kamelet relies upon the presence of the following dependencies:

- github:openshift-integration.kamelet-catalog:camel-kamelets-utils:kamelet-catalog-1.6-SNAPSHOT
- camel:kamelet
- camel:core
- camel:jackson-avro

### 3.3. USAGE

This section describes how you can use the **avro-serialize-action**.

### 3.3.1. Knative Action

You can use the **avro-serialize-action** Kamelet as an intermediate step in a Knative binding.

#### avro-serialize-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: avro-serialize-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: '{"first":"Ada","last":"Lovelace"}'
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: json-deserialize-action
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: avro-serialize-action
    properties:
      schema: '{"type": "record", "namespace": "com.example", "name": "FullName", "fields": [{"name": "first", "type": "string"}, {"name": "last", "type": "string"}]}'
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

#### 3.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

#### 3.3.1.2. Procedure for using the cluster CLI

1. Save the **avro-serialize-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f avro-serialize-action-binding.yaml
```

#### 3.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:



```
kamel bind --name avro-serialize-action-binding timer-source?
message='{ "first": "Ada", "last": "Lovelace" }' --step json-deserialize-action --step avro-serialize-action -p
step-1.schema='{ "type": "record", "namespace": "com.example", "name": "FullName", "fields":
[{"name": "first", "type": "string"}, {"name": "last", "type": "string"}]}' channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

### 3.3.2. Kafka Action

You can use the **avro-serialize-action** Kamelet as an intermediate step in a Kafka binding.

#### avro-serialize-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: avro-serialize-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: '{ "first": "Ada", "last": "Lovelace" }'
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: json-deserialize-action
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: avro-serialize-action
    properties:
      schema: '{ "type": "record", "namespace": "com.example", "name": "FullName", "fields":
[{"name": "first", "type": "string"}, {"name": "last", "type": "string"}]}'
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

#### 3.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

#### 3.3.2.2. Procedure for using the cluster CLI

1. Save the **avro-serialize-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the action by using the following command:

```
oc apply -f avro-serialize-action-binding.yaml
```

### 3.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind --name avro-serialize-action-binding timer-source?  
message='{ "first": "Ada", "last": "Lovelace" }' --step json-deserialize-action --step avro-serialize-action -p  
step-1.schema='{ "type": "record", "namespace": "com.example", "name": "FullName", "fields":  
[{"name": "first", "type": "string"}, {"name": "last", "type": "string"}]}'  
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 3.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//avro-serialize-action.kamelet.yaml>

## CHAPTER 4. AWS KINESIS SINK

Send data to AWS Kinesis.

The Kamelet expects the following header:

- **partition / ce-partition**: to set the Kinesis partition key

If the header won't be set the exchange ID will be used.

The Kamelet is also able to recognize the following header:

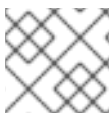
- **sequence-number / ce-sequencenumber**: to set the Sequence number

This header is optional.

### 4.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **aws-kinesis-sink** Kamelet:

Property	Name	Description	Type	Default	Example
<b>accessKey *</b>	Access Key	The access key obtained from AWS	string		
<b>region *</b>	AWS Region	The AWS region to connect to	string		<b>"eu-west-1"</b>
<b>secretKey *</b>	Secret Key	The secret key obtained from AWS	string		
<b>stream *</b>	Stream Name	The Kinesis stream that you want to access (needs to be created in advance)	string		



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 4.2. DEPENDENCIES

At runtime, the **aws-kinesis-sink** Kamelet relies upon the presence of the following dependencies:

- camel:aws2-kinesis
- camel:kamelet

### 4.3. USAGE

This section describes how you can use the **aws-kinesis-sink**.

### 4.3.1. Knative Sink

You can use the **aws-kinesis-sink** Kamelet as a Knative sink by binding it to a Knative object.

#### aws-kinesis-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-kinesis-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-kinesis-sink
  properties:
    accessKey: "The Access Key"
    region: "eu-west-1"
    secretKey: "The Secret Key"
    stream: "The Stream Name"
```

#### 4.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

#### 4.3.1.2. Procedure for using the cluster CLI

1. Save the **aws-kinesis-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f aws-kinesis-sink-binding.yaml
```

#### 4.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel aws-kinesis-sink -p "sink.accessKey=The Access Key" -p "sink.region=eu-west-1" -p "sink.secretKey=The Secret Key" -p "sink.stream=The Stream Name"
```

This command creates the KameletBinding in the current namespace on the cluster.

### 4.3.2. Kafka Sink

You can use the **aws-kinesis-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

### aws-kinesis-sink-binding.yaml

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-kinesis-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-kinesis-sink
  properties:
    accessKey: "The Access Key"
    region: "eu-west-1"
    secretKey: "The Secret Key"
    stream: "The Stream Name"

```

#### 4.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

#### 4.3.2.2. Procedure for using the cluster CLI

1. Save the **aws-kinesis-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f aws-kinesis-sink-binding.yaml
```

#### 4.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic aws-kinesis-sink -p "sink.accessKey=The Access Key" -p "sink.region=eu-west-1" -p "sink.secretKey=The Secret Key" -p "sink.stream=The Stream Name"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 4.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//aws-kinesis-sink.kamelet.yaml>

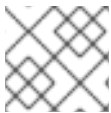
## CHAPTER 5. AWS KINESIS SOURCE

Receive data from AWS Kinesis.

### 5.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **aws-kinesis-source** Kamelet:

Property	Name	Description	Type	Default	Example
<b>accessKey *</b>	Access Key	The access key obtained from AWS	string		
<b>region *</b>	AWS Region	The AWS region to connect to	string		<b>"eu-west-1"</b>
<b>secretKey *</b>	Secret Key	The secret key obtained from AWS	string		
<b>stream *</b>	Stream Name	The Kinesis stream that you want to access (needs to be created in advance)	string		



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 5.2. DEPENDENCIES

At runtime, the **aws-kinesis-source** Kamelet relies upon the presence of the following dependencies:

- camel:gson
- camel:kamelet
- camel:aws2-kinesis

### 5.3. USAGE

This section describes how you can use the **aws-kinesis-source**.

#### 5.3.1. Knative Source

You can use the **aws-kinesis-source** Kamelet as a Knative source by binding it to a Knative object.

##### aws-kinesis-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
```

```

kind: KameletBinding
metadata:
  name: aws-kinesis-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-kinesis-source
  properties:
    accessKey: "The Access Key"
    region: "eu-west-1"
    secretKey: "The Secret Key"
    stream: "The Stream Name"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel

```

### 5.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

### 5.3.1.2. Procedure for using the cluster CLI

1. Save the **aws-kinesis-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f aws-kinesis-source-binding.yaml
```

### 5.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind aws-kinesis-source -p "source.accessKey=The Access Key" -p "source.region=eu-west-1"
-p "source.secretKey=The Secret Key" -p "source.stream=The Stream Name" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

## 5.3.2. Kafka Source

You can use the **aws-kinesis-source** Kamelet as a Kafka source by binding it to a Kafka topic.

### aws-kinesis-source-binding.yaml

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-kinesis-source-binding

```

```
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-kinesis-source
    properties:
      accessKey: "The Access Key"
      region: "eu-west-1"
      secretKey: "The Secret Key"
      stream: "The Stream Name"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

### 5.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 5.3.2.2. Procedure for using the cluster CLI

1. Save the **aws-kinesis-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f aws-kinesis-source-binding.yaml
```

### 5.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind aws-kinesis-source -p "source.accessKey=The Access Key" -p "source.region=eu-west-1"
-p "source.secretKey=The Secret Key" -p "source.stream=The Stream Name"
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 5.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//aws-kinesis-source.kamelet.yaml>



## CHAPTER 6. AWS LAMBDA SINK

Send a payload to an AWS Lambda function

### 6.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **aws-lambda-sink** Kamelet:

Property	Name	Description	Type	Default	Example
<b>accessKey *</b>	Access Key	The access key obtained from AWS	string		
<b>function *</b>	Function Name	The Lambda Function name	string		
<b>region *</b>	AWS Region	The AWS region to connect to	string		<b>"eu-west-1"</b>
<b>secretKey *</b>	Secret Key	The secret key obtained from AWS	string		



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 6.2. DEPENDENCIES

At runtime, the **aws-lambda-sink** Kamelet relies upon the presence of the following dependencies:

- camel:kamelet
- camel:aws2-lambda

### 6.3. USAGE

This section describes how you can use the **aws-lambda-sink**.

#### 6.3.1. Knative Sink

You can use the **aws-lambda-sink** Kamelet as a Knative sink by binding it to a Knative object.

##### aws-lambda-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-lambda-sink-binding
spec:
  source:
```

```

ref:
  kind: Channel
  apiVersion: messaging.knative.dev/v1
  name: mychannel
sink:
  ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: aws-lambda-sink
  properties:
    accessKey: "The Access Key"
    function: "The Function Name"
    region: "eu-west-1"
    secretKey: "The Secret Key"

```

### 6.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 6.3.1.2. Procedure for using the cluster CLI

1. Save the **aws-lambda-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f aws-lambda-sink-binding.yaml
```

### 6.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel aws-lambda-sink -p "sink.accessKey=The Access Key" -p "sink.function=The Function Name" -p "sink.region=eu-west-1" -p "sink.secretKey=The Secret Key"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 6.3.2. Kafka Sink

You can use the **aws-lambda-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

### aws-lambda-sink-binding.yaml

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-lambda-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1

```

```

  name: my-topic
sink:
  ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: aws-lambda-sink
  properties:
    accessKey: "The Access Key"
    function: "The Function Name"
    region: "eu-west-1"
    secretKey: "The Secret Key"

```

### 6.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 6.3.2.2. Procedure for using the cluster CLI

1. Save the **aws-lambda-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f aws-lambda-sink-binding.yaml
```

### 6.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic aws-lambda-sink -p "sink.accessKey=The Access Key" -p "sink.function=The Function Name" -p "sink.region=eu-west-1" -p "sink.secretKey=The Secret Key"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 6.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//aws-lambda-sink.kamelet.yaml>

## CHAPTER 7. AWS REDSHIFT SINK

Send data to an AWS Redshift Database.

This Kamelet expects a JSON as body. The mapping between the JSON fields and parameters is done by key, so if you have the following query:

```
'INSERT INTO accounts (username,city) VALUES (:#username,:#city)'
```

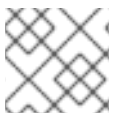
The Kamelet needs to receive as input something like:

```
{ "username": "oscerd", "city": "Rome" }
```

### 7.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **aws-redshift-sink** Kamelet:

Property	Name	Description	Type	Default	Example
<b>databaseName *</b>	Database Name	The Database Name we are pointing	string		
<b>password *</b>	Password	The password to use for accessing a secured AWS Redshift Database	string		
<b>query *</b>	Query	The Query to execute against the AWS Redshift Database	string		<b>"INSERT INTO accounts (username,city) VALUES (:#username,:#city)"</b>
<b>serverName *</b>	Server Name	Server Name for the data source	string		<b>"localhost"</b>
<b>username *</b>	Username	The username to use for accessing a secured AWS Redshift Database	string		
serverPort	Server Port	Server Port for the data source	string	<b>5439</b>	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 7.2. DEPENDENCIES

At runtime, the **aws-redshift-sink** Kamelet relies upon the presence of the following dependencies:

- camel:jackson
- camel:kamelet
- camel:sql
- mvn:com.amazon.redshift:redshift-jdbc42:2.1.0.5
- mvn:org.apache.commons:commons-dbcp2:2.7.0

## 7.3. USAGE

This section describes how you can use the **aws-redshift-sink**.

### 7.3.1. Knative Sink

You can use the **aws-redshift-sink** Kamelet as a Knative sink by binding it to a Knative object.

#### aws-redshift-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-redshift-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-redshift-sink
  properties:
    databaseName: "The Database Name"
    password: "The Password"
    query: "INSERT INTO accounts (username,city) VALUES (:#username,:#city)"
    serverName: "localhost"
    username: "The Username"
```

#### 7.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

#### 7.3.1.2. Procedure for using the cluster CLI

1. Save the **aws-redshift-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the sink by using the following command:

```
oc apply -f aws-redshift-sink-binding.yaml
```

### 7.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel aws-redshift-sink -p "sink.databaseName=The Database Name" -p "sink.password=The Password" -p "sink.query=INSERT INTO accounts (username,city) VALUES (:#username,:#city)" -p "sink.serverName=localhost" -p "sink.username=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

### 7.3.2. Kafka Sink

You can use the **aws-redshift-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

#### aws-redshift-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-redshift-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-redshift-sink
  properties:
    databaseName: "The Database Name"
    password: "The Password"
    query: "INSERT INTO accounts (username,city) VALUES (:#username,:#city)"
    serverName: "localhost"
    username: "The Username"
```

#### 7.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

#### 7.3.2.2. Procedure for using the cluster CLI

1. Save the **aws-redshift-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the sink by using the following command:

```
oc apply -f aws-redshift-sink-binding.yaml
```

### 7.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic aws-redshift-sink -p  
"sink.databaseName=The Database Name" -p "sink.password=The Password" -p  
"sink.query=INSERT INTO accounts (username,city) VALUES (:#username,:#city)" -p  
"sink.serverName=localhost" -p "sink.username=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 7.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//aws-redshift-sink.kamelet.yaml>

## CHAPTER 8. AWS SNS SINK

Send message to an AWS SNS Topic

### 8.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **aws-sns-sink** Kamelet:

Property	Name	Description	Type	Default	Example
<b>accessKey *</b>	Access Key	The access key obtained from AWS	string		
<b>region *</b>	AWS Region	The AWS region to connect to	string		<b>"eu-west-1"</b>
<b>secretKey *</b>	Secret Key	The secret key obtained from AWS	string		
<b>topicName OrArn *</b>	Topic Name	The SQS Topic name or ARN	string		
autoCreate Topic	Autocreate Topic	Setting the autocreation of the SNS topic.	boolean	<b>false</b>	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 8.2. DEPENDENCIES

At runtime, the **aws-sns-sink** Kamelet relies upon the presence of the following dependencies:

- camel:kamelet
- camel:aws2-sns

### 8.3. USAGE

This section describes how you can use the **aws-sns-sink**.

#### 8.3.1. Knative Sink

You can use the **aws-sns-sink** Kamelet as a Knative sink by binding it to a Knative object.

##### aws-sns-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
```



```

metadata:
  name: aws-sns-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-sns-sink
  properties:
    accessKey: "The Access Key"
    region: "eu-west-1"
    secretKey: "The Secret Key"
    topicNameOrArn: "The Topic Name"

```

### 8.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 8.3.1.2. Procedure for using the cluster CLI

1. Save the **aws-sns-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f aws-sns-sink-binding.yaml
```

### 8.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel aws-sns-sink -p "sink.accessKey=The Access Key" -p
"sink.region=eu-west-1" -p "sink.secretKey=The Secret Key" -p "sink.topicNameOrArn=The Topic
Name"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 8.3.2. Kafka Sink

You can use the **aws-sns-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

### aws-sns-sink-binding.yaml

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-sns-sink-binding

```

```
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-sns-sink
  properties:
    accessKey: "The Access Key"
    region: "eu-west-1"
    secretKey: "The Secret Key"
    topicNameOrArn: "The Topic Name"
```

### 8.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 8.3.2.2. Procedure for using the cluster CLI

1. Save the **aws-sns-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f aws-sns-sink-binding.yaml
```

### 8.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic aws-sns-sink -p "sink.accessKey=The Access Key" -p "sink.region=eu-west-1" -p "sink.secretKey=The Secret Key" -p "sink.topicNameOrArn=The Topic Name"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 8.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//aws-sns-sink.kamelet.yaml>

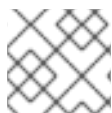
## CHAPTER 9. AWS SQS SINK

Send message to an AWS SQS Queue

### 9.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **aws-sqs-sink** Kamelet:

Property	Name	Description	Type	Default	Example
<b>accessKey *</b>	Access Key	The access key obtained from AWS	string		
<b>queueNameOrArn *</b>	Queue Name	The SQS Queue name or ARN	string		
<b>region *</b>	AWS Region	The AWS region to connect to	string		<b>"eu-west-1"</b>
<b>secretKey *</b>	Secret Key	The secret key obtained from AWS	string		
autoCreate Queue	Autocreate Queue	Setting the autocreation of the SQS queue.	boolean	<b>false</b>	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 9.2. DEPENDENCIES

At runtime, the **aws-sqs-sink** Kamelet relies upon the presence of the following dependencies:

- camel:aws2-sqs
- camel:core
- camel:kamelet

### 9.3. USAGE

This section describes how you can use the **aws-sqs-sink**.

#### 9.3.1. Knative Sink

You can use the **aws-sqs-sink** Kamelet as a Knative sink by binding it to a Knative object.

**aws-sqs-sink-binding.yaml**

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-sqs-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-sqs-sink
  properties:
    accessKey: "The Access Key"
    queueNameOrArn: "The Queue Name"
    region: "eu-west-1"
    secretKey: "The Secret Key"

```

### 9.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

### 9.3.1.2. Procedure for using the cluster CLI

1. Save the **aws-sqs-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f aws-sqs-sink-binding.yaml
```

### 9.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel aws-sqs-sink -p "sink.accessKey=The Access Key" -p "sink.queueNameOrArn=The Queue Name" -p "sink.region=eu-west-1" -p "sink.secretKey=The Secret Key"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 9.3.2. Kafka Sink

You can use the **aws-sqs-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

### aws-sqs-sink-binding.yaml

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding

```

```

metadata:
  name: aws-sqs-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-sqs-sink
  properties:
    accessKey: "The Access Key"
    queueNameOrArn: "The Queue Name"
    region: "eu-west-1"
    secretKey: "The Secret Key"

```

### 9.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 9.3.2.2. Procedure for using the cluster CLI

1. Save the **aws-sqs-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f aws-sqs-sink-binding.yaml
```

### 9.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic aws-sqs-sink -p "sink.accessKey=The Access Key" -p "sink.queueNameOrArn=The Queue Name" -p "sink.region=eu-west-1" -p "sink.secretKey=The Secret Key"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 9.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//aws-sqs-sink.kamelet.yaml>

## CHAPTER 10. AWS SQS SOURCE

Receive data from AWS SQS.

### 10.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **aws-sqs-source** Kamelet:

Property	Name	Description	Type	Default	Example
<b>accessKey *</b>	Access Key	The access key obtained from AWS	string		
<b>queueNameOrArn *</b>	Queue Name	The SQS Queue name or ARN	string		
<b>region *</b>	AWS Region	The AWS region to connect to	string		<b>"eu-west-1"</b>
<b>secretKey *</b>	Secret Key	The secret key obtained from AWS	string		
autoCreate Queue	Autocreate Queue	Setting the autocreation of the SQS queue.	boolean	<b>false</b>	
deleteAfter Read	Auto-delete Messages	Delete messages after consuming them	boolean	<b>true</b>	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 10.2. DEPENDENCIES

At runtime, the **aws-sqs-source** Kamelet relies upon the presence of the following dependencies:

- camel:aws2-sqs
- camel:core
- camel:kamelet
- camel:jackson

### 10.3. USAGE

This section describes how you can use the **aws-sqs-source**.

### 10.3.1. Knative Source

You can use the **aws-sqs-source** Kamelet as a Knative source by binding it to a Knative object.

#### aws-sqs-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-sqs-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-sqs-source
    properties:
      accessKey: "The Access Key"
      queueNameOrArn: "The Queue Name"
      region: "eu-west-1"
      secretKey: "The Secret Key"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

#### 10.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

#### 10.3.1.2. Procedure for using the cluster CLI

1. Save the **aws-sqs-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f aws-sqs-source-binding.yaml
```

#### 10.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind aws-sqs-source -p "source.accessKey=The Access Key" -p
"source.queueNameOrArn=The Queue Name" -p "source.region=eu-west-1" -p
"source.secretKey=The Secret Key" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

### 10.3.2. Kafka Source

You can use the **aws-sqs-source** Kamelet as a Kafka source by binding it to a Kafka topic.

### aws-sqs-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-sqs-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-sqs-source
    properties:
      accessKey: "The Access Key"
      queueNameOrArn: "The Queue Name"
      region: "eu-west-1"
      secretKey: "The Secret Key"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

#### 10.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

#### 10.3.2.2. Procedure for using the cluster CLI

1. Save the **aws-sqs-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f aws-sqs-source-binding.yaml
```

#### 10.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind aws-sqs-source -p "source.accessKey=The Access Key" -p
"source.queueNameOrArn=The Queue Name" -p "source.region=eu-west-1" -p
"source.secretKey=The Secret Key" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 10.4. KAMELET SOURCE FILE



<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//aws-sqs-source.kamelet.yaml>

## CHAPTER 11. AWS 2 SIMPLE QUEUE SERVICE FIFO SINK

Send message to an AWS SQS FIFO Queue

### 11.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **aws-sqs-fifo-sink** Kamelet:

Property	Name	Description	Type	Default	Example
<b>accessKey *</b>	Access Key	The access key obtained from AWS	string		
<b>queueNameOrArn *</b>	Queue Name	The SQS Queue name or ARN	string		
<b>region *</b>	AWS Region	The AWS region to connect to	string		<b>"eu-west-1"</b>
<b>secretKey *</b>	Secret Key	The secret key obtained from AWS	string		
autoCreate Queue	Autocreate Queue	Setting the autocreation of the SQS queue.	boolean	<b>false</b>	
contentBasedDeduplication	Content-Based Deduplication	Use content-based deduplication (should be enabled in the SQS FIFO queue first)	boolean	<b>false</b>	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 11.2. DEPENDENCIES

At runtime, the **aws-sqs-fifo-sink** Kamelet relies upon the presence of the following dependencies:

- camel:aws2-sqs
- camel:core
- camel:kamelet

### 11.3. USAGE

This section describes how you can use the **aws-sqs-fifo-sink**.

### 11.3.1. Knative Sink

You can use the **aws-sqs-fifo-sink** Kamelet as a Knative sink by binding it to a Knative object.

#### aws-sqs-fifo-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-sqs-fifo-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-sqs-fifo-sink
  properties:
    accessKey: "The Access Key"
    queueNameOrArn: "The Queue Name"
    region: "eu-west-1"
    secretKey: "The Secret Key"
```

#### 11.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

#### 11.3.1.2. Procedure for using the cluster CLI

1. Save the **aws-sqs-fifo-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f aws-sqs-fifo-sink-binding.yaml
```

#### 11.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel aws-sqs-fifo-sink -p "sink.accessKey=The Access Key" -p
"sink.queueNameOrArn=The Queue Name" -p "sink.region=eu-west-1" -p "sink.secretKey=The
Secret Key"
```

This command creates the KameletBinding in the current namespace on the cluster.

### 11.3.2. Kafka Sink

You can use the **aws-sqs-fifo-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

### aws-sqs-fifo-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-sqs-fifo-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-sqs-fifo-sink
  properties:
    accessKey: "The Access Key"
    queueNameOrArn: "The Queue Name"
    region: "eu-west-1"
    secretKey: "The Secret Key"
```

#### 11.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

#### 11.3.2.2. Procedure for using the cluster CLI

1. Save the **aws-sqs-fifo-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f aws-sqs-fifo-sink-binding.yaml
```

#### 11.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic aws-sqs-fifo-sink -p "sink.accessKey=The Access Key" -p "sink.queueNameOrArn=The Queue Name" -p "sink.region=eu-west-1" -p "sink.secretKey=The Secret Key"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 11.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//aws-sqs-fifo-sink.kamelet.yaml>

## CHAPTER 12. AWS S3 SINK

Upload data to AWS S3.

The Kamelet expects the following headers to be set:

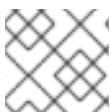
- **file** / **ce-file**: as the file name to upload

If the header won't be set the exchange ID will be used as file name.

### 12.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **aws-s3-sink** Kamelet:

Property	Name	Description	Type	Default	Example
<b>accessKey</b> *	Access Key	The access key obtained from AWS.	string		
<b>bucketNameOrArn</b> *	Bucket Name	The S3 Bucket name or ARN.	string		
<b>region</b> *	AWS Region	The AWS region to connect to.	string		<b>"eu-west-1"</b>
<b>secretKey</b> *	Secret Key	The secret key obtained from AWS.	string		
autoCreate Bucket	Autocreate Bucket	Setting the autocreation of the S3 bucket bucketName.	boolean	<b>false</b>	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 12.2. DEPENDENCIES

At runtime, the **aws-s3-sink** Kamelet relies upon the presence of the following dependencies:

- camel:aws2-s3
- camel:kamelet

### 12.3. USAGE

This section describes how you can use the **aws-s3-sink**.

#### 12.3.1. Knative Sink

You can use the **aws-s3-sink** Kamelet as a Knative sink by binding it to a Knative object.

### aws-s3-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-s3-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-s3-sink
  properties:
    accessKey: "The Access Key"
    bucketNameOrArn: "The Bucket Name"
    region: "eu-west-1"
    secretKey: "The Secret Key"
```

#### 12.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

#### 12.3.1.2. Procedure for using the cluster CLI

1. Save the **aws-s3-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f aws-s3-sink-binding.yaml
```

#### 12.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel aws-s3-sink -p "sink.accessKey=The Access Key" -p
"sink.bucketNameOrArn=The Bucket Name" -p "sink.region=eu-west-1" -p "sink.secretKey=The
Secret Key"
```

This command creates the KameletBinding in the current namespace on the cluster.

### 12.3.2. Kafka Sink

You can use the **aws-s3-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

## aws-s3-sink-binding.yaml

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-s3-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-s3-sink
  properties:
    accessKey: "The Access Key"
    bucketNameOrArn: "The Bucket Name"
    region: "eu-west-1"
    secretKey: "The Secret Key"

```

### 12.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 12.3.2.2. Procedure for using the cluster CLI

1. Save the **aws-s3-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f aws-s3-sink-binding.yaml
```

### 12.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic aws-s3-sink -p "sink.accessKey=The Access Key" -p "sink.bucketNameOrArn=The Bucket Name" -p "sink.region=eu-west-1" -p "sink.secretKey=The Secret Key"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 12.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//aws-s3-sink.kamelet.yaml>



## CHAPTER 13. AWS S3 SOURCE

Receive data from AWS S3.

### 13.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **aws-s3-source** Kamelet:

Property	Name	Description	Type	Default	Example
<b>accessKey *</b>	Access Key	The access key obtained from AWS	string		
<b>bucketNameOrArn *</b>	Bucket Name	The S3 Bucket name or ARN	string		
<b>region *</b>	AWS Region	The AWS region to connect to	string		<b>"eu-west-1"</b>
<b>secretKey *</b>	Secret Key	The secret key obtained from AWS	string		
autoCreate Bucket	Autocreate Bucket	Setting the autocreation of the S3 bucket bucketName.	boolean	<b>false</b>	
deleteAfter Read	Auto-delete Objects	Delete objects after consuming them	boolean	<b>true</b>	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 13.2. DEPENDENCIES

At runtime, the **aws-s3-source** Kamelet relies upon the presence of the following dependencies:

- camel:kamelet
- camel:aws2-s3

### 13.3. USAGE

This section describes how you can use the **aws-s3-source**.

#### 13.3.1. Knative Source

You can use the **aws-s3-source** Kamelet as a Knative source by binding it to a Knative object.

## aws-s3-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-s3-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-s3-source
    properties:
      accessKey: "The Access Key"
      bucketNameOrArn: "The Bucket Name"
      region: "eu-west-1"
      secretKey: "The Secret Key"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

### 13.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

### 13.3.1.2. Procedure for using the cluster CLI

1. Save the **aws-s3-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f aws-s3-source-binding.yaml
```

### 13.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind aws-s3-source -p "source.accessKey=The Access Key" -p
"source.bucketNameOrArn=The Bucket Name" -p "source.region=eu-west-1" -p
"source.secretKey=The Secret Key" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

## 13.3.2. Kafka Source

You can use the **aws-s3-source** Kamelet as a Kafka source by binding it to a Kafka topic.

## aws-s3-source-binding.yaml

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-s3-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-s3-source
    properties:
      accessKey: "The Access Key"
      bucketNameOrArn: "The Bucket Name"
      region: "eu-west-1"
      secretKey: "The Secret Key"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic

```

### 13.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 13.3.2.2. Procedure for using the cluster CLI

1. Save the **aws-s3-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f aws-s3-source-binding.yaml
```

### 13.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```

kamel bind aws-s3-source -p "source.accessKey=The Access Key" -p
"source.bucketNameOrArn=The Bucket Name" -p "source.region=eu-west-1" -p
"source.secretKey=The Secret Key" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic

```

This command creates the KameletBinding in the current namespace on the cluster.

## 13.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//aws-s3-source.kamelet.yaml>

## CHAPTER 14. AWS S3 STREAMING UPLOAD SINK

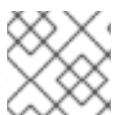
Upload data to AWS S3 in streaming upload mode.

### 14.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **aws-s3-streaming-upload-sink** Kamelet:

Property	Name	Description	Type	Default	Example
<b>accessKey *</b>	Access Key	The access key obtained from AWS.	string		
<b>bucketNameOrArn *</b>	Bucket Name	The S3 Bucket name or ARN.	string		
<b>keyName *</b>	Key Name	Setting the key name for an element in the bucket through endpoint parameter. In Streaming Upload, with the default configuration, this will be the base for the progressive creation of files.	string		
<b>region *</b>	AWS Region	The AWS region to connect to.	string		<b>"eu-west-1"</b>
<b>secretKey *</b>	Secret Key	The secret key obtained from AWS.	string		
autoCreate Bucket	Autocreate Bucket	Setting the autocreation of the S3 bucket bucketName.	boolean	<b>false</b>	
batchMessageNumber	Batch Message Number	The number of messages composing a batch in streaming upload mode	int	<b>10</b>	
batchSize	Batch Size	The batch size (in bytes) in streaming upload mode	int	<b>1000000</b>	

Property	Name	Description	Type	Default	Example
namingStrategy	Naming Strategy	The naming strategy to use in streaming upload mode. There are 2 enums and the value can be one of progressive, random	string	<b>"progressive"</b>	
restartingPolicy	Restarting Policy	The restarting policy to use in streaming upload mode. There are 2 enums and the value can be one of override, lastPart	string	<b>"lastPart"</b>	
streamingUploadMode	Streaming Upload Mode	Setting the Streaming Upload Mode	boolean	<b>true</b>	

**NOTE**

Fields marked with an asterisk (\*) are mandatory.

## 14.2. DEPENDENCIES

At runtime, the **aws-s3-streaming-upload-sink** Kamelet relies upon the presence of the following dependencies:

- camel:aws2-s3
- camel:kamelet

## 14.3. USAGE

This section describes how you can use the **aws-s3-streaming-upload-sink**.

### 14.3.1. Knative Sink

You can use the **aws-s3-streaming-upload-sink** Kamelet as a Knative sink by binding it to a Knative object.

#### aws-s3-streaming-upload-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-s3-streaming-upload-sink-binding
spec:
```

```

source:
  ref:
    kind: Channel
    apiVersion: messaging.knative.dev/v1
    name: mychannel
sink:
  ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: aws-s3-streaming-upload-sink
properties:
  accessKey: "The Access Key"
  bucketNameOrArn: "The Bucket Name"
  keyName: "The Key Name"
  region: "eu-west-1"
  secretKey: "The Secret Key"

```

### 14.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 14.3.1.2. Procedure for using the cluster CLI

1. Save the **aws-s3-streaming-upload-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f aws-s3-streaming-upload-sink-binding.yaml
```

### 14.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel aws-s3-streaming-upload-sink -p "sink.accessKey=The Access Key" -p "sink.bucketNameOrArn=The Bucket Name" -p "sink.keyName=The Key Name" -p "sink.region=eu-west-1" -p "sink.secretKey=The Secret Key"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 14.3.2. Kafka Sink

You can use the **aws-s3-streaming-upload-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

### aws-s3-streaming-upload-sink-binding.yaml

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-s3-streaming-upload-sink-binding
spec:
  source:

```

```

ref:
  kind: KafkaTopic
  apiVersion: kafka.strimzi.io/v1beta1
  name: my-topic
sink:
  ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: aws-s3-streaming-upload-sink
  properties:
    accessKey: "The Access Key"
    bucketNameOrArn: "The Bucket Name"
    keyName: "The Key Name"
    region: "eu-west-1"
    secretKey: "The Secret Key"

```

### 14.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 14.3.2.2. Procedure for using the cluster CLI

1. Save the **aws-s3-streaming-upload-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f aws-s3-streaming-upload-sink-binding.yaml
```

### 14.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic aws-s3-streaming-upload-sink -p
"sink.accessKey=The Access Key" -p "sink.bucketNameOrArn=The Bucket Name" -p
"sink.keyName=The Key Name" -p "sink.region=eu-west-1" -p "sink.secretKey=The Secret Key"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 14.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//aws-s3-streaming-upload-sink.kamelet.yaml>

## CHAPTER 15. CASSANDRA SINK

Send data to a Cassandra Cluster.

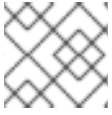
This Kamelet expects the body as JSON Array. The content of the JSON Array will be used as input for the CQL Prepared Statement set in the query parameter.

### 15.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **cassandra-sink** Kamelet:

Property	Name	Description	Type	Default	Example
<b>connection Host *</b>	Connection Host	Hostname(s) cassandra server(s). Multiple hosts can be separated by comma.	string		<b>"localhost"</b>
<b>connection Port *</b>	Connection Port	Port number of cassandra server(s)	string		<b>9042</b>
<b>keyspace *</b>	Keyspace	Keyspace to use	string		<b>"customers"</b>
<b>password *</b>	Password	The password to use for accessing a secured Cassandra Cluster	string		
<b>query *</b>	Query	The query to execute against the Cassandra cluster table	string		
<b>username *</b>	Username	The username to use for accessing a secured Cassandra Cluster	string		
consistency Level	Consistency Level	Consistency level to use. The value can be one of ANY, ONE, TWO, THREE, QUORUM, ALL, LOCAL_QUORUM, EACH_QUORUM, SERIAL, LOCAL_SERIAL, LOCAL_ONE	string	<b>"ANY"</b>	



**NOTE**

Fields marked with an asterisk (\*) are mandatory.

## 15.2. DEPENDENCIES

At runtime, the **cassandra-sink** Kamelet relies upon the presence of the following dependencies:

- camel:jackson
- camel:kamelet
- camel:cassandraql

## 15.3. USAGE

This section describes how you can use the **cassandra-sink**.

### 15.3.1. Knative Sink

You can use the **cassandra-sink** Kamelet as a Knative sink by binding it to a Knative object.

#### **cassandra-sink-binding.yaml**

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: cassandra-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: cassandra-sink
  properties:
    connectionHost: "localhost"
    connectionPort: 9042
    keyspace: "customers"
    password: "The Password"
    query: "The Query"
    username: "The Username"

```

#### 15.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

#### 15.3.1.2. Procedure for using the cluster CLI

1. Save the **cassandra-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f cassandra-sink-binding.yaml
```

### 15.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel cassandra-sink -p "sink.connectionHost=localhost" -p
sink.connectionPort=9042 -p "sink.keyspace=customers" -p "sink.password=The Password" -p
"sink.query=Query" -p "sink.username=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 15.3.2. Kafka Sink

You can use the **cassandra-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

### cassandra-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: cassandra-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: cassandra-sink
  properties:
    connectionHost: "localhost"
    connectionPort: 9042
    keyspace: "customers"
    password: "The Password"
    query: "The Query"
    username: "The Username"
```

#### 15.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

#### 15.3.2.2. Procedure for using the cluster CLI

1. Save the **cassandra-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f cassandra-sink-binding.yaml
```

### 15.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic cassandra-sink -p  
"sink.connectionHost=localhost" -p sink.connectionPort=9042 -p "sink.keyspace=customers" -p  
"sink.password=The Password" -p "sink.query=The Query" -p "sink.username=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 15.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//cassandra-sink.kamelet.yaml>

## CHAPTER 16. CASSANDRA SOURCE

Query a Cassandra cluster table.

### 16.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **cassandra-source** Kamelet:

Property	Name	Description	Type	Default	Example
<b>connection Host *</b>	Connection Host	Hostname(s) cassandra server(s). Multiple hosts can be separated by comma.	string		<b>"localhost"</b>
<b>connection Port *</b>	Connection Port	Port number of cassandra server(s)	string		<b>9042</b>
<b>keyspace *</b>	Keyspace	Keyspace to use	string		<b>"customers"</b>
<b>password *</b>	Password	The password to use for accessing a secured Cassandra Cluster	string		
<b>query *</b>	Query	The query to execute against the Cassandra cluster table	string		
<b>username *</b>	Username	The username to use for accessing a secured Cassandra Cluster	string		
consistency Level	Consistency Level	Consistency level to use. The value can be one of ANY, ONE, TWO, THREE, QUORUM, ALL, LOCAL_QUORUM, EACH_QUORUM, SERIAL, LOCAL_SERIAL, LOCAL_ONE	string	<b>"QUORUM"</b>	

Property	Name	Description	Type	Default	Example
resultStrategy	Result Strategy	The strategy to convert the result set of the query. Possible values are ALL, ONE, LIMIT_10, LIMIT_100...	string	"ALL"	



## NOTE

Fields marked with an asterisk (\*) are mandatory.

## 16.2. DEPENDENCIES

At runtime, the **cassandra-source** Kamelet relies upon the presence of the following dependencies:

- camel:jackson
- camel:kamelet
- camel:cassandraql

## 16.3. USAGE

This section describes how you can use the **cassandra-source**.

### 16.3.1. Knative Source

You can use the **cassandra-source** Kamelet as a Knative source by binding it to a Knative object.

#### **cassandra-source-binding.yaml**

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: cassandra-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: cassandra-source
    properties:
      connectionHost: "localhost"
      connectionPort: 9042
      keyspace: "customers"
      password: "The Password"
      query: "The Query"
      username: "The Username"

```

```

sink:
  ref:
    kind: Channel
    apiVersion: messaging.knative.dev/v1
    name: mychannel

```

### 16.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

### 16.3.1.2. Procedure for using the cluster CLI

1. Save the **cassandra-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f cassandra-source-binding.yaml
```

### 16.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind cassandra-source -p "source.connectionHost=localhost" -p source.connectionPort=9042 -p "source.keyspace=customers" -p "source.password=The Password" -p "source.query=The Query" -p "source.username=The Username" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

## 16.3.2. Kafka Source

You can use the **cassandra-source** Kamelet as a Kafka source by binding it to a Kafka topic.

### cassandra-source-binding.yaml

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: cassandra-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: cassandra-source
    properties:
      connectionHost: "localhost"
      connectionPort: 9042
      keyspace: "customers"
      password: "The Password"
      query: "The Query"
      username: "The Username"

```

```
sink:  
  ref:  
    kind: KafkaTopic  
    apiVersion: kafka.strimzi.io/v1beta1  
    name: my-topic
```

### 16.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 16.3.2.2. Procedure for using the cluster CLI

1. Save the **cassandra-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f cassandra-source-binding.yaml
```

### 16.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind cassandra-source -p "source.connectionHost=localhost" -p source.connectionPort=9042 -  
p "source.keyspace=customers" -p "source.password=The Password" -p "source.query=The Query" -  
p "source.username=The Username" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 16.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//cassandra-source.kamelet.yaml>

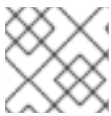
## CHAPTER 17. EXTRACT FIELD ACTION

Extract a field from the body

### 17.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **extract-field-action** Kamelet:

Property	Name	Description	Type	Default	Example
<b>field *</b>	Field	The name of the field to be added	string		



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 17.2. DEPENDENCIES

At runtime, the **extract-field-action** Kamelet relies upon the presence of the following dependencies:

- github:openshift-integration.kamelet-catalog:camel-kamelets-utils:kamelet-catalog-1.6-SNAPSHOT
- camel:kamelet
- camel:core
- camel:jackson

### 17.3. USAGE

This section describes how you can use the **extract-field-action**.

#### 17.3.1. Knative Action

You can use the **extract-field-action** Kamelet as an intermediate step in a Knative binding.

##### **extract-field-action-binding.yaml**

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: extract-field-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
  properties:
    message: "Hello"
```



```

steps:
- ref:
  kind: Kamelet
  apiVersion: camel.apache.org/v1alpha1
  name: extract-field-action
properties:
  field: "The Field"
sink:
  ref:
    kind: Channel
    apiVersion: messaging.knative.dev/v1
    name: mychannel

```

### 17.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

### 17.3.1.2. Procedure for using the cluster CLI

1. Save the **extract-field-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f extract-field-action-binding.yaml
```

### 17.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step extract-field-action -p "step-0.field=The Field"
channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

## 17.3.2. Kafka Action

You can use the **extract-field-action** Kamelet as an intermediate step in a Kafka binding.

### extract-field-action-binding.yaml

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: extract-field-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
  properties:

```

```
  message: "Hello"
steps:
- ref:
  kind: Kamelet
  apiVersion: camel.apache.org/v1alpha1
  name: extract-field-action
properties:
  field: "The Field"
sink:
  ref:
  kind: KafkaTopic
  apiVersion: kafka.strimzi.io/v1beta1
  name: my-topic
```

### 17.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 17.3.2.2. Procedure for using the cluster CLI

1. Save the **extract-field-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f extract-field-action-binding.yaml
```

### 17.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step extract-field-action -p "step-0.field=The Field"
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 17.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//extract-field-action.kamelet.yaml>

## CHAPTER 18. FTP SINK

Send data to an FTP Server.

The Kamelet expects the following headers to be set:

- **file / ce-file**: as the file name to upload

If the header won't be set the exchange ID will be used as file name.

### 18.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **ftp-sink** Kamelet:

Property	Name	Description	Type	Default	Example
<b>connection Host *</b>	Connection Host	Hostname of the FTP server	string		
<b>connection Port *</b>	Connection Port	Port of the FTP server	string	<b>21</b>	
<b>directoryName *</b>	Directory Name	The starting directory	string		
<b>password *</b>	Password	The password to access the FTP server	string		
<b>username *</b>	Username	The username to access the FTP server	string		
fileExist	File Existence	How to behave in case of file already existent. There are 4 enums and the value can be one of Override, Append, Fail or Ignore	string	<b>"Override"</b>	
passiveMode	Passive Mode	Sets passive mode connection	boolean	<b>false</b>	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 18.2. DEPENDENCIES

At runtime, the **ftp-sink** Kamelet relies upon the presence of the following dependencies:

- camel:ftp
- camel:core
- camel:kamelet

## 18.3. USAGE

This section describes how you can use the **ftp-sink**.

### 18.3.1. Knative Sink

You can use the **ftp-sink** Kamelet as a Knative sink by binding it to a Knative object.

#### ftp-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: ftp-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: ftp-sink
  properties:
    connectionHost: "The Connection Host"
    directoryName: "The Directory Name"
    password: "The Password"
    username: "The Username"
```

#### 18.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

#### 18.3.1.2. Procedure for using the cluster CLI

1. Save the **ftp-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f ftp-sink-binding.yaml
```

#### 18.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel ftp-sink -p "sink.connectionHost=The Connection Host" -p
"sink.directoryName=The Directory Name" -p "sink.password=The Password" -p "sink.username=The
Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

### 18.3.2. Kafka Sink

You can use the **ftp-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

#### ftp-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: ftp-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: ftp-sink
  properties:
    connectionHost: "The Connection Host"
    directoryName: "The Directory Name"
    password: "The Password"
    username: "The Username"
```

#### 18.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

#### 18.3.2.2. Procedure for using the cluster CLI

1. Save the **ftp-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f ftp-sink-binding.yaml
```

#### 18.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic ftp-sink -p "sink.connectionHost=The  
Connection Host" -p "sink.directoryName=The Directory Name" -p "sink.password=The Password" -p  
"sink.username=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 18.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//ftp-sink.kamelet.yaml>

## CHAPTER 19. FTP SOURCE

Receive data from an FTP Server.

### 19.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **ftp-source** Kamelet:

Property	Name	Description	Type	Default	Example
<b>connectionHost *</b>	Connection Host	Hostname of the FTP server	string		
<b>connectionPort *</b>	Connection Port	Port of the FTP server	string	<b>21</b>	
<b>directoryName *</b>	Directory Name	The starting directory	string		
<b>password *</b>	Password	The password to access the FTP server	string		
<b>username *</b>	Username	The username to access the FTP server	string		
idempotent	Idempotency	Skip already processed files.	boolean	<b>true</b>	
passiveMode	Passive Mode	Sets passive mode connection	boolean	<b>false</b>	
recursive	Recursive	If a directory, will look for files in all the sub-directories as well.	boolean	<b>false</b>	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 19.2. DEPENDENCIES

At runtime, the **ftp-source** Kamelet relies upon the presence of the following dependencies:

- camel:ftp
- camel:core

- camel:kamelet

## 19.3. USAGE

This section describes how you can use the **ftp-source**.

### 19.3.1. Knative Source

You can use the **ftp-source** Kamelet as a Knative source by binding it to a Knative object.

#### ftp-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: ftp-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: ftp-source
  properties:
    connectionHost: "The Connection Host"
    directoryName: "The Directory Name"
    password: "The Password"
    username: "The Username"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

#### 19.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

#### 19.3.1.2. Procedure for using the cluster CLI

1. Save the **ftp-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f ftp-source-binding.yaml
```

#### 19.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:



```
kamel bind ftp-source -p "source.connectionHost=The Connection Host" -p
"source.directoryName=The Directory Name" -p "source.password=The Password" -p
"source.username=The Username" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

### 19.3.2. Kafka Source

You can use the **ftp-source** Kamelet as a Kafka source by binding it to a Kafka topic.

#### ftp-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: ftp-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: ftp-source
    properties:
      connectionHost: "The Connection Host"
      directoryName: "The Directory Name"
      password: "The Password"
      username: "The Username"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

#### 19.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

#### 19.3.2.2. Procedure for using the cluster CLI

1. Save the **ftp-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f ftp-source-binding.yaml
```

#### 19.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind ftp-source -p "source.connectionHost=The Connection Host" -p  
"source.directoryName=The Directory Name" -p "source.password=The Password" -p  
"source.username=The Username" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 19.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//ftp-source.kamelet.yaml>

## CHAPTER 20. HAS HEADER FILTER ACTION

Filter based on the presence of one header

### 20.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **has-header-filter-action** Kamelet:

Property	Name	Description	Type	Default	Example
<b>name *</b>	Header Name	The header name to evaluate. The header name must be passed by the source Kamelet. For Knative only, if you are using Cloud Events, you must include the CloudEvent (ce-) prefix in the header name.	string		<b>"headerName"</b>



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 20.2. DEPENDENCIES

At runtime, the **has-header-filter-action** Kamelet relies upon the presence of the following dependencies:

- camel:core
- camel:kamelet

### 20.3. USAGE

This section describes how you can use the **has-header-filter-action**.

#### 20.3.1. Knative Action

You can use the **has-header-filter-action** Kamelet as an intermediate step in a Knative binding.

##### has-header-filter-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: has-header-filter-action-binding
```

```

spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: insert-header-action
    properties:
      name: "my-header"
      value: "my-value"
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: has-header-filter-action
    properties:
      name: "my-header"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel

```

### 20.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 20.3.1.2. Procedure for using the cluster CLI

1. Save the **has-header-filter-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f has-header-filter-action-binding.yaml
```

### 20.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind --name has-header-filter-action-binding timer-source?message="Hello" --step insert-
header-action -p "step-0.name=my-header" -p "step-0.value=my-value" --step has-header-filter-action
-p "step-1.name=my-header" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

## 20.3.2. Kafka Action

You can use the **has-header-filter-action** Kamelet as an intermediate step in a Kafka binding.

### has-header-filter-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: has-header-filter-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: insert-header-action
      properties:
        name: "my-header"
        value: "my-value"
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: has-header-filter-action
      properties:
        name: "my-header"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

#### 20.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

#### 20.3.2.2. Procedure for using the cluster CLI

1. Save the **has-header-filter-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f has-header-filter-action-binding.yaml
```

#### 20.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind --name has-header-filter-action-binding timer-source?message="Hello" --step insert-  
header-action -p "step-0.name=my-header" -p "step-0.value=my-value" --step has-header-filter-action  
-p "step-1.name=my-header" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 20.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//has-header-filter-action.kamelet.yaml>

## CHAPTER 21. HOIST FIELD ACTION

Wrap data in a single field

### 21.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **hoist-field-action** Kamelet:

Property	Name	Description	Type	Default	Example
<b>field *</b>	Field	The name of the field that will contain the event	string		



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 21.2. DEPENDENCIES

At runtime, the **hoist-field-action** Kamelet relies upon the presence of the following dependencies:

- github:openshift-integration.kamelet-catalog:camel-kamelets-utils:kamelet-catalog-1.6-SNAPSHOT
- camel:core
- camel:jackson
- camel:kamelet

### 21.3. USAGE

This section describes how you can use the **hoist-field-action**.

#### 21.3.1. Knative Action

You can use the **hoist-field-action** Kamelet as an intermediate step in a Knative binding.

##### hoist-field-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: hoist-field-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
  properties:
```

```

    message: "Hello"
  steps:
  - ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: hoist-field-action
  properties:
    field: "The Field"
  sink:
  ref:
    kind: Channel
    apiVersion: messaging.knative.dev/v1
    name: mychannel

```

### 21.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

### 21.3.1.2. Procedure for using the cluster CLI

1. Save the **hoist-field-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f hoist-field-action-binding.yaml
```

### 21.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step hoist-field-action -p "step-0.field=The Field"
channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

## 21.3.2. Kafka Action

You can use the **hoist-field-action** Kamelet as an intermediate step in a Kafka binding.

### hoist-field-action-binding.yaml

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: hoist-field-action-binding
spec:
  source:
  ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: timer-source

```



```

properties:
  message: "Hello"
steps:
- ref:
  kind: Kamelet
  apiVersion: camel.apache.org/v1alpha1
  name: hoist-field-action
  properties:
    field: "The Field"
sink:
  ref:
  kind: KafkaTopic
  apiVersion: kafka.strimzi.io/v1beta1
  name: my-topic

```

### 21.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 21.3.2.2. Procedure for using the cluster CLI

1. Save the **hoist-field-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f hoist-field-action-binding.yaml
```

### 21.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step hoist-field-action -p "step-0.field=The Field"
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 21.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//hoist-field-action.kamelet.yaml>

## CHAPTER 22. HTTP SINK

Forwards an event to a HTTP endpoint

### 22.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **http-sink** Kamelet:

Property	Name	Description	Type	Default	Example
url *	URL	The URL to send data to	string		<b>"https://my-service/path"</b>
method	Method	The HTTP method to use	string	<b>"POST"</b>	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 22.2. DEPENDENCIES

At runtime, the **http-sink** Kamelet relies upon the presence of the following dependencies:

- camel:http
- camel:kamelet
- camel:core

### 22.3. USAGE

This section describes how you can use the **http-sink**.

#### 22.3.1. Knative Sink

You can use the **http-sink** Kamelet as a Knative sink by binding it to a Knative object.

##### http-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: http-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
```

```

ref:
  kind: Kamelet
  apiVersion: camel.apache.org/v1alpha1
  name: http-sink
properties:
  url: "https://my-service/path"

```

### 22.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

### 22.3.1.2. Procedure for using the cluster CLI

1. Save the **http-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f http-sink-binding.yaml
```

### 22.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel http-sink -p "sink.url=https://my-service/path"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 22.3.2. Kafka Sink

You can use the **http-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

### http-sink-binding.yaml

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: http-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: http-sink
  properties:
    url: "https://my-service/path"

```

### 22.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 22.3.2.2. Procedure for using the cluster CLI

1. Save the **http-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f http-sink-binding.yaml
```

### 22.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic http-sink -p "sink.url=https://my-service/path"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 22.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//http-sink.kamelet.yaml>

## CHAPTER 23. INSERT FIELD ACTION

Adds a custom field with a constant value to the message in transit

### 23.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **insert-field-action** Kamelet:

Property	Name	Description	Type	Default	Example
<b>field *</b>	Field	The name of the field to be added	string		
<b>value *</b>	Value	The value of the field	string		



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 23.2. DEPENDENCIES

At runtime, the **insert-field-action** Kamelet relies upon the presence of the following dependencies:

- github:openshift-integration.kamelet-catalog:camel-kamelets-utils:kamelet-catalog-1.6-SNAPSHOT
- camel:core
- camel:jackson
- camel:kamelet

### 23.3. USAGE

This section describes how you can use the **insert-field-action**.

#### 23.3.1. Knative Action

You can use the **insert-field-action** Kamelet as an intermediate step in a Knative binding.

##### **insert-field-action-binding.yaml**

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: insert-field-action-binding
spec:
  source:
    ref:
      kind: Kamelet
```

```

  apiVersion: camel.apache.org/v1alpha1
  name: timer-source
  properties:
    message: '{"foo":"John"}'
  steps:
  - ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: json-deserialize-action
  - ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: insert-field-action
  properties:
    field: "The Field"
    value: "The Value"
  sink:
    ref:
    kind: Channel
    apiVersion: messaging.knative.dev/v1
    name: mychannel

```

### 23.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

### 23.3.1.2. Procedure for using the cluster CLI

1. Save the **insert-field-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f insert-field-action-binding.yaml
```

### 23.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind --name insert-field-action-binding timer-source?message='{"foo":"John"}' --step json-deserialize-action --step insert-field-action -p step-1.field='The Field' -p step-1.value='The Value' channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

## 23.3.2. Kafka Action

You can use the **insert-field-action** Kamelet as an intermediate step in a Kafka binding.

### insert-field-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
```

```

kind: KameletBinding
metadata:
  name: insert-field-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: '{"foo":"John"}'
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: json-deserialize-action
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: insert-field-action
    properties:
      field: "The Field"
      value: "The Value"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic

```

### 23.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 23.3.2.2. Procedure for using the cluster CLI

1. Save the **insert-field-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f insert-field-action-binding.yaml
```

### 23.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind --name insert-field-action-binding timer-source?message='{"foo":"John"}' --step json-deserialize-action --step insert-field-action -p step-1.field='The Field' -p step-1.value='The Value' kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 23.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//insert-field-action.kamelet.yaml>



## CHAPTER 24. INSERT HEADER ACTION

Adds an header with a constant value to the message in transit

### 24.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **insert-header-action** Kamelet:

Property	Name	Description	Type	Default	Example
<b>name *</b>	Name	The name of the header to be added. For Knative only, the name of the header requires a CloudEvent (ce-) prefix.	string		
<b>value *</b>	Value	The value of the header	string		



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 24.2. DEPENDENCIES

At runtime, the **insert-header-action** Kamelet relies upon the presence of the following dependencies:

- camel:core
- camel:kamelet

### 24.3. USAGE

This section describes how you can use the **insert-header-action**.

#### 24.3.1. Knative Action

You can use the **insert-header-action** Kamelet as an intermediate step in a Knative binding.

##### insert-header-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: insert-header-action-binding
spec:
  source:
    ref:
```

```

kind: Kamelet
apiVersion: camel.apache.org/v1alpha1
name: timer-source
properties:
  message: "Hello"
steps:
- ref:
  kind: Kamelet
  apiVersion: camel.apache.org/v1alpha1
  name: insert-header-action
properties:
  name: "The Name"
  value: "The Value"
sink:
  ref:
  kind: Channel
  apiVersion: messaging.knative.dev/v1
  name: mychannel

```

### 24.3.1.1. Prerequisite

Make sure you have **Red Hat Integration - Camel K** installed into the OpenShift cluster you're connected to.

### 24.3.1.2. Procedure for using the cluster CLI

1. Save the **insert-header-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f insert-header-action-binding.yaml
```

### 24.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step insert-header-action -p "step-0.name=The Name" -p "step-0.value=The Value" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

## 24.3.2. Kafka Action

You can use the **insert-header-action** Kamelet as an intermediate step in a Kafka binding.

### insert-header-action-binding.yaml

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: insert-header-action-binding
spec:

```

```

source:
  ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: timer-source
  properties:
    message: "Hello"
steps:
- ref:
  kind: Kamelet
  apiVersion: camel.apache.org/v1alpha1
  name: insert-header-action
  properties:
    name: "The Name"
    value: "The Value"
sink:
  ref:
    kind: KafkaTopic
    apiVersion: kafka.strimzi.io/v1beta1
    name: my-topic

```

### 24.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 24.3.2.2. Procedure for using the cluster CLI

1. Save the **insert-header-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f insert-header-action-binding.yaml
```

### 24.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step insert-header-action -p "step-0.name=The Name" -p "step-0.value=The Value" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 24.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//insert-header-action.kamelet.yaml>

## CHAPTER 25. IS TOMBSTONE FILTER ACTION

Filter based on the presence of body or not

### 25.1. CONFIGURATION OPTIONS

The **is-tombstone-filter-action** Kamelet does not specify any configuration option.

### 25.2. DEPENDENCIES

At runtime, the **is-tombstone-filter-action** Kamelet relies upon the presence of the following dependencies:

- camel:core
- camel:kamelet

### 25.3. USAGE

This section describes how you can use the **is-tombstone-filter-action**.

#### 25.3.1. Knative Action

You can use the **is-tombstone-filter-action** Kamelet as an intermediate step in a Knative binding.

##### is-tombstone-filter-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: is-tombstone-filter-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: is-tombstone-filter-action
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

##### 25.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

### 25.3.1.2. Procedure for using the cluster CLI

1. Save the **is-tombstone-filter-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f is-tombstone-filter-action-binding.yaml
```

### 25.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step is-tombstone-filter-action channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

## 25.3.2. Kafka Action

You can use the **is-tombstone-filter-action** Kamelet as an intermediate step in a Kafka binding.

### is-tombstone-filter-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: is-tombstone-filter-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: is-tombstone-filter-action
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

### 25.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 25.3.2.2. Procedure for using the cluster CLI

1. Save the **is-tombstone-filter-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f is-tombstone-filter-action-binding.yaml
```

### 25.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step is-tombstone-filter-action  
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 25.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//is-tombstone-filter-action.kamelet.yaml>

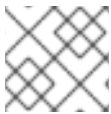
## CHAPTER 26. JIRA SOURCE

Receive notifications about new issues from Jira.

### 26.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **jira-source** Kamelet:

Property	Name	Description	Type	Default	Example
<b>jiraUrl *</b>	Jira URL	The URL of your instance of Jira	string		<b>"http://my_jira.com:8081"</b>
<b>password *</b>	Password	The password to access Jira	string		
<b>username *</b>	Username	The username to access Jira	string		
jql	JQL	A query to filter issues	string		<b>"project=MyProject"</b>



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 26.2. DEPENDENCIES

At runtime, the **jira-source** Kamelet relies upon the presence of the following dependencies:

- camel:jackson
- camel:kamelet
- camel:jira

### 26.3. USAGE

This section describes how you can use the **jira-source**.

#### 26.3.1. Knative Source

You can use the **jira-source** Kamelet as a Knative source by binding it to a Knative object.

##### **jira-source-binding.yaml**

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: jira-source-binding
```

```
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: jira-source
    properties:
      jiraUrl: "http://my_jira.com:8081"
      password: "The Password"
      username: "The Username"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

### 26.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

### 26.3.1.2. Procedure for using the cluster CLI

1. Save the **jira-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f jira-source-binding.yaml
```

### 26.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind jira-source -p "source.jiraUrl=http://my_jira.com:8081" -p "source.password=The Password" -p "source.username=The Username" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

## 26.3.2. Kafka Source

You can use the **jira-source** Kamelet as a Kafka source by binding it to a Kafka topic.

### jira-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: jira-source-binding
spec:
  source:
    ref:
      kind: Kamelet
```



```

apiVersion: camel.apache.org/v1alpha1
name: jira-source
properties:
  jiraUrl: "http://my_jira.com:8081"
  password: "The Password"
  username: "The Username"
sink:
  ref:
    kind: KafkaTopic
    apiVersion: kafka.strimzi.io/v1beta1
    name: my-topic

```

### 26.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 26.3.2.2. Procedure for using the cluster CLI

1. Save the **jira-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f jira-source-binding.yaml
```

### 26.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind jira-source -p "source.jiraUrl=http://my_jira.com:8081" -p "source.password=The Password" -p "source.username=The Username" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 26.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//jira-source.kamelet.yaml>

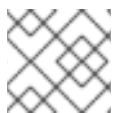
## CHAPTER 27. JMS - AMQP 1.0 KAMELET SINK

A Kamelet that can produce events to any AMQP 1.0 compliant message broker using the Apache Qpid JMS client

### 27.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **jms-amqp-10-sink** Kamelet:

Property	Name	Description	Type	Default	Example
<b>destinationName *</b>	Destination Name	The JMS destination name	string		
<b>remoteURI *</b>	Broker URL	The JMS URL	string		<b>"amqp://my-host:31616"</b>
destinationType	Destination Type	The JMS destination type (i.e.: queue or topic)	string	<b>"queue"</b>	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 27.2. DEPENDENCIES

At runtime, the **jms-amqp-10-sink** Kamelet relies upon the presence of the following dependencies:

- camel:jms
- camel:kamelet
- mvn:org.apache.qpid:qpid-jms-client:0.55.0

### 27.3. USAGE

This section describes how you can use the **jms-amqp-10-sink**.

#### 27.3.1. Knative Sink

You can use the **jms-amqp-10-sink** Kamelet as a Knative sink by binding it to a Knative object.

##### **jms-amqp-10-sink-binding.yaml**

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: jms-amqp-10-sink-binding
spec:
  source:
```

```

ref:
  kind: Channel
  apiVersion: messaging.knative.dev/v1
  name: mychannel
sink:
  ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: jms-amqp-10-sink
  properties:
    destinationName: "The Destination Name"
    remoteURI: "amqp://my-host:31616"

```

### 27.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

### 27.3.1.2. Procedure for using the cluster CLI

1. Save the **jms-amqp-10-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f jms-amqp-10-sink-binding.yaml
```

### 27.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel jms-amqp-10-sink -p "sink.destinationName=The Destination Name" -p "sink.remoteURI=amqp://my-host:31616"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 27.3.2. Kafka Sink

You can use the **jms-amqp-10-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

### jms-amqp-10-sink-binding.yaml

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: jms-amqp-10-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:

```

```
ref:
  kind: Kamelet
  apiVersion: camel.apache.org/v1alpha1
  name: jms-amqp-10-sink
properties:
  destinationName: "The Destination Name"
  remoteURI: "amqp://my-host:31616"
```

### 27.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 27.3.2.2. Procedure for using the cluster CLI

1. Save the **jms-amqp-10-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f jms-amqp-10-sink-binding.yaml
```

### 27.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic jms-amqp-10-sink -p
"sink.destinationName=The Destination Name" -p "sink.remoteURI=amqp://my-host:31616"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 27.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//jms-amqp-10-sink.kamelet.yaml>

## CHAPTER 28. JMS - AMQP 1.0 KAMELET SOURCE

A Kamelet that can consume events from any AMQP 1.0 compliant message broker using the Apache Qpid JMS client

### 28.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **jms-amqp-10-source** Kamelet:

Property	Name	Description	Type	Default	Example
<b>destinationName *</b>	Destination Name	The JMS destination name	string		
<b>remoteURI *</b>	Broker URL	The JMS URL	string		<b>"amqp://my-host:31616"</b>
destinationType	Destination Type	The JMS destination type (i.e.: queue or topic)	string	<b>"queue"</b>	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 28.2. DEPENDENCIES

At runtime, the **jms-amqp-10-source** Kamelet relies upon the presence of the following dependencies:

- camel:jms
- camel:kamelet
- mvn:org.apache.qpid:qpid-jms-client:0.55.0

### 28.3. USAGE

This section describes how you can use the **jms-amqp-10-source**.

#### 28.3.1. Knative Source

You can use the **jms-amqp-10-source** Kamelet as a Knative source by binding it to a Knative object.

##### **jms-amqp-10-source-binding.yaml**

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: jms-amqp-10-source-binding
spec:
```

```

source:
  ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: jms-amqp-10-source
  properties:
    destinationName: "The Destination Name"
    remoteURI: "amqp://my-host:31616"
sink:
  ref:
    kind: Channel
    apiVersion: messaging.knative.dev/v1
    name: mychannel

```

### 28.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

### 28.3.1.2. Procedure for using the cluster CLI

1. Save the **jms-amqp-10-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f jms-amqp-10-source-binding.yaml
```

### 28.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind jms-amqp-10-source -p "source.destinationName=The Destination Name" -p "source.remoteURI=amqp://my-host:31616" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

## 28.3.2. Kafka Source

You can use the **jms-amqp-10-source** Kamelet as a Kafka source by binding it to a Kafka topic.

### jms-amqp-10-source-binding.yaml

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: jms-amqp-10-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: jms-amqp-10-source

```

```

properties:
  destinationName: "The Destination Name"
  remoteURI: "amqp://my-host:31616"
sink:
  ref:
    kind: KafkaTopic
    apiVersion: kafka.strimzi.io/v1beta1
    name: my-topic

```

### 28.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 28.3.2.2. Procedure for using the cluster CLI

1. Save the **jms-amqp-10-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f jms-amqp-10-source-binding.yaml
```

### 28.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind jms-amqp-10-source -p "source.destinationName=The Destination Name" -p "source.remoteURI=amqp://my-host:31616" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 28.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//jms-amqp-10-source.kamelet.yaml>

## CHAPTER 29. JMS - IBM MQ KAMELET SINK

A Kamelet that can produce events to an IBM MQ message queue using JMS.

### 29.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **jms-ibm-mq-sink** Kamelet:

Property	Name	Description	Type	Default	Example
<b>channel *</b>	IBM MQ Channel	Name of the IBM MQ Channel	string		
<b>destination Name *</b>	Destination Name	The destination name	string		
<b>password *</b>	Password	Password to authenticate to IBM MQ server	string		
<b>queueManager *</b>	IBM MQ Queue Manager	Name of the IBM MQ Queue Manager	string		
<b>serverName *</b>	IBM MQ Server name	IBM MQ Server name or address	string		
<b>serverPort *</b>	IBM MQ Server Port	IBM MQ Server port	integer	<b>1414</b>	
<b>username *</b>	Username	Username to authenticate to IBM MQ server	string		
clientId	IBM MQ Client ID	Name of the IBM MQ Client ID	string		
destinationType	Destination Type	The JMS destination type (queue or topic)	string	<b>"queue"</b>	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 29.2. DEPENDENCIES

At runtime, the **jms-ibm-mq-sink** Kamelet relies upon the presence of the following dependencies:

- camel:jms



- camel:kamelet
- mvn:com.ibm.mq:com.ibm.mq.allclient:9.2.5.0

## 29.3. USAGE

This section describes how you can use the **jms-ibm-mq-sink**.

### 29.3.1. Knative Sink

You can use the **jms-ibm-mq-sink** Kamelet as a Knative sink by binding it to a Knative object.

#### jms-ibm-mq-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: jms-ibm-mq-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  properties:
    serverName: "10.103.41.245"
    serverPort: "1414"
    destinationType: "queue"
    destinationName: "DEV.QUEUE.1"
    queueManager: QM1
    channel: DEV.APP.SVRCONN
    username: app
    password: passw0rd
```

#### 29.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

#### 29.3.1.2. Procedure for using the cluster CLI

1. Save the **jms-ibm-mq-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f jms-ibm-mq-sink-binding.yaml
```

### 29.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind --name jms-ibm-mq-sink-binding timer-source?message="Hello IBM MQ!" 'jms-ibm-mq-sink?
serverName=10.103.41.245&serverPort=1414&destinationType=queue&destinationName=DEV.QUEUE.1&queueManager=QM1&channel=DEV.APP.SVRCONN&username=app&password=passw0rd'
```

This command creates the KameletBinding in the current namespace on the cluster.

### 29.3.2. Kafka Sink

You can use the **jms-ibm-mq-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

#### jms-ibm-mq-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: jms-ibm-mq-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: jms-ibm-mq-sink
  properties:
    serverName: "10.103.41.245"
    serverPort: "1414"
    destinationType: "queue"
    destinationName: "DEV.QUEUE.1"
    queueManager: QM1
    channel: DEV.APP.SVRCONN
    username: app
    password: passw0rd
```

#### 29.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

#### 29.3.2.2. Procedure for using the cluster CLI

1. Save the **jms-ibm-mq-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

-

```
oc apply -f jms-ibm-mq-sink-binding.yaml
```

### 29.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind --name jms-ibm-mq-sink-binding timer-source?message="Hello IBM MQ!" 'jms-ibm-mq-sink?serverName=10.103.41.245&serverPort=1414&destinationType=queue&destinationName=DEV.QUEUE.1&queueManager=QM1&channel=DEV.APP.SVRCONN&username=app&password=passw0rd'
```

This command creates the KameletBinding in the current namespace on the cluster.

## 29.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//jms-ibm-mq-sink.kamelet.yaml>

## CHAPTER 30. JMS - IBM MQ KAMELET SOURCE

A Kamelet that can read events from an IBM MQ message queue using JMS.

### 30.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **jms-ibm-mq-source** Kamelet:

Property	Name	Description	Type	Default	Example
<b>channel *</b>	IBM MQ Channel	Name of the IBM MQ Channel	string		
<b>destination Name *</b>	Destination Name	The destination name	string		
<b>password *</b>	Password	Password to authenticate to IBM MQ server	string		
<b>queueManager *</b>	IBM MQ Queue Manager	Name of the IBM MQ Queue Manager	string		
<b>serverName *</b>	IBM MQ Server name	IBM MQ Server name or address	string		
<b>serverPort *</b>	IBM MQ Server Port	IBM MQ Server port	integer	<b>1414</b>	
<b>username *</b>	Username	Username to authenticate to IBM MQ server	string		
clientId	IBM MQ Client ID	Name of the IBM MQ Client ID	string		
destinationType	Destination Type	The JMS destination type (queue or topic)	string	<b>"queue"</b>	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 30.2. DEPENDENCIES

At runtime, the **jms-ibm-mq-source** Kamelet relies upon the presence of the following dependencies:

- camel:jms
- camel:kamelet
- mvn:com.ibm.mq:com.ibm.mq.allclient:9.2.5.0

## 30.3. USAGE

This section describes how you can use the **jms-ibm-mq-source**.

### 30.3.1. Knative Source

You can use the **jms-ibm-mq-source** Kamelet as a Knative source by binding it to a Knative object.

#### jms-ibm-mq-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: jms-ibm-mq-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: jms-ibm-mq-source
    properties:
      serverName: "10.103.41.245"
      serverPort: "1414"
      destinationType: "queue"
      destinationName: "DEV.QUEUE.1"
      queueManager: QM1
      channel: DEV.APP.SVRCONN
      username: app
      password: passw0rd
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

#### 30.3.1.1. Prerequisite

Make sure you have "**Red Hat Integration - Camel K**" installed into the OpenShift cluster you're connected to.

#### 30.3.1.2. Procedure for using the cluster CLI

1. Save the **jms-ibm-mq-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f jms-ibm-mq-source-binding.yaml
```

### 30.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind --name jms-ibm-mq-source-binding 'jms-ibm-mq-source?
serverName=10.103.41.245&serverPort=1414&destinationType=queue&destinationName=DEV.QUEU
E.1&queueManager=QM1&channel=DEV.APP.SVRCONN&username=app&password=passw0rd'
channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

## 30.3.2. Kafka Source

You can use the **jms-ibm-mq-source** Kamelet as a Kafka source by binding it to a Kafka topic.

### jms-ibm-mq-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: jms-ibm-mq-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: jms-ibm-mq-source
    properties:
      serverName: "10.103.41.245"
      serverPort: "1414"
      destinationType: "queue"
      destinationName: "DEV.QUEUE.1"
      queueManager: QM1
      channel: DEV.APP.SVRCONN
      username: app
      password: passw0rd
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

#### 30.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

#### 30.3.2.2. Procedure for using the cluster CLI

1. Save the **jms-ibm-mq-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

-

```
oc apply -f jms-ibm-mq-source-binding.yaml
```

### 30.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind --name jms-ibm-mq-source-binding 'jms-ibm-mq-source?  
serverName=10.103.41.245&serverPort=1414&destinationType=queue&destinationName=DEV.QUEU  
E.1&queueManager=QM1&channel=DEV.APP.SVRCONN&username=app&password=passw0rd'  
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 30.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//jms-ibm-mq-source.kamelet.yaml>

## CHAPTER 31. JSON DESERIALIZE ACTION

Deserialize payload to JSON

### 31.1. CONFIGURATION OPTIONS

The **json-deserialize-action** Kamelet does not specify any configuration option.

### 31.2. DEPENDENCIES

At runtime, the **json-deserialize-action** Kamelet relies upon the presence of the following dependencies:

- camel:kamelet
- camel:core
- camel:jackson

### 31.3. USAGE

This section describes how you can use the **json-deserialize-action**.

#### 31.3.1. Knative Action

You can use the **json-deserialize-action** Kamelet as an intermediate step in a Knative binding.

##### **json-deserialize-action-binding.yaml**

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: json-deserialize-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: json-deserialize-action
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

##### 31.3.1.1. Prerequisite



Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

### 31.3.1.2. Procedure for using the cluster CLI

1. Save the **json-deserialize-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f json-deserialize-action-binding.yaml
```

### 31.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step json-deserialize-action channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

## 31.3.2. Kafka Action

You can use the **json-deserialize-action** Kamelet as an intermediate step in a Kafka binding.

### json-deserialize-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: json-deserialize-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: json-deserialize-action
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

### 31.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 31.3.2.2. Procedure for using the cluster CLI

1. Save the **json-deserialize-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f json-deserialize-action-binding.yaml
```

### 31.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step json-deserialize-action  
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 31.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//json-deserialize-action.kamelet.yaml>

## CHAPTER 32. JSON SERIALIZE ACTION

Serialize payload to JSON

### 32.1. CONFIGURATION OPTIONS

The **json-serialize-action** Kamelet does not specify any configuration option.

### 32.2. DEPENDENCIES

At runtime, the **json-serialize-action** Kamelet relies upon the presence of the following dependencies:

- camel:kamelet
- camel:core
- camel:jackson

### 32.3. USAGE

This section describes how you can use the **json-serialize-action**.

#### 32.3.1. Knative Action

You can use the **json-serialize-action** Kamelet as an intermediate step in a Knative binding.

##### json-serialize-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: json-serialize-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: json-serialize-action
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

##### 32.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

### 32.3.1.2. Procedure for using the cluster CLI

1. Save the **json-serialize-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f json-serialize-action-binding.yaml
```

### 32.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step json-serialize-action channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

## 32.3.2. Kafka Action

You can use the **json-serialize-action** Kamelet as an intermediate step in a Kafka binding.

### json-serialize-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: json-serialize-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: json-serialize-action
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

### 32.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

### 32.3.2.2. Procedure for using the cluster CLI

1. Save the **json-serialize-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f json-serialize-action-binding.yaml
```

### 32.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step json-serialize-action  
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 32.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//json-serialize-action.kamelet.yaml>

## CHAPTER 33. KAFKA SINK

Send data to Kafka topics.

The Kamelet is able to understand the following headers to be set:

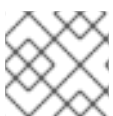
- **key / ce-key:** as message key
- **partition-key / ce-partitionkey:** as message partition key

Both the headers are optional.

### 33.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **kafka-sink** Kamelet:

Property	Name	Description	Type	Default	Example
<b>bootstrapServers *</b>	Brokers	Comma separated list of Kafka Broker URLs	string		
<b>password *</b>	Password	Password to authenticate to kafka	string		
<b>topic *</b>	Topic Names	Comma separated list of Kafka topic names	string		
<b>user *</b>	Username	Username to authenticate to Kafka	string		
saslMechanism	SASL Mechanism	The Simple Authentication and Security Layer (SASL) Mechanism used.	string	<b>"PLAIN"</b>	
securityProtocol	Security Protocol	Protocol used to communicate with brokers. SASL_PLAINTEXT, PLAINTEXT, SASL_SSL and SSL are supported	string	<b>"SASL_SSL"</b>	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

## 33.2. DEPENDENCIES

At runtime, the `kafka-sink` Kamelet relies upon the presence of the following dependencies:

- `camel:kafka`
- `camel:kamelet`

## 33.3. USAGE

This section describes how you can use the **kafka-sink**.

### 33.3.1. Knative Sink

You can use the **kafka-sink** Kamelet as a Knative sink by binding it to a Knative object.

#### kafka-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: kafka-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: kafka-sink
  properties:
    bootstrapServers: "The Brokers"
    password: "The Password"
    topic: "The Topic Names"
    user: "The Username"
```

#### 33.3.1.1. Prerequisite

Make sure you have "**Red Hat Integration - Camel K**" installed into the OpenShift cluster you're connected to.

#### 33.3.1.2. Procedure for using the cluster CLI

1. Save the **kafka-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f kafka-sink-binding.yaml
```

### 33.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel kafka-sink -p "sink.bootstrapServers=The Brokers" -p "sink.password=The Password" -p "sink.topic=The Topic Names" -p "sink.user=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

### 33.3.2. Kafka Sink

You can use the **kafka-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

#### kafka-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: kafka-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: kafka-sink
  properties:
    bootstrapServers: "The Brokers"
    password: "The Password"
    topic: "The Topic Names"
    user: "The Username"
```

#### 33.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

#### 33.3.2.2. Procedure for using the cluster CLI

1. Save the **kafka-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f kafka-sink-binding.yaml
```

#### 33.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:



```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic kafka-sink -p "sink.bootstrapServers=The  
Brokers" -p "sink.password=The Password" -p "sink.topic=The Topic Names" -p "sink.user=The  
Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 33.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//kafka-sink.kamelet.yaml>

## CHAPTER 34. KAFKA SOURCE

Receive data from Kafka topics.

### 34.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **kafka-source** Kamelet:

Property	Name	Description	Type	Default	Example
<b>topic *</b>	Topic Names	Comma separated list of Kafka topic names	string		
<b>bootstrapServers *</b>	Brokers	Comma separated list of Kafka Broker URLs	string		
securityProtocol	Security Protocol	Protocol used to communicate with brokers. SASL_PLAINTEXT, PLAINTEXT, SASL_SSL and SSL are supported	string	<b>"SASL_SSL"</b>	
saslMechanism	SASL Mechanism	The Simple Authentication and Security Layer (SASL) Mechanism used.	string	<b>"PLAIN"</b>	
<b>user *</b>	Username	Username to authenticate to Kafka	string		
<b>password *</b>	Password	Password to authenticate to kafka	string		
autoCommitEnable	Auto Commit Enable	If true, periodically commit to ZooKeeper the offset of messages already fetched by the consumer.	boolean	<b>true</b>	
allowManualCommit	Allow Manual Commit	Whether to allow doing manual commits	boolean	<b>false</b>	

Property	Name	Description	Type	Default	Example
autoOffsetReset	Auto Offset Reset	What to do when there is no initial offset. There are 3 enums and the value can be one of latest, earliest, none	string	<b>"latest"</b>	
pollOnError	Poll On Error Behavior	What to do if kafka threw an exception while polling for new messages. There are 5 enums and the value can be one of DISCARD, ERROR_HANDLER, RECONNECT, RETRY, STOP	string	<b>"ERROR_HANDLER"</b>	
deserializeHeaders	Automatically Deserialize Headers	When enabled the Kamelet source will deserialize all message headers to String representation. The default is <b>false</b> .	boolean	<b>true</b>	

**NOTE**

Fields marked with an asterisk (\*) are mandatory.

## 34.2. DEPENDENCIES

At runtime, the `kafka-source` Kamelet relies upon the presence of the following dependencies:

- camel:kafka
- camel:kamelet
- camel:core

## 34.3. USAGE

This section describes how you can use the **kafka-source**.

### 34.3.1. Knative Source

You can use the **kafka-source** Kamelet as a Knative source by binding it to a Knative object.

**kafka-source-binding.yaml**

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: kafka-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: kafka-source
    properties:
      bootstrapServers: "The Brokers"
      password: "The Password"
      topic: "The Topic Names"
      user: "The Username"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel

```

#### 34.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

#### 34.3.1.2. Procedure for using the cluster CLI

1. Save the **kafka-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f kafka-source-binding.yaml
```

#### 34.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind kafka-source -p "source.bootstrapServers=The Brokers" -p "source.password=The Password" -p "source.topic=The Topic Names" -p "source.user=The Username" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

### 34.3.2. Kafka Source

You can use the **kafka-source** Kamelet as a Kafka source by binding it to a Kafka topic.

#### kafka-source-binding.yaml

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:

```

```

name: kafka-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: kafka-source
    properties:
      bootstrapServers: "The Brokers"
      password: "The Password"
      topic: "The Topic Names"
      user: "The Username"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic

```

### 34.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 34.3.2.2. Procedure for using the cluster CLI

1. Save the **kafka-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f kafka-source-binding.yaml
```

### 34.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind kafka-source -p "source.bootstrapServers=The Brokers" -p "source.password=The Password" -p "source.topic=The Topic Names" -p "source.user=The Username"
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 34.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//kafka-source.kamelet.yaml>

## CHAPTER 35. KAFKA TOPIC NAME MATCHES FILTER ACTION

Filter based on kafka topic value compared to regex

### 35.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **topic-name-matches-filter-action** Kamelet:

Property	Name	Description	Type	Default	Example
<b>regex *</b>	Regex	The Regex to Evaluate against the Kafka topic name	string		



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 35.2. DEPENDENCIES

At runtime, the **topic-name-matches-filter-action** Kamelet relies upon the presence of the following dependencies:

- camel:core
- camel:kamelet

### 35.3. USAGE

This section describes how you can use the **topic-name-matches-filter-action**.

#### 35.3.1. Kafka Action

You can use the **topic-name-matches-filter-action** Kamelet as an intermediate step in a Kafka binding.

##### topic-name-matches-filter-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: topic-name-matches-filter-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
  properties:
    message: "Hello"
  steps:
```

```
- ref:
  kind: Kamelet
  apiVersion: camel.apache.org/v1alpha1
  name: topic-name-matches-filter-action
  properties:
    regex: "The Regex"
sink:
  ref:
    kind: KafkaTopic
    apiVersion: kafka.strimzi.io/v1beta1
    name: my-topic
```

### 35.3.1.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 35.3.1.2. Procedure for using the cluster CLI

1. Save the **topic-name-matches-filter-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f topic-name-matches-filter-action-binding.yaml
```

### 35.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step topic-name-matches-filter-action -p "step-0.regex=The Regex" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 35.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//topic-name-matches-filter-action.kamelet.yaml>

## CHAPTER 36. LOG SINK

A sink that logs all data that it receives, useful for debugging purposes.

### 36.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **log-sink** Kamelet:

Property	Name	Description	Type	Default	Example
showHeaders	Show Headers	Show the headers received	boolean	<b>false</b>	
showStreams	Show Streams	Show the stream bodies (they may not be available in following steps)	boolean	<b>false</b>	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 36.2. DEPENDENCIES

At runtime, the **log-sink** Kamelet relies upon the presence of the following dependencies:

- camel:kamelet
- camel:log

### 36.3. USAGE

This section describes how you can use the **log-sink**.

#### 36.3.1. Knative Sink

You can use the **log-sink** Kamelet as a Knative sink by binding it to a Knative object.

##### log-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: log-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
```



```

ref:
  kind: Kamelet
  apiVersion: camel.apache.org/v1alpha1
  name: log-sink

```

### 36.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

### 36.3.1.2. Procedure for using the cluster CLI

1. Save the **log-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f log-sink-binding.yaml
```

### 36.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel log-sink
```

This command creates the KameletBinding in the current namespace on the cluster.

## 36.3.2. Kafka Sink

You can use the **log-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

### log-sink-binding.yaml

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: log-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: log-sink

```

### 36.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 36.3.2.2. Procedure for using the cluster CLI

1. Save the **log-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f log-sink-binding.yaml
```

### 36.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic log-sink
```

This command creates the KameletBinding in the current namespace on the cluster.

## 36.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//log-sink.kamelet.yaml>

## CHAPTER 37. MARIADB SINK

Send data to a MariaDB Database.

This Kamelet expects a JSON as body. The mapping between the JSON fields and parameters is done by key, so if you have the following query:

```
'INSERT INTO accounts (username,city) VALUES (:#username,:#city)'
```

The Kamelet needs to receive as input something like:

```
'{"username":"oscerd", "city":"Rome"}'
```

### 37.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **mariadb-sink** Kamelet:

Property	Name	Description	Type	Default	Example
<b>databaseName *</b>	Database Name	The Database Name we are pointing	string		
<b>password *</b>	Password	The password to use for accessing a secured MariaDB Database	string		
<b>query *</b>	Query	The Query to execute against the MariaDB Database	string		<b>"INSERT INTO accounts (username,city) VALUES (:#username,:#city)"</b>
<b>serverName *</b>	Server Name	Server Name for the data source	string		<b>"localhost"</b>
<b>username *</b>	Username	The username to use for accessing a secured MariaDB Database	string		
serverPort	Server Port	Server Port for the data source	string	<b>3306</b>	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 37.2. DEPENDENCIES

At runtime, the **mariadb-sink** Kamelet relies upon the presence of the following dependencies:

- camel:jackson
- camel:kamelet
- camel:sql
- mvn:org.apache.commons:commons-dbcp2:2.7.0.redhat-00001
- mvn:org.mariadb.jdbc:mariadb-java-client

## 37.3. USAGE

This section describes how you can use the **mariadb-sink**.

### 37.3.1. Knative Sink

You can use the **mariadb-sink** Kamelet as a Knative sink by binding it to a Knative object.

#### **mariadb-sink-binding.yaml**

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: mariadb-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: mariadb-sink
  properties:
    databaseName: "The Database Name"
    password: "The Password"
    query: "INSERT INTO accounts (username,city) VALUES (:#username,:#city)"
    serverName: "localhost"
    username: "The Username"
```

#### 37.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

#### 37.3.1.2. Procedure for using the cluster CLI

1. Save the **mariadb-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the sink by using the following command:

```
oc apply -f mariadb-sink-binding.yaml
```

### 37.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel mariadb-sink -p "sink.databaseName=The Database Name" -p "sink.password=The Password" -p "sink.query=INSERT INTO accounts (username,city) VALUES (:#username,:#city)" -p "sink.serverName=localhost" -p "sink.username=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 37.3.2. Kafka Sink

You can use the **mariadb-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

### mariadb-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: mariadb-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: mariadb-sink
  properties:
    databaseName: "The Database Name"
    password: "The Password"
    query: "INSERT INTO accounts (username,city) VALUES (:#username,:#city)"
    serverName: "localhost"
    username: "The Username"
```

#### 37.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

#### 37.3.2.2. Procedure for using the cluster CLI

1. Save the **mariadb-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the sink by using the following command:

```
oc apply -f mariadb-sink-binding.yaml
```

### 37.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic mariadb-sink -p "sink.databaseName=The Database Name" -p "sink.password=The Password" -p "sink.query=INSERT INTO accounts (username,city) VALUES (:#username,:#city)" -p "sink.serverName=localhost" -p "sink.username=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 37.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//mariadb-sink.kamelet.yaml>

## CHAPTER 38. MASK FIELDS ACTION

Mask fields with a constant value in the message in transit

### 38.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **mask-field-action** Kamelet:

Property	Name	Description	Type	Default	Example
<b>fields *</b>	Fields	Comma separated list of fields to mask	string		
<b>replacement *</b>	Replacement	Replacement for the fields to be masked	string		



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 38.2. DEPENDENCIES

At runtime, the **mask-field-action** Kamelet relies upon the presence of the following dependencies:

- github:openshift-integration.kamelet-catalog:camel-kamelets-utils:kamelet-catalog-1.6-SNAPSHOT
- camel:jackson
- camel:kamelet
- camel:core

### 38.3. USAGE

This section describes how you can use the **mask-field-action**.

#### 38.3.1. Knative Action

You can use the **mask-field-action** Kamelet as an intermediate step in a Knative binding.

##### mask-field-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: mask-field-action-binding
spec:
  source:
    ref:
      kind: Kamelet
```

```

  apiVersion: camel.apache.org/v1alpha1
  name: timer-source
  properties:
    message: "Hello"
  steps:
  - ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: mask-field-action
  properties:
    fields: "The Fields"
    replacement: "The Replacement"
  sink:
  ref:
    kind: Channel
    apiVersion: messaging.knative.dev/v1
    name: mychannel

```

### 38.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

### 38.3.1.2. Procedure for using the cluster CLI

1. Save the **mask-field-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f mask-field-action-binding.yaml
```

### 38.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step mask-field-action -p "step-0.fields=The Fields" -p "step-0.replacement=The Replacement" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

## 38.3.2. Kafka Action

You can use the **mask-field-action** Kamelet as an intermediate step in a Kafka binding.

### mask-field-action-binding.yaml

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: mask-field-action-binding
spec:
  source:

```



```

ref:
  kind: Kamelet
  apiVersion: camel.apache.org/v1alpha1
  name: timer-source
properties:
  message: "Hello"
steps:
- ref:
  kind: Kamelet
  apiVersion: camel.apache.org/v1alpha1
  name: mask-field-action
properties:
  fields: "The Fields"
  replacement: "The Replacement"
sink:
  ref:
  kind: KafkaTopic
  apiVersion: kafka.strimzi.io/v1beta1
  name: my-topic

```

### 38.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 38.3.2.2. Procedure for using the cluster CLI

1. Save the **mask-field-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f mask-field-action-binding.yaml
```

### 38.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step mask-field-action -p "step-0.fields=The Fields" -p "step-0.replacement=The Replacement" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 38.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//mask-field-action.kamelet.yaml>

## CHAPTER 39. MESSAGE TIMESTAMP ROUTER ACTION

Update the topic field as a function of the original topic name and the record's timestamp field.

### 39.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **message-timestamp-router-action** Kamelet:

Property	Name	Description	Type	Default	Example
<b>timestamp Keys *</b>	Timestamp Keys	Comma separated list of Timestamp keys. The timestamp is taken from the first found field.	string		
timestampFormat	Timestamp Format	Format string for the timestamp that is compatible with <code>java.text.SimpleDateFormat</code> .	string	<b>"yyyyMMdd"</b>	
timestampKeyFormat	Timestamp Keys Format	Format of the timestamp keys. Possible values are 'timestamp' or any format string for the timestamp that is compatible with <code>java.text.SimpleDateFormat</code> . In case of 'timestamp' the field will be evaluated as milliseconds since 1970, so as a UNIX Timestamp.	string	<b>"timestamp"</b>	
topicFormat	Topic Format	Format string which can contain '[topic]' and '[timestamp]' as placeholders for the topic and timestamp, respectively.	string	<b>"topic-[timestamp]"</b>	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

## 39.2. DEPENDENCIES

At runtime, the **message-timestamp-router-action** Kamelet relies upon the presence of the following dependencies:

- mvn:org.apache.camel.kamelets:camel-kamelets-utils:1.0.0.fuse-800048-redhat-00001
- camel:jackson
- camel:kamelet
- camel:core

## 39.3. USAGE

This section describes how you can use the **message-timestamp-router-action**.

### 39.3.1. Knative Action

You can use the **message-timestamp-router-action** Kamelet as an intermediate step in a Knative binding.

#### message-timestamp-router-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: message-timestamp-router-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: message-timestamp-router-action
    properties:
      timestampKeys: "The Timestamp Keys"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

#### 39.3.1.1. Prerequisite

Make sure you have "**Red Hat Integration - Camel K**" installed into the OpenShift cluster you're connected to.

### 39.3.1.2. Procedure for using the cluster CLI

1. Save the **message-timestamp-router-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f message-timestamp-router-action-binding.yaml
```

### 39.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step message-timestamp-router-action -p "step-0.timestampKeys=The Timestamp Keys" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

## 39.3.2. Kafka Action

You can use the **message-timestamp-router-action** Kamelet as an intermediate step in a Kafka binding.

### message-timestamp-router-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: message-timestamp-router-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: message-timestamp-router-action
    properties:
      timestampKeys: "The Timestamp Keys"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

### 39.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 39.3.2.2. Procedure for using the cluster CLI

1. Save the **message-timestamp-router-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f message-timestamp-router-action-binding.yaml
```

### 39.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step message-timestamp-router-action -p "step-0.timestampKeys=The Timestamp Keys" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 39.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//message-timestamp-router-action.kamelet.yaml>

## CHAPTER 40. MONGODB SINK

Send documents to MongoDB.

This Kamelet expects a JSON as body.

Properties you can set as headers:

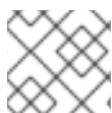
- **db-upsert** / **ce-dbupsert**: if the database should create the element if it does not exist. Boolean value.

### 40.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **mongodb-sink** Kamelet:

Property	Name	Description	Type	Default	Example
<b>collection</b> *	MongoDB Collection	Sets the name of the MongoDB collection to bind to this endpoint.	string		
<b>database</b> *	MongoDB Database	Sets the name of the MongoDB database to target.	string		
<b>hosts</b> *	MongoDB Hosts	Comma separated list of MongoDB Host Addresses in host:port format.	string		
createCollection	Collection	Create collection during initialisation if it doesn't exist.	boolean	<b>false</b>	
password	MongoDB Password	User password for accessing MongoDB.	string		
username	MongoDB Username	Username for accessing MongoDB.	string		

Property	Name	Description	Type	Default	Example
writeConcern	Write Concern	Configure the level of acknowledgment requested from MongoDB for write operations, possible values are ACKNOWLEDGED, W1, W2, W3, UNACKNOWLEDGED, JOURNALED, MAJORITY.	string		

**NOTE**

Fields marked with an asterisk (\*) are mandatory.

## 40.2. DEPENDENCIES

At runtime, the **mongodb-sink** Kamelet relies upon the presence of the following dependencies:

- camel:kamelet
- camel:mongodb
- camel:jackson

## 40.3. USAGE

This section describes how you can use the **mongodb-sink**.

### 40.3.1. Knative Sink

You can use the **mongodb-sink** Kamelet as a Knative sink by binding it to a Knative object.

#### mongodb-sink-binding.yaml

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: mongodb-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:

```

```

kind: Kamelet
apiVersion: camel.apache.org/v1alpha1
name: mongodb-sink
properties:
  collection: "The MongoDB Collection"
  database: "The MongoDB Database"
  hosts: "The MongoDB Hosts"

```

#### 40.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

#### 40.3.1.2. Procedure for using the cluster CLI

1. Save the **mongodb-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f mongodb-sink-binding.yaml
```

#### 40.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel mongodb-sink -p "sink.collection=The MongoDB Collection" -p "sink.database=The MongoDB Database" -p "sink.hosts=The MongoDB Hosts"
```

This command creates the KameletBinding in the current namespace on the cluster.

### 40.3.2. Kafka Sink

You can use the **mongodb-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

#### mongodb-sink-binding.yaml

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: mongodb-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: mongodb-sink
  properties:

```



```
collection: "The MongoDB Collection"  
database: "The MongoDB Database"  
hosts: "The MongoDB Hosts"
```

### 40.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 40.3.2.2. Procedure for using the cluster CLI

1. Save the **mongodb-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f mongodb-sink-binding.yaml
```

### 40.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic mongodb-sink -p "sink.collection=The  
MongoDB Collection" -p "sink.database=The MongoDB Database" -p "sink.hosts=The MongoDB  
Hosts"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 40.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//mongodb-sink.kamelet.yaml>

## CHAPTER 41. MONGODB SOURCE

Consume documents from MongoDB.

If the `persistentTailTracking` option will be enabled, the consumer will keep track of the last consumed message and on the next restart, the consumption will restart from that message. In case of `persistentTailTracking` enabled, the `tailTrackIncreasingField` must be provided (by default it is optional).

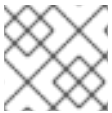
If the `persistentTailTracking` option won't be enabled, the consumer will consume the whole collection and wait in idle for new documents to consume.

### 41.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **mongodb-source** Kamelet:

Property	Name	Description	Type	Default	Example
<b>collection *</b>	MongoDB Collection	Sets the name of the MongoDB collection to bind to this endpoint.	string		
<b>database *</b>	MongoDB Database	Sets the name of the MongoDB database to target.	string		
<b>hosts *</b>	MongoDB Hosts	Comma separated list of MongoDB Host Addresses in host:port format.	string		
<b>password *</b>	MongoDB Password	User password for accessing MongoDB.	string		
<b>username *</b>	MongoDB Username	Username for accessing MongoDB. The username must be present in the MongoDB's authentication database (authenticationDatabase). By default, the MongoDB authenticationDatabase is 'admin'.	string		

Property	Name	Description	Type	Default	Example
<code>persistentTailTracking</code>	MongoDB Persistent Tail Tracking	Enable persistent tail tracking, which is a mechanism to keep track of the last consumed message across system restarts. The next time the system is up, the endpoint will recover the cursor from the point where it last stopped slurping records.	boolean	<b>false</b>	
<code>tailTrackIncreasingField</code>	MongoDB Tail Track Increasing Field	Correlation field in the incoming record which is of increasing nature and will be used to position the tailing cursor every time it is generated.	string		

**NOTE**

Fields marked with an asterisk (\*) are mandatory.

## 41.2. DEPENDENCIES

At runtime, the **mongodb-source** Kamelet relies upon the presence of the following dependencies:

- `camel:kamelet`
- `camel:mongodb`
- `camel:jackson`

## 41.3. USAGE

This section describes how you can use the **mongodb-source**.

### 41.3.1. Knative Source

You can use the **mongodb-source** Kamelet as a Knative source by binding it to a Knative object.

#### **mongodb-source-binding.yaml**

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
```

```

metadata:
  name: mongodb-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: mongodb-source
  properties:
    collection: "The MongoDB Collection"
    database: "The MongoDB Database"
    hosts: "The MongoDB Hosts"
    password: "The MongoDB Password"
    username: "The MongoDB Username"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel

```

#### 41.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

#### 41.3.1.2. Procedure for using the cluster CLI

1. Save the **mongodb-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f mongodb-source-binding.yaml
```

#### 41.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```

kamel bind mongodb-source -p "source.collection=The MongoDB Collection" -p
"source.database=The MongoDB Database" -p "source.hosts=The MongoDB Hosts" -p
"source.password=The MongoDB Password" -p "source.username=The MongoDB Username"
channel:mychannel

```

This command creates the KameletBinding in the current namespace on the cluster.

### 41.3.2. Kafka Source

You can use the **mongodb-source** Kamelet as a Kafka source by binding it to a Kafka topic.

#### mongodb-source-binding.yaml

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding

```

```

metadata:
  name: mongodb-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: mongodb-source
    properties:
      collection: "The MongoDB Collection"
      database: "The MongoDB Database"
      hosts: "The MongoDB Hosts"
      password: "The MongoDB Password"
      username: "The MongoDB Username"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic

```

### 41.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 41.3.2.2. Procedure for using the cluster CLI

1. Save the **mongodb-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f mongodb-source-binding.yaml
```

### 41.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind mongodb-source -p "source.collection=The MongoDB Collection" -p
"source.database=The MongoDB Database" -p "source.hosts=The MongoDB Hosts" -p
"source.password=The MongoDB Password" -p "source.username=The MongoDB Username"
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 41.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//mongodb-source.kamelet.yaml>

## CHAPTER 42. MYSQL SINK

Send data to a MySQL Database.

This Kamelet expects a JSON as body. The mapping between the JSON fields and parameters is done by key, so if you have the following query:

```
'INSERT INTO accounts (username,city) VALUES (:#username,:#city)'
```

The Kamelet needs to receive as input something like:

```
{ "username": "oscerd", "city": "Rome" }
```

### 42.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **mysql-sink** Kamelet:

Property	Name	Description	Type	Default	Example
<b>databaseName *</b>	Database Name	The Database Name we are pointing	string		
<b>password *</b>	Password	The password to use for accessing a secured MySQL Database	string		
<b>query *</b>	Query	The Query to execute against the MySQL Database	string		<b>"INSERT INTO accounts (username,city) VALUES (:#username,:#city)"</b>
<b>serverName *</b>	Server Name	Server Name for the data source	string		<b>"localhost"</b>
<b>username *</b>	Username	The username to use for accessing a secured MySQL Database	string		
serverPort	Server Port	Server Port for the data source	string	<b>3306</b>	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 42.2. DEPENDENCIES

At runtime, the **mysql-sink** Kamelet relies upon the presence of the following dependencies:

- camel:jackson
- camel:kamelet
- camel:sql
- mvn:org.apache.commons:commons-dbc2:2.7.0.redhat-00001
- mvn:mysql:mysql-connector-java

## 42.3. USAGE

This section describes how you can use the **mysql-sink**.

### 42.3.1. Knative Sink

You can use the **mysql-sink** Kamelet as a Knative sink by binding it to a Knative object.

#### mysql-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: mysql-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: mysql-sink
  properties:
    databaseName: "The Database Name"
    password: "The Password"
    query: "INSERT INTO accounts (username,city) VALUES (:#username,:#city)"
    serverName: "localhost"
    username: "The Username"
```

#### 42.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

#### 42.3.1.2. Procedure for using the cluster CLI

1. Save the **mysql-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the sink by using the following command:

```
oc apply -f mysql-sink-binding.yaml
```

### 42.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel mysql-sink -p "sink.databaseName=The Database Name" -p "sink.password=The Password" -p "sink.query=INSERT INTO accounts (username,city) VALUES (:#username,:#city)" -p "sink.serverName=localhost" -p "sink.username=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

### 42.3.2. Kafka Sink

You can use the **mysql-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

#### mysql-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: mysql-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: mysql-sink
  properties:
    databaseName: "The Database Name"
    password: "The Password"
    query: "INSERT INTO accounts (username,city) VALUES (:#username,:#city)"
    serverName: "localhost"
    username: "The Username"
```

#### 42.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

#### 42.3.2.2. Procedure for using the cluster CLI

1. Save the **mysql-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.



2. Run the sink by using the following command:

```
oc apply -f mysql-sink-binding.yaml
```

### 42.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic mysql-sink -p "sink.databaseName=The Database Name" -p "sink.password=The Password" -p "sink.query=INSERT INTO accounts (username,city) VALUES (:#username,:#city)" -p "sink.serverName=localhost" -p "sink.username=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 42.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//mysql-sink.kamelet.yaml>

## CHAPTER 43. POSTGRESQL SINK

Send data to a PostgreSQL Database.

This Kamelet expects a JSON as body. The mapping between the JSON fields and parameters is done by key, so if you have the following query:

```
'INSERT INTO accounts (username,city) VALUES (:#username,:#city)'
```

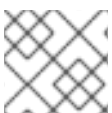
The Kamelet needs to receive as input something like:

```
{ "username": "oscerd", "city": "Rome" }
```

### 43.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **postgresql-sink** Kamelet:

Property	Name	Description	Type	Default	Example
<b>databaseName *</b>	Database Name	The Database Name we are pointing	string		
<b>password *</b>	Password	The password to use for accessing a secured PostgreSQL Database	string		
<b>query *</b>	Query	The Query to execute against the PostgreSQL Database	string		<b>"INSERT INTO accounts (username,city) VALUES (:#username,:#city)"</b>
<b>serverName *</b>	Server Name	Server Name for the data source	string		<b>"localhost"</b>
<b>username *</b>	Username	The username to use for accessing a secured PostgreSQL Database	string		
serverPort	Server Port	Server Port for the data source	string	<b>5432</b>	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

## 43.2. DEPENDENCIES

At runtime, the **postgresql-sink** Kamelet relies upon the presence of the following dependencies:

- camel:jackson
- camel:kamelet
- camel:sql
- mvn:org.postgresql:postgresql
- mvn:org.apache.commons:commons-dbcp2:2.7.0.redhat-00001

## 43.3. USAGE

This section describes how you can use the **postgresql-sink**.

### 43.3.1. Knative Sink

You can use the **postgresql-sink** Kamelet as a Knative sink by binding it to a Knative object.

#### postgresql-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: postgresql-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: postgresql-sink
  properties:
    databaseName: "The Database Name"
    password: "The Password"
    query: "INSERT INTO accounts (username,city) VALUES (:#username,:#city)"
    serverName: "localhost"
    username: "The Username"
```

#### 43.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

#### 43.3.1.2. Procedure for using the cluster CLI

1. Save the **postgresql-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f postgresql-sink-binding.yaml
```

### 43.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel postgresql-sink -p "sink.databaseName=The Database Name" -p "sink.password=The Password" -p "sink.query=INSERT INTO accounts (username,city) VALUES (:#username,:#city)" -p "sink.serverName=localhost" -p "sink.username=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

### 43.3.2. Kafka Sink

You can use the **postgresql-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

#### postgresql-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: postgresql-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: postgresql-sink
  properties:
    databaseName: "The Database Name"
    password: "The Password"
    query: "INSERT INTO accounts (username,city) VALUES (:#username,:#city)"
    serverName: "localhost"
    username: "The Username"
```

#### 43.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

#### 43.3.2.2. Procedure for using the cluster CLI

1. Save the **postgresql-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f postgresql-sink-binding.yaml
```

### 43.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic postgresql-sink -p "sink.databaseName=The Database Name" -p "sink.password=The Password" -p "sink.query=INSERT INTO accounts (username,city) VALUES (:#username,:#city)" -p "sink.serverName=localhost" -p "sink.username=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 43.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//postgresql-sink.kamelet.yaml>

## CHAPTER 44. PREDICATE FILTER ACTION

Filter based on a JsonPath Expression

### 44.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **predicate-filter-action** Kamelet:

Property	Name	Description	Type	Default	Example
<b>expression</b> *	Expression	The JsonPath Expression to evaluate, without the external parenthesis. Since this is a filter, the expression will be a negation, this means that if the foo field of the example is equals to John, the message will go ahead, otherwise it will be filtered out.	string		<b>"@.foo =~ /. *John/"</b>



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 44.2. DEPENDENCIES

At runtime, the **predicate-filter-action** Kamelet relies upon the presence of the following dependencies:

- camel:core
- camel:kamelet
- camel:jsonpath

### 44.3. USAGE

This section describes how you can use the **predicate-filter-action**.

#### 44.3.1. Knative Action

You can use the **predicate-filter-action** Kamelet as an intermediate step in a Knative binding.

**predicate-filter-action-binding.yaml**

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
```

```

metadata:
  name: predicate-filter-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: predicate-filter-action
      properties:
        expression: "@.foo =~ /. *John/"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel

```

#### 44.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

#### 44.3.1.2. Procedure for using the cluster CLI

1. Save the **predicate-filter-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f predicate-filter-action-binding.yaml
```

#### 44.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step predicate-filter-action -p "step-0.expression=@.foo =~ /. *John/" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

## 44.3.2. Kafka Action

You can use the **predicate-filter-action** Kamelet as an intermediate step in a Kafka binding.

#### predicate-filter-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
```

```

kind: KameletBinding
metadata:
  name: predicate-filter-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: predicate-filter-action
    properties:
      expression: "@.foo =~ /.*John/"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic

```

#### 44.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

#### 44.3.2.2. Procedure for using the cluster CLI

1. Save the **predicate-filter-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f predicate-filter-action-binding.yaml
```

#### 44.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step predicate-filter-action -p "step-0.expression=@.foo =~ /.*John/" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 44.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//predicate-filter-action.kamelet.yaml>



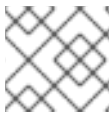
## CHAPTER 45. PROTOBUF DESERIALIZE ACTION

Deserialize payload to Protobuf

### 45.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **protobuf-deserialize-action** Kamelet:

Property	Name	Description	Type	Default	Example
<b>schema *</b>	Schema	The Protobuf schema to use during serialization (as single-line)	string		<b>"message Person { required string first = 1; required string last = 2; }"</b>



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 45.2. DEPENDENCIES

At runtime, the **protobuf-deserialize-action** Kamelet relies upon the presence of the following dependencies:

- github:openshift-integration.kamelet-catalog:camel-kamelets-utils:kamelet-catalog-1.6-SNAPSHOT
- camel:kamelet
- camel:core
- camel:jackson-protobuf

### 45.3. USAGE

This section describes how you can use the **protobuf-deserialize-action**.

#### 45.3.1. Knative Action

You can use the **protobuf-deserialize-action** Kamelet as an intermediate step in a Knative binding.

##### protobuf-deserialize-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: protobuf-deserialize-action-binding
spec:
  source:
    ref:
```

```

kind: Kamelet
apiVersion: camel.apache.org/v1alpha1
name: timer-source
properties:
  message: '{"first": "John", "last": "Doe"}'
steps:
- ref:
  kind: Kamelet
  apiVersion: camel.apache.org/v1alpha1
  name: json-deserialize-action
- ref:
  kind: Kamelet
  apiVersion: camel.apache.org/v1alpha1
  name: protobuf-serialize-action
properties:
  schema: "message Person { required string first = 1; required string last = 2; }"
- ref:
  kind: Kamelet
  apiVersion: camel.apache.org/v1alpha1
  name: protobuf-deserialize-action
properties:
  schema: "message Person { required string first = 1; required string last = 2; }"
sink:
  ref:
    kind: Channel
    apiVersion: messaging.knative.dev/v1
    name: mychannel

```

#### 45.3.1.1. Prerequisite

Make sure you have **Red Hat Integration - Camel K** installed into the OpenShift cluster you're connected to.

#### 45.3.1.2. Procedure for using the cluster CLI

1. Save the **protobuf-deserialize-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f protobuf-deserialize-action-binding.yaml
```

#### 45.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```

kamel bind --name protobuf-deserialize-action-binding timer-source?
message='{"first": "John", "last": "Doe"}' --step json-deserialize-action --step protobuf-serialize-action -p
step-1.schema='message Person { required string first = 1; required string last = 2; }' --step protobuf-
deserialize-action -p step-2.schema='message Person { required string first = 1; required string last =
2; }' channel:mychannel

```

This command creates the KameletBinding in the current namespace on the cluster.

### 45.3.2. Kafka Action

You can use the **protobuf-deserialize-action** Kamelet as an intermediate step in a Kafka binding.

#### protobuf-deserialize-action-binding.yaml

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: protobuf-deserialize-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: '{"first": "John", "last": "Doe"}'
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: json-deserialize-action
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: protobuf-serialize-action
    properties:
      schema: "message Person { required string first = 1; required string last = 2; }"
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: protobuf-deserialize-action
    properties:
      schema: "message Person { required string first = 1; required string last = 2; }"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic

```

#### 45.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

#### 45.3.2.2. Procedure for using the cluster CLI

1. Save the **protobuf-deserialize-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f protobuf-deserialize-action-binding.yaml
```

### 45.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind --name protobuf-deserialize-action-binding timer-source?  
message='{"first":"John","last":"Doe"}' --step json-deserialize-action --step protobuf-serialize-action -p  
step-1.schema='message Person { required string first = 1; required string last = 2; }' --step protobuf-  
deserialize-action -p step-2.schema='message Person { required string first = 1; required string last =  
2; }' kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 45.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//protobuf-deserialize-action.kamelet.yaml>

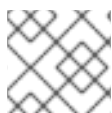
## CHAPTER 46. PROTOBUF SERIALIZE ACTION

Serialize payload to Protobuf

### 46.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **protobuf-serialize-action** Kamelet:

Property	Name	Description	Type	Default	Example
<b>schema *</b>	Schema	The Protobuf schema to use during serialization (as single-line)	string		<b>"message Person { required string first = 1; required string last = 2; }"</b>



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 46.2. DEPENDENCIES

At runtime, the **protobuf-serialize-action** Kamelet relies upon the presence of the following dependencies:

- github:openshift-integration.kamelet-catalog:camel-kamelets-utils:kamelet-catalog-1.6-SNAPSHOT
- camel:kamelet
- camel:core
- camel:jackson-protobuf

### 46.3. USAGE

This section describes how you can use the **protobuf-serialize-action**.

#### 46.3.1. Knative Action

You can use the **protobuf-serialize-action** Kamelet as an intermediate step in a Knative binding.

##### protobuf-serialize-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: protobuf-serialize-action-binding
spec:
  source:
    ref:
```

```

kind: Kamelet
apiVersion: camel.apache.org/v1alpha1
name: timer-source
properties:
  message: '{"first": "John", "last": "Doe"}'
steps:
- ref:
  kind: Kamelet
  apiVersion: camel.apache.org/v1alpha1
  name: json-deserialize-action
- ref:
  kind: Kamelet
  apiVersion: camel.apache.org/v1alpha1
  name: protobuf-serialize-action
properties:
  schema: "message Person { required string first = 1; required string last = 2; }"
sink:
  ref:
  kind: Channel
  apiVersion: messaging.knative.dev/v1
  name: mychannel

```

#### 46.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

#### 46.3.1.2. Procedure for using the cluster CLI

1. Save the **protobuf-serialize-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f protobuf-serialize-action-binding.yaml
```

#### 46.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```

kamel bind --name protobuf-serialize-action-binding timer-source?
message='{"first": "John", "last": "Doe"}' --step json-deserialize-action --step protobuf-serialize-action -p
step-1.schema='message Person { required string first = 1; required string last = 2; }'
channel:mychannel

```

This command creates the KameletBinding in the current namespace on the cluster.

### 46.3.2. Kafka Action

You can use the **protobuf-serialize-action** Kamelet as an intermediate step in a Kafka binding.

**protobuf-serialize-action-binding.yaml**

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: protobuf-serialize-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: '{"first": "John", "last": "Doe"}'
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: json-deserialize-action
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: protobuf-serialize-action
    properties:
      schema: "message Person { required string first = 1; required string last = 2; }"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic

```

### 46.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 46.3.2.2. Procedure for using the cluster CLI

1. Save the **protobuf-serialize-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f protobuf-serialize-action-binding.yaml
```

### 46.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```

kamel bind --name protobuf-serialize-action-binding timer-source?
message='{"first": "John", "last": "Doe"}' --step json-deserialize-action --step protobuf-serialize-action -p
step-1.schema='message Person { required string first = 1; required string last = 2; }'
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic

```

This command creates the KameletBinding in the current namespace on the cluster.

## 46.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//protobuf-serialize-action.kamelet.yaml>



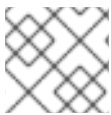
## CHAPTER 47. REGEX ROUTER ACTION

Update the destination using the configured regular expression and replacement string

### 47.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **regex-router-action** Kamelet:

Property	Name	Description	Type	Default	Example
<b>regex *</b>	Regex	Regular Expression for destination	string		
<b>replacement *</b>	Replacement	Replacement when matching	string		



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 47.2. DEPENDENCIES

At runtime, the **regex-router-action** Kamelet relies upon the presence of the following dependencies:

- github:openshift-integration.kamelet-catalog:camel-kamelets-utils:kamelet-catalog-1.6-SNAPSHOT
- camel:kamelet
- camel:core

### 47.3. USAGE

This section describes how you can use the **regex-router-action**.

#### 47.3.1. Knative Action

You can use the **regex-router-action** Kamelet as an intermediate step in a Knative binding.

##### regex-router-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: regex-router-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
```

```

properties:
  message: "Hello"
steps:
- ref:
  kind: Kamelet
  apiVersion: camel.apache.org/v1alpha1
  name: regex-router-action
  properties:
    regex: "The Regex"
    replacement: "The Replacement"
sink:
  ref:
  kind: Channel
  apiVersion: messaging.knative.dev/v1
  name: mychannel

```

#### 47.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

#### 47.3.1.2. Procedure for using the cluster CLI

1. Save the **regex-router-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f regex-router-action-binding.yaml
```

#### 47.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step regex-router-action -p "step-0.regex=The Regex" -p "step-0.replacement=The Replacement" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

### 47.3.2. Kafka Action

You can use the **regex-router-action** Kamelet as an intermediate step in a Kafka binding.

#### regex-router-action-binding.yaml

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: regex-router-action-binding
spec:
  source:
    ref:
      kind: Kamelet

```

```

  apiVersion: camel.apache.org/v1alpha1
  name: timer-source
  properties:
    message: "Hello"
  steps:
  - ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: regex-router-action
  properties:
    regex: "The Regex"
    replacement: "The Replacement"
  sink:
  ref:
    kind: KafkaTopic
    apiVersion: kafka.strimzi.io/v1beta1
    name: my-topic

```

### 47.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 47.3.2.2. Procedure for using the cluster CLI

1. Save the **regex-router-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f regex-router-action-binding.yaml
```

### 47.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step regex-router-action -p "step-0.regex=The Regex" -p "step-0.replacement=The Replacement" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 47.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//regex-router-action.kamelet.yaml>

## CHAPTER 48. REPLACE FIELD ACTION

Replace field with a different key in the message in transit.

- The required parameter 'renames' is a comma-separated list of colon-delimited renaming pairs like for example 'foo:bar,abc:xyz' and it represents the field rename mappings.
- The optional parameter 'enabled' represents the fields to include. If specified, only the named fields will be included in the resulting message.
- The optional parameter 'disabled' represents the fields to exclude. If specified, the listed fields will be excluded from the resulting message. This takes precedence over the 'enabled' parameter.
- The default value of 'enabled' parameter is 'all', so all the fields of the payload will be included.
- The default value of 'disabled' parameter is 'none', so no fields of the payload will be excluded.

### 48.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **replace-field-action** Kamelet:

Property	Name	Description	Type	Default	Example
<b>renames*</b>	Renames	Comma separated list of field with new value to be renamed	string		<b>"foo:bar,c1:c2"</b>
<b>disabled</b>	Disabled	Comma separated list of fields to be disabled	string	"none"	
<b>enabled</b>	Enabled	Comma separated list of fields to be enabled	string	"all"	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 48.2. DEPENDENCIES

At runtime, the **replace-field-action** Kamelet relies upon the presence of the following dependencies:

- github:openshift-integration.kamelet-catalog:camel-kamelets-utils:kamelet-catalog-1.6-SNAPSHOT
- camel:core
- camel:jackson
- camel:kamelet

## 48.3. USAGE

This section describes how you can use the **replace-field-action**.

### 48.3.1. Knative Action

You can use the **replace-field-action** Kamelet as an intermediate step in a Knative binding.

#### replace-field-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: replace-field-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: replace-field-action
    properties:
      renames: "foo:bar,c1:c2"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

#### 48.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

#### 48.3.1.2. Procedure for using the cluster CLI

1. Save the **replace-field-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f replace-field-action-binding.yaml
```

#### 48.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step replace-field-action -p "step-0.renames=foo:bar,c1:c2" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

### 48.3.2. Kafka Action

You can use the **replace-field-action** Kamelet as an intermediate step in a Kafka binding.

#### replace-field-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: replace-field-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: replace-field-action
      properties:
        renames: "foo:bar,c1:c2"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

#### 48.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

#### 48.3.2.2. Procedure for using the cluster CLI

1. Save the **replace-field-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f replace-field-action-binding.yaml
```

#### 48.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step replace-field-action -p "step-0.renames=foo:bar,c1:c2" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 48.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//replace-field-action.kamelet.yaml>

## CHAPTER 49. SALESFORCE SOURCE

Receive updates from Salesforce.

### 49.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **salesforce-source** Kamelet:

Property	Name	Description	Type	Default	Example
<b>clientId *</b>	Consumer Key	The Salesforce application consumer key	string		
<b>clientSecret *</b>	Consumer Secret	The Salesforce application consumer secret	string		
<b>password *</b>	Password	The Salesforce user password	string		
<b>query *</b>	Query	The query to execute on Salesforce	string		<b>"SELECT Id, Name, Email, Phone FROM Contact"</b>
<b>topicName *</b>	Topic Name	The name of the topic/channel to use	string		<b>"ContactTopic"</b>
<b>userName *</b>	Username	The Salesforce username	string		
loginUrl	Login URL	The Salesforce instance login URL	string	<b>"https://login.salesforce.com"</b>	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 49.2. DEPENDENCIES

At runtime, the **salesforce-source** Kamelet relies upon the presence of the following dependencies:

- camel:jackson
- camel:salesforce
- mvn:org.apache.camel.k:camel-k-kamelet-reify
- camel:kamelet



## 49.3. USAGE

This section describes how you can use the **salesforce-source**.

### 49.3.1. Knative Source

You can use the **salesforce-source** Kamelet as a Knative source by binding it to a Knative object.

#### salesforce-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: salesforce-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: salesforce-source
    properties:
      clientId: "The Consumer Key"
      clientSecret: "The Consumer Secret"
      password: "The Password"
      query: "SELECT Id, Name, Email, Phone FROM Contact"
      topicName: "ContactTopic"
      userName: "The Username"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

#### 49.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

#### 49.3.1.2. Procedure for using the cluster CLI

1. Save the **salesforce-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f salesforce-source-binding.yaml
```

#### 49.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind salesforce-source -p "source.clientId=The Consumer Key" -p "source.clientSecret=The Consumer Secret" -p "source.password=The Password" -p "source.query=SELECT Id, Name, Email,
```

```
Phone FROM Contact" -p "source.topicName=ContactTopic" -p "source.userName=The Username"
channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

### 49.3.2. Kafka Source

You can use the **salesforce-source** Kamelet as a Kafka source by binding it to a Kafka topic.

#### salesforce-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: salesforce-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: salesforce-source
    properties:
      clientId: "The Consumer Key"
      clientSecret: "The Consumer Secret"
      password: "The Password"
      query: "SELECT Id, Name, Email, Phone FROM Contact"
      topicName: "ContactTopic"
      userName: "The Username"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

#### 49.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

#### 49.3.2.2. Procedure for using the cluster CLI

1. Save the **salesforce-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f salesforce-source-binding.yaml
```

#### 49.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind salesforce-source -p "source.clientId=The Consumer Key" -p "source.clientSecret=The Consumer Secret" -p "source.password=The Password" -p "source.query=SELECT Id, Name, Email, Phone FROM Contact" -p "source.topicName=ContactTopic" -p "source.userName=The Username" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 49.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//salesforce-source.kamelet.yaml>

## CHAPTER 50. SALESFORCE CREATE SINK

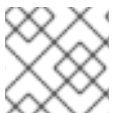
Creates an object in Salesforce. The body of the message must contain the JSON of the salesforce object.

Example body: { "Phone": "555", "Name": "Antonia", "LastName": "Garcia" }

### 50.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **salesforce-create-sink** Kamelet:

Property	Name	Description	Type	Default	Example
<b>clientId *</b>	Consumer Key	The Salesforce application consumer key	string		
<b>clientSecret *</b>	Consumer Secret	The Salesforce application consumer secret	string		
<b>password *</b>	Password	The Salesforce user password	string		
<b>userName *</b>	Username	The Salesforce username	string		
loginUrl	Login URL	The Salesforce instance login URL	string	<b>"https://login.salesforce.com"</b>	
sObjectName	Object Name	Type of the object	string		<b>"Contact"</b>



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 50.2. DEPENDENCIES

At runtime, the **salesforce-create-sink** Kamelet relies upon the presence of the following dependencies:

- camel:salesforce
- camel:kamelet

### 50.3. USAGE

This section describes how you can use the **salesforce-create-sink**.

### 50.3.1. Knative Sink

You can use the **salesforce-create-sink** Kamelet as a Knative sink by binding it to a Knative object.

#### salesforce-create-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: salesforce-create-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: salesforce-create-sink
  properties:
    clientId: "The Consumer Key"
    clientSecret: "The Consumer Secret"
    password: "The Password"
    userName: "The Username"
```

#### 50.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

#### 50.3.1.2. Procedure for using the cluster CLI

1. Save the **salesforce-create-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f salesforce-create-sink-binding.yaml
```

#### 50.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel salesforce-create-sink -p "sink.clientId=The Consumer Key" -p "sink.clientSecret=The Consumer Secret" -p "sink.password=The Password" -p "sink.userName=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 50.3.2. Kafka Sink

You can use the **salesforce-create-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

### salesforce-create-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: salesforce-create-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: salesforce-create-sink
  properties:
    clientId: "The Consumer Key"
    clientSecret: "The Consumer Secret"
    password: "The Password"
    userName: "The Username"
```

### 50.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 50.3.2.2. Procedure for using the cluster CLI

1. Save the **salesforce-create-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f salesforce-create-sink-binding.yaml
```

### 50.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic salesforce-create-sink -p "sink.clientId=The Consumer Key" -p "sink.clientSecret=The Consumer Secret" -p "sink.password=The Password" -p "sink.userName=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 50.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//salesforce-create-sink.kamelet.yaml>

## CHAPTER 51. SALESFORCE DELETE SINK

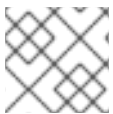
Removes an object from Salesforce. The body received must be a JSON containing two keys: `sObjectId` and `sObjectName`.

Example body: `{ "sObjectId": "XXXXX0", "sObjectName": "Contact" }`

### 51.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **salesforce-delete-sink** Kamelet:

Property	Name	Description	Type	Default	Example
<code>clientId *</code>	Consumer Key	The Salesforce application consumer key	string		
<code>clientSecret *</code>	Consumer Secret	The Salesforce application consumer secret	string		
<code>password *</code>	Password	The Salesforce user password	string		
<code>userName *</code>	Username	The Salesforce username	string		
<code>loginUrl</code>	Login URL	The Salesforce instance login URL	string	<b>"https://login.salesforce.com"</b>	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 51.2. DEPENDENCIES

At runtime, the **salesforce-delete-sink** Kamelet relies upon the presence of the following dependencies:

- `camel:salesforce`
- `camel:kamelet`
- `camel:core`
- `camel:jsonpath`

### 51.3. USAGE



This section describes how you can use the **salesforce-delete-sink**.

### 51.3.1. Knative Sink

You can use the **salesforce-delete-sink** Kamelet as a Knative sink by binding it to a Knative object.

#### salesforce-delete-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: salesforce-delete-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: salesforce-delete-sink
  properties:
    clientId: "The Consumer Key"
    clientSecret: "The Consumer Secret"
    password: "The Password"
    userName: "The Username"
```

#### 51.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

#### 51.3.1.2. Procedure for using the cluster CLI

1. Save the **salesforce-delete-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f salesforce-delete-sink-binding.yaml
```

#### 51.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel salesforce-delete-sink -p "sink.clientId=The Consumer Key" -p "sink.clientSecret=The Consumer Secret" -p "sink.password=The Password" -p "sink.userName=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

### 51.3.2. Kafka Sink

You can use the **salesforce-delete-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

#### salesforce-delete-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: salesforce-delete-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: salesforce-delete-sink
  properties:
    clientId: "The Consumer Key"
    clientSecret: "The Consumer Secret"
    password: "The Password"
    userName: "The Username"
```

#### 51.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

#### 51.3.2.2. Procedure for using the cluster CLI

1. Save the **salesforce-delete-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f salesforce-delete-sink-binding.yaml
```

#### 51.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic salesforce-delete-sink -p "sink.clientId=The Consumer Key" -p "sink.clientSecret=The Consumer Secret" -p "sink.password=The Password" -p "sink.userName=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 51.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//salesforce-delete-sink.kamelet.yaml>

## CHAPTER 52. SALESFORCE UPDATE SINK

Updates an object in Salesforce. The body received must contain a JSON key-value pair for each property to update and `sObjectName` and `sObjectId` must be provided as parameters.

Example of key-value pair: `{ "Phone": "1234567890", "Name": "Antonia" }`

### 52.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **salesforce-update-sink** Kamelet:

Property	Name	Description	Type	Default	Example
<code>clientId *</code>	Consumer Key	The Salesforce application consumer key	string		
<code>clientSecret *</code>	Consumer Secret	The Salesforce application consumer secret	string		
<code>password *</code>	Password	The Salesforce user password	string		
<code>sObjectId *</code>	Object Id	Id of the object. Only required if using key-value pair.	string		
<code>sObjectName *</code>	Object Name	Type of the object. Only required if using key-value pair.	string		<b>"Contact"</b>
<code>userName *</code>	Username	The Salesforce username	string		
<code>loginUrl</code>	Login URL	The Salesforce instance login URL	string	<b>"https://login.salesforce.com"</b>	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 52.2. DEPENDENCIES

At runtime, the **salesforce-update-sink** Kamelet relies upon the presence of the following dependencies:

- `camel:salesforce`

- camel:kamelet

## 52.3. USAGE

This section describes how you can use the **salesforce-update-sink**.

### 52.3.1. Knative Sink

You can use the **salesforce-update-sink** Kamelet as a Knative sink by binding it to a Knative object.

#### salesforce-update-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: salesforce-update-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: salesforce-update-sink
  properties:
    clientId: "The Consumer Key"
    clientSecret: "The Consumer Secret"
    password: "The Password"
    sObjectId: "The Object Id"
    sObjectName: "Contact"
    userName: "The Username"
```

#### 52.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

#### 52.3.1.2. Procedure for using the cluster CLI

1. Save the **salesforce-update-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f salesforce-update-sink-binding.yaml
```

#### 52.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel salesforce-update-sink -p "sink.clientId=The Consumer Key" -p
"sink.clientSecret=The Consumer Secret" -p "sink.password=The Password" -p "sink.sObjectId=The
Object Id" -p "sink.sObjectName=Contact" -p "sink.userName=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

### 52.3.2. Kafka Sink

You can use the **salesforce-update-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

#### salesforce-update-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: salesforce-update-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: salesforce-update-sink
  properties:
    clientId: "The Consumer Key"
    clientSecret: "The Consumer Secret"
    password: "The Password"
    sObjectId: "The Object Id"
    sObjectName: "Contact"
    userName: "The Username"
```

#### 52.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

#### 52.3.2.2. Procedure for using the cluster CLI

1. Save the **salesforce-update-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f salesforce-update-sink-binding.yaml
```

#### 52.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic salesforce-update-sink -p "sink.clientId=The Consumer Key" -p "sink.clientSecret=The Consumer Secret" -p "sink.password=The Password" -p "sink.sObjectId=The Object Id" -p "sink.sObjectName=Contact" -p "sink.userName=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 52.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//salesforce-update-sink.kamelet.yaml>

## CHAPTER 53. SFTP SINK

Send data to an SFTP Server.

The Kamelet expects the following headers to be set:

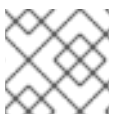
- **file / ce-file**: as the file name to upload

If the header won't be set the exchange ID will be used as file name.

### 53.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **sftp-sink** Kamelet:

Property	Name	Description	Type	Default	Example
<b>connection Host *</b>	Connection Host	Hostname of the FTP server	string		
<b>connection Port *</b>	Connection Port	Port of the FTP server	string	<b>22</b>	
<b>directoryName *</b>	Directory Name	The starting directory	string		
<b>password *</b>	Password	The password to access the FTP server	string		
<b>username *</b>	Username	The username to access the FTP server	string		
fileExist	File Existence	How to behave in case of file already existent. There are 4 enums and the value can be one of Override, Append, Fail or Ignore	string	<b>"Override"</b>	
passiveMode	Passive Mode	Sets passive mode connection	boolean	<b>false</b>	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 53.2. DEPENDENCIES

At runtime, the **sftp-sink** Kamelet relies upon the presence of the following dependencies:



- camel:ftp
- camel:core
- camel:kamelet

## 53.3. USAGE

This section describes how you can use the **sftp-sink**.

### 53.3.1. Knative Sink

You can use the **sftp-sink** Kamelet as a Knative sink by binding it to a Knative object.

#### sftp-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: sftp-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: sftp-sink
  properties:
    connectionHost: "The Connection Host"
    directoryName: "The Directory Name"
    password: "The Password"
    username: "The Username"
```

#### 53.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

#### 53.3.1.2. Procedure for using the cluster CLI

1. Save the **sftp-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f sftp-sink-binding.yaml
```

#### 53.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel sftp-sink -p "sink.connectionHost=The Connection Host" -p "sink.directoryName=The Directory Name" -p "sink.password=The Password" -p "sink.username=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

### 53.3.2. Kafka Sink

You can use the **sftp-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

#### sftp-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: sftp-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: sftp-sink
  properties:
    connectionHost: "The Connection Host"
    directoryName: "The Directory Name"
    password: "The Password"
    username: "The Username"
```

#### 53.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

#### 53.3.2.2. Procedure for using the cluster CLI

1. Save the **sftp-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f sftp-sink-binding.yaml
```

#### 53.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic sftp-sink -p "sink.connectionHost=The  
Connection Host" -p "sink.directoryName=The Directory Name" -p "sink.password=The Password" -p  
"sink.username=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 53.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//sftp-sink.kamelet.yaml>

## CHAPTER 54. SFTP SOURCE

Receive data from an SFTP Server.

### 54.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **sftp-source** Kamelet:

Property	Name	Description	Type	Default	Example
<b>connection Host *</b>	Connection Host	Hostname of the SFTP server	string		
<b>connection Port *</b>	Connection Port	Port of the FTP server	string	<b>22</b>	
<b>directoryName *</b>	Directory Name	The starting directory	string		
<b>password *</b>	Password	The password to access the SFTP server	string		
<b>username *</b>	Username	The username to access the SFTP server	string		
idempotent	Idempotency	Skip already processed files.	boolean	<b>true</b>	
passiveMode	Passive Mode	Sets passive mode connection	boolean	<b>false</b>	
recursive	Recursive	If a directory, will look for files in all the sub-directories as well.	boolean	<b>false</b>	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 54.2. DEPENDENCIES

At runtime, the **sftp-source** Kamelet relies upon the presence of the following dependencies:

- camel:ftp
- camel:core

- camel:kamelet

## 54.3. USAGE

This section describes how you can use the **sftp-source**.

### 54.3.1. Knative Source

You can use the **sftp-source** Kamelet as a Knative source by binding it to a Knative object.

#### sftp-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: sftp-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: sftp-source
  properties:
    connectionHost: "The Connection Host"
    directoryName: "The Directory Name"
    password: "The Password"
    username: "The Username"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

#### 54.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

#### 54.3.1.2. Procedure for using the cluster CLI

1. Save the **sftp-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f sftp-source-binding.yaml
```

#### 54.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind sftp-source -p "source.connectionHost=The Connection Host" -p
"source.directoryName=The Directory Name" -p "source.password=The Password" -p
"source.username=The Username" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

### 54.3.2. Kafka Source

You can use the **sftp-source** Kamelet as a Kafka source by binding it to a Kafka topic.

#### sftp-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: sftp-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: sftp-source
    properties:
      connectionHost: "The Connection Host"
      directoryName: "The Directory Name"
      password: "The Password"
      username: "The Username"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

#### 54.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

#### 54.3.2.2. Procedure for using the cluster CLI

1. Save the **sftp-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f sftp-source-binding.yaml
```

#### 54.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind sftp-source -p "source.connectionHost=The Connection Host" -p  
"source.directoryName=The Directory Name" -p "source.password=The Password" -p  
"source.username=The Username" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 54.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//sftp-source.kamelet.yaml>

## CHAPTER 55. SLACK SOURCE

Receive messages from a Slack channel.

### 55.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **slack-source** Kamelet:

Property	Name	Description	Type	Default	Example
<b>channel *</b>	Channel	The Slack channel to receive messages from	string		<b>"#myroom"</b>
<b>token *</b>	Token	The token to access Slack. A Slack app is needed. This app needs to have channels:history and channels:read permissions. The Bot User OAuth Access Token is the kind of token needed.	string		



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 55.2. DEPENDENCIES

At runtime, the **slack-source** Kamelet relies upon the presence of the following dependencies:

- camel:kamelet
- camel:slack
- camel:jackson

### 55.3. USAGE

This section describes how you can use the **slack-source**.

#### 55.3.1. Knative Source

You can use the **slack-source** Kamelet as a Knative source by binding it to a Knative object.

##### **slack-source-binding.yaml**

```
apiVersion: camel.apache.org/v1alpha1
```



```

kind: KameletBinding
metadata:
  name: slack-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: slack-source
  properties:
    channel: "#myroom"
    token: "The Token"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel

```

### 55.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

### 55.3.1.2. Procedure for using the cluster CLI

1. Save the **slack-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f slack-source-binding.yaml
```

### 55.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind slack-source -p "source.channel=#myroom" -p "source.token=The Token"
channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

## 55.3.2. Kafka Source

You can use the **slack-source** Kamelet as a Kafka source by binding it to a Kafka topic.

### slack-source-binding.yaml

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: slack-source-binding
spec:
  source:

```

```
ref:
  kind: Kamelet
  apiVersion: camel.apache.org/v1alpha1
  name: slack-source
properties:
  channel: "#myroom"
  token: "The Token"
sink:
  ref:
    kind: KafkaTopic
    apiVersion: kafka.strimzi.io/v1beta1
    name: my-topic
```

### 55.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 55.3.2.2. Procedure for using the cluster CLI

1. Save the **slack-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f slack-source-binding.yaml
```

### 55.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind slack-source -p "source.channel=#myroom" -p "source.token=The Token"
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 55.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//slack-source.kamelet.yaml>

## CHAPTER 56. MICROSOFT SQL SERVER SINK

Send data to a Microsoft SQL Server Database.

This Kamelet expects a JSON as body. The mapping between the JSON fields and parameters is done by key, so if you have the following query:

```
'INSERT INTO accounts (username,city) VALUES (:#username,:#city)'
```

The Kamelet needs to receive as input something like:

```
{ "username": "oscerd", "city": "Rome" }
```

### 56.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **sqlserver-sink** Kamelet:

Property	Name	Description	Type	Default	Example
<b>databaseName *</b>	Database Name	The Database Name we are pointing	string		
<b>password *</b>	Password	The password to use for accessing a secured SQL Server Database	string		
<b>query *</b>	Query	The Query to execute against the SQL Server Database	string		<b>"INSERT INTO accounts (username,city) VALUES (:#username,:#city)"</b>
<b>serverName *</b>	Server Name	Server Name for the data source	string		<b>"localhost"</b>
<b>username *</b>	Username	The username to use for accessing a secured SQL Server Database	string		
serverPort	Server Port	Server Port for the data source	string	<b>1433</b>	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 56.2. DEPENDENCIES

At runtime, the **sqlserver-sink** Kamelet relies upon the presence of the following dependencies:

- camel:jackson
- camel:kamelet
- camel:sql
- mvn:org.apache.commons:commons-dbc2:2.7.0.redhat-00001
- mvn:com.microsoft.sqlserver:mssql-jdbc:9.2.1.jre11

## 56.3. USAGE

This section describes how you can use the **sqlserver-sink**.

### 56.3.1. Knative Sink

You can use the **sqlserver-sink** Kamelet as a Knative sink by binding it to a Knative object.

#### sqlserver-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: sqlserver-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: sqlserver-sink
  properties:
    databaseName: "The Database Name"
    password: "The Password"
    query: "INSERT INTO accounts (username,city) VALUES (:#username,:#city)"
    serverName: "localhost"
    username: "The Username"
```

#### 56.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

#### 56.3.1.2. Procedure for using the cluster CLI

1. Save the **sqlserver-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the sink by using the following command:

```
oc apply -f sqlserver-sink-binding.yaml
```

### 56.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel sqlserver-sink -p "sink.databaseName=The Database Name" -p "sink.password=The Password" -p "sink.query=INSERT INTO accounts (username,city) VALUES (:#username,:#city)" -p "sink.serverName=localhost" -p "sink.username=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 56.3.2. Kafka Sink

You can use the **sqlserver-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

### sqlserver-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: sqlserver-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: sqlserver-sink
  properties:
    databaseName: "The Database Name"
    password: "The Password"
    query: "INSERT INTO accounts (username,city) VALUES (:#username,:#city)"
    serverName: "localhost"
    username: "The Username"
```

#### 56.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

#### 56.3.2.2. Procedure for using the cluster CLI

1. Save the **sqlserver-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the sink by using the following command:

```
oc apply -f sqlserver-sink-binding.yaml
```

### 56.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic sqlserver-sink -p "sink.databaseName=The Database Name" -p "sink.password=The Password" -p "sink.query=INSERT INTO accounts (username,city) VALUES (:#username,:#city)" -p "sink.serverName=localhost" -p "sink.username=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

## 56.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//sqlserver-sink.kamelet.yaml>

## CHAPTER 57. TELEGRAM SOURCE

Receive all messages that people send to your Telegram bot.

To create a bot, contact the @botfather account using the Telegram app.

The source attaches the following headers to the messages:

- **chat-id** / **ce-chatid**: the ID of the chat where the message comes from

### 57.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **telegram-source** Kamelet:

Property	Name	Description	Type	Default	Example
<b>authorizationToken*</b>	Token	The token to access your bot on Telegram. You you can obtain it from the Telegram @botfather.	string		



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 57.2. DEPENDENCIES

At runtime, the **telegram-source** Kamelet relies upon the presence of the following dependencies:

- camel:jackson
- camel:kamelet
- camel:telegram
- camel:core

### 57.3. USAGE

This section describes how you can use the **telegram-source**.

#### 57.3.1. Knative Source

You can use the **telegram-source** Kamelet as a Knative source by binding it to a Knative object.

##### telegram-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
```

```

name: telegram-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: telegram-source
    properties:
      authorizationToken: "The Token"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel

```

### 57.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

### 57.3.1.2. Procedure for using the cluster CLI

1. Save the **telegram-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f telegram-source-binding.yaml
```

### 57.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind telegram-source -p "source.authorizationToken=The Token" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

## 57.3.2. Kafka Source

You can use the **telegram-source** Kamelet as a Kafka source by binding it to a Kafka topic.

### telegram-source-binding.yaml

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: telegram-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: telegram-source

```



```
properties:
  authorizationToken: "The Token"
sink:
  ref:
    kind: KafkaTopic
    apiVersion: kafka.strimzi.io/v1beta1
    name: my-topic
```

### 57.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 57.3.2.2. Procedure for using the cluster CLI

1. Save the **telegram-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f telegram-source-binding.yaml
```

### 57.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind telegram-source -p "source.authorizationToken=The Token"
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 57.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//telegram-source.kamelet.yaml>

## CHAPTER 58. THROTTLE ACTION

The Throttle action allows you to ensure that a specific sink does not get overloaded.

### 58.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **throttle-action** Kamelet:

Property	Name	Description	Type	Default	Example
<b>messages *</b>	Messages Number	The number of messages to send in the time period set	integer		<b>10</b>
<b>timePeriod</b>	Time Period	Sets the time period during which the maximum request count is valid for, in milliseconds	string	<b>"1000"</b>	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 58.2. DEPENDENCIES

At runtime, the **throttle-action** Kamelet relies upon the presence of the following dependencies:

- camel:core
- camel:kamelet

### 58.3. USAGE

This section describes how you can use the **throttle-action**.

#### 58.3.1. Knative Action

You can use the **throttle-action** Kamelet as an intermediate step in a Knative binding.

##### **throttle-action-binding.yaml**

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: throttle-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
```

```

name: timer-source
properties:
  message: "Hello"
steps:
- ref:
  kind: Kamelet
  apiVersion: camel.apache.org/v1alpha1
  name: throttle-action
properties:
  messages: 1
sink:
  ref:
  kind: Channel
  apiVersion: messaging.knative.dev/v1
  name: mychannel

```

### 58.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

### 58.3.1.2. Procedure for using the cluster CLI

1. Save the **throttle-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f throttle-action-binding.yaml
```

### 58.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step throttle-action -p "step-0.messages=10"
channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

## 58.3.2. Kafka Action

You can use the **throttle-action** Kamelet as an intermediate step in a Kafka binding.

### throttle-action-binding.yaml

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: throttle-action-binding
spec:
  source:
    ref:
      kind: Kamelet

```

```
  apiVersion: camel.apache.org/v1alpha1
  name: timer-source
  properties:
    message: "Hello"
  steps:
  - ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: throttle-action
  properties:
    messages: 1
  sink:
  ref:
    kind: KafkaTopic
    apiVersion: kafka.strimzi.io/v1beta1
    name: my-topic
```

### 58.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 58.3.2.2. Procedure for using the cluster CLI

1. Save the **throttle-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f throttle-action-binding.yaml
```

### 58.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step throttle-action -p "step-0.messages=1"
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 58.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//throttle-action.kamelet.yaml>

## CHAPTER 59. TIMER SOURCE

Produces periodic events with a custom payload.

### 59.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **timer-source** Kamelet:

Property	Name	Description	Type	Default	Example
<b>message *</b>	Message	The message to generate	string		<b>"hello world"</b>
contentType	Content Type	The content type of the message being generated	string	<b>"text/plain"</b>	
period	Period	The interval between two events in milliseconds	integer	<b>1000</b>	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 59.2. DEPENDENCIES

At runtime, the **timer-source** Kamelet relies upon the presence of the following dependencies:

- camel:core
- camel:timer
- camel:kamelet

### 59.3. USAGE

This section describes how you can use the **timer-source**.

#### 59.3.1. Knative Source

You can use the **timer-source** Kamelet as a Knative source by binding it to a Knative object.

##### timer-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: timer-source-binding
spec:
  source:
```

```

ref:
  kind: Kamelet
  apiVersion: camel.apache.org/v1alpha1
  name: timer-source
properties:
  message: "hello world"
sink:
  ref:
    kind: Channel
    apiVersion: messaging.knative.dev/v1
    name: mychannel

```

### 59.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

### 59.3.1.2. Procedure for using the cluster CLI

1. Save the **timer-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f timer-source-binding.yaml
```

### 59.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind timer-source -p "source.message=hello world" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

## 59.3.2. Kafka Source

You can use the **timer-source** Kamelet as a Kafka source by binding it to a Kafka topic.

### timer-source-binding.yaml

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: timer-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "hello world"
  sink:

```

```
ref:  
  kind: KafkaTopic  
  apiVersion: kafka.strimzi.io/v1beta1  
  name: my-topic
```

### 59.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 59.3.2.2. Procedure for using the cluster CLI

1. Save the **timer-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f timer-source-binding.yaml
```

### 59.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind timer-source -p "source.message=hello world" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 59.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//timer-source.kamelet.yaml>

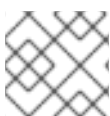
## CHAPTER 60. TIMESTAMP ROUTER ACTION

Update the topic field as a function of the original topic name and the record timestamp.

### 60.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **timestamp-router-action** Kamelet:

Property	Name	Description	Type	Default	Example
timestampFormat	Timestamp Format	Format string for the timestamp that is compatible with <code>java.text.SimpleDateFormat</code> .	string	"yyyyMMd d"	
timestampHeaderName	Timestamp Header Name	The name of the header containing a timestamp	string	"kafka.TIMESTAMP"	
topicFormat	Topic Format	Format string which can contain '[topic]' and '[timestamp]' as placeholders for the topic and timestamp, respectively.	string	"topic-\${timestamp}"	



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 60.2. DEPENDENCIES

At runtime, the **timestamp-router-action** Kamelet relies upon the presence of the following dependencies:

- `github:openshift-integration.kamelet-catalog:camel-kamelets-utils:kamelet-catalog-1.6-SNAPSHOT`
- `camel:kamelet`
- `camel:core`

### 60.3. USAGE

This section describes how you can use the **timestamp-router-action**.

#### 60.3.1. Knative Action



You can use the **timestamp-router-action** Kamelet as an intermediate step in a Knative binding.

### timestamp-router-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: timestamp-router-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timestamp-router-action
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

#### 60.3.1.1. Prerequisite

Make sure you have "Red Hat Integration - Camel K" installed into the OpenShift cluster you're connected to.

#### 60.3.1.2. Procedure for using the cluster CLI

1. Save the **timestamp-router-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f timestamp-router-action-binding.yaml
```

#### 60.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step timestamp-router-action channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

### 60.3.2. Kafka Action

You can use the **timestamp-router-action** Kamelet as an intermediate step in a Kafka binding.

## timestamp-router-action-binding.yaml

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: timestamp-router-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timestamp-router-action
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic

```

### 60.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 60.3.2.2. Procedure for using the cluster CLI

1. Save the **timestamp-router-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f timestamp-router-action-binding.yaml
```

### 60.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step timestamp-router-action
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 60.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//timestamp-router-action.kamelet.yaml>

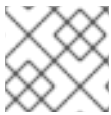
## CHAPTER 61. VALUE TO KEY ACTION

Replace the Kafka record key with a new key formed from a subset of fields in the body

### 61.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **value-to-key-action** Kamelet:

Property	Name	Description	Type	Default	Example
<b>fields *</b>	Fields	Comma separated list of fields to be used to form the new key	string		



#### NOTE

Fields marked with an asterisk (\*) are mandatory.

### 61.2. DEPENDENCIES

At runtime, the **value-to-key-action** Kamelet relies upon the presence of the following dependencies:

- github:openshift-integration.kamelet-catalog:camel-kamelets-utils:kamelet-catalog-1.6-SNAPSHOT
- camel:core
- camel:jackson
- camel:kamelet

### 61.3. USAGE

This section describes how you can use the **value-to-key-action**.

#### 61.3.1. Knative Action

You can use the **value-to-key-action** Kamelet as an intermediate step in a Knative binding.

##### value-to-key-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: value-to-key-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
```

```

properties:
  message: "Hello"
steps:
- ref:
  kind: Kamelet
  apiVersion: camel.apache.org/v1alpha1
  name: value-to-key-action
  properties:
    fields: "The Fields"
sink:
  ref:
  kind: Channel
  apiVersion: messaging.knative.dev/v1
  name: mychannel

```

### 61.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 61.3.1.2. Procedure for using the cluster CLI

1. Save the **value-to-key-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f value-to-key-action-binding.yaml
```

### 61.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step value-to-key-action -p "step-0.fields=The Fields"
channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

## 61.3.2. Kafka Action

You can use the **value-to-key-action** Kamelet as an intermediate step in a Kafka binding.

### value-to-key-action-binding.yaml

```

apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: value-to-key-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1

```

```

    name: timer-source
  properties:
    message: "Hello"
  steps:
  - ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: value-to-key-action
  properties:
    fields: "The Fields"
  sink:
  ref:
    kind: KafkaTopic
    apiVersion: kafka.strimzi.io/v1beta1
    name: my-topic

```

### 61.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

### 61.3.2.2. Procedure for using the cluster CLI

1. Save the **value-to-key-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f value-to-key-action-binding.yaml
```

### 61.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step value-to-key-action -p "step-0.fields=The Fields"
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

## 61.4. KAMELET SOURCE FILE

<https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//value-to-key-action.kamelet.yaml>