Red Hat build of Apache Camel K 1.10

Kamelets Reference

Kamelets Reference

Last Updated: 2023-03-06

Red Hat build of Apache Camel K 1.10 Kamelets Reference

Kamelets Reference

Legal Notice

Abstract

Red Hat build of Apache Camel K Kamelets are reusable route components that hide the complexity of creating data pipelines that connect to external systems.

Table of Contents

PREFACE 21
MAKING OPEN SOURCE MORE INCLUSIVE 21

CHAPTER 1. AWS DYNAMODB SINK 22
1.1. CONFIGURATION OPTIONS 22
1.2. DEPENDENCIES 23
1.3. USAGE 23
1.3.1. Knative Sink 24
1.3.1.1. Prerequisite 24
1.3.1.2. Procedure for using the cluster CLI 24
1.3.1.3. Procedure for using the Kamel CLI 24
1.3.2. Kafka Sink 24
1.3.2.1. Prerequisites 25
1.3.2.2. Procedure for using the cluster CLI 25
1.3.2.3. Procedure for using the Kamel CLI 25
1.4. KAMELET SOURCE FILE 25

CHAPTER 2. AVRO DESERIALIZE ACTION 26
2.1. CONFIGURATION OPTIONS 26
2.2. DEPENDENCIES 26
2.3. USAGE 26
2.3.1. Knative Action 27
2.3.1.1. Prerequisite 27
2.3.1.2. Procedure for using the cluster CLI 27
2.3.1.3. Procedure for using the Kamel CLI 28
2.3.2. Kafka Action 28
2.3.2.1. Prerequisites 29
2.3.2.2. Procedure for using the cluster CLI 29
2.3.2.3. Procedure for using the Kamel CLI 29
2.4. KAMELET SOURCE FILE 29

CHAPTER 3. AVRO SERIALIZE ACTION 30
3.1. CONFIGURATION OPTIONS 30
3.2. DEPENDENCIES 30
3.3. USAGE 30
3.3.1. Knative Action 31
3.3.1.1. Prerequisite 31
3.3.1.2. Procedure for using the cluster CLI 31
3.3.1.3. Procedure for using the Kamel CLI 31
3.3.2. Kafka Action 32
3.3.2.1. Prerequisites 32
3.3.2.2. Procedure for using the cluster CLI 32
3.3.2.3. Procedure for using the Kamel CLI 33
3.4. KAMELET SOURCE FILE 33

CHAPTER 4. AWS KINESIS SINK 34
4.1. CONFIGURATION OPTIONS 34
4.2. DEPENDENCIES 34
4.3. USAGE 34
4.3.1. Knative Sink 35
4.3.1.1. Prerequisite 35
4.3.1.2. Procedure for using the cluster CLI 35

4.3.1.3. Procedure for using the Kamel CLI 35
4.3.2. Kafka Sink 35
4.3.2.1. Prerequisites 36
4.3.2.2. Procedure for using the cluster CLI 36
4.3.2.3. Procedure for using the Kamel CLI 36
4.4. KAMELET SOURCE FILE 36

CHAPTER 5. AWS KINESIS SOURCE 37
5.1. CONFIGURATION OPTIONS 37
5.2. DEPENDENCIES 37
5.3. USAGE 37
5.3.1. Knative Source 37
5.3.1.1. Prerequisite 38
5.3.1.2. Procedure for using the cluster CLI 38
5.3.1.3. Procedure for using the Kamel CLI 38
5.3.2. Kafka Source 38
5.3.2.1. Prerequisites 39
5.3.2.2. Procedure for using the cluster CLI 39
5.3.2.3. Procedure for using the Kamel CLI 39
5.4. KAMELET SOURCE FILE 39

CHAPTER 6. AWS LAMBDA SINK 40
6.1. CONFIGURATION OPTIONS 40
6.2. DEPENDENCIES 40
6.3. USAGE 40
6.3.1. Knative Sink 40
6.3.1.1. Prerequisite 41
6.3.1.2. Procedure for using the cluster CLI 41
6.3.1.3. Procedure for using the Kamel CLI 41
6.3.2. Kafka Sink 41
6.3.2.1. Prerequisites 42
6.3.2.2. Procedure for using the cluster CLI 42
6.3.2.3. Procedure for using the Kamel CLI 42
6.4. KAMELET SOURCE FILE 42

CHAPTER 7. AWS REDSHIFT SINK 43
7.1. CONFIGURATION OPTIONS 43
7.2. DEPENDENCIES 43
7.3. USAGE 44
7.3.1. Knative Sink 44
7.3.1.1. Prerequisite 44
7.3.1.2. Procedure for using the cluster CLI 44
7.3.1.3. Procedure for using the Kamel CLI 45
7.3.2. Kafka Sink 45
7.3.2.1. Prerequisites 45
7.3.2.2. Procedure for using the cluster CLI 45
7.3.2.3. Procedure for using the Kamel CLI 46
7.4. KAMELET SOURCE FILE 46

CHAPTER 8. AWS SNS SINK 47
8.1. CONFIGURATION OPTIONS 47
8.2. DEPENDENCIES 47
8.3. USAGE 47
8.3.1. Knative Sink 47

8.3.1.1. Prerequisite 48
8.3.1.2. Procedure for using the cluster CLI 48
8.3.1.3. Procedure for using the Kamel CLI 48
8.3.2. Kafka Sink 48
8.3.2.1. Prerequisites 49
8.3.2.2. Procedure for using the cluster CLI 49
8.3.2.3. Procedure for using the Kamel CLI 49
8.4. KAMELET SOURCE FILE 49

CHAPTER 9. AWS SQS SINK 50
9.1. CONFIGURATION OPTIONS 50
9.2. DEPENDENCIES 50
9.3. USAGE 50
9.3.1. Knative Sink 50
9.3.1.1. Prerequisite 51
9.3.1.2. Procedure for using the cluster CLI 51
9.3.1.3. Procedure for using the Kamel CLI 51
9.3.2. Kafka Sink 51
9.3.2.1. Prerequisites 52
9.3.2.2. Procedure for using the cluster CLI 52
9.3.2.3. Procedure for using the Kamel CLI 52
9.4. KAMELET SOURCE FILE 52

CHAPTER 10. AWS SQS SOURCE 53
10.1. CONFIGURATION OPTIONS 53
10.2. DEPENDENCIES 53
10.3. USAGE 53
10.3.1. Knative Source 54
10.3.1.1. Prerequisite 54
10.3.1.2. Procedure for using the cluster CLI 54
10.3.1.3. Procedure for using the Kamel CLI 54
10.3.2. Kafka Source 54
10.3.2.1. Prerequisites 55
10.3.2.2. Procedure for using the cluster CLI 55
10.3.2.3. Procedure for using the Kamel CLI 55
10.4. KAMELET SOURCE FILE 55

CHAPTER 11. AWS 2 SIMPLE QUEUE SERVICE FIFO SINK 56
11.1. CONFIGURATION OPTIONS 56
11.2. DEPENDENCIES 56
11.3. USAGE 56
11.3.1. Knative Sink 57
11.3.1.1. Prerequisite 57
11.3.1.2. Procedure for using the cluster CLI 57
11.3.1.3. Procedure for using the Kamel CLI 57
11.3.2. Kafka Sink 57
11.3.2.1. Prerequisites 58
11.3.2.2. Procedure for using the cluster CLI 58
11.3.2.3. Procedure for using the Kamel CLI 58
11.4. KAMELET SOURCE FILE 58

CHAPTER 12. AWS S3 SINK 59
12.1. CONFIGURATION OPTIONS 59
12.2. DEPENDENCIES 59

12.3. USAGE 59
12.3.1. Knative Sink 59
12.3.1.1. Prerequisite 60
12.3.1.2. Procedure for using the cluster CLI 60
12.3.1.3. Procedure for using the Kamel CLI 60
12.3.2. Kafka Sink 60
12.3.2.1. Prerequisites 61
12.3.2.2. Procedure for using the cluster CLI 61
12.3.2.3. Procedure for using the Kamel CLI 61
12.4. KAMELET SOURCE FILE 61

CHAPTER 13. AWS S3 SOURCE 62
13.1. CONFIGURATION OPTIONS 62
13.2. DEPENDENCIES 62
13.3. USAGE 62
13.3.1. Knative Source 62
13.3.1.1. Prerequisite 63
13.3.1.2. Procedure for using the cluster CLI 63
13.3.1.3. Procedure for using the Kamel CLI 63
13.3.2. Kafka Source 63
13.3.2.1. Prerequisites 64
13.3.2.2. Procedure for using the cluster CLI 64
13.3.2.3. Procedure for using the Kamel CLI 64
13.4. KAMELET SOURCE FILE 64

CHAPTER 14. AWS S3 STREAMING UPLOAD SINK 65
14.1. CONFIGURATION OPTIONS 65
14.2. DEPENDENCIES 66
14.3. USAGE 66
14.3.1. Knative Sink 66
14.3.1.1. Prerequisite 67
14.3.1.2. Procedure for using the cluster CLI 67
14.3.1.3. Procedure for using the Kamel CLI 67
14.3.2. Kafka Sink 67
14.3.2.1. Prerequisites 68
14.3.2.2. Procedure for using the cluster CLI 68
14.3.2.3. Procedure for using the Kamel CLI 68
14.4. KAMELET SOURCE FILE 68

CHAPTER 15. AZURE STORAGE BLOB SINK 69
15.1. CONFIGURATION OPTIONS 69
15.2. DEPENDENCIES 70
15.3. USAGE 70
15.3.1. Knative Sink 70
15.3.1.1. Prerequisite 70
15.3.1.2. Procedure for using the cluster CLI 70
15.3.1.3. Procedure for using the Kamel CLI 71
15.3.2. Kafka Sink 71
15.3.2.1. Prerequisites 71
15.3.2.2. Procedure for using the cluster CLI 71
15.3.2.3. Procedure for using the Kamel CLI 72
15.4. KAMELET SOURCE FILE 72

CHAPTER 16. AZURE STORAGE BLOB SOURCE 73

16.1. CONFIGURATION OPTIONS 73
16.2. DEPENDENCIES 74
16.3. USAGE 74
16.3.1. Knative Source 74
16.3.1.1. Prerequisite 74
16.3.1.2. Procedure for using the cluster CLI 75
16.3.1.3. Procedure for using the Kamel CLI 75
16.3.2. Kafka Source 75
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 76
16.4. KAMELET SOURCE FILE 76

CHAPTER 17. AZURE STORAGE QUEUE SINK 77
17.1. CONFIGURATION OPTIONS 77
17.2. DEPENDENCIES 77
17.3. USAGE 78
17.3.1. Knative Sink 78
17.3.1.1. Prerequisite 78
17.3.1.2. Procedure for using the cluster CLI 78
17.3.1.3. Procedure for using the Kamel CLI 78
17.3.2. Kafka Sink 79
17.3.2.1. Prerequisites 79
17.3.2.2. Procedure for using the cluster CLI 79
17.3.2.3. Procedure for using the Kamel CLI 79
17.4. KAMELET SOURCE FILE 80

CHAPTER 18. AZURE STORAGE QUEUE SOURCE 81
18.1. CONFIGURATION OPTIONS 81
18.2. DEPENDENCIES 82
18.3. USAGE 82
18.3.1. Knative Source 82
18.3.1.1. Prerequisite 82
18.3.1.2. Procedure for using the cluster CLI 82
18.3.1.3. Procedure for using the Kamel CLI 83
18.3.2. Kafka Source 83
18.3.2.1. Prerequisites 83
18.3.2.2. Procedure for using the cluster CLI 83
18.3.2.3. Procedure for using the Kamel CLI 83
18.4. KAMELET SOURCE FILE 84

CHAPTER 19. CASSANDRA SINK 85
19.1. CONFIGURATION OPTIONS 85
19.2. DEPENDENCIES 86
19.3. USAGE 86
19.3.1. Knative Sink 86
19.3.1.1. Prerequisite 86
19.3.1.2. Procedure for using the cluster CLI 86
19.3.1.3. Procedure for using the Kamel CLI 87
19.3.2. Kafka Sink 87
19.3.2.1. Prerequisites 87
19.3.2.2. Procedure for using the cluster CLI 87
19.3.2.3. Procedure for using the Kamel CLI 88
19.4. KAMELET SOURCE FILE 88

CHAPTER 20. CASSANDRA SOURCE 89
20.1. CONFIGURATION OPTIONS 89
20.2. DEPENDENCIES 90
20.3. USAGE 90
20.3.1. Knative Source 90
20.3.1.1. Prerequisite 91
20.3.1.2. Procedure for using the cluster CLI 91
20.3.1.3. Procedure for using the Kamel CLI 91
20.3.2. Kafka Source 91
20.3.2.1. Prerequisites 92
20.3.2.2. Procedure for using the cluster CLI 92
20.3.2.3. Procedure for using the Kamel CLI 92
20.4. KAMELET SOURCE FILE 92

CHAPTER 21. CEPH SINK 93
21.1. CONFIGURATION OPTIONS 93
21.2. DEPENDENCIES 93
21.3. USAGE 94
21.3.1. Knative Sink 94
21.3.1.1. Prerequisite 94
21.3.1.2. Procedure for using the cluster CLI 94
21.3.1.3. Procedure for using the Kamel CLI 94
21.3.2. Kafka Sink 95
21.3.2.1. Prerequisites 95
21.3.2.2. Procedure for using the cluster CLI 95
21.3.2.3. Procedure for using the Kamel CLI 95
21.4. KAMELET SOURCE FILE 96

CHAPTER 22. CEPH SOURCE 97
22.1. CONFIGURATION OPTIONS 97
22.2. DEPENDENCIES 98
22.3. USAGE 98
22.3.1. Knative Source 98
22.3.1.1. Prerequisite 99
22.3.1.2. Procedure for using the cluster CLI 99
22.3.1.3. Procedure for using the Kamel CLI 99
22.3.2. Kafka Source 99
22.3.2.1. Prerequisites 100
22.3.2.2. Procedure for using the cluster CLI 100
22.3.2.3. Procedure for using the Kamel CLI 100
22.4. KAMELET SOURCE FILE 100

CHAPTER 23. EXTRACT FIELD ACTION 101
23.1. CONFIGURATION OPTIONS 101
23.2. DEPENDENCIES 101
23.3. USAGE 101
23.3.1. Knative Action 101
23.3.1.1. Prerequisite 102
23.3.1.2. Procedure for using the cluster CLI 102
23.3.1.3. Procedure for using the Kamel CLI 102
23.3.2. Kafka Action 102
23.3.2.1. Prerequisites 103
23.3.2.2. Procedure for using the cluster CLI 103
23.3.2.3. Procedure for using the Kamel CLI 103

23.4. KAMELET SOURCE FILE 103

CHAPTER 24. FTP SINK 104
24.1. CONFIGURATION OPTIONS 104
24.2. DEPENDENCIES 104
24.3. USAGE 105
24.3.1. Knative Sink 105
24.3.1.1. Prerequisite 105
24.3.1.2. Procedure for using the cluster CLI 105
24.3.1.3. Procedure for using the Kamel CLI 105
24.3.2. Kafka Sink 106
24.3.2.1. Prerequisites 106
24.3.2.2. Procedure for using the cluster CLI 106
24.3.2.3. Procedure for using the Kamel CLI 106
24.4. KAMELET SOURCE FILE 107

CHAPTER 25. FTP SOURCE 108
25.1. CONFIGURATION OPTIONS 108
25.2. DEPENDENCIES 108
25.3. USAGE 109
25.3.1. Knative Source 109
25.3.1.1. Prerequisite 109
25.3.1.2. Procedure for using the cluster CLI 109
25.3.1.3. Procedure for using the Kamel CLI 109
25.3.2. Kafka Source 110
25.3.2.1. Prerequisites 110
25.3.2.2. Procedure for using the cluster CLI 110
25.3.2.3. Procedure for using the Kamel CLI 110
25.4. KAMELET SOURCE FILE 111

CHAPTER 26. HAS HEADER FILTER ACTION 112
26.1. CONFIGURATION OPTIONS 112
26.2. DEPENDENCIES 112
26.3. USAGE 112
26.3.1. Knative Action 112
26.3.1.1. Prerequisite 113
26.3.1.2. Procedure for using the cluster CLI 113
26.3.1.3. Procedure for using the Kamel CLI 113
26.3.2. Kafka Action 113
26.3.2.1. Prerequisites 114
26.3.2.2. Procedure for using the cluster CLI 114
26.3.2.3. Procedure for using the Kamel CLI 114
26.4. KAMELET SOURCE FILE 115

CHAPTER 27. HOIST FIELD ACTION 116
27.1. CONFIGURATION OPTIONS 116
27.2. DEPENDENCIES 116
27.3. USAGE 116
27.3.1. Knative Action 116
27.3.1.1. Prerequisite 117
27.3.1.2. Procedure for using the cluster CLI 117
27.3.1.3. Procedure for using the Kamel CLI 117
27.3.2. Kafka Action 117
27.3.2.1. Prerequisites 118

27.3.2.2. Procedure for using the cluster CLI 118
27.3.2.3. Procedure for using the Kamel CLI 118
27.4. KAMELET SOURCE FILE 118

CHAPTER 28. HTTP SINK 119
28.1. CONFIGURATION OPTIONS 119
28.2. DEPENDENCIES 119
28.3. USAGE 119
28.3.1. Knative Sink 119
28.3.1.1. Prerequisite 120
28.3.1.2. Procedure for using the cluster CLI 120
28.3.1.3. Procedure for using the Kamel CLI 120
28.3.2. Kafka Sink 120
28.3.2.1. Prerequisites 121
28.3.2.2. Procedure for using the cluster CLI 121
28.3.2.3. Procedure for using the Kamel CLI 121
28.4. KAMELET SOURCE FILE 121

CHAPTER 29. INSERT FIELD ACTION 122
29.1. CONFIGURATION OPTIONS 122
29.2. DEPENDENCIES 122
29.3. USAGE 122
29.3.1. Knative Action 122
29.3.1.1. Prerequisite 123
29.3.1.2. Procedure for using the cluster CLI 123
29.3.1.3. Procedure for using the Kamel CLI 123
29.3.2. Kafka Action 123
29.3.2.1. Prerequisites 124
29.3.2.2. Procedure for using the cluster CLI 124
29.3.2.3. Procedure for using the Kamel CLI 124
29.4. KAMELET SOURCE FILE 125

CHAPTER 30. INSERT HEADER ACTION 126
30.1. CONFIGURATION OPTIONS 126
30.2. DEPENDENCIES 126
30.3. USAGE 126
30.3.1. Knative Action 126
30.3.1.1. Prerequisite 127
30.3.1.2. Procedure for using the cluster CLI 127
30.3.1.3. Procedure for using the Kamel CLI 127
30.3.2. Kafka Action 127
30.3.2.1. Prerequisites 128
30.3.2.2. Procedure for using the cluster CLI 128
30.3.2.3. Procedure for using the Kamel CLI 128
30.4. KAMELET SOURCE FILE 128

CHAPTER 31. IS TOMBSTONE FILTER ACTION 129
31.1. CONFIGURATION OPTIONS 129
31.2. DEPENDENCIES 129
31.3. USAGE 129
31.3.1. Knative Action 129
31.3.1.1. Prerequisite 129
31.3.1.2. Procedure for using the cluster CLI 130
31.3.1.3. Procedure for using the Kamel CLI 130

31.3.2. Kafka Action 130
31.3.2.1. Prerequisites 130
31.3.2.2. Procedure for using the cluster CLI 131
31.3.2.3. Procedure for using the Kamel CLI 131
31.4. KAMELET SOURCE FILE 131

CHAPTER 32. JIRA ADD COMMENT SINK 132
32.1. CONFIGURATION OPTIONS 132
32.2. DEPENDENCIES 132
32.3. USAGE 132
32.3.1. Knative Sink 132
32.3.1.1. Prerequisite 133
32.3.1.2. Procedure for using the cluster CLI 133
32.3.1.3. Procedure for using the Kamel CLI 133
32.3.2. Kafka Sink 134
32.3.2.1. Prerequisites 134
32.3.2.2. Procedure for using the cluster CLI 134
32.3.2.3. Procedure for using the Kamel CLI 134
32.4. KAMELET SOURCE FILE 135

CHAPTER 33. JIRA ADD ISSUE SINK 136
33.1. CONFIGURATION OPTIONS 136
33.2. DEPENDENCIES 136
33.3. USAGE 137
33.3.1. Knative Sink 137
33.3.1.1. Prerequisite 138
33.3.1.2. Procedure for using the cluster CLI 138
33.3.1.3. Procedure for using the Kamel CLI 138
33.3.2. Kafka Sink 138
33.3.2.1. Prerequisites 139
33.3.2.2. Procedure for using the cluster CLI 139
33.3.2.3. Procedure for using the Kamel CLI 140
33.4. KAMELET SOURCE FILE 140

CHAPTER 34. JIRA TRANSITION ISSUE SINK 141
34.1. CONFIGURATION OPTIONS 141
34.2. DEPENDENCIES 141
34.3. USAGE 141
34.3.1. Knative Sink 142
34.3.1.1. Prerequisite 142
34.3.1.2. Procedure for using the cluster CLI 142
34.3.1.3. Procedure for using the Kamel CLI 143
34.3.2. Kafka Sink 143
34.3.2.1. Prerequisites 144
34.3.2.2. Procedure for using the cluster CLI 144
34.3.2.3. Procedure for using the Kamel CLI 144
34.4. KAMELET SOURCE FILE 144

CHAPTER 35. JIRA UPDATE ISSUE SINK 145
35.1. CONFIGURATION OPTIONS 145
35.2. DEPENDENCIES 145
35.3. USAGE 146
35.3.1. Knative Sink 146
35.3.1.1. Prerequisite 147

35.3.1.2. Procedure for using the cluster CLI 147
35.3.1.3. Procedure for using the Kamel CLI 147
35.3.2. Kafka Sink 147
35.3.2.1. Prerequisites 148
35.3.2.2. Procedure for using the cluster CLI 148
35.3.2.3. Procedure for using the Kamel CLI 149
35.4. KAMELET SOURCE FILE 149

CHAPTER 36. JIRA SOURCE 150
36.1. CONFIGURATION OPTIONS 150
36.2. DEPENDENCIES 150
36.3. USAGE 150
36.3.1. Knative Source 150
36.3.1.1. Prerequisite 151
36.3.1.2. Procedure for using the cluster CLI 151
36.3.1.3. Procedure for using the Kamel CLI 151
36.3.2. Kafka Source 151
36.3.2.1. Prerequisites 152
36.3.2.2. Procedure for using the cluster CLI 152
36.3.2.3. Procedure for using the Kamel CLI 152
36.4. KAMELET SOURCE FILE 152

CHAPTER 37. JMS - AMQP 1.0 KAMELET SINK 153
37.1. CONFIGURATION OPTIONS 153
37.2. DEPENDENCIES 153
37.3. USAGE 153
37.3.1. Knative Sink 153
37.3.1.1. Prerequisite 154
37.3.1.2. Procedure for using the cluster CLI 154
37.3.1.3. Procedure for using the Kamel CLI 154
37.3.2. Kafka Sink 154
37.3.2.1. Prerequisites 155
37.3.2.2. Procedure for using the cluster CLI 155
37.3.2.3. Procedure for using the Kamel CLI 155
37.4. KAMELET SOURCE FILE 155

CHAPTER 38. JMS - AMQP 1.0 KAMELET SOURCE 156
38.1. CONFIGURATION OPTIONS 156
38.2. DEPENDENCIES 156
38.3. USAGE 156
38.3.1. Knative Source 156
38.3.1.1. Prerequisite 157
38.3.1.2. Procedure for using the cluster CLI 157
38.3.1.3. Procedure for using the Kamel CLI 157
38.3.2. Kafka Source 157
38.3.2.1. Prerequisites 158
38.3.2.2. Procedure for using the cluster CLI 158
38.3.2.3. Procedure for using the Kamel CLI 158
38.4. KAMELET SOURCE FILE 158

CHAPTER 39. JMS - IBM MQ KAMELET SINK 159
39.1. CONFIGURATION OPTIONS 159
39.2. DEPENDENCIES 159
39.3. USAGE 160

39.3.1. Knative Sink 160
39.3.1.1. Prerequisite 160
39.3.1.2. Procedure for using the cluster CLI 160
39.3.1.3. Procedure for using the Kamel CLI 161
39.3.2. Kafka Sink 161
39.3.2.1. Prerequisites 161
39.3.2.2. Procedure for using the cluster CLI 161
39.3.2.3. Procedure for using the Kamel CLI 162
39.4. KAMELET SOURCE FILE 162

CHAPTER 40. JMS - IBM MQ KAMELET SOURCE 163
40.1. CONFIGURATION OPTIONS 163
40.2. DEPENDENCIES 163
40.3. USAGE 164
40.3.1. Knative Source 164
40.3.1.1. Prerequisite 164
40.3.1.2. Procedure for using the cluster CLI 164
40.3.1.3. Procedure for using the Kamel CLI 165
40.3.2. Kafka Source 165
40.3.2.1. Prerequisites 165
40.3.2.2. Procedure for using the cluster CLI 165
40.3.2.3. Procedure for using the Kamel CLI 166
40.4. KAMELET SOURCE FILE 166

CHAPTER 41. JSON DESERIALIZE ACTION 167
41.1. CONFIGURATION OPTIONS 167
41.2. DEPENDENCIES 167
41.3. USAGE 167
41.3.1. Knative Action 167
41.3.1.1. Prerequisite 167
41.3.1.2. Procedure for using the cluster CLI 168
41.3.1.3. Procedure for using the Kamel CLI 168
41.3.2. Kafka Action 168
41.3.2.1. Prerequisites 168
41.3.2.2. Procedure for using the cluster CLI 169
41.3.2.3. Procedure for using the Kamel CLI 169
41.4. KAMELET SOURCE FILE 169

CHAPTER 42. JSON SERIALIZE ACTION 170
42.1. CONFIGURATION OPTIONS 170
42.2. DEPENDENCIES 170
42.3. USAGE 170
42.3.1. Knative Action 170
42.3.1.1. Prerequisite 170
42.3.1.2. Procedure for using the cluster CLI 171
42.3.1.3. Procedure for using the Kamel CLI 171
42.3.2. Kafka Action 171
42.3.2.1. Prerequisites 171
42.3.2.2. Procedure for using the cluster CLI 172
42.3.2.3. Procedure for using the Kamel CLI 172
42.4. KAMELET SOURCE FILE 172

CHAPTER 43. KAFKA SINK 173
43.1. CONFIGURATION OPTIONS 173

43.2. DEPENDENCIES 174
43.3. USAGE 174
43.3.1. Knative Sink 174
43.3.1.1. Prerequisite 174
43.3.1.2. Procedure for using the cluster CLI 174
43.3.1.3. Procedure for using the Kamel CLI 175
43.3.2. Kafka Sink 175
43.3.2.1. Prerequisites 175
43.3.2.2. Procedure for using the cluster CLI 175
43.3.2.3. Procedure for using the Kamel CLI 175
43.4. KAMELET SOURCE FILE 176

CHAPTER 44. KAFKA SOURCE 177
44.1. CONFIGURATION OPTIONS 177
44.2. DEPENDENCIES 178
44.3. USAGE 178
44.3.1. Knative Source 179
44.3.1.1. Prerequisite 179
44.3.1.2. Procedure for using the cluster CLI 179
44.3.1.3. Procedure for using the Kamel CLI 179
44.3.2. Kafka Source 179
44.3.2.1. Prerequisites 180
44.3.2.2. Procedure for using the cluster CLI 180
44.3.2.3. Procedure for using the Kamel CLI 180
44.4. KAMELET SOURCE FILE 180

CHAPTER 45. KAFKA TOPIC NAME MATCHES FILTER ACTION 181
45.1. CONFIGURATION OPTIONS 181
45.2. DEPENDENCIES 181
45.3. USAGE 181
45.3.1. Kafka Action 181
45.3.1.1. Prerequisites 182
45.3.1.2. Procedure for using the cluster CLI 182
45.3.1.3. Procedure for using the Kamel CLI 182
45.4. KAMELET SOURCE FILE 182

CHAPTER 46. LOG SINK 183
46.1. CONFIGURATION OPTIONS 183
46.2. DEPENDENCIES 183
46.3. USAGE 183
46.3.1. Knative Sink 183
46.3.1.1. Prerequisite 184
46.3.1.2. Procedure for using the cluster CLI 184
46.3.1.3. Procedure for using the Kamel CLI 184
46.3.2. Kafka Sink 184
46.3.2.1. Prerequisites 184
46.3.2.2. Procedure for using the cluster CLI 185
46.3.2.3. Procedure for using the Kamel CLI 185
46.4. KAMELET SOURCE FILE 185

CHAPTER 47. MARIADB SINK 186
47.1. CONFIGURATION OPTIONS 186
47.2. DEPENDENCIES 186
47.3. USAGE 187

47.3.1. Knative Sink 187
47.3.1.1. Prerequisite 187
47.3.1.2. Procedure for using the cluster CLI 187
47.3.1.3. Procedure for using the Kamel CLI 188
47.3.2. Kafka Sink 188
47.3.2.1. Prerequisites 188
47.3.2.2. Procedure for using the cluster CLI 188
47.3.2.3. Procedure for using the Kamel CLI 189
47.4. KAMELET SOURCE FILE 189

CHAPTER 48. MASK FIELDS ACTION 190
48.1. CONFIGURATION OPTIONS 190
48.2. DEPENDENCIES 190
48.3. USAGE 190
48.3.1. Knative Action 190
48.3.1.1. Prerequisite 191
48.3.1.2. Procedure for using the cluster CLI 191
48.3.1.3. Procedure for using the Kamel CLI 191
48.3.2. Kafka Action 191
48.3.2.1. Prerequisites 192
48.3.2.2. Procedure for using the cluster CLI 192
48.3.2.3. Procedure for using the Kamel CLI 192
48.4. KAMELET SOURCE FILE 192

CHAPTER 49. MESSAGE TIMESTAMP ROUTER ACTION 193
49.1. CONFIGURATION OPTIONS 193
49.2. DEPENDENCIES 194
49.3. USAGE 194
49.3.1. Knative Action 194
49.3.1.1. Prerequisite 194
49.3.1.2. Procedure for using the cluster CLI 195
49.3.1.3. Procedure for using the Kamel CLI 195
49.3.2. Kafka Action 195
49.3.2.1. Prerequisites 195
49.3.2.2. Procedure for using the cluster CLI 196
49.3.2.3. Procedure for using the Kamel CLI 196
49.4. KAMELET SOURCE FILE 196

CHAPTER 50. MONGODB SINK 197
50.1. CONFIGURATION OPTIONS 197
50.2. DEPENDENCIES 198
50.3. USAGE 198
50.3.1. Knative Sink 198
50.3.1.1. Prerequisite 199
50.3.1.2. Procedure for using the cluster CLI 199
50.3.1.3. Procedure for using the Kamel CLI 199
50.3.2. Kafka Sink 199
50.3.2.1. Prerequisites 200
50.3.2.2. Procedure for using the cluster CLI 200
50.3.2.3. Procedure for using the Kamel CLI 200
50.4. KAMELET SOURCE FILE 200

CHAPTER 51. MONGODB SOURCE 201
51.1. CONFIGURATION OPTIONS 201

51.2. DEPENDENCIES 202
51.3. USAGE 202
51.3.1. Knative Source 202
51.3.1.1. Prerequisite 203
51.3.1.2. Procedure for using the cluster CLI 203
51.3.1.3. Procedure for using the Kamel CLI 203
51.3.2. Kafka Source 203
51.3.2.1. Prerequisites 204
51.3.2.2. Procedure for using the cluster CLI 204
51.3.2.3. Procedure for using the Kamel CLI 204
51.4. KAMELET SOURCE FILE 204

CHAPTER 52. MYSQL SINK 205
52.1. CONFIGURATION OPTIONS 205
52.2. DEPENDENCIES 205
52.3. USAGE 206
52.3.1. Knative Sink 206
52.3.1.1. Prerequisite 206
52.3.1.2. Procedure for using the cluster CLI 206
52.3.1.3. Procedure for using the Kamel CLI 207
52.3.2. Kafka Sink 207
52.3.2.1. Prerequisites 207
52.3.2.2. Procedure for using the cluster CLI 207
52.3.2.3. Procedure for using the Kamel CLI 208
52.4. KAMELET SOURCE FILE 208

CHAPTER 53. POSTGRESQL SINK 209
53.1. CONFIGURATION OPTIONS 209
53.2. DEPENDENCIES 210
53.3. USAGE 210
53.3.1. Knative Sink 210
53.3.1.1. Prerequisite 210
53.3.1.2. Procedure for using the cluster CLI 210
53.3.1.3. Procedure for using the Kamel CLI 211
53.3.2. Kafka Sink 211
53.3.2.1. Prerequisites 211
53.3.2.2. Procedure for using the cluster CLI 211
53.3.2.3. Procedure for using the Kamel CLI 212
53.4. KAMELET SOURCE FILE 212

CHAPTER 54. PREDICATE FILTER ACTION 213
54.1. CONFIGURATION OPTIONS 213
54.2. DEPENDENCIES 213
54.3. USAGE 213
54.3.1. Knative Action 213
54.3.1.1. Prerequisite 214
54.3.1.2. Procedure for using the cluster CLI 214
54.3.1.3. Procedure for using the Kamel CLI 214
54.3.2. Kafka Action 214
54.3.2.1. Prerequisites 215
54.3.2.2. Procedure for using the cluster CLI 215
54.3.2.3. Procedure for using the Kamel CLI 215
54.4. KAMELET SOURCE FILE 215

CHAPTER 55. PROTOBUF DESERIALIZE ACTION 216
55.1. CONFIGURATION OPTIONS 216
55.2. DEPENDENCIES 216
55.3. USAGE 216
55.3.1. Knative Action 216
55.3.1.1. Prerequisite 217
55.3.1.2. Procedure for using the cluster CLI 217
55.3.1.3. Procedure for using the Kamel CLI 217
55.3.2. Kafka Action 218
55.3.2.1. Prerequisites 218
55.3.2.2. Procedure for using the cluster CLI 218
55.3.2.3. Procedure for using the Kamel CLI 219
55.4. KAMELET SOURCE FILE 219

CHAPTER 56. PROTOBUF SERIALIZE ACTION 220
56.1. CONFIGURATION OPTIONS 220
56.2. DEPENDENCIES 220
56.3. USAGE 220
56.3.1. Knative Action 220
56.3.1.1. Prerequisite 221
56.3.1.2. Procedure for using the cluster CLI 221
56.3.1.3. Procedure for using the Kamel CLI 221
56.3.2. Kafka Action 221
56.3.2.1. Prerequisites 222
56.3.2.2. Procedure for using the cluster CLI 222
56.3.2.3. Procedure for using the Kamel CLI 222
56.4. KAMELET SOURCE FILE 223

CHAPTER 57. REGEX ROUTER ACTION 224
57.1. CONFIGURATION OPTIONS 224
57.2. DEPENDENCIES 224
57.3. USAGE 224
57.3.1. Knative Action 224
57.3.1.1. Prerequisite 225
57.3.1.2. Procedure for using the cluster CLI 225
57.3.1.3. Procedure for using the Kamel CLI 225
57.3.2. Kafka Action 225
57.3.2.1. Prerequisites 226
57.3.2.2. Procedure for using the cluster CLI 226
57.3.2.3. Procedure for using the Kamel CLI 226
57.4. KAMELET SOURCE FILE 226

CHAPTER 58. REPLACE FIELD ACTION 227
58.1. CONFIGURATION OPTIONS 227
58.2. DEPENDENCIES 227
58.3. USAGE 228
58.3.1. Knative Action 228
58.3.1.1. Prerequisite 228
58.3.1.2. Procedure for using the cluster CLI 228
58.3.1.3. Procedure for using the Kamel CLI 228
58.3.2. Kafka Action 229
58.3.2.1. Prerequisites 229
58.3.2.2. Procedure for using the cluster CLI 229
58.3.2.3. Procedure for using the Kamel CLI 229

58.4. KAMELET SOURCE FILE 230

CHAPTER 59. SALESFORCE SOURCE 231
59.1. CONFIGURATION OPTIONS 231
59.2. DEPENDENCIES 231
59.3. USAGE 232
59.3.1. Knative Source 232
59.3.1.1. Prerequisite 232
59.3.1.2. Procedure for using the cluster CLI 232
59.3.1.3. Procedure for using the Kamel CLI 232
59.3.2. Kafka Source 233
59.3.2.1. Prerequisites 233
59.3.2.2. Procedure for using the cluster CLI 233
59.3.2.3. Procedure for using the Kamel CLI 233
59.4. KAMELET SOURCE FILE 234

CHAPTER 60. SALESFORCE CREATE SINK 235
60.1. CONFIGURATION OPTIONS 235
60.2. DEPENDENCIES 235
60.3. USAGE 235
60.3.1. Knative Sink 236
60.3.1.1. Prerequisite 236
60.3.1.2. Procedure for using the cluster CLI 236
60.3.1.3. Procedure for using the Kamel CLI 236
60.3.2. Kafka Sink 237
60.3.2.1. Prerequisites 237
60.3.2.2. Procedure for using the cluster CLI 237
60.3.2.3. Procedure for using the Kamel CLI 237
60.4. KAMELET SOURCE FILE 237

CHAPTER 61. SALESFORCE DELETE SINK 239
61.1. CONFIGURATION OPTIONS 239
61.2. DEPENDENCIES 239
61.3. USAGE 239
61.3.1. Knative Sink 240
61.3.1.1. Prerequisite 240
61.3.1.2. Procedure for using the cluster CLI 240
61.3.1.3. Procedure for using the Kamel CLI 240
61.3.2. Kafka Sink 241
61.3.2.1. Prerequisites 241
61.3.2.2. Procedure for using the cluster CLI 241
61.3.2.3. Procedure for using the Kamel CLI 241
61.4. KAMELET SOURCE FILE 241

CHAPTER 62. SALESFORCE UPDATE SINK 243
62.1. CONFIGURATION OPTIONS 243
62.2. DEPENDENCIES 243
62.3. USAGE 244
62.3.1. Knative Sink 244
62.3.1.1. Prerequisite 244
62.3.1.2. Procedure for using the cluster CLI 244
62.3.1.3. Procedure for using the Kamel CLI 244
62.3.2. Kafka Sink 245
62.3.2.1. Prerequisites 245

62.3.2.2. Procedure for using the cluster CLI 245
62.3.2.3. Procedure for using the Kamel CLI 245
62.4. KAMELET SOURCE FILE 246

CHAPTER 63. SFTP SINK 247
63.1. CONFIGURATION OPTIONS 247
63.2. DEPENDENCIES 247
63.3. USAGE 248
63.3.1. Knative Sink 248
63.3.1.1. Prerequisite 248
63.3.1.2. Procedure for using the cluster CLI 248
63.3.1.3. Procedure for using the Kamel CLI 248
63.3.2. Kafka Sink 249
63.3.2.1. Prerequisites 249
63.3.2.2. Procedure for using the cluster CLI 249
63.3.2.3. Procedure for using the Kamel CLI 249
63.4. KAMELET SOURCE FILE 250

CHAPTER 64. SFTP SOURCE 251
64.1. CONFIGURATION OPTIONS 251
64.2. DEPENDENCIES 251
64.3. USAGE 252
64.3.1. Knative Source 252
64.3.1.1. Prerequisite 252
64.3.1.2. Procedure for using the cluster CLI 252
64.3.1.3. Procedure for using the Kamel CLI 252
64.3.2. Kafka Source 253
64.3.2.1. Prerequisites 253
64.3.2.2. Procedure for using the cluster CLI 253
64.3.2.3. Procedure for using the Kamel CLI 253
64.4. KAMELET SOURCE FILE 254

CHAPTER 65. SLACK SOURCE 255
65.1. CONFIGURATION OPTIONS 255
65.2. DEPENDENCIES 255
65.3. USAGE 255
65.3.1. Knative Source 255
65.3.1.1. Prerequisite 256
65.3.1.2. Procedure for using the cluster CLI 256
65.3.1.3. Procedure for using the Kamel CLI 256
65.3.2. Kafka Source 256
65.3.2.1. Prerequisites 257
65.3.2.2. Procedure for using the cluster CLI 257
65.3.2.3. Procedure for using the Kamel CLI 257
65.4. KAMELET SOURCE FILE 257

CHAPTER 66. MICROSOFT SQL SERVER SINK 258
66.1. CONFIGURATION OPTIONS 258
66.2. DEPENDENCIES 258
66.3. USAGE 259
66.3.1. Knative Sink 259
66.3.1.1. Prerequisite 259
66.3.1.2. Procedure for using the cluster CLI 259
66.3.1.3. Procedure for using the Kamel CLI 260

66.3.2. Kafka Sink 260
66.3.2.1. Prerequisites 260
66.3.2.2. Procedure for using the cluster CLI 260
66.3.2.3. Procedure for using the Kamel CLI 261
66.4. KAMELET SOURCE FILE 261

CHAPTER 67. TELEGRAM SOURCE 262
67.1. CONFIGURATION OPTIONS 262
67.2. DEPENDENCIES 262
67.3. USAGE 262
67.3.1. Knative Source 262
67.3.1.1. Prerequisite 263
67.3.1.2. Procedure for using the cluster CLI 263
67.3.1.3. Procedure for using the Kamel CLI 263
67.3.2. Kafka Source 263
67.3.2.1. Prerequisites 264
67.3.2.2. Procedure for using the cluster CLI 264
67.3.2.3. Procedure for using the Kamel CLI 264
67.4. KAMELET SOURCE FILE 264

CHAPTER 68. THROTTLE ACTION 265
68.1. CONFIGURATION OPTIONS 265
68.2. DEPENDENCIES 265
68.3. USAGE 265
68.3.1. Knative Action 265
68.3.1.1. Prerequisite 266
68.3.1.2. Procedure for using the cluster CLI 266
68.3.1.3. Procedure for using the Kamel CLI 266
68.3.2. Kafka Action 266
68.3.2.1. Prerequisites 267
68.3.2.2. Procedure for using the cluster CLI 267
68.3.2.3. Procedure for using the Kamel CLI 267
68.4. KAMELET SOURCE FILE 267

CHAPTER 69. TIMER SOURCE 268
69.1. CONFIGURATION OPTIONS 268
69.2. DEPENDENCIES 268
69.3. USAGE 268
69.3.1. Knative Source 268
69.3.1.1. Prerequisite 269
69.3.1.2. Procedure for using the cluster CLI 269
69.3.1.3. Procedure for using the Kamel CLI 269
69.3.2. Kafka Source 269
69.3.2.1. Prerequisites 270
69.3.2.2. Procedure for using the cluster CLI 270
69.3.2.3. Procedure for using the Kamel CLI 270
69.4. KAMELET SOURCE FILE 270

CHAPTER 70. TIMESTAMP ROUTER ACTION 271
70.1. CONFIGURATION OPTIONS 271
70.2. DEPENDENCIES 271
70.3. USAGE 271
70.3.1. Knative Action 271
70.3.1.1. Prerequisite 272

70.3.1.2. Procedure for using the cluster CLI 272
70.3.1.3. Procedure for using the Kamel CLI 272
70.3.2. Kafka Action 272
70.3.2.1. Prerequisites 273
70.3.2.2. Procedure for using the cluster CLI 273
70.3.2.3. Procedure for using the Kamel CLI 273
70.4. KAMELET SOURCE FILE 273

CHAPTER 71. VALUE TO KEY ACTION 274
71.1. CONFIGURATION OPTIONS 274
71.2. DEPENDENCIES 274
71.3. USAGE 274
71.3.1. Knative Action 274
71.3.1.1. Prerequisite 275
71.3.1.2. Procedure for using the cluster CLI 275
71.3.1.3. Procedure for using the Kamel CLI 275
71.3.2. Kafka Action 275
71.3.2.1. Prerequisites 276
71.3.2.2. Procedure for using the cluster CLI 276
71.3.2.3. Procedure for using the Kamel CLI 276
71.4. KAMELET SOURCE FILE 276

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"}

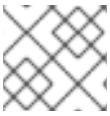
The Kamelet will insert/update an item in the given AWS DynamoDB table and set the attributes 'username' and 'city' respectively. Please note that the JSON object must include the primary key values that define the item.

1.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
region *	AWS Region	The AWS region to connect to	string		**"eu-west-1"**
table *	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	**"PutItem"**	**"PutItem"**
overrideEndpoint	Endpoint Overwrite	Set the need for overiding the endpoint URI. This option needs to be used in combination with uriEndpointOverride setting.	boolean	**false**	

Property	Name	Description	Type	Default	Example
secretKey	Secret Key	The secret key obtained from AWS	string		
uriEndpointOverride	Overwrite Endpoint URI	Set the overriding endpoint URI. This option needs to be used in combination with overrideEndpoint 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	**false**	
writeCapacity	Write Capacity	The provisioned throughput to reserved for writing resources to your table	integer	**1**	



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

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

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/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:

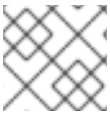
- **sequence-number** / **ce-sequencenumber**: to set the Sequence number

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
accessKey *	Access Key	The access key obtained from AWS	string		
region *	AWS Region	The AWS region to connect to	string		**"eu-west-1"**
secretKey *	Secret Key	The secret key obtained from AWS	string		
stream *	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/aws-kinesis-sink.kamelet.yaml

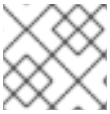
CHAPTER 5. AWS KINESIS SOURCE

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
accessKey *	Access Key	The access key obtained from AWS	string		
region *	AWS Region	The AWS region to connect to	string		**"eu-west-1"**
secretKey *	Secret Key	The secret key obtained from AWS	string		
stream *	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/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
accessKey *	Access Key	The access key obtained from AWS	string		
function *	Function Name	The Lambda Function name	string		
region *	AWS Region	The AWS region to connect to	string		**"eu-west-1"**
secretKey *	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/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
databaseName *	Database Name	The Database Name we are pointing	string		
password *	Password	The password to use for accessing a secured AWS Redshift Database	string		
query *	Query	The Query to execute against the AWS Redshift Database	string		**"INSERT INTO accounts (username,city) VALUES (:#username,:#city)"**
serverName *	Server Name	Server Name for the data source	string		**"localhost"**
username *	Username	The username to use for accessing a secured AWS Redshift Database	string		
serverPort	Server Port	Server Port for the data source	string	**5439**	

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/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
accessKey *	Access Key	The access key obtained from AWS	string		
region *	AWS Region	The AWS region to connect to	string		**"eu-west-1"**
secretKey *	Secret Key	The secret key obtained from AWS	string		
topicNameOrArn *	Topic Name	The SQS Topic name or ARN	string		
autoCreateTopic	Autocreate Topic	Setting the autocreation of the SNS topic.	boolean	**false**	

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

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/aws-sqs-sink.kamelet.yaml

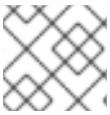
CHAPTER 10. AWS SQS SOURCE

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
accessKey *	Access Key	The access key obtained from AWS	string		
queueNameOrArn *	Queue Name	The SQS Queue name or ARN	string		
region *	AWS Region	The AWS region to connect to	string		**"eu-west-1"**
secretKey *	Secret Key	The secret key obtained from AWS	string		
autoCreateQueue	Autocreate Queue	Setting the autocreation of the SQS queue.	boolean	**false**	
deleteAfterRead	Auto-delete Messages	Delete messages after consuming them	boolean	**true**	



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/aws-sqs-source.kamelet.yaml

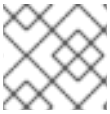
CHAPTER 11. AWS 2 SIMPLE QUEUE SERVICE FIFO SINK

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



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

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/aws-s3-sink.kamelet.yaml

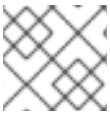
CHAPTER 13. AWS S3 SOURCE

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
accessKey *	Access Key	The access key obtained from AWS	string		
bucketNameOrArn *	Bucket Name	The S3 Bucket name or ARN	string		
region *	AWS Region	The AWS region to connect to	string		**"eu-west-1"**
secretKey *	Secret Key	The secret key obtained from AWS	string		
autoCreateBucket	Autocreate Bucket	Setting the autocreation of the S3 bucket bucketName.	boolean	**false**	
deleteAfterRead	Auto-delete Objects	Delete objects after consuming them	boolean	**true**	



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/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
accessKey *	Access Key	The access key obtained from AWS.	string		
bucketNameOrArn *	Bucket Name	The S3 Bucket name or ARN.	string		
keyName *	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		
region *	AWS Region	The AWS region to connect to.	string		**"eu-west-1"**
secretKey *	Secret Key	The secret key obtained from AWS.	string		
autoCreateBucket	Autocreate Bucket	Setting the autocreation of the S3 bucket bucketName.	boolean	**false**	
batchMessageNumber	Batch Message Number	The number of messages composing a batch in streaming upload mode	int	**10**	
batchSize	Batch Size	The batch size (in bytes) in streaming upload mode	int	**1000000**	

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	**"progressive"**	
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	**"lastPart"**	
streamingUploadMode	Streaming Upload Mode	Setting the Streaming Upload Mode	boolean	**true**	

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/aws-s3-streaming-upload-sink.kamelet.yaml

CHAPTER 15. AZURE STORAGE BLOB SINK

Upload data to Azure Storage Blob.

IMPORTANT

The Azure Storage Blob Sink Kamelet is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production.

These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process. For more information about the support scope of Red Hat Technology Preview features, see https://access.redhat.com/support/offerings/techpreview.

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.

15.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **azure-storage-blob-sink** Kamelet:

Property	Name	Description	Type	Default	Example
accessKey *	Access Key	The Azure Storage Blob access Key.	string		
accountName *	Account Name	The Azure Storage Blob account name.	string		
containerName *	Container Name	The Azure Storage Blob container name.	string		
credentialType	Credential Type	Determines the credential strategy to adopt. Possible values are SHARED_ACCOUNT_KEY, SHARED_KEY_CREDENTIAL and AZURE_IDENTITY	string	**"SHARED_ACCOUNT_KEY"**	
operation	Operation Name	The operation to perform.	string	**"uploadBlockBlob"**	

NOTE

Fields marked with an asterisk (*) are mandatory.

15.2. DEPENDENCIES

At runtime, the **azure-storage-blob-sink** Kamelet relies upon the presence of the following dependencies:

- camel:azure-storage-blob
- camel:kamelet

15.3. USAGE

This section describes how you can use the **azure-storage-blob-sink**.

15.3.1. Knative Sink

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

azure-storage-blob-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: azure-storage-blob-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: azure-storage-blob-sink
    properties:
      accessKey: "The Access Key"
      accountName: "The Account Name"
      containerName: "The Container Name"
```

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 **azure-storage-blob-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 azure-storage-blob-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 azure-storage-blob-sink -p "sink.accessKey=The Access Key" -p
"sink.accountName=The Account Name" -p "sink.containerName=The Container Name"
```

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

15.3.2. Kafka Sink

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

azure-storage-blob-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: azure-storage-blob-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: azure-storage-blob-sink
    properties:
      accessKey: "The Access Key"
      accountName: "The Account Name"
      containerName: "The Container Name"
```

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 **azure-storage-blob-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 azure-storage-blob-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 azure-storage-blob-sink -p
"sink.accessKey=The Access Key" -p "sink.accountName=The Account Name" -p
"sink.containerName=The Container Name"
```

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

15.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/azure-storage-blob-sink.kamelet.yaml

CHAPTER 16. AZURE STORAGE BLOB SOURCE

Consume Files from Azure Storage Blob.

IMPORTANT

The Azure Storage Blob Source Kamelet is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production.

These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process. For more information about the support scope of Red Hat Technology Preview features, see https://access.redhat.com/support/offerings/techpreview.

16.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **azure-storage-blob-source** Kamelet:

Property	Name	Description	Type	Default	Example
accessKey *	Access Key	The Azure Storage Blob access Key.	string		
accountName *	Account Name	The Azure Storage Blob account name.	string		
containerName *	Container Name	The Azure Storage Blob container name.	string		
period *	Period Between Polls	The interval between fetches to the Azure Storage Container in milliseconds	integer	**10000**	
credentialType	Credential Type	Determines the credential strategy to adopt. Possible values are SHARED_ACCOUNT_KEY, SHARED_KEY_CREDENTIAL and AZURE_IDENTITY	string	**"SHARED_ACCOUNT_KEY"**	

NOTE

Fields marked with an asterisk (*) are mandatory.

16.2. DEPENDENCIES

At runtime, the **azure-storage-blob-source** Kamelet relies upon the presence of the following dependencies:

- camel:azure-storage-blob
- camel:jsonpath
- camel:core
- camel:timer
- camel:kamelet

16.3. USAGE

This section describes how you can use the **azure-storage-blob-source**.

16.3.1. Knative Source

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

azure-storage-blob-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: azure-storage-blob-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: azure-storage-blob-source
    properties:
      accessKey: "The Access Key"
      accountName: "The Account Name"
      containerName: "The Container Name"
  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 **azure-storage-blob-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 azure-storage-blob-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 azure-storage-blob-source -p "source.accessKey=The Access Key" -p
"source.accountName=The Account Name" -p "source.containerName=The Container Name"
channel:mychannel
```

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

16.3.2. Kafka Source

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

azure-storage-blob-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: azure-storage-blob-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: azure-storage-blob-source
    properties:
      accessKey: "The Access Key"
      accountName: "The Account Name"
      containerName: "The Container Name"
  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 **azure-storage-blob-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 azure-storage-blob-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 azure-storage-blob-source -p "source.accessKey=The Access Key" -p
"source.accountName=The Account Name" -p "source.containerName=The Container Name"
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

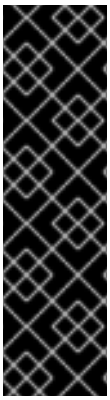
This command creates the KameletBinding in the current namespace on the cluster.

16.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/azure-storage-blob-source.kamelet.yaml

CHAPTER 17. AZURE STORAGE QUEUE SINK

Send Messages to Azure Storage queues.



IMPORTANT

The Azure Storage Queue Sink Kamelet is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production.

These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process. For more information about the support scope of Red Hat Technology Preview features, see https://access.redhat.com/support/offerings/techpreview.

The Kamelet is able to understand the following headers to be set:

- **expiration** / **ce-expiration**: as the time to live of the message in the queue.

If the header won't be set the default of 7 days will be used.

The format should be in this form: PnDTnHnMn.nS., e.g: PT20.345S – parses as 20.345 seconds, P2D – parses as 2 days.

17.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **azure-storage-queue-sink** Kamelet:

Property	Name	Description	Type	Default	Example
accessKey *	Access Key	The Azure Storage Queue access Key.	string		
accountName *	Account Name	The Azure Storage Queue account name.	string		
queueName *	Queue Name	The Azure Storage Queue container name.	string		

NOTE

Fields marked with an asterisk (*) are mandatory.

17.2. DEPENDENCIES

At runtime, the **azure-storage-queue-sink** Kamelet relies upon the presence of the following dependencies:

- camel:azure-storage-queue
- camel:kamelet

17.3. USAGE

This section describes how you can use the **azure-storage-queue-sink**.

17.3.1. Knative Sink

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

azure-storage-queue-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: azure-storage-queue-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: azure-storage-queue-sink
    properties:
      accessKey: "The Access Key"
      accountName: "The Account Name"
      queueName: "The Queue Name"
```

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 **azure-storage-queue-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 azure-storage-queue-sink-binding.yaml
```

17.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel azure-storage-queue-sink -p "sink.accessKey=The Access Key" -p
"sink.accountName=The Account Name" -p "sink.queueName=The Queue Name"
```

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

17.3.2. Kafka Sink

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

azure-storage-queue-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: azure-storage-queue-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: azure-storage-queue-sink
    properties:
      accessKey: "The Access Key"
      accountName: "The Account Name"
      queueName: "The Queue Name"
```

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 **azure-storage-queue-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 azure-storage-queue-sink-binding.yaml
   ```

17.3.2.3. 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 azure-storage-queue-sink -p
"sink.accessKey=The Access Key" -p "sink.accountName=The Account Name" -p
"sink.queueName=The Queue Name"
```

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

17.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/azure-storage-queue-sink.kamelet.yaml

CHAPTER 18. AZURE STORAGE QUEUE SOURCE

Receive Messages from Azure Storage queues.

IMPORTANT

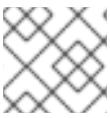
The Azure Storage Queue Source Kamelet is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production.

These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process. For more information about the support scope of Red Hat Technology Preview features, see https://access.redhat.com/support/offerings/techpreview.

18.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **azure-storage-queue-source** Kamelet:

Property	Name	Description	Type	Default	Example
accessKey *	Access Key	The Azure Storage Queue access Key.	string		
accountName *	Account Name	The Azure Storage Queue account name.	string		
queueName *	Queue Name	The Azure Storage Queue container name.	string		
maxMessages	Maximum Messages	Maximum number of messages to get, if there are less messages exist in the queue than requested all the messages will be returned. By default it will consider 1 message to be retrieved, the allowed range is 1 to 32 messages.	int	**1**	



NOTE

Fields marked with an asterisk (*) are mandatory.

18.2. DEPENDENCIES

At runtime, the **azure-storage-queue-source** Kamelet relies upon the presence of the following dependencies:

- camel:azure-storage-queue
- camel:kamelet

18.3. USAGE

This section describes how you can use the **azure-storage-queue-source**.

18.3.1. Knative Source

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

azure-storage-queue-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: azure-storage-queue-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: azure-storage-queue-source
    properties:
      accessKey: "The Access Key"
      accountName: "The Account Name"
      queueName: "The Queue Name"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

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 **azure-storage-queue-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 azure-storage-queue-source-binding.yaml
```

18.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind azure-storage-queue-source -p "source.accessKey=The Access Key" -p
"source.accountName=The Account Name" -p "source.queueName=The Queue Name"
channel:mychannel
```

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

18.3.2. Kafka Source

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

azure-storage-queue-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: azure-storage-queue-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: azure-storage-queue-source
    properties:
      accessKey: "The Access Key"
      accountName: "The Account Name"
      queueName: "The Queue Name"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

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 **azure-storage-queue-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 azure-storage-queue-source-binding.yaml
```

18.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind azure-storage-queue-source -p "source.accessKey=The Access Key" -p
"source.accountName=The Account Name" -p "source.queueName=The Queue Name"
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

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

18.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/azure-storage-queue-source.kamelet.yaml

CHAPTER 19. 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.

19.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
connectionHost *	Connection Host	Hostname(s) cassandra server(s). Multiple hosts can be separated by comma.	string		**"localhost"**
connectionPort *	Connection Port	Port number of cassandra server(s)	string		**9042**
keyspace *	Keyspace	Keyspace to use	string		**"customers"**
password *	Password	The password to use for accessing a secured Cassandra Cluster	string		
query *	Query	The query to execute against the Cassandra cluster table	string		
username *	Username	The username to use for accessing a secured Cassandra Cluster	string		
consistencyLevel	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	**"ANY"**	

NOTE

Fields marked with an asterisk (*) are mandatory.

19.2. DEPENDENCIES

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

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

19.3. USAGE

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

19.3.1. 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"
```

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 **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
```

19.3.1.3. 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.

19.3.2. 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"
```

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 **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
```

19.3.2.3. 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.

19.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/cassandra-sink.kamelet.yaml

CHAPTER 20. CASSANDRA SOURCE

Query a Cassandra cluster table.

20.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
connectionHost *	Connection Host	Hostname(s) cassandra server(s). Multiple hosts can be separated by comma.	string		**"localhost"**
connectionPort *	Connection Port	Port number of cassandra server(s)	string		**9042**
keyspace *	Keyspace	Keyspace to use	string		**"customers"**
password *	Password	The password to use for accessing a secured Cassandra Cluster	string		
query *	Query	The query to execute against the Cassandra cluster table	string		
username *	Username	The username to use for accessing a secured Cassandra Cluster	string		
consistencyLevel	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	**"QUORUM"**	

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.

20.2. DEPENDENCIES

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

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

20.3. USAGE

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

20.3.1. 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
```

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 **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
```

20.3.1.3. 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.

20.3.2. 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
```

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 **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
```

20.3.2.3. 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.

20.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/cassandra-source.kamelet.yaml

CHAPTER 21. CEPH SINK

Upload data to an Ceph Bucket managed by a Object Storage Gateway.

In the header, you can optionally set the **file** / **ce-file** property to specify the name of the file to upload.

If you do not set the property in the header, the Kamelet uses the exchange ID for the file name.

21.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
accessKey *	Access Key	The access key.	string		
bucketName *	Bucket Name	The Ceph Bucket name.	string		
cephUrl *	Ceph Url Address	Set the Ceph Object Storage Address Url.	string		**"http://ceph-storage-address.com"**
secretKey *	Secret Key	The secret key.	string		
zoneGroup *	Bucket Zone Group	The bucket zone group.	string		
autoCreateBucket	Autocreate Bucket	Specifies to automatically create the bucket.	boolean	**false**	
keyName	Key Name	The key name for saving an element in the bucket.	string		

NOTE

Fields marked with an asterisk (*) are mandatory.

21.2. DEPENDENCIES

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

- camel:core
- camel:aws2-s3
- camel:kamelet

21.3. USAGE

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

21.3.1. Knative Sink

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

ceph-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: ceph-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: ceph-sink
    properties:
      accessKey: "The Access Key"
      bucketName: "The Bucket Name"
      cephUrl: "http://ceph-storage-address.com"
      secretKey: "The Secret Key"
      zoneGroup: "The Bucket Zone Group"
```

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 **ceph-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 ceph-sink-binding.yaml
```

21.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel ceph-sink -p "sink.accessKey=The Access Key" -p
"sink.bucketName=The Bucket Name" -p "sink.cephUrl=http://ceph-storage-address.com" -p
"sink.secretKey=The Secret Key" -p "sink.zoneGroup=The Bucket Zone Group"
```

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

21.3.2. Kafka Sink

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

ceph-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: ceph-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: ceph-sink
    properties:
      accessKey: "The Access Key"
      bucketName: "The Bucket Name"
      cephUrl: "http://ceph-storage-address.com"
      secretKey: "The Secret Key"
      zoneGroup: "The Bucket Zone Group"
```

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 **ceph-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 ceph-sink-binding.yaml
```

21.3.2.3. 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 ceph-sink -p "sink.accessKey=The Access Key" -p "sink.bucketName=The Bucket Name" -p "sink.cephUrl=http://ceph-storage-address.com" -p "sink.secretKey=The Secret Key" -p "sink.zoneGroup=The Bucket Zone Group"
```

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

21.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/ceph-sink.kamelet.yaml

CHAPTER 22. CEPH SOURCE

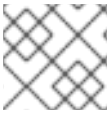
Receive data from an Ceph Bucket, managed by a Object Storage Gateway.

22.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
accessKey *	Access Key	The access key.	string		
bucketName *	Bucket Name	The Ceph Bucket name.	string		
cephUrl *	Ceph Url Address	Set the Ceph Object Storage Address Url.	string		**"http://ceph-storage-address.com"**
secretKey *	Secret Key	The secret key.	string		
zoneGroup *	Bucket Zone Group	The bucket zone group.	string		
autoCreateBucket	Autocreate Bucket	Specifies to automatically create the bucket.	boolean	**false**	
delay	Delay	The number of milliseconds before the next poll of the selected bucket.	integer	**500**	
deleteAfterRead	Auto-delete Objects	Specifies to delete objects after consuming them.	boolean	**true**	
ignoreBody	Ignore Body	If true, the Object body is ignored. Setting this to true overrides any behavior defined by the **includeBody** option. If false, the object is put in the body.	boolean	**false**	

Property	Name	Description	Type	Default	Example
includeBody	Include Body	If true, the exchange is consumed and put into the body and closed. If false, the Object stream is put raw into the body and the headers are set with the object metadata.	boolean	**true**	
prefix	Prefix	The bucket prefix to consider while searching.	string		**"folder/"**



NOTE

Fields marked with an asterisk (*) are mandatory.

22.2. DEPENDENCIES

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

- camel:aws2-s3
- camel:kamelet

22.3. USAGE

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

22.3.1. Knative Source

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

ceph-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: ceph-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: ceph-source
    properties:
      accessKey: "The Access Key"
      bucketName: "The Bucket Name"
      cephUrl: "http://ceph-storage-address.com"
```

```
    secretKey: "The Secret Key"
    zoneGroup: "The Bucket Zone Group"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

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 **ceph-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 ceph-source-binding.yaml
```

22.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind ceph-source -p "source.accessKey=The Access Key" -p "source.bucketName=The Bucket Name" -p "source.cephUrl=http://ceph-storage-address.com" -p "source.secretKey=The Secret Key" -p "source.zoneGroup=The Bucket Zone Group" channel:mychannel
```

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

22.3.2. Kafka Source

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

ceph-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: ceph-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: ceph-source
    properties:
      accessKey: "The Access Key"
      bucketName: "The Bucket Name"
      cephUrl: "http://ceph-storage-address.com"
      secretKey: "The Secret Key"
```

```
    zoneGroup: "The Bucket Zone Group"
sink:
  ref:
    kind: KafkaTopic
    apiVersion: kafka.strimzi.io/v1beta1
    name: my-topic
```

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 **ceph-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 ceph-source-binding.yaml
```

22.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind ceph-source -p "source.accessKey=The Access Key" -p "source.bucketName=The Bucket Name" -p "source.cephUrl=http://ceph-storage-address.com" -p "source.secretKey=The Secret Key" -p "source.zoneGroup=The Bucket Zone Group" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

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

22.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/ceph-source.kamelet.yaml

CHAPTER 23. EXTRACT FIELD ACTION

Extract a field from the body

23.1. CONFIGURATION OPTIONS

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

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

NOTE

Fields marked with an asterisk (*) are mandatory.

23.2. 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

23.3. USAGE

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

23.3.1. 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
```

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 **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
```

23.3.1.3. 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.

23.3.2. 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
```

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 **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
```

23.3.2.3. 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.

23.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/extract-field-action.kamelet.yaml

CHAPTER 24. 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.

24.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
connectionHost *	Connection Host	Hostname of the FTP server	string		
connectionPort *	Connection Port	Port of the FTP server	string	**21**	
directoryName *	Directory Name	The starting directory	string		
password *	Password	The password to access the FTP server	string		
username *	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	**"Override"**	
passiveMode	Passive Mode	Sets passive mode connection	boolean	**false**	

NOTE

Fields marked with an asterisk (*) are mandatory.

24.2. DEPENDENCIES

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

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

24.3. USAGE

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

24.3.1. 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"
```

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 **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
```

24.3.1.3. 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.

24.3.2. 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"
```

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 **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
```

24.3.2.3. 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.

24.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/ftp-sink.kamelet.yaml

CHAPTER 25. FTP SOURCE

Receive data from an FTP Server.

25.1. CONFIGURATION OPTIONS

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

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

NOTE

Fields marked with an asterisk (*) are mandatory.

25.2. DEPENDENCIES

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

- camel:ftp
- camel:core

- camel:kamelet

25.3. USAGE

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

25.3.1. 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
```

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 **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
```

25.3.1.3. 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.

25.3.2. 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
```

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 **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
```

25.3.2.3. 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.

25.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/ftp-source.kamelet.yaml

CHAPTER 26. HAS HEADER FILTER ACTION

Filter based on the presence of one header

26.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
name *	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		**"headerName"**

NOTE

Fields marked with an asterisk (*) are mandatory.

26.2. DEPENDENCIES

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

- camel:core
- camel:kamelet

26.3. USAGE

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

26.3.1. 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
```

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 **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
```

26.3.1.3. 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.

26.3.2. 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
```

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 **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
```

26.3.2.3. 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.

26.4. KAMELET SOURCE FILE

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

CHAPTER 27. HOIST FIELD ACTION

Wrap data in a single field

27.1. CONFIGURATION OPTIONS

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

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

NOTE

Fields marked with an asterisk (*) are mandatory.

27.2. 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

27.3. USAGE

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

27.3.1. 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
```

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 **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
```

27.3.1.3. 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.

27.3.2. 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
```

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 **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
```

27.3.2.3. 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.

27.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/hoist-field-action.kamelet.yaml

CHAPTER 28. HTTP SINK

Forwards an event to a HTTP endpoint

28.1. 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		**"https://my-service/path"**
method	Method	The HTTP method to use	string	**"POST"**	

NOTE

Fields marked with an asterisk (*) are mandatory.

28.2. DEPENDENCIES

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

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

28.3. USAGE

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

28.3.1. 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"
```

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 **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
```

28.3.1.3. 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.

28.3.2. 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"
```

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 **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
```

28.3.2.3. 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.

28.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/http-sink.kamelet.yaml

CHAPTER 29. INSERT FIELD ACTION

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

29.1. CONFIGURATION OPTIONS

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

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

NOTE

Fields marked with an asterisk (*) are mandatory.

29.2. 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

29.3. USAGE

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

29.3.1. 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
```

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 **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
```

29.3.1.3. 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.

29.3.2. 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
```

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 **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
```

29.3.2.3. 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.

29.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/insert-field-action.kamelet.yaml

CHAPTER 30. INSERT HEADER ACTION

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

30.1. CONFIGURATION OPTIONS

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

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

NOTE

Fields marked with an asterisk (*) are mandatory.

30.2. DEPENDENCIES

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

- camel:core
- camel:kamelet

30.3. USAGE

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

30.3.1. 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
```

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 **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
```

30.3.1.3. 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.

30.3.2. 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
```

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 **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
```

30.3.2.3. 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.

30.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/insert-header-action.kamelet.yaml

CHAPTER 31. IS TOMBSTONE FILTER ACTION

Filter based on the presence of body or not

31.1. CONFIGURATION OPTIONS

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

31.2. DEPENDENCIES

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

- camel:core
- camel:kamelet

31.3. USAGE

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

31.3.1. 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
```

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 **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
```

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 is-tombstone-filter-action channel:mychannel
```

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

31.3.2. 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
```

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 **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
```

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 is-tombstone-filter-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/is-tombstone-filter-action.kamelet.yaml

CHAPTER 32. JIRA ADD COMMENT SINK

Add a new comment to an existing issue in Jira.

The Kamelet expects the following headers to be set:

- **issueKey** / **ce-issueKey**: as the issue code.

The comment is set in the body of the message.

32.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **jira-add-comment-sink** Kamelet:

Property	Name	Description	Type	Default	Example
jiraUrl *	Jira URL	The URL of your instance of Jira	string		**"http://my_jira.com:8081"**
password *	Password	The password or the API Token to access Jira	string		
username *	Username	The username to access Jira	string		

NOTE

Fields marked with an asterisk (*) are mandatory.

32.2. DEPENDENCIES

At runtime, the **jira-add-comment-sink** Kamelet relies upon the presence of the following dependencies:

- camel:core
- camel:jackson
- camel:jira
- camel:kamelet
- mvn:com.fasterxml.jackson.datatype:jackson-datatype-joda:2.12.4.redhat-00001

32.3. USAGE

This section describes how you can use the **jira-add-comment-sink**.

32.3.1. Knative Sink

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

jira-add-comment-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: jira-add-comment-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: insert-header-action
    properties:
      name: "issueKey"
      value: "MYP-167"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
    properties:
      jiraUrl: "jira server url"
      username: "username"
      password: "password"
```

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 **jira-add-comment-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 jira-add-comment-sink-binding.yaml
```

32.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind --name jira-add-comment-sink-binding timer-source?message="The new
comment"\&period=60000 --step insert-header-action -p step-0.name=issueKey -p step-
0.value=MYP-167 jira-add-comment-sink?
password="password"\&username="username"\&jiraUrl="jira url"
```

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

32.3.2. Kafka Sink

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

jira-add-comment-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: jira-add-comment-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: insert-header-action
    properties:
      name: "issueKey"
      value: "MYP-167"
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: jira-add-comment-sink
    properties:
      jiraUrl: "jira server url"
      username: "username"
      password: "password"
```

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 **jira-add-comment-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 jira-add-comment-sink-binding.yaml
```

32.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind --name jira-add-comment-sink-binding timer-source?message="The new
comment"\&period=60000 --step insert-header-action -p step-0.name=issueKey -p step-
0.value=MYP-167 jira-add-comment-sink?
password="password"\&username="username"\&jiraUrl="jira url"
```

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

32.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/jira-add-comment-sink.kamelet.yaml

CHAPTER 33. JIRA ADD ISSUE SINK

Add a new issue to Jira.

The Kamelet expects the following headers to be set:

- **projectKey** / **ce-projectKey**: as the Jira project key.
- **issueTypeName** / **ce-issueTypeName**: as the name of the issue type (example: Bug, Enhancement).
- **issueSummary** / **ce-issueSummary**: as the title or summary of the issue.
- **issueAssignee** / **ce-issueAssignee**: as the user assigned to the issue (Optional).
- **issuePriorityName** / **ce-issuePriorityName**: as the priority name of the issue (example: Critical, Blocker, Trivial) (Optional).
- **issueComponents** / **ce-issueComponents**: as list of string with the valid component names (Optional).
- **issueDescription** / **ce-issueDescription**: as the issue description (Optional).

The issue description can be set from the body of the message or the **issueDescription**/**ce-issueDescription** in the header, however the body takes precedence.

33.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **jira-add-issue-sink** Kamelet:

Property	Name	Description	Type	Default	Example
jiraUrl *	Jira URL	The URL of your instance of Jira	string		**"http://my_jira.com:8081"**
password *	Password	The password or the API Token to access Jira	string		
username *	Username	The username to access Jira	string		

NOTE

Fields marked with an asterisk (*) are mandatory.

33.2. DEPENDENCIES

At runtime, the **jira-add-issue-sink** Kamelet relies upon the presence of the following dependencies:

- camel:core
- camel:jackson

- camel:jira
- camel:kamelet
- mvn:com.fasterxml.jackson.datatype:jackson-datatype-joda:2.12.4.redhat-00001

33.3. USAGE

This section describes how you can use the **jira-add-issue-sink**.

33.3.1. Knative Sink

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

jira-add-issue-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: jira-add-issue-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: insert-header-action
    properties:
      name: "projectKey"
      value: "MYP"
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: insert-header-action
    properties:
      name: "issueTypeName"
      value: "Bug"
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: insert-header-action
    properties:
      name: "issueSummary"
      value: "The issue summary"
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: insert-header-action
    properties:
      name: "issuePriorityName"
      value: "Low"
```

```
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
    properties:
      jiraUrl: "jira server url"
      username: "username"
      password: "password"
```

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 **jira-add-issue-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 jira-add-issue-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 --name jira-add-issue-sink-binding timer-source?message="The new
comment"\&period=60000 --step insert-header-action -p step-0.name=projectKey -p step-
0.value=MYP --step insert-header-action -p step-1.name=issueTypeName -p step-1.value=Bug --
step insert-header-action  -p step-2.name=issueSummary -p step-2.value="This is a bug" --step
insert-header-action -p step-3.name=issuePriorityName -p step-3.value=Low jira-add-issue-sink?
jiraUrl="jira url"\&username="username"\&password="password"
```

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

33.3.2. Kafka Sink

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

jira-add-issue-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: jira-add-issue-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

```
steps:
- ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: insert-header-action
  properties:
    name: "projectKey"
    value: "MYP"
- ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: insert-header-action
  properties:
    name: "issueTypeName"
    value: "Bug"
- ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: insert-header-action
  properties:
    name: "issueSummary"
    value: "The issue summary"
- ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: insert-header-action
  properties:
    name: "issuePriorityName"
    value: "Low"
sink:
  ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: jira-add-issue-sink
  properties:
    jiraUrl: "jira server url"
    username: "username"
    password: "password"
```

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 **jira-add-issue-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 jira-add-issue-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 --name jira-add-issue-sink-binding timer-source?message="The new comment"\&period=60000 --step insert-header-action -p step-0.name=projectKey -p step-0.value=MYP --step insert-header-action -p step-1.name=issueTypeName -p step-1.value=Bug --step insert-header-action  -p step-2.name=issueSummary -p step-2.value="This is a bug" --step insert-header-action -p step-3.name=issuePriorityName -p step-3.value=Low jira-add-issue-sink?jiraUrl="jira url"\&username="username"\&password="password"
```

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

33.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/jira-add-issue-sink.kamelet.yaml

CHAPTER 34. JIRA TRANSITION ISSUE SINK

Sets a new status (transition to) of an existing issue in Jira.

The Kamelet expects the following headers to be set:

- **issueKey** / **ce-issueKey**: as the issue unique code.
- **issueTransitionId** / **ce-issueTransitionId**: as the new status (transition) code. You should carefully check the project workflow as each transition may have conditions to check before the transition is made.

The comment of the transition is set in the body of the message.

34.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **jira-transition-issue-sink** Kamelet:

Property	Name	Description	Type	Default	Example
jiraUrl *	Jira URL	The URL of your instance of Jira	string		**"http://my_jira.com:8081"**
password *	Password	The password or the API Token to access Jira	string		
username *	Username	The username to access Jira	string		

NOTE

Fields marked with an asterisk (*) are mandatory.

34.2. DEPENDENCIES

At runtime, the **jira-transition-issue-sink** Kamelet relies upon the presence of the following dependencies:

- camel:core
- camel:jackson
- camel:jira
- camel:kamelet
- mvn:com.fasterxml.jackson.datatype:jackson-datatype-joda:2.12.4.redhat-00001

34.3. USAGE

This section describes how you can use the **jira-transition-issue-sink**.

34.3.1. Knative Sink

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

jira-transition-issue-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: jira-transition-issue-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: insert-header-action
    properties:
      name: "issueTransitionId"
      value: 701
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: insert-header-action
    properties:
      name: "issueKey"
      value: "MYP-162"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
    properties:
      jiraUrl: "jira server url"
      username: "username"
      password: "password"
```

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 **jira-transition-issue-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 jira-transition-issue-sink-binding.yaml
```

34.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind --name jira-transition-issue-sink-binding timer-source?message="The new comment 123"\&period=60000 --step insert-header-action -p step-0.name=issueKey -p step-0.value=MYP-170 --step insert-header-action -p step-1.name=issueTransitionId -p step-1.value=5 jira-transition-issue-sink?jiraUrl="jira url"\&username="username"\&password="password"
```

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

34.3.2. Kafka Sink

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

jira-transition-issue-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: jira-transition-issue-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: insert-header-action
    properties:
      name: "issueTransitionId"
      value: 701
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: insert-header-action
    properties:
      name: "issueKey"
      value: "MYP-162"
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: jira-transition-issue-sink
    properties:
      jiraUrl: "jira server url"
      username: "username"
      password: "password"
```

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 **jira-transition-issue-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 jira-transition-issue-sink-binding.yaml
```

34.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind --name jira-transition-issue-sink-binding timer-source?message="The new comment 123"\&period=60000 --step insert-header-action -p step-0.name=issueKey -p step-0.value=MYP-170 --step insert-header-action -p step-1.name=issueTransitionId -p step-1.value=5 jira-transition-issue-sink?jiraUrl="jira url"\&username="username"\&password="password"
```

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

34.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/jira-transition-issue-sink.kamelet.yaml

CHAPTER 35. JIRA UPDATE ISSUE SINK

Update fields of an existing issue in Jira. The Kamelet expects the following headers to be set:

- **issueKey** / **ce-issueKey**: as the issue code in Jira.
- **issueTypeName** / **ce-issueTypeName**: as the name of the issue type (example: Bug, Enhancement).
- **issueSummary** / **ce-issueSummary**: as the title or summary of the issue.
- **issueAssignee** / **ce-issueAssignee**: as the user assigned to the issue (Optional).
- **issuePriorityName** / **ce-issuePriorityName**: as the priority name of the issue (example: Critical, Blocker, Trivial) (Optional).
- **issueComponents** / **ce-issueComponents**: as list of string with the valid component names (Optional).
- **issueDescription** / **ce-issueDescription**: as the issue description (Optional).

The issue description can be set from the body of the message or the **issueDescription**/**ce-issueDescription** in the header, however the body takes precedence.

35.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **jira-update-issue-sink** Kamelet:

Property	Name	Description	Type	Default	Example
jiraUrl *	Jira URL	The URL of your instance of Jira	string		**"http://my_jira.com:8081"**
password *	Password	The password or the API Token to access Jira	string		
username *	Username	The username to access Jira	string		

NOTE

Fields marked with an asterisk (*) are mandatory.

35.2. DEPENDENCIES

At runtime, the **jira-update-issue-sink** Kamelet relies upon the presence of the following dependencies:

- camel:core
- camel:jackson

- camel:jira
- camel:kamelet
- mvn:com.fasterxml.jackson.datatype:jackson-datatype-joda:2.12.4.redhat-00001

35.3. USAGE

This section describes how you can use the **jira-update-issue-sink**.

35.3.1. Knative Sink

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

jira-update-issue-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: jira-update-issue-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: insert-header-action
    properties:
      name: "issueKey"
      value: "MYP-163"
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: insert-header-action
    properties:
      name: "issueTypeName"
      value: "Bug"
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: insert-header-action
    properties:
      name: "issueSummary"
      value: "The issue summary"
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: insert-header-action
    properties:
      name: "issuePriorityName"
      value: "Low"
```

```
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
    properties:
      jiraUrl: "jira server url"
      username: "username"
      password: "password"
```

35.3.1.1. Prerequisite

Make 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 **jira-update-issue-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 jira-update-issue-sink-binding.yaml
```

35.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind --name jira-update-issue-sink-binding timer-source?message="The new
comment"\&period=60000 --step insert-header-action -p step-0.name=issueKey -p step-
0.value=MYP-170 --step insert-header-action -p step-1.name=issueTypeName -p step-1.value=Story
--step insert-header-action  -p step-2.name=issueSummary -p step-2.value="This is a story 123" --
step insert-header-action -p step-3.name=issuePriorityName -p step-3.value=Highest jira-update-
issue-sink?jiraUrl="jira url"\&username="username"\&password="password"
```

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

35.3.2. Kafka Sink

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

jira-update-issue-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: jira-update-issue-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

```
steps:
- ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: insert-header-action
  properties:
    name: "issueKey"
    value: "MYP-163"
- ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: insert-header-action
  properties:
    name: "issueTypeName"
    value: "Bug"
- ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: insert-header-action
  properties:
    name: "issueSummary"
    value: "The issue summary"
- ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: insert-header-action
  properties:
    name: "issuePriorityName"
    value: "Low"
sink:
  ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: jira-update-issue-sink
  properties:
    jiraUrl: "jira server url"
    username: "username"
    password: "password"
```

35.3.2.1. 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.

35.3.2.2. Procedure for using the cluster CLI

1. Save the **jira-update-issue-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 jira-update-issue-sink-binding.yaml
```

35.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind --name jira-update-issue-sink-binding timer-source?message="The new comment"\&period=60000 --step insert-header-action -p step-0.name=issueKey -p step-0.value=MYP-170 --step insert-header-action -p step-1.name=issueTypeName -p step-1.value=Story --step insert-header-action  -p step-2.name=issueSummary -p step-2.value="This is a story 123" --step insert-header-action -p step-3.name=issuePriorityName -p step-3.value=Highest jira-update-issue-sink?jiraUrl="jira url"\&username="username"\&password="password"
```

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

35.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/jira-update-issue-sink.kamelet.yaml

CHAPTER 36. JIRA SOURCE

Receive notifications about new issues from Jira.

36.1. CONFIGURATION OPTIONS

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

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

NOTE

Fields marked with an asterisk (*) are mandatory.

36.2. DEPENDENCIES

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

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

36.3. USAGE

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

36.3.1. 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
```

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 **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
```

36.3.1.3. 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.

36.3.2. 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
```

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 **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
```

36.3.2.3. 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.

36.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/jira-source.kamelet.yaml

CHAPTER 37. 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

37.1. CONFIGURATION OPTIONS

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

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

NOTE

Fields marked with an asterisk (*) are mandatory.

37.2. 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

37.3. USAGE

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

37.3.1. 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"
```

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 **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
```

37.3.1.3. 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.

37.3.2. 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"
```

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 **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
```

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 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.

37.4. KAMELET SOURCE FILE

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

CHAPTER 38. 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

38.1. CONFIGURATION OPTIONS

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

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

NOTE

Fields marked with an asterisk (*) are mandatory.

38.2. 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

38.3. USAGE

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

38.3.1. 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
```

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 **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
```

38.3.1.3. 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.

38.3.2. 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
```

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 **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
```

38.3.2.3. 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.

38.4. KAMELET SOURCE FILE

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

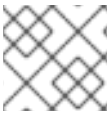
CHAPTER 39. JMS - IBM MQ KAMELET SINK

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

39.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
channel *	IBM MQ Channel	Name of the IBM MQ Channel	string		
destinationName *	Destination Name	The destination name	string		
password *	Password	Password to authenticate to IBM MQ server	string		
queueManager *	IBM MQ Queue Manager	Name of the IBM MQ Queue Manager	string		
serverName *	IBM MQ Server name	IBM MQ Server name or address	string		
serverPort *	IBM MQ Server Port	IBM MQ Server port	integer	**1414**	
username *	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	**"queue"**	



NOTE

Fields marked with an asterisk (*) are mandatory.

39.2. 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

39.3. USAGE

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

39.3.1. 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
```

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 **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
```

39.3.1.3. 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.

39.3.2. 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
```

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 **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
```

39.3.2.3. 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.

39.4. KAMELET SOURCE FILE

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

CHAPTER 40. JMS - IBM MQ KAMELET SOURCE

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

40.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
channel *	IBM MQ Channel	Name of the IBM MQ Channel	string		
destinationName *	Destination Name	The destination name	string		
password *	Password	Password to authenticate to IBM MQ server	string		
queueManager *	IBM MQ Queue Manager	Name of the IBM MQ Queue Manager	string		
serverName *	IBM MQ Server name	IBM MQ Server name or address	string		
serverPort *	IBM MQ Server Port	IBM MQ Server port	integer	**1414**	
username *	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	**"queue"**	

NOTE

Fields marked with an asterisk (*) are mandatory.

40.2. 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

40.3. USAGE

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

40.3.1. 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
```

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 **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
```

40.3.1.3. 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.

40.3.2. 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
```

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 **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
```

40.3.2.3. 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.

40.4. KAMELET SOURCE FILE

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

CHAPTER 41. JSON DESERIALIZE ACTION

Deserialize payload to JSON

41.1. CONFIGURATION OPTIONS

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

41.2. DEPENDENCIES

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

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

41.3. USAGE

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

41.3.1. 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
```

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 **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
```

41.3.1.3. 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.

41.3.2. 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
```

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 **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
```

41.3.2.3. 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.

41.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/json-deserialize-action.kamelet.yaml

CHAPTER 42. JSON SERIALIZE ACTION

Serialize payload to JSON

42.1. CONFIGURATION OPTIONS

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

42.2. DEPENDENCIES

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

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

42.3. USAGE

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

42.3.1. 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
```

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 **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
```

42.3.1.3. 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.

42.3.2. 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
```

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 **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
```

42.3.2.3. 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.

42.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/json-serialize-action.kamelet.yaml

CHAPTER 43. 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.

43.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
bootstrapServers *	Brokers	Comma separated list of Kafka Broker URLs	string		
password *	Password	Password to authenticate to kafka	string		
topic *	Topic Names	Comma separated list of Kafka topic names	string		
user *	Username	Username to authenticate to Kafka	string		
saslMechanism	SASL Mechanism	The Simple Authentication and Security Layer (SASL) Mechanism used.	string	**"PLAIN"**	
securityProtocol	Security Protocol	Protocol used to communicate with brokers. SASL_PLAINTEXT, PLAINTEXT, SASL_SSL and SSL are supported	string	**"SASL_SSL"**	

NOTE

Fields marked with an asterisk (*) are mandatory.

43.2. DEPENDENCIES

At runtime, the `kafka-sink Kamelet relies upon the presence of the following dependencies:

- camel:kafka
- camel:kamelet

43.3. USAGE

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

43.3.1. 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"
```

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 **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
```

43.3.1.3. 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.

43.3.2. 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"
```

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 **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
```

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 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.

43.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/kafka-sink.kamelet.yaml

CHAPTER 44. KAFKA SOURCE

Receive data from Kafka topics.

44.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
topic *	Topic Names	Comma separated list of Kafka topic names	string		
bootstrapServers *	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	**"SASL_SSL"**	
saslMechanism	SASL Mechanism	The Simple Authentication and Security Layer (SASL) Mechanism used.	string	**"PLAIN"**	
user *	Username	Username to authenticate to Kafka	string		
password *	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	**true**	
allowManualCommit	Allow Manual Commit	Whether to allow doing manual commits	boolean	**false**	

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	**"latest"**	
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	**"ERROR_HANDLER"**	
deserializeHeaders	Automatically Deserialize Headers	When enabled the Kamelet source will deserialize all message headers to String representation. The default is **false**.	boolean	**true**	
consumerGroup	Consumer Group	A string that uniquely identifies the group of consumers to which this source belongs	string		"my-group-id"

NOTE

Fields marked with an asterisk (*) are mandatory.

44.2. DEPENDENCIES

At runtime, the `kafka-source Kamelet relies upon the presence of the following dependencies:

- camel:kafka
- camel:kamelet
- camel:core

44.3. USAGE

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

44.3.1. 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
```

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 **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
```

44.3.1.3. 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.

44.3.2. 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
```

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 **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
```

44.3.2.3. 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.

44.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/kafka-source.kamelet.yaml

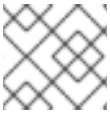
CHAPTER 45. KAFKA TOPIC NAME MATCHES FILTER ACTION

Filter based on kafka topic value compared to regex

45.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **topic-name-matches-filter-action** Kamelet:

Property	Name	Description	Type	Default	Example
regex *	Regex	The Regex to Evaluate against the Kafka topic name	string		



NOTE

Fields marked with an asterisk (*) are mandatory.

45.2. DEPENDENCIES

At runtime, the **topic-name-matches-filter-action** Kamelet relies upon the presence of the following dependencies:

- camel:core
- camel:kamelet

45.3. USAGE

This section describes how you can use the **topic-name-matches-filter-action**.

45.3.1. 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
```

45.3.1.1. 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.

45.3.1.2. 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
```

45.3.1.3. 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.

45.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/topic-name-matches-filter-action.kamelet.yaml

CHAPTER 46. LOG SINK

A sink that logs all data that it receives, useful for debugging purposes.

46.1. 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	**false**	
showStreams	Show Streams	Show the stream bodies (they may not be available in following steps)	boolean	**false**	

NOTE

Fields marked with an asterisk (*) are mandatory.

46.2. DEPENDENCIES

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

- camel:kamelet
- camel:log

46.3. USAGE

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

46.3.1. 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
```

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 **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
```

46.3.1.3. 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.

46.3.2. 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
```

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 **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
```

46.3.2.3. 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.

46.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/log-sink.kamelet.yaml

CHAPTER 47. 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"}'

47.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
databaseName *	Database Name	The Database Name we are pointing	string		
password *	Password	The password to use for accessing a secured MariaDB Database	string		
query *	Query	The Query to execute against the MariaDB Database	string		**"INSERT INTO accounts (username,city) VALUES (:#username,:#city)"**
serverName *	Server Name	Server Name for the data source	string		**"localhost"**
username *	Username	The username to use for accessing a secured MariaDB Database	string		
serverPort	Server Port	Server Port for the data source	string	**3306**	

NOTE

Fields marked with an asterisk (*) are mandatory.

47.2. 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

47.3. USAGE

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

47.3.1. 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"
```

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 **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
```

47.3.1.3. 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.

47.3.2. 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"
```

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 **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
```

47.3.2.3. 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.

47.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/mariadb-sink.kamelet.yaml

CHAPTER 48. MASK FIELDS ACTION

Mask fields with a constant value in the message in transit

48.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
fields *	Fields	Comma separated list of fields to mask	string		
replacement *	Replacement	Replacement for the fields to be masked	string		

NOTE

Fields marked with an asterisk (*) are mandatory.

48.2. 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

48.3. USAGE

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

48.3.1. 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
```

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 **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
```

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 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.

48.3.2. 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
```

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 **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
```

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 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.

48.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/mask-field-action.kamelet.yaml

CHAPTER 49. MESSAGE TIMESTAMP ROUTER ACTION

Update the topic field as a function of the original topic name and the record's timestamp field.

49.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **message-timestamp-router-action** Kamelet:

Property	Name	Description	Type	Default	Example
timestampKeys *	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 java.text.SimpleDateFormat.	string	**"yyyyMMdd"**	
timestampKeyFormat	Timestamp Keys Format	Format of the timestamp keys. Possible values are 'timestamp' or any format string for the timestamp that is compatible with java.text.SimpleDateFormat. In case of 'timestamp' the field will be evaluated as milliseconds since 1970, so as a UNIX Timestamp.	string	**"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.

49.2. 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

49.3. USAGE

This section describes how you can use the **message-timestamp-router-action**.

49.3.1. 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
```

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 **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
```

49.3.1.3. 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.

49.3.2. 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
```

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 **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
```

49.3.2.3. 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.

49.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/message-timestamp-router-action.kamelet.yaml

CHAPTER 50. MONGODB SINK

Send documents to MongoDB.

This Kamelet expects a JSON as body.

Properties you can set as headers:

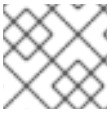
- **db-upsert** / **ce-dbupsert**: if the database should create the element if it does not exist. Boolean value.

50.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
collection *	MongoDB Collection	Sets the name of the MongoDB collection to bind to this endpoint.	string		
database *	MongoDB Database	Sets the name of the MongoDB database to target.	string		
hosts *	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	**false**	
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.

50.2. DEPENDENCIES

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

- camel:kamelet
- camel:mongodb
- camel:jackson

50.3. USAGE

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

50.3.1. 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"
```

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 **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
```

50.3.1.3. 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.

50.3.2. 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"
```

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 **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
```

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 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.

50.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/mongodb-sink.kamelet.yaml

CHAPTER 51. 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.

51.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
collection *	MongoDB Collection	Sets the name of the MongoDB collection to bind to this endpoint.	string		
database *	MongoDB Database	Sets the name of the MongoDB database to target.	string		
hosts *	MongoDB Hosts	Comma separated list of MongoDB Host Addresses in host:port format.	string		
password *	MongoDB Password	User password for accessing MongoDB.	string		
username *	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
persistentTailTracking	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	**false**	
tailTrackIncreasingField	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.

51.2. DEPENDENCIES

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

- camel:kamelet
- camel:mongodb
- camel:jackson

51.3. USAGE

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

51.3.1. 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
```

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 **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
```

51.3.1.3. 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.

51.3.2. 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
```

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 **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
```

51.3.2.3. 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.

51.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/mongodb-source.kamelet.yaml

CHAPTER 52. 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)'

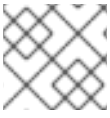
The Kamelet needs to receive as input something like:

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

52.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
databaseName *	Database Name	The Database Name we are pointing	string		
password *	Password	The password to use for accessing a secured MySQL Database	string		
query *	Query	The Query to execute against the MySQL Database	string		**"INSERT INTO accounts (username,city) VALUES (:#username,:#city)"**
serverName *	Server Name	Server Name for the data source	string		**"localhost"**
username *	Username	The username to use for accessing a secured MySQL Database	string		
serverPort	Server Port	Server Port for the data source	string	**3306**	



NOTE

Fields marked with an asterisk (*) are mandatory.

52.2. 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-dbcp2:2.7.0.redhat-00001
- mvn:mysql:mysql-connector-java

52.3. USAGE

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

52.3.1. 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"
```

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 **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
```

52.3.1.3. 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.

52.3.2. 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"
```

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 **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
```

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 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.

52.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/mysql-sink.kamelet.yaml

CHAPTER 53. 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"}'

53.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
databaseName *	Database Name	The Database Name we are pointing	string		
password *	Password	The password to use for accessing a secured PostgreSQL Database	string		
query *	Query	The Query to execute against the PostgreSQL Database	string		**"INSERT INTO accounts (username,city) VALUES (:#username,:#city)"**
serverName *	Server Name	Server Name for the data source	string		**"localhost"**
username *	Username	The username to use for accessing a secured PostgreSQL Database	string		
serverPort	Server Port	Server Port for the data source	string	**5432**	

NOTE

Fields marked with an asterisk (*) are mandatory.

53.2. 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

53.3. USAGE

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

53.3.1. 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"
```

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 **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
```

53.3.1.3. 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.

53.3.2. 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"
```

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 **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
```

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 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.

53.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/postgresql-sink.kamelet.yaml

CHAPTER 54. PREDICATE FILTER ACTION

Filter based on a JsonPath Expression

54.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
expression *	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		**"@.foo =~ /.*John/"**

NOTE

Fields marked with an asterisk (*) are mandatory.

54.2. DEPENDENCIES

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

- camel:core
- camel:kamelet
- camel:jsonpath

54.3. USAGE

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

54.3.1. 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
```

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 **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
```

54.3.1.3. 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.

54.3.2. 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
```

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 **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
```

54.3.2.3. 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.

54.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/predicate-filter-action.kamelet.yaml

CHAPTER 55. PROTOBUF DESERIALIZE ACTION

Deserialize payload to Protobuf

55.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
schema *	Schema	The Protobuf schema to use during serialization (as single-line)	string		**"message Person { required string first = 1; required string last = 2; }"**

NOTE

Fields marked with an asterisk (*) are mandatory.

55.2. 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

55.3. USAGE

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

55.3.1. 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
```

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 **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
```

55.3.1.3. 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.

55.3.2. 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
```

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 **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
```

55.3.2.3. 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.

55.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/protobuf-deserialize-action.kamelet.yaml

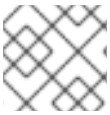
CHAPTER 56. PROTOBUF SERIALIZE ACTION

Serialize payload to Protobuf

56.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
schema *	Schema	The Protobuf schema to use during serialization (as single-line)	string		**"message Person { required string first = 1; required string last = 2; }"**



NOTE

Fields marked with an asterisk (*) are mandatory.

56.2. 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

56.3. USAGE

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

56.3.1. 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
```

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 **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
```

56.3.1.3. 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.

56.3.2. 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
```

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 **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
```

56.3.2.3. 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.

56.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/protobuf-serialize-action.kamelet.yaml

CHAPTER 57. REGEX ROUTER ACTION

Update the destination using the configured regular expression and replacement string

57.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **regex-router-action** Kamelet:

Property	Name	Description	Type	Default	Example
regex *	Regex	Regular Expression for destination	string		
replacement *	Replacement	Replacement when matching	string		

NOTE

Fields marked with an asterisk (*) are mandatory.

57.2. 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

57.3. USAGE

This section describes how you can use the **regex-router-action**.

57.3.1. 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
```

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 **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
```

57.3.1.3. 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.

57.3.2. 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
```

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 **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
```

57.3.2.3. 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.

57.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/regex-router-action.kamelet.yaml

CHAPTER 58. 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.

58.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
renames *	Renames	Comma separated list of field with new value to be renamed	string		**"foo:bar,c1:c2"**
disabled	Disabled	Comma separated list of fields to be disabled	string	"none"	
enabled	Enabled	Comma separated list of fields to be enabled	string	"all"	

NOTE

Fields marked with an asterisk (*) are mandatory.

58.2. 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

58.3. USAGE

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

58.3.1. 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
```

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 **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
```

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 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.

58.3.2. 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
```

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 **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
```

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 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.

58.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/replace-field-action.kamelet.yaml

CHAPTER 59. SALESFORCE SOURCE

Receive updates from Salesforce.

59.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
clientId *	Consumer Key	The Salesforce application consumer key	string		
clientSecret *	Consumer Secret	The Salesforce application consumer secret	string		
password *	Password	The Salesforce user password	string		
query *	Query	The query to execute on Salesforce	string		**"SELECT Id, Name, Email, Phone FROM Contact"**
topicName *	Topic Name	The name of the topic/channel to use	string		**"ContactTopic"**
userName *	Username	The Salesforce username	string		
loginUrl	Login URL	The Salesforce instance login URL	string	**"https://login.salesforce.com"**	

NOTE

Fields marked with an asterisk (*) are mandatory.

59.2. 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

59.3. USAGE

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

59.3.1. 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
```

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 **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
```

59.3.1.3. 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.

59.3.2. 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
```

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 **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
```

59.3.2.3. 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.

59.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/salesforce-source.kamelet.yaml

CHAPTER 60. 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" }

60.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
clientId *	Consumer Key	The Salesforce application consumer key	string		
clientSecret *	Consumer Secret	The Salesforce application consumer secret	string		
password *	Password	The Salesforce user password	string		
userName *	Username	The Salesforce username	string		
loginUrl	Login URL	The Salesforce instance login URL	string	**"https://login.salesforce.com"**	
sObjectName	Object Name	Type of the object	string		**"Contact"**

NOTE

Fields marked with an asterisk (*) are mandatory.

60.2. DEPENDENCIES

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

- camel:salesforce
- camel:kamelet

60.3. USAGE

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

60.3.1. 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"
```

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 **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
```

60.3.1.3. 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.

60.3.2. 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"
```

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 **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
```

60.3.2.3. 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.

60.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/salesforce-create-sink.kamelet.yaml

CHAPTER 61. SALESFORCE DELETE SINK

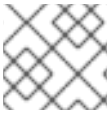
Removes an object from Salesforce. The body received must be a JSON containing two keys: sObjectId and sObjectName.

Example body: { "sObjectId": "XXXXX0", "sObjectName": "Contact" }

61.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
clientId *	Consumer Key	The Salesforce application consumer key	string		
clientSecret *	Consumer Secret	The Salesforce application consumer secret	string		
password *	Password	The Salesforce user password	string		
userName *	Username	The Salesforce username	string		
loginUrl	Login URL	The Salesforce instance login URL	string	**"https://login.salesforce.com"**	



NOTE

Fields marked with an asterisk (*) are mandatory.

61.2. DEPENDENCIES

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

- camel:salesforce
- camel:kamelet
- camel:core
- camel:jsonpath

61.3. USAGE

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

61.3.1. 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"
```

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 **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
```

61.3.1.3. 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.

61.3.2. 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"
```

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 **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
```

61.3.2.3. 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.

61.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/salesforce-delete-sink.kamelet.yaml

CHAPTER 62. 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" }

62.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
clientId *	Consumer Key	The Salesforce application consumer key	string		
clientSecret *	Consumer Secret	The Salesforce application consumer secret	string		
password *	Password	The Salesforce user password	string		
sObjectId *	Object Id	Id of the object. Only required if using key-value pair.	string		
sObjectName *	Object Name	Type of the object. Only required if using key-value pair.	string		**"Contact"**
userName *	Username	The Salesforce username	string		
loginUrl	Login URL	The Salesforce instance login URL	string	**"https://login.salesforce.com"**	

NOTE

Fields marked with an asterisk (*) are mandatory.

62.2. DEPENDENCIES

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

- camel:salesforce

- camel:kamelet

62.3. USAGE

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

62.3.1. 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"
```

62.3.1.1. Prerequisite

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

62.3.1.2. 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
```

62.3.1.3. 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.

62.3.2. 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"
```

62.3.2.1. 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.

62.3.2.2. 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
```

62.3.2.3. 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.

62.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/salesforce-update-sink.kamelet.yaml

CHAPTER 63. 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.

63.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
connectionHost *	Connection Host	Hostname of the FTP server	string		
connectionPort *	Connection Port	Port of the FTP server	string	**22**	
directoryName *	Directory Name	The starting directory	string		
password *	Password	The password to access the FTP server	string		
username *	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	**"Override"**	
passiveMode	Passive Mode	Sets passive mode connection	boolean	**false**	

NOTE

Fields marked with an asterisk (*) are mandatory.

63.2. DEPENDENCIES

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

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

63.3. USAGE

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

63.3.1. 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"
```

63.3.1.1. Prerequisite

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

63.3.1.2. 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
```

63.3.1.3. 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.

63.3.2. 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"
```

63.3.2.1. 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.

63.3.2.2. 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
```

63.3.2.3. 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.

63.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/sftp-sink.kamelet.yaml

CHAPTER 64. SFTP SOURCE

Receive data from an SFTP Server.

64.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
connectionHost *	Connection Host	Hostname of the SFTP server	string		
connectionPort *	Connection Port	Port of the FTP server	string	**22**	
directoryName *	Directory Name	The starting directory	string		
password *	Password	The password to access the SFTP server	string		
username *	Username	The username to access the SFTP server	string		
idempotent	Idempotency	Skip already processed files.	boolean	**true**	
passiveMode	Passive Mode	Sets passive mode connection	boolean	**false**	
recursive	Recursive	If a directory, will look for files in all the sub-directories as well.	boolean	**false**	

NOTE

Fields marked with an asterisk (*) are mandatory.

64.2. DEPENDENCIES

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

- camel:ftp
- camel:core

- camel:kamelet

64.3. USAGE

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

64.3.1. 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
```

64.3.1.1. Prerequisite

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

64.3.1.2. 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
```

64.3.1.3. 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.

64.3.2. 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
```

64.3.2.1. 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.

64.3.2.2. 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
```

64.3.2.3. 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.

64.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/sftp-source.kamelet.yaml

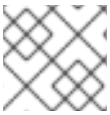
CHAPTER 65. SLACK SOURCE

Receive messages from a Slack channel.

65.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
channel *	Channel	The Slack channel to receive messages from	string		**"#myroom"**
token *	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.

65.2. DEPENDENCIES

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

- camel:kamelet
- camel:slack
- camel:jackson

65.3. USAGE

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

65.3.1. 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
```

65.3.1.1. Prerequisite

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

65.3.1.2. 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
```

65.3.1.3. 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.

65.3.2. 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
```

65.3.2.1. 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.

65.3.2.2. 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
```

65.3.2.3. 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.

65.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/slack-source.kamelet.yaml

CHAPTER 66. 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)'

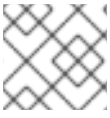
The Kamelet needs to receive as input something like:

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

66.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
databaseName *	Database Name	The Database Name we are pointing	string		
password *	Password	The password to use for accessing a secured SQL Server Database	string		
query *	Query	The Query to execute against the SQL Server Database	string		**"INSERT INTO accounts (username,city) VALUES (:#username,:#city)"**
serverName *	Server Name	Server Name for the data source	string		**"localhost"**
username *	Username	The username to use for accessing a secured SQL Server Database	string		
serverPort	Server Port	Server Port for the data source	string	**1433**	



NOTE

Fields marked with an asterisk (*) are mandatory.

66.2. 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-dbcp2:2.7.0.redhat-00001
- mvn:com.microsoft.sqlserver:mssql-jdbc:9.2.1.jre11

66.3. USAGE

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

66.3.1. 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"
```

66.3.1.1. Prerequisite

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

66.3.1.2. 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
```

66.3.1.3. 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.

66.3.2. 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"
```

66.3.2.1. 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.

66.3.2.2. 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
```

66.3.2.3. 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.

66.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/sqlserver-sink.kamelet.yaml

CHAPTER 67. 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

67.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
authorizationToken *	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.

67.2. DEPENDENCIES

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

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

67.3. USAGE

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

67.3.1. 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
```

67.3.1.1. Prerequisite

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

67.3.1.2. 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
```

67.3.1.3. 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.

67.3.2. 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
```

67.3.2.1. 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.

67.3.2.2. 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
```

67.3.2.3. 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.

67.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/telegram-source.kamelet.yaml

CHAPTER 68. THROTTLE ACTION

The Throttle action allows you to ensure that a specific sink does not get overloaded.

68.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
messages *	Messages Number	The number of messages to send in the time period set	integer		**10**
timePeriod	Time Period	Sets the time period during which the maximum request count is valid for, in milliseconds	string	**"1000"**	

NOTE

Fields marked with an asterisk (*) are mandatory.

68.2. DEPENDENCIES

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

- camel:core
- camel:kamelet

68.3. USAGE

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

68.3.1. 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
```

68.3.1.1. Prerequisite

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

68.3.1.2. 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
```

68.3.1.3. 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.

68.3.2. 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
```

68.3.2.1. 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.

68.3.2.2. 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
```

68.3.2.3. 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.

68.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/throttle-action.kamelet.yaml

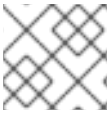
CHAPTER 69. TIMER SOURCE

Produces periodic events with a custom payload.

69.1. CONFIGURATION OPTIONS

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

Property	Name	Description	Type	Default	Example
message *	Message	The message to generate	string		**"hello world"**
contentType	Content Type	The content type of the message being generated	string	**"text/plain"**	
period	Period	The interval between two events in milliseconds	integer	**1000**	
repeatCount	Repeat Count	Specifies the maximum limit of the number of fires	integer		



NOTE

Fields marked with an asterisk (*) are mandatory.

69.2. DEPENDENCIES

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

- camel:core
- camel:timer
- camel:kamelet

69.3. USAGE

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

69.3.1. 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
```

69.3.1.1. Prerequisite

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

69.3.1.2. 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
```

69.3.1.3. 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.

69.3.2. 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
```

69.3.2.1. 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.

69.3.2.2. 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
```

69.3.2.3. 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.

69.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/timer-source.kamelet.yaml

CHAPTER 70. TIMESTAMP ROUTER ACTION

Update the topic field as a function of the original topic name and the record timestamp.

70.1. 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 java.text.SimpleDateFormat.	string	**"yyyyMMdd"**	
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.

70.2. 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

70.3. USAGE

This section describes how you can use the **timestamp-router-action**.

70.3.1. 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
```

70.3.1.1. Prerequisite

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

70.3.1.2. 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
```

70.3.1.3. 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.

70.3.2. 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
```

70.3.2.1. 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.

70.3.2.2. 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
```

70.3.2.3. 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.

70.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/timestamp-router-action.kamelet.yaml

CHAPTER 71. VALUE TO KEY ACTION

Replace the Kafka record key with a new key formed from a subset of fields in the body

71.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **value-to-key-action** Kamelet:

Property	Name	Description	Type	Default	Example
fields *	Fields	Comma separated list of fields to be used to form the new key	string		

NOTE

Fields marked with an asterisk (*) are mandatory.

71.2. 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

71.3. USAGE

This section describes how you can use the **value-to-key-action**.

71.3.1. 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
```

71.3.1.1. Prerequisite

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

71.3.1.2. 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
```

71.3.1.3. 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.

71.3.2. 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
```

71.3.2.1. 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.

71.3.2.2. 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
```

71.3.2.3. 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.

71.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/value-to-key-action.kamelet.yaml