Operator	Flowable	Observable	Maybe	Single	Completable
all	✅	✅	⭕ (1)	⭕ (1)	⭕ (2)
amb	✅	✅	✅	✅	✅
ambArray	✅	✅	✅	✅	✅
ambWith	✅	✅	✅	✅	✅
andThen	⭕ (3)	⭕ (3)	⭕ (3)	⭕ (3)	✅
any	✅	✅	⭕ (1)	⭕ (1)	⭕ (2)
blockingAwait	⭕ (4)	⭕ (4)	⭕ (5)	⭕ (5)	✅
blockingFirst	✅	✅	⭕ (6)	⭕ (6)	⭕ (7)
blockingForEach	✅	✅	⭕ (8)	⭕ (8)	⭕ (8)
blockingGet	⭕ (4)	⭕ (4)	✅	✅	⭕ (7)
blockingIterable	✅	✅	⭕ (6)	⭕ (6)	⭕ (7)
blockingLast	✅	✅	⭕ (6)	⭕ (6)	⭕ (7)

blockingLatest	✅	✅	⭕ (6)	⭕ (6)	⭕ (7)
blockingMostRecent	✅	✅	⭕ (6)	⭕ (6)	⭕ (7)
blockingNext	✅	✅	⭕ (6)	⭕ (6)	⭕ (7)
blockingSingle	✅	✅	⭕ (6)	⭕ (6)	⭕ (7)
blockingStream	✅	✅	⭕ (6)	⭕ (6)	⭕ (7)
blockingSubscribe	✅	✅	✅	✅	✅
buffer	✅	✅	⭕ (9)	⭕ (10)	⭕ (11)
cache	✅	✅	✅	✅	✅
cacheWithInitialCapacity	✅	✅	⭕ (12)	⭕ (12)	⭕ (12)
cast	✅	✅	✅	✅	⭕ (2)
collect	✅	✅	⭕ (13)	⭕ (14)	⭕ (15)
collectInto	✅	✅	⭕ (13)	⭕ (14)	⭕ (15)
combineLatest	✅	✅	⭕ (16)	⭕ (16)	⭕ (17)
combineLatestArray	✅	✅	⭕ (18)	⭕ (18)	⭕ (19)
combineLatestArrayDelayError					

	✅	✅	⭕ (18)	⭕ (18)	⭕ (20)
combineLatestDelayError	✅	✅	⭕ (16)	⭕ (16)	⭕ (21)
complete	⭕ (22)	⭕ (22)	⭕ (22)	⭕ (23)	✅
compose	✅	✅	✅	✅	✅
concat	✅	✅	✅	✅	✅
concatArray	✅	✅	✅	✅	✅
concatArrayDelayError	✅	✅	✅	✅	✅
concatArrayEager	✅	✅	✅	✅	⭕ (24)
concatArrayEagerDelayError	✅	✅	✅	✅	⭕ (25)
concatDelayError	✅	✅	✅	✅	✅
concatEager	✅	✅	✅	✅	⭕ (26)
concatEagerDelayError	✅	✅	✅	✅	⭕ (27)
concatMap	✅	✅	✅	✅	⭕ (28)
concatMapCompletable	✅	✅	✅	✅	⭕ (28)
concatMapCompletableDelayError	✅	✅	⭕ (29)	⭕ (29)	⭕ (28)

concatMapDelayError	✅	✅	⭕ (30)	⭕ (30)	⭕ (28)
concatMapEager	✅	✅	⭕ (31)	⭕ (31)	⭕ (28)
concatMapEagerDelayError	✅	✅	⭕ (31)	⭕ (31)	⭕ (28)
concatMapIterable	✅	✅	⭕ (32)	⭕ (32)	⭕ (28)
concatMapMaybe	✅	✅	⭕ (33)	✅	⭕ (28)
concatMapMaybeDelayError	✅	✅	⭕ (34)	⭕ (34)	⭕ (28)
concatMapSingle	✅	✅	✅	⭕ (35)	⭕ (28)
concatMapSingleDelayError	✅	✅	⭕ (36)	⭕ (36)	⭕ (28)
concatMapStream	✅	✅	⭕ (37)	⭕ (37)	⭕ (28)
concatWith	✅	✅	✅	✅	✅
contains	✅	✅	✅	✅	⭕ (2)
count	✅	✅	✅	⭕ (38)	⭕ (39)
create	✅	✅	✅	✅	✅
debounce	✅	✅	⭕ (40)	⭕ (40)	⭕ (41)
defaultIfEmpty	✅	✅	✅	⭕ (23)	⭕ (42)

defer	✅	✅	✅	✅	✅
delay	✅	✅	✅	✅	✅
delaySubscription	✅	✅	✅	✅	✅
dematerialize	✅	✅	✅	✅	⭕ (41)
distinct	✅	✅	⭕ (43)	⭕ (43)	⭕ (41)
distinctUntilChanged	✅	✅	⭕ (43)	⭕ (43)	⭕ (41)
doAfterNext	✅	✅	⭕ (44)	⭕ (44)	⭕ (2)
doAfterSuccess	⭕ (45)	⭕ (45)	✅	✅	⭕ (41)
doAfterTerminate	✅	✅	✅	✅	✅
doFinally	✅	✅	✅	✅	✅
doOnCancel	✅	⭕ (46)	⭕ (46)	⭕ (46)	⭕ (46)
doOnComplete	✅	✅	✅	⭕ (47)	✅
doOnDispose	⭕ (48)	✅	✅	✅	✅
doOnEach	✅	✅	⭕ (49)	⭕ (49)	⭕ (41)
doOnError					

	✅	✅	✅	✅	✅
doOnEvent	⭕ (50)	⭕ (50)	✅	✅	✅
doOnLifecycle	✅	✅	✅	✅	✅
doOnNext	✅	✅	⭕ (51)	⭕ (51)	⭕ (41)
doOnRequest	✅	⭕ (52)	⭕ (52)	⭕ (52)	⭕ (52)
doOnSubscribe	✅	✅	✅	✅	✅
doOnSuccess	⭕ (53)	⭕ (53)	✅	✅	⭕ (41)
doOnTerminate	✅	✅	✅	✅	✅
elementAt	✅	✅	⭕ (54)	⭕ (55)	⭕ (41)
elementAtOrError	✅	✅	⭕ (56)	⭕ (55)	⭕ (41)
empty	✅	✅	✅	⭕ (23)	⭕ (57)
error	✅	✅	✅	✅	✅
filter	✅	✅	✅	✅	⭕ (41)
first	✅	✅	⭕ (58)	⭕ (59)	⭕ (42)
firstElement	✅	✅	⭕ (60)	⭕ (61)	⭕ (2)

`firstOrError`	✅	✅	⭕ (60)	⭕ (61)	⭕ (62)
`firstOrErrorStage`	✅	✅	⭕ (63)	⭕ (63)	⭕ (64)
`firstStage`	✅	✅	⭕ (63)	⭕ (63)	⭕ (63)
`flatMap`	✅	✅	✅	✅	⭕ (28)
`flatMapCompletable`	✅	✅	✅	✅	⭕ (28)
`flatMapIterable`	✅	✅	⭕ (32)	⭕ (32)	⭕ (28)
`flatMapMaybe`	✅	✅	⭕ (65)	✅	⭕ (28)
`flatMapObservable`	⭕ (66)	⭕ (67)	✅	✅	⭕ (28)
`flatMapPublisher`	⭕ (67)	⭕ (68)	✅	✅	⭕ (28)
`flatMapSingle`	✅	✅	✅	⭕ (65)	⭕ (28)
`flatMapStream`	✅	✅	⭕ (37)	⭕ (37)	⭕ (28)
`flattenAsFlowable`	⭕ (69)	⭕ (69)	✅	✅	⭕ (28)
`flattenAsObservable`	⭕ (69)	⭕ (69)	✅	✅	⭕ (28)
`flattenStreamAsFlowable`	⭕ (70)	⭕ (70)	✅	✅	⭕ (28)
`flattenStreamAsObservable`	⭕ (70)	⭕ (70)	✅	✅	⭕ (28)

forEach	✅	✅	⭕ (71)	⭕ (71)	⭕ (71)
forEachWhile	✅	✅	⭕ (71)	⭕ (71)	⭕ (71)
fromAction	✅	✅	✅	⭕ (23)	✅
fromArray	✅	✅	⭕ (72)	⭕ (73)	⭕ (74)
fromCallable	✅	✅	✅	✅	✅
fromCompletable	✅	✅	✅	⭕ (75)	⭕ (76)
fromCompletionStage	✅	✅	✅	✅	✅
fromFuture	✅	✅	✅	✅	✅
fromIterable	✅	✅	⭕ (72)	⭕ (73)	⭕ (74)
fromMaybe	✅	✅	⭕ (76)	✅	✅
fromObservable	✅	⭕ (76)	✅	✅	✅
fromOptional	✅	✅	✅	⭕ (73)	⭕ (74)
fromPublisher	✅	✅	✅	✅	✅
fromRunnable	✅	✅	✅	⭕ (23)	✅
fromSingle					

	✅	✅	✅	⭕ (76)	✅
`fromStream`	✅	✅	⭕ (72)	⭕ (73)	⭕ (74)
`fromSupplier`	✅	✅	✅	✅	✅
`generate`	✅	✅	⭕ (77)	⭕ (77)	⭕ (77)
`groupBy`	✅	✅	⭕ (78)	⭕ (78)	⭕ (79)
`groupJoin`	✅	✅	⭕ (78)	⭕ (78)	⭕ (80)
`hide`	✅	✅	✅	✅	✅
`ignoreElement`	⭕ (81)	⭕ (81)	✅	✅	⭕ (2)
`ignoreElements`	✅	✅	⭕ (82)	⭕ (82)	⭕ (2)
`interval`	✅	✅	⭕ (83)	⭕ (83)	⭕ (83)
`intervalRange`	✅	✅	⭕ (83)	⭕ (83)	⭕ (83)
`isEmpty`	✅	✅	✅	⭕ (59)	⭕ (2)
`join`	✅	✅	⭕ (84)	⭕ (84)	⭕ (80)
`just`	✅	✅	✅	✅	⭕ (2)
`last`	✅	✅	⭕ (58)	⭕ (59)	⭕ (42)

`lastElement`	✅	✅	⭕ (60)	⭕ (61)	⭕ (2)
`lastOrError`	✅	✅	⭕ (60)	⭕ (61)	⭕ (62)
`lastOrErrorStage`	✅	✅	⭕ (63)	⭕ (63)	⭕ (64)
`lastStage`	✅	✅	⭕ (63)	⭕ (63)	⭕ (63)
`lift`	✅	✅	✅	✅	✅
`map`	✅	✅	✅	✅	⭕ (28)
`mapOptional`	✅	✅	✅	✅	⭕ (28)
`materialize`	✅	✅	✅	✅	✅
`merge`	✅	✅	✅	✅	✅
`mergeArray`	✅	✅	✅	✅	✅
`mergeArrayDelayError`	✅	✅	✅	✅	✅
`mergeDelayError`	✅	✅	✅	✅	✅
`mergeWith`	✅	✅	✅	✅	✅
`never`	✅	✅	✅	✅	✅
`observeOn`	✅	✅	✅	✅	✅

ofType	✅	✅	✅	✅	⭕ (85)
onBackpressureBuffer	✅	⭕ (52)	⭕ (52)	⭕ (52)	⭕ (52)
onBackpressureDrop	✅	⭕ (52)	⭕ (52)	⭕ (52)	⭕ (52)
onBackpressureLatest	✅	⭕ (52)	⭕ (52)	⭕ (52)	⭕ (52)
onErrorComplete	✅	✅	✅	✅	✅
onErrorResumeNext	✅	✅	✅	✅	✅
onErrorResumeWith	✅	✅	✅	✅	✅
onErrorReturn	✅	✅	✅	✅	✅
onErrorReturnItem	✅	✅	✅	✅	✅
onTerminateDetach	✅	✅	✅	✅	✅
parallel	✅	⭕ (86)	⭕ (86)	⭕ (86)	⭕ (86)
publish	✅	✅	⭕ (87)	⭕ (88)	⭕ (89)
range	✅	✅	⭕ (90)	⭕ (90)	⭕ (74)
rangeLong	✅	✅	⭕ (90)	⭕ (90)	⭕ (74)
rebatchRequests					

	✅	⭕ (52)	⭕ (52)	⭕ (52)	⭕ (52)
reduce	✅	✅	⭕ (91)	⭕ (91)	⭕ (92)
reduceWith	✅	✅	⭕ (91)	⭕ (91)	⭕ (92)
repeat	✅	✅	✅	✅	✅
repeatUntil	✅	✅	✅	✅	✅
repeatWhen	✅	✅	✅	✅	✅
replay	✅	✅	⭕ (87)	⭕ (88)	⭕ (89)
retry	✅	✅	✅	✅	✅
retryUntil	✅	✅	✅	✅	✅
retryWhen	✅	✅	✅	✅	✅
safeSubscribe	✅	✅	✅	✅	✅
sample	✅	✅	⭕ (60)	⭕ (60)	⭕ (41)
scan	✅	✅	⭕ (91)	⭕ (91)	⭕ (92)
scanWith	✅	✅	⭕ (91)	⭕ (91)	⭕ (92)
sequenceEqual	✅	✅	✅	✅	✅

serialize	✓	✓	○ (93)	○ (93)	○ (93)
share	✓	✓	○ (87)	○ (88)	○ (89)
single	✓	✓	○ (58)	○ (59)	○ (42)
singleElement	✓	✓	○ (60)	○ (61)	○ (2)
singleOrError	✓	✓	○ (60)	○ (61)	○ (62)
singleOrErrorStage	✓	✓	○ (63)	○ (63)	○ (64)
singleStage	✓	✓	○ (63)	○ (63)	○ (63)
skip	✓	✓	○ (60)	○ (60)	○ (60)
skipLast	✓	✓	○ (60)	○ (60)	○ (60)
skipUntil	✓	✓	○ (94)	○ (94)	○ (94)
skipWhile	✓	✓	○ (95)	○ (95)	○ (2)
sorted	✓	✓	○ (78)	○ (78)	○ (78)
startWith	✓	✓	✓	✓	✓
startWithArray	✓	✓	○ (96)	○ (96)	○ (96)
startWithItem	✓	✓	○ (97)	○ (97)	○ (97)

startWithIterable	✅	✅	⭕ (98)	⭕ (98)	⭕ (98)
subscribe	✅	✅	✅	✅	✅
subscribeOn	✅	✅	✅	✅	✅
subscribeWith	✅	✅	✅	✅	✅
switchIfEmpty	✅	✅	✅	⭕ (23)	⭕ (99)
switchMap	✅	✅	⭕ (100)	⭕ (100)	⭕ (28)
switchMapCompletable	✅	✅	⭕ (100)	⭕ (100)	⭕ (28)
switchMapCompletableDelayError	✅	✅	⭕ (100)	⭕ (100)	⭕ (28)
switchMapDelayError	✅	✅	⭕ (100)	⭕ (100)	⭕ (28)
switchMapMaybe	✅	✅	⭕ (100)	⭕ (100)	⭕ (28)
switchMapMaybeDelayError	✅	✅	⭕ (100)	⭕ (100)	⭕ (28)
switchMapSingle	✅	✅	⭕ (100)	⭕ (100)	⭕ (28)
switchMapSingleDelayError	✅	✅	⭕ (100)	⭕ (100)	⭕ (28)
switchOnNext	✅	✅	✅	✅	✅
switchOnNextDelayError					

	✅	✅	✅	✅	✅
take	✅	✅	⭕ (60)	⭕ (60)	⭕ (60)
takeLast	✅	✅	⭕ (60)	⭕ (60)	⭕ (60)
takeUntil	✅	✅	✅	✅	✅
takeWhile	✅	✅	⭕ (95)	⭕ (95)	⭕ (2)
test	✅	✅	✅	✅	✅
throttleFirst	✅	✅	⭕ (40)	⭕ (40)	⭕ (41)
throttleLast	✅	✅	⭕ (40)	⭕ (40)	⭕ (41)
throttleLatest	✅	✅	⭕ (40)	⭕ (40)	⭕ (41)
throttleWithTimeout	✅	✅	⭕ (40)	⭕ (40)	⭕ (41)
timeInterval	✅	✅	✅	✅	⭕ (41)
timeout	✅	✅	✅	✅	✅
timer	✅	✅	✅	✅	✅
timestamp	✅	✅	✅	✅	⭕ (41)
to	✅	✅	✅	✅	✅

toCompletionStage	○ (101)	○ (101)	✓	✓	✓
toFlowable	○ (102)	✓	✓	✓	✓
toFuture	✓	✓	✓	✓	✓
toList	✓	✓	○ (13)	○ (14)	○ (15)
toMap	✓	✓	○ (13)	○ (14)	○ (15)
toMaybe	○ (103)	○ (103)	○ (102)	✓	✓
toMultimap	✓	✓	○ (13)	○ (14)	○ (15)
toObservable	✓	○ (102)	✓	✓	✓
toSingle	○ (104)	○ (104)	✓	○ (102)	✓
toSingleDefault	○ (105)	○ (105)	○ (106)	○ (102)	✓
toSortedList	✓	✓	○ (13)	○ (14)	○ (15)
unsafeCreate	✓	✓	✓	✓	✓
unsubscribeOn	✓	✓	✓	✓	✓
using	✓	✓	✓	✓	✓
window	✓	✓	○ (107)	○ (108)	○ (109)

`withLatestFrom`	✅	✅	⭕ (16)	⭕ (16)	⭕ (17)
`wrap`	⭕ (110)	✅	✅	✅	✅
`zip`	✅	✅	✅	✅	⭕ (111)
`zipArray`	✅	✅	✅	✅	⭕ (112)
`zipWith`	✅	✅	✅	✅	⭕ (113)
237 operators	**216**	**210**	**118**	**108**	**84**

Notes

[1] Use `contains()`.

[2] Always empty.

[3] Use `concatWith`.

[4] Use `blockingFirst()`, `blockingSingle()` or `blockingLast()`.

[5] Use `blockingGet()`.

[6] At most one element to get. Use `blockingGet()`.

[7] No elements to get. Use `blockingAwait()`.

[8] Use `blockingSubscribe()`

[9] Use `map()` and `switchIfEmpty()` to transform into a list/collection.

[10] Use `map()` to transform into a list/collection.

[11] Always empty. Use `andThen()` to bring in a list/collection.

[12] At most one element to store. Use `cache()`.

[13] At most one element to collect. Use `map()` and `switchIfEmpty()` to transform into a list/collection.

[14] One element to collect. Use `map()` to transform into a list/collection.

[15] Always empty. Use `andThen()` to bring in a collection.

[16] At most one element per source. Use `zip()`.

[17] Always empty. Use `merge()`.

[18] At most one element per source. Use `zipArray()`.

[19] Always empty. Use `mergeArray()`.

[20] Always empty. Use `mergeArrayDelayError()`.

[21] Always empty. Use `mergeDelayError()`.

[22] Use `empty()`.

[23] Never empty.

[24] No items to keep ordered. Use `mergeArray()`.

[25] No items to keep ordered. Use `mergeArrayDelayError()`.

[26] No items to keep ordered. Use `merge()`.

[27] No items to keep ordered. Use `mergeDelayError()`.

[28] Always empty thus no items to map.

[29] Either the upstream fails (thus no inner) or the mapped-in source, but never both. Use `concatMapCompletable`.

[30] Either the upstream fails (thus no inner) or the mapped-in source, but never both. Use `concatMap`.

[31] At most one item to map. Use `concatMap()`.

[32] At most one item. Use `flattenAsFlowable` or `flattenAsObservable`.

[33] Use `concatMap`.

[34] Either the upstream fails (thus no inner) or the mapped-in source, but never both. Use `concatMapMaybe`.

[35] Use `concatMap()`.

[36] Either the upstream fails (thus no inner) or the mapped-in source, but never both. Use `concatMapSingle`.

[37] At most one item. Use `flattenStreamAsFlowable` or `flattenStreamAsObservable`.

[38] Never empty thus always 1.

[39] Always empty thus always 0.

[40] At most one item signaled so no subsequent items to work with.

[41] Always empty thus no items to work with.

[42] Always empty. Use `andThen()` to chose the follow-up sequence.

[43] At most one item, always distinct.

[44] Different terminology. Use `doAfterSuccess()`.

[45] Different terminology. Use `doAfterNext()`.

[46] Different terminology. Use `doOnDispose()`.

[47] Always succeeds or fails, there is no `onComplete` signal.

[48] Different terminology. Use `doOnCancel()`.

[49] At most one item. Use `doOnEvent()`.

[50] Use `doOnEach()`.

[51] Different terminology. Use `doOnSuccess()`.

[52] Backpressure related and not supported outside `Flowable`.

[53] Different terminology. Use `doOnNext()`.

[54] At most one item with index 0. Use `defaultIfEmpty`.

[55] Always one item with index 0.

[56] At most one item with index 0. Use `toSingle`.

[57] Use `complete()`.

[58] At most one item. Use `defaultIfEmpty`.

[59] Always one item.

[60] At most one item, would be no-op.

[61] Always one item, would be no-op.

[62] Always empty. Use `andThen()` and `error()`.

[63] At most one item. Use `toCompletionStage()`.

[64] Always empty. Use `andThen()`, `error()` and `toCompletionStage()`.

[65] Use `flatMap()`.

[66] Not supported. Use `flatMap` and `toFlowable()`.

[67] Use `flatMap`.

[68] Not supported. Use `flatMap` and `toObservable()`.

[69] Use `flatMapIterable()`.

[70] Use `flatMapStream()`.

[71] Use `subscribe()`.

[72] At most one item. Use `just()` or `empty()`.

[73] Always one item. Use `just()`.

[74] Always empty. Use `complete()`.

[75] Always error.

[76] Use `wrap()`.

[77] Use `fromSupplier()`.

[78] At most one item.

[79] Always empty thus no items to group.

[80] Always empty thus no items to join.

[81] Use `ignoreElements()`.

[82] Use `ignoreElement()`.

[83] At most one item. Use `timer()`.

[84] At most one item. Use `zip()`

[85] Always empty thus no items to filter.

[86] Needs backpressure thus not supported outside `Flowable`.

[87] Connectable sources not supported outside `Flowable` and `Observable`. Use a `MaybeSubject`.

[88] Connectable sources not supported outside `Flowable` and `Observable`. Use a `SingleSubject`.

[89] Connectable sources not supported outside `Flowable` and `Observable`. Use a `ConnectableSubject`.

[90] At most one item. Use `just()`.

[91] At most one item. Use `map()`.

[92] Always empty thus no items to reduce.

[93] At most one signal type.

[94] At most one item. Use `takeUntil()`.

[95] At most one item. Use `filter()`.

[96] Use `startWith()` and `fromArray()` of `Flowable` or `Observable`.

[97] Use `startWith()` and `just()` of another reactive type.

[98] Use `startWith()` and `fromIterable()` of `Flowable` or `Observable`.

[99] Always empty. Use `defaultIfEmpty()`.

[100] At most one item. Use `flatMap()`.

[101] Use `firstStage`, `lastStage` or `singleStage`.

[102] Would be no-op.

[103] Use `firstElement`, `lastElement` or `singleElement`.

[104] Use `firstOrError`, `lastOrError` or `singleOrError`.

[105] Use `first`, `last` or `single`.

[106] Use `defaultIfEmpty()`.

[107] Use `map()` and `switchIfEmpty()` to transform into a nested source.

[108] Use `map()` to transform into a nested source.

[109] Always empty. Use `andThen()` to bring in a nested source.

[110] Use `fromPublisher()`.

[111] Use `merge()`.

[112] Use `mergeArray()`.

[113] Use `mergeWith()`.

Under development

Currently, all intended operators are implemented.