カート機能導入に伴うAPIへの影響

1. はじめに

メルカリShopsでは、複数商品を同時に購入できるカート機能の導入を予定しています。

1.1 カート機能により可能になること

- 複数商品の一括受注: 複数の異なる商品または同一商品複数個を1つの注文として受注
- 分割発送: 複数商品を任意のタイミングや組み合わせで分けて発送
- 部分キャンセル・返金: 注文内の一部商品のみをキャンセル・返金
- 送料割引の適用: 複数商品をまとめてカート購入した際に送料割引を適用

この機能により、購入者は複数の商品をまとめて購入し、出品者は送料割引、効率的な発送や部分的なキャンセル対応が可能になります。

1.2 APIへの影響

現在提供中の注文を操作するためのAPI（旧API群）は、1商品1個の注文処理を前提として設計されており、複数商品の注文や分割発送に対応できません。そのため、カート機能の実現に向けて新API群の提供を開始いたします。

新API群は2025年10月より提供開始いたします。カート機能は段階的にリリースされます:

リリース順	フェーズ名	予定時期	リリース内容
1	複数個フェーズ	2026年2月予定	同一商品を複数個購入可能になります
2	複数商品フェーズ	2026年3月予定	異なる商品を複数購入可能になります
3	旧API群廃止フェーズ	時期未定	旧API群を廃止します

※ 上記の予定時期は開発状況により変更される可能性があります。正確な提供開始時期については、別途アナウンスいたします。

旧API群はカート機能のリリース後、一定の移行期間の後に廃止予定です。注文管理API（注文の参照、発送、キャンセル、購入者とのコミュニケーション）をご利用の方は、新API群への移行をお願いいたします。商品出品など注文管理以外のAPIのみをご利用の場合は、移行の必要はありません。

2. 変更点概要

以下では、移行に伴う主要な変更点について説明いたします。

「今後追加予定」としている機能については、それぞれの提供開始時期に改めて詳細なアナウンスを行います

2.1 複数商品・複数個の商品注文を表すリソースを追加

現在の注文を表すOrderは複数商品・複数個の商品を表現することができません。そのため、カート機能による注文を表現できるOrderTransactionリソースを追加します。既存のOrderは廃止予定ですが、既存(カート機能リリース以前)の注文についてはOrderTransactionとして引き続き取得することが可能です。

注文を表すリソースがOrderTransactionに変わるため、キャンセルや取引メッセージ送信などのAPIもOrderTransactionに合わせたものに変更されます。また、OrderTransactionを対象にした新しいWebhookも追加されます。

2.2 発送フローの変更

completeOrderが廃止になります。

複数商品の分割発送に対応するため、従来の一段階の発送フローから二段階の発送フローに変更されます:

1. createOrderShipping: 発送対象の商品を選択
2. completeOrderShipping: 発送完了と売上反映

分割発送を行う場合は、未発送の商品がなくなるまで上記の発送プロセスを繰り返します。商品の分割発送を管理するOrderShippingリソースが新たに追加されます。

2.3 部分キャンセル・返金API（今後追加予定）

複数商品の注文において、一部の商品のみをキャンセル・返金できる部分キャンセル用のAPIを今後追加予定です。キャンセル対象の商品だけを指定してキャンセルと返金をすることができるようになります。

なお、一部の注文では部分キャンセルができないケースがあります。具体的な条件については、部分キャンセル用のAPIを公開する際に詳細をアナウンスいたします。

2.4 送料割引設定用API（今後追加予定）

送料割引の適用条件と割引額をAPIで設定できるようにします。複数個購入時の送料割引ルールを柔軟に設定可能になります。送料割引を設定しない場合、デフォルトで割引はされません。

2.5 テスト注文用API（今後追加予定）

現在提供しているdebugCreateOrder APIは、単一商品の注文のみ作成可能です。そのため、複数商品の注文にも対応できる新しいテスト注文APIを追加予定です。

2.6 一部APIの廃止

新API群への移行に伴い、一定の移行期間の後に一部のAPIは廃止される予定です。具体的な廃止対象APIについては、次章の影響範囲一覧で詳しく説明します。

3. 影響範囲一覧

複数商品フェーズ（異なる商品を複数購入可能）では、既存APIへの新たな影響はありません。影響があるのは複数個フェーズ（同一商品を複数個購入可能）と旧API群廃止フェーズとなります。

以下の表は、各APIがいつ、どのような影響を受けるかをまとめたものです。ご利用中のAPIを確認し、システムへの影響範囲をご確認ください。

旧**API**群	複数個フェーズでの影響	廃止時期	必要なアクション
query order/orders（type Order）	一部機能が制限されます	旧API群廃止フェーズ	orderTransactionへの移行
mutation cancelOrder	廃止されます	複数個フェーズ	cancelOrderTransactionへの移行 部分キャンセル用API（今後提供予定）への移行
mutation completeOrder	商品1個に対する発送に限定されます	旧API群廃止フェーズ	orderShippingへの移行
mutation updateShippingTrackingCode	商品1個に対する追跡情報の更新に限定されます COMPLETEDステータスのOrderに対する処理に限定されます	旧API群廃止フェーズ	updateOrderShippingTrackingCodeへの移行
mutation addTransactionMessage	廃止されます	複数個フェーズ	addOrderTransactionMessageへの移行
webhook order_created	1回の注文で複数回送信されることがあります	旧API群廃止フェーズ	order_transaction_createdへの移行
webhook order_paid	1回の注文で複数回送信されることがあります	旧API群廃止フェーズ	order_transaction_paidへの移行
webhook order_canceled	1回の注文で複数回送信されることがあります	旧API群廃止フェーズ	order_transaction_canceledへの移行
webhook transactionmessage_created	廃止されます	複数個フェーズ	order_transaction_message_createdへの移行
mutation debugCreateOrder	-	新テスト注文APIリリース時	新テスト注文API（今後提供予定）への移行

4. 影響詳細

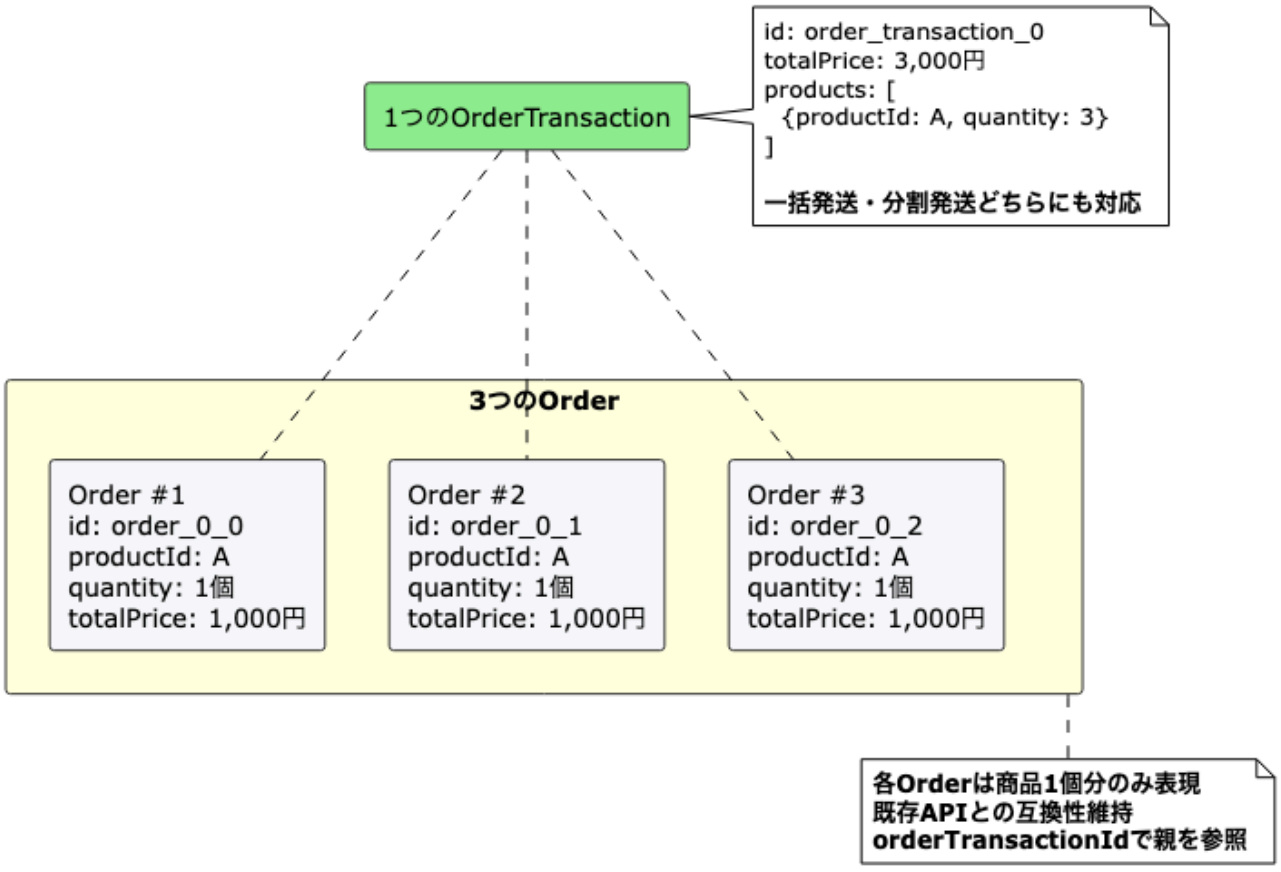
4.1 query order/orders（type Order）

4.1.1 複数個フェーズでの変更点

OrderとOrderTransactionの関係性について

既存のOrderリソースでは、カート機能による複数商品・複数個の注文を表現することができません。そのため、カート機能による注文を表現するリソースとして、OrderTransactionが新たに追加されます。

複数個フェーズ以降に注文が発生すると、1つのOrderTransactionと複数の（商品個数分の）のOrderが作成されます。例えば、商品A×3個の注文が発生した場合、以下のような構造になります：



なお、複数商品の注文（例：商品Aが2個と商品Bが1個）でも同様に、OrderTransactionは1個、Orderは商品の個数分（この場合は3個）作成されます。

OrderTransactionIdとOrderIdは異なる値になります。同一の値になる場合もありますが、これは保証されません。

API操作の対象について

Orderリソースに対しては旧API群で操作可能です（一部制限あり）。OrderTransactionに対しては新API群で操作をする必要があります。

例えば、あるOrderに対してcompleteOrderを実行すると、OrderTransaction側では商品1個が発送された状態となります。このように、旧API群でOrderを操作した結果は、OrderTransactionにも反映されます。

フィールドの制限について

複数個フェーズ以降、Orderの一部フィールドに制約が発生します。

フィールド名	制約
buyerShippingFee	送料割引が発生した場合、常に0となります。商品の個体に紐づく送料を確定できないためです。送料が割引されておらず、購入者負担の場合は従来通り値が入ります。
messages	常にnullとなります。取引メッセージはOrderTransactionレベルで管理されるようになり、個別のOrderには含まれません。
orderCoupon	クーポンが使用されている場合でも、常にnullとなります。クーポン情報はOrderTransactionレベルで管理されます。
updatedAt	常にゼロ値となります。

ordersクエリの制限について

複数個フェーズ以降、ordersクエリの下記引数が利用できなくなります。後方互換性のためパラメータとして指定してもエラーにはなりませんが、指定された値は無視されます。

- updatedDateGte, updatedDateLt
- canceled, completed
- keyword
- statuses

新しいフィールドの追加

今後、親OrderTransactionを示すOrderTransactionIdフィールドが新たに追加されます。これにより、各OrderからOrderTransactionを参照することが可能になります。

4.1.2 廃止・移行について

複数個フェーズにおいて、query order/ordersとtype Orderは商品1個を扱うリソースとなり、一部フィールドが制限されます。代替として、以下のAPIへの移行を推奨します:

- query orderTransaction / orderTransactions: 複数商品・複数個の注文情報取得
- type OrderTransaction: 複数商品・複数個の注文を表現するリソース

旧API群廃止フェーズにおいて、query order/ordersとtype Orderは廃止されます。代替APIへの移行をお願いします。

4.2 mutation cancelOrder

4.2.1 複数個フェーズでの変更点

複数個フェーズにおいて、mutation cancelOrderは廃止されます。

4.2.2 廃止・移行について

複数個フェーズにおいて、mutation cancelOrderは廃止される予定です。代替として、以下のAPIへの移行をお願いします:

- mutation cancelOrderTransaction: 注文全体のキャンセル
- 部分キャンセル用API（今後提供予定）: 特定の商品のみのキャンセル

4.3 mutation completeOrder

4.3.1 複数個フェーズでの変更点

複数個フェーズでは、completeOrderはOrderに紐づく商品1個のみを発送する挙動に変更されます。OrderTransactionに対してcompleteOrderを実行することはできません。

4.3.2 廃止・移行について

旧API群廃止フェーズにおいて、mutation completeOrderは廃止される予定です。代替として、以下のAPIへの移行をお願いします:

- query orderShippings: 発送情報の取得
- mutation createOrderShipping: 発送処理の開始
- mutation deleteOrderShipping: 発送処理の取りやめ
- mutation completeOrderShipping: 発送処理の完了

4.4 mutation updateShippingTrackingCode

4.4.1 複数個フェーズでの変更点

複数個フェーズにおいて、updateShippingTrackingCodeに以下の制限が追加されます:

- 対象の限定: Orderに紐づく商品1個に対する追跡情報の更新に限定されます
- ステータス制限: COMPLETEDステータスのOrderに対してのみ実行可能になります。COMPLETED以外のOrderに対して実行するとエラーが返されます

4.4.2 廃止・移行について

旧API群廃止フェーズにおいて、mutation updateShippingTrackingCodeは廃止される予定です。代替として、以下のAPIへの移行をお願いします:

- mutation updateOrderShippingTrackingCode: 新しい発送フローでの追跡情報更新

4.5 mutation addTransactionMessage

4.5.1 複数個フェーズでの変更点

複数個フェーズにおいて、mutation addTransactionMessageは廃止されます。

4.5.2 廃止・移行について

複数個フェーズにおいて、mutation addTransactionMessageは廃止される予定です。代替として、以下のAPIへの移行をお願いします:

- mutation addOrderTransactionMessage: OrderTransactionレベルでの取引メッセージ投稿

4.6 webhook order_created

4.6.1 複数個フェーズでの変更点

複数個フェーズでは、商品数量1ごとに1回Webhookが送信されます。例えば、商品3個の注文を受けた場合、Webhookは3回送信されます。

これは、Webhookが個別のOrderを基準として送信されるためです。OrderTransactionが持つ商品個数分のOrderが作成されるため、それぞれ異なるOrderIDを持つWebhookが発行されます。

Webhookペイロードの詳細

- 同じ商品を複数個注文した場合：各Webhookのproductフィールドは同じ内容になります
- 異なる商品を複数注文した場合：各Webhookのproductフィールドはそれぞれ異なる商品の情報が含まれます
- 同一注文の識別：Webhookの内容のみから同一のカート注文であることを判別する方法はありません。今後追加予定のOrderTransactionIdフィールドを使用して判別する必要があります

4.6.2 廃止・移行について

旧API群廃止フェーズにおいて、webhook order_createdは廃止される予定です。代替として、以下のWebhookへの移行をお願いします：

- webhook order_transaction_created: OrderTransactionレベルでの注文作成通知

4.7 webhook order_paid

4.7.1 複数個フェーズでの変更点

webhook order_createdと同様に、商品数量1ごとに1回Webhookが送信されます。Webhookペイロードの詳細についてもwebhook order_createdと同じです。

4.7.2 廃止・移行について

旧API群廃止フェーズにおいて、webhook order_paidは廃止される予定です。代替として、以下のWebhookへの移行をお願いします：

- webhook order_transaction_paid: OrderTransactionレベルでの支払い完了通知

4.8 webhook order_canceled

4.8.1 複数個フェーズでの変更点

webhook order_createdと同様に、商品数量1ごとに1回Webhookが送信されます。Webhookペイロードの詳細についてもwebhook order_createdと同じです。

4.8.2 廃止・移行について

旧API群廃止フェーズにおいて、webhook order_canceledは廃止される予定です。代替として、以下のWebhookへの移行をお願いします:

- webhook order_transaction_canceled: OrderTransactionレベルでのキャンセル通知

4.9 webhook transactionmessage_created

4.9.1 複数個フェーズでの変更点

複数個フェーズにおいて、webhook transactionmessage_createdは廃止されます。

4.9.2 廃止・移行について

複数個フェーズにおいて、webhook transactionmessage_createdは廃止される予定です。代替として、以下のWebhookへの移行をお願いします:

- webhook order_transaction_message_created: OrderTransactionレベルでの取引メッセージ作成通知

4.10 mutation debugCreateOrder

mutation debugCreateOrderは、複数商品に対応した新しいテスト注文APIがリリースされた時に廃止される予定です。

現在のdebugCreateOrderは単一商品・1個の注文のみ対応しているため、複数商品・複数個の注文に対応した新しいテスト注文API（今後提供予定）への移行をお願いします。

5. 新API群詳細

5.1 OrderTransaction

OrderTransactionは複数商品・複数個の注文を表現するリソースです。従来のOrderとは異なり、カート機能による複数の商品を1つの取引として管理します。

なお、単一商品の注文であっても、新API群ではOrderTransactionとして表現されます。

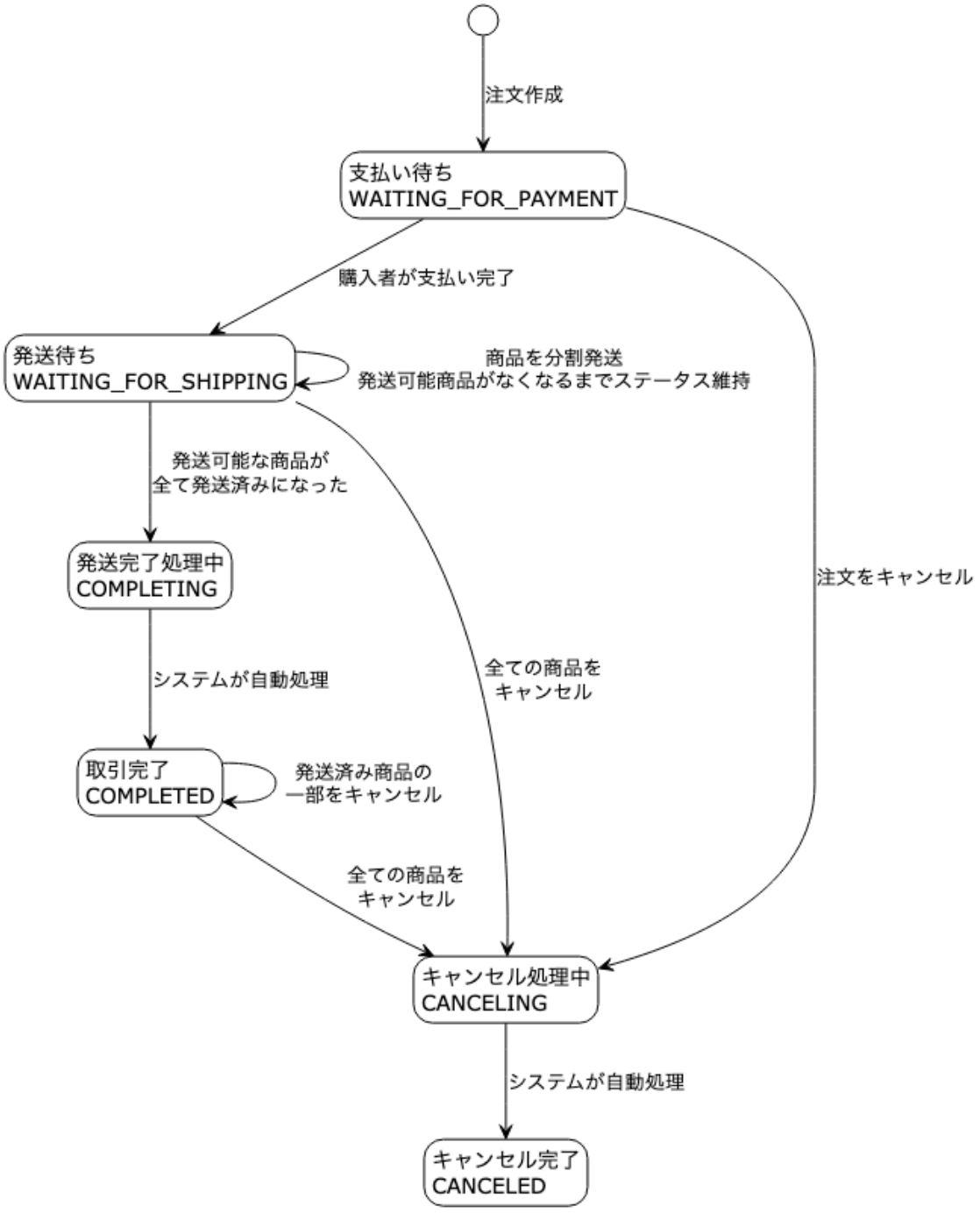
OrderTransactionの情報はquery orderTransactionまたはquery orderTransactionsを使用して取得できます。

5.1.1 ステータス遷移

OrderTransactionのステータスは、基本的には既存のOrderと同じように遷移します。ただし、複数商品を扱うため一部の遷移条件に変更があります。

変更点は以下の通りです:

- WAITING_FOR_SHIPPING: 発送可能な商品がなくなるまでこのステータスを維持します（発送可能な商品の定義については、後述の「商品毎のステータスと数量」で説明します）
- CANCELING, CANCELED: 注文内の全ての商品がキャンセルされた時に遷移します
- COMPLETING, COMPLETED: 発送可能な商品がなく、かつ全ての商品がキャンセルされていない場合に遷移します

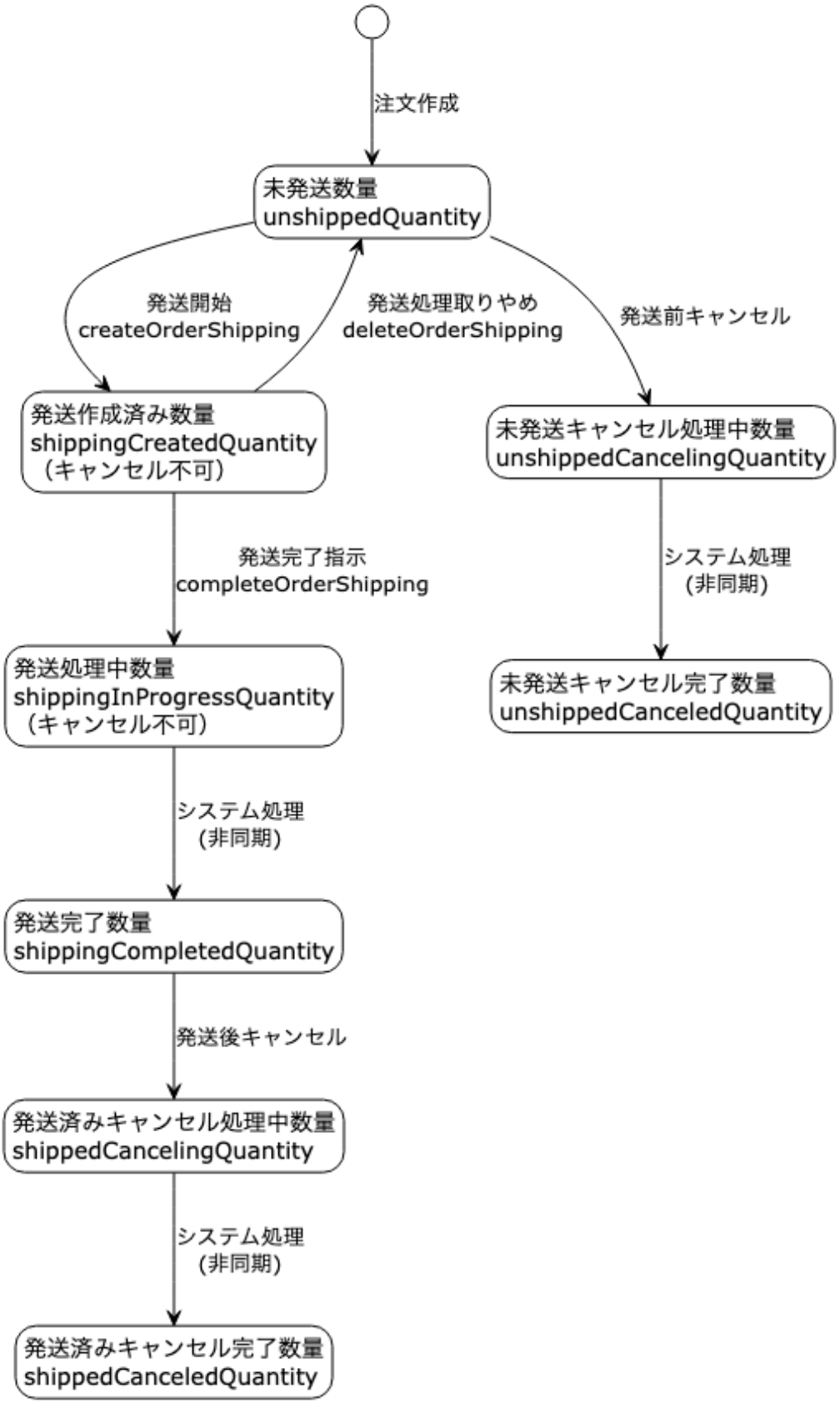


商品3個の注文におけるステータス遷移の例:

1. 購入者が支払いを完了し、WAITING_FOR_SHIPPINGになります
2. 商品2個を先行して発送します。発送可能な商品が残っているため、WAITING_FOR_SHIPPINGを維持します
3. 3個目の商品が発送前にキャンセルされます。発送可能な商品がなくなったため、COMPLETINGを経てCOMPLETEDに遷移します
4. 発送済みの商品2個がキャンセルされます。全ての商品がキャンセルされたため、CANCELINGを経てCANCELEDに遷移します

5.1.2 商品毎のステータスと数量

商品の状態はOrderTransactionProductによって管理されます。各商品の数量は複数のquantityフィールドによってそれぞれの状態が表現されます。



数量フィールド一覧

フィールド名	説明
purchasedQuantity	注文された商品毎の総数量（不変）。他の全ての数量フィールドの合計と一致します
unshippedQuantity	未発送の数量。発送処理により減少します
shippingCreatedQuantity	発送開始済みの数量。createOrderShippingによりunshippedQuantityから移動します。この状態ではキャンセル不可です
shippingInProgressQuantity	システムによる発送処理中の数量。completeOrderShippingによりshippingCreatedQuantityから遷移し、システムの非同期処理でshippingCompletedQuantityへ移動します。この状態ではキャンセル不可です
shippingCompletedQuantity	発送完了済みの数量。システムの非同期処理によりshippingInProgressQuantityから遷移します
unshippedCancelingQuantity	未発送キャンセル処理中の数量。発送前キャンセルによりunshippedQuantityから遷移し、システムの非同期処理でunshippedCanceledQuantityへ移動します
unshippedCanceledQuantity	未発送状態からキャンセル完了した数量。システムの非同期処理によりunshippedCancelingQuantityから遷移します（終端状態）
shippedCancelingQuantity	発送済みキャンセル処理中の数量。発送後キャンセルによりshippingCompletedQuantityから遷移し、システムの非同期処理でshippedCanceledQuantityへ移動します
shippedCanceledQuantity	発送後にキャンセル完了した数量。システムの非同期処理によりshippedCancelingQuantityから遷移します（終端状態）

発送可能な商品の定義

OrderTransactionのステータス遷移で使用される「発送可能な商品」とは、unshippedQuantity + shippingCreatedQuantity + shippingInProgressQuantity が1以上の商品を指します。つまり、未発送または発送処理中の商品がある状態です。

商品**A×5**個の注文における数量変化の例

下記の順序で処理が進行した場合の、数量変化の例です。

1. 注文確定
2. 3個発送開始
3. 3個発送完了指示
4. システム処理（3個）
5. 未発送2個キャンセル開始
6. システム処理（2個）
7. 発送済み1個キャンセル開始
8. システム処理（1個）

段階	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
unshipped	5	2	2	2	0	0	0	0
shippingCreated	0	3	0	0	0	0	0	0
shippingInProgress	0	0	3	0	0	0	0	0
shippingCompleted	0	0	0	3	3	3	2	2
unshippedCanceling	0	0	0	0	2	0	0	0
unshippedCanceled	0	0	0	0	0	2	2	2
shippedCanceling	0	0	0	0	0	0	1	0
shippedCanceled	0	0	0	0	0	0	0	1

5.1.3 クーポン

複数商品・複数個の注文に対応するため、クーポン情報の管理方法が変更されます。

従来のOrderではクーポン情報は注文レベルで管理されていましたが、OrderTransactionでは商品単位でクーポン情報を持つことになります。これにより、商品個別にクーポンを使用することが可能になり、複数商品を注文する際は各商品につき最大1種類のクーポンを使用できます。

クーポン情報はOrderTransactionProductCouponとして表現され、以下のフィールドによってクーポンの状態を管理します:

- reservedCount: 注文時に使用されたクーポンの枚数（不変）
- usedCount: 実際に使用されたクーポン枚数（初期値は0、値引きされた商品を発送すると増加）
- canceledCount: 取り消しされたクーポンの枚数（初期値は0、商品キャンセルによって増加）

商品5個注文時（クーポン5枚使用）におけるクーポン数量変化の例:

段階	処理	**reservedCount**	**usedCount**	**canceledCount**
1	注文確定	5	0	0
2	2個発送完了	5	2	0
3	1個発送前キャンセル	5	2	1
4	2個発送完了	5	4	1
5	1個発送後キャンセル	5	3	2

5.1.4 宛先住所

OrderTransactionでは、複数の配送方法が混在する注文に対応するため、宛先住所の公開ルールが変更されます。

- 従来のルール（Order）:
    - メルカリ便（匿名配送）の場合、shippingフィールドはnull
- 新しいルール（OrderTransaction）:
    - 匿名配送のみの注文: 宛先住所は非公開（shippingAddressフィールドがnull）
    - 匿名配送と非匿名配送が混在する注文: 宛先住所が公開される

例として、メルカリ便商品と未定配送商品を同時に注文した場合、未定配送のために住所情報が必要となるため、宛先住所が確認可能になります。

宛先住所はshippingAddressフィールドで取得できます。

5.2 OrderShipping

OrderShippingは分割発送を管理するためのリソースです。複数商品の注文では、注文全体の状態と個別の発送状態が異なるため、OrderShippingによって各発送処理を独立して管理します。

OrderShippingの情報はquery orderShippingsを使用して取得できます。

5.2.1 発送処理の手順

OrderShippingを使用した発送処理は、以下の2段階で行います:

- createOrderShipping: 発送対象の商品を選択し、発送処理を開始
- completeOrderShipping: 発送完了と売上反映

発送処理を取りやめる場合は、deleteOrderShippingを使用して処理をキャンセルできます。

なお、completeOrderShippingにより最後の商品を発送すると、OrderTransactionがCOMPLETING(COMPLETED)ステータスに遷移します。

発送処理のフロー:

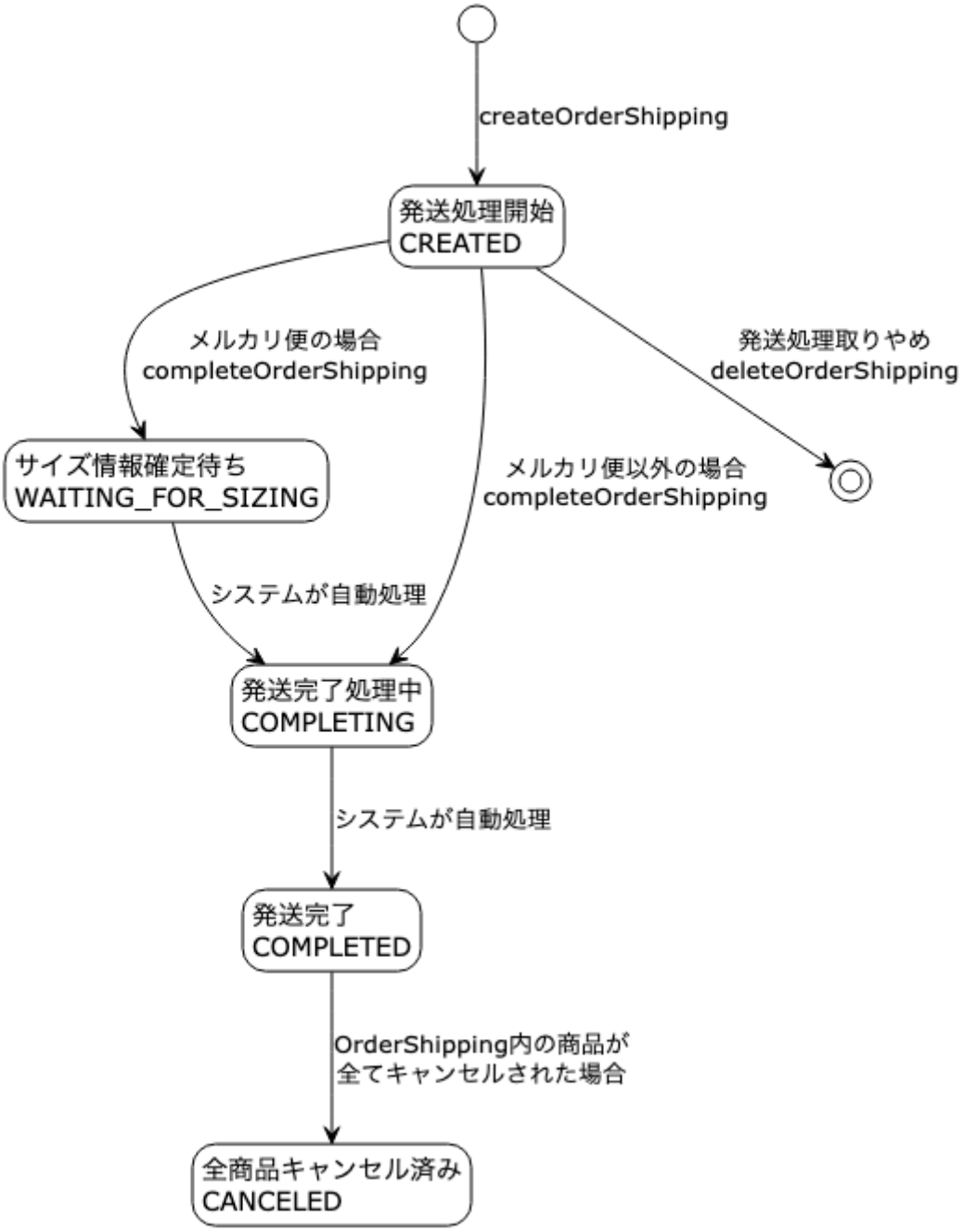
- 発送処理開始〜完了: createOrderShipping → completeOrderShipping
- 発送処理取りやめ: createOrderShipping → deleteOrderShipping

制限事項:

- 商品キャンセルとの制約: 発送処理中（OrderShippingが存在する状態）の商品はキャンセルできません。商品をキャンセルする場合は、先にdeleteOrderShippingで発送処理を取りやめてください
- 旧API群との制約: OrderShippingが存在する状態では、cancelOrderやcompleteOrderなどの旧APIはエラーになります。旧APIを使用する場合は、先にdeleteOrderShippingで発送処理を削除してください
- 配送方法の統一: OrderShippingを作成する際は、同じ配送方法を持つ商品のみを対象にする必要があります。例えば、メルカリ便商品と未定配送商品を混在させることはできません
- 梱包単位: メルカリ便では、1つのOrderShippingにつき1つの伝票が発行されるため、1つの梱包に相当する商品をcreateOrderShippingで指定する必要があります
- 荷物の追跡情報: 荷物の追跡情報は、createOrderShippingを実行した後に設定可能になります

5.2.2 ステータス遷移

OrderShippingのステータスは、発送処理の段階に応じて以下のように遷移します:



基本的なステータス遷移

- CREATED: createOrderShippingにより発送処理が開始された状態
- COMPLETED: completeOrderShippingにより発送完了と売上反映が行われた状態

メルカリ便の場合の追加ステータス

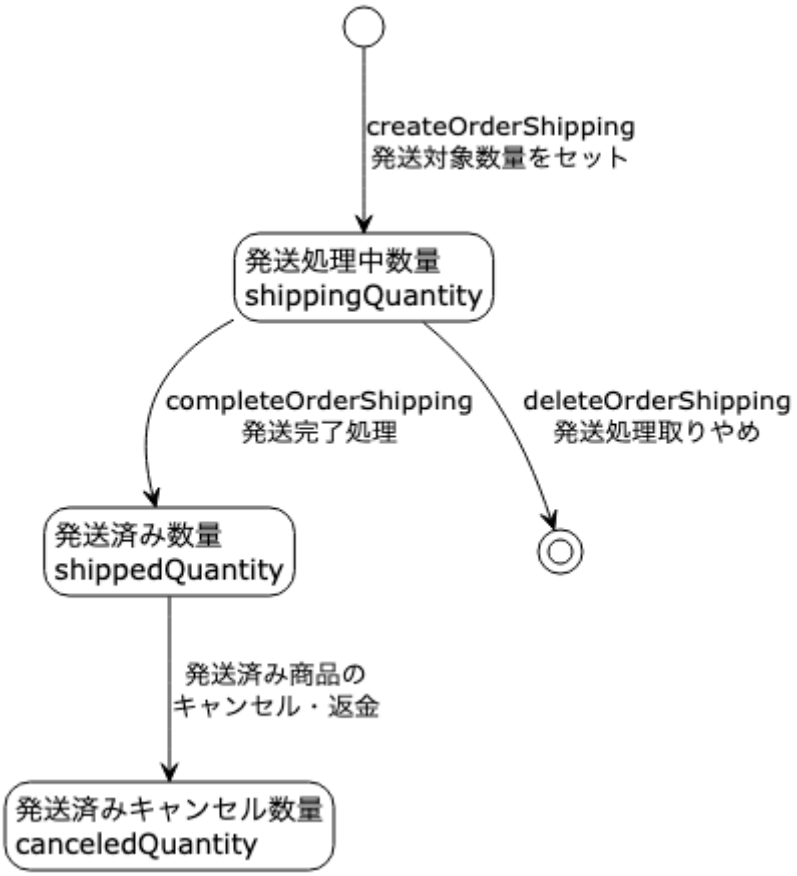
メルカリ便を使用する場合、発送処理には追加のステータスが存在します。これらのステータスはメルカリ便の配送状況と連動して自動的に更新されます。

発送処理の取りやめ

deleteOrderShippingを実行すると、OrderShippingは削除され、以降のクエリで取得できなくなります。削除されたOrderShippingに含まれていた商品は、再び未発送状態に戻ります。

5.2.3 数量の遷移

OrderShipping内の商品数量は、発送処理の進行に応じて管理されます。各商品の状態はOrderShippingProductの数量フィールドによって表現されます。



- shippingQuantity: 発送処理中の数量。createOrderShippingにより発送対象数量がセットされます
- shippedQuantity: 発送完了済みの数量。completeOrderShippingによりshippingQuantityから遷移します
- canceledQuantity: 発送済み商品のキャンセル・返金によりshippedQuantityから遷移します

5.2.4 idempotencyKey

createOrderShippingでは、重複処理を防止するためにidempotencyKeyの指定が必要です。これは、ネットワークの問題やタイムアウトによってAPIレスポンスが失われた場合にも、同じ処理が重複実行されることを防ぐためのものです。

使用例

商品2個の注文で1個のみを対象に発送処理を開始する場合を考えてみます:

- createOrderShippingを実行（idempotencyKey: "ship-001"）
- 内部処理は成功したが、レスポンスがネットワーク問題で失われる
- クライアントがリトライを実行（同じidempotencyKey: "ship-001"）
- 冪等処理により、新たなOrderShippingは作成されず、最初の処理結果が返される

もし、idempotencyKeyフィールドが存在しなかった場合、システムは冪等処理すべきかの判断ができず、リトライ時に2個目の商品を対象とした別のOrderShippingが作成されてしまいます。重複処理を防ぐため、idempotencyKeyフィールドが必要になります。

仕様

- 必須フィールド: createOrderShippingで必ず指定する必要があります
- スコープ: OrderTransaction内でユニークである必要があります。異なるOrderTransactionであれば、同じ値を使用可能です
- 形式: 半角英数大文字・小文字、ハイフン、アンダースコア、最大255文字
- 保持期間: 永続的にシステム内に保持されます

リトライ時の注意点

- 同一パラメータ: リトライ時は同じidempotencyKeyと同じパラメータを使用してください
- パラメータ変更時のエラー: idempotencyKeyを変更せずに他のパラメータを変更すると、FAILED_PRECONDITIONエラーが返されます
- バリデーションエラー: バリデーションエラーなどリトライできないエラーの場合、idempotencyKeyは記録されないため、同じ値を再利用できます

5.2.5 送料フィールド

OrderShippingでは、送料負担者により送料の記録場所が異なります。送料を表すフィールドは以下の2つです:

- OrderShipping.sellerShippingFee: 出品者負担の送料（メルカリ便）
- OrderShippingProduct.buyerShippingFee: 購入者負担の送料

購入者負担送料の場合

購入者負担の送料では、商品ごとに個別の送料が設定されているため、発送対象商品を表すOrderShippingProductのbuyerShippingFeeとして記録されます。

メルカリ便の場合

メルカリ便では、複数商品を同一のOrderShippingで発送する場合でも、梱包のサイズによって送料が決まります。個々の商品に対する送料を紐づけることができないため、OrderShippingのsellerShippingFeeとして記録されます。

送料割引が適用された場合

送料割引が適用された注文では、個別の商品に送料を紐づけることができなくなります。そのため、buyerShippingFeeやsellerShippingFeeとは異なるフィールドで送料を表現することになります。具体的なフィールドについては、送料割引用のAPIを追加するタイミングで詳細をアナウンスいたします。

5.3 OrderとOrderTransactionの互換性

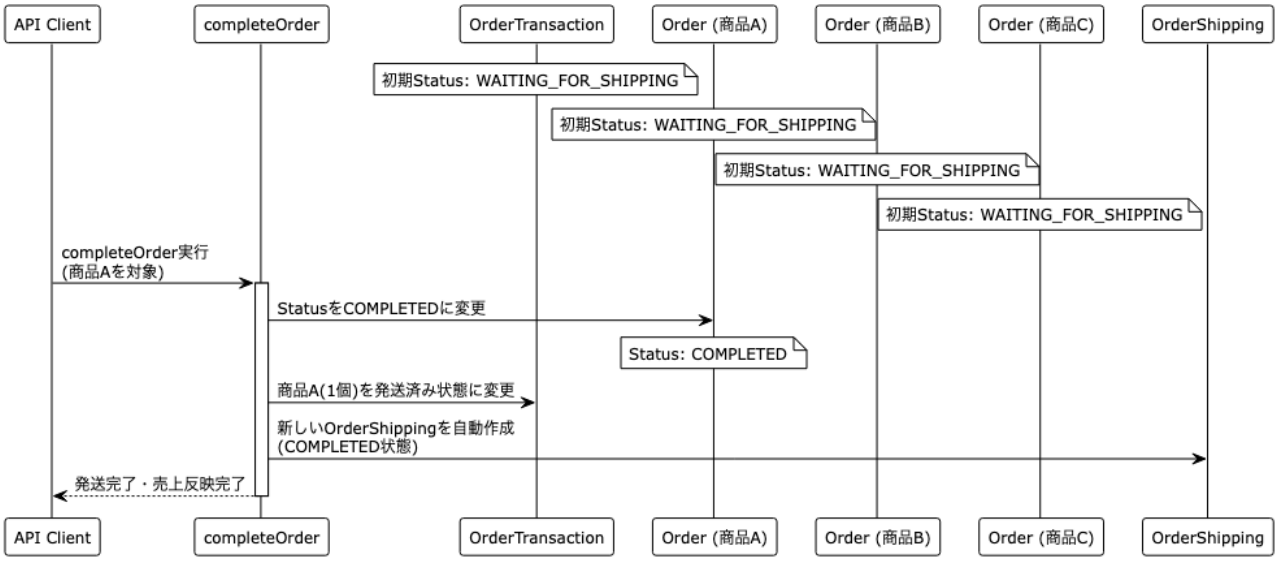
複数個フェーズ以降、1つのOrderTransactionに対して複数のOrderが作成されるようになります。この際、旧API群と新API群の間には相互に影響を与える関係があります。

5.3.1 旧API群から新API群への影響

旧API群でOrderを操作すると、親OrderTransactionにも影響が反映されます。以下、各操作の影響について図解します。

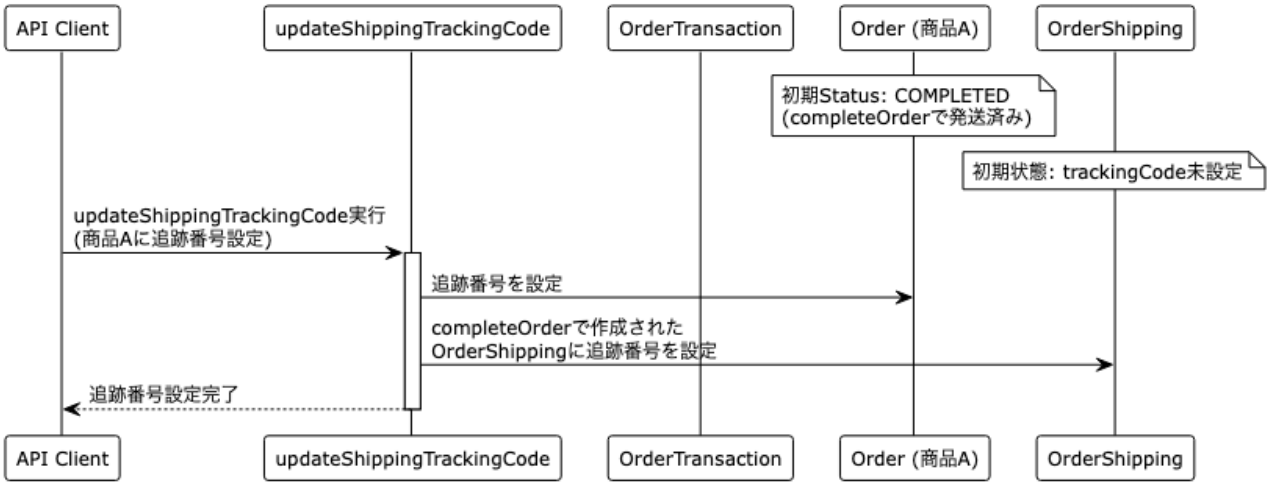
completeOrder実行時の影響

OrderTransactionの対象商品が1個発送された状態になり、COMPLETED状態のOrderShippingも自動的に作成されます。



updateShippingTrackingCode実行時の影響

completeOrderで作成されたOrderShippingの追跡情報が更新されます。

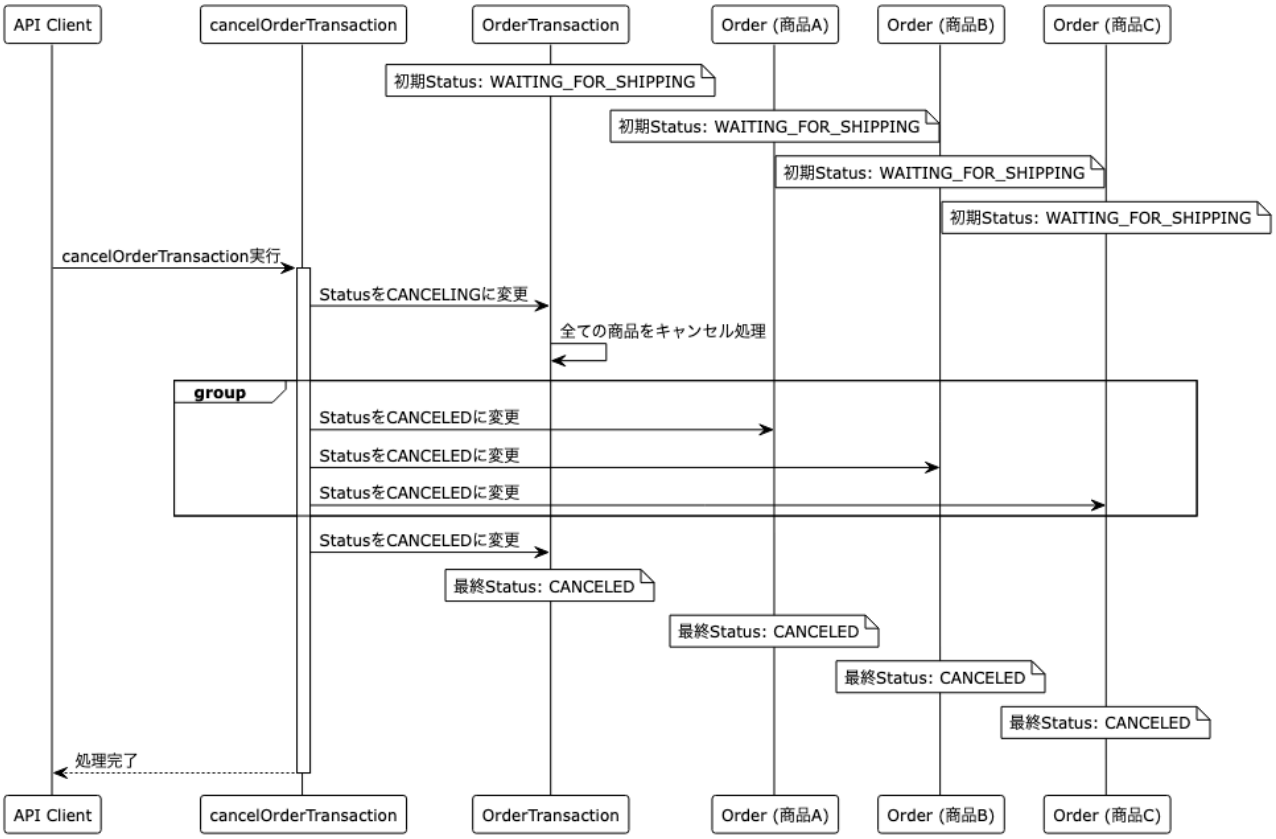


5.3.2 新API群から旧API群への影響

新API群でOrderTransactionを操作すると、関連するOrderにも影響が反映されます。以下、各操作の影響について図解します。

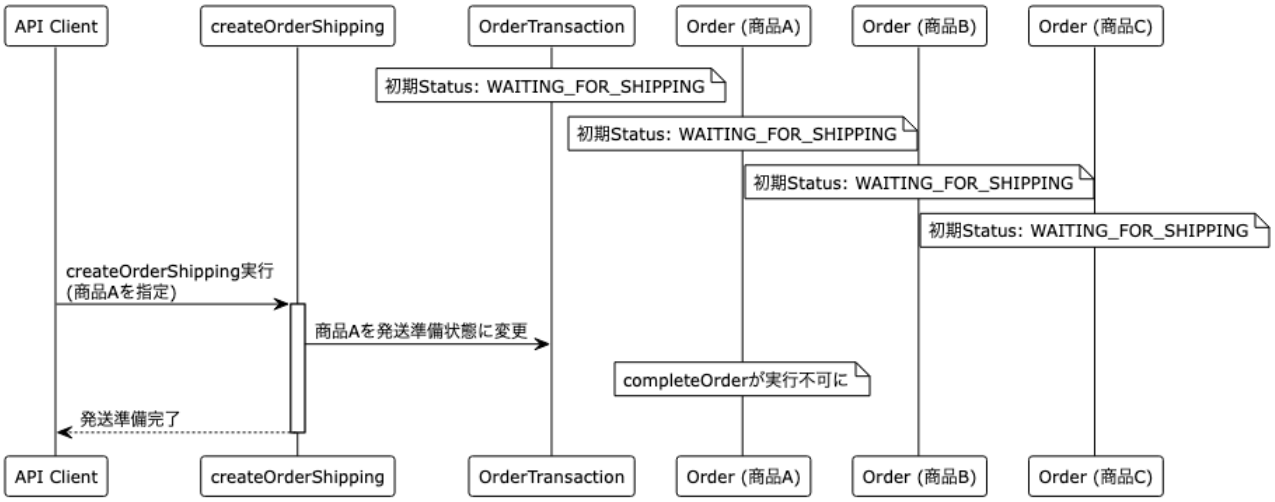
cancelOrderTransaction実行時の影響

OrderTransactionに紐づく全てのOrderがCANCELED状態になります。



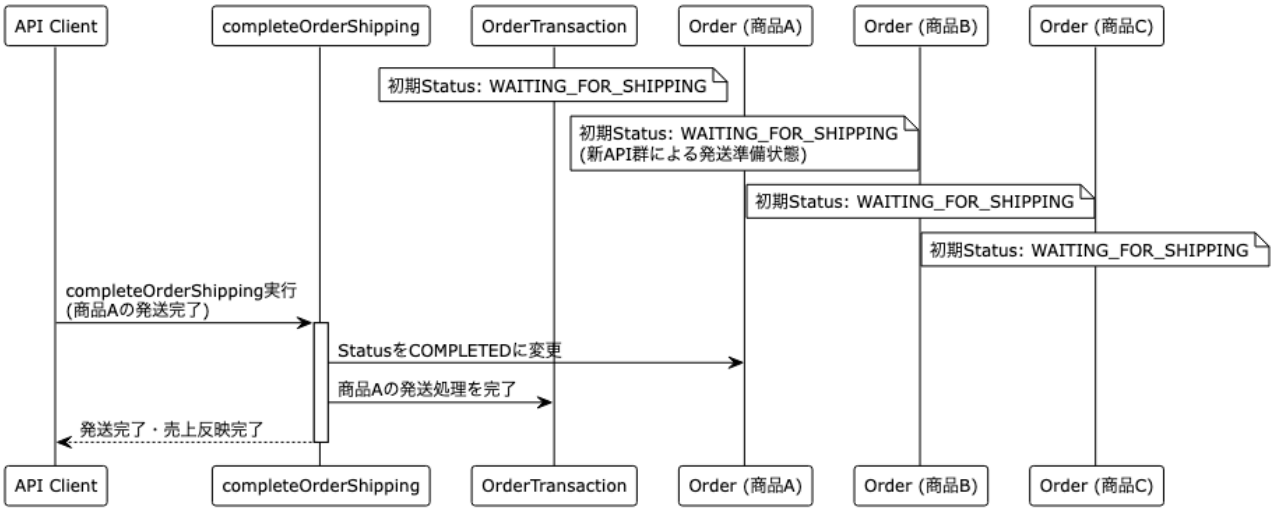
createOrderShipping実行時の影響

指定された商品に紐づくOrderに対してcompleteOrderが実行できなくなります（新API群による発送準備状態になります）。



completeOrderShipping実行時の影響

createOrderShippingによって発送準備状態になっていたOrderがCOMPLETED状態になります。



addOrderTransactionMessage実行時の影響

Order側のmessagesフィールドには影響を与えません（nullのままとなります）。