ItkTwoManHandling

Entwicklerdokumentation

André Fischer

30.03.2026

Inhaltsverzeichnis

1 Paket- und Namespace-Daten

- **Plugin-Klasse:** `ItkComputerGmbh\\ItkTwoManHandling\\ItkTwoManHandling`
- PSR-4: **`ItkComputerGmbh\\ItkTwoManHandling\\`** → `src/`
- **Shopware:** `shopware/core` ~6.6.0

---

2 Vollständige PHP-Klassenliste

Klasse	Rolle
ItkTwoManHandling	**install** / **activate** / **deactivate** / **uninstall**, legt Custom Field Set itk_two_man_handling an bzw. entfernt es
Service\TwoManHandlingService	Liest Config-Produkt-IDs, lädt Service-Produkte aus dem Repository
Core\Checkout\Cart\TwoManHandlingCart Processor	**CartProcessorInterface**, Priority `4500` – Children anlegen/entfernen
Core\Checkout\Cart\Delivery\DeliveryCalculator Decorator	Decorates **Shopware\Core\Checkout\Cart\Delivery\Delivery Calculator** – summiert **shipping-surcharge**-Children in die Lieferkosten
Core\Checkout\Cart\Price\CartPriceCalculator	**CartProcessorInterface**, Priority `−5000` – schließt ITK-Children vom Warenkorbsummen-Total aus
Subscriber\LineItemPayloadSubscriber	**BeforeLineItemAddedEvent** – liest Request, setzt Payload (**itkTwoManHandlingOptions**, ggf. `api_nr`)
Subscriber\LineItemAddedSubscriber	**LineItemAddedEvent**, **CartChangedEvent**, **Before LineItemAddedEvent** – synchronisiert Optionen mit LineItem
Subscriber\ProductPageSubscriber	**ProductPageLoadedEvent** – Extension für Twig, ob Checkboxen angezeigt werden

Storefront-Assets: `src/Resources/app/storefront/src/main.js` registriert das Plugin; Implementierung in `src/Resources/app/storefront/src/plugin/two-man-`

handling.plugin.js. **Twig:** u. a. buy-widget.html.twig, product.html.twig (Line Item), offcanvas-cart-summary.html.twig.

---

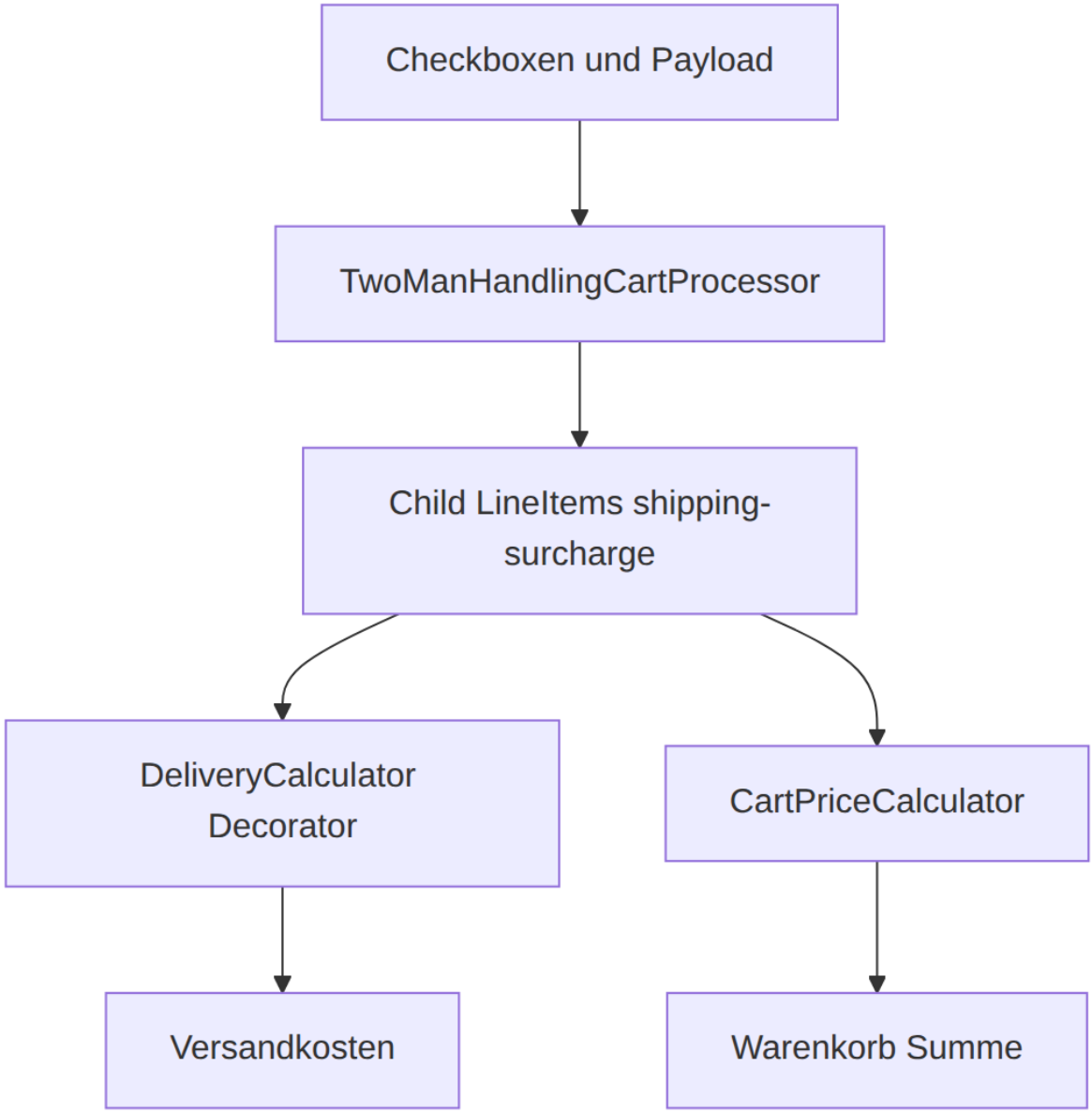
3 Architektur-Überblick

Das Plugin klinkt sich tief in die Shopware-Cart-Pipeline ein. Die Kernaufgabe ist simpel – zwei Zusatzleistungen als Child-LineItems einem Hauptprodukt zuordnen – aber die Umsetzung ist komplex, weil Shopware's Cart-System viele bewegliche Teile hat.

Das Grundprinzip: Wenn ein Kunde ein Produkt mit aktiviertem 2-Mann-Handling in den Warenkorb legt und dabei die Checkboxen anklickt, werden im Warenkorb zusätzliche LineItems vom Typ **shipping-surcharge** als Kind-Elemente des Hauptprodukts hinzugefügt. Die Kosten dieser Kind-Elemente fließen in die Versandkosten ein (über den DeliveryCalculator-Decorator) und werden aus dem Warenkorb-Gesamtbetrag herausgerechnet (über den CartPriceCalculator).

Symfony/Shopware**:** Ein **CartProcessor** läuft in einer **priorisierten Liste** – der Code entscheidet anhand von Payload/Checkboxen, ob er Line Items ergänzt. **Decorators** auf **DeliveryCalculator** bzw. **CartPriceCalculator** „wickeln" die Original-Services: Mittels **inner** aufruf passen sich danach Beträge/Kosten an.

3.1 Diagramm: Warenkorb-Pipeline



4 Custom Fields

Beim Aktivieren des Plugins legt die **ItkTwoManHandling**-Hauptklasse ein Custom-Field-Set an. Dabei wird **customFieldSetRepository** genutzt.

Field-Set-Name: `itk_two_man_handling`

Zugeordnete Entity: product

Die zwei Felder:

Feld-Key	Typ	Verwendungszweck
itk_supports_two_man_handling	**bool**	Steuert, ob die Zusatzoptionen am Produkt angezeigt werden
itk_two_man_handling_hide_product	**bool**	Versteckt das Produkt im regulären Storefront (für Service-Produkte)

Beim Deinstallieren des Plugins werden das Field-Set und seine Felder wieder entfernt. Das geschieht in der `uninstall()`-Methode der Hauptklasse.

---

5 Plugin-Konfiguration

Die Datei `src/Resources/config/config.xml` definiert in einer Karte zwei Admin-Komponenten **sw-entity-single-select** mit `<entity>product</entity>`:

- **twoManHandlingProductId** → `ItkTwoManHandling.config.twoManHandlingProductId`
- **oldDeviceReturnProductId** → `ItkTwoManHandling.config.oldDeviceReturnProductId`

Die Werte werden im **TwoManHandlingService** per `SystemConfigService::getString(..., $salesChannelId)` gelesen (sales-channel-spezifisch möglich).

---

6 Symfony-Dienste (`services.xml`)

Service-ID	Kurzbeschreibung
itk_two_man_handling_logger	**Monolog\Logger**, Kanal/Name itk_two_man_handling

Service-ID	Kurzbeschreibung
itk_two_man_handling_log_handler	**StreamHandler** → %kernel.project_dir%/var/log/itk_two_man_handling.log (Level-Argument im XML: `100` = DEBUG)
ItkComputerGmbh\ItkTwoManHandling\Service\TwoManHandlingService	Config, Produkt-Repo, Logger
...\TwoManHandlingCartProcessor	Tag shopware.cart.processor, priority `4500`
...\Delivery\DeliveryCalculatorDecorator	**decorates Shopware\Core\Checkout\Cart\Delivery\DeliveryCalculator**
...\Price\CartPriceCalculator	Tag shopware.cart.processor, priority `-5000`
...\Subscriber\LineItemPayloadSubscriber	kernel.event_subscriber
...\Subscriber\LineItemAddedSubscriber	kernel.event_subscriber
...\Subscriber\ProductPageSubscriber	kernel.event_subscriber (+ monolog.logger channel)

7 Cart-Pipeline im Detail

Die Logik verteilt sich auf mehrere Klassen, die in einer bestimmten Reihenfolge aktiv werden. Hier der Gesamtflow:

```
Kunde sendet Add-to-Cart-Request
            |
            ▼
LineItemPayloadSubscriber::onBeforeLineItemAdded()  [BeforeLineItemAddedEvent]
  → Liest Checkbox-Parameter aus dem HTTP-Request
  → Setzt Payload `itkTwoManHandlingOptions` (und ggf. weitere Payload-Felder)

LineItemAddedSubscriber (LineItemAddedEvent / CartChangedEvent / BeforeLineItemAddedEvent)
  → Schreibt Debug-/Logger-Einträge, prüft Produkttyp und 2-Mann-Fähigkeit erneut und stößt die Service-Logik an, damit Parent-LineItem, Payload und spätere Cart-Verarbeitung zusammenpassen
            |
            ▼
  [Cart Recalculation startet]
```

```
                │
                ▼
TwoManHandlingCartProcessor  (Priorität 4500)
  → Prüft für jedes LineItem ob itk_supports_two_man_handling gesetzt
    ist
  → Liest itkTwoManHandlingOptions aus dem Payload
  → Fügt Child-LineItems hinzu oder entfernt sie
                │
                ▼
DeliveryCalculatorDecorator
  → Erkennt Child-LineItems mit customLineItemType = "shipping-
    surcharge"
  → Addiert ihre Preise zu den Delivery-Kosten
                │
                ▼
CartPriceCalculator  (Priorität -5000)
  → Schließt Children mit itk_two_man_handling_child-Flag
    aus dem Warenkorb-Gesamtbetrag aus
```

7.1 LineItemPayloadSubscriber (Request → Payload)

Dieser Subscriber hört auf **BeforeLineItemAddedEvent**. **Bevor** das LineItem in den Warenkorb aufgenommen wird, liest er aus dem aktuellen Request per `$request->request->all('customFields')` das Array der **Formular-Custom-Fields** aus (das Storefront-JS legt Hidden-Inputs `customFields[itk_two_man_handling]` und `customFields[itk_old_device_return]` an, Wert **1**).

Das komplette Array wird unter **itkTwoManHandlingOptions** im LineItem-Payload abgelegt – **nicht** unter dem Schlüssel **customFields** im Payload, damit Shopwares spätere Überschreiberei auf **customFields** die ITK-Auswahl nicht zerstört.

Wichtiger Hinweis: Der **TwoManHandlingCartProcessor** wertet **itkTwoManHandlingOptions** aus (`itk_two_man_handling` / `itk_old_device_return` als truthy Werte). Die Methode `TwoManHandlingService::handleTwoManHandling()` (aufgerufen vom **LineItemAdded-Subscriber**) liest dagegen zusätzlich `$lineItem->getPayload()['customFields']` – das kann je nach Request/Shopware-Version von den ITK-Optionen abweichen; für die **Anlage der Child-LineItems** ist der Processor maßgeblich.

Zusätzlich kann der Subscriber `custom_product_infos_important_api_artnr` vom **Hauptprodukt** laden und als `api_nr` in die Payload schreiben.

7.2 LineItemAddedSubscriber

Reagiert auf **LineItemAddedEvent**, **CartChangedEvent** und **BeforeLineItemAddedEvent**. Schreibt ausführliche Logs (`error_log` + Monolog) und ruft für Produkt-LineItems **`TwoManHandlingService::handleTwoManHandling()`** auf (Prüfung `itk_supports_two_man_handling`, Auswertung von Payload-Daten). Die **materialisierten** Service-Children kommen wie beschrieben aus dem **TwoManHandlingCartProcessor**. Für die **Übernahme der Checkboxen aus dem HTTP-Request** ist der **LineItemPayloadSubscriber** zuständig.

7.3 TwoManHandlingCartProcessor

Der **TwoManHandlingCartProcessor** wird mit Priorität `4500` registriert. Das ist höher als die meisten Standard-Shopware-Processors, damit wir sicher vor den Preisberechnungen laufen.

In PHP erklärt: Ein Cart Processor ist eine Klasse, die Shopware bei jeder Warenkorb-Neuberechnung aufruft. Jede solche Klasse bekommt den aktuellen Warenkorb, darf ihn verändern und gibt ihn dann weiter an den nächsten Processor. Die Priorität steuert die Reihenfolge.

Was der Processor konkret macht:

1. Er iteriert über alle LineItems im Warenkorb.
2. Für jedes LineItem prüft er, ob das zugehörige Produkt das Custom Field **itk_supports_two_man_handling = true** hat.
3. Er liest aus dem LineItem-Payload den Schlüssel **itkTwoManHandlingOptions**.
4. Basierend auf diesen Optionen entscheidet er, welche Child-LineItems vorhanden sein sollen.
5. Er vergleicht den Soll-Zustand (aus den Optionen) mit dem Ist-Zustand (vorhandene Children im Warenkorb) und legt fehlende Children an oder entfernt überschüssige.

Die Child-LineItems werden im **TwoManHandlingService** (**addTwoManHandlingChild** / **addOldDeviceReturnChild**) als **`LineItem::CUSTOM_LINE_ITEM_TYPE`** mit Referenz auf die konfigurierte Produkt-ID erzeugt, Preis per **`calculateProductPrice()`**, Labels fest im Code („DHL 2-Mann-Handling“ bzw. „Altgeräte-Rücknahme“). Wesentliche Payload-Marker:

```php
$childLineItem = new LineItem($lineItemId, LineItem::
    CUSTOM_LINE_ITEM_TYPE, $productId, 1);
$childLineItem->setGood(false);
$childLineItem->setStackable(false);
$childLineItem->setRemovable(true); // im aktuellen Code: einzeln
    entfernbar
$childLineItem->setPayloadValue('itk_two_man_handling_child', true); //
     bzw. itk_old_device_return_child
$childLineItem->setPayloadValue('customLineItemType', 'shipping-
    surcharge');
```

Der **DeliveryCalculatorDecorator** erkennt die Zusatzkosten über **customLineItemType === 'shipping-surcharge'** (nicht über den Shopware-LineItem-Typ).

7.4 DeliveryCalculatorDecorator

Dieser Decorator ummantelt `Shopware\\Core\\Checkout\\Cart\\Delivery\\DeliveryCalculator` (siehe `services.xml`, **decorates="..."**). Das Decorator-Muster: gleiche öffentliche API wie der innere Service, Aufruf von `.inner`, danach Anpassung der Lieferkosten.

Was der Decorator macht: Er iteriert nach dem Original-Calculations über die berechneten Deliveries und sucht nach Child-LineItems mit **customLineItemType = 'shipping-surcharge'**. Wenn er welche findet, addiert er deren Preise zu den Delivery-Kosten hinzu.

Das Ergebnis: Die Zusatzleistungen erscheinen in der Versandkosten-Zeile der Bestellung, nicht als separate Produktpreise.

7.5 CartPriceCalculator

Der **CartPriceCalculator** läuft mit Priorität `-5000`, also als letzter in der Kette. Er stellt sicher, dass die Child-LineItems nicht in den Warenkorbgesamtpreis einfließen – denn die Kosten sind ja bereits in den Versandkosten.

Konkret: Beim Berechnen des Warenkorb-Gesamtbetrags schließt dieser Calculator alle LineItems aus, die das Payload-Flag `itk_two_man_handling_child` oder `itk_old_device_return_child` gesetzt haben.

---

8 Storefront-JS: `two-man-handling.plugin.js`

Produktseite: Listener auf den Checkboxen; beim Klick auf **In den Warenkorb** wird `event.preventDefault()` gesetzt, `customFields[itk_two_man_handling]` und ggf. `customFields[itk_old_device_return]` als Hidden-Inputs ins **data-product-detail-form** eingefügt, danach `form.submit()`. Ohne dieses JS erreicht der Server keine **customFields**-Struktur → **itkTwoManHandlingOptions** bleibt leer → kein Child im Processor.

Altgerät-Checkbox: Ist nur aktiv, wenn 2-Mann angehakt ist (JS deaktiviert und leert sie sonst).

Umfang: Die Datei endet nach der Button-Text-Aktualisierung; **kein** separates Warenkorb-Plugin in derselben Datei – Darstellung/Verhalten von Children im Checkout erfolgt über **Twig** und Shopware-Standard. Child-Items sind im Service aktuell `setRemovable(true)`.

---

9 ProductPageSubscriber

Der **ProductPageSubscriber** hört auf das **ProductPageLoadedEvent** von Shopware. Er liest für die aktuell angezeigte Produktseite das Custom Field `itk_supports_two_man_handling` aus und setzt eine Twig-Extension:

```
1 $page->addExtension('itkTwoManHandlingConfig', [
2     'supportsTwoManHandling' => $product->getCustomFields()['
    itk_supports_two_man_handling'] ?? false,
3     'enableOldDeviceReturn'  => true, // im Code fest true
4 ]);
```

Die Twig-Templates lesen `page.extensions.itkTwoManHandlingConfig`, um Checkboxen und Sektionen einzublenden.

---

10 Logging und Debugging

Das Plugin schreibt Logs nach `var/log/itk_two_man_handling.log`. Neben dem dedizierten Logger gibt es im Code auch direkte **`error_log()`**-Aufrufe (vor allem in der Hauptklasse `ItkTwoManHandling.php`). Diese landen im PHP-Fehler-Log des Servers, nicht in der Shopware-Log-Datei.

Debug-Tipps:

Wenn Child-LineItems nicht hinzugefügt werden: Im **TwoManHandlingCartProcessor** am Anfang der **`process()`**-Methode einen Log-Eintrag setzen und prüfen, ob der Processor überhaupt aufgerufen wird. Oft liegt das Problem daran, dass der Processor deregistriert ist (Services-XML prüfen) oder die Priorität falsch ist.

Wenn die Optionen zwar im Processor ankommen, aber die Children nicht korrekt hinzugefügt werden: Den Payload des LineItems loggen und prüfen, ob **itkTwoManHandlingOptions** tatsächlich drin steckt und die Werte korrekt sind.

Wenn die Preise nicht stimmen: Den **DeliveryCalculatorDecorator** ist der Anlaufpunkt. Prüfen, ob die Child-Items korrekt als **shipping-surcharge** markiert sind.

---

11 Bekannte Eigenheiten und Fallstricke

Zwei Payload-Pfade: LineItemPayloadSubscriber füllt **itkTwoManHandlingOptions** aus dem Request-**customFields**-Array. `TwoManHandlingService::handleTwoManHandling()` wertet zusätzlich **payload['customFields']** aus. Der **TwoManHandlingCartProcessor** nutzt für Children ausschließlich **itkTwoManHandlingOptions**. Bei Debugging beide Stellen und den rohen Request vergleichen.

Service-Produkte müssen kaufbar sein: Die Service-Produkte müssen technisch als aktive Shopware-Produkte existieren, damit der Cart Processor sie über das Repository laden kann. Sie dürfen nur nicht im Storefront sichtbar sein. Das `itk_two_man_handling_hide_product`-Custom-Field sorgt für die Ausblendung – aber wenn jemand direkt die Produkt-URL aufruft, ist die Produktseite trotzdem erreichbar. Das ist kein kritisches Problem, aber es sollte beim Onboarding kommuniziert werden.

Entfernen von Child-Items: Im aktuellen Code sind Children `setRemovable(true)`. Ob sie nach dem Entfernen beim nächsten Recalculate wieder auftauchen, hängt davon ab, ob **itkTwoManHandlingOptions** im Parent-LineItem noch die jeweilige Option enthält – der Processor fügt fehlende Children wieder hinzu, solange die Flags gesetzt sind.