ItkReturnForm

Entwicklerdokumentation

André Fischer

30.03.2026

Inhaltsverzeichnis

1 Technischer Überblick

Namespace: `ItkComputerGmbh\\ReturnForm`
Plugin-Klasse: `ItkComputerGmbh\\ReturnForm\\ItkReturnForm`
Config: `ItkReturnForm.config.*` (**returnDeadlineInDays**, **supportMailAddress**, **returnMailAddress**)

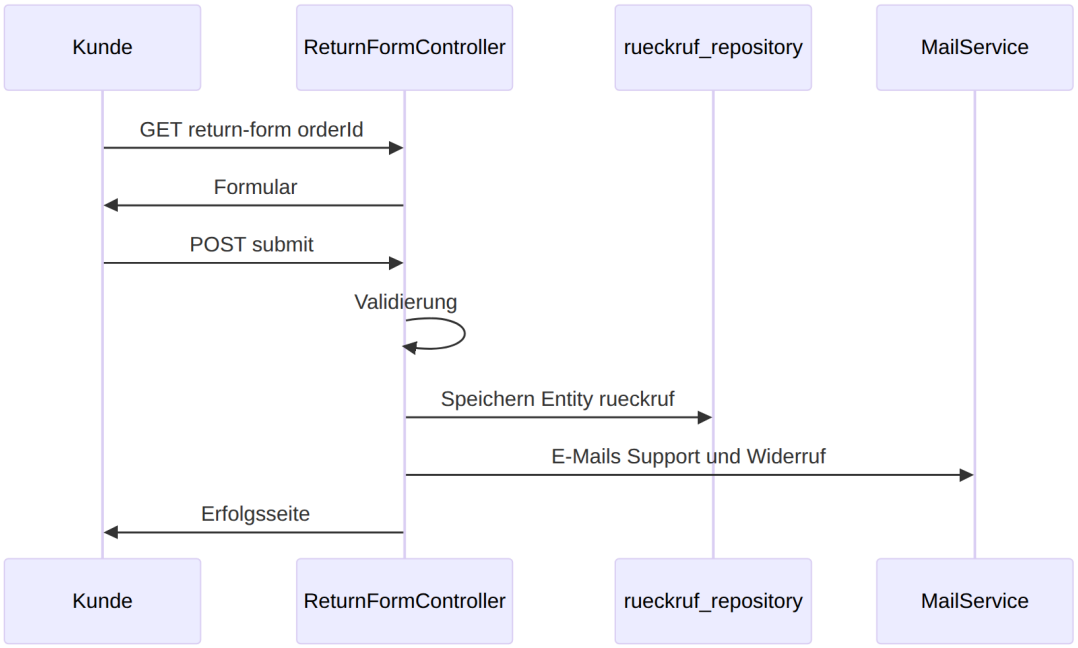
Storefront-Controller: `Controller\\ReturnFormController`
Twig: `Twig\\ReturnFormExtension`
DAL: `Entity\\RueckrufEntityDefinition` → Entity-Name **rueckruf**, Repository `rueckruf.repository`
CLI: `Command\\ListRueckrufeCommand` → `itk:list:rueckrufe`

1.1 Symfony / Shopware kurz

- **#[Route]** auf dem Controller: Mappt URLs auf Methoden; `_routeScope` = **storefront** hält die Route im Storefront-Kontext (Session, Theme).
- **Twig Extension:** Registriert Funktionen/Filter für Templates (`getReturnDaysLimit()` etc.) – kommt global in Twig an, sobald das Bundle geladen ist.
- **DAL Entity: rueckruf** ist eine eigene Tabelle; du arbeitest mit **EntityRepository** und **Criteria**, nicht mit `$connection->query()`.

Hier gibt es zusätzlich ein Shopware-spezifisches Detail aus `services.xml`: Der Controller bekommt per `setContainer()` und `setTwig()` den Service-Container und Twig gesetzt. Das ist bei älteren oder stärker auf Storefront-Controller zugeschnittenen Plugins ein übliches Muster und wichtig, wenn du denselben Stil beim Erweitern beibehalten willst.

1.2 Diagramm: Submit Ablauf (vereinfacht)



2 Routen (ReturnFormController)

Methode	Pfad	Name
GET	/return-form/{orderId}	frontend.return.form
POST	/return-form/submit	frontend.return.form.submit
GET	/account/widerrufe	frontend.account.rueckrufungen.page

`_routeScope`: **storefront**

Validierung (Submit, Kurz): Login; Order gehört zum Customer; Mengen gegen Summe aus Order-LineItems minus bereits gespeicherte **returnedProductIds**; **privacyPolicy** gesetzt; **receivedDate** oder **wareNotReceived**; Datum nicht in der Zukunft.

3 Fachlicher Ablauf

3.1 Formularanzeige

Die GET-Route `/return-form/{orderId}` lädt Bestellung, Positionen und Kundenzuordnung. Geschäftskunden werden bewusst abgefangen: Wenn `$customer->getCompany() !== null`, leitet der Controller mit Flash-Meldung um. Das ist keine UI-Laune, sondern harte fachliche Regel im Code.

3.2 Submit-Flow

Beim POST auf `/return-form/submit` passiert technisch in dieser Reihenfolge:

1. Request-Daten lesen (`products_ids`, `products_quantities`, **reason**, **privacyPolicy**, **receivedDate**, **wareNotReceived**, **orderId**)
2. Kunden- und Bestellzuordnung prüfen
3. Mengen gegen bereits vorhandene Widerrufe derselben Bestellung validieren
4. **returnedProductIds** als JSON-Struktur bauen
5. Datensatz in `rueckruf.repository` anlegen
6. Bestellung mit allen Mail-relevanten Associations neu laden
7. zwei E-Mails versenden: intern an Support und extern als Empfangsbestätigung an den Kunden
8. Erfolgsseite rendern

Die JSON-Struktur von **returnedProductIds** besteht aus Objekten mit:

- **id**
- **quantity**

Genau diese Struktur ist wichtig, wenn du Daten migrierst oder Widerrufe manuell im Admin / per DAL anlegst.

---

4 Twig-Extension

ReturnFormExtension:

- **`getReturnDaysLimit()`** → `ItkReturnForm.config.returnDeadlineInDays` oder 28

- **get_product_by_id(productId, context?)** → lädt **ProductEntity** mit Cover, Manufacturer, Prices

Registrierung in `services.xml` des Plugins.

---

5 Installation / Datenbank (wichtig)

`ItkReturnForm::createDatabase()` führt **CREATE TABLE rueckruf ohne IF NOT EXISTS** aus und wird von **`install()`** und **`update()`** aufgerufen.

Folge: Auf Umgebungen, in denen die Tabelle bereits existiert, kann das Update eine **Exception** werfen (wird je nach Umgebung gefangen oder bricht ab). Für saubere Deployments sollte langfristig auf **Shopware-Migrationen** mit **IF NOT EXISTS** / Versionsprüfung umgestellt werden.

Hinweis: Im SQL der Plugin-Klasse ist **customer_id BINARY(32)** definiert – Shopware-UUIDs sind üblicherweise 16 Byte; mit der DAL-Definition abgleichen (**StringField**/**BinaryField** in **RueckrufEntityDefinition**).

---

6 Persistierte Felder in rueckruf

Die Doku ist hier absichtlich konkreter als der kurze Validierungsabschnitt oben. In der Praxis speichert das Plugin u. a.:

- **orderId**
- **customerId**
- **reason**
- **processed**
- **createdAt**
- **updatedAt**
- **returnedProductIds**
- **wareNotReceived**
- **receivedDate**
- **voucherWanted** (fachlich vorhanden, im normalen Storefront-Submit aber nicht aktiv gesetzt)

Wenn du Supportfälle aus der Datenbank rekonstruierst, reichen diese Felder schon, um den kompletten Formularzustand nachzuvollziehen.

---

7 PHP-Klassen (vollständig unter `src/`)

Klasse	Rolle
ItkReturnForm	Plugin lifecycle, **createDatabase**, **getCommands**
Controller\ReturnFormController	Formular + Submit + Mail
Entity\RueckrufEntityDefinition	DAL
Entity\RueckrufEntity	Entity
Entity\RueckrufCollection	Collection
Twig\ReturnFormExtension	Twig-Funktionen
Command\ListRueckrufeCommand	CLI-Liste

---

8 Assets / Templates

- Storefront-Templates: `Resources/views/storefront/…` (z. B. `return-form.html.twig`, Erfolgsseite)
- E-Mail-Templates: `rueckruf-eingegangen.html.twig` und `rueckruf-bestaetigung.html.twig`
- Snippets, JS, CSS je nach Theme-Integration unter **Resources**

---

9 E-Mail-Flow

Der Controller baut nach dem Speichern zwei getrennte Mail-Payloads:

9.1 Interne Mail an Support

- Empfänger: `ItkReturnForm.config.supportMailAddress`
- Absendername: **Widerruf**
- Absenderadresse: `ItkReturnForm.config.returnMailAddress`
- Betreff mit Bestellnummer
- Template: `@ItkReturnForm/storefront/email/rueckruf-eingegangen.html.twig`

9.2 Bestätigung an den Kunden

- Empfänger: Kunden-E-Mail aus der Bestellung / dem eingeloggten Kontext
- Absendername: im Code fest `25now`
- Absenderadresse: `ItkReturnForm.config.returnMailAddress`
- Template: `@ItkReturnForm/storefront/email/rueckruf-bestaetigung.html.twig`

Beide Mails bekommen weitgehend dieselben Variablen:

- **order**
- **rueckruf**
- **shippingAddress**
- **customer**
- **paymentMethod**
- **shippingMethod**
- **products**

Wenn Mails falsch aussehen oder Informationen fehlen, ist genau diese Variablenliste dein Debug-Einstieg.

---

10 Konto-Seite / Rückrufübersicht

Die Route `/account/widerrufe` lädt die Datensätze des eingeloggten Kunden paginiert:

- **limit = 10**
- Sortierung **createdAt DESC**

Zusätzlich lädt der Controller die betroffenen Bestellungen mit Associations nach, damit die Übersicht nicht nur rohe **rueckruf**-Datensätze, sondern auch lesbare Bestellinformationen anzeigen kann.

---

11 CLI

11.1 itk:list:rueckrufe

Der Command wird nicht über eine normale `services.xml`-Registrierung bereitgestellt, sondern über **`ItkReturnForm::getCommands()`** in der Plugin-Klasse. Das ist ein älteres, aber in Shopware weiterhin mögliches Muster.

Im Alltag ist der Befehl vor allem für schnelle technische Sichtprüfungen nützlich:

- welche Einträge existieren überhaupt
- in welcher Reihenfolge wurden sie angelegt
- ob Felder wie **processed**, **createdAt** oder **orderId** plausibel aussehen

Wenn du erwartest, einen Command in `services.xml` zu finden, such hier also stattdessen zuerst in der Plugin-Hauptklasse.