ItkNewsletterActiveCampaign

Entwicklerdokumentation

André Fischer

07.04.2026

Inhaltsverzeichnis

1 Ausgangssituation

Shopware 6 bringt im Storefront ein **Newsletter-Formular** mit (über den Formular-Typ „Newsletter“ bzw. die zugehörige Storefront-Anbindung): Nutzer können ihre E-Mail hinterlegen, der Versand läuft über die **Newsletter-Empfänger-Verwaltung** (`newsletter_recipient`) des Verkaufskanals. **Double Opt-In (DOI)** und die grundsätzliche Subscribe-/Unsubscribe-Abwicklung über die **Sales-Channel-Routen** sind **Kernfunktionen von Shopware**.

Was der **Standard** in der Praxis typischerweise **nicht** abdeckt:

- Es gibt **keine** eingebaute Möglichkeit, sich im selben Formular **gezielt bei mehreren, fachlich getrennten Listen** anzumelden (z. B. „Produktnews“, „Club“, „Events“) – fachlich bleibt es oft bei **einem** Newsletter-Ja/Nein bzw. einer undifferenzierten Anmeldung.
- Es gibt **keine** direkte Anbindung an **ActiveCampaign** (Kontakte, Listenzugehörigkeiten, Tags, externe Abmeldungen per Webhook), sodass Marketing-Listen und Shopware-Empfänger **ohne Erweiterung nicht** synchron gehalten werden können.

Ziel dieses Plugins: Die bestehende Shopware-Newsletter-Strecke beizubehalten, wo sinnvoll, und sie so zu **erweitern**, dass **mehrere konfigurierbare Listen** (mit optional je eigener DOI-Mail und Redirect), **Kategorie-/Consent-Logik** und eine **synchrone Anbindung an ActiveCampaign** möglich sind – ohne den Core zu patchen (Decorator, eigene Controller-Ebene, Subscriber).

---

2 Shopware-Standard und Plugin-Erweiterung

Die folgende Tabelle fasst zusammen, **was unverändert aus dem Shopware-Kern genutzt wird** und **wo dieses Plugin eingreift oder ersetzt**. Details zu Klassen und Abläufen folgen in den späteren Kapiteln.

Bereich	Bleibt Shopware-Standard (Kern)	Wird durch das Plugin ergänzt oder ersetzt
Datenmodell Basis	Entity **Newsletter-Empfänger**, Zuordnung zum **Verkaufskanal**, grundsätzliches Subscribe/Unsubscribe über **Sales-Channel-API** / Routen	Zusätzliche **Custom Fields** am Empfänger (Listenstatus, DOI je Liste, Consent-Metadaten, App-Tracking usw.) per **Migrationen**; optional **Kategorie-Feld** zur Steuerung sichtbarer Listen

Bereich	Bleibt Shopware-Standard (Kern)	Wird durch das Plugin ergänzt oder ersetzt
Formular im Storefront	Ursprünglich verarbeitet der **Core** das Newsletter-Formular (z. B. unter **/form/newsletter**) und spricht die Subscribe-Route an	Ein **Interceptor** leitet die Bearbeitung auf einen **eigenen** NewsletterFormController um: nach dem Decorator-Lauf folgen u. a. **ActiveCampaign-Sync**, **Audit-Log**, Anpassungen für Club-/Consent-Szenarien (**JSON-Antwort** bleibt für das Formular-Konzept kompatibel)
Subscribe-Logik	Die **innere** NewsletterSubscribeRoute bleibt die referenzierte Implementierung; bei option !== subscribe delegiert der Decorator **vollständig** an den Kern	Bei **option === subscribe** fängt **ListDoiNewsletter SubscribeRouteDecorator** ab: **Mehrfachauswahl Listen**, Filter nach Konfiguration/Kategorie, **DOI-Orchestrierung** (u. a. über **DoiMailSend Service**), Re-Subscribe-Verhalten; danach ggf. **AC-Updates** im Formular-Controller
Unsubscribe	Kern-**UnsubscribeRoute** wird weiter genutzt; Abmeldung aus Shopware-Sicht	Ergänzung um **ActiveCampaign** (z. B. Kontakt/Liste), **Webhook** für extern ausgelöste Abmeldungen mit **Geheimnisprüfung**
DOI-Bestätigung	**NewsletterConfirmEvent** und Empfängerstatus im Kern bleiben maßgeblich	**Subscriber** reagieren auf Bestätigung und Abmeldung: AC synchronisieren, **Redirects** nach DOI, konsistent mit Listen-Status
Admin / Konfiguration	Verkaufskanal-Einstellungen, allgemeines Newsletter-Verhalten wo nicht überschrieben	Plugin-`config.xml`: AC-URL, Token, Webhook-Secret, **JSON der Listen** (Keys, AC-List-IDs, Templates, Redirects), Tag-Namen für Tracking-Consent
Konto / Profil	Standard-Konto-Oberfläche	Zusätzliche **Routen**, **Widget** und **Services**, um Listen im **Kundenkonto** zu pflegen (inkl. DOI erneut senden)

Kurz gesagt: **Shopware liefert Empfänger, Kanal-Kontext und die Basis-Routen inkl. DOI-Mechanik.** Dieses Plugin **schaltet sich für die Subscribe-Schicht und das Formular davor/dahinter** ein, **erweitert das Datenmodell** und **synchronisiert mit ActiveCampaign** – ohne die Kernklassen im `vendor`-Verzeichnis zu ändern.

---

3 Technischer Überblick

Plugin-Klasse: `ItkComputerGmbh\\ItkNewsletterActiveCampaign\\ItkNewsletterActiveCampaign`
Admin-Label: „Newsletter ActiveCampaign“
Namespace: `ItkComputerGmbh\\ItkNewsletterActiveCampaign`
Config-Präfix: `ItkNewsletterActiveCampaign.config.*`

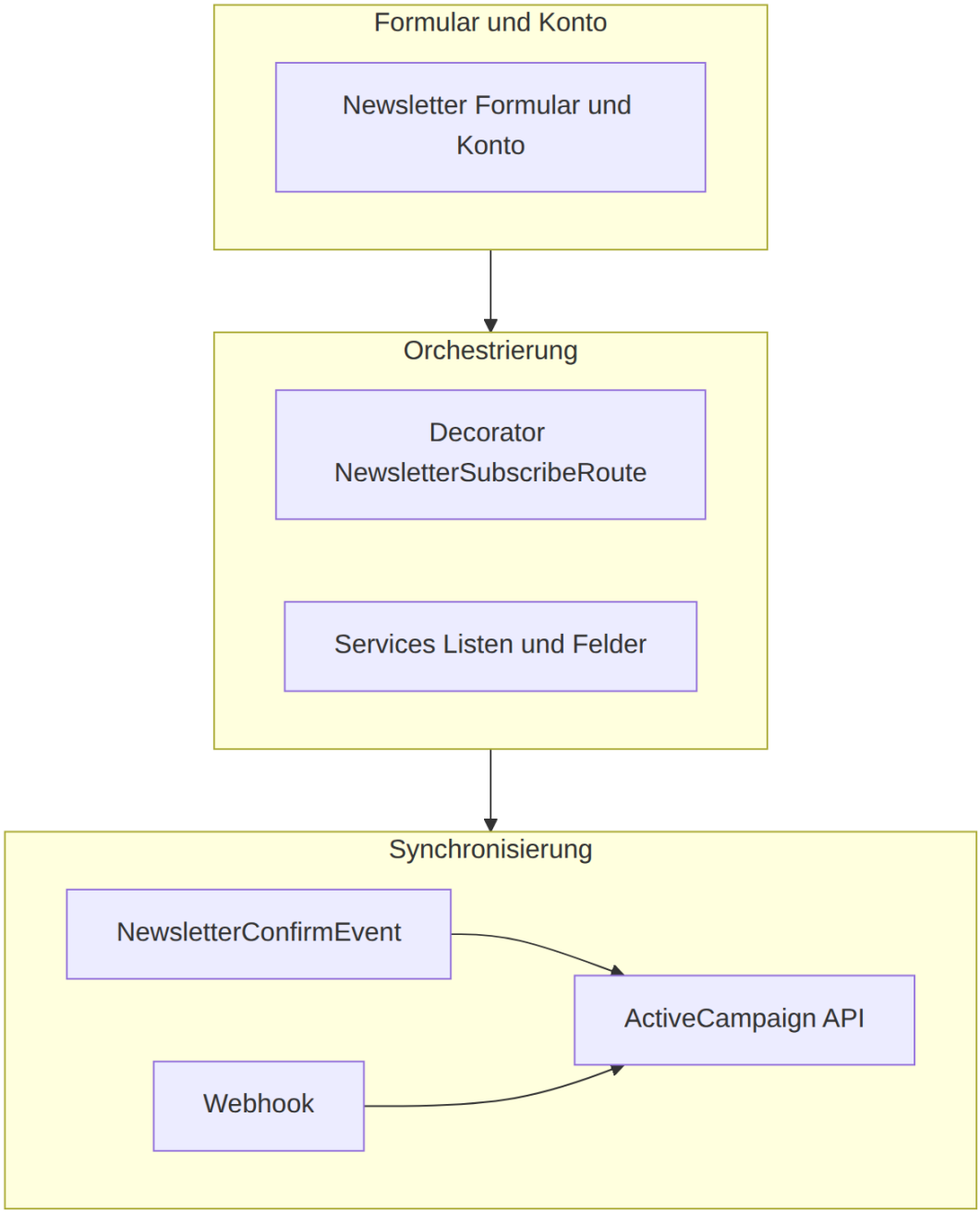
Kernmuster:

- **Decorator ListDoiNewsletterSubscribeRouteDecorator** auf **NewsletterSubscribeRoute** – Multi-Listen, DOI, AC-Unsubscribe bei Entfernen, Custom-Mail-Template
- **Subscriber** auf **NewsletterConfirmEvent**, **NewsletterUnsubscribeEvent**, Kernel **RESPONSE**, Account-Seiten, Profil-Widget, ggf. Friendly Captcha
- **Services:** Listen-Config, Orchestrierung Custom Fields, DOI-Mail, AC-API, Account-Listen, Audit-Log
- **Storefront-Controller:** Account-Listen, Webhook, Ersatz-Logik für Newsletter-Formular (Interceptor)

Technisch besteht das Plugin aus drei Schichten:

1. **Form- und Account-Ebene**: Storefront-Formulare, Konto-Seite und Widget sammeln Listen-Auswahlen und Consent.
2. **Orchestrierungsebene**: Decorator + Services entscheiden, welche Listen neu hinzukommen, welche pending bleiben, welche bestätigt sind und was zu ActiveCampaign synchronisiert wird.
3. **Synchronisierungsebene**: Subscriber, DOI-Mail und Webhook halten Shopware-Recipient, ActiveCampaign-Listen und Redirect-Verhalten zusammen.

3.1 Diagramm: Schichten und Datenfluss



3.2 Decorator vs. Subscriber (für Symfony-Umsteiger)

- **Decorator** auf **NewsletterSubscribeRoute**: Shopware ruft beim Newsletter-Abonnieren nicht mehr direkt die Core-Implementierung auf, sondern deine Klasse, die intern **–>**

`getDecorated()->subscribe(...)` aufrufen kann. So erweiterst du **eine einzelne Route-Logik**, ohne `vendor/shopware` zu patchen.

- **Event Subscriber:** Für Dinge, die **global** als Reaktion auf ein Ereignis passieren (z. B. **NewsletterConfirmEvent**), nutzt du Subscriber – sie sind entkoppelt von einer konkreten Route.

---

4 Systemkonfiguration (`config.xml`)

Alle Keys liegen unter `ItkNewsletterActiveCampaign.config.*` und sind sales-channel-spezifisch.

4.1 ActiveCampaign API

Feld	Key	Zweck
Basis-URL	**activeCampaignBaseUrl**	API-Root von ActiveCampaign, ohne abschließenden Slash
API-Token	**activeCampaignApiToken**	Authentifizierung für Kontakt-, Listen- und Tag-Calls
Webhook-Geheimnis	**activeCampaignWebhookSecret**	Prüfung für den Unsubscribe-Webhook; akzeptiert Query `?secret=` oder Header **X-Itk-Newsletter-Webhook-Secret**
Tracking-Tag Name	**activeCampaignTrackingTagName**	Tag für Empfänger mit aktivem App-Tracking-Consent
No-Tracking-Tag Name	**activeCampaignNoTrackingTagName**	Gegentag für Empfänger ohne App-Tracking

4.2 Newsletter-Listen

Feld	Key	Zweck
Listen (JSON)	**newsletterLists**	JSON-Array der angebotenen Listen

Erwartete Struktur pro Listen-Objekt:

- **key**: interner Schlüssel, mit dem das Plugin arbeitet
- **label**: sichtbarer Name im Formular / Konto
- **acListId**: Ziel-Liste in ActiveCampaign
- **templateId**: optionales DOI-Mail-Template nur für diese Liste
- **redirectUrl**: optionale Zielseite nach DOI für diese Liste
- **hidden**: optional; blendet die Liste im Storefront/Konto aus, ohne sie fachlich zu entfernen

Die Kategorie-Steuerung läuft zusätzlich über das Custom Field `itk_nl_category_list_id`: leer bedeutet „alle konfigurierten Listen“, ein gesetzter Key bedeutet „nur diese eine Liste“.

---

5 Services und `services.xml`

Das Plugin hängt stark an Symfony-Services. Für das Verständnis reicht meist diese Faustregel:

- **Controller** beantworten HTTP-Requests.
- **Services** enthalten die eigentliche Fachlogik.
- **Subscriber** reagieren auf Ereignisse.
- **Der Decorator** hängt sich direkt in den Shopware-Newsletter-Flow.

Wichtige Service-Verdrahtung aus `services.xml`:

Service	Wichtige Abhängigkeiten	Warum wichtig
ActiveCampaign Service	**SystemConfigService**, **HttpClientInterface**, monolog.logger	baut und sendet alle AC-Requests
ListDoi Orchestrator Service	**NewsletterListConfigService**, **ActiveCampaign Service**, Logger	zentrale Listen-Delta-Logik
DoiMailSend Service	**SendMailTemplate**, event_dispatcher, newsletter_recipient.repository	DOI-Mail-Versand
Account NewsletterLists Service	**router**, request_stack, **SystemConfigService**, **NewsletterDoiConsentLogService**, **ActiveCampaignService**	Account-Änderungen, DOI-Resend, Redirects

Service	Wichtige Abhängigkeiten	Warum wichtig
ListDoi Newsletter SubscribeRoute Decorator	NewsletterSubscribeRoute.inner, newsletter_recipient.repository, **ListDoi OrchestratorService**, **DoiMailSendService**	Kernstück für Multi-Listen-Subscribe

Wichtig für Symfony-Einsteiger: Der Decorator wird nicht „einfach so“ entdeckt, sondern in `services.xml` über `decorates="Shopware\\Core\\Content\\Newsletter\\SalesChannel\\NewsletterSubscribeRoute"` registriert. Dadurch ersetzt deine Klasse den Core-Service nach außen, kann intern aber immer noch die originale Route über `.inner` aufrufen.

---

6 Alle PHP-Klassen unter `src/`

Klasse	Rolle
ItkNewsletterActiveCampaign	Plugin-Bootstrap
Service\NewsletterListConfigService	JSON-Listen parsen, sichtbare Listen, Keys, AC-ID-Auflösung
Service\ListDoiOrchestratorService	Delta toAdd/toRemove, Custom-Field-Payloads, Normalisierung RequestDataBag
Service\DoiMailSendService	Custom-Template vs. Standard-Event
Service\ActiveCampaignService	AC API v3: Kontakt, Listen, Tags
Service\AccountNewsletterListsService	Account speichern, Resend DOI
Service\NewsletterDoiConsentLogService	Audit nach var/log/itk_newsletter_doi.log
Service\NewsletterSubscribeDiagnostics	sichere Log-Fingerprints
Storefront\Controller\AccountNewsletterLists Controller	/account/newsletter-lists GET/POST, resend DOI
Storefront\Controller\AccountNewsletterLists WidgetController	Widget-Fragment
Storefront\Controller\ActiveCampaignNewsletter WebhookController	Webhook Abmeldung

Klasse	Rolle
Storefront\Controller\NewsletterFormController	Logik für CMS-Newsletter-Submit (wird vom Interceptor aufgerufen)
Storefront\Twig\NewsletterListsTwigExtension	**newsletterLists()**, **newsletterVisibleLists()**
Subscriber\NewsletterFormControllerInterceptor	leitet Newsletter-Form-Requests auf Plugin-Controller um
Subscriber\NewsletterActiveCampaignSubscriber	Confirm / Unsubscribe → AC + Felder
Subscriber\NewsletterConfirmRedirect Subscriber	Redirect nach DOI aus Session
Subscriber\AccountNewsletterListsSubscriber	Account-Layout erweitern
Subscriber\AccountProfileNewsletterWidget Subscriber	Profil-Widget einbinden
Subscriber\NewsletterFriendlyCaptchaReset Subscriber	Captcha-Kompatibilität
Subscriber\NewsletterRegisterMailListsEnrich Subscriber	reichert Register-/Mail-Flow nach dem Core-Recipient-Schreiben an
Command\ActiveCampaignContactTags Command	CLI itk:ac:contact-tags
Core\Newsletter\SalesChannel\ListDoiNewsletter SubscribeRouteDecorator	Decorator für **NewsletterSubscribeRoute** (Multi-Listen, DOI, AC)
Service\NewsletterRegisterRecipientEnrichment Buffer	Puffert Listen-/Consent-Daten für nachgelagerte Enrichment-Schritte
Service\NewsletterSubscribeDebugLogService	zusätzliche technische Diagnose für Subscribe-/Register-Flows
Exception\NewsletterAccountConsentRequired Exception	fachliche Exception, wenn im Konto zusätzlicher Consent erforderlich ist

---

7 Custom Fields (`newsletter_recipient`)

Die vier wichtigsten Felder auf `newsletter_recipient.customFields`:

- `newsletter_selected_lists`: Was der Nutzer im Formular oder im Account aktuell ausgewählt hat.

- `newsletter_confirmed_lists`: Listen, die nach DOI bzw. bestätigtem Opt-in wirklich aktiv sind.
- `newsletter_pending_lists`: Listen, die angefragt wurden, aber noch auf DOI-Bestätigung warten.
- `newsletter_app_tracking_active`: Boolean-Flag für das zusätzliche App-/Tracking-Opt-in.

Kategorie: `itk_nl_category_list_id` (Select, Migration **Migration1746200000…**)

---

8 Ablauf beim Abonnieren

8.1 1. Storefront- oder CMS-Formular

Ein Formular liefert E-Mail, Consent und eine oder mehrere Listen-Keys. Bei CMS-Newsletter-Formularen kann der **NewsletterFormControllerInterceptor** den Request auf den plugin-eigenen **NewsletterFormController** umlenken, damit die Multi-Listen-Logik auch dort greift.

8.2 2. Decorator auf NewsletterSubscribeRoute

Der **ListDoiNewsletterSubscribeRouteDecorator** ist die zentrale Schaltstelle:

- liest die angeforderten Listen aus der **RequestDataBag**
- filtert auf wirklich konfigurierte und im Storefront zulässige Listen
- baut die drei Recipient-Felder **selected / confirmed / pending**
- übernimmt optional das App-Tracking-Consent in `newsletter_app_tracking_active`
- ruft bei bestehenden Empfängern Unsubscribe-Logik für Listen auf, die entfernt wurden
- delegiert danach an die originale **NewsletterSubscribeRoute**

8.3 3. DOI-Bestätigung

Wenn der Empfänger den DOI-Link klickt, reagiert **NewsletterActiveCampaignSubscriber** auf **NewsletterConfirmEvent**. Ab dann werden pending-Listen in den bestätigten Zustand überführt und mit **ActiveCampaignService** synchronisiert.

8.4 4. Account-Verwaltung

Über **AccountNewsletterListsController** und **AccountNewsletterListsService** kann der eingeloggte Kunde seine Listen anpassen und DOI bei Bedarf neu anstoßen. Das Widget und die Konto-Seite lesen dabei denselben Custom-Field-Zustand wie der Subscribe-Flow.

8.5 5. Webhook-Abmeldung

Wenn ActiveCampaign eine Abmeldung meldet, nimmt **ActiveCampaignNewsletterWebhookController** die Anfrage an, prüft das Secret und überführt die Listen auf Shopware-Seite in den passenden Zustand.

---

9 Migrationen (`src/Migration/`)

- **Migration1746000000AddNewsletterCustomFields**
- **Migration1746100000AddNewsletterListStateFields**
- **Migration1746200000CategoryNewsletterListSelectField**
- **Migration1746210000AddNewsletterRecipientAppTrackingField**

---

10 Wichtige Routen

Name	Pfad / Methode
frontend.account.newsletter.lists.page	**GET /account/newsletter-lists**
frontend.account.newsletter.lists.save	**POST /account/newsletter-lists**
frontend.account.newsletter.lists.resendDoi	**POST /account/newsletter-lists/resend-doi**
frontend.account.newsletter.lists.widget	**GET /account/newsletter-lists-widget**

Name	Pfad / Methode
frontend.itk.ac.newsletter.unsubscribe_webhook	**POST /itk-ac/newsletter/unsubscribe-webhook**

Webhook: **_loginRequired: false**, Secret-Pflicht.

---

11 Webhook: Sicherheit und Verhalten

Der Unsubscribe-Webhook ist bewusst ohne Storefront-Login erreichbar, wird aber über das konfigurierte **shared secret** abgesichert. Zwei Übergabewege sind gültig:

- Query-Parameter `?secret=...`
- HTTP-Header **X-Itk-Newsletter-Webhook-Secret**

Typische Fehlerbilder:

- **403/401-artiges Verhalten**: Secret fehlt oder stimmt nicht.
- `400`: Payload liefert keine verwertbare E-Mail oder die Datenstruktur ist unvollständig.
- `503`: Shopware ist für AC zwar erreichbar, aber die AC-Konfiguration selbst ist unvollständig.

Für den operativen Betrieb heißt das: Wenn der Webhook „nichts macht“, zuerst Secret, Payload und die ankommende E-Mail prüfen, erst danach die eigentliche AC-Synchronisierung.

---

12 Wichtige Services im Detail

12.1 NewsletterListConfigService

Liest und normalisiert das JSON aus **newsletterLists**. Hier entscheidet sich:

- welche Listen überhaupt existieren
- welche Listen per **hidden** im Storefront unsichtbar sind
- welche **AC-List-ID** zu welchem **key** gehört
- welche **redirectUrl** oder **templateId** an einer Liste hängen

12.2 ListDoiOrchestratorService

Das ist die fachliche Kernlogik des Plugins. Die Klasse berechnet:

- **toAdd** / **toRemove** zwischen vorhandenen und neu angefragten Listen
- den Inhalt von `newsletter_selected_lists`
- den Inhalt von `newsletter_confirmed_lists`
- den Inhalt von `newsletter_pending_lists`
- das App-Tracking-Flag

Die Klasse ist deshalb der richtige Einstiegspunkt, wenn du falsche Listen-Zustände debuggen musst.

Wichtige Methoden im Alltag:

- **`getListDelta()`**: berechnet **toAdd**, **toRemove** und den bestätigten Ausgangszustand
- **`buildCustomFieldsForRecipient()`**: schreibt **selected** / **confirmed** / **pending** in die Recipient-Custom-Fields
- **`getRequestedListsFromDataBag()`**: liest die Listen aus Form-/POST-Daten
- **`applyAppTrackingConsentFromDataBag()`**: übernimmt die zweite Consent-Checkbox
- **`getConfirmedListsWithFallback()`**: schützt ältere Datenbestände vor leeren `confirmed_lists`

12.3 DoiMailSendService

Versendet DOI-Mails entweder über ein listenbezogenes Template oder über den Shopware-Standard. Wenn du wissen willst, warum bei einer Liste ein anderes DOI-Mail rausgeht als bei einer anderen, landest du hier.

12.4 AccountNewsletterListsService

Verarbeitet Änderungen aus dem Kundenkonto. Technisch ist das nicht nur ein simples Upsert, sondern ein Abgleich zwischen aktuellem Recipient-Zustand, neu ausgewählten Listen, DOI-Pending und ggf. AC-Unsubscribe.

Hier entsteht auch ein wichtiger fachlicher Sonderfall: Wenn für eine Änderung zusätzlicher Consent nötig ist, wird **NewsletterAccountConsentRequiredException** geworfen. Der Controller fängt das ab und überführt es in ein sinnvolles Storefront-Verhalten statt in einen blanken Laufzeitfehler.

12.5 NewsletterDoiConsentLogService

Schreibt strukturierte Audit-Ereignisse in `var/log/itk_newsletter_doi.log`. Typische Event-Typen sind:

- `EVENT_DOI_CONFIRM`
- `EVENT_ACCOUNT_LIST_DELTA`
- `EVENT_AC_WEBHOOK_UNSUBSCRIBE`

Wenn du Support für „Warum ist diese Liste plötzlich weg?“ machst, ist das fast immer die erste Logdatei.

12.6 NewsletterSubscribeDiagnostics

Erzeugt sichere Debug-Informationen für den Subscribe-Flow, ohne blind alle personenbezogenen Inhalte in Logs zu kippen.

12.7 Buffer-/Enrichment-Pfad für Register- und Sonderfälle

Neben Decorator und Account-Service gibt es im Code noch einen zweiten technischen Pfad:

- **NewsletterRegisterRecipientEnrichmentBuffer**
- **NewsletterRegisterMailListsEnrichSubscriber**
- **NewsletterSubscribeDebugLogService**

Dieser Teil ist relevant, wenn Recipient-Daten erst **nach** dem normalen Core-Flow nachangereichert werden, zum Beispiel bei Register- oder Formularvarianten, bei denen die Listen- oder Consent-Informationen nicht in exakt derselben Form wie im Standard-Subscribe-Flow ankommen. Wenn DOI-Mails, Recipient-Custom-Fields oder Listen-Zustände „eine Anfrage später“ richtig werden, suchst du hier.

---

13 CLI

13.1 `itk:ac:contact-tags`

Technischer Zweck: Kontakt- und Tag-Zustand eines Empfängers in ActiveCampaign prüfen. Relevant, wenn Shopware-Zustand und AC-Zustand auseinanderlaufen.

Typisch im Einsatz:

- E-Mail-Adresse als Pflichtargument
- optionaler `--sales-channel-id`, damit die richtige AC-Konfiguration genutzt wird

Der Command ist damit ein Debug-Werkzeug für Live-Daten, nicht nur ein „Info-Befehl“.

14 Templates (Auswahl)

- CMS Newsletter-Form: `Resources/views/storefront/element/cms-element-form/form-types/newsletter-form.html.twig`
- Listen-Checkboxen: `component/newsletter-list-selection.html.twig`
- Account: `page/account/newsletter/index.html.twig`
- Sidebar-Link: `page/account/sidebar.html.twig` (Plugin-Erweiterung)

---

15 Logging & Debugging

- `var/log/itk_newsletter_doi.log` – strukturierte Audit-Ereignisse (`EVENT_DOI_CONFIRM`, `EVENT_ACCOUNT_LIST_DELTA`, `EVENT_AC_WEBHOOK_UNSUBSCRIBE`, …)
- Shopware-Logs mit Plugin-Präfix `[itk-ac]` bzw. Exceptions aus **ActiveCampaignService** helfen bei API-Fehlern
- Bei „Liste bleibt pending“ immer drei Dinge zusammen prüfen: Recipient-Custom-Fields, DOI-Bestätigung, AC-Request
- Bei „Liste wird nicht angezeigt“ zuerst **newsletterLists**-JSON, dann **hidden**, dann das Kategorie-Custom-Field `itk_nl_category_list_id`
- Bei „Konto-Änderung springt zurück / speichert nicht“ zusätzlich prüfen, ob **NewsletterAccountConsentRequiredException** fachlich greift
- Bei Register-/CMS-Sonderfällen zusätzlich Buffer-/Enrichment-Pfad und Debug-Log-Service ansehen

---

16 End-to-End-Ablauf

1. Ein Newsletter-Formular im Storefront liefert eine oder mehrere Listen-Keys, optional App-Tracking-Consent und die Empfänger-Daten.
2. Der **Decorator** auf **NewsletterSubscribeRoute** normalisiert die gewünschten Listen, filtert auf wirklich erlaubte Listen und baut daraus die Custom-Fields-Struktur für den Recipient.
3. Für neue Listen landet der Kontakt zunächst in **pending**; bestehende bestätigte Listen bleiben unter **confirmed** erhalten.
4. Bei DOI-Bestätigung reagiert **NewsletterActiveCampaignSubscriber** auf **NewsletterConfirmEvent** und synchronisiert die bestätigten Listen zu **ActiveCampaign**.
5. Bei Abmeldungen oder Account-Änderungen werden Listen-Deltas berechnet; Listen, die entfallen, werden über **ActiveCampaignService** auch in AC deaktiviert.
6. Das Konto-UI und das Widget lesen denselben Listen-Zustand wieder aus den Recipient-Custom-Fields aus.

---

17 Erweiterungspunkte

- **Neue Listenquelle:** Wenn du zusätzliche Listen anbieten willst, gehst du praktisch immer über **NewsletterListConfigService** und die Kategorie-/Config-Logik, nicht direkt über hartcodierte Twig-Checkboxen.
- **Andere DOI-Mail:** Der richtige Einstiegspunkt ist **DoiMailSendService**. Dort entscheidet das Plugin, ob ein eigenes Template oder der Shopware-Standard genutzt wird.
- **Andere AC-Synchronisierung:** Änderungen an Kontakt-/Listen-Calls gehören in **ActiveCampaignService**, nicht in Controller oder Subscriber.
- **Andere Formularlogik:** Für CMS-Newsletter-Formulare ist der **NewsletterFormControllerInterceptor** der zentrale Hook; für den eigentlichen Subscribe-Flow bleibt der Decorator auf **NewsletterSubscribeRoute** die Kernstelle.
- **Eigene Reaktion auf DOI oder Unsubscribe:** Zusätzliche Subscriber auf **NewsletterConfirmEvent** oder **NewsletterUnsubscribeEvent** sind der saubere Weg, statt den Decorator mit Seiteneffekten aufzuladen.