AutoAssignCustomerGroups

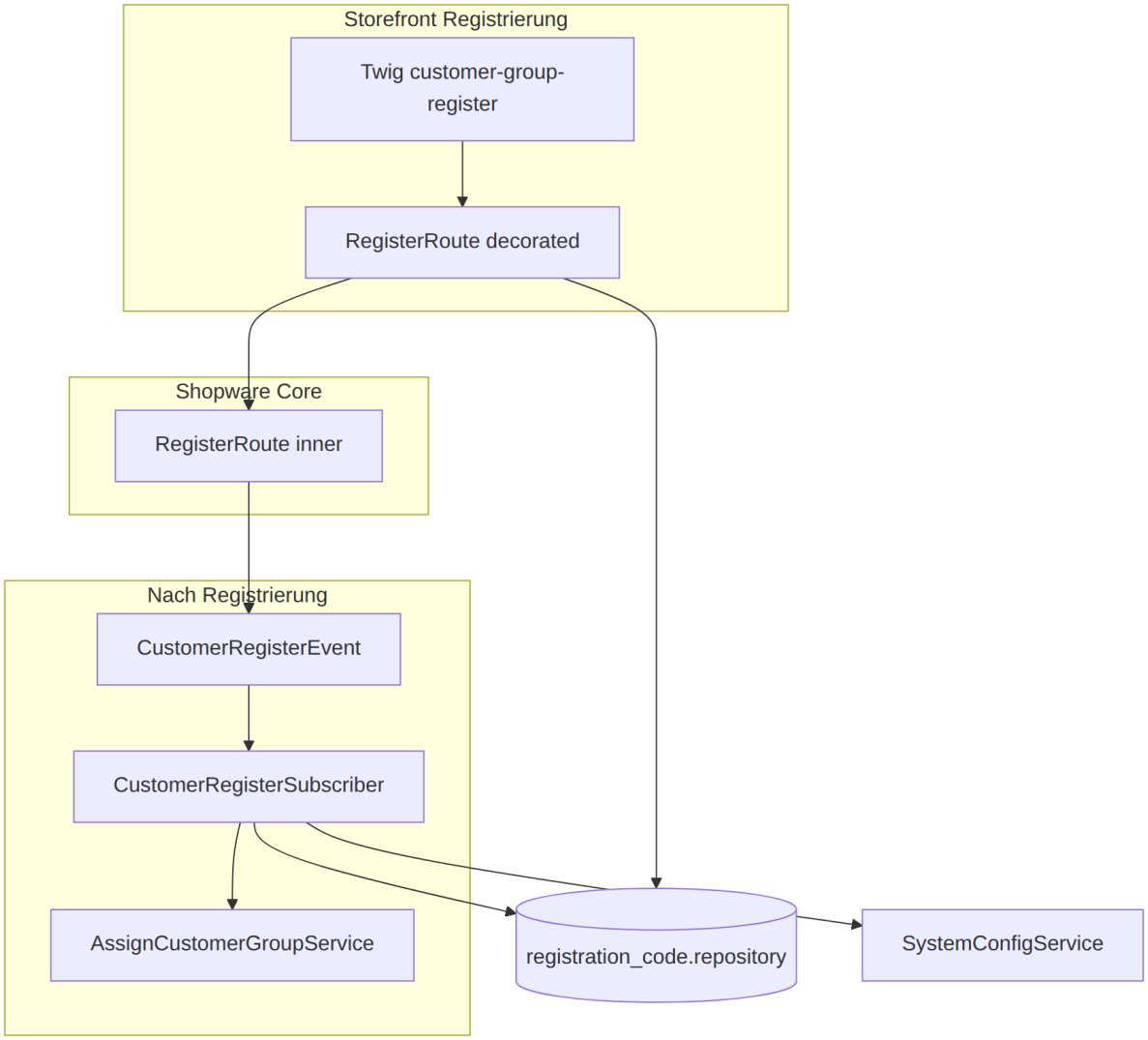
Entwicklerdokumentation

André Fischer

30.03.2026

Inhaltsverzeichnis

1 Architekturüberblick



1. **Twig** blendet das Feld **rcode** ein, wenn die **Kundengruppe** das Custom Field `custom_itk_computer_gmbh_use_registration_codes` gesetzt hat.
2. **RegisterControllerExtension** (Decorator auf **`Shopware\\Core\\Checkout\\Customer\\SalesChannel\\RegisterRoute`**) validiert **rcode** gegen die DAL-Entity `registration_code` und schreibt bei Erfolg **customerEmail** und **usedAt**, **bevor** die innere Route registriert.
3. **CustomerRegisterSubscriber** lauscht auf **CustomerRegisterEvent**, prüft die **angeforderte Kundengruppe** auf `custom_itk_computer_gmbh_auto_assign_customer_group`, weist dann per `customer.repository` die Gruppe final zu, optional **HTTP POST** (**callApiUrlPost**), optional **RedirectResponse** aus **SystemConfigService**

(`AutoAssignCustomerGroups.config.redirectUrl`), und aktualisiert ggf. erneut einen `registration_code`-Datensatz per Suche auf **customerEmail**.

---

2 Datenmodell

2.1 Tabelle `registration_code`

Angelegt/ergänzt durch Migrationen **Migration1727764775**, **Migration1727764776** (Spalten: **code**, `school_name`, `project_name`, `created_at`, `used_at`, `updated_at`). `customer_email` wird in diesen Migrationen nicht angelegt** – das Feld existiert aber in der DAL-Definition (siehe unten). Ohne zusätzliche Migration oder manuelles **ALTER TABLE** können Schreibzugriffe auf **customerEmail** zur Laufzeit scheitern.

DAL: RegistrationCodeDefinition – Entity-Name `registration_code`, Repository-Alias `registration_code.repository`.

Feld (DB/API)	Property (Entity)	Hinweis
id	**id**	UUID
code	**code**	Pflicht
school_name	**schoolName**	Pflicht
project_name	**projectName**	optional
`created_at`	**createdAt**	Pflicht
`used_at`	**usedAt**	null = unbenutzt
customer_email	**customerEmail**	in **RegistrationCode Definition** definiert, in **keiner** der beiden Standard-Migrationen als Spalte angelegt → Projekt-Verantwortung

---

3 Custom Fields (Kundengruppe)

In `AutoAssignCustomerGroups::createCustomField()` wird ein Set mit Relation `customer_group` angelegt (Label u. a. „BYOD zuweisung“):

Name	Typ
custom_itk_computer_gmbh_auto_assign_customer_group	bool
custom_itk_computer_gmbh_use_registration_codes	bool
custom_itk_computer_gmbh_call_api_url	string

Hinweis: Die Methode löscht per **name** ein Set und legt es neu an, wenn neue Felder fehlen – beim Review von Updates Vorsicht bzgl. Datenhaltung.

---

4 RegisterControllerExtension (Decorator)

Klasse: `ItkComputerGmbh\\AutoAssignCustomerGroups\\Controller\\RegisterControllerExtension`
Decorates: `Shopware\\Core\\Checkout\\Customer\\SalesChannel\\RegisterRoute`
Datei: `src/Controller/RegisterControllerExtension.php`

Ablauf `register()`:

1. Fehlt **storefrontUrl** im **RequestDataBag**, wird die URL aus der ersten Sales-Channel-Domain gesetzt (hilft bei Validierung der Storefront-URL).
2. Wenn **rcode** im Bag ist:
   - Suche per **EqualsFilter** auf **code**.
   - Kein Treffer → **\Exception('Invalid registration code.')**
   - **usedAt !== null** → **\Exception('Registration code already used.')**
   - Sonst **update** mit **customerEmail** (aus **email** im Bag) und **usedAt = now**.
3. Aufruf `$this->getDecorated()->register(...)`.

Konsequenz: Die Markierung „verwendet“ passiert **vor** dem erfolgreichen Abschluss der inneren Registrierung. Scheitert danach noch die Kundenanlage, ist der Code ggf. bereits mit **usedAt**/E-Mail belegt – ein bekanntes Ablaufrisiko für fachliche Prozesse.

---

5 AssignCustomerGroupService

Datei: `src/Service/AssignCustomerGroupService.php`

- **confirmCustomerGroup(CustomerEntity, Context): bool**
  Liest **requestedGroupId**, lädt die `customer_group`, prüft `custom_itk_computer_gmbh_auto_assign_customer_group`. Bei Verstoß oder fehlender Gruppe wird eine **Exception** geworfen (nicht **false** zurückgegeben).
- **assignCustomerGroupToUser(CustomerEntity, Context): void**
  Setzt **groupId** auf **requestedGroupId** und leert **requestedGroupId** per `customer.repository->update`.
- **callApiUrlPost(string $url, array $data): ?string**
  JSON-POST via curl, Header **Content-Type** und **Accept: application/json**. HTTP ≥ 400 wirft Exception.

---

6 CustomerRegisterSubscriber

Datei: `src/Subscriber/CustomerRegisterSubscriber.php`
Event: `CustomerRegisterEvent::class` → **onCustomerRegister**

Ablauf (vereinfacht):

1. **confirmCustomerGroup** → bei Erfolg **assignCustomerGroupToUser**.
2. Optional: API-Call, wenn an der **Kundengruppe** `custom_itk_computer_gmbh_call_api_url` gesetzt ist (Payload aktuell **customerId**).
3. Wenn `SystemConfigService->get('AutoAssignCustomerGroups.config.redirectUrl')` nicht leer: `RedirectResponse->send()` (beendet die Response – beachte Auswirkungen auf den restlichen Request).
4. Suche `registration_code` per **EqualsFilter** auf **customerEmail**, dann erneutes **update** von **usedAt**.

---

7 Storefront

Template: `src/Resources/views/storefront/component/account/customer-group-register.html.twig`
Erweitert `@Storefront/storefront/component/account/customer-group-register.html.twig`, Block `component_account_register_address`.
Bedingung: `page.getGroup().customFields['custom_itk_computer_gmbh_use_registration_']`.
Feldname: **rcode** (required).

Snippets: unter `src/Resources/snippet/` (Storefront-Übersetzungen für Titel/Label/Placeholder).

---

8 Administration

Einstieg: `src/Resources/app/administration/src/main.js` → Modul **registration-code**.
`index.js`: Navigation **parent: 'sw-customer'**, Entity `registration_code`, Routen **list**, **create**, `edit/:id`, **generate**.

---

9 Plugin-Konfiguration

`config.xml`: ein Feld **redirectUrl**.
Zugriff im Code: `AutoAssignCustomerGroups.config.redirectUrl` (Shopware-Konvention: `<PluginName>.config.<fieldName>`).

---

10 CLI-Befehle

Alle unter `src/Command/`, registriert mit Tag `console.command`:

Klasse	Name
ListRegistrationCodesCommand	itk:registration-code:list
CreateRegistrationCodeCommand	itk:registration-code:create
EditRegistrationCodeCommand	itk:registration-code:edit
DeleteRegistrationCodeCommand	itk:registration-code:delete
GenerateRegistrationCodesCommand	itk:registration-code:generate
ExportRegistrationCodesCommand	itk:registration-code:export
StatisticsRegistrationCodesCommand	itk:registration-code:stats
SetupRegistrationCodesCommand	itk:registration-code:setup
TestCommand	`itk:test`

CreateRegistrationCodeCommand verlangt mindestens `--school` (oder interaktiv); Code kann automatisch generiert werden.

---

11 Logging

Services `custom_logger` / `custom_log_handler` schreiben nach:

`%kernel.project_dir%/var/log/auto_assign_customer_groups.log`

Level laut Argument `200` (= INFO) im **StreamHandler**.

---

12 Symfony-Building-Blocks in diesem Plugin

Konzept	Wo es vorkommt
Service-Definition	src/Resources/config/services.xml
Decorator	**RegisterControllerExtension** mit **decorates="...RegisterRoute"** und `.inner`
Event Subscriber	**CustomerRegisterSubscriber** + Tag kernel.event_subscriber

Konzept	Wo es vorkommt
Entity Definition	Tag shopware.entity.definition auf **RegistrationCodeDefinition**
SystemConfig	`config.xml` + **SystemConfigService**