Sales channel – Verkaufskanäle – Forum-slack – 23-10-2020

Shopware 6 Sales channel – Verkaufskanäle – Forum-slack – 23-10-2020

[expand title="mehr lesen…"]

Hendrik11:20 Uhr

What is the best way to create a SalesChannelContext, eg. when using a CLI command? I loaded a SalesChannelEntity and used SalesChannelContext::createFrom($loadedSalesChannelEntity), but the SalesChannelContext generated this way is missing the generic Context and doesn't contain a setter to add one.
What is the best way to add it there?

there is a sales channel context factory service

```
$salesChannelContext                                    =
$this->salesChannelContextFactory->create(
    Uuid::randomHex(),
    $bid->getSalesChannel()->getId(),
    [
            SalesChannelContextService::LANGUAGE_ID =>
$bid->getLanguage()->getId(),
            SalesChannelContextService::CUSTOMER_ID =>
$bid->getCustomer()->getId(),
             SalesChannelContextService::COUNTRY_ID =>
$bid->getCustomer()->getDefaultBillingAddress()->getCountry()-
```

```
>getId()
    ]
);
```

[/expand]

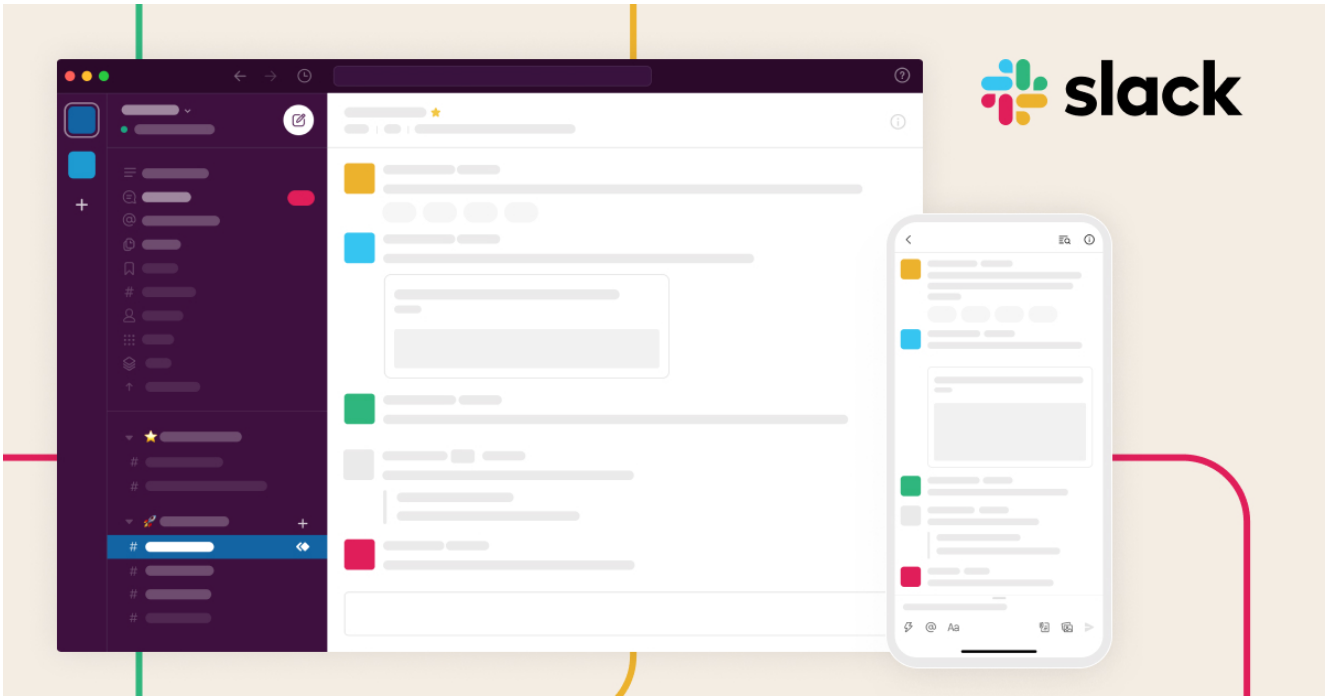
---

roles and permissions – ACL – https://slack.shopware.com/

roles and permissions – ACL – https://slack.shopware.com/

[expand title="mehr lesen…"]

Hannes Wernery 13:40 Uhr
Hey! The ACL system is supposed to be upwards and downwards compatible https://hi.shopware.com/ACL
But as I see it, if I activate this feature and set up a role for a user, any models that are not listed in the role/permission management are *not allowed*. Meaning: entities that are not explicitly listed and activated in the management cannot be used by that user. This requires changes in our plugin.
code: „0“
detail: „Missing privilege my_custom_model:read“
meta: {,…}
status: „403“
title: „Forbidden“
This is not quite downwards compatible, is it? Or am I missing something?



Join Shopware Community

Slack is where work flows. It’s where the people you need, the information you share, and the tools you use come together to get things done.
[Meteor] Brent Robert10:55 UhrWe seem to be having an issue with a template of the B2B suite:This is default shopware:

```
{% block   page_checkout_confirm_address_billing_actions_link
%}
<a   href="{{   path('frontend.account.address.edit.page',
{'addressId': billingAddress.id}) }}"
title="{{ "account.overviewChangeBilling"|trans|striptags }}"
class="btn btn-light"
data-address-editor="true"
data-address-editor-options='{{
addressEditorOptions|json_encode }}'>
{{ "account.overviewChangeBilling"|trans|sw_sanitize }}
</a>
{% endblock %}
```

This is B2B suite:

```
{% block   page_checkout_confirm_address_shipping_actions_link
%}
{% if b2bSuite %}
```

```
<a
title="{{ "account.overviewChangeShipping"|trans|striptags }}"
class="btn     btn-light     ajax-panel-link     {{
b2b_acl('b2bcontactaddress', 'list') }}"
data-target="address-select"
href="{{ path('frontend.b2b.b2baddressselect.index', {'type':
'shipping', 'selectedId': shippingAddress.id}) }}"
>
{{ "account.overviewChangeShipping"|trans|sw_sanitize }}
</a>
{%  else %}
{{ parent() }}
{%  endif %}
{% endblock %}
```

The B2B suite is losing the data-addresss-editor(-options) tags and breaking the layout?

[/expand]

---

shopware6-rule builder

Shopware 6 – Rule Builder

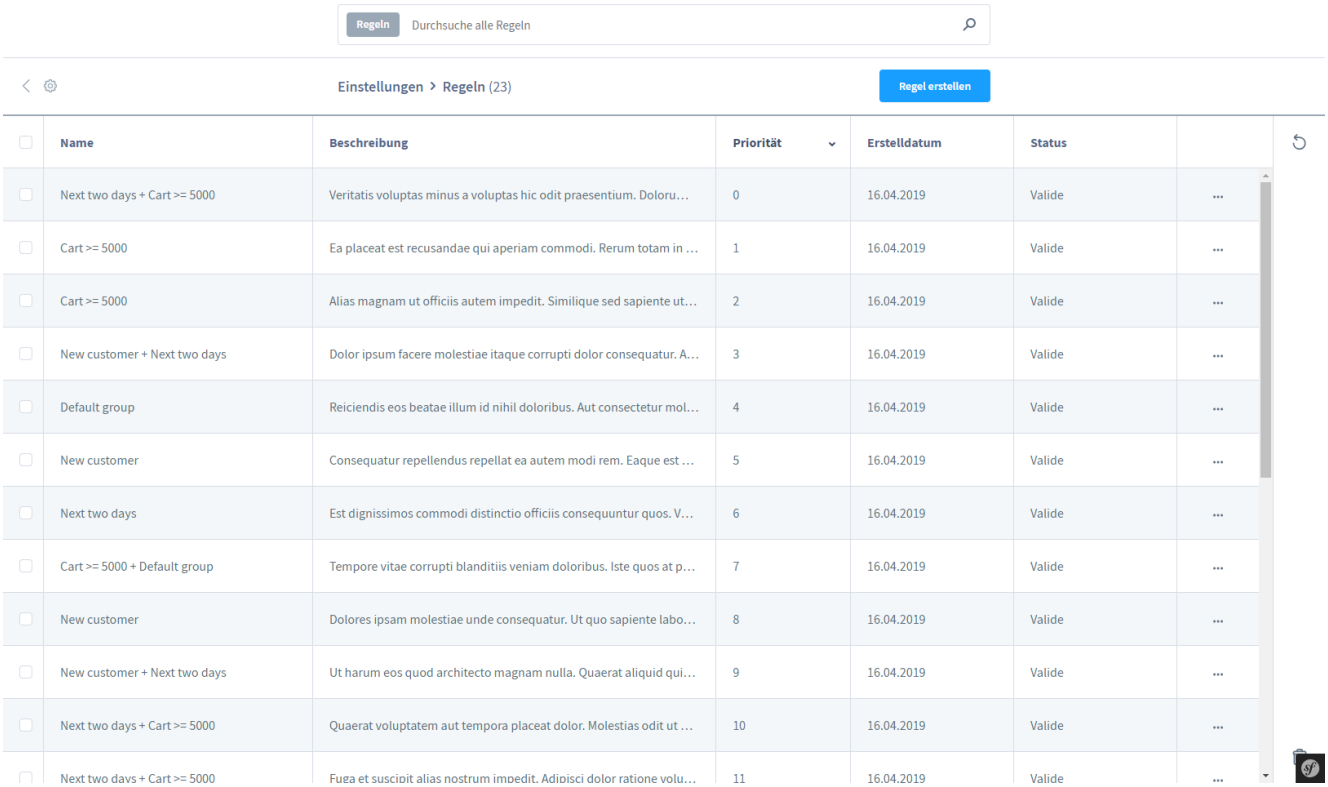
Der Rule Builder bietet Dir die einfache Möglichkeit, auch komplexe Logiken abzubilden. Diese Regeln kannst Du z.B. in den Versandkosten und Zahlungsarten verwenden, um diese nur unter bestimmten Voraussetzungen anzubieten.

[expand title=“mehr lesen…“]

Rule Builder

Über den in Shopware 6 enthaltenen **Rule Builder** kannst Du anhand von Bedingungen individuelle Regeln erstellen, die z.B. für die Berechnung von Versandkosten oder von kundenspezifischen Produktpreisen verwendet werden können.

Übersicht



Beim Aufruf des Menüpunkts **Einstellungen > Shop > Rule Builder** erhältst Du eine Übersicht über alle bereits angelegten Regeln. Die Übersicht ist in mehrere Spalten aufgeteilt, die Dir direkt die folgenden Informationen bieten:

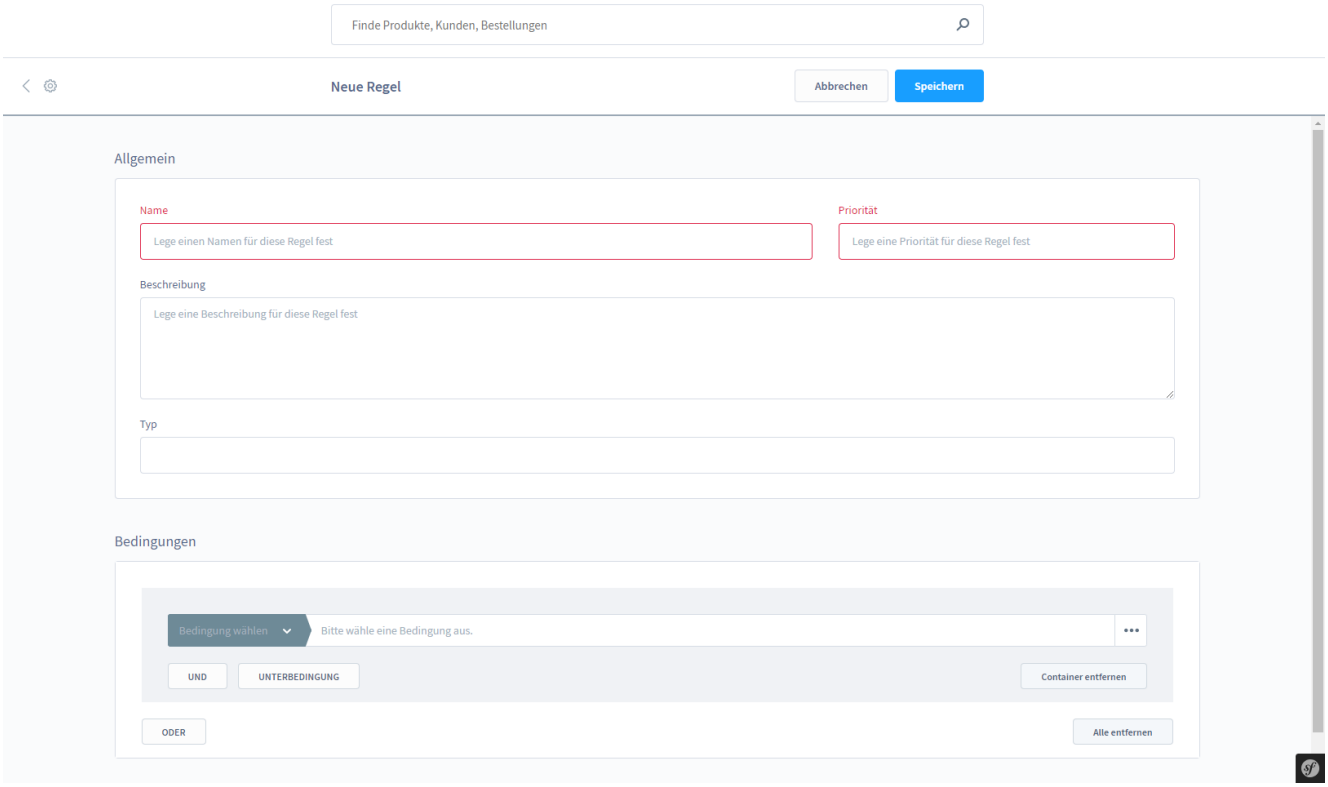
- **Name (1)** Enthält den für die Regel hinterlegten Namen. Dieser wird z.B. in der Administration für die Zuweisung der Regel in anderen Modulen verwendet
- **Beschreibung (2)** Zeigt die (optionale) Beschreibung der Regel, die ggfs. mehr Informationen zum Anwendungsfall enthält.
- **Priorität (3)** Die Priorität gibt an, ob eine Regel ggfs.

vor einer anderen Regel angewandt wird. Je höher die Priorität, umso eher wird diese ausgeführt.
- **Erstelldatum (4)** An diesem Datum wurde die Regel angelegt.
- **Status (4)** Gibt an, ob die Regel Valide ist und verwendet werden kann. Über die Administration angelegte Regeln werden hierhingehend geprüft und können nur gespeichert werden, wenn alle Pflichtfelder mit validen Werten gefüllt sind.

Regel anlegen

Hier zeigen wir Dir, wie Du Regeln erstellst und welche Bedingungen und Operatoren Dir dafür zur Verfügung stehen.
Außerdem findest Du am Ende diverse Beispiele für gängige Szenarien, die Du als Grundlage für Deine eigenen Regeln verwenden kannst.

Eine Regel besteht aus mindestens einer **Bedingung**, kann aber auch mehrere Bedingungen enthalten die über eine **UND / ODER** – Verknüpfung verbunden werden. Außerdem besteht die Möglichkeit, eine **Unterbedingung** hinzuzufügen.
Beim Erstellen der Bedingung stehen dir je nach Bedingung diverse **Operatoren** zur Verfügung, um diese näher definieren zu können.
Um eine neue Regel anzulegen, klicke auf den Button **Regel erstellen**.
Anschließend öffnet sich die folgende Maske.



Allgemein

In den Allgemeinen Einstellungen der Regel definierst Du einen **Namen** und die **Priorität** der Regel.

Über die Priorität kannst Du bei mehreren existierenden Regeln definieren, welche Regel als erstes ausgeführt werden soll. Je höher der hinterlegte Wert im Vergleich zu den anderen Regeln, umso eher wird die Regel ausgeführt. (Eine Regel mit Priorität 5 wird vor einer Regel mit Priorität 3, aber nach einer Regel mit Priorität 9 ausgeführt)

Zusätzlich kannst Du einen **Beschreibungstext** hinterlegen, um z. B. zu erläutern, wofür diese Regel verwendet wird.
Diese Beschreibung wird nur in der Administration angezeigt und ist nicht im Frontend sichtbar.

Außerdem kannst Du den **Typ** der Regel festlegen um diese Regel nur in bestimmten Programmmodulen verfügbar zu machen, z.B. für die Versandkosten.

Operatoren

Die meisten Bedingungen werden über zusätzliche Operatoren näher definiert.

Die verfügbaren Operatoren variieren zwischen den einzelnen Bedingungen und nicht bei jeder Bedingung stehen alle Optionen zur Verfügung.

Im Folgenden erläutern wir Dir zunächst die Funktion der einzelnen Operatoren

- **Gleich**
  Die Bedingung trifft zu, wenn der hinterlegte Wert der Eingabe gleicht
- **Ungleich**
  Die Bedingung trifft zu, wenn der hinterlegte Wert von der Angabe abweicht
- **Ist eine von**
  Die Bedingung trifft zu, wenn einer der hinterlegten Werte mit der Angabe übereinstimmt
- **Ist keine von**
  Die Bedingung trifft zu, wenn keiner der hinterlegten Werte mit der Angabe übereinstimmt
- **Größer**
  Die Bedingung ist erfüllt, wenn der entsprechende Wert größer als der hier definierte Wert ist.
- **Größer gleich**
  Die Bedingung ist erfüllt, wenn der entsprechende Wert dem hier definierten Wert entspricht oder größer ist.
- **Kleiner**
  Die Bedingung ist erfüllt, wenn der entsprechende Wert geringer als der hier definierte Wert ist.
- **Kleiner gleich**
  Die Bedingung ist erfüllt, wenn der entsprechende Wert dem hier definierten Wert entspricht oder kleiner ist.

Bedingungen

Es stehen folgende Bedingungen zur Verfügung

Abweichende Adressen

Die Bedingung Abweichende Adressen bezieht sich auf eine von der Rechnungsadresse abweichende Lieferadresse, daher stehen als Optionen „Ja“ oder „Nein“ zur Verfügung.
Geprüft wird hier, ob der Kunde die Option „Lieferadresse weicht von der Rechnungsadresse ab“ aktiviert hat.

Beispiel:
Rechnungsadresse – Musterstraße 123, 12345 Musterstadt
Lieferadresse – Testweg 45, 45678 Testort

- *Ja*
  Die Bedingung trifft zu, da die Rechnungs- und Lieferadresse abweichen
- *Nein*
  Die Bedingung trifft nicht zu, da die Rechnungs- und Lieferadresse abweichen

Anzahl Bestellungen

Prüfe hierüber die Anzahl der bisherigen Bestellungen Deines Kunden

Beispiel: 5

- *Gleich*
  Die Bedingung ist erfüllt, wenn der Kunde bisher 5 Bestellungen durchgeführt hat.
- *Größer*
  Die Bedingung ist erfüllt, wenn der Kunde bisher mehr als 5 Bestellungen durchgeführt hat.
- *Größer gleich*
  Die Bedingung ist erfüllt, wenn der Kunde bisher 5 oder

mehr Bestellungen durchgeführt hat.
- *Kleiner*
  Die Bedingung ist erfüllt, wenn der Kunde bisher weniger als 5 Bestellungen durchgeführt hat
- *Kleiner gleich*
  Die Bedingung ist erfüllt, wenn der Kunde bisher 5 oder weniger Bestellungen durchgeführt hat.
- *Ungleich*
  Die Bedingung ist erfüllt, wenn der Kunde bisher eine andere Anzahl als 5 Bestellungen durchgeführt hat.

Anzahl Warenkorb-Güter

Bei der Anzahl der Warenkorb-Güter kannst Du entweder einfach nur auf die Gesamtzahl der Produkte im Warenkorb prüfen, alternativ stehen Dir aber auch noch wesentlich umfangreichere Zusatzbedingungen zur Verfügung.

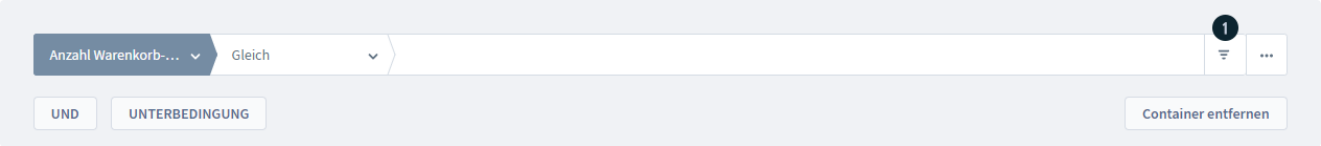
Für die Prüfung der Anzahl stehen Dir diverse Optionen zur Verfügung.

Beispiel: 10

- *Gleich*
  Die Bedingung ist erfüllt, wenn 10 Warenkorb-Güter vorhanden sind.
- *Größer*
  Die Bedingung ist erfüllt, wenn mehr als 10 Warenkorb-Güter vorhanden sind.
- *Größer gleich*
  Die Bedingung ist erfüllt, wenn 10 oder mehr Warenkorb-Güter vorhanden sind.
- *Kleiner*
  Die Bedingung ist erfüllt, wenn weniger als 10 Warenkorb-Güter vorhanden sind.
- *Kleiner gleich*
  Die Bedingung ist erfüllt, wenn 10 oder weniger Warenkorb-Güter vorhanden sind.

- *Ungleich*
  Die Bedingung ist erfüllt, wenn nicht 10 Warenkorb-Güter vorhanden sind.

Die optionalen Zusatzfilter öffnest Du über den Filter-Button (1) auf der rechten Seite. Es öffnet sich dann ein Modal für die Erstellung der untergeordneten Bedingungen.



Mögliche Unterbedingungen sind

- Hersteller
- Neuheiten
- Position des Typs
- Positionen
- Positionen Einkaufspreis
- Positionen Gesamtpreis
- Positionen Preis
- Positionen in Kategorie
- Positionen mit Abverkauf
- Positionen mit Anzahl
- Positionen mit Breite
- Positionen mit Eigenschaften
- Positionen mit Erscheinungsdatum
- Positionen mit Erstelldatum
- Positionen mit Gewicht
- Positionen mit Höhe
- Positionen mit Länge
- Positionen mit Steuersatz
- Positionen mit Streichpreis
- Positionen mit Tag
- Hervorgehobene Positionen

Datumsbereich

Die Bedingung ist innerhalb des gesetzten Datumsbereichs

gültig.
Es wird zwischen den Optionen „Mit Zeitabfrage“ und „Ohne Zeitabfrage“ unterschieden. Beispiel: 2019-05-23 (06:00) – 2019-05-31 (22:00)

- *Ohne Zeitabfrage:* Hier wird lediglich ein Start- und ein Enddatum ausgewählt
  Die Bedingung ist vom 23.05.2019 bis einschließlich 31.05.2019 gültig
- *Mit Zeitabfrage:* Zusätzlich zum Datum wird die Uhrzeit angegeben
  Die Bedingung ist vom 23.05.2019 06:00 Uhr bis einschließlich 31.05.2019 22:00 Uhr gültig

Firmenkunde

Hierüber kannst Du prüfen, ob es sich um einen Firmenkunden handelt.
Dazu stehen Dir die Optionen „Ja“ und „Nein“ zur Verfügung.

Beispiel:

- *Ja*
  Die Bedingung ist erfüllt, wenn es sich um einen Firmenkunden handelt
- *Nein*
  Die Bedingung ist erfüllt, wenn der Kunde kein Firmenkunde ist.

Hersteller

Ermöglicht die Prüfung auf die Hersteller-Angabe der Produkte. Als Optionen stehen „Ist eine von“ und „ist keine von“ zur Verfügung. Diese beiden Optionen ermöglichen aus eine Mehrfachauswahl.

Beispiel: shopware AG

- *Ist eine von*
  Die Bedingung ist erfüllt, wenn der Hersteller die shopware AG ist
- *Ist keine von*
  Die Bedingung ist erfüllt, wenn der Hersteller nicht die shopware AG ist.

Immer zutreffend

Diese Bedingung trifft immer zu, eine tiefergehende Konfiguration ist hierfür nicht notwendig.

Kundengruppe

Wenn Du die Kundengruppe als Bedingung verwendest, kannst Du zwischen den beiden erweiterten Operatoren „Ist eine von“ und „Ist keine von“ wählen.

Beispiel: Nettopreis-Kundengruppe

- *Ist eine von*
  Die Bedingung ist erfüllt, wenn der Kunden der Nettopreis-Kundengruppe zugewiesen ist.
- *Ist keine von*
  Die Bedingung trifft zu, wenn der Kunde einer anderen Kundengruppe als Händler zugewiesen ist.

Kundennummer

Für eine genaue Einschränkung steht die Kundennummer zur Verfügung.
Hier wählst Du zwischen den Operatoren „Ist eine von“ oder „Ist keine von“.

Beispiel: 12345, 23456

- *Ist eine von*
  Die Bedingung trifft zu, wenn es sich um einen der

Kunden mit der Kundennummer 12345 oder 23456 handelt
- *Ist keine von*
  Wenn der Kunde nicht eine der Kundennummern 12345 oder 23456, trifft diese Bedingung zu

Lieferadresse-PLZ

Für die PLZ der Lieferadresse kannst Du die Operatoren „Ist eine von“ und „Ist keine von“ verwenden.

Beispiel: 48612, 48624, 48683

- *Ist eine von*
  Die Bedingung trifft zu, wenn die PLZ der Lieferadresse 48612, 48624 oder 48683 ist.
- *Ist keine von*
  Die Bedingung trifft zu, wenn die PLZ der Lieferadresse nicht 48612, 48624 oder 48683 ist.

Lieferland

So, wie Du auf die Daten der Rechnungsanschrift prüfen kannst, stehen Dir auch die Informationen aus der Lieferanschrift zur Verfügung.
Beim Lieferland definierst Du die Bedingung über die Operatoren „Ist eine von“ oder „Ist keine von“.

Beispiel: Schweiz

- *Ist eine von*
  Die Bedingung ist erfüllt, wenn als Lieferland Schweiz angegeben ist
- *Ist keine von*
  Die Bedingung trifft zu, wenn ein anderes Lieferland als Schweiz gewählt wurde

Lieferstraße

Für die Bedingung Lieferstraße stehen Dir die Operatoren

„Gleich“ oder „Ungleich“ zur Verfügung.

Beispiel: Ebbinghoff 10

- *Gleich*
  Die Bedingung trifft zu, wenn als Lieferstraße Ebbinghoff 10 hinterlegt ist.
- *Ungleich*
  Die Bedingung ist erfüllt wenn die Lieferstraße nicht Ebbinghoff 10 ist.

Nachname

Neben der Kundennummer kannst Du auch auf den Kunden-Nachnamen prüfen.
Dazu kannst Du einer der Optionen „Gleich“ oder „Ungleich“ verwenden.

Beispiel: Müller

- *Gleich*
  Die Bedingung ist erfüllt, wenn ein Kunde mit dem Nachnamen Müller eingeloggt ist.
- *Ungleich*
  Die Bedingung trifft zu, wenn ein Kunde, der nicht Müller heißt, eingeloggt ist.

Neuer Kunde

Du hast hierüber die Möglichkeit, als Bedingung auf Neukunden zu prüfen.

Hierzu stehen die Optionen „Ja“ und „Nein“ zur Verfügung und Du kannst somit z.B. eine Zahlungsart nur für Kunden freigeben, die bereits einmal in Deinem Shop bestellt haben.

Definition Neukunde:
Als Neukunde wird ein Kunde deklariert, bei dem das Datum des ersten Logins gleich dem heutigen Datum ist.

- *Ja*
  Die Bedingung trifft zu, wenn es sich bei dem Kunden um einen Neukunden handelt
- *Nein*
  Wenn der Kunde kein Neukunde ist, trifft diese Bedingung zu

Neuheiten

Über diese Bedingung kannst Du definieren, ob in der Regel eine Prüfung auf Produkt-Neuheiten erfolgen soll.
Als Operatoren stehen hierfür „Ja“ und „Nein“ zur Verfügung.

Damit ein Produkt als Neuheit gekennzeichnet wird, ist es erforderlich, dass das Erscheinungsdatum gepflegt ist und max. 30 Tage in der Vergangenheit liegt.

Beispiel:

- *Ja*
  Die Bedingung ist erfüllt, wenn mindestens ein abgefragtes Produkt als Neuheit deklariert ist.
- *Nein*
  Die Bedingung ist erfüllt, wenn kein Produkt aus der Abfrage als Neuheit gilt.

Position des Typs

Über den Typ eine Position kannst Du abfragen, ob ein Produkt oder ein Rabatt bzw. Aufschlag, z.B. im Warenkorb, vorhanden ist.
Als Operatoren kannst Du „Gleich“ und „Ungleich“ verwenden.

Beispiel: Rabatt / Aufschlag

- *Gleich*
  Die Bedingung ist erfüllt, wenn mindest eine Postion vom Typ Rabatt oder Aufschlag vorhanden ist
- *Ungleich*

Die Bedingung ist erfüllt, wenn keine Postion Rabatt bzw. Aufschlag vorhanden ist

Positionen

Über die Positionen kannst Du bestimmte Produkte prüfen.
Hierzu stehen Dir die Operatoren „Ist eine von“ und „Ist keine von“ zur Verfügung. Eine Mehrfachauswahl von Produkten ist möglich.

Beispiel: Enormous Copper Cubicide

- *Ist eine von*
  Die Bedingung trifft zu, wenn es sich um das Produkt „Enormous Copper Cubicide“ handelt
- *Ist keine von*
  Die Bedingung ist erfüllt, wenn das Produkt „Enormous Copper Cubicide“ nicht in der Abfrage enthalten ist.

Positionen Einkaufspreis

Über diese Bedingung kannst Du den Einkaufspreis von Produkten abfragen. Der Einkaufspreis bezieht sich hier auf den Preis eines einzelnen Produkts. Eine Aufsummierung erfolgt nicht.
Für die Definition stehen Dir diverse Operatoren zur Verfügung.

Beispiel: 20,00

- *Gleich*
  Die Bedingung ist erfüllt, wenn der Einkaufspreis genau 20,00 ist.
- *Größer*
  Die Bedingung ist erfüllt, wenn der Einkaufspreis größer als 20,00 ist
- *Größer gleich*
  Die Bedingung ist erfüllt, wenn der Einkaufspreis 20,00 oder mehr beträgt
- *Kleiner*

Die Bedingung ist erfüllt, wenn der Einkaufspreis geringer als 20,00 ist.

- *Kleiner gleich*
  Die Bedingung ist erfüllt, wenn der Einkaufspreis 20,00 oder weniger ist.
- *Ungleich*
  Die Bedingung ist erfüllt, wenn der Einkaufspreis nicht 20,00 ist.

Positionen Gesamtpreis

Der Gesamtpreis der Positionen bezieht sich auf den Gesamtpreis einer Position im Warenkorb. Wenn Du also mehrmals das gleiche Produkt in den Warenkorb legst, wird der Preis in dieser Bedingung aufsummiert.

Beispiel: 50,00

- *Gleich*
  Die Bedingung ist erfüllt, wenn der Gesamtpreis einer Position genau 50,00 ist.
- *Größer*
  Die Bedingung ist erfüllt, wenn der Gesamtpreis einer Position größer als 50,00 ist
- *Größer gleich*
  Die Bedingung ist erfüllt, wenn der Gesamtpreis einer Position 50,00 oder mehr beträgt
- *Kleiner*
  Die Bedingung ist erfüllt, wenn der Gesamtpreis einer Position geringer als 50,00 ist.
- *Kleiner gleich*
  Die Bedingung ist erfüllt, wenn der Gesamtpreis einer Position 50,00 oder weniger ist.
- *Ungleich*
  Die Bedingung ist erfüllt, wenn der Gesamtpreis einer Position nicht 50,00 ist.

Positionen Preis

Der Preis der Positionen bezieht sich auf den Preis eines einzelnen Produkts. Für den Abgleich stehen Dir verschiedene Operatoren zur Verfügung.

Beispiel: 30,00

- *Gleich*
  Die Bedingung ist erfüllt, wenn der Preis einer Position genau 30,00 ist.
- *Größer*
  Die Bedingung ist erfüllt, wenn der Preis einer Position größer als 30,00 ist
- *Größer gleich*
  Die Bedingung ist erfüllt, wenn der Preis einer Position 30,00 oder mehr beträgt
- *Kleiner*
  Die Bedingung ist erfüllt, wenn der Preis einer Position geringer als 30,00 ist.
- *Kleiner gleich*
  Die Bedingung ist erfüllt, wenn der Preis einer Position 30,00 oder weniger ist.
- *Ungleich*
  Die Bedingung ist erfüllt, wenn der Preis einer Position nicht 30,00 ist.

Positionen im Warenkorb

Hierüber kannst Du mittels der Operatoren *Ist eine von* und *Ist keine von* prüfen, ob ein bestimmtes Produkt im Warenkorb enthalten ist.

Beispiel: Synergistic Aluminum Mainstay

- *Ist eine von*
  Die Bedingung ist erfüllt, wenn das Produkt „Synergistic Aluminum Mainstay“ im Warenkorb vorhanden ist.

- *Ist keine von*
  Die Bedingung ist erfüllt, wenn das Produkt „Synergistic Aluminum Mainstay“ nicht im Warenkorb vorhanden ist.

Positionen in Kategorie

Mittels der Operatoren „Ist eine von“ und „Ist keine von“ kannst Du prüfen, ob ein Produkt einer bestimmten Kategorie zugewiesen ist. Die Struktur des Kategoriebaums wird hierbei berücksichtigt. Wenn Du hier also eine Oberkategorie auswählst, gelten die Bedingungen auch für die darunter befindlichen Kategorien.

Beispiel: Die Kategoriestruktur ist „Katalog #1 > Nahrung > Backwaren“. Das Produkt ist der Kategorie Backwaren zugeordnet. In der Bedingung prüfen wir auf die Kategorie Nahrung.

- *Ist eine von*
  Die Bedingung ist erfüllt, da die Kategorie des Produkts im Kategoriebaum der Kategorie aus der Bedingung untergeordnet ist.
- *Ist keine von*
  Die Bedingung ist nicht erfüllt, da eine Beziehung zu der Kategorie besteht.

Positionen mit Abverkauf

Hierüber kannst Du prüfen, ob ein Produkt als Abverkauf markiert ist. Dazu stehen Dir die Operatoren „Ja“ und „Nein“ zur Verfügung.

- *Ja*
  Die Bedingung ist erfüllt, wenn einer der abgefragten Artikel als Abverkauf markiert ist.
- *Nein*
  Die Bedingung ist erfüllt, wenn keiner der abgefragten Artikel als Abverkauf markiert ist.

Positionen mit Anzahl

Ermöglicht die Prüfung auf die Anzahl eines bestimmten Produkts im Warenkorb. Hierzu wählst Du zum einen das Produkt, das abgefragt werden soll. Zusätzlich legst Du dann einen Operator und die gewünschte Anzahl fest.

Beispiel: Produkt „Sleek Wool Wasabi Fresh“ mit der Anzahl 5

- *Gleich*
  Die Bedingung ist erfüllt, wenn das Produkt „Sleek Wool Wasabi Fresh“ genau 5 mal im Warenkorb vorhanden ist.
- *Größer*
  Die Bedingung ist erfüllt, wenn das Produkt „Sleek Wool Wasabi Fresh“ mehr als 5 mal im Warenkorb vorhanden ist.
- *Größer gleich*
  Die Bedingung ist erfüllt, wenn das Produkt „Sleek Wool Wasabi Fresh“ 5 mal oder mehr im Warenkorb vorhanden ist.
- *Kleiner*
  Die Bedingung ist erfüllt, wenn das Produkt „Sleek Wool Wasabi Fresh“ weniger als 5 mal im Warenkorb vorhanden ist.
- *Kleiner gleich*
  Die Bedingung ist erfüllt, wenn das Produkt „Sleek Wool Wasabi Fresh“ 5 mal oder weniger im Warenkorb vorhanden ist.
- *Ungleich*
  Die Bedingung ist erfüllt, wenn das Produkt „Sleek Wool Wasabi Fresh“ nicht 5 mal im Warenkorb vorhanden ist.

Positionen mit Breite

Für die Prüfung der Breite eines Produkts stehen Dir diverse Operatoren zur Verfügung. Beachte hierbei, dass sich die Breite immer auf ein einzelnes Produkt bezieht. Eine Aufsummierung erfolgt nicht.

Beispiel: 750

- *Gleich*
  Die Bedingung ist erfüllt, wenn ein Produkt genau 750 breit ist.
- *Größer*
  Die Bedingung ist erfüllt, wenn ein Produkt□□□□□□ mehr als 750 breit ist.
- *Größer gleich*
  Die Bedingung ist erfüllt, wenn ein Produkt 750 oder breiter ist.
- *Kleiner*
  Die Bedingung ist erfüllt, wenn ein Produkt schmaler als 750 ist.
- *Kleiner gleich*
  Die Bedingung ist erfüllt, wenn ein Produkt 750 oder schmaler ist.
- *Ungleich*
  Die Bedingung ist erfüllt, wenn ein Produkt □□□nicht 750 breit ist.

Positionen mit Eigenschaften

Mittels der Operatoren „Ist eine von“ und „Ist keine von“ kannst Du Prüfungen auf die Eigenschaften der Produkte durchführen.

Beispiel: XL

- *Ist eine von*
  Die Bedingung ist erfüllt, wenn einem Produkt die Eigenschaft „XL“ zugewiesen ist.
- *Ist keine von*
  Die Bedingung ist erfüllt, wenn keinem Produkt die Eigenschaft „XL“ zugewiesen ist.

Positionen mit Erscheinungsdatum

Ermöglicht die Abfrage des Erscheinungsdatums der Produkte.

Diese Bedingung steht aktuell noch nicht zur Verfügung und wird mit einem der kommenden Updates bereitgestellt.

Positionen mit Erstelldatum

Hierüber kannst Du auf das Erstelldatum prüfen. Das Erstelldatum bezieht sich auf den Zeitpunkt, an dem das Produkt im Shop angelegt wurde.

Diese Bedingung steht aktuell noch nicht zur Verfügung und wird mit einem der kommenden Updates bereitgestellt.

Positionen mit Gewicht

Für die Prüfung des Gewichts eines Produkts stehen Dir diverse Operatoren zur Verfügung. Beachte hierbei, dass sich das Gewicht immer auf ein einzelnes Produkt bezieht. Eine Aufsummierung erfolgt nicht.

Beispiel: 5

- *Gleich*
  Die Bedingung ist erfüllt, wenn das Gewicht eines Produkts genau 5 beträgt.
- *Größer*
  Die Bedingung ist erfüllt, wenn das Gewicht eines Produkts mehr als 5 beträgt.
- *Größer gleich*
  Die Bedingung ist erfüllt, wenn das Gewicht eines Produkts 5 oder mehr beträgt.
- *Kleiner*
  Die Bedingung ist erfüllt, wenn das Gewicht eines Produkts weniger als 5 beträgt.
- *Kleiner gleich*
  Die Bedingung ist erfüllt, wenn das Gewicht eines

Produkts 5 oder weniger beträgt.

- *Ungleich*
  Die Bedingung ist erfüllt, wenn das Gewicht eines Produkts nicht 5 beträgt.

Positionen mit Höhe

Für die Prüfung der Höhe eines Produkts stehen Dir diverse Operatoren zur Verfügung. Beachte hierbei, dass sich die Höhe immer auf ein einzelnes Produkt bezieht. Eine Aufsummierung erfolgt nicht.

Beispiel: 350

- *Gleich*
  Die Bedingung ist erfüllt, wenn ein Produkt genau 350 hoch ist.
- *Größer*
  Die Bedingung ist erfüllt, wenn ein Produkt □□□□□□mehr als 350 hoch ist.
- *Größer gleich*
  Die Bedingung ist erfüllt, wenn ein Produkt 350 oder höher ist.
- *Kleiner*
  Die Bedingung ist erfüllt, wenn ein Produkt kleiner als 350 ist.
- *Kleiner gleich*
  Die Bedingung ist erfüllt, wenn ein Produkt 350 oder kleiner ist.
- *Ungleich*
  Die Bedingung ist erfüllt, wenn ein Produkt□□ nicht 350 breit ist.

Positionen mit Länge

Für die Prüfung der Länge eines Produkts stehen Dir diverse Operatoren zur Verfügung. Beachte hierbei, dass sich die Länge immer auf ein einzelnes Produkt bezieht. Eine Aufsummierung

erfolgt nicht.

Beispiel: 475

- *Gleich*
  Die Bedingung ist erfüllt, wenn ein Produkt genau 475 lang ist.
- *Größer*
  Die Bedingung ist erfüllt, wenn ein Produkt □□□□□□mehr als 475 lang ist.
- *Größer gleich*
  Die Bedingung ist erfüllt, wenn ein Produkt 475 oder länger ist.
- *Kleiner*
  Die Bedingung ist erfüllt, wenn ein Produkt kürzer als 475 ist.
- *Kleiner gleich*
  Die Bedingung ist erfüllt, wenn ein Produkt 475 oder kürzer ist.
- *Ungleich*
  Die Bedingung ist erfüllt, wenn ein Produkt□□ nicht 475 lang ist.

Positionen mit Steuersatz

Die Prüfung der Steuersätze eines Produkts ist über die Operatoren „Ist eine von“ und „Ist keine von“ möglich. Es ist eine Mehrfachauswahl möglich. Hierzu kann aus einem Dropdown der im System angelegten Steuersätze gewählt werden.

- *Ist eine von*
  Die Bedingung ist erfüllt, wenn einem Produkt einer der ausgewählten Steuersätze zugewiesen ist.
- *Ist keine von*
  Die Bedingung ist erfüllt, wenn einem Produkt keiner der ausgewählten Steuersätze zugewiesen ist.

Positionen mit Streichpreis

Ermöglicht die Verwendung des Streichpreises eines Produkts als Bedingung.

Beispiel: 15

- *Gleich*
  Die Bedingung ist erfüllt, wenn der Streichpreis eines Produkt□s□□□□□□ genau 15 beträgt.
- *Größer*
  Die Bedingung ist erfüllt, wenn der Streichpreis eines Produkt□s□□□□□□ mehr als 15 beträgt.
- *Größer gleich*
  Die Bedingung ist erfüllt, wenn der Streichpreis eines Produkt□s□□□□□□ 15 oder mehr beträgt..
- *Kleiner*
  Die Bedingung ist erfüllt, wenn der Streichpreis eines Produkt□s□□□□□□ weniger als 15 beträgt..
- *Kleiner gleich*
  Die Bedingung ist erfüllt, wenn der Streichpreis eines Produkt□s□□□□□□ 15 oder weniger beträgt.
- *Ungleich*
  Die Bedingung ist erfüllt, wenn der Streichpreis eines Produkt□s□□□□□□ nicht 15 beträgt.

Positionen mit Tag

Hierüber kannst Du prüfen, ob einem Produkt eine bestimmte Kennzeichnung (engl. „Tag“) zugewiesen ist. Für die Konfiguration der Bedingung stehen die Operatoren „Ist eine von“ und „Ist keine von“ zur Verfügung. Die Tags werden in der Produktverwaltung hinzugefügt. Informationen hierzu erhältst Du hier.

Beispiel: Testtag1

- *Ist eine von*

Die Bedingung ist erfüllt, wenn einem Produkt der Tag „Testtag1“ zugewiesen ist

- *Ist keine von*
  Die Bedingung ist erfüllt, wenn einem Produkt nicht der Tag „Testtag1“ zugewiesen ist

Positionsanzahl im Warenkorb

Ermöglicht die Prüfung, wie viele Positionen sich im Warenkorb befinden.
Auch dazu stehen Dir diverse Optionen zur Verfügung.

Beispiel: 10

- *Gleich*
  Die Bedingung ist erfüllt, wenn 10 Positionen im Warenkorb vorhanden sind.
- *Größer*
  Die Bedingung ist erfüllt, wenn mehr als 10 Positionen im Warenkorb vorhanden sind.
- *Größer gleich*
  Die Bedingung ist erfüllt, wenn 10 oder mehr Positionen im Warenkorb vorhanden sind
- *Kleiner*
  Die Bedingung ist erfüllt, wenn weniger als 10 Positionen im Warenkorb vorhanden sind
- *Kleiner gleich*
  Die Bedingung ist erfüllt, wenn 10 oder weniger Positionen im Warenkorb vorhanden sind
- *Ungleich*
  Die Bedingung ist erfüllt, wenn nicht 10 Positionen im Warenkorb vorhanden sind.

Preis Warenkorbgüter

Beim Preis der Warenkorb-Güter werden nur die Preise der Produkte zusammengerechnet, es werden keine Rabatte, Gutscheine oder Versandkosten berücksichtigt.

Auch hier kannst Du aus einer Vielzahl an Operatoren wählen

Beispiel: 49,99

- *Gleich*
  Die Bedingung ist erfüllt, wenn der Gesamtpreis der Warenkorb-Güter genau 49,99 beträgt.
- *Größer*
  Die Bedingung ist erfüllt, wenn der Gesamtpreis der Warenkorb-Güter größer als 49,99 ist
- *Größer gleich*
  Die Bedingung ist erfüllt, wenn der Gesamtpreis der Warenkorb-Güter 49,99 oder mehr beträgt
- *Kleiner*
  Die Bedingung ist erfüllt, wenn der Gesamtpreis der Warenkorb-Güter geringer als 49,99 ist.
- *Kleiner gleich*
  Die Bedingung ist erfüllt, wenn der Gesamtpreis der Warenkorb-Güter 49,99 oder weniger ist.
- *Ungleich*
  Die Bedingung ist erfüllt, wenn der Gesamtpreis der Warenkorb-Güter nicht 49,99 ist.

Rechnungsadresse-PLZ

Neben Rechnungsland und -straße kannst Du auch auf die Rechnungsadressen-PLZ prüfen.
Die Bedingung kann entweder als Einschluss (Ist eine von) oder Ausschluss (Ist keine von) konfiguriert werden.

Beispiel: 48612, 48624, 48683

- *Ist eine von*
  Die Bedingung trifft zu, wenn die PLZ der Rechnungsadresse 48612, 48624 oder 48683 ist.
- *Ist keine von*
  Die Bedingung trifft zu, wenn die PLZ der Rechnungsadresse nicht 48612, 48624 oder 48683 ist.

Rechnungsland

Als Bedingung kannst Du ebenfalls auf das Rechnungsland prüfen.
Auch hier definierst Du über „Ist eine von“ oder „Ist keine von“, ob die hinterlegten Länder bei Übereinstimmung gewertet werden oder wenn die Länder nicht zutreffen.

Beispiel: Schweiz, USA

- *Ist eine von*
  Die Bedingung trifft zu, wenn als Rechnungsland Schweiz oder USA hinterlegt ist.
- *Ist keine von*
  Die Bedingung ist erfüllt wenn das Rechnungsland nicht Schweiz oder USA ist.

Rechnungsstraße

Es ist auch möglich, die Rechnungsstraße als Bedingung zu verwenden.
Hier kannst Du zwischen den Operatoren „Gleich“ und „Ungleich“ wählen.

Beispiel: Ebbinghoff 10

- *Gleich*
  Die Bedingung trifft zu, wenn als Rechnungsstraße Ebbinghoff 10 hinterlegt ist.
- *Ungleich*
  Die Bedingung ist erfüllt wenn die Rechnungsstraße nicht Ebbinghoff 10 ist.

Tage seit letzter Bestellung

Ermöglicht Dir die Prüfung, wie viele Tage seit der letzten Bestellung des Kunden vergangen sind.

Beispiel: 10

- *Gleich*
  Die Bedingung ist erfüllt, wenn seit der letzten Bestellung genau 10 Tage vergangen sind.
- *Größer*
  Die Bedingung ist erfüllt, wenn seit der letzten Bestellung mehr als 10 Tage vergangen sind.
- *Größer gleich*
  Die Bedingung ist erfüllt, wenn seit der letzten Bestellung 10 oder mehr Tage vergangen sind.
- *Kleiner*
  Die Bedingung ist erfüllt, wenn seit der letzten Bestellung weniger als 10 Tage vergangen sind.
- *Kleiner gleich*
  Die Bedingung ist erfüllt, wenn seit der letzten Bestellung 10 oder weniger Tage vergangen sind.
- *Ungleich*
  Die Bedingung ist erfüllt, wenn seit der letzten Bestellung nicht 10 Tage vergangen sind.

Hervorgehobene Positionen

Hierüber kannst Du prüfen, ob es sich bei einer Position um einen Hervorgehobene Position handelt. Um ein Produkt manuell als Hervorgehobene Position zu deklarieren, kannst Du in den Einstellungen des Produkts Produkt hervorheben aktivieren.
Als Operatoren kannst Du zwischen „Ja“ und „Nein“ wählen.

- *Ja*
  Die Bedingung ist erfüllt, wenn ein Produkt Hervorgehobene Position ist
- *Nein*
  Die Bedingung ist erfüllt, wenn kein Produkt Hervorgehobene Position ist

Verkaufskanal

Wähle hier aus den vorhandenen Verkaufskanälen Einen oder Mehrere aus.

Zusätzlich wählst Du zwischen den Operatoren „Ist eine von“ oder „Ist keine von“.

Beispiel: Hauptshop

- *Ist eine von*
  Die Bedingung trifft nur zu, wenn der Einkauf über den Hauptshop erfolgt.
- *Ist keine von*
  Die Bedingung trifft zu, wenn der Einkauf nicht über den Hauptshop erfolgt.

Versandart

Prüfe mittels dieser Bedingung, ob es sich um eine bestimmte Versandart handelt. Dazu wählst Du aus den Operatoren „Ist eine von“ und „Ist keine von“. Anschließend kannst Du aus einem Dropdown-Menü eine oder mehrere der vorhandenen Versandarten hinzufügen.

Dies kannst Du z.B. verwenden, um Versandarten nicht mit bestimmten Zahlungsarten zusammen anzubieten.

Beispiel: Express-Versand

- *Ist eine von*
  Die Bedingung ist erfüllt, wenn der Express-Versand ausgewählt wird
- *Ist keine von*
  Die Bedingung ist erfüllt, wenn nicht Express-Versand ausgewählt wird

Versandkostenfrei

Über die Operatoren „Ja“ und „Nein“ kannst Du prüfen, ob sich im Warenkorb aktuell Artikel befinden, die als versandkostenfrei markiert sind.

- *Ja*

Die Bedingung ist erfüllt, wenn sich versandkostenfreie Artikel im Warenkorb befinden
- *Nein*
  Die Bedingung ist erfüllt, wenn sich keine versandkostenfreien Artikel im Warenkorb befinden

Währung

Wähle hier die Währung(en) aus, die für diese Bedingungen berücksichtigt werden sollen.
Es können die unter Einstellungen > Währungen definierten Währungen ausgewählt werden.
Als Operator wählst Du zwischen „Ist eine von“ oder „Ist keine von“.

Beispiel: Euro, Pfund

- *Ist eine von*
  Die Bedingung ist gültig, wenn als Währung Euro oder Pfund verwendet werden.
- *Ist keine von*
  Die Bedingung trifft zu, wenn als Währung nicht Euro oder Pfund ausgewählt ist.

Warenkorb Gewicht

Prüfung auf das Gewicht des gesamten Warenkorbs.

Beispiel: 2,0

- *Gleich*
  Die Bedingung ist erfüllt, wenn das Warenkorb Gewicht genau 2,0 beträgt.
- *Größer*
  Die Bedingung ist erfüllt, wenn das Warenkorb Gewicht größer als 2,0 ist
- *Größer gleich*
  Die Bedingung ist erfüllt, wenn das Warenkorb Gewicht 2,0 oder mehr beträgt

- *Kleiner*
  Die Bedingung ist erfüllt, wenn das Warenkorb Gewicht geringer als 2,0 ist.
- *Kleiner gleich*
  Die Bedingung ist erfüllt, wenn das Warenkorb Gewicht 2,0 oder weniger ist.
- *Ungleich*
  Die Bedingung ist erfüllt, wenn das Warenkorb Gewicht nicht 2,0 ist.

Warenkorbwert

Für die Prüfung auf den Warenkorbwert stehen umfangreiche Möglichkeiten zur Verfügung.
Bitte beachte hierbei, dass dieser Bedingung keine Währung zugewiesen wird.
Wenn du dies zusätzlich an eine Währung koppeln möchtest, kannst du hierzu eine weitere, über UND verknüpfte Bedingung für Währung hinzufügen.

Beispiel: 49,99

- *Gleich*
  Die Bedingung ist erfüllt, wenn der Warenkorbwert genau 49,99 aufweist.
- *Größer*
  Die Bedingung ist erfüllt, wenn der Warenkorbwert größer als 49,99 ist
- *Größer gleich*
  Die Bedingung ist erfüllt, wenn der Warenkorbwert 49,99 oder mehr beträgt
- *Kleiner*
  Die Bedingung ist erfüllt, wenn der Warenkorbwert geringer als 49,99 ist.
- *Kleiner gleich*
  Die Bedingung ist erfüllt, wenn der Warenkorbwert 49,99 oder weniger ist.
- *Ungleich*

Die Bedingung ist erfüllt, wenn der Warenkorbwert nicht 49,99 entspricht.

Wochentag

Hierüber kannst Du eine Prüfung auf einen Wochentag durchführen.
Eine genaue Definition ist über die Operatoren „Gleich“ und „Ungleich“ möglich

Beispiel: Donnerstag

- *Gleich*
  Die Bedingung ist erfüllt, wenn der aktuelle Wochentag Donnerstag ist.
- *Ungleich*
  Die Bedingung ist erfüllt, wenn der aktuelle Wochentag nicht Donnerstag ist.

Zahlungsart

Prüfe mittels dieser Bedingung, ob es sich um eine bestimmte Zahlungsart handelt. Dazu wählst Du aus den Operatoren „Ist eine von“ und „Ist keine von“. Anschließend kannst Du aus einem Dropdown-Menü eine oder mehrere der vorhandenen Zahlungsarten hinzufügen.

Dies kannst Du z.B. verwenden, um Zahlungsarten nicht mit bestimmten Versandarten zusammen anzubieten.

Beispiel: Rechnung

- *Ist eine von*
  Die Bedingung ist erfüllt, wenn Rechnung als Zahlungsart ausgewählt wird
- *Ist keine von*
  Die Bedingung ist erfüllt, wenn eine andere Zahlungsart als Rechnung ausgewählt wird

Zeitraum

Hierüber definierst du einen Zeitraum, in dem die Bedingung zutrifft. Beispiel: 06:00 – 22:00

- Die Bedingung ist zwischen 06:00 Uhr und 22:00 Uhr gültig

Bedingungen verknüpfen

Innerhalb einer Regel kannst Du mehrere Bedingungen miteinander verknüpfen, dadurch ist es Dir möglich, auch sehr komplexe Szenarien abzubilden.
Bei einer Und-Verknüpfung müssen alle Bedingungen zutreffen, damit die Regel erfüllt ist.
Für die Erfüllung einer Oder-Verknüpfung muss nur eine der Bedingungen übereinstimmen.
Außerdem hast Du die Möglichkeit, Unterbedingungen zu erstellen. Hierbei wird zunächst die Oberbedingung geprüft und wenn diese erfüllt ist, erfolgt die Prüfung der untergeordneten Bedingung(en).

[/expand]

---

roles and permissions – november 2020

WHRGDr0pQG5E%253D

[expand title="mehr lesen…"]

Roles & Permissions is planned for NovemberPrepare your plugin now:

In November we plan to release the highly demanded feature „Roles & Permissions“ (ACL). With this feature you will be able to assign rights and roles to users in the administration.

Prepare your plugins:
In order for store operators to be able to use the feature „Roles & Permissions“ to its full extent, you should prepare the rights in your plugins now.

What you have to do for that?
The feature „Roles & Permissions“ is already completely implemented on the master (GitHub) and can be tested by activating a feature flag. A detailed instruction on how to activate the feature flag can be found below in the mailing.

Important to know:
All customizations are upward and downward compatible. That means you can make the changes now and your plugin is still compatible in older Shopware 6 versions and of course in future versions.

Documentation
Activate feature flagDevelopment Template

If the Development Template is used, the feature flag can be activated as follows:

1.) The following code should be integrated in the *.psh.yaml.override*

Check out the Code on GitHub

2.) After that the cache should be cleared with the following command:

./psh.phar cache

Activate feature flagProduction Template and all other installation types

For all other types of installation, proceed as follows:

1.) An environment variable should be created with the following values:

FEATURE_NEXT_3722=1
You can also use e.g. the .env file.

2.) The cache should be cleared with the following command:

bin/console cache:clear

1000 +GitHub Stars

800,000 +Downloads

2000 +Developers on Slack

shopware AG
Ebbinghoff 10
Schöppingen

DE: +49 (0) 2555 928850
UK: +44 (0) 203 095 2445
World: 00 800 746 7626 0
info@shopware.com

Board: Stefan Hamann, Sebastian Hamann
Supervisory Board: Reinhold Wellers (Chairman), Christoph Hertz, Christoph Pliete
Register Court: Amtsgericht Coesfeld HRB 11471

[/expand]

Shopware 6 Theme entwickeln – Livestream #4

Shopware 6 Theme entwickeln – Livestream #4

[expand title="mehr lesen…"]

Your browser does not support HTML5 video.

[/expand]

Shopware 6 Theme entwickeln – Livestream #3

Shopware 6 Theme entwickeln – Livestream #3

[expand title="mehr lesen…"]

Your browser does not support HTML5 video.

[/expand]

Shopware 6 Theme entwickeln – Community store/Compile a theme

Shopware 6 Theme entwickeln – Community store/Compile a theme

[expand title="mehr lesen…"]

Community store

To publish your plugins in the Shopware Community Store, you need to register your own developer prefix in your Shopware account.

After that process you can compile your created theme, then create a zip file of the folder and upload your theme to the Shopware Community Store.

Compile a theme

```
# run this to compile your theme
$ bin/console theme:compile
```

[/expand]

Shopware 6 Theme entwickeln –

JavaScript

Shopware 6 Theme entwickeln – Shopware 6 Theme entwickeln – JavaScript

[expand title="mehr lesen…"]

You can add JavaScript to theme to interact with the DOM, make some API calls or change the behavior of the storefront.

By default, Shopware 6 look inside the <plugin root>/src/Resources/app/storefront/src folder of your plugin to load a main.js file.

You can simple put your own JavaScript code in here. Shopware 6 support the ECMAScript 6 and will transpile your code to ES5 for legacy browser support.

The recommended way to add JavaScript is to write a JavaScript Plugin for the storefront.

Writing a JavaScript plugin

Storefront JavaScript plugins are vanilla JavaScript ES6 classes that extend from our Plugin base class. For more background information on JavaScript classes, take a look here. To get started create a src/Resources/app/storefront/src/example-plugin folder and put an example-plugin.plugin.js file in there. Inside that file create and export a ExamplePlugin class that extends the base Plugin class:

```
//  src/Resources/app/storefront/src/example-plugin/example-plugin.plugin.js

import Plugin from 'src/plugin-system/plugin.class';

export default class ExamplePlugin extends Plugin {
}
```

Each Plugin has to implement the init() method. This method will be called when your plugin gets initialized and is the entrypoint to your custom logic. In your case you add an callback to the onScroll event from the window and check if the user has scrolled to the bottom of the page. If so we display an alert. Your full plugin now looks like this:

```
//  src/Resources/app/storefront/src/example-plugin/example-plugin.plugin.js

import Plugin from 'src/plugin-system/plugin.class';

export default class ExamplePlugin extends Plugin {
    init() {
        window.onscroll = function() {
             if ((window.innerHeight + window.pageYOffset) >= document.body.offsetHeight) {
                 alert('seems like there\'s nothing more to see here.');
             }
        };
    }
}
```

Registering your plugin

Next you have to tell Shopware that your plugin should be loaded and executed. Therefore you have to register your plugin in the PluginManager. Open up main.js file inside your src/Resources/app/storefront/src folder and add the following example code to register your plugin in teh PluginManager.

```
// src/Resources/app/storefront/src/main.js

// import all necessary storefront plugins
import ExamplePlugin from './example-plugin/example-plugin.plugin';

// register them via the existing PluginManager
const PluginManager = window.PluginManager;
PluginManager.register('ExamplePlugin', ExamplePlugin);
```

You also can bind your plugin to an DOM element by providing an css selector:

```
// src/Resources/app/storefront/src/main.js

// import all necessary storefront plugins
import ExamplePlugin from './example-plugin/example-plugin.plugin';

// register them via the existing PluginManager
const PluginManager = window.PluginManager;
PluginManager.register('ExamplePlugin', ExamplePlugin, '[data-example-plugin]');
```

In this case the plugin just gets executed if the HTML document contains at least one element with the data-scroll-detector attribute. You could use this.el inside your plugin to access the DOM element your plugin is bound to.

Loading your plugin

You bound your plugin to the css selector „[data-example-plugin]“ so you have to add DOM elements with this attribute on the pages you want your plugin to be active. Therefore create an Resources/views/storefront/page/content/ folder and create an index.html.twig template. Inside this template extend from the @Storefront/storefront/page/content/index.html.twig and overwrite the base_main_inner block. After the parent content of the blog add an template tag that has the data-example-

plugin attribute. For more information on how template extension works, take a look here.

```
{% sw_extends '@Storefront/storefront/page/content/index.html.twig' %}

{% block base_main_inner %}
    {{ parent() }}

    <template data-example-plugin></template>
{% endblock %}
```

With this template extension your plugin is active on every content page, like the homepage or category listing pages.

More about Storefront JavaScript: Storefront JavaScript reference documentation

[/expand]

Shopware 6 Theme entwickeln – SCSS and Styling

Shopware 6 Theme entwickeln – SCSS and Styling

Shopware 6: SCSS and Styling

In our documentation you will find all the information you

need for your daily work with Shopware.

[expand title="mehr lesen…"]

The stylesheets are written in SASS. The folder structure is inspired by the 7-1 pattern structure.

```
<platform/src/Storefront/Resources/app/storefront/src/scss>
└── abstract
└── base
└── component
└── layout
└── page
└── skin
└── vendor
└── base.scss
```

Shopware 6 looks inside your theme.json file to find a „style“ array which contains all SCSS files which should be loaded by your theme. By default you get the Shopware Storefront SCSS plus an additional entry point for your own SCSS. You can also extend this array with more SCSS files.

```
{
  "name": "Just another theme",

  ...

  "style": [
    "@Storefront",
    "app/storefront/src/scss/base.scss" <-- Theme SCSS entry
  ],

  ...

}
```

To try it out, open up the base.scss file from your theme.

Inside of the .scss file, add some styles like this:

```
// src/Resources/app/storefront/src/scss/base.scss
```

```
body {
    background: blue;
}
```

To see if it’s working you have to re-build the storefront. Use the following command to do that.

```
# run this to re-compile the current storefront theme
$ ./psh.phar storefront:build

# or run this to start a watch-mode (the storefront will re-compile when you make sytle changes)
$ ./psh.phar storefront:hot-proxy
```

In this example, the background of the body will be changed.

Working with variables

In case you want to use the same color in several places, but want to define it just one time you can use variables for this.

Create a abstract/variables.scss file inside your „scss“ folder and define your background color variable.

```
// in variables.scss
$sw-storefront-assets-color-background: blue;
```

Inside your base.scss file you can now import your previously defined variables and use them:

```
// in base.scss
@import 'abstract/variables.scss';

body {
    background: $sw-storefront-assets-color-background;
}
```

This has the advantage that when you want to change the values of your variables you just have one location to change them and the hard coded values are not cluttered all over the codebase.

Bootstrap

The storefront theme is implemented as a skin on top of the Boostrap toolkit.

Override default SCSS variables

To override default variables like for example $border-radius from Bootstrap you should use a slightly different approach then explained in the storefront assets how-to.

Bootstrap 4 is using the !default flag for it’s own default variables. Variable overrides have to be declared beforehand.

More information: https://getbootstrap.com/docs/4.0/getting-started/theming/#variable-defaults

To be able to override Bootstrap variables you can define an additional SCSS entry point in your theme.json which is declared before @Storefront. This entry point is called overrides.scss:

```
{
  "name": "Just another theme",
  "author": "Just another author",
  "views": [
     "@Storefront",
     "@Plugins",
     "@JustAnotherTheme"
  ],
  "style": [
     "app/storefront/src/scss/overrides.scss", <-- Variable overrides
    "@Storefront",
    "app/storefront/src/scss/base.scss"
  ],
  "script": [
    "@Storefront",
    "app/storefront/dist/storefront/js/just-another-theme.js"
```

```
  ],
  "asset": [
    "app/storefront/src/assets"
  ]
}
```

In the overrides.scss you can now override default variables like $border-radius globally:

```
/*
Override variable defaults
==========================================================
This file is used to override default SCSS variables from the Shopware Storefront or Bootstrap.

Because of the !default flags, theme variable overrides have to be declared beforehand.
https://getbootstrap.com/docs/4.0/getting-started/theming/#variable-defaults
*/

$disabled-btn-bg: #f00;
$disabled-btn-border-color: #fc8;
$font-weight-semibold: 300;
$border-radius: 0;
$icon-base-color: #f00;
$modal-backdrop-bg: rgba(255, 0, 0, 0.5);
```

Please only add variable overrides in this file. You should not write CSS code like .container { background: #f00 } in this file. When running storefront:hot or storefront:hot-proxy SCSS variables will be injected dynamically by webpack. When writing selectors and properties in the overrides.scss the code can appear multiple times in your built CSS.

Using Bootstrap SCSS only

The Shopware default theme is using Bootstrap with additional custom styling.

If you want to build your theme only upon the Bootstrap SCSS you can use the @StorefrontBootstrap placeholder instead of the @Storefront bundle in the style section of your theme.json. This gives you the ability to use the Bootstrap SCSS without the Shopware Storefront „skin“. Therefore all the SCSS from src/Storefront/Resources/app/storefront/src/scss/skin wil l not be available in your theme.

```
{
  "style": [
    "@StorefrontBootstrap",
    "app/storefront/src/scss/base.scss"
  ]
}
```

- This option can only be used in the style section of the theme.json. You must not use it in views or script.
- All theme variables like $sw-color-brand-primary are also available when using the Bootstrap option.
- You can only use either @StorefrontBootstrap or @Storefront. They should not be used at the same time. The @Storefront bundle **includes** the Bootstrap SCSS already.

[/expand]

Shopware 6 Theme entwickeln – Snippets

Shopware 6 Theme entwickeln – Snippets

[expand title="mehr lesen…"]

To extend a language in Shopware 6 you can add your own snippets in your theme. You can also add a completely new language to Shopware 6.

General snippet structure

To organize your snippets you can add them to .json files, so structuring and finding snippets you want to change is very easy.

Adding snippets

You can add new snippets by adding files to the follow structure like this:

```
# move into your theme folder
$ cd custom/plugins/MyTheme

# structure of theme
└─ src
   ├─ Resources
   │  ├─ config
   │  │  └─ services.xml
   │  └─ snippet
   │     ├─ de_DE
   │     │  ├─ SnippetFile_de_DE.php
   │     │  └─ storefront.de-DE.json
   │     └─ en_GB
   │        ├─ SnippetFile_en_GB.php
   │        └─ storefront.en-GB.json
   └─ MyTheme.php
```

There is no explicit syntax for variables in the storefront. It is nevertheless recommended to encompass them with % symbols to be extra clear on what their purpose is. Pluralization works for any natural number. Just remember to explicitly define the intervals‘ amounts and ranges for that snippet.

Example of src/Resources/snippet/storefront.en-GB.json:

```
{
    "my-theme": {
        "productDetail": {
             "headLineText": "There are %count% discounts available for %product%:",
             "description": "Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam ..."
        },
        "cart": {
            "itemCounter": "{1} 1 discount item | ]1,Inf[ %count% discount items"
        }
    }
}
```

Adding Storefront snippets

Storefront snippets additionally require a class that extends the SnippetFileInterface. A suitable name would e.g. be SnippetFile_en_GB.php. Having created that file, you will have to implement the following five methods:

- getName: Returns the name of the snippet file as a string. By referring to this name, you can access the translations later. It is **required** to use messages.en-GB, if you provide a whole new language. By default, an extension should call its Storefront extension storefront.en-GB. Otherwise a describing domain, like shopware’s PayPal plugin using paypal.en-GB, is also okay.

- getPath: Each SnippetFile class has to point to the .json file, that contains the actual translations. Return its path here. We suggest using the name already chosen in getName for your file name.
- getIso: Return the ISO string of the supported locale here. This is important, because the Translator collects every snippet file with this locale and merges them to generate the snippet catalogue used by the storefront.
- getAuthor: Return your vendor name here. This can be used to distinguish your snippets from all the other available ones. The Administration snippet set module offers a filter, so users are able to easily find plugin specific snippets.
- isBase: Return true here, if your theme implements a whole new language, such as providing french snippets for the whole Shopware 6 system. Don’t forget to watch your getName method then. Most of the time, you’re just adding your own snippets to an existent language, then false will be your way to go.

Example for the language en-GB :

```
<?php declare(strict_types=1);

namespace MyTheme\Resources\snippet\en_GB;

use Shopware\Core\System\Snippet\Files\SnippetFileInterface;

class SnippetFile_en_GB implements SnippetFileInterface
{
    public function getName(): string
    {
        return 'storefront.en-GB';
    }

    public function getPath(): string
    {
        return __DIR__ . '/storefront.en-GB.json';
    }
```

```
    public function getIso(): string
    {
        return 'en-GB';
    }

    public function getAuthor(): string
    {
        return 'Enter developer name here';
    }

    public function isBase(): bool
    {
        return false;
    }
}
```

Registering your service

Lastly you have to register your snippet files by creating file src/Resources/config/services.xml.

Example:

```
<!-- src/Resources/config/services.xml -->
<?xml version="1.0" ?>

<container xmlns="http://symfony.com/schema/dic/services"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://symfony.com/schema/dic/services
http://symfony.com/schema/dic/services/services-1.0.xsd">

    <!-- Translations -->
    <services>
                                                    <service
id="MyTheme\Resources\snippet\en_GB\SnippetFile_en_GB"
public="true">
            <tag name="shopware.snippet.file"/>
        </service>
    </services>

    <services>
                                                    <service
```

```
id="MyTheme\Resources\snippet\de_DE\SnippetFile_de_DE"
public="true">
            <tag name="shopware.snippet.file"/>
        </service>
    </services>
</container>
```

Using your snippets in your templates

With the trans filter you can use your snippets in the twig templates and they will be translated automatically. You also can pass values to replace the placeholders in the snippets.

```
<div class="product-detail-headline">
            {{ 'my-theme.productDetail.headLineText' |
trans({'%count%': count, '%product%': product}) }}
    </div>


[/expand]
```