projekt kivitendo - Fehler #82

Berechnete Preiswerte von PTC weichen von oberflächen Werten aus den Masken ab

02.09.2015 13:23 - Jan Büren

Status:	Erledigt	**Beginn:**	02.09.2015
Priorität:	Hoch	**Abgabedatum:**	
Zugewiesen an:		**% erledigt:**	0%
Kategorie:		**Geschätzter Aufwand:**	0:00 Stunde
Zielversion:		**Aufgewendete Zeit:**	0:00 Stunde

Beschreibung

Es war ja gefühlt schon zu erwarten, aber bei meinem derzeitigen Testfall habe ich folgende Abweichungen:

testefeste=# select netamount,amount,marge_percent,marge_total from ar;
netamount	amount	marge_percent	marge_total
1137.96000	1354.17000	50.88580	579.06000
1137.98000	1354.20000	50.88666	579.08000

Das Problem hierbei ist, dass ich bei folgenden Verfahren dann eine Abweichung habe:

Benutzer legt Auftrag A16399 (od.rnr anonymisiert an) und erhält folgende werte:

jan_rb=# select netamount,amount,marge_percent,marge_total from oe where ordnumber='A16399';
netamount	amount	marge_percent	marge_total
1137.98000	1354.20000	50.88666	579.08000

Der Lieferschein wird erstellt und mittels convert_invoice erstellt man eine in Summe abweichende Rechnung.

Ich prüf mal kurz in der console gegen, wie es sich von sales_order nach invoice verhält:

```
my $order = SL::DB::Manager::Order->find_by( ordnumber => 'A16399' );
 my $inv_conv = $order->convert_to_invoice;
 pp $inv_conv; # sieht entsprechend schlecht aus
 > $inv_conv->invnumber;
  R18084
 > $inv_conv->save;
```

Das ergibt dann:

1. select netamount,amount,marge_percent,marge_total from ar where invnumber='R18084';
netamount	amount	marge_percent	marge_total
1137.96000	1354.17000	50.88580	579.06000

Bernd hat etwas ähnliches schon mal in Commit: a607a2d0854ef2b2e85bff4ab9 angesprochen.

Denn Rabatt-Fall hab ich auch. Für den Test kommentier ich das erstmal mit Verweis auf dieses Ticket und rechne mit den Zahlen aus PTC

Historie

#1 - 03.02.2016 15:39 - Bernd Bleßmann

Da wir wieder drüber gestolpert sind, fasse ich hier nochmal die/einige Probleme bzgl. Rundung zusammen:

Beispiel: 2 Artikel, Menge jeweils 1, Preis 1,004 (ähnliche Probleme kann man auch mit Rabatten nachstellen, auch wenn der Preis nur 2 Nachkommastellen hat)

Auftrags-Maske:

```
1 x 1,004 = 1,00
```

```
1 x 1,004 = 1,00
----------------
Zwisch......2,00
USt.........0,38
Summe.......2,38
```

Aber gespeichert in oe.amount/oe.netamount (und angezeigt in Berichte->Aufträge (Summe/Betrag)): 2,39 / 2,01
Also anders als in der Maske.

Rechnungs-Maske (Workflow oder Neueingabe):

```
1 x 1,004 = 1,00
1 x 1,004 = 1,00
----------------
Zwisch......2,00
USt.........0,38
Summe.......2,38
```

Wie im Auftrag, aber richtig gespeichert in ar.amount/ar.netamount (und angezeigt Berichte->Rechnungen (Summe/Betrag)): 2,38 / 2,00
(also wie in der Maske.)

PriceTaxCalculator (und damit vermutlich Massendruck/wiederkehrende Rechnungen/neuer Auftrags-Controller):

```
1 x 1,004 = 1,00
1 x 1,004 = 1,00
----------------
Zwisch......2,01
USt.........0,38
Summe.......2,39
```

Gespeichert in oe.amount/oe.netamount (und angezeigt in Berichte->Aufträge (Summe/Betrag)): 2,39 / 2,01
Also wie in der Maske. Allerdings ist die Summe von 1+1=2,01 in der Maske (neuer Controller) verwirrend.

Der PriceTaxCalculator liefert allerdings (wenn im Array-Kontext aufgerufen) pro Erlöskonto einen Betrag (amount zur entspr. chart_id), wo der gerundetet Wert drinsteht. Hier im Beispiel 2,00.
Zudem wird auch amount/netamount des Objekts gesetzt. Hier im Beispiel 2,39 / 2,01, also unterschiedlich.

#2 - 12.05.2016 16:09 - G. Richardson

Ist bei mir jetzt auch wieder hochgekommen.

6 Artikel zu 0.6€ mit 3% Rabatt

```
PTC:  ( 0.6 - round(0.6*0.03,2) ) * 6 = 3.48
 Form: 0.6 * 6 * ( 1 - 0.03 )          = 3.49
```

Die unterschiedlichen Berechnungsmethoden sindauch schön in t/db_helper/price_tax_calculator.t beschrieben.

#3 - 12.05.2016 16:50 - G. Richardson

Mit dem Runden ist das so eine Sache, vielleicht hat die PTC Methode ja andere Vorteile, als einfach alles zu multiplizieren und am Schluß zu runden. Aber die Methode ist sehr anfällig bei großen Mengen und kleinen Beträgen und kleinen Rabattgrößen. Im neuen OrderController, der den PTC nutzt, kann man das auch schön schnell nachvollziehen, z.B.:

```
100000 Artikel zu 0,10€ mit 15,00% Rabatt:  8000€      -> 8500 wären korrekt
100000 Artikel zu 0,10€ mit 14,99% Rabatt:  9000€
100000 Artikel zu 0,10€ mit 14,00% Rabatt:  9000€      -> 9600
100000 Artikel zu 0,10€ mit 10.00% Rabatt:  9000€      -> 9000 ok
100000 Artikel zu 0,10€ mit  5.00% Rabatt:  9000€      -> 9500
100000 Artikel zu 0,10€ mit  4.99% Rabatt: 10000€
100000 Artikel zu 0,10€ mit  4.00% Rabatt: 10000€      -> 9600
100000 Artikel zu 0,10€ mit  1.00% Rabatt: 10000€      -> 9900
```

Und korrekt wären 9500€ bei 5%!

#4 - 13.05.2016 14:00 - G. Richardson

- Datei 0001-PTC-rundet-nicht-mehr-Rabatt-vor-Mengenmultiplikatio.patch wurde hinzugefügt

Hier ein Patch wie es wahrscheinlich sein sollte, zumindest stimmen
dann die gespeicherten und gedruckten Zahlen wieder mit den anderen Masken überein.
Das löst aber natürlich noch nicht das "Rückwärtskompatibilitätsproblem".
Und höchstwahrscheinlich löst das an anderer Stelle auch noch ein anderes Problem aus.

#5 - 17.05.2016 16:54 - Bernd Bleßmann

G. Richardson schrieb:

> Hier ein Patch wie es wahrscheinlich sein sollte, zumindest stimmen
> dann die gespeicherten und gedruckten Zahlen wieder mit den anderen Masken überein.
> Das löst aber natürlich noch nicht das "Rückwärtskompatibilitätsproblem".
> Und höchstwahrscheinlich löst das an anderer Stelle auch noch ein anderes Problem aus.

Zum Berechnen sollte meiner Meinung nach der fxsellprice genommen werden, denn der sellprice hat bei gespeicherten Rechnungen schon den Rabatt abgezogen (und ist auch schon umgerechnet, aber mit exchangerate wird auch noch multipliziert):
my $linetotal = _round($sellprice * (1-$item->discount) * $item->qty / $item->price_factor, 2) * $data->{exchangerate};

Aber der fxsellprice wird vorher aus dem sellprice gesetzt:
$item->fxsellprice($item->sellprice) if $data->{is_invoice};

Hier müsste dann unterschieden werden, ob das Objekt/die Rechnung den fxsellprice schon hat oder nicht, oder der fxsellprice ist für Rechnungen obligatorisch und beinhaltet den eingegebenen Wert.

Aufträge verhalten sich anders - da gibt es kein fxsellprice.

#6 - 16.02.2018 11:28 - Jan Büren

- Priorität wurde von Normal zu Hoch geändert

Ich hab jetzt nochmal mit Bernd diskutiert. Alle unsere Kunden haben diesen Patch drin.

Ich hab das Problem gestern nochmal anders bei einem anderen Kunden gepatcht:

```
  my $num_dec    = max 2, _num_decimal_places($item->sellprice);
+  $num_dec      = 5;
```

Vielleicht kann Moritz sich damit eher anfreunden, dann bleibt die Berechnungsgrundlade im PTC unangetastet, aber die Abweichung vom neuen zum alten Verfahren ist nicht mehr bei 40 Cent pro Maske, sondern eher bei 0 <= x <=1 Cent

Eine unabdingbare Abhängigkeit zu #342 sehen wir so auch nicht, das kann aber gerne beim Partnertreffen nochmal in Ruhe diskutiert werden.

Ich will an der Stelle nur kurz den ungüngstigen Gesamtzusammenhang im Programm darstellen:

Ist-Zustand:

a) Denselben Auftrag im neuen Code und im alten Code anzeigen, kann unterschiedliche Zeilensummen und somit Gesamtsummen liefern.

b) Wiederkehrende Rechnungen können diesselbe Ungenauigkeit wie a) haben

Daraus ergibt sich foglende organisatorische Herausforderung für alle kivi-Mandanten bei denen a) oder b) greift:

a) Mahnungen mahnen einen niedrigen Betrag an als in der ursprüngliche gedruckten Rechnung (yiha)

b) Der tatsächliche Zahlungseingang ist immer höher als der von kivi erwartete

c) Die OPOS-Listen werden nie sauber gepfelgt ausgebucht, da Forderungen überzahlt sind, obwohl diese in der Rechungsmaske als ausgeglichen stehen

Die Lösung an der Stelle kann auch nicht sein, wiederkehrende Rechnungen und damit den PTC wieder aus dem Programm zu schmeißen (mit dem Argument hab ich den Kunden als Lösungsszenario wieder eingefangen ...).

#7 - 14.11.2018 17:32 - Bernd Bleßmann

- Datei 0001-Zahlen-in-Tests-anngepasst-nach-PTC-Rundungs-Patch.patch wurde hinzugefügt

- Datei 0002-PTC-item-discount-auf-0-wenn-nicht-definiert-um-Warn.patch wurde hinzugefügt

Ich hatte die Test nochmal angepasst und eine Warnung im PTC vermieden - siehe Patchs

#8 - 12.12.2018 17:32 - Bernd Bleßmann

- Status wurde von Neu zu Erledigt geändert

Ich habe Geoffreys Patch eingebaut, etwas erweitert und die Tests angepasst. Folgende commits sind das:

0d0c160c76a1c747ca1295cf9cefeff76661dbb3 PTC-Tests: ein weiterer Test mit großen Mengen und kleinen Preisen
ebcd0a69b1a9b8ac7150f3599392a6d410687ae6 PTC-Tests angepasst nach PTC-Rundungs-Patch …

faf42bec5d778fa8d35ad30a4c90e803d675753b PTC: item->discount auf 0, wenn nicht definiert, um Warnungen zu vermeiden
36bdd4875c67f916e4a885191f7870d301ae2c11 PTC: Rückgabe sellprice für items: Steuer und Rabatt berücksichtigen.
fa5d2a24fc338e4c50951bbf3946b2b9a9f99ec2 PTC: Kosmetik/Kommentare
dd75973a716c63d66df1348dc928a64f3f260352 PTC rundet nicht mehr Rabatt vor Mengenmultiplikation
c979352b982f25a5548a08d2ab06d3059479b5f2 PTC: nicht einfach die Rundungsgenauigkeiten erhöhen …

Dateien

0001-PTC-rundet-nicht-mehr-Rabatt-vor-Mengenmultiplikatio.patch	8,15 KB	13.05.2016	G. Richardson
0002-PTC-item-discount-auf-0-wenn-nicht-definiert-um-Warn.patch	1,15 KB	14.11.2018	Bernd Bleßmann
0001-Zahlen-in-Tests-anngepasst-nach-PTC-Rundungs-Patch.patch	7,71 KB	14.11.2018	Bernd Bleßmann