FAKULTÄT
FÜR !NFORMATIK

Faculty of Informatics

ebInterface 4.0
AK Meeting

26. Juni 2013

Philipp Liegl

Agenda

- Forumsupdate
- Änderungsvorschläge für ebInterface
- Requirements für weitere Dokumenttypen

Diskussionspunkt 1: für OrderID von Hr. Helger (BRZ)

Anbei möchte ich ihnen einen Änderungsvorschlag für die nächste ebInterface Version unterbreiten:
Das Feld „OrderID“ – also die Bestellreferenz – ist derzeit vom Typ „AlphaNumType“. Mein Vorschlage ist, diesen auf „xs:string“ zu ändern, um auch andere Zeichen wie z.B. „:“ oder „/“ unterbringen zu können!

Aufgefallen ist mir das deshalb weil wir eine Support-Anfrage bzgl. eines ungültigen Beispiels für die Auftragsreferenz des Bundes bekommen haben (auf http://test.erb.gv.at/index.jsp?p=info_channel).
Dort wurde moniert, dass „/“ kein gültiges Zeichen ist.
Im Zuge der Recherche ist mir auch aufgefallen, dass unser Trennzeichen „:“ gar kein gültiger Teil der OrderID ist, unsere Implementierung sich aber nie darüber beschwert hat.
D.h. wir müssen unser Trennzeichen jetzt anpassen.

Diskussionspunkt 1

Der jetztige SimpleType für einen Alphanumerischen Wert:

```
<xs:simpleType name="AlphaNumType">
        <xs:restriction base="xs:string">
                <xs:pattern value="[0-9 | A-Z | a-z | -_äöüÄÖÜß]+"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="AlphaNumIDType">
        <xs:restriction base="AlphaNumType">
                <xs:maxLength value="35"/>
        </xs:restriction>
</xs:simpleType>
```

ergibt leider wirklich keinen Sinn, da die Regular Expression nicht korrekt ist.

Den "Fehler" schleifen wir bereits seit Version 1.0 mit - aufgefallen ist er bisher noch nie, da eben "jede" OrderID gültig war.

Diskussionspunkt 1

Lösung

A) Ändern von AlphaNumType

```
<xs:simpleType name="AlphaNumType">
  <xs:restriction base="xs:string">
    <xs:pattern value="([0-9A-Za-z]|[-_äöüÄÖÜß])+"/>
  </xs:restriction>
</xs:simpleType>
```

B) OrderID auf xs:string ohne Längenbeschränkung ändern.

AlphaNumType und AlphaNumIDType sollen rausfallen und durch xs:string ersetzt werden.

Diskussionspunkt 2 - BICType

- Aktueller BIC Typ ist nicht korrekt (Regular Expression ist fehlerhaft)

```
<xs:simpleType name="BICType">
        <xs:restriction base="xs:string">
                <xs:pattern value="[0-9 | A-Z | a-z]{8}([0-9 | A-Z | a-z]{3})?"/>
        </xs:restriction>
</xs:simpleType>
```

Diskussionspunkt 2 - BICType

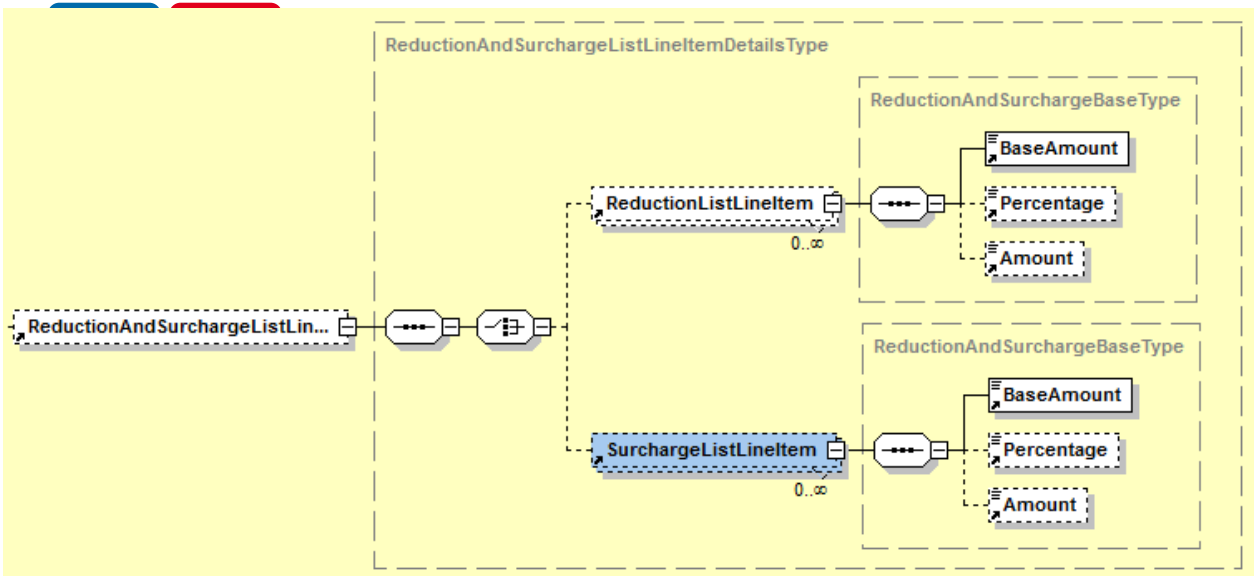
- Vorgeschlagene Korrektur

```
<xs:simpleType name="BICType">
        <xs:restriction base="xs:string">
            <xs:whiteSpace value="collapse"/>
            <xs:pattern value="[0-9A-Za-z]{8}([0-9A-Za-z]{3})?"/>
        </xs:restriction>
</xs:simpleType>
```

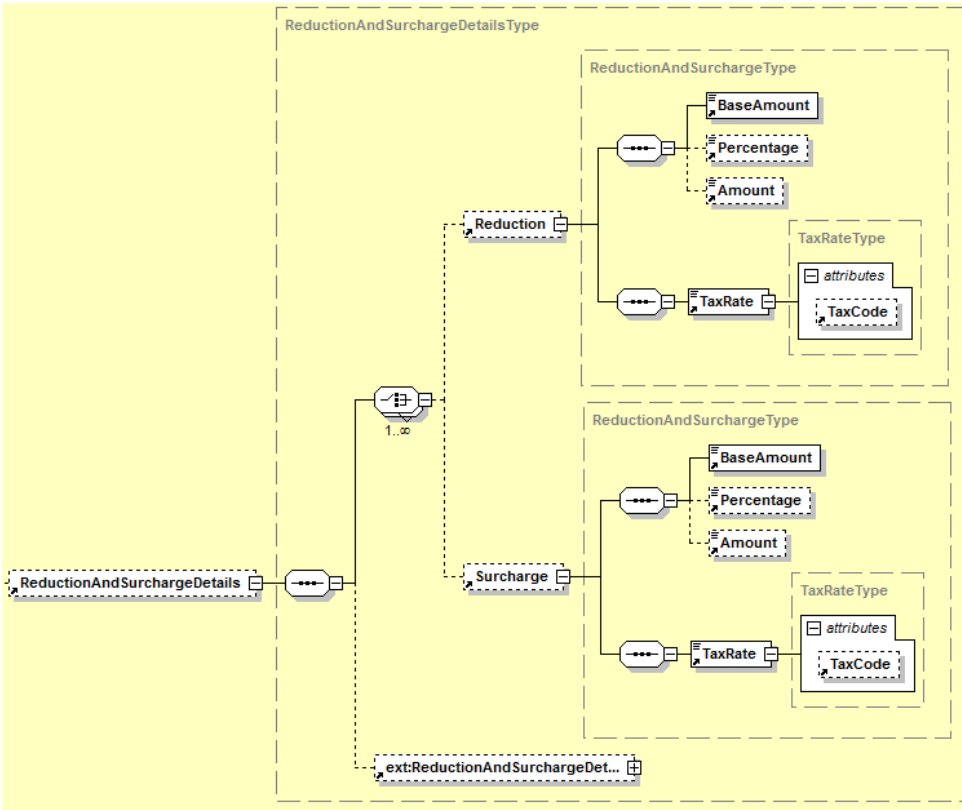
Diskussionspunkt 3 – Grund für Aufschlag bzw. Abschlag

- Lt. Diskussion zwischen Post und Bund bezüglich Reduction und Surcharge (global und auf Item-Ebene):
  - Es wäre wünschenswert, wenn es pro Eintrag ein optionales Beschreibungselement gibt, d.h. was der Grund des Aufschlags oder Abschlags ist

Diskussionspunkt 3 cont'd



Optionales "Comment" Feld hinzufügen



Diskussionspunkt 4 – Limitierung von Auf- und Abschlägen

- Lt. Diskussion zwischen Post und Bund bezüglich Reduction und Surcharge (global und auf Item-Ebene):
  - Die Limitierung von nur Aufschlägen bzw. nur Abschlägen ist auf Dauer ein Hindernis. Es führt dazu, dass im Falle von einem Aufschlag und einem Abschlag stattdessen 2 Aufschläge (oder Abschläge) verwendet werden, wobei einer einen positiven Betrag hat, und einer einen negativen Betrag. Daher sollten diese Einträge auch mischbar sein. Dazu braucht es dann aber auch genaue Regeln, in welcher Reihenfolge die Einträge anzuwenden sind, da der BaseAmount meiner Ansicht nach hier nicht ausreicht!

Bereits als Anforderung erfasst
Siehe Anforderung 3 auf der ebInterface Wunschliste.

Diskussionspunkt 5 – Steuerrelevanz von Auf- und Abschlägen

- Lt. Diskussion zwischen Post und Bund bezüglich Reduction und Surcharge (global und auf Item-Ebene):
  - Es ist nicht klar definiert, ob die Auf- und Abschläge steuer-relevant sind oder nicht

Diskussionspunkt 6 - Preisberechnungen

- Lt. Diskussion zwischen Post und Bund
  - Sowohl für jede Rechnungszeile als auch für den Gesamtbetrag sollten eindeutige Rechenregeln definiert werden, da diese in der Praxis unterschiedlich interpretiert werden. D.h. pro Rechnungszeile sollte definiert werden, wie der LineItemAmount zustande kommt, und auf globaler Ebene sollte definiert werden, wie der TotalGrossAmount zustande kommt. Diese Regeln sollten sowohl für ebInterface 3.0.x als auch für ebInterface 4.0 definiert werden.

Diskussionspunkt 6 cont'd – Vorschläge des BRZ

- LineItemAmount:
  - UnitNetAmount / BaseAmount * Amount +- (Reduction/Surcharge in der angegebenen Reihenfolge)
  - Das Ergebnis auf 2 Dezimalstellen im definierten Rundungsmodus runden
- LineItemGrossAmount:
  - Ungerundeter LineItemAmount * (100 + TaxRate) / 100
  - Das Ergebnis auf 2 Dezimalstellen im definierten Rundungsmodus runden
- TotalRoundingAmount:
  - Summe(ungerundeter LineItemGrossAmount) - Summe(gerundeter LineItemGrossAmount)
  - Das Ergebnis auf 2 Dezimalstellen im definierten Rundungsmodus runden
- TotalGrossAmount:
  - Summe(gerundeter LineItemGrossAmount) +- (Reduction/Surcharge in der angegebenen Reihenfolge) + TotalRoundingAmount
  - Das Ergebnis auf 2 Dezimalstellen im definierten Rundungsmodus runden

Diskussionspunkt 7 – Rounding Amount

- Lt. Diskussion zwischen Post und Bund
  - Auf globaler Ebene könnte es auch ein neues, optionales Feld namens „TotalRoundingAmount“ geben, das im Falle von Rundungsfehlern das Delta aufnimmt, damit auf jeden Fall eine genaue Berechnung möglich ist.

Diskussionspunkt 8 - Rundungsmodus

- Lt. Diskussion zwischen Post und Bund
  - Es sollte eine Definition des zu verwendenden Rundungsmodus gegeben werden.
    - Der Bund verwendet derzeit den Modus „HALF_UP“ der bei uns wie folgt definiert ist:
    - International üblich ist der Rundungsmodus “HALF_EVEN”:

Diskussionspunkt 9 – Optionales Feld BaseAmount

- Lt. Diskussion zwischen Post und Bund
  - Ein optionales Feld “BaseAmount” mit dem Standardwert “1” wäre hilfreich. Dieses kann verwendet werden, wenn Elemente in bestimmten Verpackungseinheiten (z.B: 1000) verkauft werden. Dadurch kann verhindert werden, dass sich der Einzelpreis nicht mehr mit 4 Dezimalstellen ausdrücken lässt.

Diskussionspunkt 10 – Neues Element LineItemGrossAmount

- Lt. Diskussion zwischen Post und Bund
  - Zu Berechnungszwecken wäre ein Feld „LineItemGrossAmount“ hilfreich, welches den Gesamtbruttobetrag einer Rechnungszeile enthält

Diskussionspunkt 11 - Duplikat

- Lt. Gesprächen zwischen BRZ, BIG und Hr. Oman
  - Wie kann ich eine ebInterface-Rechnung strukturiert als "Duplikat" kennzeichnen.
    Mein (Hr. Helger) Vorschlag wäre die Einführung eines Attributs "IsDuplicate:boolean" mit dem Standardwert "false" auf Headerebene.
    Wenn dieses Attribut gesetzt ist, dann wäre es hochoffiziell ein Duplikat.

Diskussionspunkt 12 – Payment Method – Vorschlag BRZ

- Derzeit wird die Zahlungsart in ebInterface über das Element "PaymentMethod" abgewickelt. Um zu spezifizieren, um welchen Zahlungstyp es sich handelt (NoPayment, UniversalBankTransaction oder DirectDebit) muss ein relativ komplexes Konstrukt in der folgenden Form in einer XML-Datei angegeben werden:
  <eb:PaymentMethod xsi:type="eb:UniversalBankTransactionType">...

  Meiner Ansicht nach wäre es einfacher (und schöner, und es gäbe besseren Tool-Support), wenn man die Spezifikation im XSD von
  <xs:element ref="PaymentMethod" minOccurs="0"/>

  auf folgendes Konstrukt ändert:

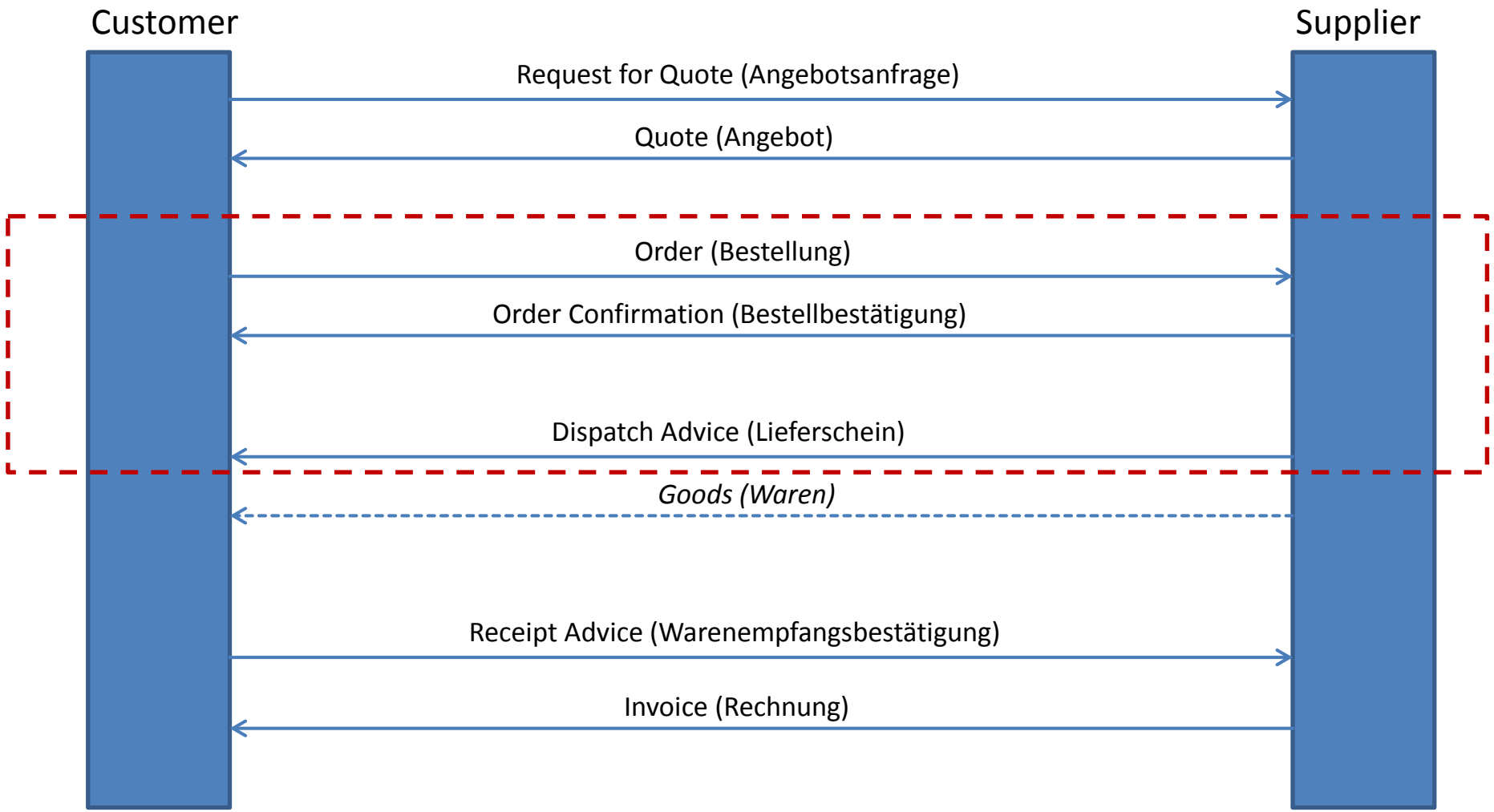
```
<xs:choice minOccurs="0">
  <xs:element ref="NoPayment" />
  <xs:element ref="DirectDebit"/>
  <xs:element ref="UniversalBankTransaction"/>
</xs:choice>
```

Agenda

- Forumsupdate
- Änderungsvorschläge für ebInterface
- Requirements für weitere Dokumenttypen

Weitere Dokumenttypen - Hintergrund

Aktuell vom AK favorisierte Typen:



Konzeptioneller Ansatz zur Definition von weiteren Dokumenttypen

- **Option A:** Definition von eigenen XML-Schemata für jeden einzelnen Dokumenttypen
  - ERP Hersteller müssen mehrere einzelne Schemata unterstützen
  - Änderungen müssen in jedem einzelnen XML Schema umgesetzt werden

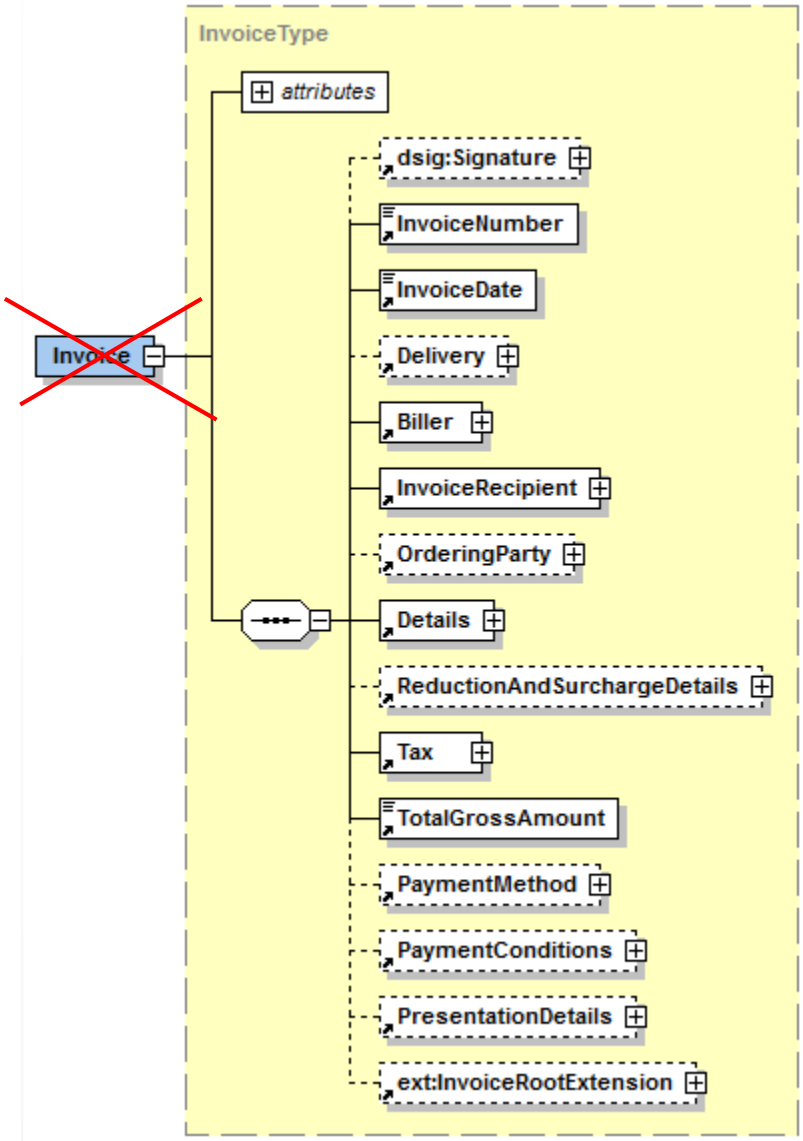
- **Option B:** Definition eines einheitlichen Basis XML-Schemas, welches die Ausgangsbasis für alle weiteren Dokumenttypen ist
  - Einzelne Dokumenttypen werden mit Hilfe von Schematronregeln geprüft
  - ERP Hersteller müssen lediglich ein einzelnes Schema unterstützen
  - Änderungen werden einheitlich in einem einzelnen Schema gepflegt

Requirements für weitere Dokumenttypen

- Vorgehensweise
  - ebInterface Schema dient als Basis
  - ROOT Element auf "Document" umändern
  - DocumentType erweitern (um Order, OrderConfirmation und DispatchAdvice)
  - Diskussion über die genaue Namensgebung von Elementen (Biller, InvoiceRecipient, OrderingParty)
  - Pro Dokumenttyp gilt es zu analysieren:
    - Welche Elemente sollen aus dem Kernstandard mittels Schematronregeln "ausgeschlossen" werden
    - Welche Elemente fehlen für einen bestimmen Dokumenttypen noch im Kernstandard

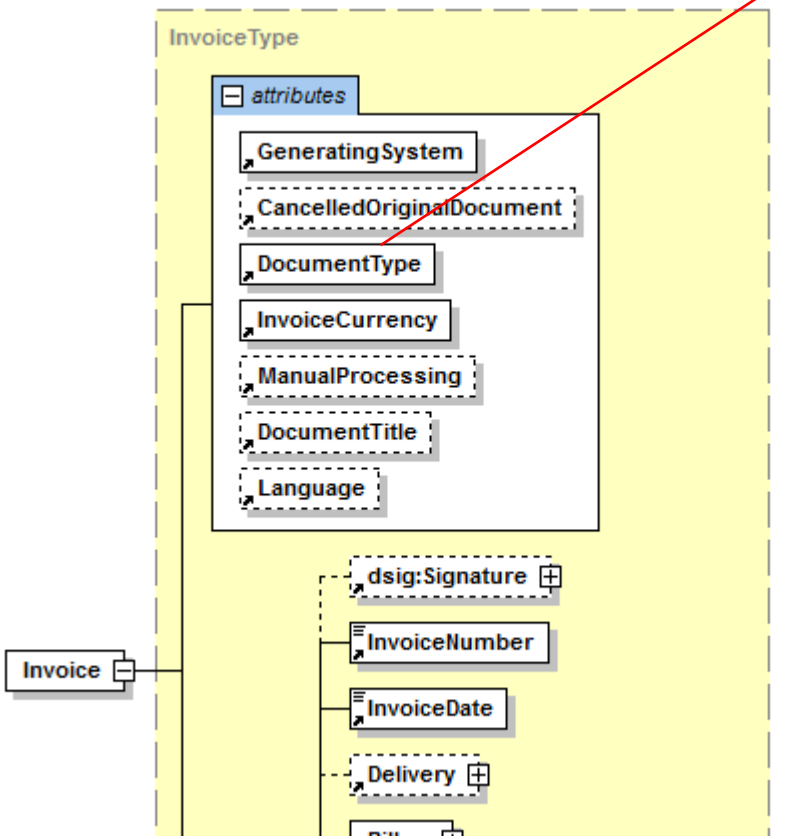
Änderung des ROOT-Elements

Document



DocumentType erweitern

- Aktueller Stand




```
<xs:simpleType name="DocumentTypeType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Invoice"/>
    <xs:enumeration value="InvoiceForAdvancePayment"/>
    <xs:enumeration value="InvoiceForPartialDelivery"/>
    <xs:enumeration value="FinalSettlement"/>
    <xs:enumeration value="SubsequentCredit"/>
    <xs:enumeration value="CreditMemo"/>
    <xs:enumeration value="SubsequentDebit"/>
    <xs:enumeration value="SelfBilling"/>
  </xs:restriction>
</xs:simpleType>
```

DocumentType erweitern

```
<xs:simpleType name="DocumentTypeType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Invoice"/>
    <xs:enumeration value="InvoiceForAdvancePayment"/>
    <xs:enumeration value="InvoiceForPartialDelivery"/>
    <xs:enumeration value="FinalSettlement"/>
    <xs:enumeration value="SubsequentCredit"/>
    <xs:enumeration value="CreditMemo"/>
    <xs:enumeration value="SubsequentDebit"/>
    <xs:enumeration value="SelfBilling"/>
    <xs:enumeration value="Order"/>
    <xs:enumeration value="OrderConfirmation"/>
    <xs:enumeration value="DispatchAdvice"/>
  </xs:restriction>
</xs:simpleType>
```

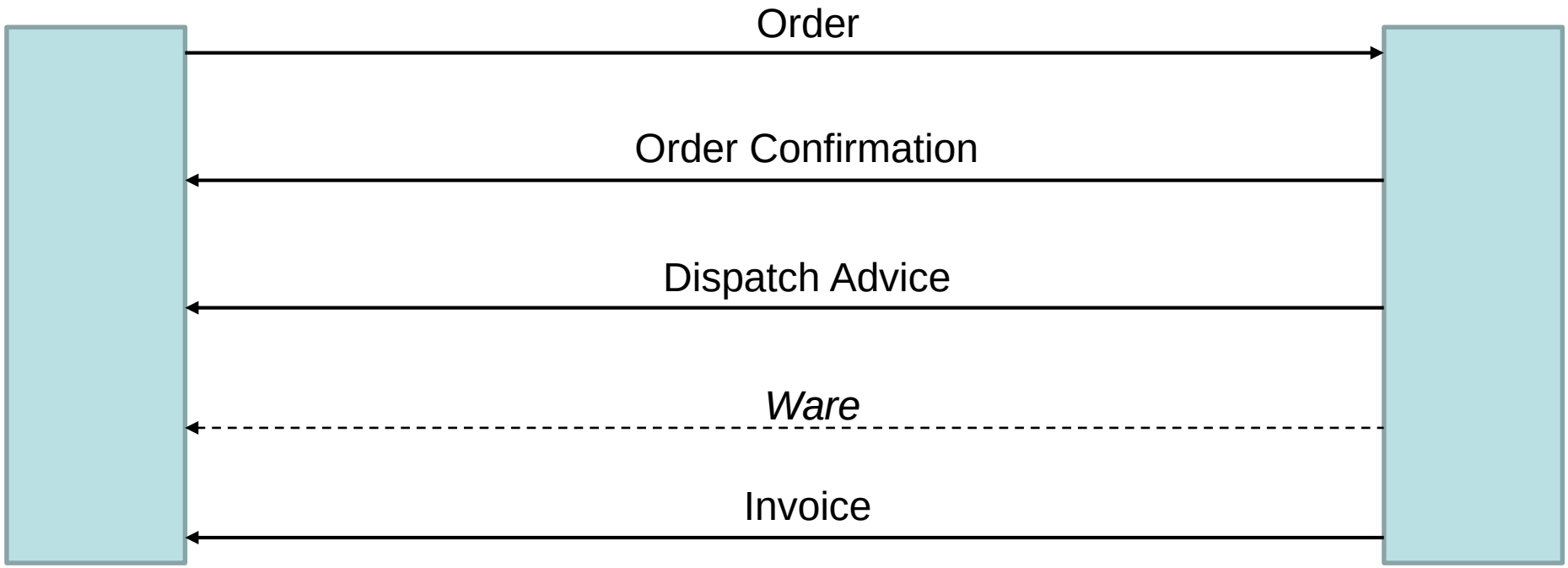
Neu

Namensgebung von Elementen: Bisher

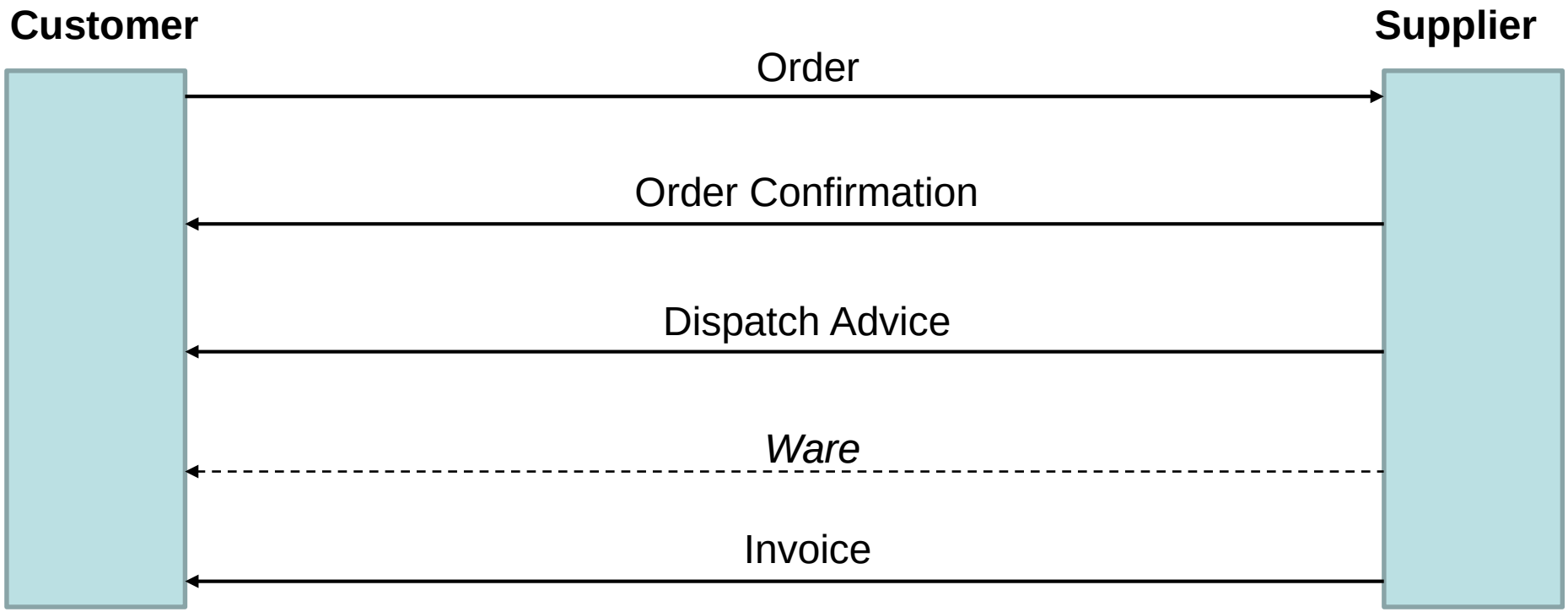
- Delivery
  - Angaben zur Lieferadresse
- Biller
  - Angaben zum Rechnungssteller (= lieferndes oder leistendes Unternehmen)
- InvoiceRecipient
  - Angaben zum Rechnungsempfänger. Entspricht demjenigen, der aus der Rechnung den Vorsteuerabzug geltend machen kann.
- OrderingParty
  - Angaben zum Auftraggeber (nicht notwendigerweise gleich dem Rechnungsempfänger

Neue Namensgebung

Biller, Delivery, InvoiceRecipient, OrderingParty?



Neue Namensgebung (laut ERPEL)



Delivery	= Delivery
Biller	= Supplier
InvoiceRecipient	= Customer
OrderingParty	= OrderingParty

Elemente pro Subtyp

- Diskussion anhand von Vorgaben von ERPEL und dem aktuellen ebInterface 4.0 Schema