

Fachhochschule Wiesbaden - FB Design Informatik Medien

# 7437 - EDI und E-Business Standards

Electronic Data Interchange  
(Elektronischer Datenaustausch)

21.04.2006 H. Wertiges, FB Design > Informatik > Medien, FH Wiesbaden 1

Fachhochschule Wiesbaden - FB Design Informatik Medien

# Klassisches EDI - der Kern

Einleitung - die Kernkomponenten  
File Transfer- und Messaging-Standards  
UN/EDIFACT und EANCOM im Detail  
**Applikationsschnittstellen**  
**Konverter- und Mappingtechniken**

21.04.2006 H. Wertiges, FB Design > Informatik > Medien, FH Wiesbaden 2

Fachhochschule Wiesbaden - FB Design Informatik Medien

# EDI: Die Kernkomponenten

- EDI-Standardaustauschformate
- **Applikationsschnittstellen**
- Mapping
- Routing
- Messaging / File Transfer
- Extras
  - Archivierung
  - Reporting
  - Alarmierung
  - Tracking & Tracing

21.04.2006 H. Wertiges, FB Design > Informatik > Medien, FH Wiesbaden 3

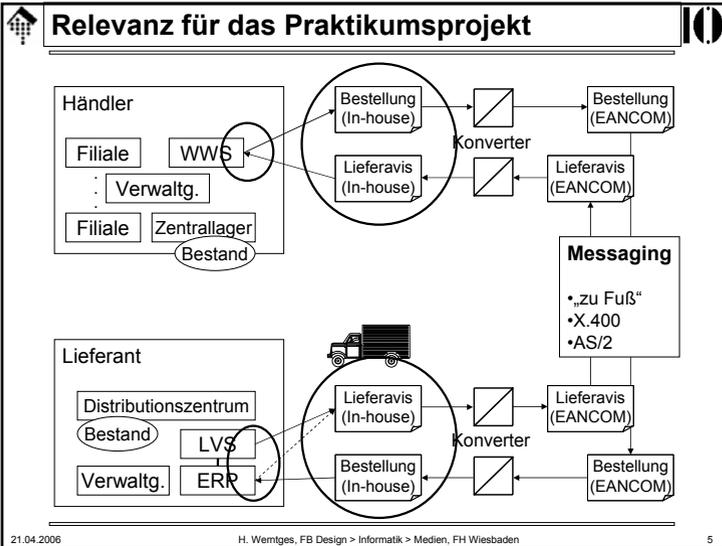
Fachhochschule Wiesbaden - FB Design Informatik Medien

# Einordnung

Applikationsschnittstellen und -formate      Mapping

Unternehmen X      EDI Clearing Server      Daten n      VANS

21.04.2006 H. Wertiges, FB Design > Informatik > Medien, FH Wiesbaden 4



## Fachhochschule Wiesbaden - FB Design Informatik Medien

# Vorbereitung: Struktur typischer Geschäftsdokumente

## ... am Beispiel Bestelldaten

21.04.2006 H. Werntges, FB Design > Informatik > Medien, FH Wiesbaden 6

## Bestelldaten: Welche Datenelemente?

Gemeinsame Entwicklung (Tafel)!

- Gliederung:
  - Kopfdaten
  - Positionsdaten
  - Summendaten
- Kopfdaten (Pflichtfelder)
  - Genaue Belegart (Abruf, Eil-...)
  - Belegnummer (Bestell-Nr.)
  - Belegdatum
  - Käufer / Besteller
  - Lieferant
- Kopfdaten (optional)
  - Gewünschtes, spätestes Lieferdatum
  - Separate Identnummern für Lieferstelle, Rechnungsstelle, Endkunde etc.
- Kopfdaten (optional, Forts.)
  - Postalische Adressen (Straße/Postfach, PLZ, Ort, Land)
  - Lieferbedingungen (frei Haus, versichert bis...)
  - Zahlungsbedingungen (sofort netto, Skonti)
  - Koprabatte
  - Werbekennzeichen
  - Referenzen
    - Angebot
    - Katalog, Periode
    - Frühere Bestellung
  - Freitexte
    - Anweisungen für Anlieferung
    - Hinweise auf Werbeaktionen
  - Weitere, je nach Branche...

21.04.2006 H. Werntges, FB Design > Informatik > Medien, FH Wiesbaden 7

## Bestelldaten: Welche Datenelemente?

- Positionsdaten (Pflichtfelder)
  - Artikelnummer
  - Menge
- Positionsdaten (optional)
  - Positionsnummer
  - Separate Identnummern, z.B.
    - Materialnummer des Lief.
    - Katalognr. des Händlers
  - Mengeneinheit
  - Art.-Beschreibung
    - Kurz-/Langtext, Matchcode
  - Nähere Angaben
    - Farbe, Größe (Kleidung!)
  - Preise
    - Netto, Brutto, Liste, EVP
  - Beträge (analog)
- Positionsdaten (optional, Forts.)
  - Besondere Weisungen
  - Allgemein:
    - Alle Kopfdaten auch auf Positionsebene möglich, wenn für diesen Artikel andere Werte gelten! Beispiele:
      - Separate Versandanschrift
      - Separates Lieferdatum
  - Freitexte
    - Anweisungen für Anlieferung
    - Hinweise auf Werbeaktionen
  - Artikel-Hierarchie
    - Position x ist Unterposition von y, etwa bei Displays & Sortimenten
  - Weitere, je nach Branche...

21.04.2006 H. Werntges, FB Design > Informatik > Medien, FH Wiesbaden 8

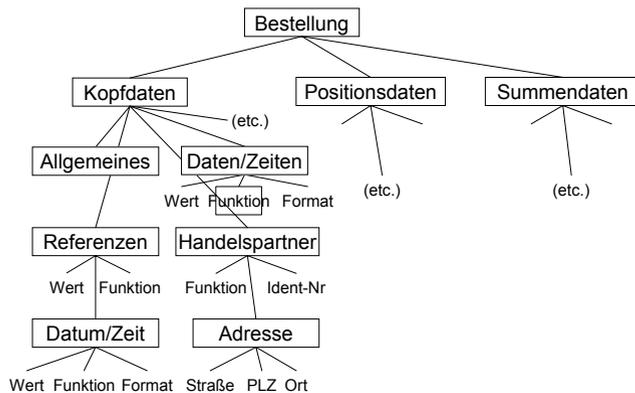
## Bestelldaten: Welche Datenelemente?

- Summendaten (Pflichtfelder)
  - (keine)
- Summendaten (optional)
  - Gesamtzahl Bestellpositionen
  - Gesamtzahl bestellter Artikel
  - Gesamtbetrag der Bestellung
- Weitere, je nach Branche...  
(sehr verbreitet bei Rechnungsdaten!)

## Gruppenbildung

- **Gruppenbildung**
  - Strukturierung der Belegarten durch Sortierung der Felder nach Zusammenhängen.
  - Erste Ebene: Kopf, Positionen, Summe
- **Beispiel Kopfdaten:**
  - Allgemeine Kopfdaten
  - Daten & Zeiten: Datum, Funktion, Format
  - Referenzen: Nummer, Datum (s.o.)
  - Handelspartner
    - Identnummer, Rolle
    - Adresse
      - Straße, PLZ, Ort
  - Zahlungsbedingungen: Art, Frist
  - Lieferbedingungen: Art, Ort, Zusatzangaben
  - etc.

## Gruppenbildung



## Gruppenbildung

- **Konsequenzen für abbildende Datenstrukturen**
  - Komplexität der Daten erfordert geeignete Datenstrukturen!
    - Reale Geschäftsdaten sind recht komplex bzw. werden es rasch.
    - Aber: Bei einfachen Daten genügt auch eine einfache Datenstruktur.
- **Grundregel: Modular vorgehen!**
  - Segmentbildung
    - Zusammenhängende Datenelemente zu Segmenten zusammenfassen
    - Beispiel: Segment = Datensatz, z.B. eine Datenzeile oder ein "Record"
    - Identifizierung der Segmente erforderlich
      - Segmentnamen, Satzart-IDs, Tags ...
  - Gruppenbildung
    - Zusammenhängende Segmente bilden Segmentgruppen

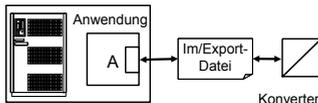
- Hierarchiebildung
  - Segmentgruppen können Segmentgruppen enthalten
  - Bsp: Adressen in Partnerdaten in Kopfdaten
- Möglichkeiten der Serialisierung
  - Abbildung der Segmenthierarchie in den Daten selbst
    - Beispiel: XML-Dokumententypen
    - Dokumente sind formal beschreibbar (DTDs, Schemata) & validierbar
  - Übertragung der reinen Segmentfolge, separate Dokumentation der Hierarchie
    - Beispiel: In-house Formate (s.u.), UN/EDIFACT
    - Dokumenthierarchie ist ohne die separaten Spezifikationen nicht verständlich
    - Dokumente sind höchstens syntaktisch prüfbar
    - Kompakte Datenformate!

## Applikationsschnittstellen, Applikationsformate

Schnittstellenarten  
Die SAP IDoc-Schnittstelle - ein "prominentes" Beispiel

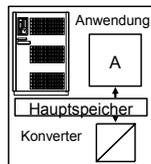
### 1. Dateischnittstellen

- die traditionelle Methode, unser Schwerpunkt



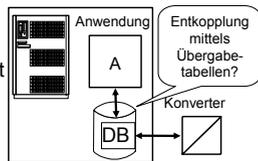
### 2. Austausch von Speicherstrukturen durch IPC-Verfahren

- für zeitkritische Anforderungen
- Meist (nicht immer!) mit Konverter auf gleichem Host wie Anwendg.



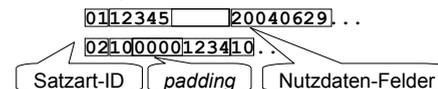
### 3. Direkte Kopplung über Datenbankzugriffe auf Applikationstabellen

- nur in speziellen Umgebungen geeignet
- risikoreich aus Applikationssicht
- nicht gerade modular
- elegant, wo der Einsatz vertretbar ist



### • Fixed record-Strukturen

- Positiv:
  - einfach
  - schnell in der Verarbeitung
  - gut zu parsen
- Negativ:
  - unflexibel bei späteren Anwendungen
  - verschwendet viel Platz (*padding* mit Blanks oder Nullen)
- Beispiele:



- In der Praxis eingesetzt von: SAP IDocs; SEDAS, GENCOD

## Dateischnittstellen: Datenformat-Arten

- Fixed record-Strukturen: Lesetechnik. Ein einfaches Ruby-Beispiel:**

```
class MyRecs
  @@fldLen = {'01'=>[10,8], '02'=>[2,8,2]}
  @@fldType = {'01'=>['an..10','an8'], '02'=>['n2','n..8','n..2']}

  def MyRecs.split( rec )
    rid = rec[0..1]; a = [ rid ]; offset = rid.length
    @@fldLen[rid].each_with_index do |len, i|
      next_offset = offset + len
      field = rec[offset..next_offset]
      case @@fldType[rid][i]
      when /^n.*;/; a << field.to_i
      when /^an\d+;/; a << field
      when /^an\\.\\.\\.d+;/; a << field.sub(/s*$/, '')
      end
      offset = next_offset
    end
    a
  end
end
p MyRecs.split "0112345 20040629" # Tests
P MyRecs.split "02100000123410"
```

21.04.2006 H. Werntges, FB Design > Informatik > Medien, FH Wiesbaden 17

## Dateischnittstellen: Datenformat-Arten

- Variable record-Strukturen**
  - Positiv:
    - Kompakt
    - Flexibel bei Feldlängenänderungen
  - Negativ:
    - Trennzeichen erforderlich (verkomplizieren das Parsen: ESC-Mechanismus erforderlich (ESC = "escape"))
    - Nur seriell verarbeitbar (!)
  - Beispiel:
    - CSV (*comma separated variables*), TSV (*tab separated var.*):
 

```
01;12345;20040629; ...
02;10;1234;10; ...
```

21.04.2006 H. Werntges, FB Design > Informatik > Medien, FH Wiesbaden 18

## Dateischnittstellen: Datenformat-Arten

- Variable length record-Strukturen: Lesetechnik. Ein einfaches Ruby-Beispiel:**

```
# Für CSV muss man praktisch nicht programmieren:

# Mit Ausgabe der Satzart-ID:
p "01;12345;20040629".split(';')
["01", "12345", "20040629"]

# Ohne Ausgabe der Satzart-ID:
p "02;10;1234;10".split(';').slice(1..-1)
["10", "1234", "10"]
```

Vorsicht: Ohne zusätzliche Formatbeschreibung sind alle Felder zunächst vom Typ "String"!

```
# Schreiben - auch ganz einfach
["01", 12345, "20040629"].join(';')
"01;12345;20040629"
```

21.04.2006 H. Werntges, FB Design > Informatik > Medien, FH Wiesbaden 19

## Dateischnittstellen: Datenformat-Arten

- Sonstige: key/value-Listen**
  - Auch "stanzas" bzw. Strukturen wie Windows \*.ini-Dateien
  - Positiv:
    - sehr flexibel, auch bei Aufnahme neuer Felder:
      - weder Feldlänge noch Feldreihenfolge sind fixiert
    - selbst-dokumentierend (über sprechende keys)
  - Negativ:
    - Overhead
    - kein a priori-Wissen über den Ort eines erwarteten Wertes
  - Beispiel:
 

```
recID=header,orderNo=12345,orderDate=20040629, ...
recID=item,matNo=1234,quantity=10, ...
```

21.04.2006 H. Werntges, FB Design > Informatik > Medien, FH Wiesbaden 20

## Dateischnittstellen: Datenformat-Arten

- **Key/value-Listen: Lesetechnik. Ein einfaches Ruby-Beispiel:**

```
class MyRecs
  @@fieldSep, @@kvSep = ',', '=' # i.A. noch "escaping" notwendig

  def MyRecs.to_h( rec ) # Record in Hash umwandeln
    h = Hash.new
    rec.split(@@fieldSep).each do |kv|
      key, value = kv.split(@@kvSep)
      h[key.strip] = value.strip
    end
    h
  end
end

# Tests:
p r01=MyRecs.to_h("recID=header,orderNo=12345,orderDate=20040629")
{"recID"=>"header", "orderDate"=>"20040629", "orderNo"=>"12345"}
p r02=MyRecs.to_h("recID=item, matNo = 1234 , quantity= 10")
{"quantity"=>"10", "recID"=>"item", "matNo"=>"1234"}
```

---

21.04.2006 H. Wertiges, FB Design > Informatik > Medien, FH Wiesbaden 21

## Dateischnittstellen: Datenformat-Arten

- **Key/value-Listen: Codelisten, Umgang mit Teilfeldern Ein einfaches Ruby-Beispiel (Forts.):**

```
Month = { # Hash als Implementierung einer Codeliste
  '01'=>'Jan', '02'=>'Feb', '03'=>'Mar', '04'=>'Apr',
  '05'=>'May', '06'=>'Jun', '07'=>'Jul', '08'=>'Aug',
  '09'=>'Sep', '10'=>'Oct', '11'=>'Nov', '12'=>'Dec'
}

# Extraktion von Teilfeldern mit Regulären Ausdrücken,
# Mapping eines Teilfeld per Codeliste 'Month':
re = Regexp.new('\d{4}) (\d{2}) (\d{2})') # YYYYMMDD erwartet
md = re.match(x01['orderDate']) # Ein MatchData-Objekt...
printf "Oder date is: %s-%s-%s\n", md[3], Month[md[2]], md[1]
```

- **Merke:**
  - Codelisten = Tabellen, die i.a. Codes auf Bedeutungen abbilden
  - Sie lassen sich oft direkt als Hashes implementieren!

---

21.04.2006 H. Wertiges, FB Design > Informatik > Medien, FH Wiesbaden 22

## Dateischnittstellen: Datenformat-Arten

- **Sonstige: Markup, insb. XML-Dokumente**
  - Positiv:
    - ideal für hierarchische *record*-Strukturen
    - selbst-dokumentierend
      - in einfacher Form über sprechende *tags*
      - ggf. auch detailliert, über DTD bzw. Schema
    - validierbar
    - Maßgeschneiderte Datentypen möglich (per XML Schema)
  - Negativ:
    - sehr großer *overhead*
    - hoher Speicherverbrauch - massendatentauglich?
    - komplexes Interface (DOM)
    - Eigene Selektionssprachen (XPath, XQuery)

---

21.04.2006 H. Wertiges, FB Design > Informatik > Medien, FH Wiesbaden 23

## Dateischnittstellen: Datenformat-Arten

- **Beispiel zu XML (fiktiv)**

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<DOCTYPE my_orders ...>
<my_orders>
  <order>
    <header>
      <refno>12345</refno>
      <date fmt="...">20040629</date>
    </header> <!-- ... -->
    <items>
      <item no="10">
        <matNo>1234</matNo>
        <quantity unit="pieces">10</quantity>
      </item> <!-- etc. -->
    </items>
  </order>
</my_orders>
```

---

21.04.2006 H. Wertiges, FB Design > Informatik > Medien, FH Wiesbaden 24

## Dateischnittstellen: Datenformat-Arten

- **XML-Dokumente:** Lesetechnik. Ein einfaches Ruby-Beispiel:

```
require 'rexml/document'

xmldoc = REXML::Document.new(File.open("myorders.xml"))

# Zugriff auf XML-Element z.B. mit XPath:
orderDate = xmldoc.elements["/my_orders/order/header/date[1]"].text

# Ausgabe analog zu key/value-Beispiel:
mtch = Regexp.new('\d{4}) (\d{2}) (\d{2})').match(orderDate)
puts "Oder date is: %s-%s-%s" % [mtch[3], Month[mtch[2]], mtch[1]]

# Noch ein XML-Zugriff per XPath:
path = "/my_orders/order/items/item[@no='10']/matNo"
matNo = xmldoc.elements[path].text
puts "Mat no. of item 10 is \"%s\"" % matNo
```

---

21.04.2006 H. Werntges, FB Design > Informatik > Medien, FH Wiesbaden 25

## Dateischnittstellen: Datenformat-Arten

- Bemerkungen zum Begriff **flat file**
  - Vorsicht - keine einheitliche Verwendung!
  - Zwei recht konträre, aber gebräuchliche Bedeutungen:
    - a) serialisierte Speicherstruktur eines komplexen Typs (etwa: "flachgeklappte Hierarchie")
    - b) 1:1-Abbildung des EDIFACT *interchange* als *fixed record*-Format, 1 *record* pro Segment

---

21.04.2006 H. Werntges, FB Design > Informatik > Medien, FH Wiesbaden 26

Fachhochschule Wiesbaden - FB Design Informatik Medien

### Die SAP EDI-Schnittstelle- ein "prominentes" Beispiel

Das IDoc-Konzept  
IDoc-Struktur  
IDoc-Verwaltung  
Besonderheiten, Ausblick

---

21.04.2006 H. Werntges, FB Design > Informatik > Medien, FH Wiesbaden 27

## Das IDoc-Konzept

- IDoc = *intermediate document* = Zwischenbeleg
- Eine Struktur...
  - zwischen Anwendungsbeleg einerseits
  - und klassischem, sehr dem EDI-Format nachempfundenen "flat file"
- Kernidee:
  - Abstraktionsschicht oberhalb konkreter EDI-Standards wie UN/EDIFACT oder ANSI X12
- Einheitliche Kommunikationsform zwischen
  - verschiedenen SAP-Systemen bzw.
  - SAP und sog. "Sub-Systemen", insb. EDI-Servern

---

21.04.2006 H. Werntges, FB Design > Informatik > Medien, FH Wiesbaden 28

## IDoc-Struktur

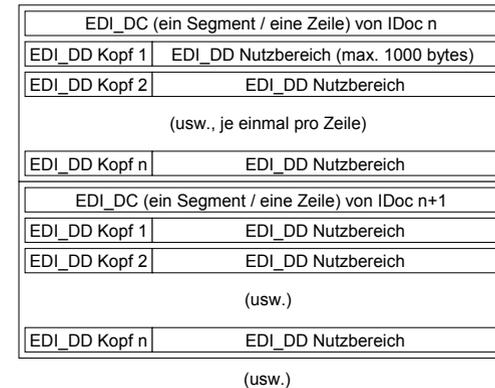
- **EDI\_DC**
  - Allgemeiner Verwaltungskopfsatz
  - In etwa analog zu einer Kombination aus UNH und UNB
  - Geht jedem IDoc genau einmal voraus
- **EDI\_DD**
  - Verwaltungsrahmen für Nutzsegmente
  - Nutzsegmente (je max. 1000 Bytes):
    - mit Versionsnummer
    - fester Satzaufbau
    - Hierarchie
    - Erweiterbarkeit
- **EDI\_DS**
  - Statussatz, für Rückmeldungen ausgehender Belege

21.04.2006

H. Werntges, FB Design > Informatik > Medien, FH Wiesbaden

29

## Struktur einer IDoc-Datei, schematisch



21.04.2006

H. Werntges, FB Design > Informatik > Medien, FH Wiesbaden

30

## IDoc-Struktur

- XML-IDocs
  - Gleicher Inhalt, neue Verpackung
  - Eher noch größeres Datenvolumen als bei den bereits recht großen IDocs mit *fixed record*-Struktur
- **IDoc-Dokumentation** - *online*-Beispiele
  - HTML-Dokumentation, wie man sie aus SAP R/3 exportieren kann
  - a) EDI\_DC, EDI\_DD, EDI\_DS
  - b) Nutzdaten-Beispiel: ZINVOIC1

21.04.2006

H. Werntges, FB Design > Informatik > Medien, FH Wiesbaden

31

## IDoc-Verwaltung

- Übergabetabellen (DB)
- Eigene Statusverwaltung der IDocs
  - intern wie extern
  - mit eigenem *user interface*
- Dokumentation in IDoc-Verwaltung integriert
  - IDoc-Formatbeschreibungen sind jederzeit exportierbar
    - in strukturierter Form, vergleichbar XML-DTD/Schema
    - als HTML-Dokumentation
    - als „C“-Headerdateien (Makros, *struct's*, ...)
- Technische Schnittstellen
  - EDI (i.w. Dateischnittstelle plus *RFC* (SAP's RPC-Methode)
  - ALE („*application link enabling*“) - *high-level RFC*

21.04.2006

H. Werntges, FB Design > Informatik > Medien, FH Wiesbaden

32

## Die technische EDI-Schnittstelle

- EDI Ports - oder: SAP erreicht das Betriebssystem
  - In SAP werden (logische) EDI-Ports angelegt
  - Auf Betriebssystemseite werden ihnen Unterverzeichnisse zugewiesen
- Typische derartige Verzeichnisstruktur:
  - Basis-Verzeichnis
    - z.B. auf einem Unix-Host: `/usr/sap/edi/<SAPSYS>/<PORTNAME>`
    - Konkret etwa (fiktiv): `/usr/sap/edi/PW1/DCSM`

- Unterverzeichnisse:
 

<code>./in</code>	für eintreffende IDoc-Dateien
<code>./out</code>	für ausgehende IDoc-Dateien
<code>./status</code>	für eingehende Dateien mit Statusätzen

21.04.2006 H. Werntges, FB Design > Informatik > Medien, FH Wiesbaden 33

## Die technische EDI-Schnittstelle

- Signalisierung bei ausgehenden Daten:
  - Drei Betriebsarten:
    - Polling – keine Signalisierung
      - SAP erzeugt IDoc-Dateien in `./out`
      - EDI-Subsystem holt sie zeitgesteuert ab
      - Einfach, aber mit Kollisionsgefahr. Für Batchbetrieb ausreichend
    - Triggern pro Exportdatei
      - SAP erzeugt IDoc-Dateien in `./out`
      - SAP startet Folgeverarbeitung für jede Datei mittels `rfcexec`-Aufruf
      - Effizient, guter Kompromiss, Regelfall. Komplexere Systemkopplung
    - Triggern für jedes IDoc
      - SAP erzeugt für jedes IDoc eine eigene Datei in `./out`
      - SAP startet Folgeverarbeitung für jedes IDoc mittels `rfcexec`-Aufruf
      - Hohe Systemlast, geeignet für sehr zeitkritische Abläufe, ALE-artig.
  - **rfcexec**: Nahtstelle zwischen SAP-Programmen und dem BS
    - `rfcexec` ist ein von SAP bereitgestelltes, auf dem jeweiligen BS lauffähiges Binärprogramm, das aus SAP heraus per RFC gestartet wird.
    - Seine Aufgabe: Ausführung „beliebiger“ externer Programme / Scripte

21.04.2006 H. Werntges, FB Design > Informatik > Medien, FH Wiesbaden 34

## Die technische EDI-Schnittstelle

- Signalisierung bei eingehenden Daten:
  - Zwei Betriebsarten:
    - Polling – keine Signalisierung
      - EDI-Subsystem erzeugt IDoc-Dateien in `./in` bzw. Statussatz-Dat. in `./status`
      - SAP holt sie zeitgesteuert ab
      - Einfach, aber mit Kollisionsgefahr. Für Batchbetrieb ausreichend
    - Triggern pro Eingangsdatei
      - EDI-Subsystem erzeugt IDoc-Dateien in `./in` bzw. Statussatz-Dat. in `./status`
      - ... und startet Folgeverarbeitung in SAP mittels `startRFC`-Aufruf
      - Der Regelfall. Konfiguration der zahlreichen `startRFC`-Optionen nicht immer einfach.
  - **startRFC**: Nahtstelle zwischen BS und SAP-Programmen
    - `startRFC` ist ein von SAP bereitgestelltes, auf dem jeweiligen BS lauffähiges Binärprogramm.
    - Es wird von externen Programmen (typisch sind Scripte) aufgerufen, um in SAP per RFC bestimmte Funktionsmodule aufzurufen, etwa die zum IDoc-Import und deren Verarbeitung.

21.04.2006 H. Werntges, FB Design > Informatik > Medien, FH Wiesbaden 35

## Die technische EDI-Schnittstelle: Eingehende Daten

Kopplung, z.B. per NFS

21.04.2006 H. Werntges, FB Design > Informatik > Medien, FH Wiesbaden 36

## Die technische EDI-Schnittstelle

### Fehlerbehandlung

#### Ausgehende Daten:

- Verfolgung von Fehlern auf IDoc-Ebene durch Abfolge von Statuswerten
  - Interne Statuswechsel protokolliert SAP automatisch
  - Externe Statuswechsel muss das EDI-Subsystem per Statussatz mitteilen
  - Beispiele: Subsystem getriggert, IDoc übersetzt / übermittelt, *func. ackn.*
- WICHTIG: SAP-interne Überwachung der Ende- und Fehler-Zustände**

#### Eingehende Daten

- Grundregel:**  
Datei von SAP gelöscht = Verantwortung an SAP übergeben
- Rückgabewerte von `startRFC` nicht erhältlich bzw. nicht aussagekräftig!

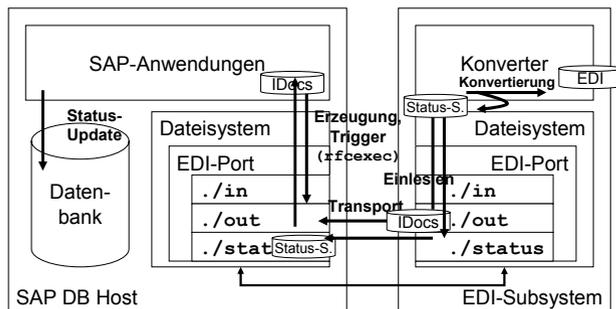
## Die technische EDI-Schnittstelle

### Problemfall bei eingehenden Daten:

- Datei enthält mehrere IDocs, Übernahme scheitert
- Problem:
  - Welche IDocs wurden bereits von SAP übernommen (und dürfen daher nicht erneut eingespielt werden!)?
  - Welche müssen repariert bzw. ausgesondert werden?
  - Mit welchen IDocs darf (bzw. muss schleunigst) ein erneuter Versuch starten?
- Lösung 1 (manuell und mühsam!)
  - Manuell in SAP nachsehen, für welche IDocs die Übernahme gelang
  - Diese (die ersten  $n-1$ ) aus der IDoc-Datei entfernen
  - IDoc  $n$  ist vermutlich defekt – ebenfalls aussondern, Fehler analysieren
  - Erneuter Übernahmeversuch mit dem Rest ( $n+1 \dots N$ )
- Lösung 2 (in der Praxis bewährt)
  - Eingehende IDoc-Dateien splitten:
    - Ein IDoc pro Datei
    - SAP für jedes IDoc separat triggern
    - Aussondern der Fehlerfälle + Alarmierung ist dann einfach
  - Nachteil: Höhere Systemlast seitens SAP
    - Beherrschbar, solange nur IDoc-Übernahme und nicht auch Verbuchen getriggert wird.



## Die technische EDI-Schnittstelle: Ausgehende Daten

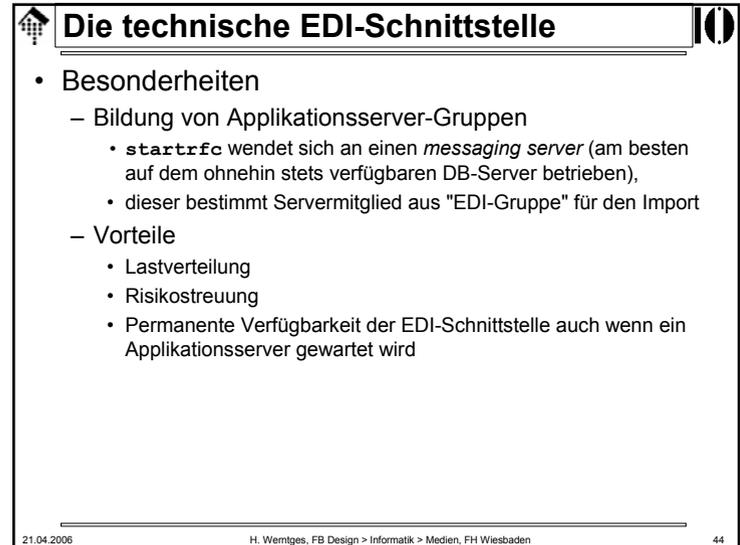
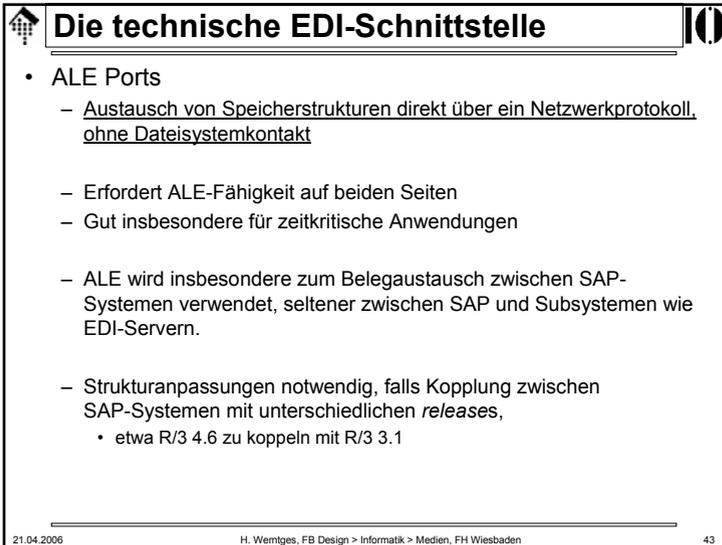
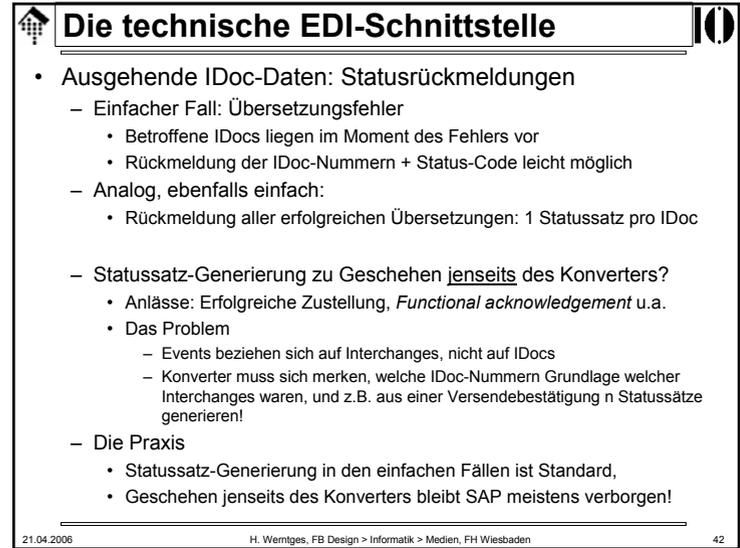
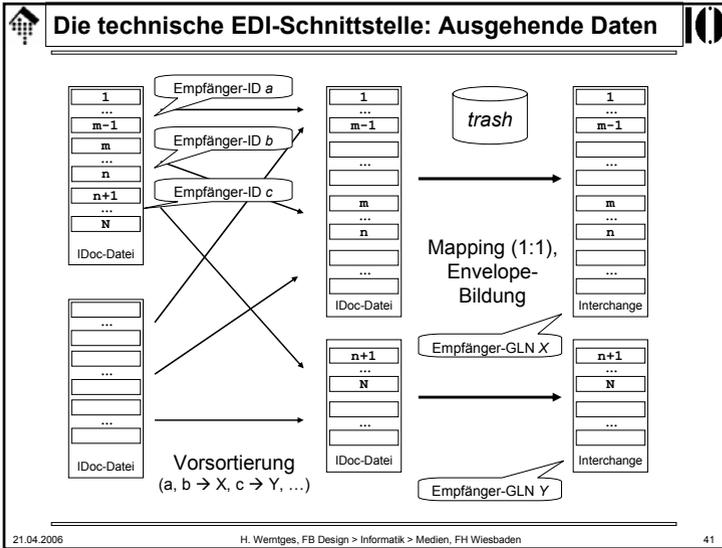


Kopplung, z.B. per NFS

## Die technische EDI-Schnittstelle

### Ausgehende IDoc-Daten: Vorverarbeitung

- IDoc-Dateien sind nach SAP-Kriterien gebündelt, z.B.
  - Alle Rechnungen eines *batch*-Laufs in einer Datei
  - „Nachzügler“ in einer anderen Datei
  - Ausgehende Lieferavise in dritter Datei
- Zur Konvertierung sollten alle Belege zusammengefasst werden, die in einen *interchange* gehören
  - Sortierung nach Empfänger-ID (in EDI\_DC), evtl. auch noch nach Sender-ID (auch in EDI\_DC) – etwa falls mehrere Geschäftseinheiten existieren und auch nach außen zu unterscheiden sind.
- Bündelung / ID *mapping*
  - Interne Empfänger-IDs sind umzucodieren in extern verwendbare IDs auf *envelope/interchange*-Ebene, insbesondere in GLNs!
  - Dabei möglich:  $n : 1$ -*mapping* (mehrere interne IDs auf eine externe ID)
  - NICHT möglich:  $1 : n$ -*mapping* !
- Bündelung auf *messaging*-Ebene
  - Mehrere unterschiedliche Empfänger teilen sich eine Empfangsadresse





- Ausblick / Verwandte Themen
  - BAPIs (Business APIs)
    - genormte *high-level* Schnittstellen z.B. zur Erzeugung eines kompletten Geschäftsdokuments durch ein externes System
  - Business Integration Server (MOM, XML, *Web services*)
  - SAP Business Connector
    - Kostenlose Zusatzkomponente, Einstiegshilfe in XML-Tech / WS
    - Reduzierte Version von einem Produkt von WebMethods
  - SAP NetWeaver
    - SAP's neue eigene Integrations-Technologie
    - SOAP-basierte Web Services bilden eine wesentliche Grundlage