



# 7437 - EDI und E-Business Standards

Praktikumsaufgabe 01:  
Programmierung kleiner  
„Organisationshilfen“ rund um die  
EAN/GTIN



## Die Aufgabe im größeren Kontext



- Kontext
  - Als Handelspartner (Lieferant, Händler) müssen Sie GLNs, GTINs und NVEs vergeben und/oder überprüfen.
  - Bei der Erzeugung von EDI-Daten werden an verschiedenen Stellen Seriennummern benötigt, die jeweils nur einmal vergeben werden dürfen.
- Zweck der Übung
  - Schaffung einfacher Werkzeuge mit Kommandozeilen-Interface zur Lösung der o.g. Aufgaben
  - Grundlage für spätere Praktikumsaufgaben
  - Einarbeitung in die Implementierungssprache des Projekts
    - **Empfehlung: Ruby**
    - Online-Lehrbuch und –Referenz: Das „Pickaxe“-Buch
      - Siehe Linksammlung zu meiner LV „Ruby“
    - Bemerkung: Die folgenden Beispiele beruhen jeweils auf Ruby.



## ean\_util: GTIN, GLN, NVE prüfen und vergeben



- Schreiben Sie ein Kommandozeilen-Programm „**ean\_util**“ mit zwei Betriebsarten:
  - Prüfsummenziffer nachrechnen
    - `% ean_util.rb string`
      - RC = 0 falls Prüfsumme ok, -1 falls nicht
      - Meldung "Pruefsummenfehler" (RC = 1) oder "Argument unzulessig" (RC = -1) nach stderr im Fehlerfall, etwa wenn *string* nicht nur Ziffern enthält
  - Prüfsummenziffer ermitteln, ggf. links mit Nullen auffüllen
    - `% ean_util.rb -l n [ string ]`
      - RC = 0 falls ok, -1 sonst (z.B. wenn *string* zu lang oder unzulässig)
      - Auf stdout: *string* um Prüfziffer ergänzt und ggf. links auf *n* Stellen mit Nullen aufgefüllt. Typische Werte für *n*: 13 (GTIN), 18 (NVE)
      - Pipe mode*: Eingabe von stdin lesen, falls nicht als Argument übergeben



## numgen: Einfacher Nummerngenerator



- Nummerngenerator:
  - Um NVE o.ä. lückenlos und eindeutig vergeben zu können, benötigen wir einen Nummerngenerator. Entwickeln Sie „numgen“ wie folgt:
    - `% numgen.rb [ nrkreis ]`
      - RC = 0 falls ok, -1 falls *nrkreis* nicht existiert
      - Rückgabe-String als Zeile auf stdout (natürliche Zahl)
  - Persistenz
    - Führen Sie eine Datei `~/numgen` ein
    - Darin befinden sich *key/value*-Paare (je eins pro Zeile) der Art `nrkreis = naechster_wert`
    - Jeder Aufruf von "numgen" liefert den zu *nrkreis* hinterlegten Wert und inkrementiert diesen in der Datei.
    - Alternativ: Nutzen Sie die eingebauten Persistenz-Techniken Ihrer Projektsprache!
  - *Locking*
    - Damit "numgen" parallel von mehreren Anwendungen benutzt werden kann, sollte ein *file locking* auf `~/numgen` zum Einsatz kommen!
  - Initialisierung
    - `~/numgen` enthalte anfangs die Zeile `"nve = 1"` (*nve* = Default für *nrkreis*)



## nvegen: NVE-Generator



- NVE-Generator:
  - Schreiben Sie ein Programm "nvegen", das pro Aufruf eine neue NVE liefert:

```
% nvegen.rb GLN_BASE
```

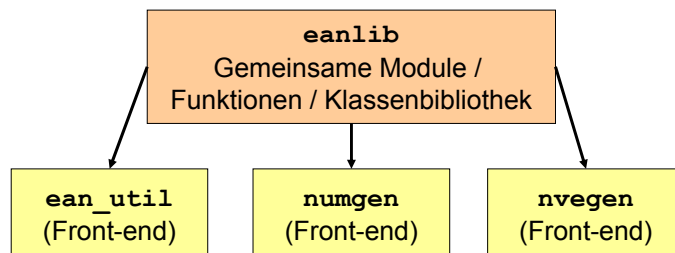
    - RC = 0 falls ok, -1 bei internen Fehlern (z.B. numgen nicht vorhanden oder length(GLN\_BASE) != 7,8 oder 9)
    - Rückgabe-String als Zeile auf stdout (NVE)
  - Vorgehen:
    - Erstellen & testen Sie zunächst „numgen“ und „ean\_util“ !
    - „nvegen“ lässt sich leicht durch Kombination dieser beiden Werkzeuge wie folgt erzeugen:
      - Rufen Sie einfach „numgen“ für die nächste Seriennummer auf
      - Konkatenieren Sie diese Nummer, die benötigten Anteile der übergebenen GLN sowie das NVE-Präfix in der richtigen Reihenfolge
      - Rufen Sie damit „ean\_util -l 18“ auf, um die noch fehlende Prüfziffer zu ergänzen.



## Anleitung & Tipps



- Festlegung der Implementierungssprache
  - Skriptsprachen wie Perl, Ruby, Python sind für diese Aufgaben ideal.
  - Empfehlung: **Ruby** wählen
    - Später sehr hilfreiche EDI-Bibliothek gibt es nur für Ruby
    - Modern, konsequent objekt-orientiert, in Fernost weit verbreitet
- Modulares Vorgehen empfohlen:





## Tests



- Unit Tests
  - In Ruby inzwischen üblich
  - Unterstützt z.B. durch Modul „test/unit“, Klasse Test::Unit::TestCase
- Zu Ihrer Verfügung
  - In Verzeichnis ~werntges/lv/edi/01 finden Sie Datei `ean_tests.rb`
  - Dies ist eine „Testsuite“ mit Aufrufen der zu schreibenden Utilities und den erwarteten Ergebnissen.
- Empfehlung
  - Nutzen Sie die Test-Suite - auch wenn Sie in einer anderen Sprache als Ruby entwickeln sollten; ggf. einfach Aufrufe anpassen.
  - Ihre Programme sind bereit für die Abgabe, wenn `ean_tests.rb` ohne Fehler durchläuft.
  - Fall Ruby: Tests für Klassen in `eanlib` am besten gleich integrieren.



## Abgabe



- Abgabefrist
  - Abgabeschluss ist verlegt auf **Freitag, den 21.4.06 (zus. mit P02)**
- Was abgeben?
  - `ean_util*`, `numgen*`, `nvegen*`, `eanlib*` (z.B.: `ean_util.c`, `numgen.rb`)
- Bedingungen
  - Teambildung bereits empfohlen, jedoch:
  - Durchführung (noch) pro Person – nicht: pro Gruppe!
  - Abgabe daher auch: pro Person
- Abgabe = Kopieren von Dateien in „Ihr“ Abgabeverzeichnis
  - `/local10/werntges/lv/edi/abgaben/a/xxxxxx`  
(xxxxxx = Ihre Matrikelnummer)
  - Ändern Sie die Berechtigungen im Abgabeverzeichnis:
    - `chmod 640 <file>` (für normale Dateien)
    - `chmod 750 <file>` (für ausführbare Dateien)