http://intevation.net

## edbsilon

# Dokumentation

**Version 0.9.1**

Intevation GmbH
Georgstrasse 4
49074 Osnabrück
Germany

Datum: 29.01.2007

Dieses Dokument wurde erzeugt mit LaTeX.

# Inhaltsverzeichnis

# 1 Das Programm EDBSilon

## 1.1 Einleitung

edbsilon ist ein Konverter für Daten im EDBS- oder BZSN-Format. Die EDBS-Dateien werden eingelesen und nach Folien getrennt in andere Formate transformiert.
Es stehen zwei Modi zur Verfügung:

1. SQL-Modus (siehe Abschnitt 3.1)
2. OGR-Modus (siehe Abschnitt 3.2)

Standardmäig werden SQL-Kommandos zum Eintrag in eine Oracle Spatial Geodatenbank ausgegeben.

## 1.2 Warum wurde edbsilon entwickelt?

Es wurde ein EDBS-/BSZN-Konverter benötigt, der die folgenden Eigenschaften besitzen sollte:

- Freie Software
- lauffähig unter GNU/Linux und Windows
- leicht auf andere Bundesländer anpassbar
- Konvertierung von BZSN
- auch im Batch-Mode benutzbar

Dies ist in edbsilon vereint.

# 2 Die Konfigurationsdatei

## 2.1 Allgemeine Informationen

Die Konfigurationsdateien *konf/land.konf* ist dazu gedacht, dass Programm variable zu machen für die ALK-Daten der verschiedenen Länder. Es besteht aus Zeilen der Form:
`foliennr,BesondereInfo,Darstellungsart|Tabellenname,Tabellenflags|Attributliste`

Eine beispielhafte Konfigurationsdatei ist der Software im Ordner *konf* beigelegt.
Grundsätzlich unterscheiden sich 2 Fälle in jeder Konfigurationszeile:

1. Wenn der *Tabellenname* leer ist, wird der Textteil der Besonderen Informationen der Art *foliennr, BesondereInfo, Darstellungsart* je nach Attributliste in die Folientabelle geschrieben.

2. Wenn die *Tabelle* einen Wert beinhaltet, wird der Textteil der Besonderen Information je nach Attributliste in die Tabelle *Tabelle* geschrieben.

Es gibt Zeilen, in denen nur Tabellenflags und foliennr besetzt sind. Diese Zeile geben an, welche Tabellen für die eigentlichen Objekte (also nicht Teilobjekte) einer Folie angelegt werden sollen. Damit ist es z.B. möglich die Gebäude aus den Foliennummern 011 und 082 in einer gemeinsamen Tabelle *gebaeude* zu speichern.

**Beispiel**

```
#
# Folie 001 - Flurstücke
#
001,,|f001,f|
```

Zeilen, die mit # beginnen werden als Kommentar betrachtet.

## 2.2 Genauere Beschreibung

Die Einstellungsmöglichkeiten in der Konfigurationsdatei im Detail:

**foliennr** Diese Variable beinhaltet die Foliennummer für die diese Zeile die Vorgehensweise angibt. Foliennummer, Art der Besonderen Info und Darstellungsart geben eindeutig den Fall an, um den es sich handelt.

**BesondereInfo** Diese Variable beinhaltet die Art der Besonderen Info. Sie dient zur Fallbestimmung.

**Darstellungsart** Diese Variable beinhaltet die Darstellungsart der Besonderen Information. Sie dient zur Fallbestimmung.

**Tabelle** Diese Variable ist leer, wenn die Daten in die Folientabelle geschrieben werden sollen. Sonst beinhaltet sie den Namen der Tabelle in der je nach Attributliste die Besonderen Informationen abgespeichert werden sollen.

**Tabellenflags** In diesen Feld wird festgehalten, welche Tabellen erzeugt werden sollen:
*n* für ohne Anhang, *p* für Punkte (_p), *l* für Linien (_l), *f* für Fläche (_f) und *r* für Rahmen (_r).

**Attributliste** Diese Variable hat die Form
`Attribut,Attributslaenge;Attribut,Attributslaenge;...`
Dabei gibt *Attribut* den Spaltennamen an, in der die Zeichen der Anzahl *Attributslaenge* gespeichtert werden sollen.

### Beispiel

```
001,14,1230|folie001_namenp,nlp|text,1:33
```

Folgende Zeile in der Konfigurationsdatei besagt, dass alle Informationen der Folie 001 mit der besonderen Info 14 und der Darstellungsart 1230 in der Tabelle *folie001_namenp* in der Spalte *text* eingetragen werden sollen. Das ganze Textfeld hat nur 33 Zeichen. Also wird alles in text gespeichert.

edbsilon enthält eine Beispielkonfiguration für die ALK-Daten des Landes Brandenburg. Diese Tabellenstruktur baut auf den „Richtlinien für die Einrichtung der Automatisierten Liegenschaftskarte in Brandenburg“[1]

[1] Im Internet sind die Richtlinien unter `http://www.vermessung.brandenburg.de/sixcms/media.php/1071/73000501.pdf` zu finden. Stand 1. September 2004

# 3 Benutzung von edbsilon

Edbsilon kann auf zwei unterschiedliche Arten genutzt werden. Im Normalfall werden standardmäßig SQL-Dateien zum Import in eine Oracle-Datenbank erstellt.

Um ESRI-kompatible Shapedateien (oder andere durch OGR unterstützte Formate) zu erstellen, kann die OGR-Bibliothek über die Python-Schnittstelle eingebunden werden. Dadurch erweitert sich die Exportfunktionalität auf alle durch OGR unterstützten Formate.

## 3.1 Erstellung von SQL-Dateien

Bevor man die INSERT-SQL-Kommandos für die nach Folien separierten SQL-Dateien von edbsilon benutzen kann, muss man die richtigen Tabellen in der Oracle-Datenbank angelegt haben.
Dazu wird die Datei *createalk.sql* erstellt, die Informationen über die Tabellenstruktur, Indizes und Metainformationen enthält und diese anlegt. Diese ist vor dem eigentlichen Import der Folien-SQL-Skripte auszuführen.

Ferner wird die Datei *drop.sql* erstellt, die ein rückstandsfreies Löschen der Tabellen, Indizes und Metainformationen ermöglicht.

## 3.2 Erstellung von OGR-unterstützten Formaten

Neben den SQL-Dateien kann edbsilon als Ausgabeformat auch andere Formate schreiben. Dazu wird über die Python-Schnittstelle der OGR-Bibliothek auf die unterstützten Formate zugegriffen.
Eine Liste der von OGR unterstützen Vektorformate sind auf der Webseite `http://www.gdal.org/ogr/ogr_formats.html` aufgelistet.
Der folgende Befehl gibt Auskunft darüber, welche Formate von der installierten OGR-Version unterstützt werden:

```
ogrinfo --formats
Loaded OGR Format Drivers:
  -> "ESRI Shapefile" (read/write)
  -> "UK .NTF" (readonly)
  -> "SDTS" (readonly)
  -> "TIGER" (read/write)
  -> "S57" (read/write)
```

```
  -> "MapInfo File" (read/write)
  -> "DGN" (read/write)
  -> "VRT" (readonly)
  -> "AVCBin" (readonly)
  -> "REC" (readonly)
  -> "Memory" (read/write)
  -> "CSV" (read/write)
  -> "GML" (read/write)
  -> "ODBC" (read/write)
  -> "PostgreSQL" (read/write)
```

## 3.3 Aufruf des Konverters

Das Programm edbsilon.py ist der Hauptteil des EDBS-Konverters. Es wird aufgerufen mit folgender Zeile:

```
python edbsilon.py [-h] [-o Ziel] [-l Logbuch ] [-f Format] \
[-a EDBS-Format] [-k Konfdatei ] Quellen...
```

Die Angabe der fünf in eckigen Klammern genannten Parameter ist optional. Werden sie nicht angegeben, so werden default-Werte für die Parameter eingesetzt und SQL-Dateien für Oracle erstellt.

Die Flagge -h gibt folgenden Hilfetext aus:

```
edbsilon -- Konvertiert EDBS-Daten
Benutzung: python edbsilon.py [Optionen] Quelldatein...
Optionen: -o <Ziel>         -- SQL-Kommandos [Default: edbsilon.sql]
          -k <Konfdatei>    -- Konfigurationsdatei [Default: alk.konf]
          -l <Logbuch>      -- Logbuch [Default: edbsilon.log]
          -f <Format>       -- Format [Default: ORACLE]
          -a [alk|atkis]    -- Setzt EDBS-Format auf ALK
                               bzw.  ATKIS [Default: alk]
          -h                -- für diese Hilfe
```

### 3.3.1 SQL-Modus (für Oracle)

Ruft man edbsilon mit den Standardwerten auf, geht es die EDBS-Quellen der Reihenfolge nach durch und erzeugt in der Standardkonfiguration INSERT-SQL Kommandos für die einzelnen ALK-Objekte in der Quelle und die einzelnen Besonderen Informationen der ALK-Objekte. Dann werden SQL-Kommandos erzeugt, die die Spatial Indizes neu aufbauen.

Die SQL-Kommados werden in die Datei *fFolienummer.sql* eingetragen. Existiert bereits eine Datei für diese Folie, wird die Ausgabe an diese Datei angehängt. Wird dies nicht gewünscht, sollte die Datei vor dem Aufruf von edbsilon gelöscht werden.

Die *Konfigurationsdatei* muss die wie in Kapitel 2 beschriebene Form haben.

Ein Aufruf zur Erstellung von SQL-Dateien sieht wie folgt aus:

```
# Ordner für Ausgabe erstellen
mkdir ausgabe

# edbsilon starten
python Konverter/edbsilon.py -o ausgabe/ -k <konfiguration.konf> \
-l ausgabe/edbsilon_sql-log.txt -a alk <Edbsdatei.edb>
```

### 3.3.2 OGR-Modus (für OGR-unterstützte Formate)

Für die Erstellung von Shapefiles wird der OGR-Modus von edbsilon genutzt. Dazu ist es unbedingt nötig, dass die von OGR bereitgestellten Python-Bindungen installiert sind.
Ist dies der Fall, können sämtlich von OGR unterstützten Formate exportiert werden. Im folgenden Beispiel wird die Syntax zur Generierung von ESRI Shapefiles exemplarisch dargestellt:

```
# Ordner für Ausgabe erstellen
mkdir ausgabe

# edbsilon starten
python Konverter/edbsilon.py -o ausgabe/ -k <konfiguration.konf> \
-l ausgabe/edbsilon_log.txt -f "ESRI Shapefile" \
-a alk <Edbsdatei.edb>
```

Die resultierenden Shapedateien sind im Ordber *ausgabe* enthalten, der vor dem Start des Programms erstellt wurde.